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

Huan Li

Database Laboratory, Zhejiang University

lihuancs@zju.edu.cn

March 24, 2016

Overview

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

About This Work...

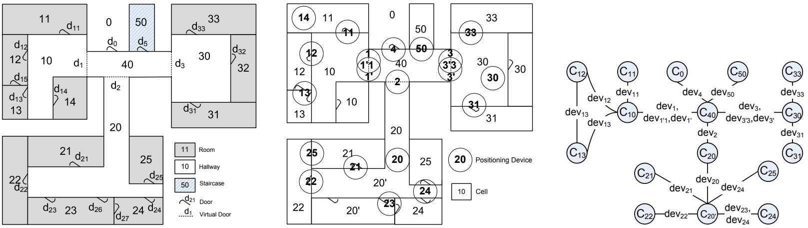
Spatio-temporal Joins on Symbolic Indoor Tracking Data. [4]
H. Lu, B. Yang, and C. S. Jensen.

- Published at *ICDE' 2011*.
- Studies the probabilistic, spatio-temporal joins on historical indoor tracking data.
- Two-phase hash-based algorithms are proposed for the point and interval joins.
- A filter-and-refine framework, along with spatial indexes and pruning rules.

Motivation

- Huge amount of tracking data serves as a foundation for a wide variety of indoor applications and services.[5]
  - shopping mall, airports, office buildings, akin to those enabled by outdoor GPS
  - hot area detection, space planning, security control, movement pattern discovery
- Spatio-temporal joins fall short in indoor setting.
  - indoor space consists of semantic entities enable or constrain movement
  - semantics of indoor space call for novel spatio-temporal join predicates
  - indoor positioning technologies differ fundamentally from outdoor setting, low accuracy and sampling frequency
- Joins on indoor tracking data call for new definition and new implementation techniques that take into account:
  - specifics of indoor space
  - limitations of indoor positioning

Preliminaries: Symbolic Indoor Tracking



(a) Floor Plan (b) Devices Deployment (c) Deployment Graph

1. $C2P : C \rightarrow 2^P$ maps a cell to a set of indoor partitions
2. $D2C : D \rightarrow 2^C$ maps a device to a set of corresponding cells
3. According to Deployment Graph, for partitioning device, $D2C(device_{13}) = \{C_{10}, C_{13}\} \cup \{C_{12}, C_{13}\} = \{C_{10}, C_{12}, C_{13}\}$
4. For presence device, $D2C(device_{25}) = \{C_{21}, C_{22}\}$ as the cells intersect its detection range.
5. $D2C : D \rightarrow 2^C$ is useful as it captures the possible movements of objects.

Preliminaries: Symbolic Indoor Tracking

TABLE I
OBJECT TRACKING TABLE (OTT)

ID	$objectID$	$deviceID$	t_s	t_e
rd_1	o_1	dev_4	t_1	t_2
rd_2	o_2	dev_4	t_1	t_2
rd_3	o_1	dev_2	t_5	t_6
rd_4	o_2	$dev_{1'}$	t_7	t_8
rd_5	o_1	dev_1	t_9	t_{10}
rd_6	o_1	dev_{12}	t_{15}	t_{16}
rd_7	o_2	dev_{13}	t_{20}	t_{21}
rd_8	o_1	dev_{13}	t_{21}	t_{22}
rd_9	o_2	dev_{13}	t_{29}	t_{30}
...	...	...	...	...

- **Object Tracking Table** OTT records the converted trajectories with schema $(ID, objectID, deviceID, t_s, t_e)$
- a record states that the object $objectID$ is observed by the device $deviceID$ in the closed interval from time t_s to t_e.

Problem Definitions

Given an OTT, it is of interesting to identify object pairs that join w.r.t some specific spatio-temporal join predicate.

- to know all pair of individuals that were probably at the same gate when a particular event (terrorist attack) occurred in a large airport.

Due to tracking uncertainty, only interested in those objects that satisfy the join predicate with some given probability (specified threshold).

The joins are effectively *self-joins* because all tracking data is contained in a single OTT.

Problem Definition I

One can apply a join predicate to a time point to find pairs that join at that particular time point...

Definition (Probabilistic Threshold Indoor Spatio-temporal Join–PTISSJ)

Given an OTT, a join predicate P, a time point t, and a threshold value $M \in (0, 1]$, a probabilistic threshold indoor spatio-temporal join $\bowtie_{P,t,M} (OTT) = \{(o_i, o_j) | o_i, o_j \in O \wedge o_i \neq o_j \wedge pr(P(o_i, o_j, t)) > M\}$, where $pr(P(o_i, o_j, t))$ is the **Timeslice Join Probability** of o_i, o_j at time t, i.e., the probability that predicate $P(o_i, o_j, t)$ is true.

Problem Definition II

It's also interesting to know object pairs satisfy the predicate for some consecutive timestamp...

Definition (Probabilistic Threshold k Indoor Spatio-temporal Join–PTkISSJ)

Given an OTT, a join predicate P, a time interval $I = [t_m, t_n](m < n)$, an integer $k(0 < k \leq n - m)$, and a threshold value $M \in (0, 1]$, a probabilistic k threshold indoor spatio-temporal join

$$\bowtie_{P,I,k,M} (OTT) = \{(o_i, o_j)|o_i, o_j \in O \wedge o_i \neq o_j \wedge$$
$$\exists s \in m...n - k + 1(\forall \delta \in 0...k - 1(pr(P(o_i, o_j, t_{s+\delta})) > M))\}$$

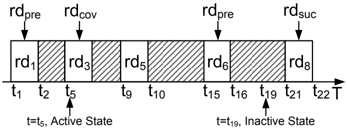
Uncertainty Model for Indoor Tracking

For outdoor moving objects [6], **Uncertainty Region**, denoted by $UR(o_i, t)$, is a region such that o_i must be in this region at time t.

In general terms, an object o_i's location can be modeled as a random variable l associated with a probability density function $f_{o_i}(l, t)$ that has non-zero values only in o_i's uncertainty region $UR(o_i, t)$. [3]

$$\int_{l \in UR(o_i,t)} f_{o_i}(l, t) dl = 1 \tag{1}$$

Object State in OTT



Definition (Active State)

Given an object o_i and a time point t, if a tracking record rd_{cov} is found in OTT such that $rd_{cov}.objectID = o_i$ and $t \in [rd_{cov}.t_s, rd_{cov}.t_e]$, o_i is in the **active state** at time t.

Definition (Inactive State)

Given an object o_i and a time point t, if no record rd_{cov} is found in OTT, o_i is in the **inactive state** at time t. Instead, two tracking records of o_i called rd_{pre} and rd_{suc}, can be found in OTT, such that they are consecutive in the sense that $rd_{pre}.t_e < t < rd_{suc}.t_s$ and there is no record for o_i between times $rd_{pre}.t_e$ and $rd_{suc}.t_s$.

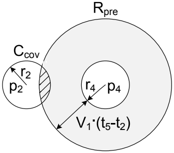
Uncertainty Region in the Active State

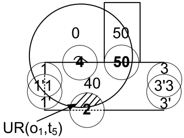
TABLE I
OBJECT TRACKING TABLE (OTT)

ID	$objectID$	$deviceID$	t_s	t_e
rd_1	o_1	dev_4	t_1	t_2
rd_2	o_2	dev_4	t_1	t_2
rd_3	o_1	dev_2	t_5	t_6
rd_4	o_2	$dev_{1'}$	t_7	t_8
rd_5	o_1	dev_1	t_9	t_{10}
rd_6	o_1	dev_{12}	t_{15}	t_{16}
rd_7	o_2	dev_{13}	t_{20}	t_{21}
rd_8	o_1	dev_{13}	t_{21}	t_{22}
rd_9	o_2	dev_{13}	t_{29}	t_{30}
...	...	...	...	...

Example

$t = t_5$, $rd_{cov} = rd_3$ and $rd_{pre} = rd_1$, which tells o_i left dev_4's detection range at time t_2, and is currently detected by dev_2.





Step 1: UR is the detection range of device $rd_{cov}.deviceID$, denote as:

$$C_{cov} = Cir(Loc(rd_{cov}.deviceID), Rad(rd_{cov}.deviceID))$$

Step 2: UR should consider the rd_{pre}'s *maximum speed bounding ring*(MSBR):

$$UR(o_i, t) = C_{cov} \cap Ring(Loc(rd_{pre}.deviceID), Rad(rd_{pre}.deviceID), V_i \cdot (t - rd_{pre}.t_e))$$

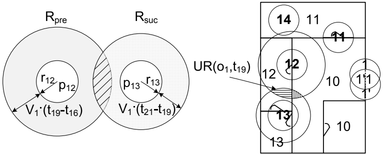
Uncertainty Region in the Inactive State

TABLE I
OBJECT TRACKING TABLE (OTT)

ID	$objectID$	$deviceID$	t_s	t_e
rd_1	o_1	dev_4	t_1	t_2
rd_2	o_2	dev_4	t_1	t_2
rd_3	o_1	dev_2	t_5	t_6
rd_4	o_2	$dev_{1'}$	t_7	t_8
rd_5	o_1	dev_1	t_9	t_{10}
rd_6	o_1	dev_{12}	t_{15}	t_{16}
rd_7	o_2	dev_{13}	t_{20}	t_{21}
rd_8	o_1	dev_{13}	t_{21}	t_{22}
rd_9	o_2	dev_{13}	t_{29}	t_{30}
...	...	...	...	...

Example

$t = t_{19}$, $rd_{pre} = rd_6$ and $rd_{suc} = rd_8$, since $rd_6.t_e = t_{16} < t_{19} < rd_8.t_s = t_{21}$. we have $dev_p = dev_{12}$ and $dev_s = dev_{13}$



Step 1: Determine the possible cells in which the object can be in the inactive period:

$$Cells_{mid} = D2C(dev_p) \cup D2C(dev_s)$$

Step 2: UR is constrained by two *maximum speed bounding ring*(MSBR)s of rd_{pre} and rd_{suc}:

$$UR(o_i, t) = \bigcup_{c \in Cells_{mid}} c \cap R_{pre} \cap R_{suc}$$

Join Probability Evaluation

Definition (the *same* X predicate)

termed as P_X, where X represents an indoor region type. IR_X represents all X type regions (X-regions).

Example (the *same room* predicate)

Given two objects o_i, o_j at a time point t, the *same room* predicate $P_X(o_i, o_j, t)$ evaluates to true if both o_i, o_j were located in a same room $rm \in IR_X$. Other predicates can be *same floor, same research group (maps to several rooms)*.

The *same* X predicates are more practical than Euclidean distance based join predicates in indoor space.

Join Probability Evaluation

Definition ("be located at" predicate probability)

Given an object o_i, an X-region $x_l \in IR_X$, and a time t, predicate $\Theta(o_i, x_l, t)$ indicate that o_i was located in x_l at t. The probability that Θ is satisfied is defined as:

$$pr(\Theta(o_i, x_l, t)) = \frac{Area(UR(o_i, t) \cap x_l)}{Area(UR(o_i, t))}$$

Definition (the *same* X predicate probability)

The probability that o_i and o_j were located in the same x_l at t, indicated by $pr(P_{x_l}(o_i, o_j, t))$ is defined as:

$$pr(P_{x_l}(o_i, o_j, t)) = pr(\Theta(o_i, x_l, t)) \cdot pr(\Theta(o_j, x_l, t))$$

Therefore, the probability that o_i and o_j satisfy a *same* X predicate at time t can be defined as:

$$pr(P_X(o_i, o_j, t)) = \max_{x_l \in IR_X} pr(P_{x_l}(o_i, o_j, t))$$

Indexing the Indoor Tracking Data

to determine the *Uncertainty Region* during join processing, it needs to retrieve the records rd_{cov} and rd_{pre} for active objects or rd_{pre} and rd_{suc} for inactive state.

to index OTT with an augmented 1D R-tree, where each leaf entry has the form $(t^{\vdash}, t^{\dashv}, Ptr_p, Ptr_c)$. $t^{\vdash} = rd_p.t_e$, $t^{\dashv} = rd_c.t_e$, Ptr_p and Ptr_c points to rd_p and rd_c respectively.

- if $t \geq rd_c.t_s$, o_i is active, $rd_p \rightarrow rd_{pre}$ and $rd_c \rightarrow rd_{cov}$;
- if $t < rd_c.t_s$, o_i is inactive, $rd_p \rightarrow rd_{pre}$ and $rd_c \rightarrow rd_{suc}$;

Accessing X-Regions

object locations are bounded by either device detection ranges or cells.

Algorithm 1 CD2XMappingInit(XRegionSet X)

Initialize $CovD2X$, $IntD2X$, $CovC2X$, $IntC2X$;
RTree $rt \leftarrow \emptyset$;
for each device dev in D **do**
  Add $Cir(Loc(dev), Rad(dev))$ into rt;
for each cell c in C **do**
  Add the spatial extent of c into rt;
for each X-region x in X **do**
  ResultSet $covR \leftarrow$ Search(rt, x, $COVER$);
  for each item a in $covR$ **do**
    if a indicates device dev's detection range **then**
      $CovD2X[dev] \leftarrow x$;
    else if a indicates cell c's detection range **then**
      $CovC2X[c] \leftarrow x$;
  ResultSet $intR \leftarrow$ Search(rt, x, $INTERSECT$);
  for each item a in $(intR \setminus covR)$ **do**
    if a indicates device dev's detection range **then**
      $IntD2X[dev] \leftarrow IntD2X[dev] \cup \{x\}$;
    else if a indicates cell c's detection range **then**
      $IntC2X[c] \leftarrow IntC2X[c] \cup \{x\}$;

- $CovD2X : D \rightarrow IR_X$ maps a device to an X-Region that fully covers the device's detection range;
- $IntD2X : D \rightarrow IR_X$ maps a device to an X-Region that only intersects the device's detection range;
- $CovC2X : C \rightarrow IR_X$ maps a cell to an X-Region that fully covers this cell;
- $IntC2X : C \rightarrow IR_X$ maps a cell to an X-Region that only intersects with;

Processing PTISSJ Queries: Partitioning Phase

Algorithm 2 PTISSJ_Part(A1Rtree *tree*, Timestamp t, Threshold M)

LeafEntrySet $leR \leftarrow tree$.RangeQuery(t);
HashTable $XRegionHT1 \leftarrow \emptyset$
for each leaf entry le in leR **do**
  OTTTuple $rd1 \leftarrow OTT[le.Ptr_p]$, $rd2 \leftarrow OTT[le.Ptr_c]$;
  DeviceID $dev1 \leftarrow rd1.deviceID$, $dev2 \leftarrow rd2.deviceID$;
  ObjectID $o \leftarrow rd1.objectID$;
  if $t \geq rd1.t_s$ **then**
    if $CovD2X$ is not null **then**
      XRegion $x \leftarrow CovD2X(dev1)$;
      $XRegionHT1[x] \leftarrow \{(o, 1.0)\} \cup XRegionHT1[x]$;
    else
      for each XRegion x in $IntD2X(dev1)$ **do**
        double $p \leftarrow pr(\Theta(o, x, t))$;
        if $p > M$ **then**
          $XRegionHT1[x] \leftarrow \{(o, p)\} \cup XRegionHT1[x]$;
  else
    Boolean $flag \leftarrow true$;
    CellSet $CSet \leftarrow D2C(dev1) \cap D2C(dev2)$;
    if $|CSet|=1$ **then**
      Cell $c \leftarrow$ the singleton element of $CSet$;
      if $CovC2X(c)$ is not null **then**
        XRegion $x \leftarrow CovC2X(c)$;
        $XRegionHT1[x] \leftarrow \{(o, 1.0)\} \cup XRegionHT1[x]$;
        $flag \leftarrow false$;
    if $flag$ **then**
      for each cell c in $CSet$ **do**
        for each XRegion x in $CovC2X(c) \cup IntC2X(c)$ **do**
          double $p \leftarrow pr(\Theta(o, x, t))$;
          if $p > M$ **then**
            $XRegionHT1[x] \leftarrow \{(o, p)\} \cup XRegionHT1[x]$;
return $XRegionHT1$;

- all indoor objects are partitioned into buckets that each refers to a distinct X-region
- first, A1R-tree is searched to get all leaf entries whose interval $(t^{\vdash}, t^{\dashv}]$ contains the join time t
- second, the spatial examination obtains all relevant X-region in which o_i may be at time t
- the relevant probabilities are evaluated for each object, and the necessary records are generated and added to relevant buckets, for each $p_l = pr(\Theta(o_i, x_l, t))$, if it is larger than threshold M, insert the record into buckets.

Processing PTISSJ Queries: Partitioning Phase

Active State

object o_i must be in device dev's detection range at time t.

1. if the detection range is fully covered by an X-region x_l, as indicated by $CovD2X(dev_c) = x_l$, a record $(o_i, 1.0)$ is added to x_l's bucket;
2. otherwise, dev_c's detection range intersects with each X-region in $CovD2X(dev_c)$, evaluated the probability, if it is larger than M, add to the bucket.

Inactive State

object o_i must be in a cell in $Cells_{mid} = D2C(dev_p) \cap D2C(dev_c)$.

1. if $Cells_{mid}$ is the singleton set and the cell is covered by one X-region x_l, indicated by $CovC2X(c) = x_l$, a record $(o_i, 1.0)$ is added to x_l's bucket;
2. otherwise, the single cell c in $Cells_{mid}$ intersects with several X-regions (indicated by $CovC2X(c)$), or $Cells_{mid}$ contains several cells.

Processing PTISSJ Queries: Join Phase

Algorithm 3 **PTISSJ_Join**(double M, HashTable $XRegionHT1$, XRegionSet X)

```
ObjectPairSet R(objectID1, objectID2) ← ∅;
for each X-region x in X do
    RecordSet RR(objectID, prob) ← XRegionHT1[x];
    if |RR| > 1 then
        for i=0; i<|RR|-1; i++ do
            for j=i+1; j<|RR|; j++ do
                if RR[i].prob · RR[j].prob > M then
                    R ← R ∪ (RR[i].objectID, RR[j].objectID);
return R;
```

- the records in the same bucket indicate all those objects that may join with other objects in the corresponding X-region with probabilities greater than M
- the join probability is evaluated according to the equation:

$$pr(P_X(o_i, o_j, t)) = \max_{x_l \in IR_X} pr(P_{x_l}(o_i, o_j, t))$$

Processing PTkISSJ Queries: Partitioning Phase

Algorithm 4 PTkISSJ_Part(A1Rtree $tree$, TimeInterval $[t_m, t_n]$, Integer k)

LeafEntrySet $leR \leftarrow tree$.RangeQuery($[t_m, t_n]$);
HashTable $XRegionHT2 \leftarrow \emptyset$
for each leaf entry le in leR **do**
  OTTTuple $rd1 \leftarrow OTT[le.Ptr_p]$, $rd2 \leftarrow OTT[le.Ptr_c]$;
  DeviceID $dev1 \leftarrow rd1.deviceID$, $dev2 \leftarrow rd2.deviceID$;
  ObjectID $o \leftarrow rd1.objectID$;
  CellSet $CSet \leftarrow D2C(dev1) \cap D2C(dev2)$;
  for each cell c in $CSet$ **do**
    for each X-region x in $CovC2X(c) \cup IntC2X(c)$ **do**
      TimeInterval $tI \leftarrow [t_m, t_n] \cap [le.t^{\vdash}, rd1.t_s]$;
      if tI is not null **then**
        $XRegionHT2[x] \leftarrow \{(o, tI)\} \cup XRegionHT2[x]$;
  for each X-region x in $CovD2X(dev1) \cup IntD2X(dev1)$ **do**
    TimeInterval $tI \leftarrow [t_m, t_n] \cap [rd1.t_s, le.t^{\dashv}]$];
    if tI is not null **then**
      $XRegionHT2[x] \leftarrow \{(o, tI)\} \cup XRegionHT2[x]$;

- first, A1R-tree is searched to get all leaf entries whose interval $(t^{\vdash}, t^{\dashv}]$ contains the join interval I.
- second, the spatial examination is conducted on each retrieved leaf entry to obtain all relevant X-regions for each object.
- after the spatial examination, the bucket of each X-region x_l involved contains a set of records of the form $(objectID, [t_x, t_y])$.

Processing PTkISSJ Queries: Join Phase

Pruning Rule 2

A record $(objectID, [t_x, t_y])$ is pruned if $t_y - t_x + 1 < k$.

Pruning Rule 3

If a bucket contains only one record, or all its records involve only one object, the bucket is pruned.

Pruning Rule 4

Two records $(objectID_1, [t_x, t_y])$ and $(objectID_2, [t_u, t_v])$ do not satisfy the join predicate if $|[t_x, t_y] \cap [t_u, t_v]| < k$.

Processing PTkISSJ Queries: Join Phase

Algorithm 5 PTkISSJ_Join(double M, HashTable $XRegionHT2$, XRegionSet X, Integer k)

```
ObjectPairSet R(objectID1, objectID2) ← ∅;
for each X-region x in X do
    RecordSet RR(objectID, tI) ← XRegionHT2[x];
    Combine records with same object and adjacent time intervals in RR;
    Remove any record if it satisfies tI.t_y − tI.t_x + 1 < k;
    if |RR| > 1 then
        Sort the records in RR according to its tI.t_x;
        for i=0; i<|RR|-1; i++ do
            ObjectID o_i ← RR[i].objectID;
            for j=i+1; j<|RR|; j++ do
                ObjectID o_j ← RR[j].objectID;
                if o_i ≠ o_j and (o_i, o_j) ∉ R and |RR[i].tI ∩ RR[j].tI|≥k then
                    TimeInterval [t_u, t_v] ← RR[i].tI ∩ RR[j].tI;
                    Integer count ← 0;
                    for t_c=t_u; t_c<t_v; t_c++ do
                        if pr(Θ(o_i, x, t_c)) · pr(Θ(o_j, x, t_c)) > M then
                            if count = k − 1 then
                                R ← R ∪ {(o_i, o_j)};
                                break;
                            else
                                count++;
                        else
                            if t_v − t_c + 1 ≤ k then
                                break;
                            count ← 0;
return R;
```

- first, records with as same $objectID$ and adjacent time intervals are merged into one records, after merging, pruning is done on the bucket according to the above-mentioned rules.
- next, the remaining records in X-region buckets are processed with relevant probabilities evaluated.
- *the plane sweep* [7], only if two object's record pair have their overlapping time interval contain at least k timestamps, the probability is evaluated.
- if two objects have been found to join in one bucket, skip all record pairs involving the two in all remaining buckets.
- skip the remaining evaluation, when the calculated probability (after k time points) has already greater than M, even the whole time interval is not finished.
- stop the remaining evaluation, when the calculated probability has already less than M, even the whole time interval is not finished.

Processing PTkISSJ Queries: Join Phase

- it's of interest to define join predicates beyond the $same\ X$ predicates, available non-spatial properties of objects may be used to define alternative join types.
- it's also relevant to use more complex PDFs for indoor moving objects, as well as to take into account possible dependencies between objects.
- more general types of queries, e.g., range queries can be defined for indoor moving objects w.r.t. uncertainty analysis.

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] R. Cheng, D. V. Kalashnikov and S.Prabhakar.
Querying imprecise data in moving object environments.
In *TKDE*, pp. 1112–1127, 2004.

[7] D. Pfoser and C. S. Jensen.
Capturing the uncertainty of moving-object representations
In *Advances in Spatial Databases*, pp. 111–131, 1999.

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 :)