Effective Density Queries on Continuously Moving Objects

By Christian S. Jensen, Dan Lin, Beng Chin Ooi, Rui Zhang

Sari Haj Hussein[1]

[1]Department of Computer Science
Aalborg University

2012-04-19

What & Why

- What? $\rightsquigarrow$ The paper studies density queries
- Why? $\rightsquigarrow$ In traffic management systems, they can be used to identify regions of traffic jams

Problem Description

Density

Density of R at t is the number of objects in R at t divided by the area of R

Dense Region

R is dense at t if its density is higher than a threshold ρ

Effective Density Query

Report all dense regions at t that satisfy:

1. Answer meaningfulness: any reported region is constrained to a certain shape and an area range (square, circle, etc)
2. Non-redundancy: reported regions do not overlap
3. No answer loss: any dense region in the query input appears in the results (incorporating evidence)

Problem Description

Density

Density of R at t is the number of objects in R at t divided by the area of R

Dense Region

R is dense at t if its density is higher than a threshold ρ

Effective Density Query

Report all dense regions at t that satisfy:

1. Answer meaningfulness: any reported region is constrained to a certain shape and an area range (square, circle, etc)
2. Non-redundancy: reported regions do not overlap
3. No answer loss: any dense region in the query input appears in the results (incorporating evidence)

Problem Description

Density

Density of R at t is the number of objects in R at t divided by the area of R

Dense Region

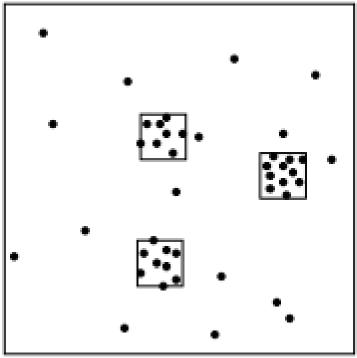
R is dense at t if its density is higher than a threshold ρ

Effective Density Query

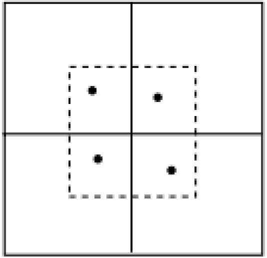
Report all dense regions at t that satisfy:

1. Answer meaningfulness: any reported region is constrained to a certain shape and an area range (square, circle, etc)
2. Non-redundancy: reported regions do not overlap
3. No answer loss: any dense region in the query input appears in the results (incorporating evidence)

We want something like this...



But we do not want this!



Problem Parameters

Problem Parameters

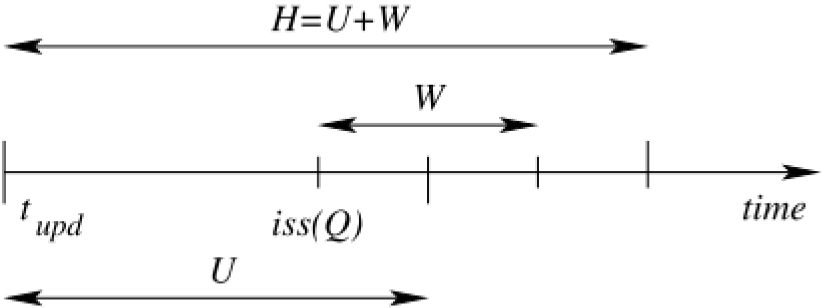
- Linear model for the position of a moving object $\rightsquigarrow$ $\bar{x}(t) = \bar{x} + \bar{v}(t - t_{upd})$
  - $t \rightsquigarrow$ current time
  - $t_{up} \rightsquigarrow$ latest update time
  - $\bar{x}(t) \rightsquigarrow$ object position at t
  - $\bar{x} \rightsquigarrow$ object position at t_{upd}
  - $\bar{v} \rightsquigarrow$ object velocity at t_{upd}
- Object position at $t \rightsquigarrow (\bar{x}, \bar{v}, t_{upd})$

Problem Parameters Continued

Problem Parameters Continued

- $U \rightsquigarrow$ maximum update time $\rightsquigarrow$ maximum duration in-between 2 updates of a moving object position
- $t_q \rightsquigarrow$ query time (execution time)
- $t_{issue} \rightsquigarrow$ query issue time
- $W \rightsquigarrow$ query reach into the future starting from t_{issue}
- $H = U + W \rightsquigarrow$ query horizon $\rightsquigarrow$ query reach into the future starting from t_{upd}
- $[t_q, t_q + H] \rightsquigarrow$ query time window

Problem Parameters Illustrated



Summary

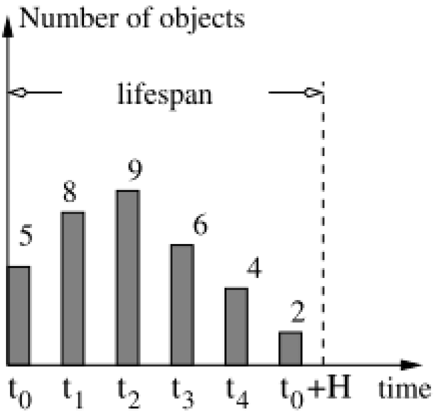
- Moving objects are maintained in some index structure
- Data space is partitioned into small cells of equal sizes
- Dense regions are squares of certain sizes
- They can intersect with the cell partitioning
- Steps involved:
    1. Counter maintenance
    2. Filtering phase of query processing
    3. Refinement phase of query processing

1. Introduction
2. Problem
3. Problem Parameters
4. The MODQ Framework
   - Counter Maintenance
   - Filtering Phase of Query Processing
   - Refinement Phase of Query Processing

Counter Maintenance

- A counter of the number of objects in each cell at all times in $[t_q, t_q + H]$
- Update counters as objects move between cells
  - An object is inserted into a cell $\Rightarrow$ increase the corresponding counter & update the index
  - An object is deleted from a cell $\Rightarrow$ decrease the corresponding counter & update the index
- Large number of cells $\rightsquigarrow$ counter maintenance becomes a problem

Density Histogram



Compressing the Histogram

- Compression is done using Discrete Cosine Transform (DCT)
- DCT is commonly used in loosy compression e.g., MP3, JPEG, etc.

Compressing the Histogram

Discrete Cosine Transform (DCT)

- $G(k) = c(k)\sum_{t=0}^{H-1} s(t) cos\frac{\pi(2t+1)k}{2H}$
- $c(0) = \sqrt{1/H}, c(k) = \sqrt{2/H}, k = 0, 1, ..., (H-1)$
- $s(t)$ is a signal and $G(k)$ is the transformed signal
- We store only $10-20\%$ of $G(k)$ and there lies the compression
- $s(t)$ is eventually a variable that changes over time
- In our scenario, $s(t)$ is the number of objects in each cell

Inverse Discrete Cosine Transform (IDCT)

- $s(t) = \sum_{k=0}^{H-1} c(k) G(k) cos\frac{\pi(2t+1)k}{2H}$
- $t = 0, 1, ..., (H-1)$

Compressing the Histogram

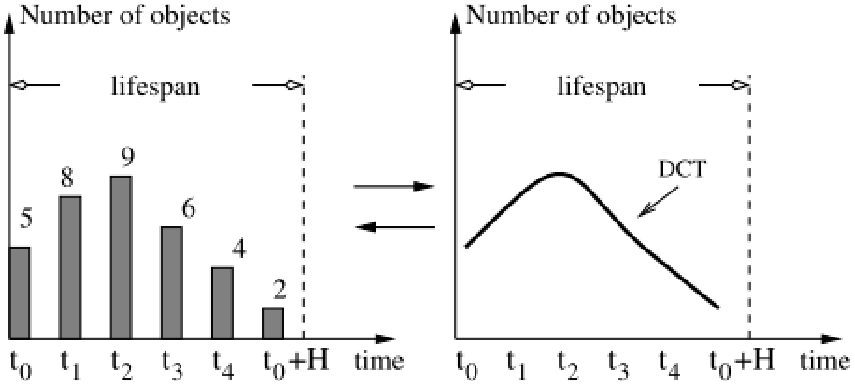
Discrete Cosine Transform (DCT)

- $G(k) = c(k) \sum_{t=0}^{H-1} s(t) cos \frac{\pi(2t+1)k}{2H}$
- $c(0) = \sqrt{1/H}, c(k) = \sqrt{2/H}, k = 0, 1, ..., (H-1)$
- $s(t)$ is a signal and $G(k)$ is the transformed signal
- We store only $10 - 20\%$ of $G(k)$ and there lies the compression
- $s(t)$ is eventually a variable that changes over time
- In our scenario, $s(t)$ is the number of objects in each cell

Inverse Discrete Cosine Transform (IDCT)

- $s(t) = \sum_{k=0}^{H-1} c(k) G(k) cos \frac{\pi(2t+1)k}{2H}$
- $t = 0, 1, ..., (H-1)$

Compressing the Histogram



Compressing the Histogram

- Storing only $10 - 20\%$ of $G(k)$ $\Rightarrow$ information loss $\Rightarrow$ restored $s'(t)$ differ from original $s(t)$
- $s'(t)$ overestimates $s(t)$ $\rightsquigarrow$ false positive query results suggesting that a cell is dense when it is not $\rightsquigarrow$ increase query processing cost!
- $s'(t)$ underestimates $s(t)$ $\rightsquigarrow$ false negative query results suggesting that a cell is not dense when it is $\rightsquigarrow$ answer loss!

Compressing the Histogram

- Storing only $10 - 20\%$ of $G(k)$ $\Rightarrow$ information loss $\Rightarrow$ restored $s'(t)$ differ from original $s(t)$
- $s'(t)$ overestimates $s(t)$ $\rightsquigarrow$ false positive query results suggesting that a cell is dense when it is not $\rightsquigarrow$ increase query processing cost!
- $s'(t)$ underestimates $s(t)$ $\rightsquigarrow$ false negative query results suggesting that a cell is not dense when it is $\rightsquigarrow$ answer loss!

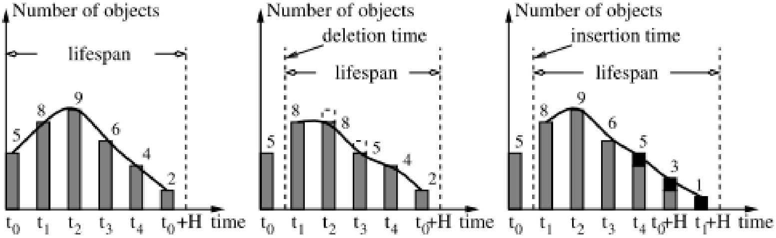
Compressing the Histogram

- Storing only $10-20\%$ of $G(k)$ $\Rightarrow$ information loss $\Rightarrow$ restored $s'(t)$ differ from original $s(t)$
- $s'(t)$ overestimates $s(t)$ $\rightsquigarrow$ false positive query results suggesting that a cell is dense when it is not $\rightsquigarrow$ increase query processing cost!
- $s'(t)$ underestimates $s(t)$ $\rightsquigarrow$ false negative query results suggesting that a cell is not dense when it is $\rightsquigarrow$ answer loss!

Compressing the Histogram

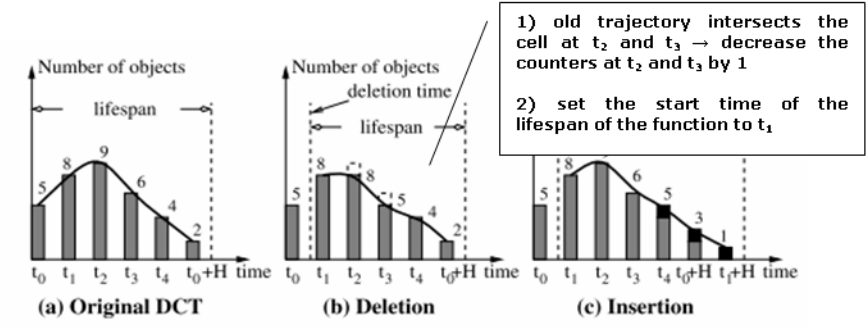
- The error bound $E_b = s(t) - s'(t)$ (formula does not show)
- Before reducing $G(k)$, compute and store a term in E_b
- Adding this E_b to $s'(t)$ guarantees the absence of false negatives!

Maintaining the Histogram



(a) Original DCT (b) Deletion (c) Insertion

Maintaining the Histogram

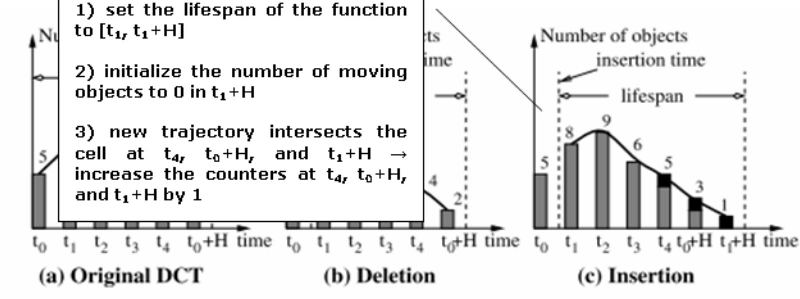


Maintaining the Histogram

1) set the lifespan of the function to $[t_1, t_1+H]$

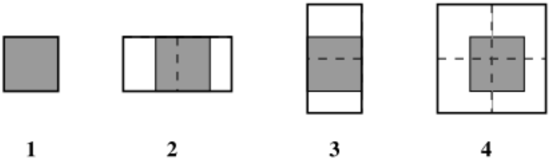
2) initialize the number of moving objects to 0 in t_1+H

3) new trajectory intersects the cell at t_4, t_0+H, and t_1+H → increase the counters at t_4, t_0+H, and t_1+H by 1



Aim, Input, and Output

- Aim $\rightsquigarrow$ identify areas that may contain answers to the density query
- Input $\rightsquigarrow$ query spatial window R, threshold ρ, query time t_q
  - the minimum number of objects that should occupy a dense square $N_{min} = R.\rho$
- Output $\rightsquigarrow$ dense regions of size $1 - 4$ times larger than R



Filtering Phase

Filtering Phase

- For each cell C, compute N_c
- If $N_c \geq N_{min}$, add C to the answer
- If $N_c < N_{min}$, look at the 4-cell square $4C$ having C at the top-left corner
- If $N_{4c} \geq N_{min}$, look at each $1C$ area in $4C$
- If $N_{1c} \geq N_{min}$, add $1C$ to the answer
- Look at each $2C$ and $3C$ areas in $4C$
- If $N_{2c} \geq N_{min}$ or $N_{3c} \geq N_{min}$, pass $2C$ or $3C$ to the refinement phase
- If an answer is returned, modify the histogram

1. Introduction
2. Problem
3. Problem Parameters
4. The MODQ Framework
   - Counter Maintenance
   - Filtering Phase of Query Processing
   - Refinement Phase of Query Processing

Refinement Phase

Refinement Phase

- Retrieve objects in the candidate areas $2C$ or $3C$ by issuing a spatial window query on the index
- If we have $2C$
  - Sort object positions according to their x coordinates
  - $N_{\sqrt{R}} \rightsquigarrow$ the number of objects every $\sqrt{R}$
    - If $N_{\sqrt{R}} \geq N_{min}$, report $2C$ as an answer to the filtering phase
- The same applies if we have $3C$, although the sorting is done according to the y coordinates

Thank You!