EFFECTIVE DENSITY QUERIES ON CONTINOUSLY MOVING OBJECTS

Authors:

Christian S. Jensen – Aalborg University, Denmark

Dan Lin, Beng Chin Ooi, Rui Zhang – National University of Singapore

Published:

Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)

Presenter:

Jonas Hansen – Aalborg University, Denmark

Outline

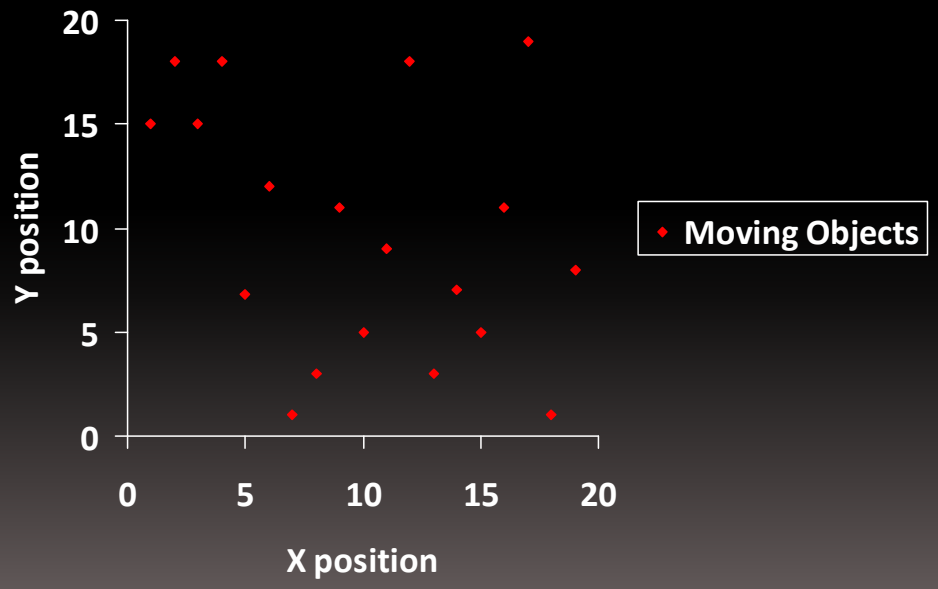
- Introduction (FIX EXAMPLE)
- Effective Density Query
- Moving Object Density Query (MODQ) Framework
- Histogram and Discrete Cosine Transform (DCT)
- Query Processing
- Experiments
- Conclusion
- Related Work
- Evaluation (MISSING)

INTRODUCTION

Introduction

- The paper defines a new type of density query with desirable properties and then proposes an algorithm for the efficient computation of density queries.
- The position of moving objects are tracked over time

Snapshot of moving objects at time ***t***



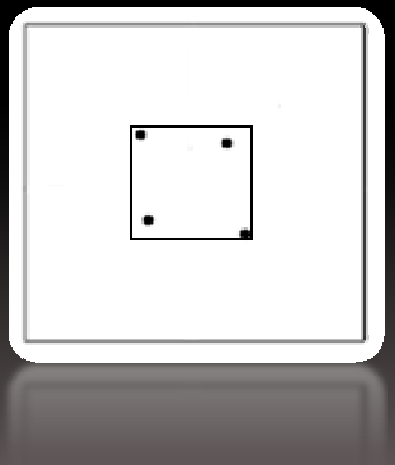
Introduction

Definitions used:

- Density: The number of objects in a region ***R*** at time ***t*** divided by the area of ***R***
- Dense Region: A region ***R*** whose density at time ***t*** is larger than a threshold ***ρ***
- Density Query: A query that finds regions in space with associated points in time, where the density exceeds the threshold ***ρ***

$$\frac{NumObj_t(R)}{Area(R)}$$

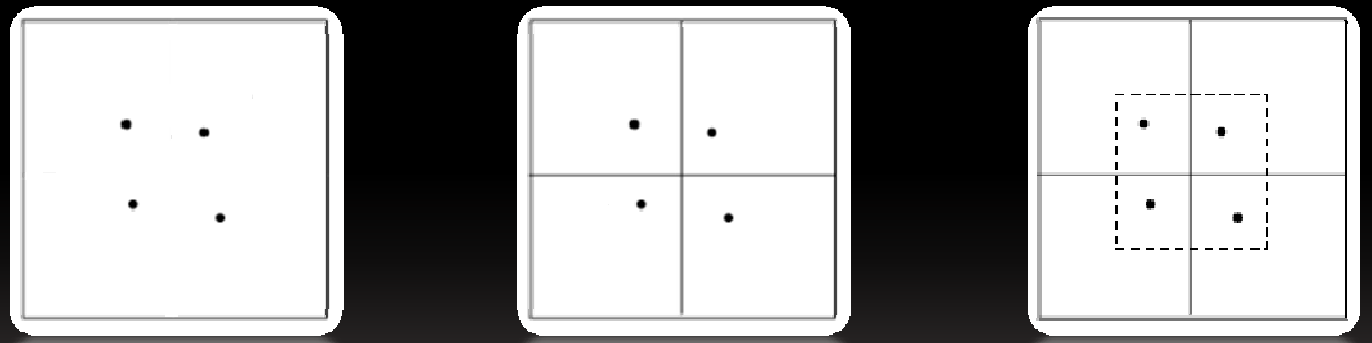
$$\frac{NumObj_t(R)}{Area(R)} > \rho$$



Area(R) = 4
ρ = 1

Introduction

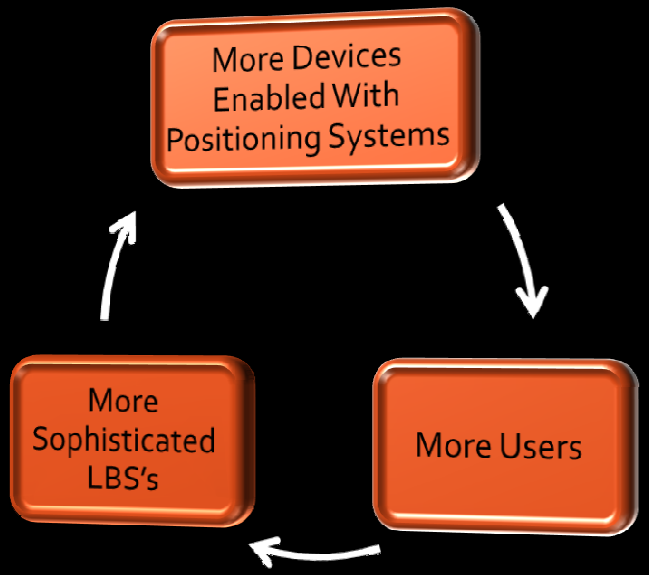
- The most common techniques for finding dense regions partition the data space in equal sized windows
- Each window is queried for dense regions
- Possibility of answer loss



Introduction

- Density query example: ????
- Who are interested in density queries
  - ??????
  - Traffic management – for identifying regions with potential for congestion traffic control systems, bandwidth management, collision probability evaluation, real-time traffic density distribution

Introduction



Problem: Requires a more effective data management foundation

Solution: Effective Density Query

EFFECTIVE DENSITY QUERY

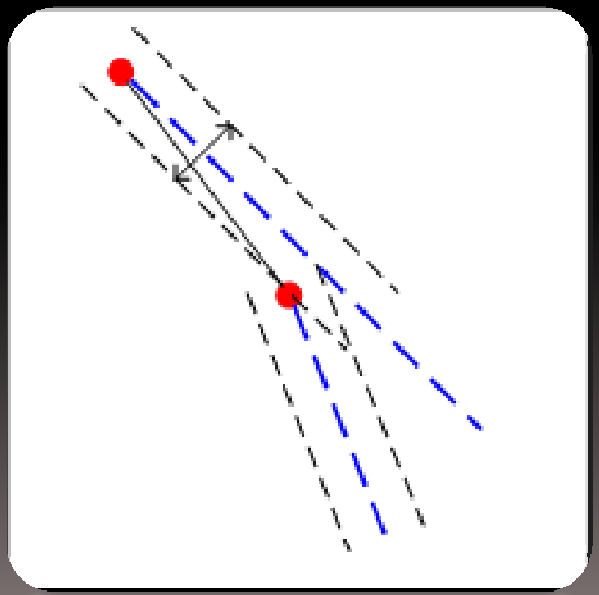
Effective Density Query

- Tracking definitions
  - In order to provide an effective means of tracking
- Objects tracked is capable of sending their location to a central server
  - Typical LBS client/server architecture
- Data format
  - Client position and velocity saved at each update

$$\overline{x}(t) = \overline{x}_{upd} + \overline{v}(t - t_{upd})$$

Effective Density Query

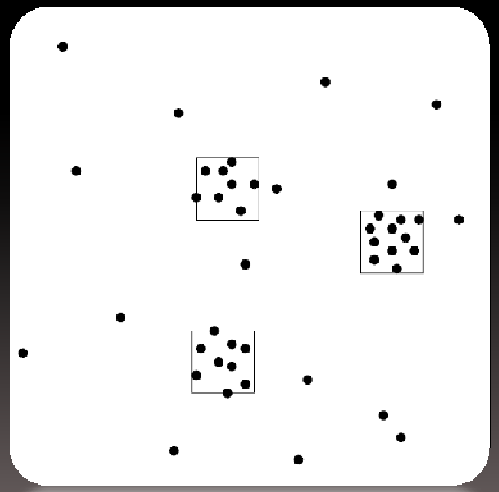
- Update strategy reduces number of updates by a factor of three
- Clients send an updated location to the server when:
  - The maximum update time ***U*** is reached
  - The deviation between actual location and the server side location exceeds a specified threshold



Effective Density Query

Effective Density Query: A query that finds all dense regions at time ***t*** that satisfies

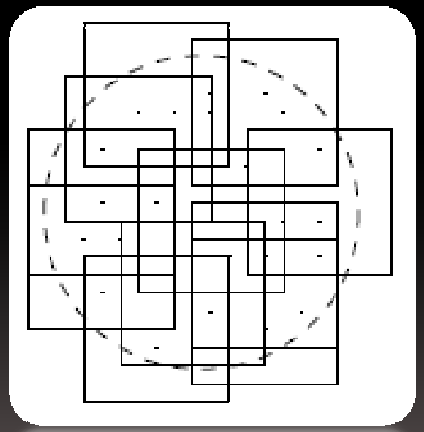
1. Any reported region is constrained to a certain shape and area (ensures meaningful answers)



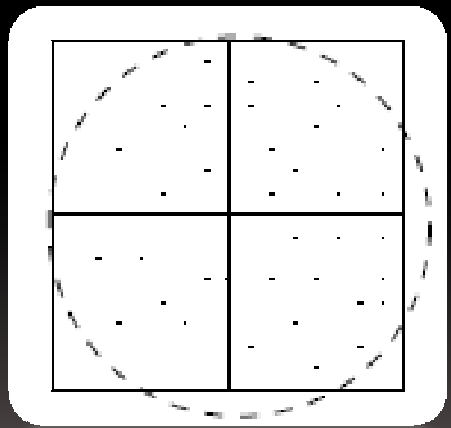
Effective Density Query

Effective Density Query: A query that finds all dense regions at time ***t*** that satisfies

1. Any reported region is constrained to a certain shape and area
2. No two regions in the result overlap



Overlapping

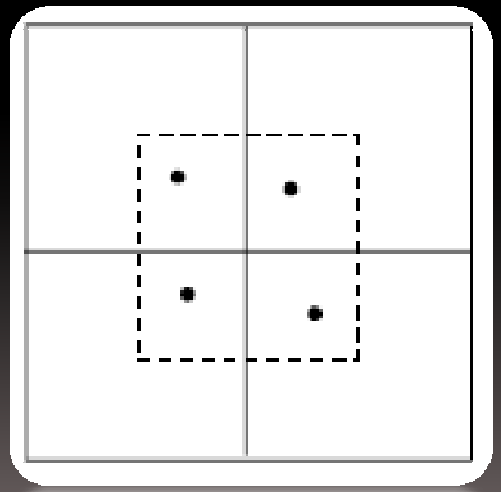


Non-overlapping

Effective Density Query

Effective Density Query: A query that finds all dense regions at time ***t*** that satisfies

1. Any reported region is constrained to a certain shape and area
2. No two regions in the result overlap
3. All dense regions must be reported – no answer loss



MOVING OBJECT DENSITY QUERY (MODQ) FRAMEWORK

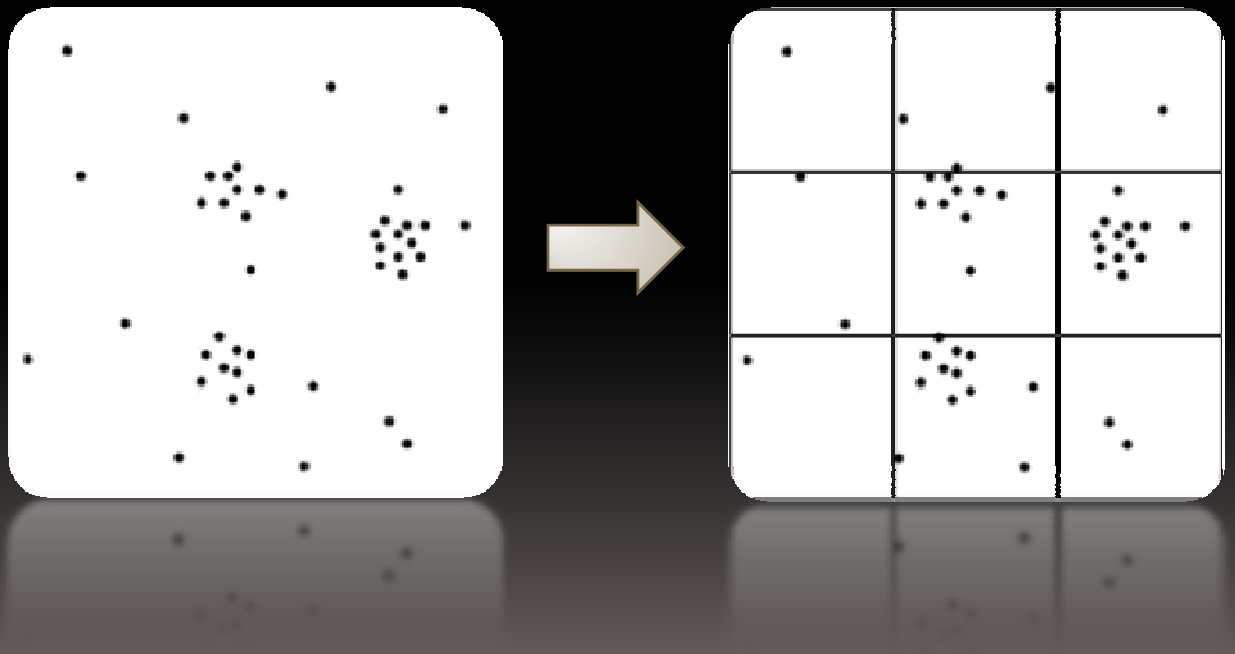
Moving Object Density Query (MODQ) Framework

- Definition of the Effective Density Query allows the general setting of the environment to be set up, so that the algorithm used for processing the queries can be defined and tested
- The MODQ framework used to compute the queries in this paper is a set of constraints posed on the density queries
  - Dense regions must be square shaped
  - Dense regions must be of a fixed and equal size
  - Moving objects are maintained in a B^x index [1]

1 - MODQ can use any index for moving objects that supports predictive window queries. C.S. Jensen et al "Query and update efficient B+ tree based indexing of moving objects".

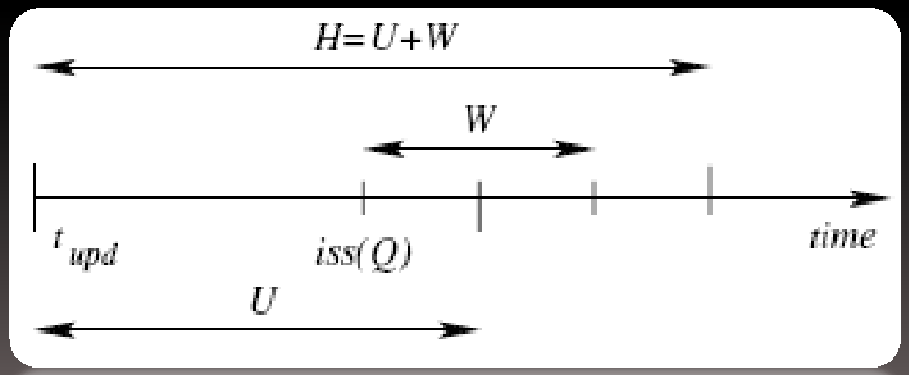
Moving Object Density Query (MODQ) Framework

- The dataspace is partitioned in windows equal to the size of the dense regions



Moving Object Density Query (MODQ) Framework

- The time span in which a query can be asked has to be confined to some bounds in order to provide meaningful results
  - ***W*** specifies how far into the future a query can be asked
  - ***iss(Q) <= t_q <= iss(Q) + W***
  - ***U*** is the max. update time defined earlier
  - Time horizon ***H*** is the max. time since last update an object can be queried

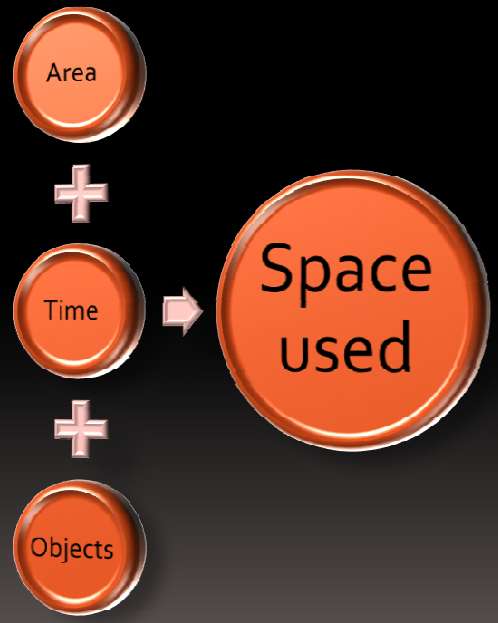


Moving Object Density Query (MODQ) Framework

- In order to query for dense regions in the partitioned space, a count of all objects, in each cell, at all times needs to be maintained

Problem: Space consumption increases as space, time and objects increase

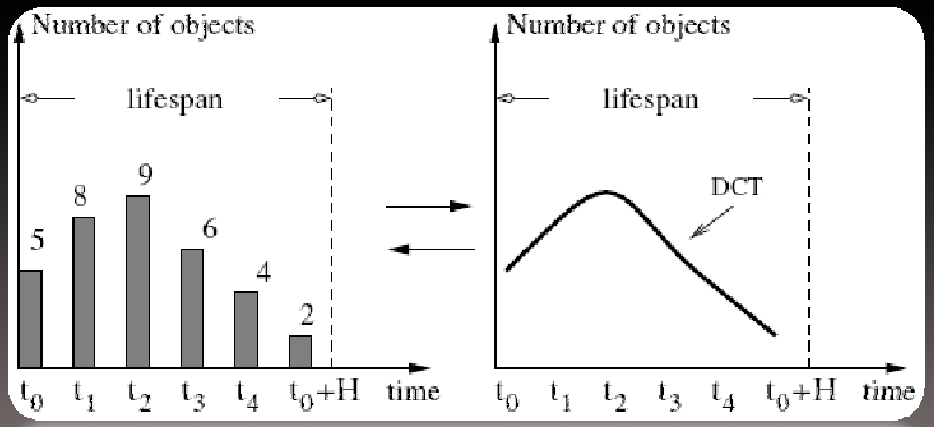
Solution: Compressed density histogram



HISTOGRAM AND DISCRETE COSINE TRANSFORM (DCT)

Histogram and DCT

- A density histogram (DH) containing a counter of all objects, in all cells, at all times in $[t_{now}, t_{now}+H]$ is maintained.
- This could be very large and require substantial I/O cost to maintain and use
- A critical need for compression
- Using the Discrete Cosine Transform (DCT) will transform the counters into a signal. This transformation can be reversed using the inverse DCT



Histogram and DCT

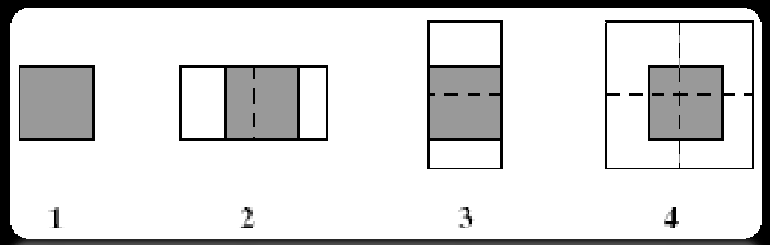
- The DCT reduces the storage cost by up to 90%
- Lossy transformation
  - False positives and negatives => over- or underestimate counters
  - Need something to specify the amount of errors willing to tolerate
- Error factor: $e_f = [0, 1]$
  - $e_f = 1$ => no false negatives but more false positives
    - More processing needed
  - $e_f = 0$ => more false negatives , but fewer false positives
    - Possibility of answer loss

QUERY PROCESSING

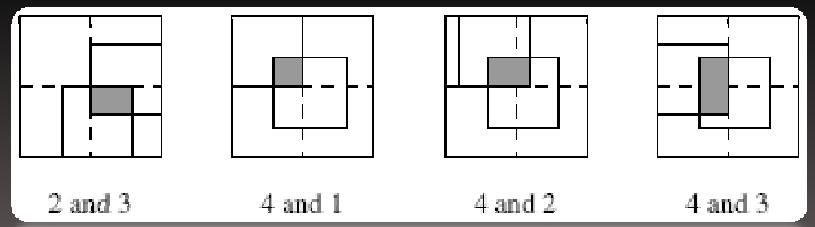
Query Processing

- 2-phase algorithm:
  - **Filtering**: Find candidate cells using the counters of the histogram
  - **Refinement**: Find any dense regions within the candidate cells

- **Filtering** phase:
  - Identify possible dense cells and discard the rest
  - Sends up to four window cells per cell to phase 2
  - Outputs the dense regions found



Intersections



Conflicts

Query Processing

Filter algorithm:

- Check single cell
  - If found report answer
- Check square
  - Check combos of 2 cells
  - If found invoke refinement
  - Else check entire square

Algorithm Density_query(ρ, R, t_q)
Input: threshold ρ, query window R, query time t_q

```
1.  N_min ← R · ρ
2.  for each cell in the space do
3.      N_b ← number of objects in the cell at t_q
4.      if N_b > N_min then
5.          report this cell as a final answer
6.      else
7.          N_s ← number of objects in square S_4
            // S_4 consists of four cells
8.          if N_s ≥ N_min then
9.              flag ← true
10.             for each combination of two cells S_2 do
11.                 N_2 ← the density in the two cells
12.                 if N_2 ≥ N_min then
13.                     invoke Refinement(S_2, ρ, R, t_q)
14.                     if an answer is found then
15.                         modify histogram
16.                         flag ← false
17.             if flag then
18.                 invoke Refinement(S_4, ρ, R, t_q)
19.                 if an answer is found then
20.                     modify histogram
end Density_query.
```

Query Processing

Refinement algorithm:

- Discard all objects already part of an answer and check the rest
- Sort objects according to type
  - Count objects for every length of √R, if above the density level report an answer

Algorithm Refinement(S, ρ, R, t)
Input: candidate area S, density threshold ρ
query window R, query time t

1. $N_{min} \leftarrow R \cdot \rho$, $S_r \leftarrow \phi$, $L_1 \leftarrow \phi$
2. **for** each cell B in S **do**
3. **if** the cell B has been retrieved **then**
4. load objects from object pool to L_1
5. $S_r \leftarrow S_r \bigcup B$
6. $L_2 \leftarrow$ WindowQuery $(S - S_r, t)$
7. $L \leftarrow L_1 \bigcup L_2$
8. $l \leftarrow \sqrt{R}$
9. **if** S is of type 2 or 4 **then**
10. sort objects in L along x-axis
11. project objects to x-axis
12. **else**
13. sort objects in L along y-axis
14. project objects to y-axis
15. $N \leftarrow$ the number of objects within each l length
16. **if** any N larger than N_{min} **then**
17. **if** S is not of type 4 **then**
18. report an answer
19. **else**
20. project objects to y-axis
21. $M \leftarrow$ number of objects within each l length
22. **if** any M larger than N_{min} **then**
23. report an answer

end Refinement.

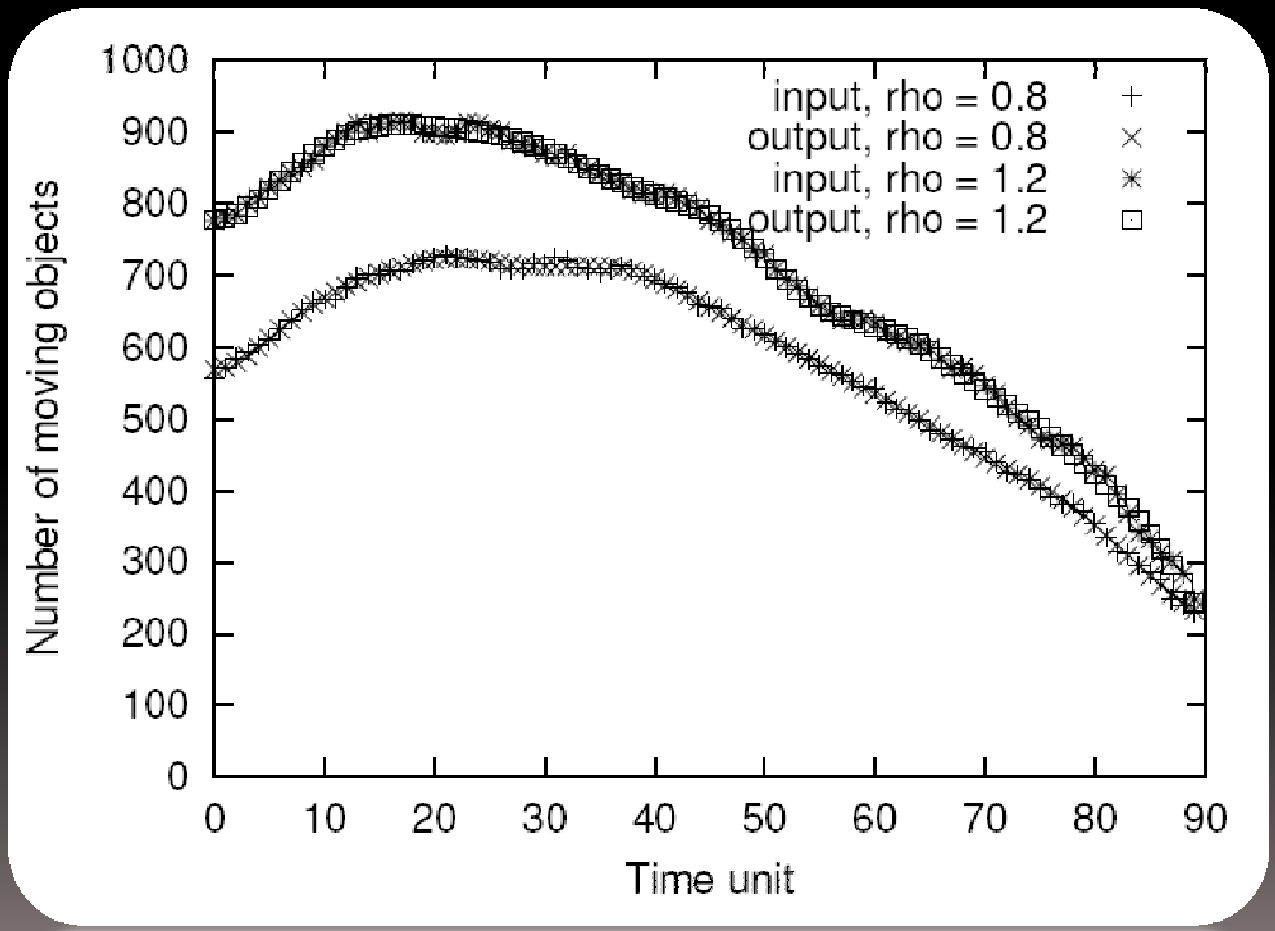
EXPERIMENTS

Experiments

- Space: 1.000 x 1.000 units
- Area: 20, **25,** 50
- Objects: **100.000,** … , 1.000.000
- Max update interval: **60,** 120, 240
- Max predication lenght: 30 time units
- Density threshold ρ : 0.8, … , **1.0** , … , 1.2

Experiments

- Number of actual (input) objects vs restored (output) objects from the DCT

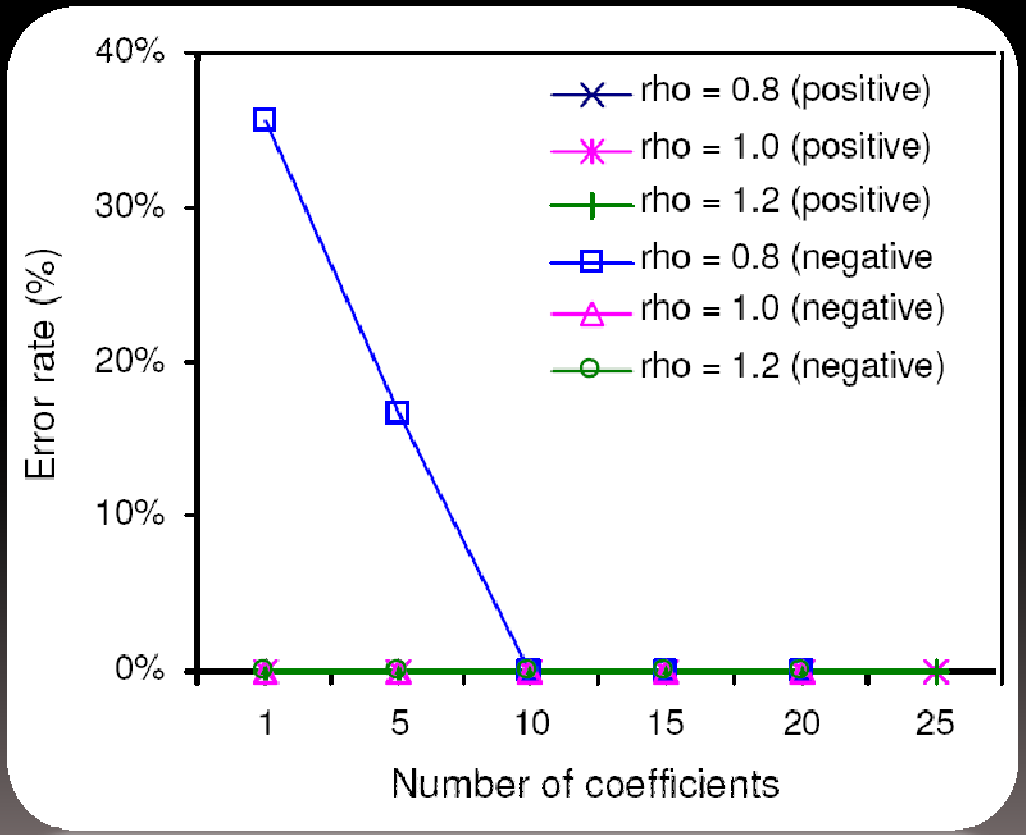


Experiments

- The percentage of errors as a result of the number of coefficients used of the original 90 coefficients of the DCT

Suggests a rather good accuracy at 10 coefficients, saving 90% of the space

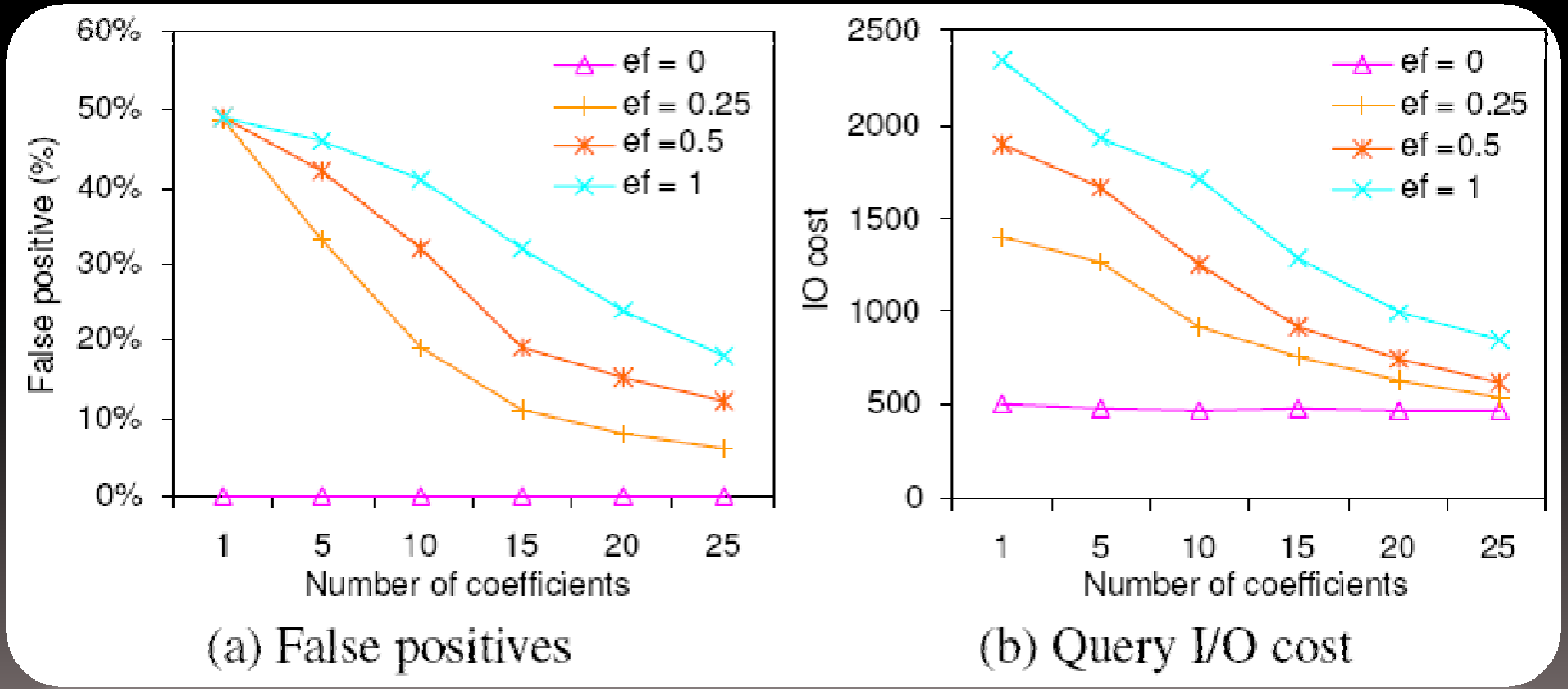
Lower density threshold results in more false negatives – i.e. wrongly discarded cells



Experiments

a) The more the signal is overestimated, the more false positives occur - the more coefficients included in the signal, the fewer false positives occur

a) I/O cost higher with $e_f = 1$ due to more false positives



CONCLUSION

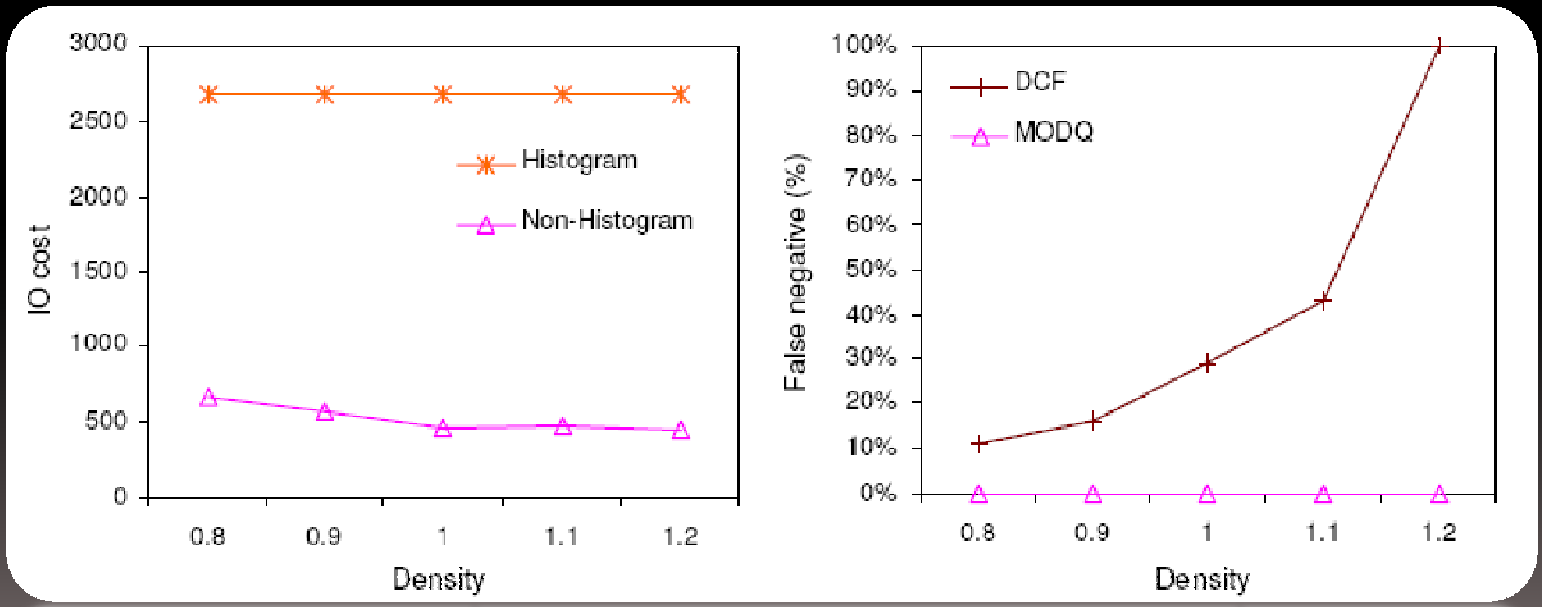
Conclusion

- Proposed and tested an Effective Density Query that is capable of:
  - Providing meaningful answers
  - Eliminating overlapping regions
  - Finding all dense regions
- Highly efficient even with an error factor of 1, but requiring more I/O's
- Even with a low error factor there is hardly any false negatives
- Virtually no compression loss while saving 90% of the space used otherwise

RELATED WORK

Related Work

- Dense Cell Filter (DCF)
  - Increasingly number of lost answers as the density threshold grow
- Histogram outperforms non-histogram with a factor 4 in regards to I/O



Related Work

- EDQ in Road Network article build upon this paper [1]

- Own Project
  - Tracking of mobile devices in airport terminals
  - Can be used to identify congestions
  - Alert personnel when many people are en-route
  - Deploy personnel where needed
  - Reroute passengers when congested areas exist
  - Advance passengers in small waves when no congestion exists

1 - http://idke.ruc.edu.cn/publications/2007/Effective%20Density%20Queries%20for%20Moving%20Objects%20in%20Road%20Networks.pdf

EVALUATION

Evaluation

- Difficult to read
  - Highly technical – not much explanation offered
- Introduction/Definitions first then problem definition then constraints in order to simplify the algorithms then definition of algorithms then experiments -> good flow
- The graphs could use an overhaul
  - Can't distinguish elements somtimes
  - Scale a bit too large sometimes
  - Fig 17 I/O COST
- Good relevant technical article!!!

THANKYOU