Efficient Object Search in Game Maps
IJCAI 2023

Jinchun Du, Bojie Shen, Shizhe Zhao, Muhammad Aamir Cheema, Adel Nadjaran Toosi

Monash University

Outline

- Introduction
- Problem definition
- Previous work
  - Interval Heuristic Polyanya
  - Incremental Euclidean Restriction (IER)
- Grid Tree
  - Euclidean Hub Labeling
  - Grid Tree
- Results

Object Search

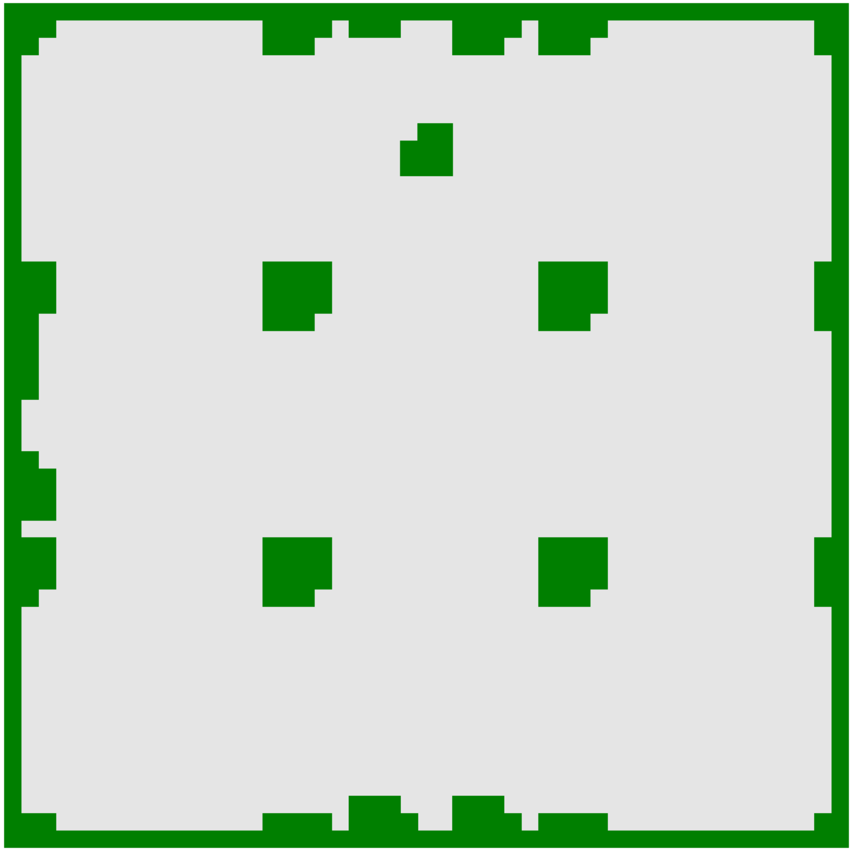
Object Search

Outline

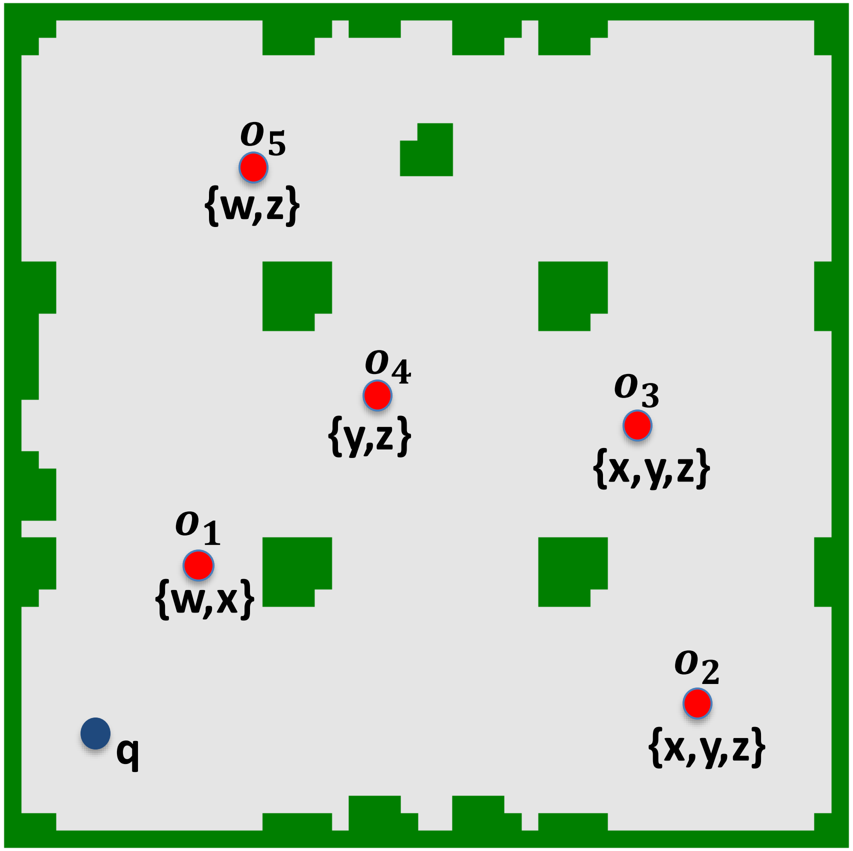
- Introduction
- Problem definition
- Previous work
  - Interval Heuristic Polyanya
  - Incremental Euclidean Restriction (IER)
- Grid Tree
  - Euclidean Hub Labeling
  - Grid Tree
- Results

Problem Definition

- Euclidean plane:
  - polygonal obstacles
    - a set of vertices
    - set of closed edges

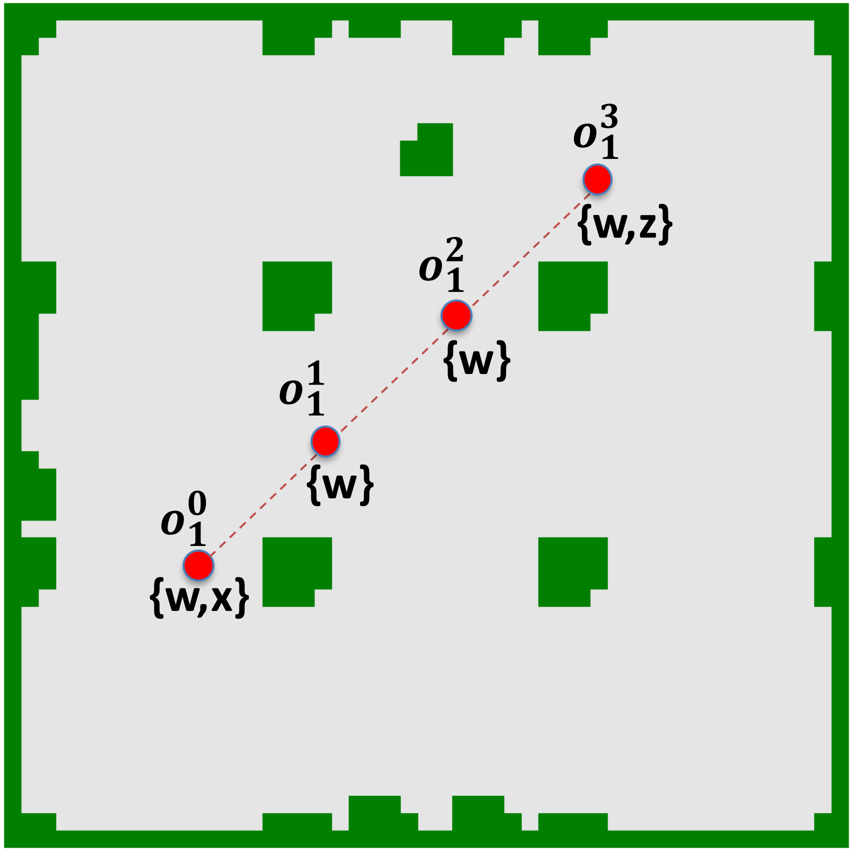


- Euclidean plane
- Objects
  - Each $o_i \in O$ is represented as a tuple $(o_i.\rho, o_i.\tau)$
  - $o_i.\rho$ is a two-dimensional point representing location
  - $o_i.\tau$ is the textual description represented as a set of keywords



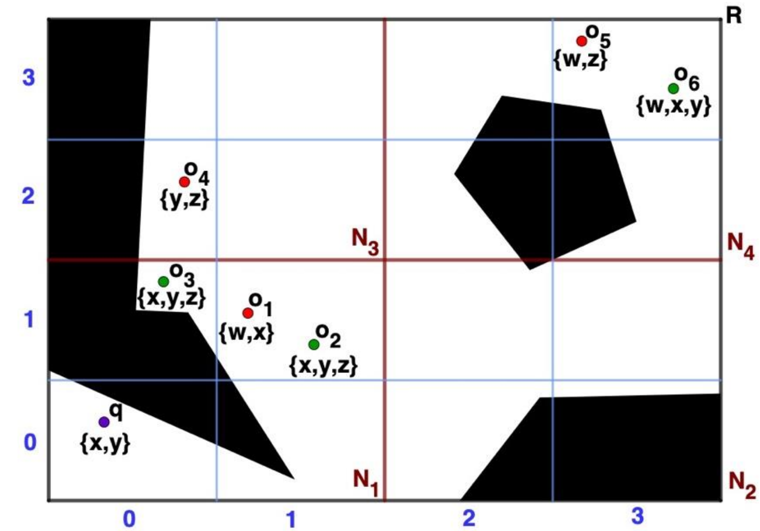
Problem Definition

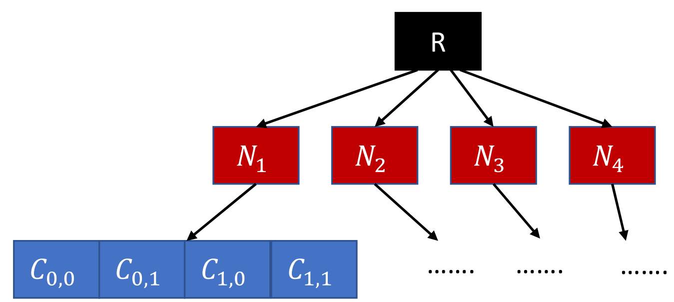
- Euclidean plane
- Objects
- Timestamp
  - Discretised into a set of timestamps T
  - O^t denote the set of objects at a timestamp $t \in T$
  - $o_i^t = (o_i^t.\rho, o_i^t.\tau)$



Problem Definition

- Euclidean plane
- Objects
- Timestamp
- Problem definition
  - Given a query q, issued at timestamp t and the set of objects O^t, find up to k objects closest from the query location among the objects that contain all query keywords





Object Lists

Cell	Objects
$C_{1,1}$	o_1, o_2
$C_{0,1}$	o_3
$C_{0,2}$	o_4
$C_{3,3}$	o_5, o_6

Keyword Lists of some nodes

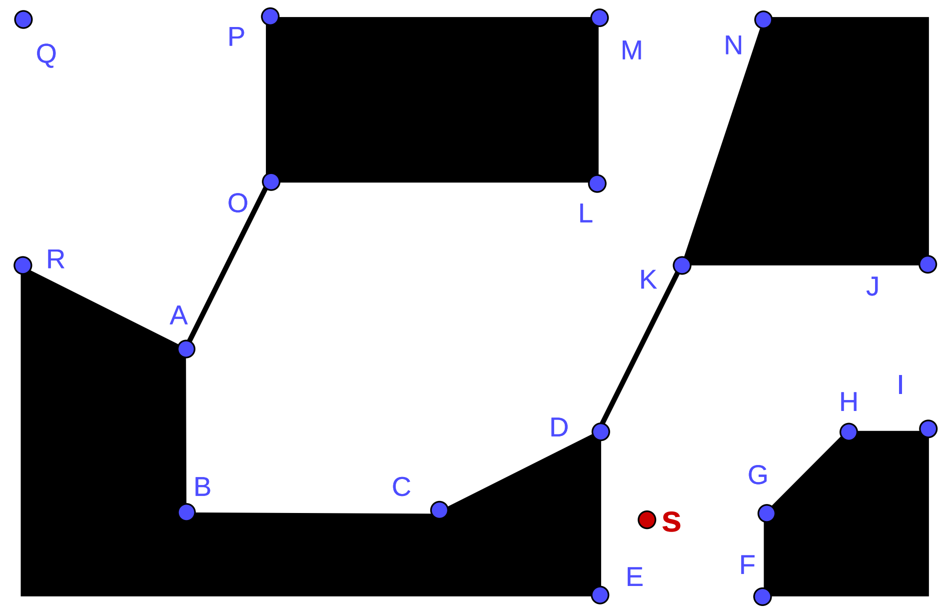
Node	Keywords
R	w: 3, x: 4, y: 4, z: 4
N_1	w: 1, x: 3, y: 2, z: 2
$C_{1,1}$	w: 1, x: 2, y: 1, z: 1
$C_{0,1}$	x: 1, y: 1, z: 1

Outline

- Introduction
- Problem definition
- Previous work
  - Interval Heuristic Polyanya
  - Incremental Euclidean Restriction (IER)
- Grid Tree
  - Euclidean Hub Labeling
  - Grid Tree
- Results

Interval Heuristic Polyanya [1]

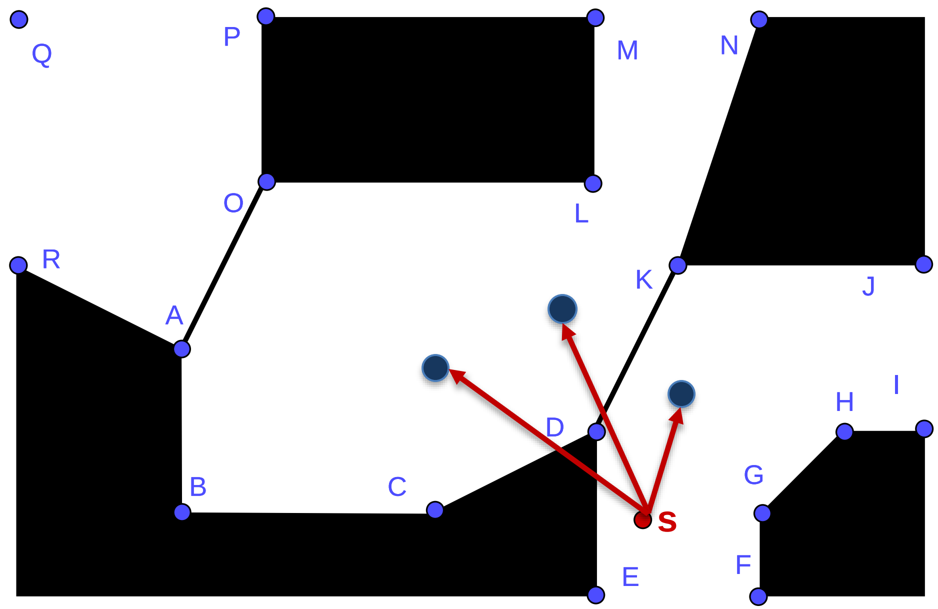
- Runs an A*-like search starting from s on navigation mesh with a set of convex polygons



[1] Zhao, S., Taniar, D., & Harabor, D. (2018). Fast k-nearest neighbor on a navigation mesh. In *Proceedings of the International Symposium on Combinatorial Search* (Vol. 9, No. 1, pp. 124-131).

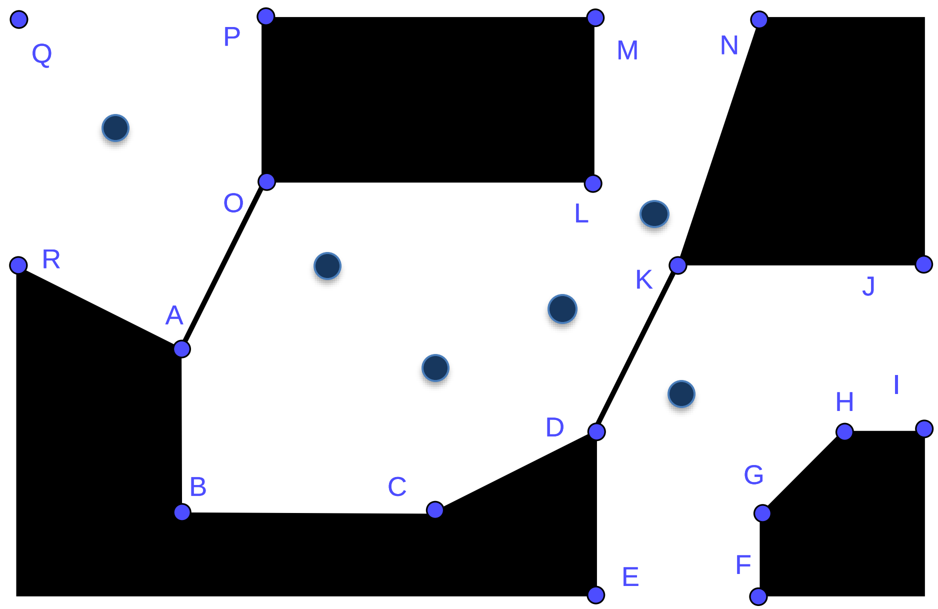
Interval Heuristic Polyanya

- Runs an A*-like search starting from s on navigation mesh with a set of convex polygons
- When the search reaches a polygon that contains an object, the object matching the keyword is added to the queue



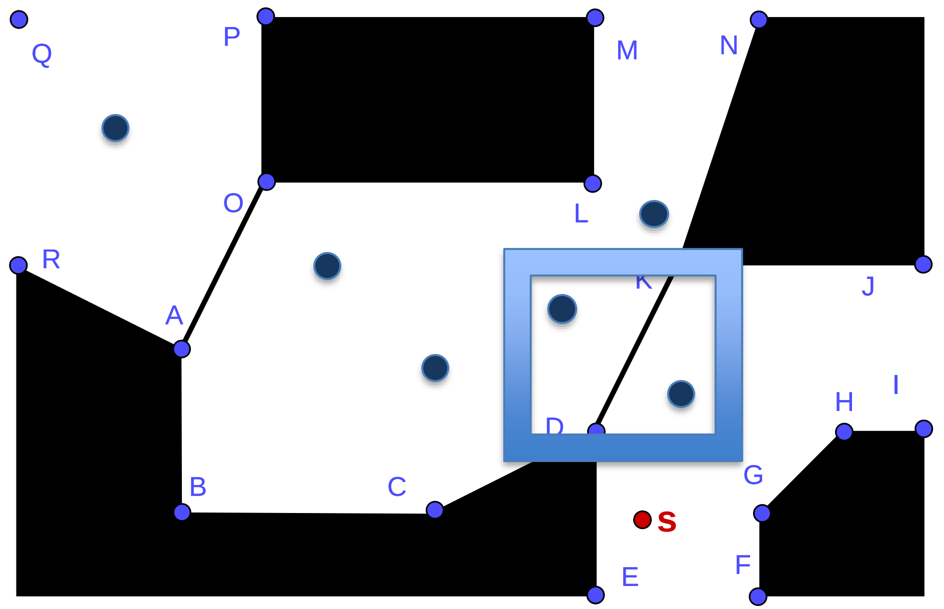
Previous work
Interval Heuristic Polyanya

- Runs an A*-like search starting from s on navigation mesh with a set of convex polygons
- When the search reaches a polygon that contains an object, the object matching the keyword is added to the queue
- Objects are stored in navigation mesh, handling object updates is quite efficient



IER [2]

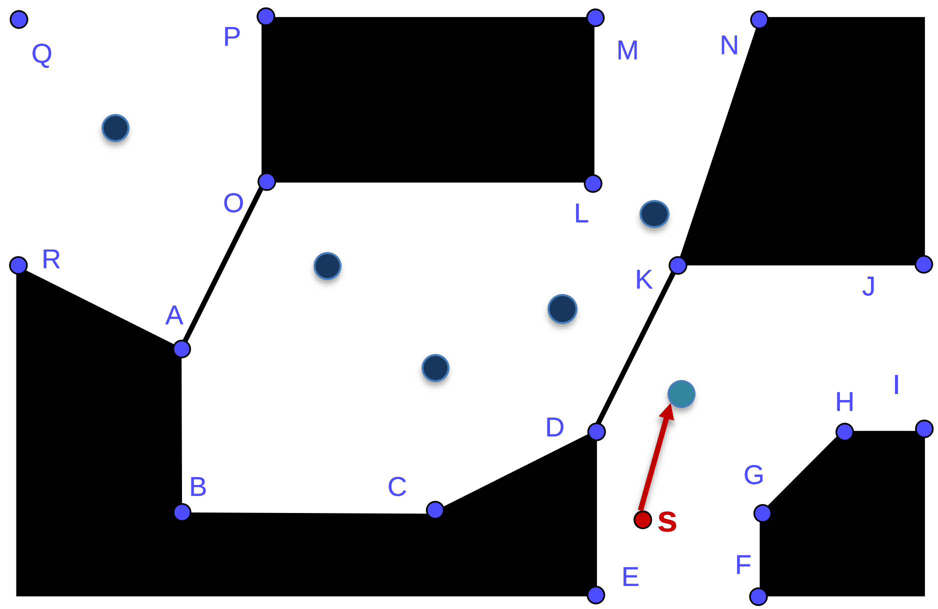
- Employs IR-tree to store the objects and incrementally retrieves nearest objects to the query location



[2] Zhao, S., Harabor, D. D., & Taniar, D. (2018). Faster and more robust mesh-based algorithms for obstacle k-nearest neighbour. *arXiv preprint arXiv:1808.04043*.

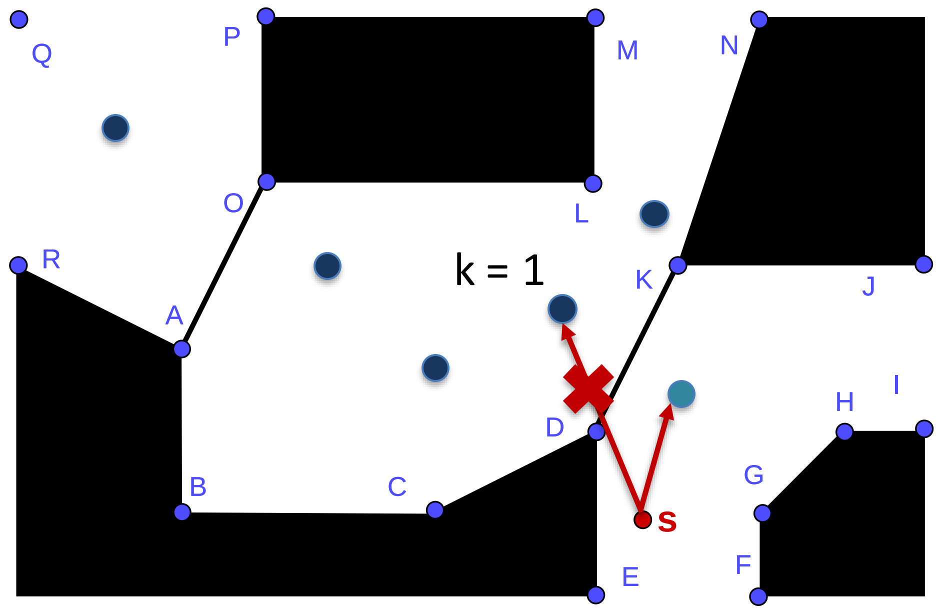
Previous work
IER

- Employs IR-tree to store the objects and incrementally retrieves nearest objects to the query location
- For each retrieved object, it calls Polyanya to compute its actual distance from the query



Previous work
IER

- Employs IR-tree to store the objects and incrementally retrieves nearest objects to the query location
- For each retrieved object, it calls Polyanya to compute its actual distance from the query
- Terminates when the Euclidean distance of next retrieved object is no smaller than the actual distances of kNNs



Outline

- Introduction
- Problem definition
- Previous work
  - Interval Heuristic Polyanya
  - Incremental Euclidean Restriction (IER)
- Grid Tree
  - Euclidean Hub Labeling
  - Grid Tree
- Results

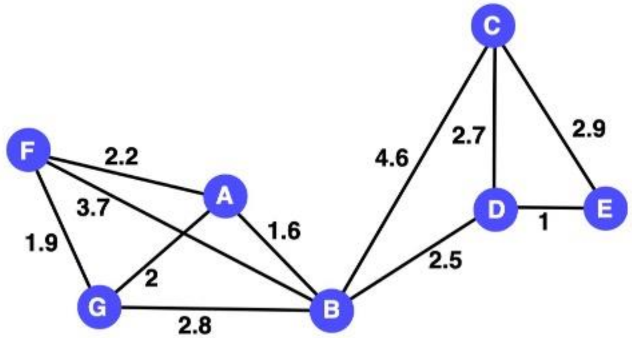
Grid Tree
Euclidean Hub Labeling [3]

- State-of-the-art Euclidean shortest pathfinding algorithm

[3] Du, J., Shen, B., & Cheema, M. A. (2023). Ultrafast Euclidean Shortest Path Computation Using Hub Labeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(10), 12417-12426. https://doi.org/10.1609/aaai.v37i10.26463

Grid Tree
Euclidean Hub Labeling

- State-of-the-art Euclidean shortest pathfinding algorithm
- Employs the popular hub labeling technique [4]



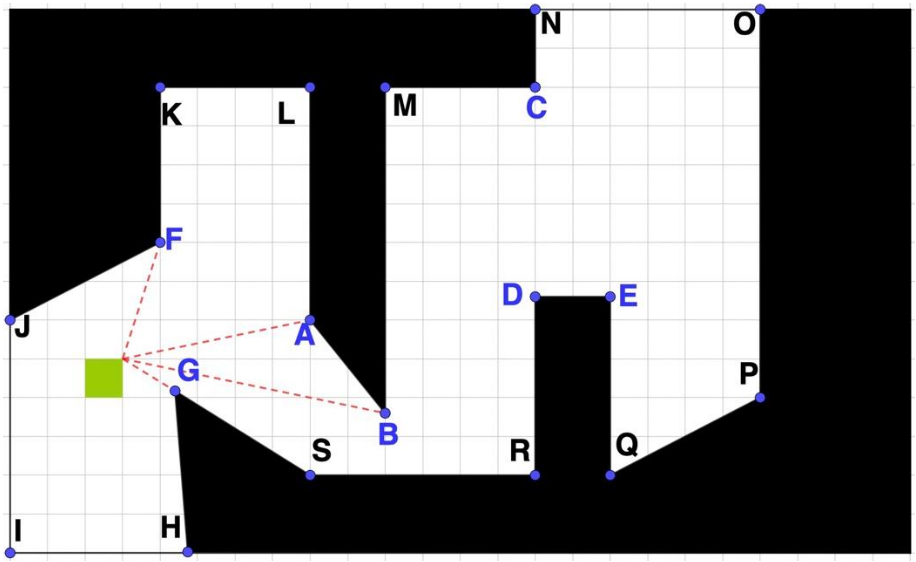
Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labels constructed for graph above

[4] Abraham, I., Delling, D., Goldberg, A. V., & Werneck, R. F. (2011). A hub-based labeling algorithm for shortest paths in road networks. In *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10* (pp. 230-241). Springer Berlin Heidelberg.

Euclidean Hub Labeling

- State-of-the-art Euclidean shortest pathfinding algorithm
- Employs the popular hub labeling technique
- Superimpose uniform grids across a given map, and store via labels in the grids to allow fast lookup and retrieval during query phase



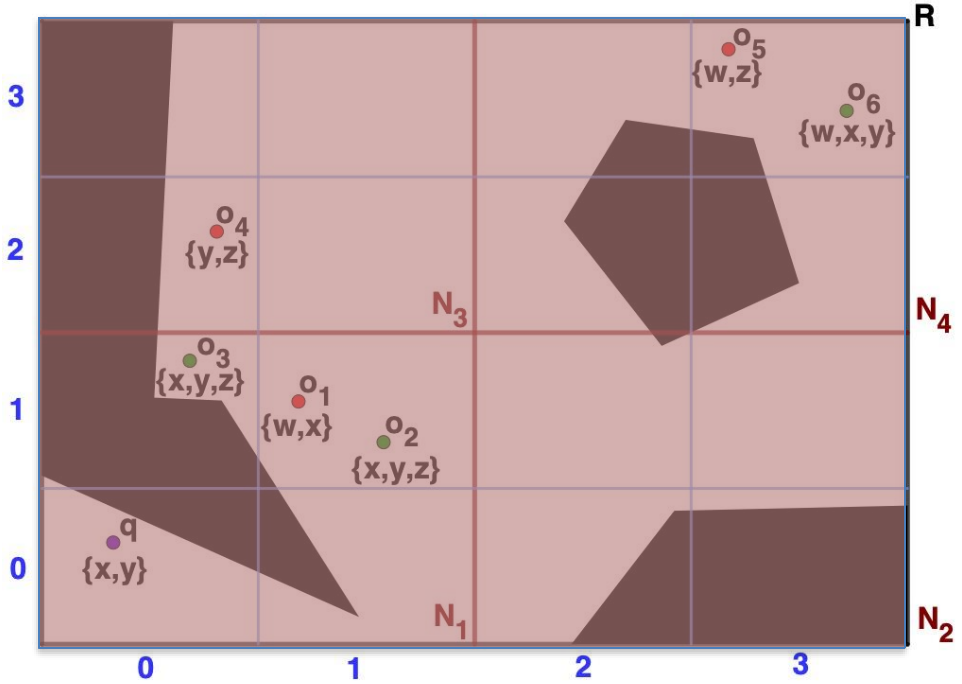
Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Hub nodes	Via labels
A	(A, 0), (B, 1.6), (G, 2), (F, 2.2)
B	(B, 0), (G, 2.8), (F, 3.7)
F	(F, 0), (G, 1.9)
G	(G, 0)

Table 2: Hub labels and via labels for the figure above

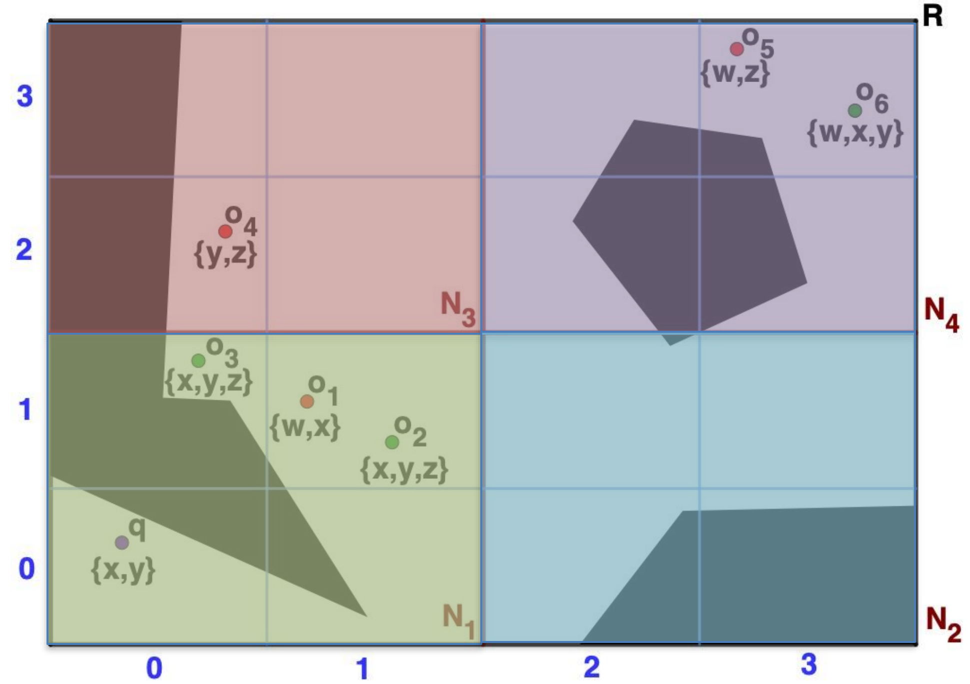
Grid Tree Structure

- Starting from root node, recursively divides each node into 4 equal sized children until size of each child node is smaller than a threshold



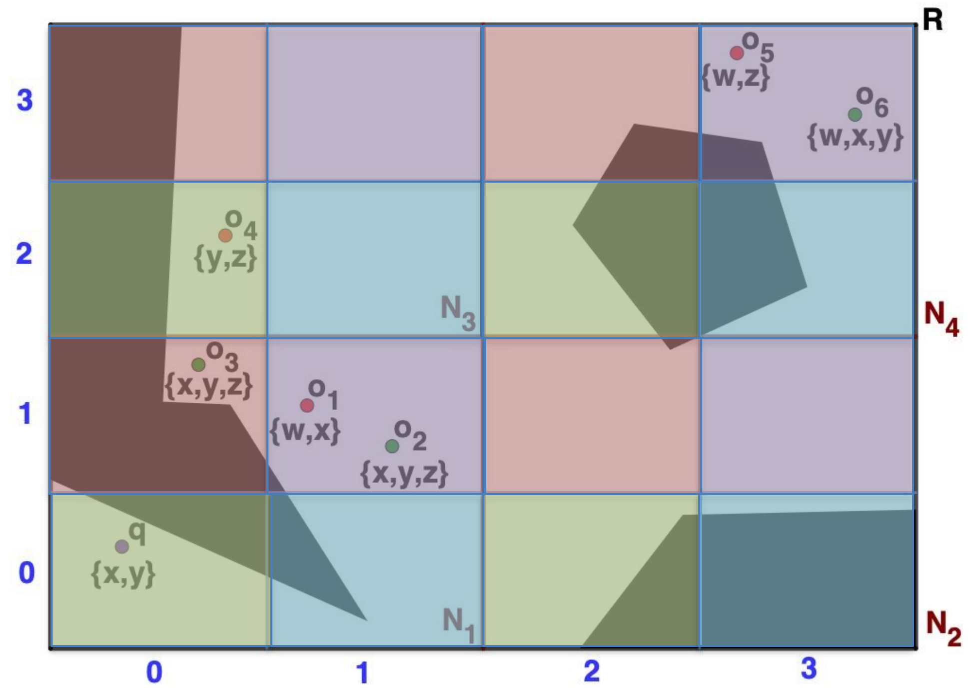
Grid Tree Structure

- Starting from root node, recursively divides each node into 4 equal sized children until size of each child node is smaller than a threshold



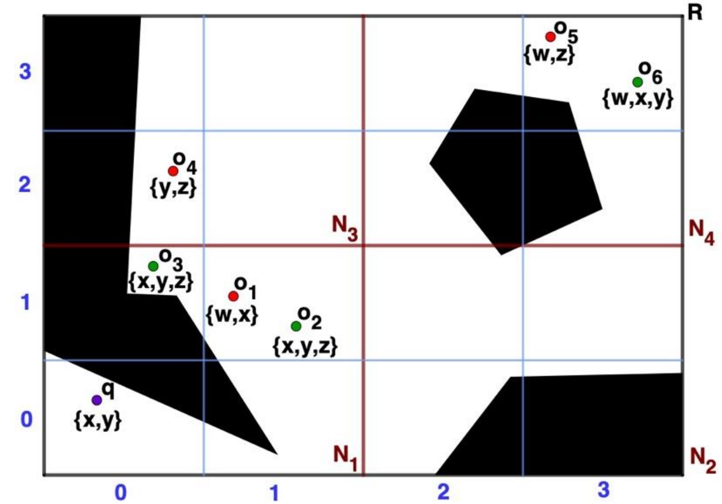
Grid Tree Structure

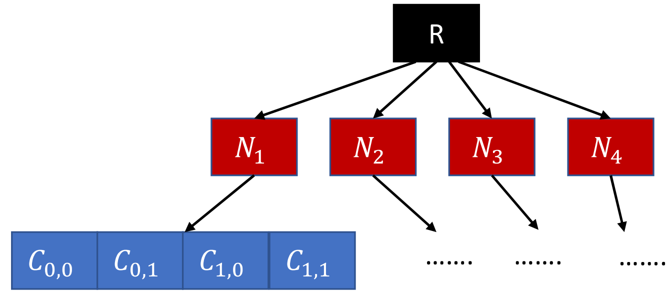
- Starting from root node, recursively divides each node into 4 equal sized children until size of each child node is smaller than a threshold



Grid Tree Structure

- Starting from root node, recursively divides each node into 4 equal sized children until size of each child node is smaller than a threshold
- Each node stores an object list containing the IDs of the objects that are located inside it. It also stores a keyword list containing unique keywords with its frequency





Object Lists

Cell	Objects
$C_{1,1}$	o_1, o_2
$C_{0,1}$	o_3
$C_{0,2}$	o_4
$C_{3,3}$	o_5, o_6

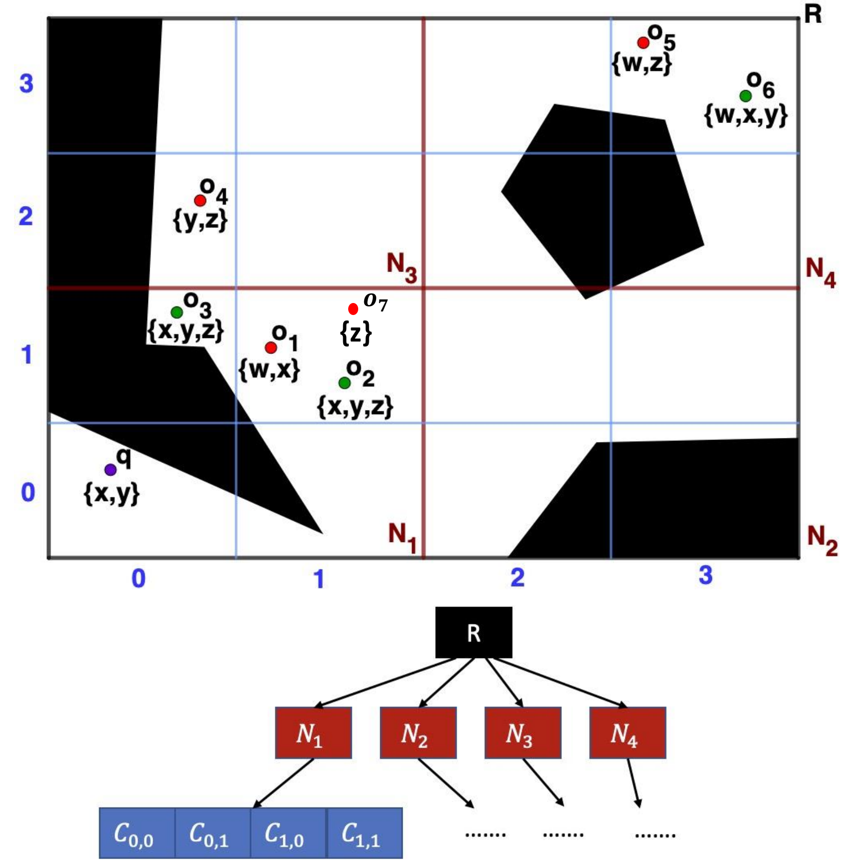
Keyword Lists of some nodes

Node	Keywords
R	w: 3, x: 4, y: 4, z: 4
N_1	w: 1, x: 3, y: 2, z: 2
$C_{1,1}$	w: 1, x: 2, y: 1, z: 1
$C_{0,1}$	x: 1, y: 1, z: 1

Grid Tree

Update

- Insert



Object Lists

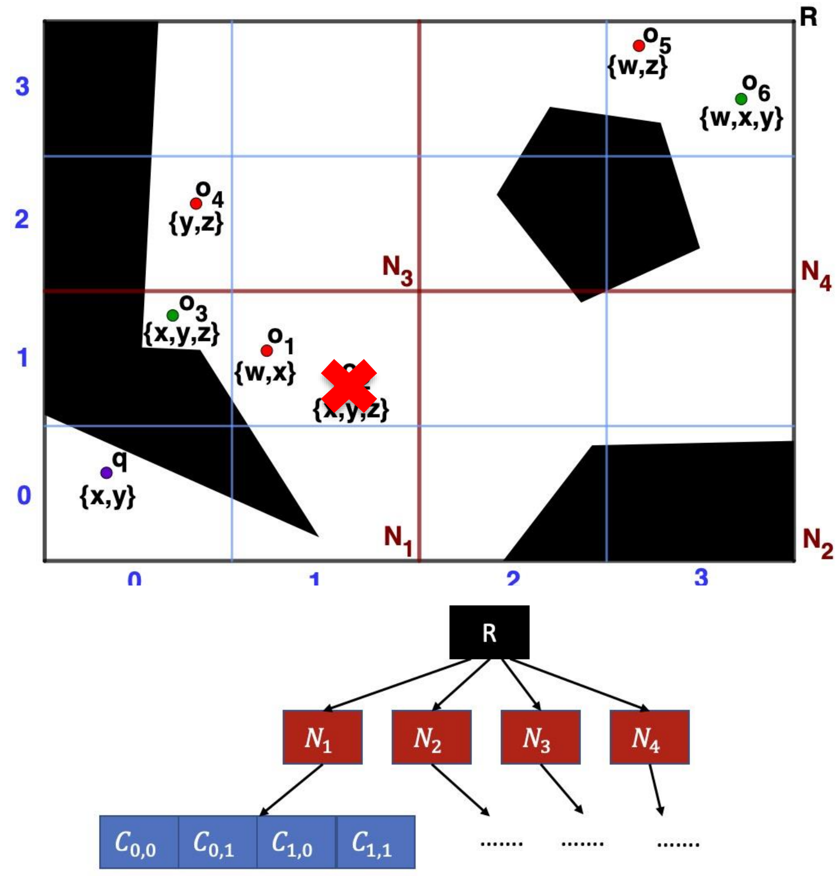
Cell	Objects
$C_{1,1}$	o_1, o_2, o_7
$C_{0,1}$	o_3
$C_{0,2}$	o_4
$C_{3,3}$	o_5, o_6

Keyword Lists of some nodes

Node	Keywords
R	w: 3, x: 4, y: 4, z: 4
N_1	w: 1, x: 3, y: 2, z: 2
$C_{1,1}$	w: 1, x: 2, y: 1, z: 2
$C_{0,1}$	x: 1, y: 1, z: 1

Grid Tree
Update

- Insert
- Delete



Object Lists

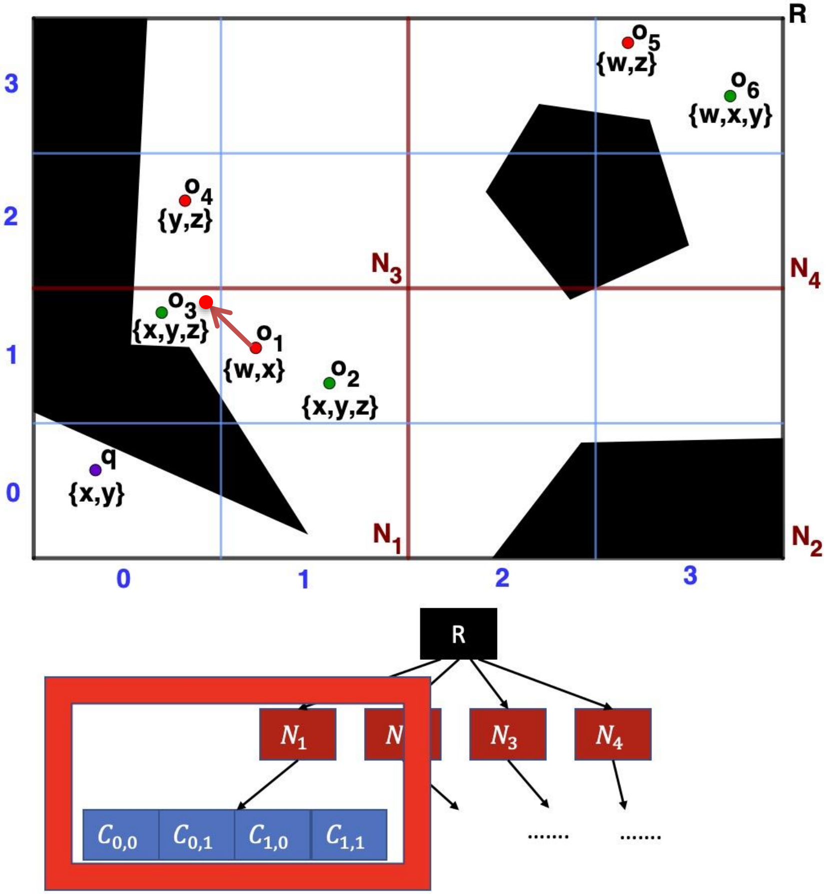
Cell	Objects
$C_{1,1}$	$o_1, \cancel{o_2}$
$C_{0,1}$	o_3
$C_{0,2}$	o_4
$C_{3,3}$	o_5, o_6

Keyword Lists of some nodes

Node	Keywords
R	w: 3, x: 4, y: 4, z: 4
N_1	w: 1, x: 3, y: 2, z: 2
$C_{1,1}$	w: 1, x: 1
$C_{0,1}$	x: 1, y: 1, z: 1

Grid Tree
Update

- Insert
- Delete
- Location change of an object



Object Lists

Cell	Objects
$C_{1,1}$	~~o_1~~, o_2
$C_{0,1}$	o_3, o_1
$C_{0,2}$	o_4
$C_{3,3}$	o_5, o_6

Keyword Lists of some nodes

Node	Keywords
R	w: 3, x: 4, y: 4, z: 4
N_1	w: 1, x: 3, y: 2, z: 2
$C_{1,1}$	x: 1, y: 1, z: 1
$C_{0,1}$	w: 1, x: 2, y: 1, z: 1

Grid Tree
Query

- Query is similar to the standard best-first search algorithm on trees

Grid Tree
Query

- Query is similar to the standard best-first search algorithm on trees
- The grid tree is traversed based on the minimum distances of the nodes and the nodes that do not contain query keywords are removed

Grid Tree
Query

- Query is similar to the standard best-first search algorithm on trees
- The grid tree is traversed based on the minimum distances of the nodes and the nodes that do not contain query keywords are removed
- Euclidean Hub Labeling (EHL) is used to compute shortest distance from the query to the objects containing query keywords

Grid Tree

Query algorithm

Algorithm 1: Boolean kNN query processing

Input: $q.\rho, q.\tau, k$: query location, query keywords and k
Output: R: query results

$R = \phi$; $d^k = \infty$;
Initialise a min-heap H with the root node of Grid Tree ;
while $H \neq \phi$ **do**
  deheap an entry e from H ;
  if $e.key \geq d^k$ **then**
    return R;
  if e *is an object* **then**
    compute $d(q.\rho, e.\rho)$;
    if $d(q.\rho, e.\rho) < d^k$ **then**
      update R and d^k by object e;
  else if e *is a leaf node* **then**
    for *each object* o_i^t *in the object list of* e **do**
      if $q.\tau \subseteq o_i^t.\tau$ **then**
        insert o_i^t in H with key $mindist(q.\rho, o_i^t.\rho)$;
  else
    for *each child node* c *of* e **do**
      if c *contains all query keywords* $q.\tau$ **then**
        insert c in H with key $mindist(q.\rho, c)$;
return R;

Extensions

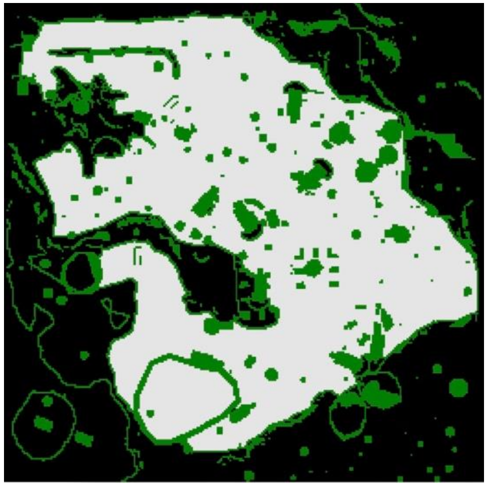
- Generalisation of boolean kNN query
- Top-k spatial keyword query
- Keyword range query

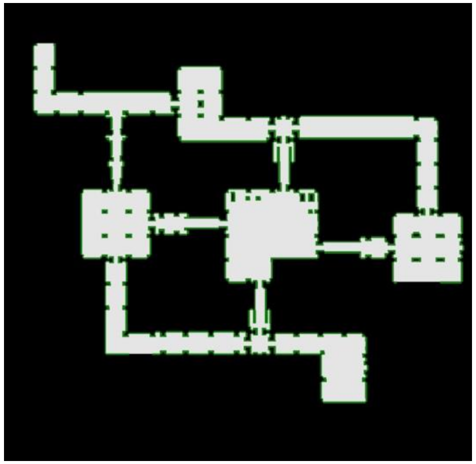
Outline

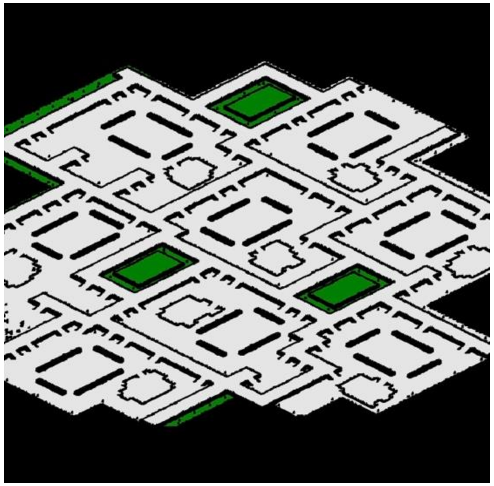
- Introduction
- Problem definition
- Previous work
  - Interval Heuristic Polyanya
  - Incremental Euclidean Restriction (IER)
- Grid Tree
  - Euclidean Hub Labeling
  - Grid Tree
- Results

Experimental setup

- Benchmark maps: widely used game map benchmarks [5]





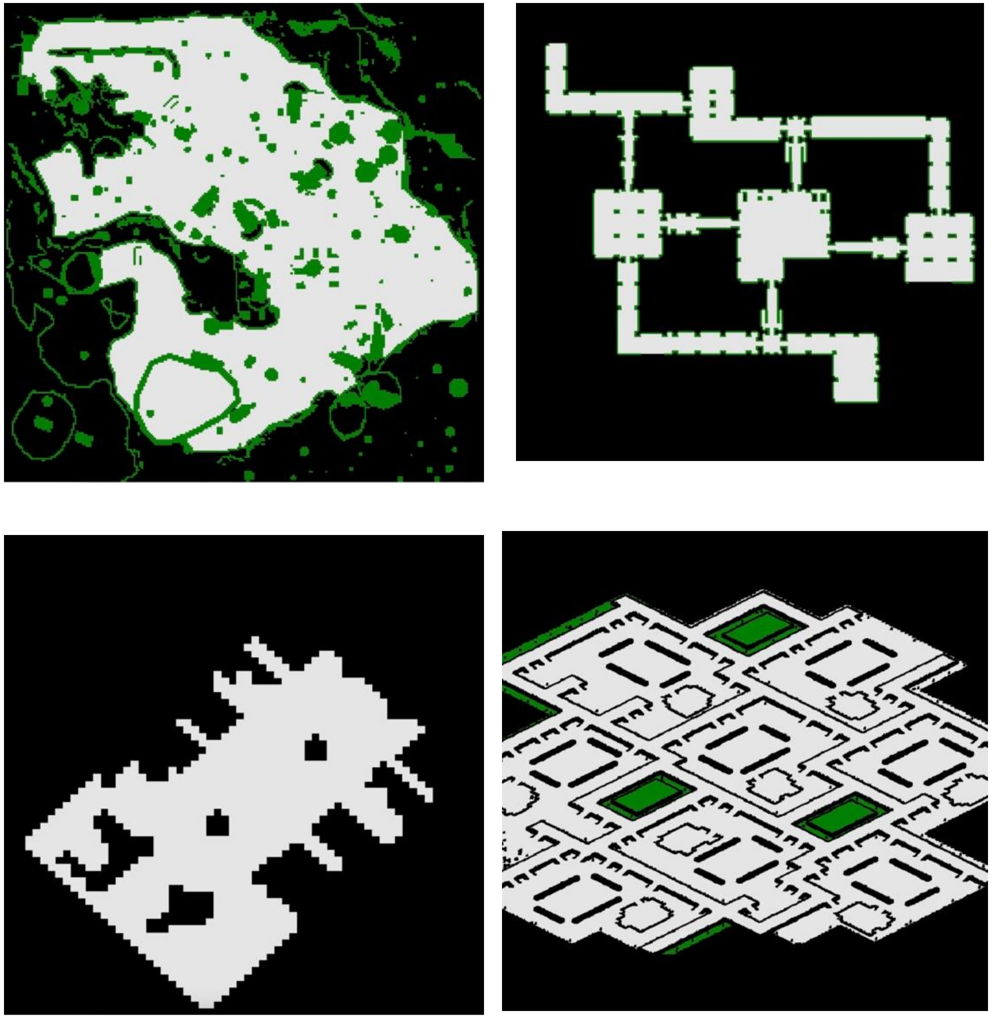


Game	#Maps	# Cells	# Trav. Cells	# Vertices
DA	67	151,420	15,911	1182.9
DAO	156	134,258	21,322	1727.6
BG	75	262,144	73,930	1294.4
SC	75	446,737	263,782	11487.5

Table 1: Total number of maps, and average number of total cells, traversable cells and vertices in each benchmark.

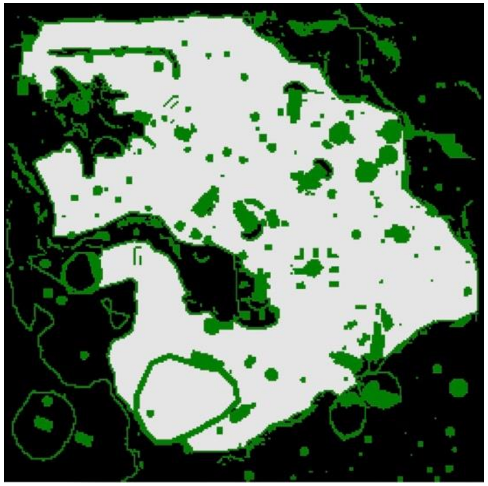
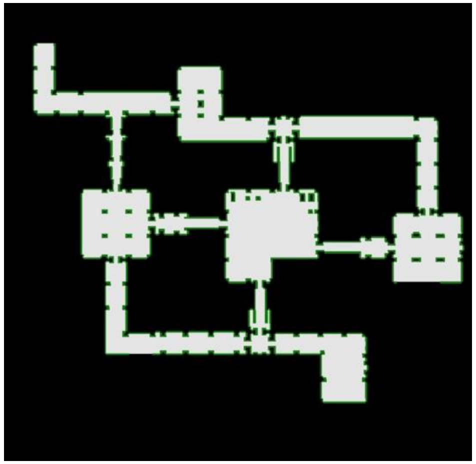
Experimental setup

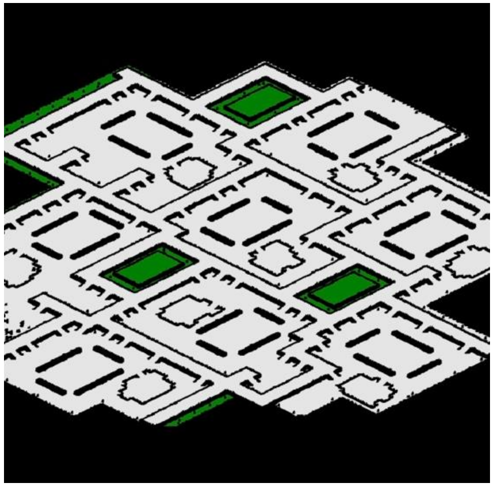
- Benchmark maps: widely used game map benchmarks [5]
- Object density: 0.1%, 1% and 10%



Results
Experimental setup

- Benchmark maps: widely used game map benchmarks [5]
- Object density: 0.1%, 1% and 10%
- Keyword: 100 item descriptions obtained from ChatGPT for each benchmark and extracted keywords from it

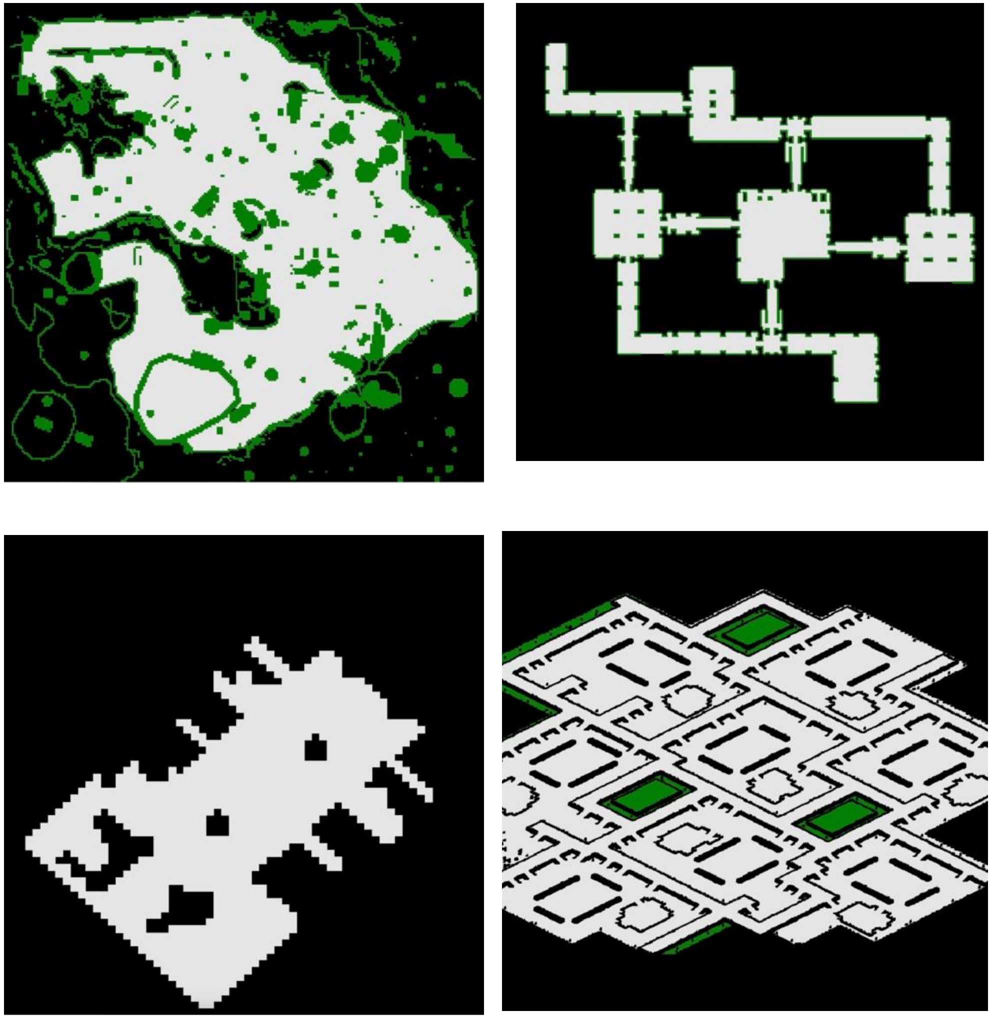





Results

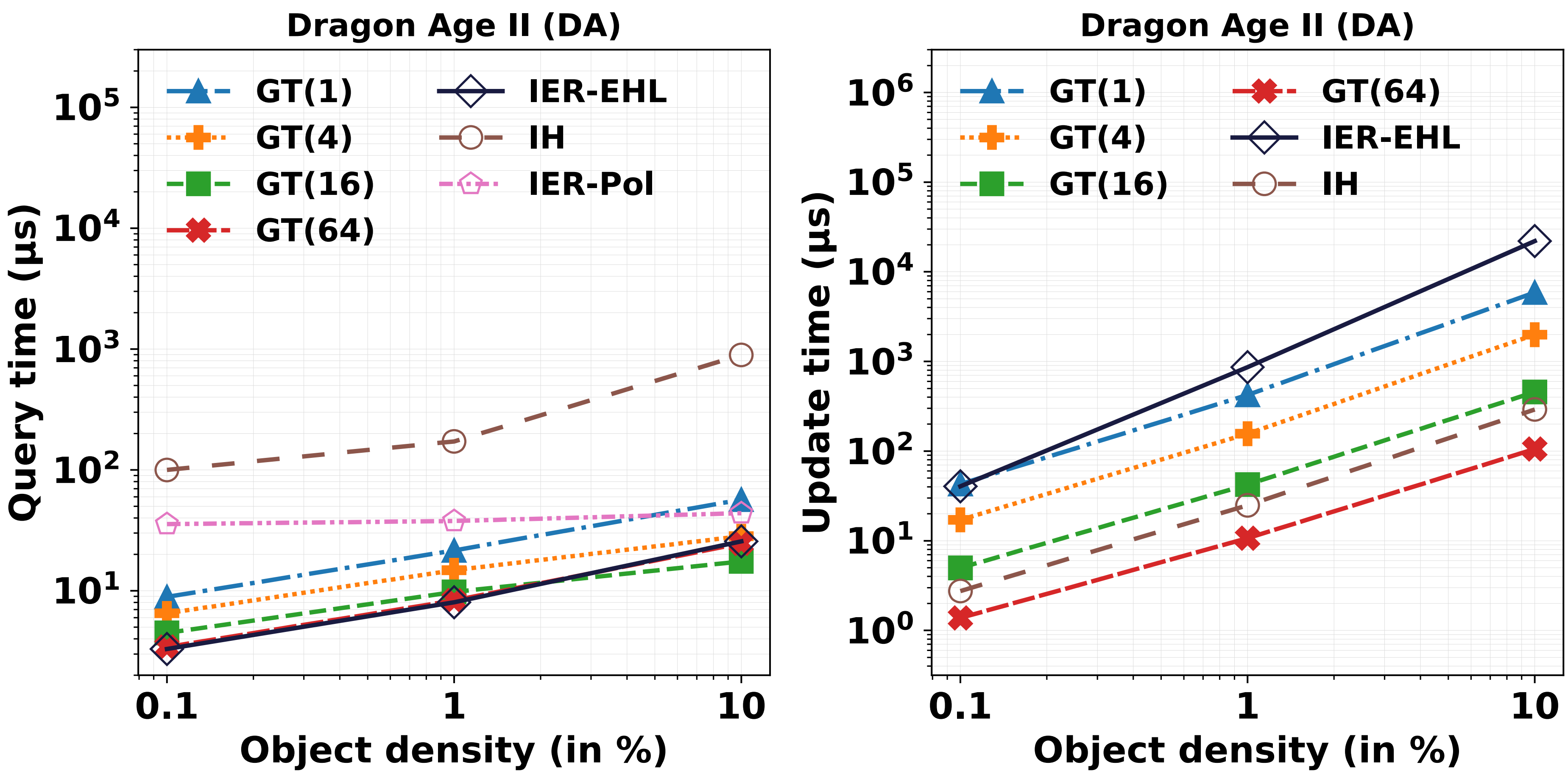
Experimental setup

- Benchmark maps: widely used game map benchmarks [5]
- Object density: 0.1%, 1% and 10%
- Keyword: 100 item descriptions obtained from ChatGPT for each benchmark and extracted keywords from it
- Query: randomly generate 100 queries per timestamp for 50 timestamps



Results

Effect of object density



Effect of object density on query time and update time for DA on default settings (k=3, mobility = 70%, # of query keywords = 2)

Other experiments

- Mobility of objects
- k
- Number of query keywords
- Object distribution

Thank you for listening ☺

References

[1] Zhao, S., Taniar, D., & Harabor, D. (2018). Fast k-nearest neighbor on a navigation mesh. In *Proceedings of the International Symposium on Combinatorial Search* (Vol. 9, No. 1, pp. 124-131).

[2] Zhao, S., Harabor, D. D., & Taniar, D. (2018). Faster and more robust mesh-based algorithms for obstacle k-nearest neighbour. *arXiv preprint arXiv:1808.04043*.

[3] Du, J., Shen, B., & Cheema, M. A. (2023). Ultrafast Euclidean Shortest Path Computation Using HubLabeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(10), 12417-12426. https://doi.org/10.1609/aaai.v37i10.26463

[4] Abraham, I., Delling, D., Goldberg, A. V., & Werneck, R. F. (2011). A hub-based labeling algorithm for shortest paths in road networks. In *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10* (pp. 230-241). Springer Berlin Heidelberg.

Image sources:

https://www.geospatialworld.net/news/sensewheres-indoor-location-volume-hits-50-billion-requests/

https://www.mecalux.com/blog/fully-automated-warehouse

https://www.theverge.com/2022/2/22/22945814/amazon-astro-home-robot-photo-video