

Graph Model Based Indoor Tracking

Christian S. Jensen[†] Hua Lu[†] Bin Yang^{†‡}

[†] Department of Computer Science, Aalborg University, Denmark

[‡] School of Computer Science, Fudan University, China

{csj, luhua, yang}@cs.aau.dk

Abstract

The tracking of the locations of moving objects in large indoor spaces is important, as it enables a range of applications related to, e.g., security and indoor navigation and guidance. This paper presents a graph model based approach to indoor tracking that offers a uniform data management infrastructure for different symbolic positioning technologies, e.g., Bluetooth and RFID. More specifically, the paper proposes a model of indoor space that comprises a base graph and mappings that represent the topology of indoor space at different levels. The resulting model can be used for one or several indoor positioning technologies. Focusing on RFID-based positioning, an RFID specific reader deployment graph model is built from the base graph model. This model is then used in several algorithms for constructing and refining trajectories from raw RFID readings. Empirical studies with implementations of the models and algorithms suggest that the paper's proposals are effective and efficient.

1 Introduction

People spend large parts of their lives in indoor spaces such as office buildings, shopping centers, airports, other transport infrastructures, etc. Meanwhile, such spaces are becoming increasingly large and complex. For example, the London Underground has 268 stations and a network of 408 kilometers [1]. Each hour, 146,000 passengers enter its tube system; the total number of daily passengers exceeds 4 million. Tracking moving objects in large and complex indoor spaces is very useful. Further, in smaller indoor spaces such as office buildings, important services are also enabled by tracking.

In particular, tracking enables a range of services akin to those enabled by GPS-based tracking in outdoor settings. For example, indoor tracking can enable indoor navigation and route guidance; it can provide insight into how and how much the indoor space is being used, which is important in

the pricing of advertisement space and store rental, and in planning applications. As a result, it is desired that indoor tracking can be carried out both effectively and efficiently.

This paper proposes a graph model-based indoor tracking approach that aims to support a range of positioning technologies. It thus offers a uniform data management infrastructure for positioning technologies such as Bluetooth and RFID.

The paper proposes a base graph model for indoor space. Given the floor plan of an indoor space, we create several base graphs and relevant mappings that represent the topology of the indoor space at different levels. On this foundation, a so-called deployment graph can then be constructed according to each deployment of a particular indoor positioning technology. A deployment graph enables tracking using the associated positioning technology.

The paper focuses on RFID-based positioning. Thus, a graph model is proposed that corresponds to a deployment of RFID readers that are classified as either *partitioning readers* that partition the indoor space into cells due to their location and *presence readers* that simply sense the presence of RFID tags. The cells obtained correspond to the vertices in the deployment graph and the partitioning readers correspond to the edges.

Several algorithms are proposed that utilize the data structures for constructing and refining trajectories from raw RFID readings. The algorithms enable both off-line and on-line, or real-time, construction of trajectories. Together with maximum-speed constraints, the data structures enable the algorithms to refine the trajectories to be more accurate, by limiting the possible regions in which a moving object can be. An empirical study with implementations of the data structures and algorithms suggest that the proposed indoor tracking framework is effective and efficient.

To summarize, the paper's contribution is fourfold. First, a versatile base graph model is proposed for indoor space, which accommodates one or several deployments of positioning technologies in a uniform manner. Second, a specific graph model is provided for RFID-based positioning. Third, detailed tracking algorithms are developed for RFID-

based positioning. Fourth, an empirical evaluation is reported.

The remainder of this paper is organized as follows. Following a brief review of related work in Section 2, Section 3 presents the base graph model, and Section 4 covers the RFID-specific deployment graph model. Section 5 details the tracking algorithms, and Section 6 presents the empirical study. Finally, Section 7 concludes and discusses possible future research.

2 Related Work

Modeling Indoor Space Symbolic models of indoor space are often preferred over traditional geometric coordinate models because they are able to capture the semantics associated with indoor entities [5].

A 3D Geometric Network Model exists that treats the vertical and horizontal connectivity relationship among 3D spatial cells separately [13]. A related 3D Indoor Geo-Coding technique employs the 3D *Poincaré Duality* [15] transformation to map 3D spatial cells from primal space to dual space. This technique is used for planing an evacuation network in a multi-level building [12]. A 3D metrical-topological model [22] describes both the shapes and connectivity of spatial cells for navigation purpose. A recent 3D model [6] combines space partitions with possible events in dual space for navigation in multi-layered buildings. A lattice-based semantic location model [14] is available that preserves semantic relationships and distances, e.g., the nearest neighbor relationship among indoor entities. This model is used to find the optimal route in indoor navigation, but is not applicable to tracking.

Different ways of transforming a floor plan to a graph are also discussed in the literature [8, 21]. Unlike the existing proposals, the base graph model proposed in this paper is specifically intended to support efficient indoor tracking.

Indoor Positioning and Tracking In outdoor settings, GPS is the dominant positioning technology. However, in indoor space, different positioning technologies including infrared, Bluetooth [3], RFID [2, 16] and Wi-Fi [10] are being used. RFID technology [19] has recently entered the mainstream because of reduced deployment costs. As RFID readers are much more expensive than tags, readers are often embedded in indoor spaces while the tags are attached to the moving objects. To increase the positioning accuracy in such a scenario, a small number of tags can be installed at fixed locations as reference points [16]. In special scenarios, e.g., guiding of the blind, large numbers of inexpensive tags are deployed in the indoor space, and each blind person is equipped with a reader [2].

Most works, including those cited above, are focused on hardware deployment for reliable indoor positioning ser-

VICES. In contrast, this paper aims to improve the indoor tracking accuracy from a data management perspective assuming a popular setting: RFID readers embedded in the indoor space and tags being associated with the objects.

3 Base Graph Model

By capturing the essential connectivity and accessibility, the base graph describes the topology of a floor plan of a possibly complex indoor space. For simplicity and due to space limitations, we use the floor plan in Figure 1(a) as the running example. Rooms, a staircase, a hallway, and doors are shown in the figure.

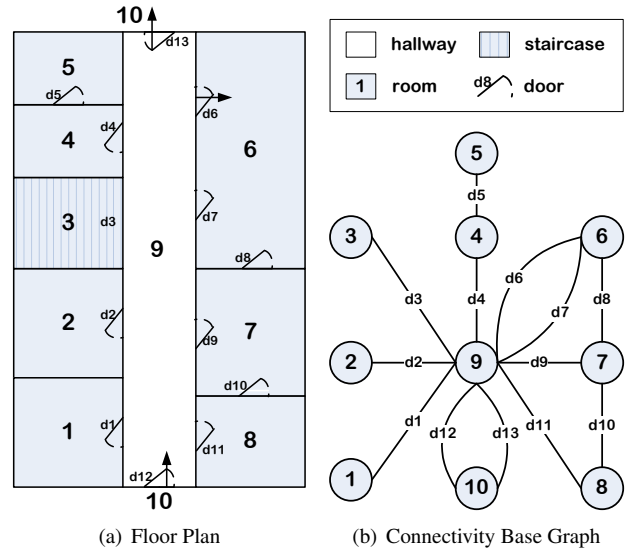


Figure 1. Floor plan and connectivity graph

3.1 Connectivity Base Graph

A base graph is constructed from a floor plan based on the *Poincaré Duality* [15]. Each separate partition in the floor plan, such as a single room, a staircase, and a hallway, is represented as a vertex in the base graph. In addition, the exterior of the indoor space is represented by a single vertex.

Edges are used for capturing different relationships between vertices. An edge can capture the *connectivity* between two partitions. Each connection in the floor plan, such as a door, a hatch, or an escape window, is then represented by an edge. For example, if two rooms are connected by a door, the two corresponding vertices are connected by an edge. Since several doors may connect two rooms, several edges between the same pair of vertices must be accommodated. Thus, the *connectivity base graph* is a labeled, undirected multi-graph [18], as shown in Figure 1(b).

The connectivity base graph G_{conn} is defined by the triple (V, E_d, Σ_{door}) , where V is the set of the vertices; E_d is the set of edges, where an edge is a pair $(\{v_i, v_j\}, k)$ such that $v_i, v_j \in V$ and $k \in \Sigma_{door}$; and Σ_{door} is a set of

edge labels that represent connections. Also, if two edges are different, their k values are distinct.

3.2 Accessibility Graph

Sometimes, a door may permit movement in only one direction. For example, a security check point in an airport allows only one-way movement. This also applies at entrances and exits in underground stations. Consider the floor plan in Figure 1(a): The door d12 only permits entry into the building, door d13 only permits exit from the building, and door d6 only permits entry into room 6 from the hallway. Accordingly, the *accessibility graph*, a labeled, directed graph, is constructed to represent the movement permitted by doors or connections. The graph G_{acc} is given by the triple $(V, E, \Sigma_{door}, l_e)$, where:

1. V is the set of the vertices.
2. E is the set of directed edges: $E = \{\{v_i, v_j\} \mid v_i, v_j \in V \wedge v_i \neq v_j\}$.
3. l_e is a function that maps edges to subsets of the set of doors: $l_e : E \rightarrow 2^{\Sigma_{door}}$.

The accessibility graph for our example is shown in Figure 2. The direction of an edge indicates the movement direction permitted by the corresponding connection.

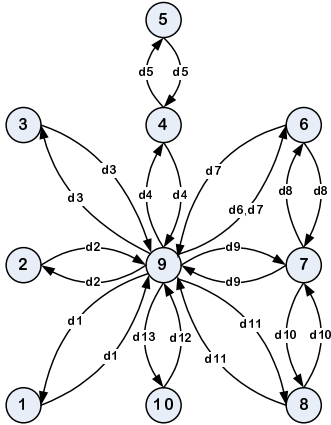


Figure 2. Accessibility Graph

3.3 Geometrical Information Mappings

In addition to the topological information of a floor plan, its geometrical information should also be captured. Polygons are used for representing partitions, so each vertex in V is mapped to a polygon. The *Building Partitions* mapping is defined as:

$$BuildingPartitions : V \rightarrow Polygons$$

Line segments are used for representing doors and other connections. Each edge in graph G_{conn} is thus mapped to a line segment. For simplicity, the label of an edge can be used to differentiate doors. Thus, the *Doors* mapping is defined as:

$$Doors : \Sigma_{door} \rightarrow Line\ Segments$$

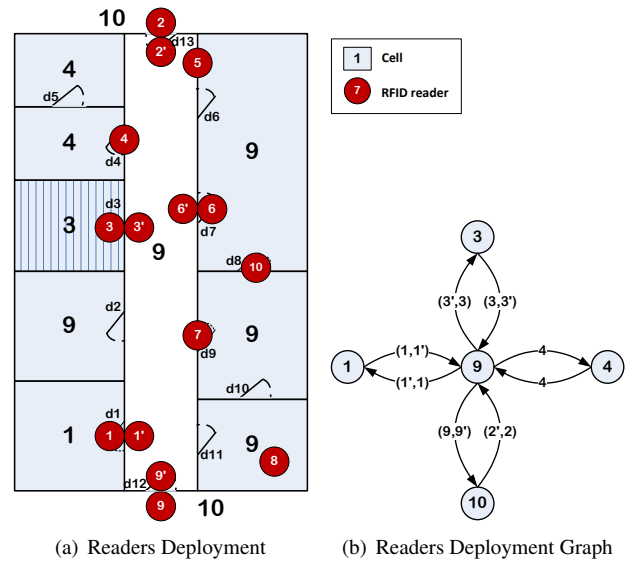
4 Deployment Graph for Indoor Positioning

Different positioning technologies, e.g. RFID and Bluetooth, can be deployed in the same indoor space [3, 4, 9]. Each deployment may cover only part of the space, or it may be capable of only detecting some movements in the space. Using the base graphs covered already, a so-called *deployment graph* can be constructed for each deployment of a positioning technology. This section discusses the deployment graph for the case of RFID-based positioning, in which a vertex corresponds to a subgraph in the base graph.

4.1 RFID Deployment Graph Model

RFID technology uses proximity analysis [11] for determining when a tag approaches a reader. We assume that the readers are embedded in the indoor space in known positions and that the tags are associated with the moving objects. Each reader has an activation range: Only tags that enter within this range will be seen. The typical activation range of RFID readers varies from tens of centimeters to 3 meters [19]. We assume that RFID readers are deployed according to requirements that stem from the intended applications. We also assume that all RFID readers deployed have disjoint activation ranges. We leave the more complex scenario of large, non-disjoint activation ranges for future work.

One possible RFID readers deployment for our example floor plan is shown in Figure 3(a). The RFID readers belong to one of two types.



(a) Readers Deployment

(b) Readers Deployment Graph

Figure 3. RFID Readers Deployment Graph

Partitioning Readers partition the indoor space into cells in the sense that an object cannot move from one cell to another without being observed. For example, a reader deployed by the single door of a room partitions the indoor space into two cells, one cell being the room and the other

being everything outside the room. In Figure 3(a), $reader_4$ is a partitioning reader.

It may be advantageous to deploy RFID readers by the doors and other connections so that the space is partitioned into cells. There are two options. (1) Only one reader is deployed by the door or connection, e.g., $reader_4$ in Figure 3(a). However, a single reader only cannot detect the movement direction of an object being observed. (2) A pair of partitioning readers (an exit/entry pair) is deployed by the door or connection, e.g., $reader_1$ and $reader_{1'}$ in Figure 3(a). The direction of an object moving through the door can be inferred from the reader sequence. For example, the reader sequence $reader_{1'}$ followed by $reader_1$ implies that the object enters room 1.

Presence Readers, e.g., $reader_8$ in Figure 3(a), are readers that do not contribute to the partitioning of the indoor space. These readers simply observe the presence (and non-presence) of tags in their activation ranges.

The partitioning capabilities of the partitioning readers can be captured using a deployment graph that captures the cells of the indoor space and the related topological information. Such a graph should satisfy two principles. (1) **Completeness**: The graph should capture all the cells present and all the connections between them. (2) **Minimality**: The graph should have as few vertices and edges as possible.

In the RFID deployment graph, vertices represent cells. A directed edge indicates that one can move from one cell to another without entering other cells, which is detected by a corresponding partitioning reader. Assume a set Σ_{reader} of readers and a set of vertices C that represents the cells created by the partitioning readers. The RFID deployment graph G_{RFID} is defined as a directed, labeled graph $(C, E_r, \Sigma_{reader}, l_e)$ where:

1. C is the set of the vertices.
2. E_r is the set of edges. An edge is an ordered pair $\langle c_i, c_j \rangle$ of distinct vertices from C .
3. $l_e : E_r \rightarrow 2^{\Sigma_{reader}} \cup 2^{\Sigma_{reader} \times \Sigma_{reader}}$ maps an edge to a partitioning reader or a partitioning reader pair.

The deployment graph of the RFID deployment in Figure 3(a) is shown in Figure 3(b).

The definition of l_e uses power sets because it is possible for two cells to be connected by two or even more doors, all of which correspond to one edge only in the deployment graph. The power sets enable the modeling of such cases.

Each cell created by the partitioning readers corresponds to one or more base graph partitions. In other words, several vertices in the connectivity base graph may correspond to one vertex in the RFID deployment graph. The following *Cells* mapping maintains the corresponding relationship.

$$Cells : V \rightarrow C$$

The bold numbers used as labels in Figure 3(a) capture the cells created by the partitioning readers and thus reflect the *Cells* mapping. For example, vertex v_7 is mapped to cell c_9 because an object can go from room 7 to hall 9 through door 10 and door 11 without being observed by any readers. Similarly, vertices v_2 , v_6 , and v_8 are mapped to cell c_9 , and vertex v_5 is mapped to cell c_4 .

The deployment graph and the *Cells* mapping will be used to facilitate the moving object trajectory construction and refinement, to be detailed in Section 5. We proceed to show how the deployment graph is constructed on top of the base graphs proposed in Section 3.

4.2 RFID Deployment Graph Construction

Before the deployment graph is constructed, the accurate deployment location and activation range of each RFID reader are recorded. For simplicity, we assume that the activation range (the maximum sensing radius) of a reader is a circle with maximum radius d_{max} . The following mapping is used:

$$\text{Mapping1: } \Sigma_{reader} \rightarrow \{(loc, range, flag) \mid loc \in R^2 \wedge range \in (0, d_{max}] \wedge flag \in \{true, false\}\}$$

Each reader $r_i \in \Sigma_{reader}$ is mapped to a triple $(loc, range, flag)$, where loc is the 2D location of the reader, $range$ is the reader's activation range, and $flag$ indicates whether the reader is a partitioning (*true*) or a presence (*false*) reader.

A mapping of readers to the cells that their activation ranges intersect is also introduced. This mapping can be derived from the floor plan and the location and activation range of each reader. The mapping will prove helpful during the trajectory construction.

$$\text{Mapping2: } \Sigma_{reader} \rightarrow 2^C$$

The RFID deployment graph is constructed according to Algorithm 1. The input is the set of all readers R , the connectivity base graph G_{conn} , and the accessibility graph G_{accs} . Following the definition of variables, the deployment graph is initialized (lines 1–2). The corresponding door of each edge in G_{conn} is checked to determine whether it is covered by a reader (lines 3–8). If so, the movement through the door can be detected by a reader. The edge is deleted from G_{conn} and the relationship of which door is covered by which readers is captured. The connected components of the remaining G_{conn} are determined (line 9).

A deployment graph vertex is created for each connected component, and the mapping from each vertex in the connected component to the new vertex is added to the *Cells* mapping (lines 10–13).

For each door covered by a reader, the corresponding edges are obtained from G_{accs} . If an edge's head and tail

Algorithm 1 RFIDGraphConstruction (Readers R , ConnectivityBaseGraph G_{conn} , AccessibilityGraph G_{accs})

```

1: Readers  $R' \leftarrow \emptyset$ ;  $DR(\langle \Sigma_{door} k', Readers RSet \rangle) \leftarrow \emptyset$ ;
   Connected Component  $CCs \leftarrow \emptyset$ ; int  $m \leftarrow 0$ ;
2:  $G_{RFID}(C, E_r, \Sigma_{reader} l_e) \leftarrow (\emptyset, \emptyset, R, NULL)$ ;
3: for each edge  $d_a$  in  $G_{conn}.E_d$  do
4:   for each reader  $r_b$  in  $R$  do
5:     if Circle(Mapping1( $r_b$ ).loc, Mapping1( $r_b$ ).range) covers
       Doors( $d_a.k$ ) then
6:       insert  $r_b$  into  $R'$ ;
7:   if  $|R'| > 0$  then
8:     delete  $d_a$  from  $G_{conn}.E_d$ ; insert  $(d_a.k, R')$  into  $DR$ ;
        $R' \leftarrow \emptyset$ ;
9: store all connected components of  $G_{conn}$  in  $CCs$ ;
10: for each connected component  $cc_c$  in  $CCs$  do
11:   create a new vertex  $c_m$  and add it to  $G_{RFID}.C$ ;  $m++$ ;
12:   for each vertex  $v_x$  in the vertices of  $cc_c$  do
13:     add the mapping  $(v_x \rightarrow c_m)$  to  $Cells$ ;
14:   for each  $dr_n$  in  $DR$  do
15:     for each  $e_l$  in  $G_{accs}.l_e^{-1}(dr_n.k')$  do
16:        $c_p \leftarrow Cells(e_l.v_i)$ ;  $c_q \leftarrow Cells(e_l.v_j)$ ;
17:       if  $c_p \neq c_q$  then
18:         if  $\langle c_p, c_q \rangle$  is not in  $G_{RFID}.E_r$  then
19:           add  $\langle c_p, c_q \rangle$  to  $G_{RFID}.E_r$ ;
20:         add the mapping  $(\langle c_p, c_q \rangle \rightarrow$ 
           readersequence( $dr_n.RSet$ )) to  $G_{RFID}.l_e$ ;

```

are mapped to different cells, an edge from the head's corresponding cell to the tail's corresponding cell is created in the deployment graph, and a mapping from the edge to the RFID reader sequence is added to the mapping l_e (lines 14–20). Here, function *readersequence* returns the possible RFID reader sequence for the edge.

Note that the deployment graph captures all movements that can be detected by deployed readers, which guarantees the completeness. Further, if any vertex or edge is deleted, the graph will lose the corresponding movement information, which implies minimality.

5 RFID-Based Indoor Tracking

This section covers the construction of moving-object trajectories from RFID readings. We first describe the RFID tag reading format and a pre-processing procedure. We then detail the trajectory construction in off-line and on-line settings.

5.1 Raw Trajectories: Sequences of RFID Tag Observation

Each RFID reader continuously detects and reports tags with a frequency determined by its sampling rate. Each tag reading is of the format $\langle readerID, tagID, t \rangle$, where *readerID* is the reader identifier, *tagID* is the identifier of the tag detected, and *t* is the time when the detection occurred.

Let the floor plan and the reader deployment be as shown in Figure 1(a) and Figure 3(a), respectively. A set of RFID readings for a moving object attached with tag_1 is shown in Table 1. This is the raw object's trajectory. Since we assume that RFID readers do not overlap (see Section 4.1), each tag can be detected by at most one reader at the same time.

Table 1. Raw Trajectory of tag_1

readerID	tagID	t	readerID	tagID	t
<i>reader₉</i>	<i>tag₁</i>	<i>t₁</i>	<i>reader₁₀</i>	<i>tag₁</i>	<i>t₄₀</i>
<i>reader₉</i>	<i>tag₁</i>	<i>t₂</i>	<i>reader₈</i>	<i>tag₁</i>	<i>t₅₅</i>
<i>reader_{9'}</i>	<i>tag₁</i>	<i>t₃</i>	<i>reader₈</i>	<i>tag₁</i>	<i>t₅₆</i>
<i>reader_{9'}</i>	<i>tag₁</i>	<i>t₄</i>	<i>reader_{3'}</i>	<i>tag₁</i>	<i>t₇₉</i>
<i>reader_{1'}</i>	<i>tag₁</i>	<i>t₇</i>	<i>reader_{3'}</i>	<i>tag₁</i>	<i>t₈₀</i>
<i>reader_{1'}</i>	<i>tag₁</i>	<i>t₈</i>	<i>reader₄</i>	<i>tag₁</i>	<i>t₈₅</i>
<i>reader₁</i>	<i>tag₁</i>	<i>t₉</i>	<i>reader₄</i>	<i>tag₁</i>	<i>t₈₆</i>
<i>reader₁</i>	<i>tag₁</i>	<i>t₁₀</i>	<i>reader₄</i>	<i>tag₁</i>	<i>t₈₇</i>
<i>reader₁</i>	<i>tag₁</i>	<i>t₂₁</i>	<i>reader₄</i>	<i>tag₁</i>	<i>t₁₀₀</i>
<i>reader₁</i>	<i>tag₁</i>	<i>t₂₂</i>	<i>reader₄</i>	<i>tag₁</i>	<i>t₁₀₁</i>
<i>reader_{1'}</i>	<i>tag₁</i>	<i>t₂₃</i>	<i>reader₄</i>	<i>tag₁</i>	<i>t₁₀₂</i>
<i>reader_{1'}</i>	<i>tag₁</i>	<i>t₂₄</i>	<i>reader_{6'}</i>	<i>tag₁</i>	<i>t₁₀₉</i>
<i>reader₇</i>	<i>tag₁</i>	<i>t₃₀</i>	<i>reader_{6'}</i>	<i>tag₁</i>	<i>t₁₁₀</i>
<i>reader₇</i>	<i>tag₁</i>	<i>t₃₁</i>	<i>reader₆</i>	<i>tag₁</i>	<i>t₁₁₁</i>
<i>reader₇</i>	<i>tag₁</i>	<i>t₃₂</i>	<i>reader₆</i>	<i>tag₁</i>	<i>t₁₁₂</i>
<i>reader₁₀</i>	<i>tag₁</i>	<i>t₃₉</i>	<i>reader₅</i>	<i>tag₁</i>	<i>t₁₂₅</i>

Plotting this raw trajectory along the time axis gives the result shown in Figure 4. The shaded blocks represent the time intervals during which position information is provided in the raw trajectory. There are also time intervals during which we do not observe the object. We call these *vacant time intervals*.

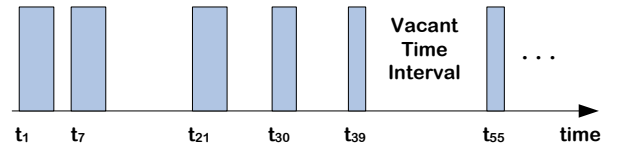


Figure 4. Vacant Time Intervals

We are interested in building a more detailed and accurate trajectory for the moving object, based on the raw trajectory. In particular, we aim to determine the moving object's locations during the vacant time intervals. By searching the RFID deployment graph, we can infer the possible regions of the moving object during the vacant time intervals. By applying maximum speed based position interpolation, we can further shrink the possible regions.

To facilitate the trajectory construction, we introduce a two-step pre-processing module in-between the RFID readers and the trajectory construction module. This module organizes all tag readings into suitable formats, as outlined in Figure 5 and explained next. The module differs from other

raw RFID data organizations (e.g., [20]) in that it is dedicated to indoor tracking, covering both off-line and on-line trajectory construction.

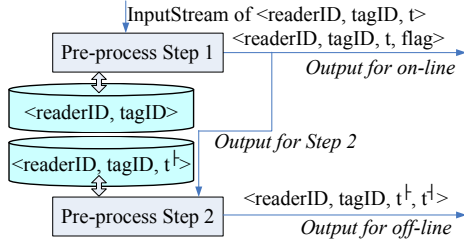


Figure 5. Pre-processing Module

Step 1 continuously receives tag readings in a stream from all readers. A local database maintains records in the format $\langle reader_i, tag_j \rangle$, which means that $reader_i$ has detected tag_j and the latter remains in the former’s activation range. If the reader and tag pair in an incoming reading is not stored in the current database, the tag is output with the current timestamp and the flag *START*, indicating that the tag is detected by the reader for the first time.

For each record in the database, if it does not appear in the current stream, which indicates that the tag has left the reader’s activation range, it is output with the previous sampling timestamp and the flag *END*, and it is removed from the database. The output of step 1 is intended for both on-line trajectory construction and for step 2 in the pre-processing.

Step 2 maintains a local database of records in the format $\langle reader_i, tag_j, t^+ \rangle$, which means that $reader_i$ detects tag_j for the first time at timestamp t^+ . When a record $\langle reader_i, tag_j, t_k, flag \rangle$ is received, the *flag* value is checked. If it is *START*, a new record $\langle reader_i, tag_j, t_k \rangle$ is added to the database, with t_k being used as the t^+ . Otherwise, a record $\langle reader_i, tag_j, t^+, t_k \rangle$ is output to indicate that during the time period $[t^+, t_k]$, tag_j is in $reader_i$ ’s range, but that it has now left. The corresponding record in the database is removed. The output of step 2 is used in off-line trajectory construction, since it returns the complete appearance period of a tag in a reader’s range.

5.2 Off-Line Tracking

An off-line trajectory of a moving object is a sequence of records of the form $\langle readerID, tagID, t^+, t^- \rangle$ where *readerID* and *tagID* identify a reader and a tag, respectively, while t^+ (t^-) is the first (last) times point when *readerID* detects *tagID*. Table 2 shows the off-line trajectory obtained from the raw trajectory in Table 1.

We first refine the trajectory obtained thus far with respect to the cells in the indoor space. For example, a refined trajectory can be: a person entered the building by the main entrance, walked through the hallway, entered a room; then

the person left the room, walked through the hallway, and left through the exit.

Table 2. Off-Line Trajectory of tag_1

readingID	readerID	tagID	t^+	t^-
reading ₁	reader ₉	tag ₁	t_1	t_2
reading ₂	reader _{9'}	tag ₁	t_3	t_4
reading ₃	reader _{1'}	tag ₁	t_7	t_8
reading ₄	reader ₁	tag ₁	t_9	t_{10}
reading ₅	reader ₁	tag ₁	t_{21}	t_{22}
reading ₆	reader _{1'}	tag ₁	t_{23}	t_{24}
reading ₇	reader ₇	tag ₁	t_{30}	t_{32}
reading ₈	reader ₁₀	tag ₁	t_{39}	t_{40}
reading ₉	reader ₈	tag ₁	t_{55}	t_{56}
reading ₁₀	reader _{3'}	tag ₁	t_{79}	t_{80}
reading ₁₁	reader ₄	tag ₁	t_{85}	t_{87}
reading ₁₂	reader ₄	tag ₁	t_{100}	t_{102}
reading ₁₃	reader _{6'}	tag ₁	t_{109}	t_{110}
reading ₁₄	reader ₆	tag ₁	t_{111}	t_{112}
reading ₁₅	reader ₅	tag ₁	t_{125}	t_{125}

Let V_{max} be the maximum speed of the object of interest. The off-line trajectory of the object is refined in three steps. (1) The relevant graph elements for each reader are obtained. (2) The possible cells of the object during the vacant time intervals are inferred from the deployment graph. (3) The candidate cells and indicated regions are further pruned according to the maximum speed constraints of the object [7, 17].

Refinement Step 1. Step 1 transforms an RFID reading sequence to corresponding vertices or edges in the RFID deployment graph, as described in Algorithm 2. Let a data structure TStep1 be defined as a sequence of $\langle tagID, t^+, t^-, GraphElement, Reader_1, Reader_2 \rangle$. By the definition of the RFID deployment graph, the inverse mapping $G_{RFID}.l_e^{-1}$ maps readers to edges. If two consecutive reading sequences are contiguous according to the sampling period ΔT , the two consecutive readings should stem from a partitioning pair, which map to an edge (lines 3–4). Otherwise, a single reading may come from either a partitioning reader or a presence reader. A partitioning reader is replaced by the set of corresponding edges according to $G_{RFID}.l_e^{-1}$ (lines 6–7). In contrast, a presence reader always corresponds to one or several cells according to Mapping 2 (lines 8–9). Table 3 shows the result of applying step 1 to the running example.

Having completed step 1, we are concerned with the relevant regions the moving object may be in during the vacant time intervals. For example, where is the moving object with tag_1 during the time interval $[t_5, t_6]$? During vacant time intervals, the moving object is beyond the activation range of any RFID reader and therefore cannot be detected by any RFID reader. For these intervals, we can limit the

Algorithm 2 TConstructStep1 (OfflineTrajectory $RSeq$)

```

1: TStep1  $tStep1 \leftarrow \emptyset$ ;
2: for each  $reading_i$  in  $RSeq$  do
3:   if  $reading_i.t^+ = reading_{i-1}.t^- + \Delta T$  then
4:     insert  $\langle reading_{i-1}.tagID, reading_{i-1}.t^+, reading_i.t^+, G_{RFID}.l_e^{-1}(reading_{i-1}.readerID, reading_i.readerID), reading_{i-1}.readerID, reading_i.readerID \rangle$  into  $tStep1$ 
5:   else
6:     if  $Mapping1(reading_{i-1}.readerID).flag$  then
7:       insert  $\langle reading_{i-1}.tagID, reading_{i-1}.t^+, reading_i.t^+, G_{RFID}.l_e^{-1}(reading_{i-1}.readerID), reading_{i-1}.readerID, reading_{i-1}.readerID \rangle$  into  $tStep1$ 
8:     else
9:       insert  $\langle reading_{i-1}.tagID, reading_{i-1}.t^+, reading_i.t^+, Mapping2(reading_{i-1}.Reader), reading_{i-1}.readerID, reading_{i-1}.readerID \rangle$  into  $tStep1$ 

```

relevant regions as much as possible. In step 2, we make use of the topology indicated by graph elements. In step 3, we introduce additional constraints imposed by the maximum speeds of the moving objects.

Refinement Step 2. Lines 1–11 of Algorithm 3 cover Step 2. Let a data structure TStep23 be defined as a sequence of $\langle tagID, t^+, t^-, GeometryRegion \rangle$. Each vacant time interval, as shown in Figure 4, has two adjacent records in the trajectory $tSeq$ obtained from step 1. Either record, by the graph element(s) it contains, indicates some region(s) within which the object may be in during the vacant time interval.

For each record $tseq_i$ in trajectory $tSeq$, the type of the graph elements in it and its previous record are checked (lines 2–6). If the previous one contains edges, the tail vertices of the edges are used as candidate vertices. Otherwise, the vertices themselves are used as candidate vertices. The case of the current record is similar, but the head vertices of the relevant edges are used instead (lines 7–10). Then the intersection of these two candidate vertex sets indicates the possible region(s) of the object during the vacant time interval. Table 4 shows the result of applying step 2 to the running example.

Refinement Step 3. We proceed to further reduce the parts of space an object may be in. Remember that the activation range of an RFID reader is a circular region around the reader’s location. Consider two consecutive reader observations in Table 2, e.g., $reading_2$ and $reading_3$. Two circular regions are determined by the activation ranges of $reader_{g'}$ at position $P_{g'}$ and $reader_{1'}$ at $P_{1'}$, as shown in Figure 6(a). Let $R_{g'}$ ($R_{1'}$) be the radius of the circular region centered at $P_{g'}$ ($P_{1'}$), $t_x \in [t_5, t_6]$, $\Delta t_1 = t_x - reading_2.t^+$, and $\Delta t_2 = reading_3.t^+ - t_x$.

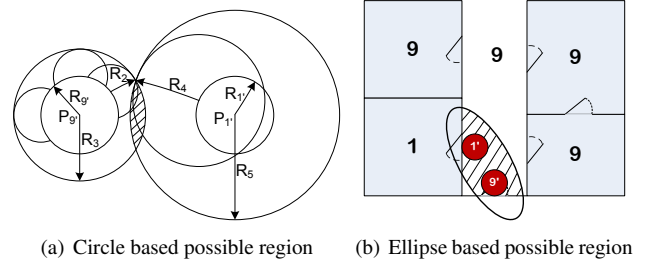
Algorithm 3 TConstructSteps23 (TStep1 $TSeq$)

```

1: VertexSet  $v_1 \leftarrow \emptyset, v_2 \leftarrow \emptyset$ ; Ellipse  $el \leftarrow NULL$ ; Polygon  $pl \leftarrow NULL$ ; TStep23  $tStep23 \leftarrow \emptyset$ ;
2: for each  $tseq_i$  in  $TSeq$  do
3:   if  $tseq_{i-1}.GraphElement$  are Edges then
4:      $v_1 \leftarrow$  tail vertices of  $tseq_{i-1}.GraphElement$ 
5:   else if  $tseq_{i-1}.GraphElement$  are Vertices then
6:      $v_1 \leftarrow tseq_{i-1}.GraphElement$ 
7:   if  $tseq_i.GraphElement$  are Edges then
8:      $v_2 \leftarrow$  head vertices of  $tseq_i.GraphElement$ 
9:   else if  $tseq_i.GraphElement$  are Vertices then
10:     $v_2 \leftarrow tseq_i.GraphElement$ 
11:    $pl \leftarrow BuildingPartitions(Cells^{-1}(v_1 \cap v_2))$ 
12:    $el \leftarrow \Theta(tseq_{i-1}.reader2, tseq_i.reader1, tseq_{i-1}.t^+, tseq_i.t^+)$ 
13:   insert  $\langle tseq_i.tagID, tseq_{i-1}.t^+ + \Delta T, tseq_i.t^+ - \Delta T, el \cap pl \rangle$  into  $tStep23$ 

```

If the object moves at its maximum speed V_{max} from any point inside the activation range of $reader_{g'}$ and its trajectory is a straight line, its position at time t_x will be bounded by circles centered at all possible start points with radius $R_2 = V_{max} * \Delta t_1$. Taking the activation range of $reader_{g'}$ into account, the possible positions are bounded by the circle with radius of $R_3 = R_{g'} + R_2$ centered at $P_{g'}$, as shown in the left part of Figure 6(a). Note that more generally, when the object’s speed is lower than V_{max} , or its trajectory is not a straight line, its position must be somewhere within the bounding circle.

**Figure 6. Maximum Speed Constraints**

Following the same line of reasoning, the object’s position at time t_x is also bounded by the circle centered at $P_{1'}$ with radius $R_5 = R_{1'} + V_{max} * \Delta t_2$, as shown in the right part in Figure 6(a).

The above constraints combine to indicate that at any time $t_x \in [t_5, t_6]$, the object can be anywhere in the intersection of the two relevant circular regions. This is illustrated by the shaded region in Figure 6(a).

Remember that an ellipse is the locus in a plane, such that for any point in the locus, the sum of its distances to two fixed points is a constant $2a$. The two fixed points, separated by a constant distance $2c$, are called the foci. Therefore, an ellipse can be used to infer the possible regions of a moving object with respect to two readers.

Table 3. Trajectory After Step 1

tagID	$[t^-, t^+]$	Graph Element	Reader ₁	Reader ₂
tag ₁	$[t_1, t_4]$	$e(10, 9)$	reader ₉	reader _{9'}
tag ₁	$[t_7, t_{10}]$	$e(9, 1)$	reader _{1'}	reader ₁
tag ₁	$[t_{21}, t_{24}]$	$e(1, 9)$	reader ₁	reader _{1'}
tag ₁	$[t_{30}, t_{32}]$	c ₉	reader ₇	reader ₇
tag ₁	$[t_{39}, t_{40}]$	c ₉	reader ₁₀	reader ₁₀
tag ₁	$[t_{55}, t_{56}]$	c ₉	reader ₈	reader ₈
tag ₁	$[t_{79}, t_{80}]$	c ₉	reader _{3'}	reader _{3'}
tag ₁	$[t_{85}, t_{87}]$	$e(4, 9), e(9, 4)$	reader ₄	reader ₄
tag ₁	$[t_{100}, t_{102}]$	$e(4, 9), e(9, 4)$	reader ₄	reader ₄
tag ₁	$[t_{109}, t_{112}]$	c ₉	reader _{6'}	reader ₆
tag ₁	$[t_{125}, t_{125}]$	c ₉	reader ₅	reader ₅

We continue to use *reading₂* and *reading₃* from above as an example. Due to the maximum speed limit, the possible positions of the moving object at time t_x is constrained by the ellipse whose foci are two points belonging to the two circles centered at $P_{9'}$ and $P_{1'}$. The length of the major axis is $2a = (\Delta t_1 + \Delta t_2) * V_{max} = V_{max} * (reading_3.t^+ - reading_2.t^-)$.

If the activation range of a reader is much smaller than the area of the ellipse, a simplified ellipse can be constructed as follows. The major axis is $2a = (R_{9'} + V_{max} * \Delta t_1) + (R_{1'} + V_{max} * \Delta t_2) = V_{max} * (reading_3.t^+ - reading_2.t^-) + R_{9'} + R_{1'}$. The foci are simply $P_{9'}$ and $P_{1'}$.

Consider again the running example. After step 2, the possible positions of the moving object in time interval $[t_5, t_6]$ are in cell 9, but not in any reader's activation range. In step 3, we are able to further prune the possible region by intersecting the ellipse described above and the region obtained in step 2. The position can be restricted to the intersection of the hallway and the ellipse, but not any reader's activation range: see the shaded region in Figure 6(b).

The above pruning based on the maximum speed limit is implemented by lines 11–13 in Algorithm 3. The polygon of the inferred cell is derived using the *Cells* mapping and the *BuildingPartitions* mapping (line 11). Given two reader identifiers and a time interval, function Θ returns the ellipse excluding the relevant readers' activation ranges (line 12). Finally, the intersection of the polygon and the ellipse are the refined trajectory after step 3 (line 13). The result for the running example is shown in Table 4.

5.3 On-Line Tracking

On-line tracking is intended to infer the trajectory in the time interval between the last RFID observation and the current time t_{now} or even into the future. The format of an off-line trajectory is not suitable here because we cannot wait for the information pertaining to the time t^- to become available. Instead, the on-line trajectory is a sequence of records in the format $\langle readerID, tagID, t, flag \rangle$.

In the on-line case, only first and last observations of a tag are involved: $flag = START$ indicates that t is the time of the first appearance with respect to *readerID*, and

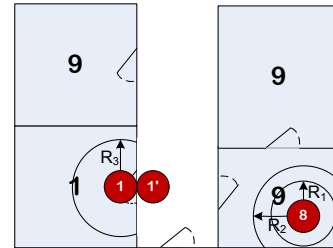
Table 4. Trajectory After Steps 2 and 3

tagID	$[t^-, t^+]$	Step 2	Step 3
tag ₁	$[t_5, t_6]$	c ₉	$c_9 \cap \Theta(reader_{9'}, reader_{1'}, t_5, t_6)$
tag ₁	$[t_{11}, t_{20}]$	c ₁	$c_1 \cap \Theta(reader_{1'}, reader_{1'}, t_{11}, t_{20})$
tag ₁	$[t_{25}, t_{29}]$	c ₉	$c_9 \cap \Theta(reader_{1'}, reader_{7'}, t_{25}, t_{29})$
tag ₁	$[t_{33}, t_{38}]$	c ₉	$c_9 \cap \Theta(reader_{7'}, reader_{10'}, t_{33}, t_{38})$
tag ₁	$[t_{41}, t_{54}]$	c ₉	$c_9 \cap \Theta(reader_{10'}, reader_{8'}, t_{41}, t_{54})$
tag ₁	$[t_{57}, t_{78}]$	c ₉	$c_9 \cap \Theta(reader_{8'}, reader_{3'}, t_{57}, t_{78})$
tag ₁	$[t_{81}, t_{84}]$	c ₉	$c_9 \cap \Theta(reader_{3'}, reader_{4'}, t_{81}, t_{84})$
tag ₁	$[t_{88}, t_{99}]$	c ₄ , c ₉	$(c_4, c_9) \cap \Theta(reader_{4'}, reader_{4'}, t_{88}, t_{89})$
tag ₁	$[t_{103}, t_{108}]$	c ₉	$c_9 \cap \Theta(reader_{4'}, reader_{6'}, t_{103}, t_{108})$
tag ₁	$[t_{113}, t_{114}]$	c ₉	$c_9 \cap \Theta(reader_{6'}, reader_{5'}, t_{113}, t_{114})$

$flag = END$ indicates t is the time of the last appearance. The data is obtained from pre-processing step 1 as described in Section 5.1.

Let the last observation of a specific moving object be $\langle readerID, tagID, t, flag \rangle$. If $tag=START$, the object is currently in the activation range of *readerID*. For example, if the latest output of pre-processing step 1 is $\langle reader_8, tag_1, t_{55}, START \rangle$, the moving object is in the activation range of *reader₈* at time t_{now} .

Next, $tag=END$ indicates that the object is currently beyond the the activation range of the *readerID* and not in the range of any other reader. Given the maximum speed V_{max} , the possible position of the object is constrained by a circle determined by the most recent observing reader's activation range. The floor plan also constrains the possible positions of the moving object. For example, if the most recent output of pre-processing step 1 is $\langle reader_8, tag_1, t_{56}, END \rangle$, the object is located inside the circle centered at $mapping_1(reader_8).loc$ with radius $V_{max} * (t_{now} - t_{56})$. Depending on V_{max} and t_{now} the circle can be either fully or partially within the indoor space, as shown at the bottom-right corner in Figure 7.

**Figure 7. Online Tracking**

In addition, if an object has recently been detected by a partitioning reader pair, its current possible region can be refined further. Consider the left part in Figure 7. If the two most recent outputs from step 1 are $\langle reader_{1'}, tag_1, t_8, END \rangle$ and $\langle reader_{1'}, tag_1, t_9, START \rangle$, the object has entered the cell (room). Therefore, it must be located in the intersection of the room and the circle centered at $mapping_1(reader_{1'}).loc$ with radius $V_{max} * (t_{now} - t_9)$.

6 Experimental Study

We generate moving objects using a floor plan with 30 rooms and 2 staircases, all of which are connected by a hallway. An RFID reader is deployed by the doors of all rooms. Readers are also deployed along the hallway and in the staircases. Three rules are used to generate movements: 1) an object in a room can move to the hallway or move inside the room; 2) an object in a staircase can move to the hallway, or move in the staircase; 3) an object in the hallway can move in the hallway, move to one of the staircases, or move to one of the rooms. All objects move with constant speed 4 km/hour.

We vary three parameters: the number of moving objects, the lifespan of the objects, and the activation ranges of readers. The settings of these parameters are listed in Table 5, with default values given in bold. In each single experiment, average results for all involved objects are reported. All experiments are implemented in C# and run on

Table 5. Parameter Settings

Parameters	Settings
Number of objects	10, 100, 500, 1000 , 2000, ..., 5000
Lifespan of objects	50, 100 , 200, 300 (sec)
Activation range	100 , 150, 200 (cm)

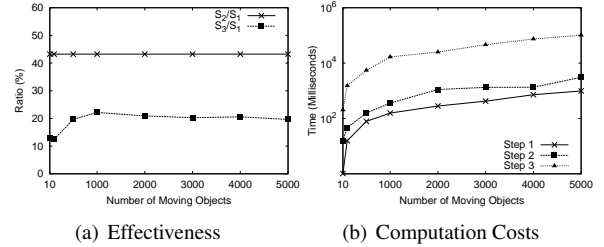
Windows XP professional with a 2.66GHz Core2 Duo CPU and 3.25GB main memory.

We do not provide comparisons of our proposal with existing, related proposals (see Section 2) because these are all focused on improving the positioning accuracy at either the hardware or the deployment level. This is orthogonal to our focus that simply assumes a given positioning accuracy.

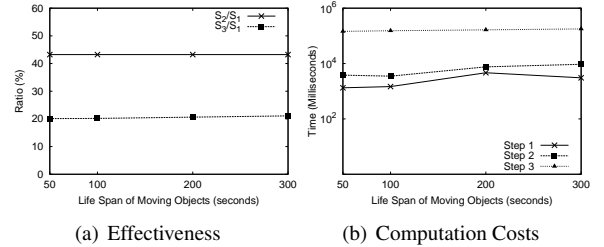
6.1 Results for Off-Line Tracking

We consider two performance metrics: the refinement effectiveness and the refinement computation cost. After each refinement step, the effectiveness is measured as the area of the regions in which objects can be during the vacant intervals. In particular, for each individual object, we calculate the area averaged over all its vacant intervals. We use S_i , $i = 1, 2, 3$ to denote the area averaged over all moving object after each step.

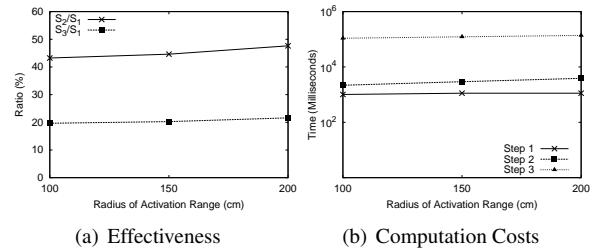
Recall that refinement step 1 only obtains relevant edges and vertices in the RFID deployment graph without constraining the possible region for each object. Therefore, S_1 equals the area of the building minus the sum of the areas of all activation ranges of readers. Step 2 constrains the possible regions to several cells in the RFID deployment graphs. Therefore, S_2 equals the sum of the areas of all cells obtained by step 2. Step 3 constrains the possible regions using ellipses based on the maximum speed. Therefore, S_3 equals the area of the intersection region of the cells in step 2 and the ellipses.



(a) Effectiveness (b) Computation Costs
Figure 8. Effect of Object Number



(a) Effectiveness (b) Computation Costs
Figure 9. Effect of Lifespan



(a) Effectiveness (b) Computation Costs
Figure 10. Effect of Radius

To compare the effectiveness among the refinements, we plot the ratios S_2/S_1 and S_3/S_1 in Figures 8(a), 9(a), and 10(a). It can be seen that both steps 2 and 3 improve the tracking accuracy substantially. In particular, step 2 reduces the possible regions to less than half for almost all settings except (very) large reader activation ranges. Step 3 then further doubles the accuracy. These findings suggest that the proposed graph model-based tracking is effective.

Results on the CPU cost with different parameter settings are shown in Figures 8(b), 9(b), and 10(b). Step 3 incurs higher CPU cost than steps 1 and 2 because it involves the computation of ellipse-based intersections. Step 2 incurs slightly higher CPU cost than does step 1. However, the overall CPU costs, the sum of all three steps, are still low considering off-line tracking is conducted here. These findings suggest that the proposal is efficient.

6.2 Results of On-Line Tracking

In this experiment, we use 1000 objects and vary the reader activation range. We compare the area of the possible region before online refinement with that after online refinement. The area before online refinement is the area of the floor plan minus the sum of the areas of the activation ranges of the readers, i.e., S_1 from the off-line tracking. The area after online refinement is the intersection of the maxi-

mum speed-based circle and a relevant cell.

The accuracy results are reported in Figure 11(a). The x-axis records the time since the last reading. Online tracking exhibits marked accuracy improvements for shorter time periods. As time passes, the effectiveness decreases because larger circles are used to constrain the objects.

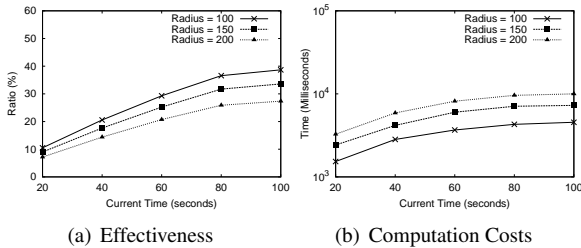


Figure 11. Results of Online Tracking

The CPU costs are shown in Figure 11(b). Large activation ranges yield lower accuracy and higher CPU cost because larger circles are used when computing intersections. However, for small and medium reader activation ranges, on-line tracking is quite efficient as the CPU cost is no more than seconds.

7 Conclusion and Research Directions

We propose a uniform graph model framework for indoor tracking that consists of base graphs and deployment graphs. From an indoor space floor plan, connectivity and accessibility graphs are constructed as base graphs to represent different kinds of topological information. A deployment graph is then used for each specific deployment of a positioning technology in the indoor space. Focusing on RFID positioning technology, the paper presents both off-line and on-line tracking algorithms. The experimental results show that our proposal can considerably improve the indoor tracking accuracy at low computational cost.

Several directions for future research exist. First, it is relevant to extend the deployment graphs to accommodate RFID readers with large and overlapping activation ranges. Second, it is of interest to attempt to also accommodate Wi-Fi-based positioning in our approach. Third, it is possible to further improve the tracking accuracy by concurrently using multiple deployment graphs for several positioning technologies. Fourth, we expect that on-line tracking can be enhanced, e.g., to give better predictions, by applying association rules which can be mined from large amount of historical data.

Acknowledgments

This research was partially supported by the Indoor Spatial Awareness project of the Korean Land Spatialization Group and BK21 program. C. S. Jensen is currently a Visiting Scientist at Google Inc.

References

- [1] Transport for London. <http://www.tfl.gov.uk/>.
- [2] T. Amemiya, J. Yamashita, K. Hirota, and M. Hirose. Virtual leading blocks for the deaf-blind: A real-time way-finder by verbal-nonverbal hybrid interface and high-density RFID tag space. In *Virtual Reality*, pages 165–172, 2004.
- [3] G. Anastasi, R. Bandelloni, M. Conti, F. Delmastro, E. Gregori, and G. Mainetto. Experimenting an indoor bluetooth-based positioning service. In *Proc. ICDCS Workshops*, pages 480–483, 2003.
- [4] P. Bahl and V. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proc. INFOCOM*, pages 775–784, 2000.
- [5] C. Becker and F. Dürr. On location models for ubiquitous computing. *Personal Ubiquitous Computing*, 9(1):20–31, 2005.
- [6] T. Becker, C. Nagel, and T. H. Becker. A Multilayered space-event model for navigation in indoor spaces. *Proc. 3D Geo-Info*, 2008 (to appear).
- [7] A. Civilis, C. S. Jensen, and S. Pakalnis. Techniques for efficient road-network-based tracking of moving objects. *TKDE*, 17(5):698–712, 2005.
- [8] G. Franz, H. Mallot, J. Wiener, and K. Neurowissenschaft. Graph-based Models of Space in Architecture and Cognitive Science - a Comparative Analysis. In *Proc. ICSCI*, pages 30–38, 2005.
- [9] I. Guvenc, C. Abdallah, R. Jordan, and O. Dedeoglu. Enhancements to RSS Based Indoor Tracking Systems Using Kalman Filters. *Proc. GSPx*, 2003.
- [10] M. Hermersdorf. Indoor Positioning with a WLAN Access Point List on a Mobile Device. In *Proc. Workshop on World-Sensor-Web*, 2006.
- [11] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *Computer*, 34:57–66, 2001.
- [12] J. Lee. 3D GIS for geo-coding human activity in micro-scale urban environments. In *GIScience*, pages 162–178, 2004.
- [13] J. Lee. A spatial access-oriented implementation of a 3-D GIS topological data model for urban entities. *Geoinformatica*, 8(3):237–264, 2004.
- [14] D. Li and D. L. Lee. A lattice-based semantic location model for indoor navigation. In *Proc. MDM*, pages 17–24, 2008.
- [15] J. Munkres. *Elements of algebraic topology*. Addison Wesley Publishing Company, 1993.
- [16] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. LANDMARC: Indoor location sensing using active RFID. In *Pervasive Computing and Communications*, pages 407–415, 2003.
- [17] D. Pfoser and C. S. Jensen. Capturing the Uncertainty of Moving-Object Representations. In *Proc. SSTD*, pages 111–131. Springer, 1999.
- [18] N. Trinajstić. *Chemical graph theory*. CRC Press, 1983.
- [19] R. Want. *RFID Explained: A Primer on Radio Frequency Identification Technologies*. Morgan and Claypool, 2006.
- [20] C.H. Lee and C.W. Chung. *Efficient storage scheme and query processing for supply chain management using RFID*. Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008.
- [21] C. van Treeck and E. Rank. *Analysis of Building Structure and Topology Based on Graph Theory*. Proceeding of the 10th International Conference on Computing in Civil and Building Engineering. Weimar, 2004.
- [22] E. Whiting, J. Battat, and S. Teller. Topology of urban environments. In *Proc. CAAD*. Springer Verlag, 2007.