Capturing Hotspots For Constrained Indoor Movement

Ahmed, Tanvir; Pedersen, Torben Bach; Lu, Hua

Published in:
21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2013)

DOI (link to publication from Publisher):
10.1145/2525314.2525463

Publication date:
2013

Document Version
Accepted author manuscript, peer reviewed version

Citation for published version (APA):
Ahmed, T., Pedersen, T. B., & Lu, H. (2013). Capturing Hotspots For Constrained Indoor Movement. In *21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2013)* (pp. 462-465). Association for Computing Machinery. https://doi.org/10.1145/2525314.2525463

Capturing Hotspots for Constrained Indoor Movement

Tanvir Ahmed, Torben Bach Pedersen, Hua Lu
Department of Computer Science
Aalborg University, Denmark
{tanvir, tbp, luhua}@cs.aau.dk

ABSTRACT

Finding the hotspots in large indoor spaces is very important for getting overloaded locations, security, crowd management, indoor navigation and guidance. The tracking data coming from indoor tracking are huge in volume and not readily available for finding hotspots. This paper presents a graph-based model for constrained indoor movement that can map the tracking records into mapping records which represent the entry and exit times of an object in a particular location. Then it discusses the hotspots extraction technique from the mapping records.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph labeling; H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Algorithms, Design, Reliability, Theory

Keywords

Indoor tracking, graph based model, RFID, moving objects

1. INTRODUCTION

Technologies like RFID, Bluetooth, etc., enable a variety of indoor tracking applications like people's movement tracking in large indoor space (e.g., airport, shopping mall, museum, etc.), airport baggage tracking, items movement tracking in supply chain system, etc. The huge amount of tracking data generated by these types of systems is very useful for analyzing and decision making. Detection of hotspots in an indoor space like airport baggage tracking will help the authority to manage the overloaded locations of the baggage handling and handle the bags efficiently. In case of airport people movement, detection of hotspots will give the idea about where and when most of the peoples generally gather that can help the authority to manage the crowd and for business it can be a good idea for different location-based services.

It is unsuitable for indoor trajectories to use the geometric polyline representation that is used for outdoor trajectories. For example if an object moves from one room to another then we will get two consecutive tracking records which represent the object location in different rooms. But due to the drawback of indoor positioning technologies, the locations between these two records are not obtained. As a result, it is not easily available when an object enters and exits a particular location. Thus, it is also not easily available how dense a location is. We take all of these complexities into consideration and propose an approach for extracting hotspots from indoor tracking data. To the best of our knowledge, this is the first paper to consider how to capture hotspots from indoor tracking data with constrained object movement.

Indoor space modeling for tracking of moving objects has been proposed in [4,6]. We propose a graph based model which is highly motivated by the model proposed in [4]. Their model converts the raw RFID readings into tracking records containing the first and last time of an object appeared within a reader's activation range. In our previous work [1], we converted the tracking records into stay records containing the transition time between readers. In the present paper, the tracking records are converted into mapping records showing when an object actually entered and exited the corresponding location. There are many works available for online density queries and hot route queries on road networks [2,3,5]. However, the scenario of symbolic indoor tracking is different from outdoor tracking as the geometric position of the object is not available in the indoor setting.

The remainder of the paper is organized as follows. Section 2 discusses the problem formulation. Section 3 describes the mapping of tracking records for semantic locations with graph-based model. Section 4 presents the hotspot queries. Finally, Section 5 concludes the paper and discusses possible future research.

2. PROBLEM FORMULATION

Problem Scenario. We assume a setting where the paths between the locations are constrained and objects are continuously moving from one location to another. We call such location as constrained path (CP) symbolic location. The objects cannot move freely and the locations are in some sense one dimensional. The size of a CP symbolic location is measured by length not by area. Fig. 1 shows an example of a CP, which is a conveyor of an airport baggage handling system. The conveyor is divided into different symbolic locations like check-in 1, check-in 2, screening, sorter-1, sorter-2 and chutes. More detail about the baggage tracking process can be found in [1]. In our setting, the tracking devices are strategically deployed at different fixed locations inside the indoor space, e.g., each section of conveyor belts. The objects contain tags or devices which can be tracked by the tracking devices. For example, in case of RFID technology, the tracking devices are RFID readers and the objects contain RFID tags. Different tracking devices have different sensing ranges. After deployment of the track-

SIGSPATIAL'13, Nov 05-08 2013, Orlando, FL, USA
ACM 978-1-4503-2521-9/13/11.
http://dx.doi.org/10.1145/2525314.2525463 .

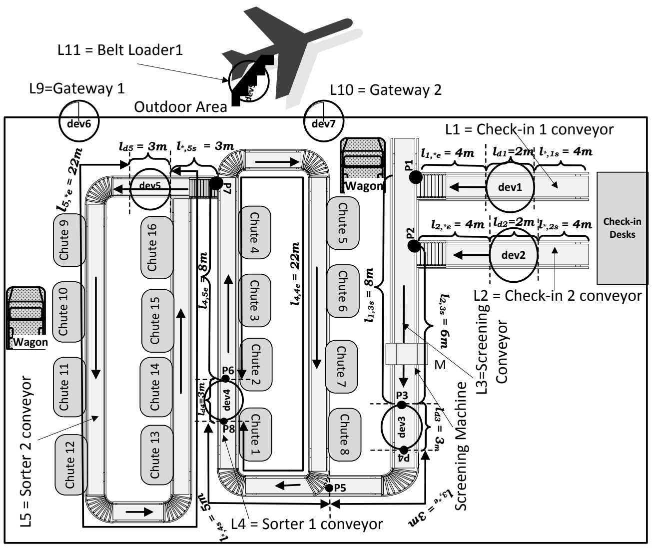


Figure 1: Constrained Path in airport baggage management

ing devices, their positions are recorded in the database. In Fig. 1 the circles represent the deployment of the RFID readers and their tracking ranges. When an object comes under a tracking device's activation range, it is continuously detected by the tracking device with a sampling rate and it generates raw reading records with the form: *(trackingDeviceID, ObjectID, t)*. It means that a tracking device *trackingDeviceID* detects a moving object *ObjectID* in its activation range at timestamp *t*. A *TrackingRecord(recordID, ObjectID, TrackingDeviceID, t_{in}, t_{out})* table [1] is constructed from the raw tracking sequence, where *recordID* is tracking record identifier and t_{in}, t_{out} respectively represent the timestamps of first reading and last reading of *ObjectID* by *TrackingDeviceID* in its activation range. An example of a table containing tracking records of an object $o1$ from Fig. 1 is shown in Table 1. In this table the record rec_1 means that object o_1 is observed by tracking device dev_1 from time 4 to 5, and record rec_3 means that o_1 is observed by dev_3 from time 15 to 18. Due to the limitation of indoor positioning systems, it is unknown what position of o_1 is between 6 and 14 without knowing the floor plan.

Table 1: Tracking Records of Indoor Moving Objects

RecordID	ObjectID	TrackingDeviceID	t$_{in}$	t$_{out}$
rec1	*o1*	*dev1*	*4*	*5*
rec3	*o1*	*dev3*	*15*	*18*
rec5	*o1*	*dev4*	*26*	*29*
rec8	*o1*	*dev4*	*51*	*54*
...	...	...	...	...

Problem Definition. Let L be the set of all symbolic locations inside a large indoor space, $L = \{l_1, l_2, l_3, \ldots , l_k\}$. The *capacity* of location l_i is denoted by c_i = *capacity*(l_i). The *capacity* of a CP symbolic location is a function of *length*. For example, the *capacity* of *check-in 1* conveyor in Fig. 1 depends on its length.

Definition 1 (Capacity). The *capacity* of a location l_i is the numbers of objects that can be reside at l_i during a defined time unit.

For example, the *capacity* of *check-in 1* conveyor in Fig. 1 can be 20 objects per minute.

Definition 2 (Density). Let n_i be the number of distinct objects at location l_i during the time interval, $w = [t_{start}\ , t_{end}]$ and c_i = capacity(l_i) be the capacity of location l_i. Then *density* of location l_i for interval w is defined as,

$d_i = \frac{n_i}{\Delta t \times capacity(l_i)} \times 100\%$, where $\Delta t = t_{end} - t_{start}$.

From the definition we can see that, the value of *density* gives us how dense a location is as a percentage value.

Definition 3 (Hotspot). A location l_i can be considered as a *hotspot* for interval w if d_i exceeds a given threshold θ.

Definition 4 (Hotspot Query). Find all the *hotspots* $H \subseteq L$, for time interval w.

3. SEMANTIC LOCATION MAPPINGS

A tracking device covers a very small portion of a location. As a result it is not sufficient to know when an object actually entered (*time$_{start}$*) and exited (*time$_{end}$*) the corresponding location. So there must be a mapping strategy for retrieving such location and timing information.

Modeling Symbolic Locations. In our setting each symbolic location contains only one tracking device deployed in it. For example in Fig. 1 *check-in 1* is represented by *dev1*. After passing *dev1* and *dev3* when a bag goes to *sorter-1* it will be read by *dev4* and then it may go to *sorter-2* or *chute* or it may circulate within *sorter-1*. For mapping between tracking records and the semantic locations, a reader deployment graph (RDG) can be constructed from the indoor plan given in Fig 1. Relevant details about the concept of reader deployment graph can be found elsewhere [4]. Although an RDG is capable of mapping the location of an object from the tracking records, it does not provide sufficient information for mapping the *tracking* entry and exit time to the *actual* entry and exit time. For more precise entry time and exit time we extend the RDG with a more detailed model called the Extended Reader Deployment Graph (ERDG). For this, some definitions are needed:

Definition 5 (Covered distance). Given a path p and a tracking device d, the Covered distance (CD) is the length of the part of p that is covered by d's detection range. CD for a tracking device dev_i is denoted as l_{di}. For example in Fig. 1 $l_{d1} = 2m$ shows the CD of $dev1$ at $L1$.

Definition 6 (Entry lag distance). The *entry lag distance (ENLD)* from location L_x to L_y denoted as l_{x,y_s} is the distance from the ending point of L_x to the first reading point at L_y.

For example, consider Fig. 1. The journey of an object at location $L3$ can start from either points $P1$ or $P2$ depending on whether the object is coming from $L1$ or $L2$. While moving at $L3$ the object will be first tracked by *dev3* when it comes at point $P3$. Here the distance between the point $P1$ and $P3$ is the *Entry lag distance (ENLD)* which is denoted as $l_{1,3s}$ and similarly ENLD between $P2$ and $P3$ is denoted as $l_{2,3s}$. It can be seen that a location L_y can have many ENLDs depending how many locations end at L_y. In our running example $l_{1,3s} = 8m$ and $l_{2,3s} = 6m$. However we use a special notation $l_{*,ys}$, which indicates that the ENLD at L_y is same regardless of where an object is coming from. In our example $l_{*,4s} = 5m$ is the ENLD of location $L4$ from any location ended at $L4$.

Definition 7 (Exit lag distance). Conversely the *exit lag distance (EXLD)* from location L_x to L_y denoted as l_{x,y_e} is the distance from the last reading point at L_x to the exit point of L_x that leads to location L_y.

Similar to ENLD, let us consider Fig. 1. The journey of an object at location $L3$ ends when it passes the point $P5$ and reaches location $L4$. While traveling through $L3$ the object was last detected by $dev3$ when it was at point $P4$. Here the distance between P4 and P5 is the *Exit lag distance (EXLD)* of $L3$ which is denoted as $l_{3,4e}$. As $L3$ has only one destination, the EXLD of $L3$ is always same regardless of destination. So instead of using $l_{3,4e}$ we use $l_{3,*e}$ in this case. In our example the value of $l_{3,*e}$ is 3 meters. Similar to ENLD, a location can have many EXLDs. For example an object can leave location $L4$ by going to $L5$ through $P7$ or can circulate in $L4$ and leave within any point between $P6$ and $P8$. As a result

$L4$ has two EXLDs $l_{4,5e} = 8m$ and $l_{4,4e} = 22m$.

The ERDG is formally defined by a labeled directed graph *G = (L, E, T, lb_E)*:

1. *L* is the set of locations where each location is represented as a vertex in the graph. If a location does not contain any tracking device deployed in it then the corresponding location is labeled as a *virtual location* L_{v_x} where x is an integer.
2. *E* is the set of directed edges: $E = \{(l_i,l_j) \mid l_i, l_j \in L\}$.
3. *T* is a set of tuples of the form $\langle$D, Flag, L_{dx}, $\{L_s\}$, $\{L_e\}$ $\rangle$, where D is a tracking device, Flag indicates whether it is a CP or not, $\{L_{dx}\}$ is the CD of D, $\{L_s\}$ is a collection of ENLDs and L_e is a collection of EXLDs.
4. lb_E is a function lb_E: $E \rightarrow T$ that labels an edge by a tuple from T. An edge $(l_i,\ l_j) \in E$ is labeled by a tuple $T_{i,j}\langle d_k, l_{d_k}, l_{i,js}, l_{j,*e}\rangle \in T$ where d_k is a tracking device deployed at location l_j, l_{d_k} is the CD for d_k, $l_{i,js}$ is an ENLD and EXLD for all the out-going locations from l_j is shown as $l_{j,*e}$. An edge is labeled by a tuple $T_{v_x,j} \in T$ if virtual tracking device dev_{v_x} is assumed to be deployed at location L_{v_x}. Fig. 2 shows an example of the ERDG of the floor plan of Fig. 1. Let us consider *edge (L1,L3)* where the tuple $T_{1,3}$ is assigned. The content of the tuple is the tracking device $dev3$ which is deployed at $L3$, l_{d3} which is CD for $dev3$, $l_{1,3s}$ is the ENLD from $L1$ to $L3$ and $l3, *e$ is the EXLD from $L3$ to any next destination.

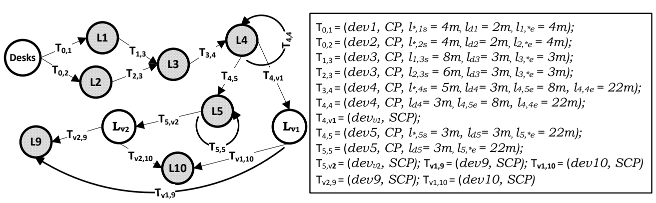
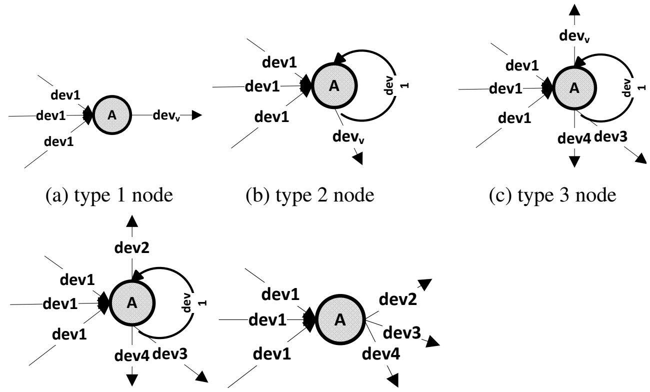


Figure 2: Extended Reader Deployment Graph (ERDG)

We define three mapping structures: location to device-In $L2DIn$: L $\rightarrow D$, location to device-Out $L2DOut$: L $\rightarrow 2^D$ and Device to Location $D2L$: D $\rightarrow L$, where L is the set of all locations and D is the set of all tracking devices. For a location l, $L2DIn(l)$ returns the tracking device deployed at l. From the graph it returns the tracking device which is labeled in any edge(s) where l is the destination. Since a CP location contains only one tracking device, all the incoming edges of a location will be labeled by same tracking device. On the other hand $L2DOut(l)$ returns all the tracking device(s) which are labeled in edge(s) where l is the source. These devices are deployed in the adjacent next locations of l. In the third mapping for a tracking device dev, $D2L(dev)$ returns the location of dev, that means the destination vertex of the edge that has dev in its label. In the running example of Fig. 2 $L2DOut(L4)$ = {$dev4$, $dev5$, dev_{v1}}, $L2DIn(L4)$ = $dev4$, and $D2L(dev4)$ = $L4$.

Mapping for CP Symbolic Locations. For mapping the *time$_{in}$* and *time$_{out}$* of an object *o* at a tracking device *dev* into the *time$_{start}$* and *time$_{end}$* of *o* at location l we use the topological information described in the ERDG in Fig. 2. However both of these values depend on the speed of o at l. We use Eq. (1), (2) and (3) for deriving the *speed*, *time$_{start}$* and *time$_{end}$* respectively. In all these equations *time$_{in}$* and *time$_{out}$* are taken from tracking records at *L2DIn(l)*. In Eq. (1) *CD (dev$_x$)* represents the *CD* of *L2DIn(l) = dev$_x$*. In Eq. (2) the *ENLD* depends on where the object is coming from and the value is taken from tuple $T_{prevLoc,l}$. In Eq. (3) the *EXLD* is taken from tuple $T_{l,nextLoc}$.

$$Speed := \frac{CD(dev_x)}{(time_{out} - time_{in})} \tag{1}$$



(a) type 1 node (b) type 2 node (c) type 3 node

(d) type 4 node (e) type 5 node

Figure 3: Types of Nodes in CP symbolic locations

$$time_{start} := time_{in} - \frac{\mathit{ENLD}}{\mathit{Speed}} \tag{2}$$

$$time_{end} := time_{out} + \frac{\mathit{EXLD}}{\mathit{Speed}} \tag{3}$$

For example, consider the second tracking record $\langle o_1$, *dev$_3$, 15, 18*$\rangle$ of Table 1. From the graph, CD of *dev3* = 3 meters and *D2L(dev3)* = L3. So the *speed* of o_1 at location $L3$ is : *speed* = $\frac{3}{18-15}$ = 1 meter/second (we assume the duration is measured in seconds). Similarly we can find the *time$_{start}$* of o_1 in *D2L(dev$_3$) = L3*. The previous tracking record says that the object o_1 was tracked at dev_1 before dev_3. So from ERDG we need to get the information from the edge, E(*D2L(dev$_1$) = L1*, *D2L(dev$_3$) = L3*). The ENLD from L1 to L3 is $l_{1,3s}$ = 8m. Now with the help of Eq. (1) and (2), the *time$_{start}$* = 15 - $\frac{8m}{3m/(18-15)}$ = 7.

Depending on the topological structure of the location, an object may have many $time_{out}$s from the same tracking device. For example $L4$ and $L5$ has loops where an object can circulate in the location which may results in multiple tracking records for same object from the devices $L2DIn(L4)$ and $L2DIn(L5)$. Based on the topological connectivity of a location we classified the nodes of the deployment graph into five types. Fig. 3 shows the five node types. Different types of nodes and the way of deriving the exit time of objects from that node is explained next.

Node Types. ***Node type 1*** contains only one outgoing edge and the outgoing edge is labeled by dev_{v_x}. A location l falls in *Node type 1* if $L2DOut(l)$ = {dev_{v_x}}. Fig. 3a shows an example of *Node type 1*. As the next location of this type of node has no tracking device deployed, it is certain that the object left the location through virtual tracking device dev_{v_x} which actually does not generate any tracking record. In Eq. (3) the $time_{out}$ of an object o_i at this type of location l_i is taken from the tracking record of o_i at $L2DIn(l_i)$ and EXLD $l_i, *e$ is taken from the tuple T_{L_{prev},l_i} of $edge(L_{prev}, l_i)$.

Node type 2 contains two outgoing edges. One outgoing edge is labeled by dev_{v_x} and another one is a loop. A location l falls in *Node type 2* if $L2DOut(l)$ = {$L2DIn(l)$, dev_{v_x}}. Fig. 3b shows an example of *Node type 2*. In our example, $L5$ is this type of node. Here an object can circulate within the location which generates multiple tracking records and at the end the object leaves the location through dev_{v_x}. The $time_{end}$ of the object is calculated using Eq. (3), where $time_{out}$ is taken and $speed$ is calculated from the last tracking record of the object from the tracking device of that location. Suppose an object $o2$ contains a single record from $dev5$: (o_2, $dev5$, 36, 39). It means that o_2 did not circulate at *D2L(dev5) = L5* and left the location to any one of the chutes. It is not possible to know when the object actually left $L5$. However we can get the maximum possible value of $time_{end}$ with the help of EXLD from the *edge (*, L5)* which is $l_{5,*e}$ = 22m and CD for $dev5$ = l_{d5} = 3m. So the $time_{end}$ = $\lceil 39 + \frac{22m}{3m/(39-36)} \rceil$ = 61.

Node type 3. In addition to the two outgoing edges like *Node type 2*, it has one or more edge(s) where destination locations have tracking devices deployed. A location l is considered to be *Node type 3* if $|L2DOut(l)| > 2$ and $\{L2DIn(l), dev_{v_x}\} \subset L2DOut(l)$. Here an object can circulate in the same location and it can leave the location through dev_{v_x} or other tracking devices. Fig. 3c shows an example of *Node type 3*. In our running example $L4$ falls in this type of node. As the object may circulate within the location we take *time*$_{out}$ in the similar way of *Node type 2*. However, the EXLD in Eq. (3) depends on the destination of the object. If the object has any tracking record from $L2DOut(l) \setminus \{dev_{v_x}\}$ (where l is *Node type 3*) then the object did not leave the location l through dev_{v_x}. Otherwise it has left the location through dev_{v_x} without generating any tracking record. For the first case we take the corresponding EXLD otherwise we take the EXLD for the loop. For example, for $L4$, *L2Dout(L4) ={dev4, dev5, dev$_v$}* where $dev4$ = *L2In(L4)*. As the object $o1$ in Table 1 has no tracking record from $dev5$, the object $o1$ should circulate at $L4$ and left the location through dev_{v_x} without generating any tracking record. So, the *time*$_{end}$ of object $o1$ from $L4$: *time*$_{end}$ = $\lceil 54 + \frac{22m}{3m/(54-51)} \rceil$ = 76.

Node type 4 and ***Node type 5*** do not contain any outgoing edge with dev_{v_x} in label. These two node types are very similar except that *Node type 4* contains a loop and *Node type 5* does not. Fig. 3d and Fig. 3e show examples of *Node type 4* and *Node type 5* respectively. In our running example L1, L2, L3 falls in *Node type 5*. As *Node type 4* contains a loop, the $time_{end}$ of an object o from a location l of *Node type 4* is calculated from the last $time_{out}$ from the tracking records like *Node type 2 and 3*. However the EXLD $l_{l,L_{nexte}}$ is taken from the *edge(L$_{prev}$, l)*. For *Node type 5* the $time_{out}$ is directly taken from the tracking record as there is no loop in it. The EXLD in *Node type 5* is taken similarly as in *Node type 4*. In our running example the $time_{end}$ of $o1$ from $L3$ is calculated as: *time*$_{end}$ = $\lceil 18 + \frac{3m}{3m/(18-15)} \rceil$ = 21.

Table 2 shows the results after mapping from Table 1.

Table 2: MappingTable for Table 1

MappingID	ObjectID	LocationID	time$_{start}$	time$_{end}$
map1	*o1*	*L1*	*2*	*7*
map3	*o1*	*L3*	*7*	*21*
map5	*o1*	*L4*	*21*	*76*

4. HOTSPOT QUERIES

The hotspots can now be extracted from the tracking records after mapping into *MappingTable*. A hotspot query $HQ[q_s, q_e, \theta]$ finds the hotspots between time q_s and q_e where θ is the density threshold. In the inner part of a HQ, there is a density query (DQ), a count query (CQ) and a tracking record query (RQ). Fig. 4 shows the approach for processing a hotspot query. When a $HQ[q_s, q_e, \theta]$ query is asked, the system issues a $DQ[q_s, q_e]$, the $DQ[q_s, q_e]$ then issues a $CQ[q_s, q_e]$ which issues an $RQ[q_s, q_e]$. The *RQ* gets the mapping table from the database and returns the mapping records where $[time_{start}, time_{end}]$ intersects with $[q_s, q_e]$. From the relevant records, the *CQ* counts the number of objects for each location. The *DQ* then finds the density of each location from the count results with the help of capacity of the corresponding location. The *HQ* then returns the locations with $density$>θ. All of these queries can be combined into a single query and can be executed jointly. For a relational database the joint query becomes the following SQL statement. In the joint query, the *RQ* becomes the part of the *WHERE* condition, the *COUNT(DISTINCT ObjectID)* is used for *CQ*, the *DQ* is represented in the column list and is computed with the help of *CQ* and a *Capacity(Location)* function. The results are grouped based on location using *GROUP BY* and temporary stored in an inline view. Finally the *HQ* is completed with the help of a *WHERE* condition on the results from the inline view.

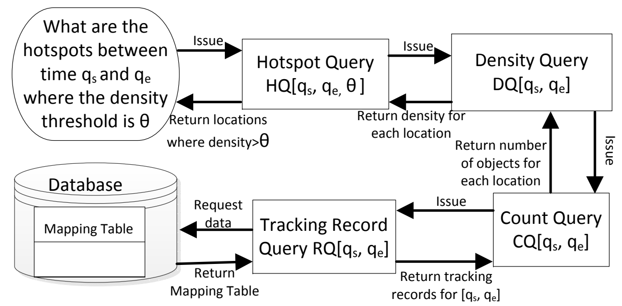


Figure 4: Query steps

SQL:*SELECT location, density FROM (SELECT location, (COUNT (DISTINCT ObjectID) / (t_e-t_s)) / Capacity (location) * 100 AS density FROM MappingTable m WHERE (m.t$_{in}$* BETWEEN t_s AND t_e) OR (m.t$_{out}$ BETWEEN t_s AND t_e) OR (t_s BETWEEN m.t$_{in}$ AND m.t$_{out}$) GROUP BY location) WHERE density > θ

5. CONCLUSION AND FUTURE WORK

We proposed an approach to extract the hotspots from indoor tracking data. We developed a graph-based model for mapping the tracking records with the semantic location so that it is possible to know the entry and exit times of an object at a constrained path symbolic location. Then the mapping records are used for hotspots extraction. The mapping records are also very useful for other kind of analyses e.g., stay duration, travel time estimation etc.

Future work will be to model more complex indoor topologies for mapping the same information we did for constrained path. An indexing technique for efficient query processing can be developed. Hotspot query for online indoor tracking data will be another relevant future work.

Acknowledgment

This work is supported by the BagTrack project funded by the Danish National Advanced Technology Foundation under grant no. 010-2011-1.

6. REFERENCES

[1] T. Ahmed, T. B. Pedersen, and H. Lu. A data warehouse solution for analyzing RFID-based baggage tracking data. In *MDM (1)*, pages 283–292, 2013.

[2] M. Hadjieleftheriou, G. Kollios, D. Gunopulos, and V. J. Tsotras. On-line discovery of dense areas in spatio-temporal databases. In *SSTD*, pages 306–324, 2003.

[3] C. S. Jensen, D. Lin, B. C. Ooi, and R. Zhang. Effective density queries on continuously moving objects. In *ICDE*, page 71, 2006.

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

[5] X. Li, J. Han, J.-G. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. In *SSTD*, pages 441–459, 2007.

[6] M. F. Worboys. Modeling indoor space. In *ISA*, pages 1–6, 2011.