# Manage the Data from Indoor Spaces: Models, Indexes & Query Processing

Huan Li

Database Laboratory, Zhejiang University

*lihuancs@zju.edu.cn*

March 25, 2016

# Overview


1. 1. Outlines
2. 2. Indoor Space Models & Applications
3. 3. Indoor Data Cleansing
4. 4. Indoor Movement Analysis
5. 5. Appendix

1. 1. Outlines
2. 2. Indoor Space Models & Applications
3. 3. Indoor Data Cleansing
4. 4. Indoor Movement Analysis
5. 5. Appendix

1. Outlines

2. Indoor Space Models & Applications

3. Indoor Data Cleansing

4. Indoor Movement Analysis

5. Appendix

# About This Work...

*A Foundation for Efficient Indoor Distance-Aware Query Processing*. [4]
H. Lu, X. Cao, and C. S. Jensen.

- Published at *ICDE' 2012*.
- First time to propose a distance-aware indoor space model that integrates indoor distance seamlessly.
- Accompanying, efficient algorithms for computing indoor distances.
- Indexing framework that accommodates indoor distances.

# Motivation

- A variety of LBS services are useful in indoor space.
  - a museum guidance service in a complex exhibition
  - boarding reminder service in an airport, to remind the passengers especially those far away from their gates or departures
- Such indoor LBSs will benefit from the availability of accurate indoor distances.
  - indoor space entities enable as well as constrain indoor movement, thus makes traditional space model for Euclidean/spatial network spaces unsuitable.
  - existing indoor space models [7, 8, 9] pay little attention to indoor distances.

# Indoor Topology Mapping Structures

Mapping $D2P$ maps a door $d_k$ to one or two partition pairs [1] $(v_i, v_j)$ such that one can move from partition [2] $v_i$ to partition $v_j$ through door $d_k$:

[1] the basic assumption that a door corresponds to two doors can be extended by converting a door to multiple doors.

[2] a partition indicates a room, a hallway or a staircase.

# Indoor Topology Mapping Structures

Mapping $D2P$ maps a door $d_k$ to one or two partition pairs [1] $(v_i, v_j)$ such that one can move from partition [2] $v_i$ to partition $v_j$ through door $d_k$:

$$D2P : \mathcal{S}_{door} \rightarrow 2^{\mathcal{S}_{partition}} \times 2^{\mathcal{S}_{partition}} \tag{1}$$

[1] the basic assumption that a door corresponds to two doors can be extended by converting a door to multiple doors.

[2] a partition indicates a room, a hallway or a staircase.

# Indoor Topology Mapping Structures

Mapping $D2P$ maps a door $d_k$ to one or two partition pairs [1] $(v_i, v_j)$ such that one can move from partition [2] $v_i$ to partition $v_j$ through door $d_k$:

$$D2P : \mathcal{S}_{door} \rightarrow 2^{\mathcal{S}_{partition}} \times 2^{\mathcal{S}_{partition}} \quad (1)$$

For *enterable partition* of door $d_k$:

[1] the basic assumption that a door corresponds to two doors can be extended by converting a door to multiple doors.

[2] a partition indicates a room, a hallway or a staircase.

# Indoor Topology Mapping Structures

Mapping $D2P$ maps a door $d_k$ to one or two partition pairs [1] $(v_i, v_j)$ such that one can move from partition [2] $v_i$ to partition $v_j$ through door $d_k$:

$$D2P : \mathcal{S}_{door} \rightarrow 2^{\mathcal{S}_{partition}} \times 2^{\mathcal{S}_{partition}} \quad (1)$$

For *enterable partition* of door $d_k$:

$$D2P_{\sqsupset} : \mathcal{S}_{door} \rightarrow 2^{\mathcal{S}_{partition}} \quad (2)$$

[1] the basic assumption that a door corresponds to two doors can be extended by converting a door to multiple doors.

[2] a partition indicates a room, a hallway or a staircase.

# Indoor Topology Mapping Structures

Mapping $D2P$ maps a door $d_k$ to one or two partition pairs [1] $(v_i, v_j)$ such that one can move from partition [2] $v_i$ to partition $v_j$ through door $d_k$:

$$D2P : \mathcal{S}_{door} \rightarrow 2^{\mathcal{S}_{partition}} \times 2^{\mathcal{S}_{partition}} \tag{1}$$

For *enterable partition* of door $d_k$:

$$D2P_{\sqsupset} : \mathcal{S}_{door} \rightarrow 2^{\mathcal{S}_{partition}} \tag{2}$$

For *leaveable partition* of door $d_k$:

[1] the basic assumption that a door corresponds to two doors can be extended by converting a door to multiple doors.

[2] a partition indicates a room, a hallway or a staircase.

# Indoor Topology Mapping Structures

Mapping $D2P$ maps a door $d_k$ to one or two partition pairs [1] $(v_i, v_j)$ such that one can move from partition [2] $v_i$ to partition $v_j$ through door $d_k$:

$$D2P : \mathcal{S}_{door} \to 2^{\mathcal{S}_{partition}} \times 2^{\mathcal{S}_{partition}} \tag{1}$$

For *enterable partition* of door $d_k$:

$$D2P_{\sqsupset} : \mathcal{S}_{door} \to 2^{\mathcal{S}_{partition}} \tag{2}$$

For *leaveable partition* of door $d_k$:

$$D2P_{\sqsubset} : \mathcal{S}_{door} \to 2^{\mathcal{S}_{partition}} \tag{3}$$

[1] the basic assumption that a door corresponds to two doors can be extended by converting a door to multiple doors.

[2] a partition indicates a room, a hallway or a staircase.

# Indoor Topology Mapping Structures

The mapping $P2D_{\sqsupset}$ maps a partition $v$ to all the doors through which one can enter $v$:

# Indoor Topology Mapping Structures

The mapping $P2D_{\sqsupset}$ maps a partition $v$ to all the doors through which one can enter $v$:

$$P2D_{\sqsupset} : \mathcal{S}_{partition} \rightarrow 2^{\mathcal{S}_{door}} \tag{4}$$

# Indoor Topology Mapping Structures

The mapping $P2D_{\sqsupset}$ maps a partition $v$ to all the doors through which one can enter $v$:

$$P2D_{\sqsupset} : \mathcal{S}_{partition} \rightarrow 2^{\mathcal{S}_{door}} \tag{4}$$

The mapping $P2D_{\sqsubset}$ maps a partition $v$ to all the doors through which one can leave $v$:

# Indoor Topology Mapping Structures

The mapping $P2D_{\sqsupset}$ maps a partition $v$ to all the doors through which one can enter $v$:

$$P2D_{\sqsupset} : \mathcal{S}_{partition} \rightarrow 2^{\mathcal{S}_{door}} \tag{4}$$

The mapping $P2D_{\sqsubset}$ maps a partition $v$ to all the doors through which one can leave $v$:

$$P2D_{\sqsubset} : \mathcal{S}_{partition} \rightarrow 2^{\mathcal{S}_{door}} \tag{5}$$

# Indoor Topology Mapping Structures

The mapping $P2D_{\sqsupset}$ maps a partition $v$ to all the doors through which one can enter $v$:

$$P2D_{\sqsupset} : \mathcal{S}_{partition} \rightarrow 2^{\mathcal{S}_{door}} \tag{4}$$

The mapping $P2D_{\sqsubset}$ maps a partition $v$ to all the doors through which one can leave $v$:

$$P2D_{\sqsubset} : \mathcal{S}_{partition} \rightarrow 2^{\mathcal{S}_{door}} \tag{5}$$

The mapping $P2D$ is used when there's no need to differentiate the directionality:

# Indoor Topology Mapping Structures

The mapping $P2D_{\sqsupset}$ maps a partition $v$ to all the doors through which one can enter $v$:

$$P2D_{\sqsupset} : \mathcal{S}_{partition} \rightarrow 2^{\mathcal{S}_{door}} \tag{4}$$

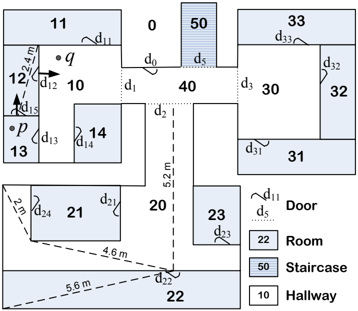
The mapping $P2D_{\sqsubset}$ maps a partition $v$ to all the doors through which one can leave $v$:

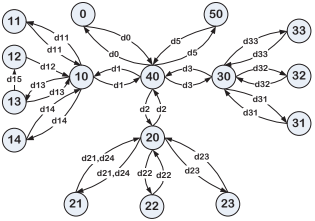
$$P2D_{\sqsubset} : \mathcal{S}_{partition} \rightarrow 2^{\mathcal{S}_{door}} \tag{5}$$

The mapping $P2D$ is used when there's no need to differentiate the directionality:

$$P2D(v_i) : P2D_{\sqsupset}(v_i) \cup P2D_{\sqsubset}(v_i) \tag{6}$$

# Accessibility Base Graph




## Example

$D2P_{\sqsupset}(d_{12}) = \{v_{10}\}$, $D2P_{\sqsubset}(d_{12}) = \{v_{12}\}$

$P2D_{\sqsupset}(v_{13}) = \{d_{13}\}$,

$P2D_{\sqsubset}(v_{13}) = \{d_{13}, d_{15}\}$

## Accessibility Base Graph

- $G_{accs} = \{V, E_a, L\}$
- $V = \mathcal{S}_{partition}$ is the set of vertices
- $E_a = \{(v_i, v_j, d_k) | (v_i, v_j) \in D2P(d_k)\}$ is the set of labeled, directed edges
- $L = \mathcal{S}_{door}$ is the set of edge labels

# Distance-Aware Model

The $G_{accs}$ graph does not capture indoor distance information. **Extended Graph Model** is proposed to integrate indoor distances into the graph in a seamless way. *Minimum Indoor Walking Distance*(MIWD) is used.

## Extended Graph Model $G_{dist} = \{V, E_a, L, f_{dv}, f_{d2d}\}$

- $V = \mathcal{S}_{partition}$ is the set of vertices
- $E_a = G_{accs}.E_a$
- $L = \mathcal{S}_{door}$ is the set of edge labels
- $f_{dv} = \mathcal{S} \times V \to \mathcal{R} \cup \{\infty\}$ maps an edge to a distance value.

$$f_{dv} = \begin{cases} \max_{p \in v_j} ||d_i, p||, & if \quad v_j \in D2P_{\sqsupset}; \\ \infty, & otherwise. \end{cases}$$

- $f_{d2d} = V \times \mathcal{S}_{door} \times \mathcal{S}_{door} \to \mathcal{R} \cup \{\infty\}$ maps a 3-tuple to a distance value.

$$f_{dv} = \begin{cases} ||d_i, d_j||_{v_k}, & if \quad d_i \in P2D_{\sqsupset}(v_k) and d_j \in P2D_{\sqsubset}(v_k); \\ \infty, & if \quad d_i = d_j and d_i, d_j \in P2D(v_k); \\ 0, & otherwise. \end{cases}$$

# Indoor Distance Computation: *door-to-door distance*

**Algorithm 1 d2dDistance**(Source door $d_s$, destination door $d_t$)

initialize a min-heap $H$
**for** each door $d_i \in \mathcal{S}_{door}$ **do**
  **if** $d_i \neq d_s$ **then**
    $dist[d_i] \leftarrow \infty$
  **else**
    $dist[d_i] \leftarrow 0$
  enheap$(H, \langle d_i, dist[d_i] \rangle)$
  $prev[d_i] \leftarrow$ null
**while** $H$ is not empty **do**
  $\langle d_i, dist[d_i] \rangle \leftarrow$ deheap$(H)$
  **if** $d_i = d_t$ **then**
    **return** $dist[d_i]$
  mark door $d_i$ as visited
  $parts \leftarrow D2P_{\sqsubset}(d_i)$
  **for** each partition $v \in parts$ **do**
    **for** each unvisited door $d_j \in P2D_{\sqsubset}(v)$ **do**
      **if** $dist[d_i] + G_{dist}.f_{d2d}(v, d_i, d_j) < dist[d_j]$ **then**
        $dist[d_j] \leftarrow dist[d_i] + G_{dist}.f_{d2d}(v, d_i, d_j)$
        replace $d_j$'s element in $H$ by $\langle d_j, dist[d_j] \rangle$
        $prev[d_j] \leftarrow (v, d_i)$

# Indoor Distance Computation: *point-to-point distance*

**Algorithm 2 pt2ptDistance**(Source indoor position $p_s$, destination indoor position $p_t$)

$v_s \leftarrow$ getHostPartition($p_s$)
$v_t \leftarrow$ getHostPartition($p_t$)
$dist \leftarrow \infty$
**for** each door $d_s \in P2D_{\sqsubset}(v_s)$ **do**
    $dist_1 \leftarrow dist_V(p_s, d_s)$
    **for** each door $d_t \in P2D_{\sqsupset}(v_t)$ **do**
        $dist_2 \leftarrow dist_V(p_t, d_t)$
        **if** $dist > dist_1 +$ d2dDistance$(d_s, d_t) + dist_2$ **then**
            $dist \leftarrow dist_1 +$ d2dDistance$(d_s, d_t) + dist_2$
**return** $dist$

# Indoor Distance Computation: *point-to-point distance* (I)

---

**Algorithm 2 pt2ptDistance**(Source indoor position $p_s$, destination indoor position $p_t$)

---

$v_s \leftarrow$ getHostPartition($p_s$)
$v_t \leftarrow$ getHostPartition($p_t$)
$dist \leftarrow \infty$
**for** each door $d_s \in P2D_{\sqsubset}(v_s)$ **do**
    $dist_1 \leftarrow dist_V(p_s, d_s)$
    **for** each door $d_t \in P2D_{\sqsupset}(v_t)$ **do**
        $dist_2 \leftarrow dist_V(p_t, d_t)$
        **if** $dist > dist_1+$ d2dDistance($d_s, d_t$) $+ dist_2$ **then**
            $dist \leftarrow dist_1+$ d2dDistance($d_s, d_t$) $+ dist_2$
**return** $dist$

---

# Indoor Distance Computation: *point-to-point distance* (II)

**Algorithm 3 pt2ptDistance2**(Source indoor position $p_s$, destination indoor position $p_t$)

$v_s \leftarrow$ getHostPartition($p_s$)
$v_t \leftarrow$ getHostPartition($p_t$)
$doors_s \leftarrow P2D_{\sqsubset}(v_s)$
$doors_t \leftarrow P2D_{\sqsupset}(v_t)$
**for** each door $d_s \in doors_s$ **do**
    $np \leftarrow$ the partition in $D2P_{\sqsubset}(d_s) \setminus \{v_s\}$
    **if** $P2D_{\sqsubset}(np) = \{d_s\}$ **and** $np \neq v_t$ **then**
        remove $d_s$ from $doors_s$
$dist_m \leftarrow \infty$
**for** each door $d_s \in doors_s$ **do**
    $doors \leftarrow \emptyset$
    **for** each door $d_t \in doors_t$ **do**
        **if** $dist_V(p_s, d_s) + dist_V(p_t, d_t) < dist_m$ **then**
            add $d_t$ to $doors$
    initialize a min-heap $H$
    **for** each door $d_i \in \mathcal{S}_{door}$ **do**
        **if** $d_i \neq d_s$ **then**
            $dist[d_i] \leftarrow \infty$
        **else**
            $dist[d_i] \leftarrow 0$
        enheap($H, \langle d_i, dist[d_i] \rangle$)
    **while** $H$ is not empty **do**
        $\langle d_i, dist[d_i] \rangle \leftarrow$ deheap($H$)
        **if** $d_i \in doors$ **then**
            $doors \leftarrow doors \setminus \{d_i\}$
            **if** $dist_m > dist_V(p_s, d_s) + dist[d_i] + dist_V(p_t, d_i)$ **then**
                $dist_m \leftarrow dist_V(p_s, d_s) + dist[d_i] + dist_V(p_t, d_i)$
            **if** $doors = \emptyset$ **then**
                **break**
        mark door $d_i$ as visited
        $parts \leftarrow D2P_{\sqsubset}(d_i)$
        **for** each partition $v \in parts$ **do**
            **for** each unvisited door $d_j \in P2D(v)$ **do**
                **if** $d_j \in P2D_{\sqsubset}(v)$ **then**
                    **if** $dist[d_i] + G_{dist}.f_{d2d}(v, d_i, d_j) < dist[d_j]$ **then**
                        $dist[d_j] \leftarrow dist[d_i] + G_{dist}.f_{d2d}(v, d_i, d_j)$
**return** $dist_m$

# Indoor Distance Computation: *point-to-point distance* (III)

**Algorithm 3 pt2ptDistance2**(Source indoor position $p_s$, destination indoor position $p_t$)

$v_s \leftarrow$ getHostPartition($p_s$)
$v_t \leftarrow$ getHostPartition($p_t$)
$doors_s \leftarrow P2D_{\sqsubset}(v_s)$
$doors_t \leftarrow P2D_{\sqsupset}(v_t)$
**for** each door $d_s \in doors_s$ **do**
    $np \leftarrow$ the partition in $D2P_{\sqsubset}(d_s) \setminus \{v_s\}$
    **if** $P2D_{\sqsubset}(np) = \{d_s\}$ **and** $np \neq v_t$ **then**
        remove $d_s$ from $doors_s$
$dist_m \leftarrow \infty$
**for** each door $d_s \in doors_s$ **do**
    $doors \leftarrow \emptyset$
    **for** each door $d_t \in doors_t$ **do**
        **if** $dist_V(p_s, d_s) + dist_V(p_t, d_t) < dist_m$ **then**
            add $d_t$ to $doors$
    initialize a min-heap $H$
    **for** each door $d_i \in \mathcal{S}_{door}$ **do**
        **if** $d_i \neq d_s$ **then**
            $dist[d_i] \leftarrow \infty$
        **else**
            $dist[d_i] \leftarrow 0$
        enheap($H, \langle d_i, dist[d_i] \rangle$)
    **while** $H$ is not empty **do**
        $\langle d_i, dist[d_i] \rangle \leftarrow$ deheap($H$)
        **if** $d_i \in doors$ **then**
            $doors \leftarrow doors \setminus \{d_i\}$
            **if** $dist_m > dist_V(p_s, d_s) + dist[d_i] + dist_V(p_t, d_i)$ **then**
                $dist_m \leftarrow dist_V(p_s, d_s) + dist[d_i] + dist_V(p_t, d_i)$

# References I

[1] C. S. Jensen, H. Lu, and B. Yang.
Graph model based indoor tracking.
In *MDM*, pp. 122–131, 2009.

[2] B. Yang, H. Lu, and C. S. Jensen.
Scalable continuous range monitoring of moving objects in symbolic indoor space.
In *CIKM*, pp. 671–680, 2009.

[3] B. Yang, H. Lu, and C. S. Jensen.
Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space.
In *EDBT*, pp. 335–346, 2010.

[4] H. Lu, B. Yang, and C. S. Jensen.
Spatio-temporal Joins on Symbolic Indoor Tracking Data.
In *ICDE*, pp. 816–827, 2011.

# References II

[5] C. S. Jensen, H. Lu and B. Yang.
Indoor-A New Data Management Frontier.
In *IEEE Data Eng. Bull.*, pp. 12–17, 2010.

[6] H. Lu, X. Cao, and C. S. Jensen.
A foundation for efficient indoor distance-aware query processing.
In *ICDE*, pp. 438–449, 2012.

[7] C. Becker and F. Dürr.
On location models for ubiquitous computing.
In *Personal and Ubiquitous Computing*, pp. 20–31, 2005.

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

[9] T. Becker, C. Nagel and T. H. Kolbe
A multilayered space-event model for navigation in indoor spaces.
In *3D Geo-Information Sciences*, pp. 61–77, 2009.

1 1. Outlines

2 2. Indoor Space Models & Applications

3 3. Indoor Data Cleansing

4 4. Indoor Movement Analysis

5 5. Appendix

1. Outlines

2. Indoor Space Models & Applications

3. Indoor Data Cleansing

4. Indoor Movement Analysis

5. Appendix

1 1. Outlines

2 2. Indoor Space Models & Applications

3 3. Indoor Data Cleansing

4 4. Indoor Movement Analysis

5 5. Appendix

# The End. Thanks :)