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

Huan Li

Database Laboratory, Zhejiang University

lihuancs@zju.edu.cn

March 15, 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. Outlines

2. Indoor Space Models & Applications

3. Indoor Data Cleansing

4. Indoor Movement Analysis

5. Appendix

Aims

- To give a brief review introduction to *indoor data management techniques*.
- To review a series of works in this field, including their proposed *models*, *indexes* and *algorithms*.
- To discuss how to bring those advanced theoretical contents into practice.

1. Outlines

2. Indoor Space Models & Applications

3. Indoor Data Cleansing

4. Indoor Movement Analysis

5. Appendix

About This Work...

Scalable Continuous Range Monitoring of Moving Objects in Symbolic Indoor Space. [3]
B. Yang, H. Lu, and C. S. Jensen.

- Published in *CIKM' 2009*.
- Application: continuously monitor indoor moving objects for space use analysis or security purposes.
- An incremental, query-aware continuous range query processing technique for objects moving in indoor space.
- Use maxmum-speed constraint on object movement to refine the uncertain results.

Motivation

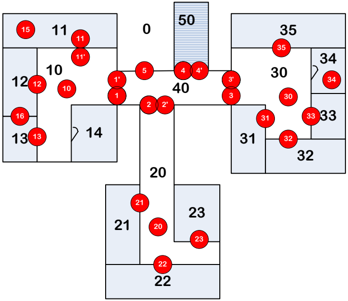
- People spend much time in indoor spaces.
- Indoor spaces are becoming increasingly larger and complex.
  - E.g., London Underground, 268 stations, 408 kilometers of network, +4 million daily passegers.
- Indoor monitoring of people can help support.
  - space use analysis
  - security purposes

Preliminaries: Indoors vs. Outdoors

- Modeling of indoor spaces do not assume
  - Euclidean space. (since obstacles render movement more constrained)
  - Spatial network. (since indoor movement is less constrained than movements in polylines)
- Instead indoor spaces are characterized by entities.
  - Doors, rooms, hallways, staircase, etc.
- **Symbolic models** are more suitable.
- *GPS* and *cellular tracking* do not work indoors.
- Sensing devices are used to detect objects within their activation range, e.g., RFID readers or Bluetooth hotspots.

Positioning Devices Deployment Graph

- Two types of positioning devices
  - Partitioning Device – *undirected* (**UP**), e.g., d_{21} – *directed* (**DP**), e.g., d_{11} and $d_{11'}$
  - Presence Device – (**PR**)
- Note an indoor space is partitioned into *activation ranges* and *cells*

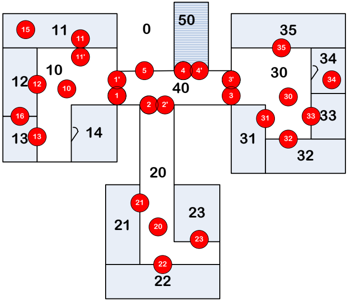


Deployment Graph

- $G = \{C, E, \Sigma_{devices}, l_E\}$
- C: the set of cells
- E: the set of edges, $\{c_i, c_j\}$ where $c_i, c_j \in C$
- $\Sigma_{devices}$: a mapping from $deviceID$ to activation range and type
- l_E maps an edge to a set of positioning devices, i.e., $E \to 2^{\Sigma_{devices}}$

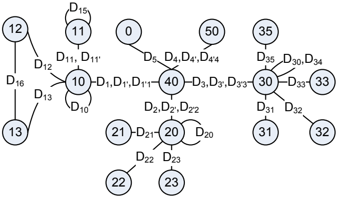
Positioning Devices Deployment Graph

- Two types of positioning devices
  - Partitioning Device – *undirected* (**UP**), e.g., d_{21} – *directed* (**DP**), e.g., d_{11} and $d_{11'}$
  - Presence Device – (**PR**)
- Note an indoor space is partitioned into *activation ranges* and *cells*

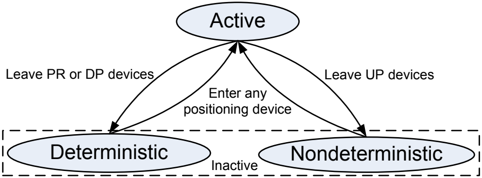


Deployment Graph

- $G = \{C, E, \Sigma_{devices}, l_E\}$
- C: the set of cells
- E: the set of edges, $\{c_i, c_j\}$ where $c_i, c_j \in C$
- $\Sigma_{devices}$: a mapping from $deviceID$ to activation range and type
- l_E maps an edge to a set of positioning devices, i.e., $E \to 2^{\Sigma_{devices}}$



States of Indoor Moving Objects



- An object is in an **active state** when it is inside the activation range of a positioning device.
- Otherwise the object is in an **inactive state**
- When an object is in the inactive state it is
  - **nondeterministic** if it can be in more than one cell
  - **deterministic** if it is in one specific cell

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

$$DHT[deviceID] = O_A;\ deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

$$DHT[deviceID] = O_A;\ deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Cell Deterministic Hash Table(CDHT) maps each cell to a set of deterministic objects:

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

$$DHT[deviceID] = O_A;\ deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Cell Deterministic Hash Table(CDHT) maps each cell to a set of deterministic objects:

$$CDHT[cellID] = O_D;\ cellID \in C, O_D \subseteq O_{indoor}$$

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

$$DHT[deviceID] = O_A;\ deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Cell Deterministic Hash Table(CDHT) maps each cell to a set of deterministic objects:

$$CDHT[cellID] = O_D;\ cellID \in C, O_D \subseteq O_{indoor}$$

Cell Nondeterministic Hash Table(CNHT) maps each cell to a set of nondeterministic objects:

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

$$DHT[deviceID] = O_A;\ deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Cell Deterministic Hash Table(CDHT) maps each cell to a set of deterministic objects:

$$CDHT[cellID] = O_D;\ cellID \in C, O_D \subseteq O_{indoor}$$

Cell Nondeterministic Hash Table(CNHT) maps each cell to a set of nondeterministic objects:

$$CNHT[cellID] = O_N;\ cellID \in C, O_N \subseteq O_{indoor}$$

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

$$DHT[deviceID] = O_A;\ deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Cell Deterministic Hash Table(CDHT) maps each cell to a set of deterministic objects:

$$CDHT[cellID] = O_D;\ cellID \in C, O_D \subseteq O_{indoor}$$

Cell Nondeterministic Hash Table(CNHT) maps each cell to a set of nondeterministic objects:

$$CNHT[cellID] = O_N;\ cellID \in C, O_N \subseteq O_{indoor}$$

Object Hash Table(OHT) maps objects to their current data(state, time, cell(s) the object can be in)

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

$$DHT[deviceID] = O_A;\ deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Cell Deterministic Hash Table(CDHT) maps each cell to a set of deterministic objects:

$$CDHT[cellID] = O_D;\ cellID \in C, O_D \subseteq O_{indoor}$$

Cell Nondeterministic Hash Table(CNHT) maps each cell to a set of nondeterministic objects:

$$CNHT[cellID] = O_N;\ cellID \in C, O_N \subseteq O_{indoor}$$

Object Hash Table(OHT) maps objects to their current data(state, time, cell(s) the object can be in)

$$OHT[objectID] = (STATE, t, IDSet);\ objectID \in O_{indoor}$$

Indexing Indoor Moving Objects

The proposed indexing scheme uses 4 hash tables

Device Hash Table(DHT) maps each device to a set of active objects:

$$DHT[deviceID] = O_A;\ deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Cell Deterministic Hash Table(CDHT) maps each cell to a set of deterministic objects:

$$CDHT[cellID] = O_D;\ cellID \in C, O_D \subseteq O_{indoor}$$

Cell Nondeterministic Hash Table(CNHT) maps each cell to a set of nondeterministic objects:

$$CNHT[cellID] = O_N;\ cellID \in C, O_N \subseteq O_{indoor}$$

Object Hash Table(OHT) maps objects to their current data(state, time, cell(s) the object can be in)

$$OHT[objectID] = (STATE, t, IDSet);\ objectID \in O_{indoor}$$

RFID Deployment Graph Construction

Algorithm 1 updateHashTables(Pre-processing output O, DeploymentGraph G)

```
1:  IDSet sSet ← ∅;
2:  if O.flag = ENTER then
3:      sSet ← OHT[O.objectID].IDSet;
4:      if OHT[O.objectID].STATE = Active then
5:          for the single element c in sSet do
6:              Delete O.objectID from DHT[c];
7:      else if OHT[O.objectID].STATE = Deterministic then
8:          for the single element c in sSet do
9:              Delete O.objectID from CDHT[c];
10:     else
11:         for each element c in sSet do
12:             Delete O.objectID from CNHT[c];
13:     Add O.objectID to DHT[O.deviceID];
14:     OHT[O.objectID] ← (Active, O.t, {O.deviceID});
15: else
16:     Delete O.objectID from DHT[O.deviceID];
17:     sSet ← G.ℓ_E^{-1}(O.deviceID);
18:     if Devices(O.deviceID).TYPE = UP then
19:         OHT[O.objectID] ← (Nondeterministic,O.t,sSet);
20:         for each element c in sSet do
21:             Add O.objectID to CNHT[c];
22:     else
23:         OHT[O.objectID] ← (Deterministic,O.t,sSet);
24:         for the single element c in sSet do
25:             Add O.objectID to CDHT[c];
```

1. Lines 1: reset $IDSet$

RFID Deployment Graph Construction

Algorithm 1 updateHashTables(Pre-processing output O, DeploymentGraph G)

1: IDSet $sSet \leftarrow \emptyset$;
2: **if** $O.flag$ = $ENTER$ **then**
3: $\quad sSet \leftarrow OHT[O.objectID].IDSet$;
4: $\quad$ **if** $OHT[O.objectID].STATE = Active$ **then**
5: $\quad\quad$ **for** the single element c in $sSet$ **do**
6: $\quad\quad\quad$ Delete $O.objectID$ from $DHT[c]$;
7: $\quad$ **else if** $OHT[O.objectID].STATE = Deterministic$ **then**
8: $\quad\quad$ **for** the single element c in $sSet$ **do**
9: $\quad\quad\quad$ Delete $O.objectID$ from $CDHT[c]$;
10: $\quad$ **else**
11: $\quad\quad$ **for** each element c in $sSet$ **do**
12: $\quad\quad\quad$ Delete $O.objectID$ from $CNHT[c]$;
13: $\quad$ Add $O.objectID$ to $DHT[O.deviceID]$;
14: $\quad OHT[O.objectID] \leftarrow (Active, O.t, \{O.deviceID\})$;
15: **else**
16: $\quad$ Delete $O.objectID$ from $DHT[O.deviceID]$;
17: $\quad sSet \leftarrow G.\ell_E^{-1}(O.deviceID)$;
18: $\quad$ **if** $Devices(O.deviceID).TYPE$ = UP **then**
19: $\quad\quad OHT[O.objectID] \leftarrow (Nondeterministic, O.t, sSet)$;
20: $\quad\quad$ **for** each element c in $sSet$ **do**
21: $\quad\quad\quad$ Add $O.objectID$ to $CNHT[c]$;
22: $\quad$ **else**
23: $\quad\quad OHT[O.objectID] \leftarrow (Deterministic, O.t, sSet)$;
24: $\quad\quad$ **for** the single element c in $sSet$ **do**
25: $\quad\quad\quad$ Add $O.objectID$ to $CDHT[c]$;

1. Lines 1: reset $IDSet$
2. Lines 2–12: $O.flag$ is ENTER so check the object's previous state. Remove O from the corresponding table according its previous state

RFID Deployment Graph Construction

Algorithm 1 **updateHashTables**(Pre-processing output O, DeploymentGraph G)

```
1:  IDSet sSet ← ∅;
2:  if O.flag = ENTER then
3:      sSet ← OHT[O.objectID].IDSet;
4:      if OHT[O.objectID].STATE = Active then
5:          for the single element c in sSet do
6:              Delete O.objectID from DHT[c];
7:      else if OHT[O.objectID].STATE = Deterministic then
8:          for the single element c in sSet do
9:              Delete O.objectID from CDHT[c];
10:     else
11:         for each element c in sSet do
12:             Delete O.objectID from CNHT[c];
13:     Add O.objectID to DHT[O.deviceID];
14:     OHT[O.objectID] ← (Active, O.t, {O.deviceID});
15: else
16:     Delete O.objectID from DHT[O.deviceID];
17:     sSet ← G.ℓ_E^{-1}(O.deviceID);
18:     if Devices(O.deviceID).TYPE = UP then
19:         OHT[O.objectID] ← (Nondeterministic,O.t,sSet);
20:         for each element c in sSet do
21:             Add O.objectID to CNHT[c];
22:     else
23:         OHT[O.objectID] ← (Deterministic,O.t,sSet);
24:         for the single element c in sSet do
25:             Add O.objectID to CDHT[c];
```

1. Lines 1: reset $IDSet$
2. Lines 2–12: $O.flag$ is ENTER so check the object's previous state. Remove O from the corresponding table according its previous state
3. Lines 13–14: add O to table of active objects (DHT), and update O's in the objects' table (OHT)

RFID Deployment Graph Construction

Algorithm 1 updateHashTables(Pre-processing output O, DeploymentGraph G)

```
1:  IDSet sSet ← ∅;
2:  if O.flag = ENTER then
3:      sSet ← OHT[O.objectID].IDSet;
4:      if OHT[O.objectID].STATE = Active then
5:          for the single element c in sSet do
6:              Delete O.objectID from DHT[c];
7:      else if OHT[O.objectID].STATE = Deterministic then
8:          for the single element c in sSet do
9:              Delete O.objectID from CDHT[c];
10:     else
11:         for each element c in sSet do
12:             Delete O.objectID from CNHT[c];
13:     Add O.objectID to DHT[O.deviceID];
14:     OHT[O.objectID] ← (Active, O.t, {O.deviceID});
15: else
16:     Delete O.objectID from DHT[O.deviceID];
17:     sSet ← G.ℓ_E^{-1}(O.deviceID);
18:     if Devices(O.deviceID).TYPE = UP then
19:         OHT[O.objectID] ← (Nondeterministic,O.t,sSet);
20:         for each element c in sSet do
21:             Add O.objectID to CNHT[c];
22:     else
23:         OHT[O.objectID] ← (Deterministic,O.t,sSet);
24:         for the single element c in sSet do
25:             Add O.objectID to CDHT[c];
```

1. Lines 1: reset $IDSet$
2. Lines 2–12: $O.flag$ is ENTER so check the object's previous state. Remove O from the corresponding table according its previous state
3. Lines 13–14: add O to table of active objects (DHT), and update O's in the objects' table (OHT)
4. Lines 16–17: $O.flag$ is LEAVE so remove the object from DHT. Get the possible cells that O can move to

RFID Deployment Graph Construction

Algorithm 1 updateHashTables(Pre-processing output O, DeploymentGraph G)

1: IDSet $sSet \leftarrow \emptyset$;
2: **if** $O.flag$ = $ENTER$ **then**
3: $\quad sSet \leftarrow OHT[O.objectID].IDSet$;
4: $\quad$ **if** $OHT[O.objectID].STATE$ = $Active$ **then**
5: $\quad\quad$ **for** the single element c in $sSet$ **do**
6: $\quad\quad\quad$ Delete $O.objectID$ from $DHT[c]$;
7: $\quad$ **else if** $OHT[O.objectID].STATE$ = $Deterministic$ **then**
8: $\quad\quad$ **for** the single element c in $sSet$ **do**
9: $\quad\quad\quad$ Delete $O.objectID$ from $CDHT[c]$;
10: $\quad$ **else**
11: $\quad\quad$ **for** each element c in $sSet$ **do**
12: $\quad\quad\quad$ Delete $O.objectID$ from $CNHT[c]$;
13: $\quad$ Add $O.objectID$ to $DHT[O.deviceID]$;
14: $\quad OHT[O.objectID] \leftarrow (Active, O.t, \{O.deviceID\})$;
15: **else**
16: $\quad$ Delete $O.objectID$ from $DHT[O.deviceID]$;
17: $\quad sSet \leftarrow G.\ell_E^{-1}(O.deviceID)$;
18: $\quad$ **if** $Devices(O.deviceID).TYPE$ = UP **then**
19: $\quad\quad OHT[O.objectID] \leftarrow (Nondeterministic, O.t, sSet)$;
20: $\quad\quad$ **for** each element c in $sSet$ **do**
21: $\quad\quad\quad$ Add $O.objectID$ to $CNHT[c]$;
22: $\quad$ **else**
23: $\quad\quad OHT[O.objectID] \leftarrow (Deterministic, O.t, sSet)$;
24: $\quad\quad$ **for** the single element c in $sSet$ **do**
25: $\quad\quad\quad$ Add $O.objectID$ to $CDHT[c]$;

1. Lines 1: reset $IDSet$
2. Lines 2–12: $O.flag$ is ENTER so check the object's previous state. Remove O from the corresponding table according its previous state
3. Lines 13–14: add O to table of active objects (DHT), and update O's in the objects' table (OHT)
4. Lines 16–17: $O.flag$ is LEAVE so remove the object from DHT. Get the possible cells that O can move to
5. Lines 18–25: if the device is undirected, set O in OHT and add O to CNHT for the cells in sSet, else apply the same to CDHT

Continuous Range Monitoring: Query Definition

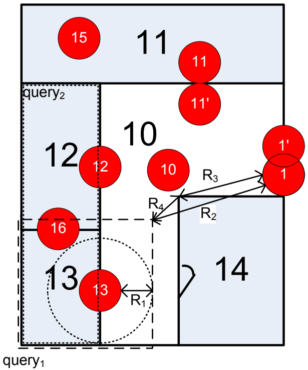
- A *Continuous Range Monitoring Query* (CRMQ)
  - takes an **indoor spatial range** R as parameter
  - keeps reporting the objects when it is registered for a certain time frame $[t_s, t_e]$
- The **query result** $\mathcal{M}$ – the set of moving objects in R - is maintained as follows:

  $$\forall t \in [t_s, t_e] : o \in CRMQ[R](\mathcal{M}) \Leftrightarrow o \in \mathcal{M} \wedge pos_{\mathcal{M}}(o, t) \in R$$

  where $pos_{\mathcal{M}}$ is a function that can determine the position of object o at time t
- Multiple monitoring queries may coexist

Critical Devices

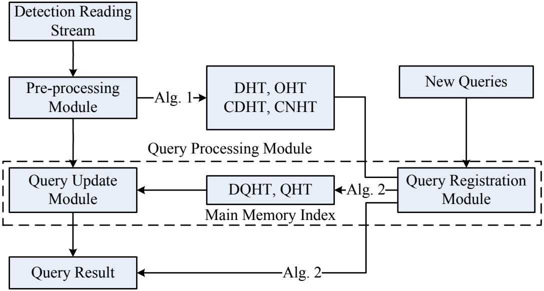
For a CRMQ query, a *critical device* is one from which a new observation can potentially change the query result (either certain or uncertain)



- CLASS1 – Device is fully covered in R along with cells, e.g., $(device_{16}, query_2)$
- CLASS2 – Device is fully covered but corresponding cells are not, e.g., $(device_{13}, query_1)$
- CLASS3 – Device intersects with the query range R, e.g., $(device_{16}, query_1)$
- CLASS4 – Device is disjoint from R and at least one of its corresponding cells in $C_{ic} = \{c | c \sqcap R \neq \varnothing\}$, e.g., $(device_1, query_1)$
- CLASS5 – Device is disjoint from R and at least one of its corresponding cells in $C_{ex} = \{c | \{c, c'\} \in G.E, c' \in C_{ic}\}$, but none of them are in C_{ic}, e.g., $(device_{10}, query_2)$

Critical Devices

- To handle concurrent CRMQs, a *Query Hash Table* is created hold the results
  - $QHT[queryID] = (CR, UR);\ CR \subseteq O_{indoor}, UR \subseteq O_{indoor}$
  - where CR is the certain result and UR is the uncertain result
- Overview



1. Outlines

2. Indoor Space Models & Applications

3. Indoor Data Cleansing

4. Indoor Movement Analysis

5. Appendix

1. Outlines

2. Indoor Space Models & Applications

3. Indoor Data Cleansing

4. Indoor Movement Analysis

5. Appendix

1. Outlines

2. Indoor Space Models & Applications

3. Indoor Data Cleansing

4. Indoor Movement Analysis

5. Appendix

References

M. F. Worboys.
Modeling indoor space.
In *ISA*, pp. 1–6, 2011.

X. Xie, H. Lu, and T. B. Pedersen.
Efficient distance-aware query evaluation on indoor moving objects.
In *ICDE*, pp. 434–445, 2013.

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.

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.

References

- H. Lu, X. Cao, and C. S. Jensen.
  A foundation for efficient indoor distance-aware query processing.
  In *ICDE*, pp. 438–449, 2012.
- C. S. Jensen, H. Lu, and B. Yang.
  Graph model based indoor tracking.
  In *MDM*, pp. 122–131, 2009.
- H. Lu, B. Yang, and C. S. Jensen.
  Spatio-temporal Joins on Symbolic Indoor Tracking Data.
  In *ICDE*, pp. 816–827, 2011.

The End. Thanks :)