In Search of Indoor Dense Regions: An Approach Using Indoor Positioning Data

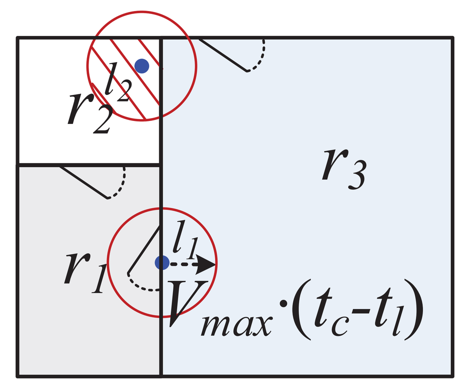
[†]Huan Li, [‡]Hua Lu, [†]Lidan Shou, [†]Gang Chen, and [†]Ke Chen

[†]College of Computer Science and Technology, Zhejiang University, China
[‡]Department of Computer Science, Aalborg University, Denmark

1. Introduction

- People spend 87% of their daily time indoors. It is useful to measure the *indoor densities* and find the dense regions.
  - *Application in flow and security control*: authority in large airport ➞ find most crowded regions ➞ open more fast tracks ➞ help passengers timely.
- Multiple challenges in measuring indoor densities.
  - *Complex indoor topology* enables as well as constrains indoor object movements.
  - Using *counting sensors*: extra hardware investment, rigorous sensor deployment, no support for user-defined indoor regions.
  - Using *indoor positioning data*: Lower sampling issue and discrete location reports that leave considerable uncertainty at a particular time.
- A **low-cost** approach using *online uncertain indoor positioning data*.
  - We design an indoor density definition *amenable to indoor object location uncertainty*, and formulate the problem of finding top-k indoor dense regions.
  - We analyze the indoor location uncertainty, derive *bounds of indoor densities*, and introduce *distance decaying effect* into indoor density computing.
  - By making use of the uncertainty analysis outcomes, we design *efficient algorithms* to search for the current top-k indoor dense regions.

2. Problem Formulation



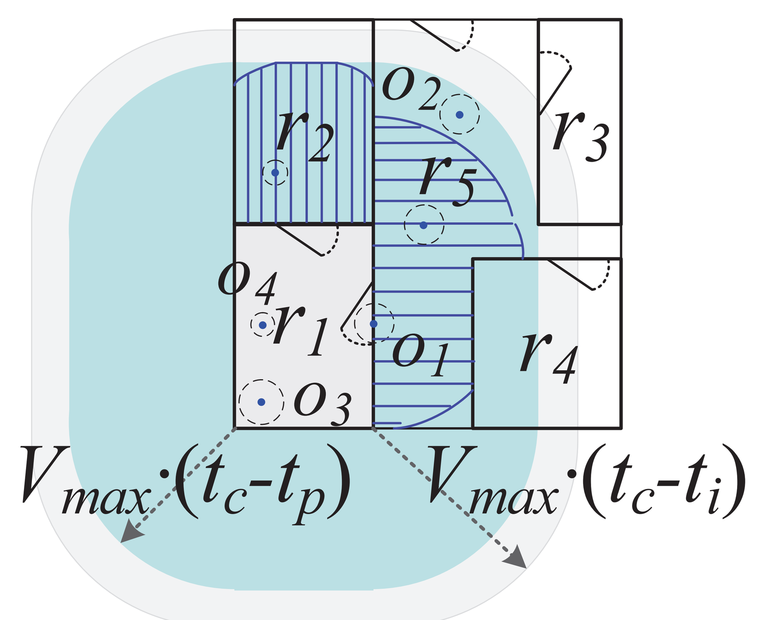
- ***Online Indoor Positioning Table.*** Only the *latest* positioning information (o, loc, t) is maintained, and there is no more recent information available.
- ***Indoor Uncertainty Region.*** $UR_l(loc, t_c, t_l)$ describes the indoor portions where the object can reach at the current time t_c under the maximum speed constraint V_{max}.
- ***Distance Decaying Object Presence.*** Given an indoor region r, an object o's indoor uncertainty region $UR_l(loc)$ with the distance decaying function Γ (a *monotone nonincreasing* function with indoor distance δ), o's presence in r is $\phi_r^{\Gamma}(o) = \frac{\int_{l \in (UR_l(loc) \cap r)} \Gamma(dist_l(loc, l)) \mathrm{d}l}{\int_{l \in UR_l(loc)} \Gamma(dist_l(loc, l)) \mathrm{d}l}$.
- ***Density.*** Given a set O of indoor objects, an indoor region r's density is $\tau_O(r) = \frac{\sum_{o \in O} \phi_r^{\Gamma}(o)}{Area(r)}$.
- ***Top-k Indoor Dense Region Search.*** Given a set O of indoor objects, the top-k indoor dense region search returns k densest indoor regions in a k-subset $Q_k \subseteq Q$ such that $\forall r \in Q_k, \forall r' \in Q \setminus Q_k, \tau_O(r) \geq \tau_O(r')$.
- Our problem setting allows users to customize *semantic-dependent query regions* according to their practical needs.

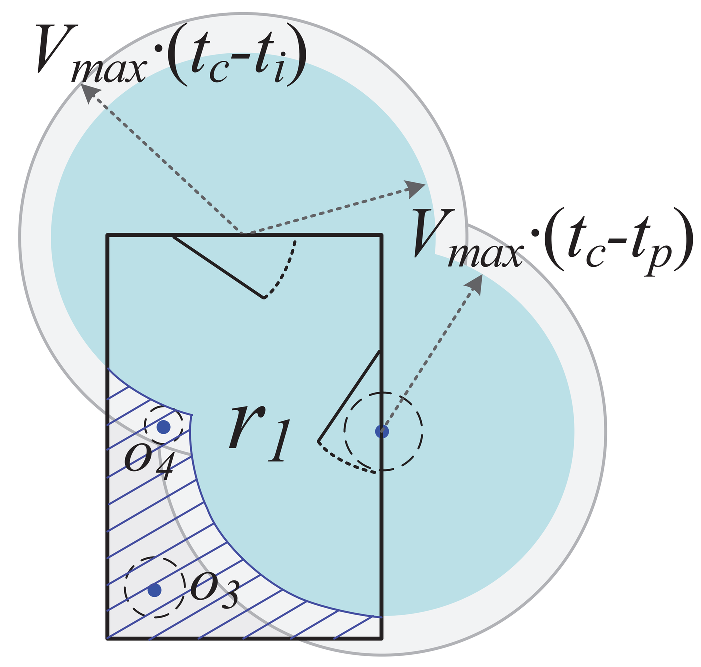
3. Summary of Our Approach

It's complex to compute precise indoor densities. The discrete nature of indoor positioning makes the object location already **out-of-date** at the search time.

3.1 Bounds of Indoor Density — concentrate on relevant objects only.

- A region r where there was no object at a past time t_p can contain objects at time t_c. However, they can only come from a buffer region that contains r, i.e., a δ-*Minkowski region* where $\delta = V_{max} \cdot (t_c - t_p)$.
- Considering indoor topology, we define r's ***indoor buffer region*** $\Theta_l^{\triangleright}(r)$ as intersection of r's buffer region and the indoor parts from where one can reach r within time interval $[t_p, t_c]$.
- Oppositely, we define r's ***indoor core region*** $\Theta_l^{\triangleleft}(r)$ as a reduced region of r from where one cannot leave r within $[t_p, t_c]$.





- ***Indoor Density Bounds.*** $\frac{COUNT(\Theta_l^{\triangleleft}(r))}{Area(r)} \leq \tau_O(r) \leq \frac{COUNT(\Theta_l^{\triangleright}(r))}{Area(r)}$, where function $COUNT(r)$ obtains the number of objects whose last reported location is contained by region r.
- ***Temporal Loose Bounds.*** For two past timestamps t_i and t_p, if $t_i \leq t_p$ we have $\frac{COUNT(\Theta_l^{\triangleleft}(r, t_c, t_i))}{Area(r)} \leq \frac{COUNT(\Theta_l^{\triangleleft}(r, t_c, t_p))}{Area(r)} \leq \tau_O(r) \leq \frac{COUNT(\Theta_l^{\triangleright}(r, t_c, t_p))}{Area(r)} \leq \frac{COUNT(\Theta_l^{\triangleright}(r, t_c, t_i))}{Area(r)}$.

3.2 Top-k Search Algorithms based on derived density bounds.

- The overall framework uses a max-heap to control the processing order of query regions.
  - It uses the oldest timestamp in OIPT to derive $\Theta_l^{\triangleright}(r)$ and $\Theta_l^{\triangleleft}(r)$ for each r (c.f. *Temporal Loose Bounds*).
  - It overestimates (underestimates) r's density by counting all objects whose last reported location is in $\Theta_l^{\triangleright}(r)$ ($\Theta_l^{\triangleleft}(r)$).
  - Only regions whose upper bound density is no less than *the current k-th highest lower bound density* should be further processed.
- The framework calls a top-k search algorithm that gives priority to the regions with higher overestimated density values.
  - ***One-pass Search*** gets the overestimated object set $set_{\top}$ and underestimated object set $set_{\perp}$, and continues to update r's object load by only going through the variable part in $set_{\top} \setminus set_{\perp}$.
  - ***Improved Search*** decomposes the one-pass load computing into two passes, it counts the objects whose uncertainty region is contained by and overlaps r, respectively.
  - Between the two passes, a ***tighter upper bound*** $\tau_O(r) \leq \frac{OverCount(r)}{Area(r)} \leq \frac{COUNT(\Theta_l^{\triangleright}(r))}{Area(r)}$ is utilized, where function $OverCount(r)$ obtains the number of objects whose uncertainty region only overlaps with region r.
  - In a two-pass way, we expect to avoid part of expensive counting for more uncertain objects as *the tighter overestimated density may be lower than other query regions' final densities that are either already computed or to be computed soon.*

4. Experimental Results

4.1 Efficiency studies using a large synthetic dataset.

- Alternative indoor dense region search methods.
  - DC directly counts the objects contained by r and thus has the lowest time and memory cost.
  - Nested-loop variants (NL*) sum up object presences to r's density based on our definition. NLRegion (NLObject)'s outer-loop is oriented to regions (objects); NLwgbr and NLwibr use general buffer region and indoor buffer region to reduce the search space, respectively.
- Efficiency comparison in default parameter setting.
  - The pruning ratio indicates that the bounds of indoor density in our methods are very effective in pruning objects.
  - TopkIDRsImprd has a higher pruning capability than TopkIDRs1Pass, as it can early prune some query regions whose final density is already lower than other regions' tighter overestimated bounds.

Algorithms	Running time (ms.)	Pruning ratio	Memory cost (MB.)
TopKIDRs1Pass	399.7	81.56%	147.8
TopKIDRsImprvd	365.7	**85.02%**	156.1
DC	**68.5**	-	**2.2**
NLRegion	148386.2	0	342.5
NLObject	2248.1	0	321.3
NLwibr	1082.2	60.74%	68.6
NLwgbr	1597.7	34.85%	51.2

4.2 Effectiveness studies using a real university dataset.

- We compare our uncertainty model based method (UM) to DC in terms of Kendall coefficient and recall.
  - ***Kendall coefficient*** τ is a rank correlation measure between the ranking of the top-k search result and the top-k ground truth. It varies within $[-1, 1]$.
- The effect of changing $|Q|$ and the query time interval Δt.
  - UM always outperforms DC. UM's two measures decrease when more query regions are involved. Its τ is close to 0.7 when using all query regions.
  - An object's uncertainty region becomes larger when Δt increases, which reduces the accuracy of indoor densities computed. Nevertheless, UM's recall is still higher than 0.78, showing that our method is very effective in finding correct results even the OIPT contains relatively old location reports.