Slides adapted from Cyrill Stachniss, Michael Kaess, S.Scherer

Map Representations

Sanjiban Choudhury

TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

Announcements

Deadline for lab1 extended to Wednesday 4/30 at 11:59 p.m

This is the due date for the writeup

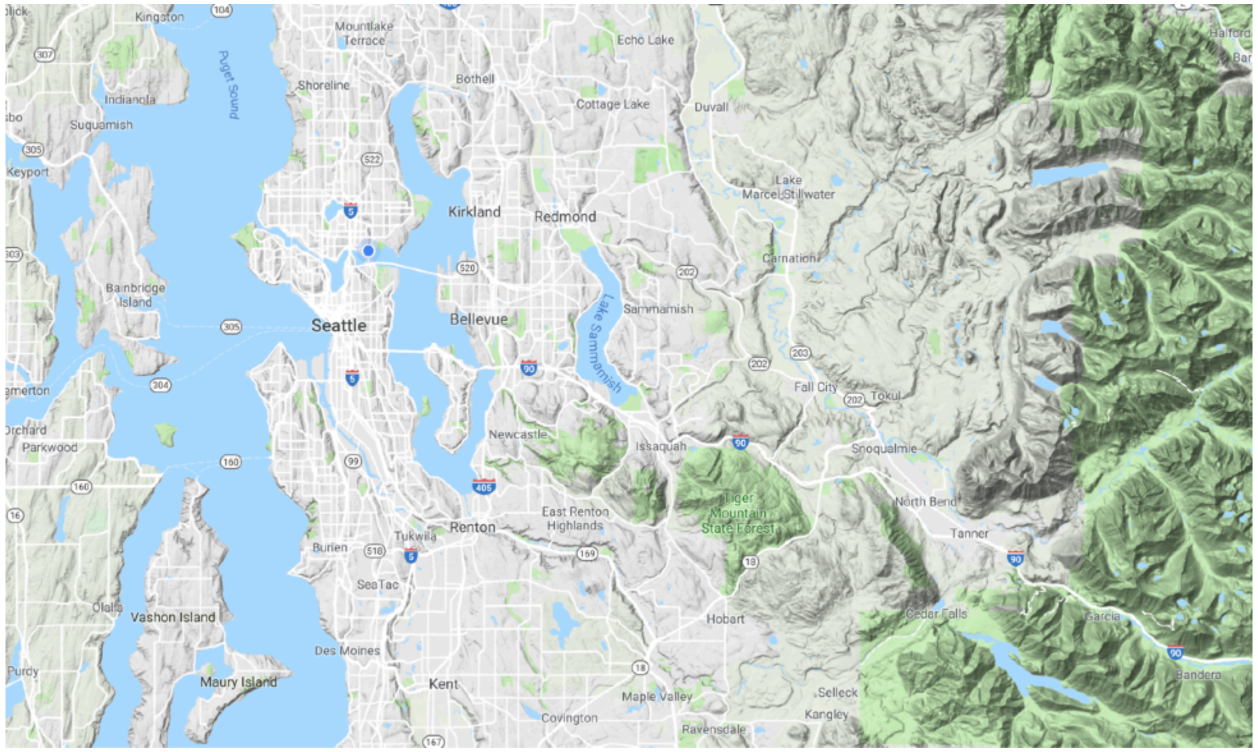
The lab evaluation is still on Thursday 4/25
from 9.00 a.m - 12:00 p.m

Please continue to update blogs by Friday of each week

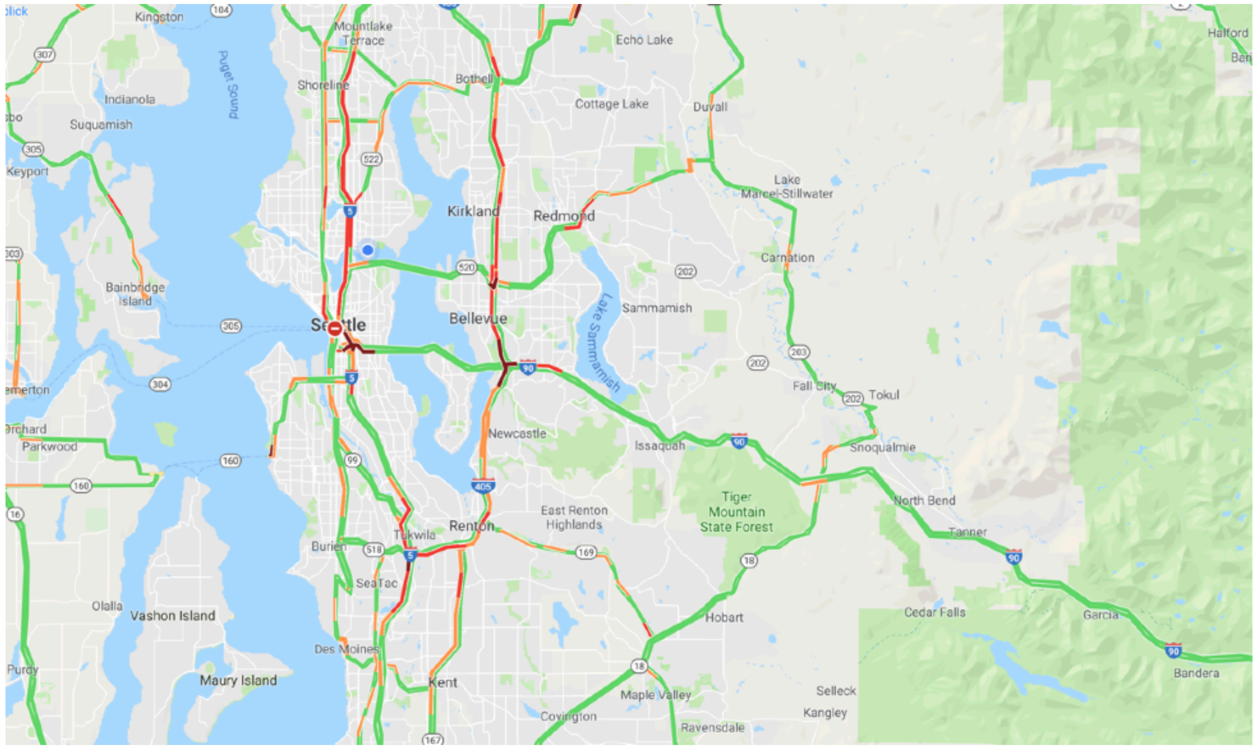
What is a map?

Do all maps convey the same information?

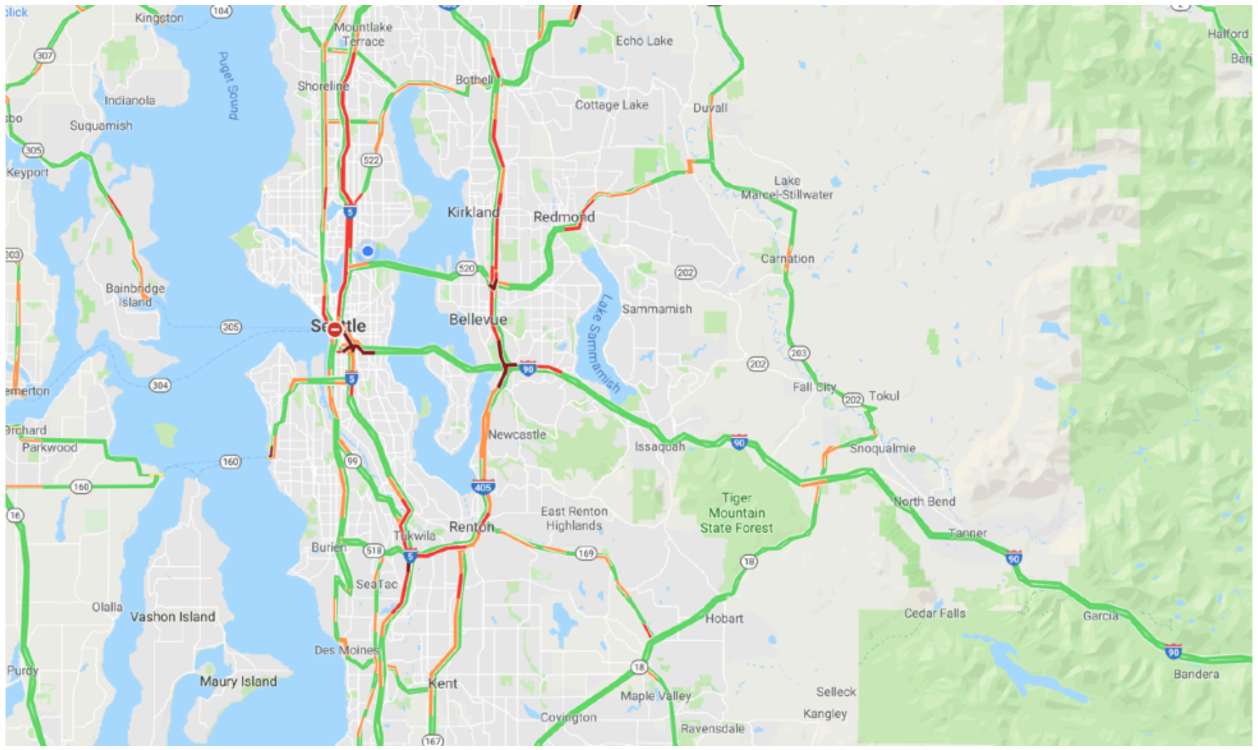
Do all maps convey the same information?

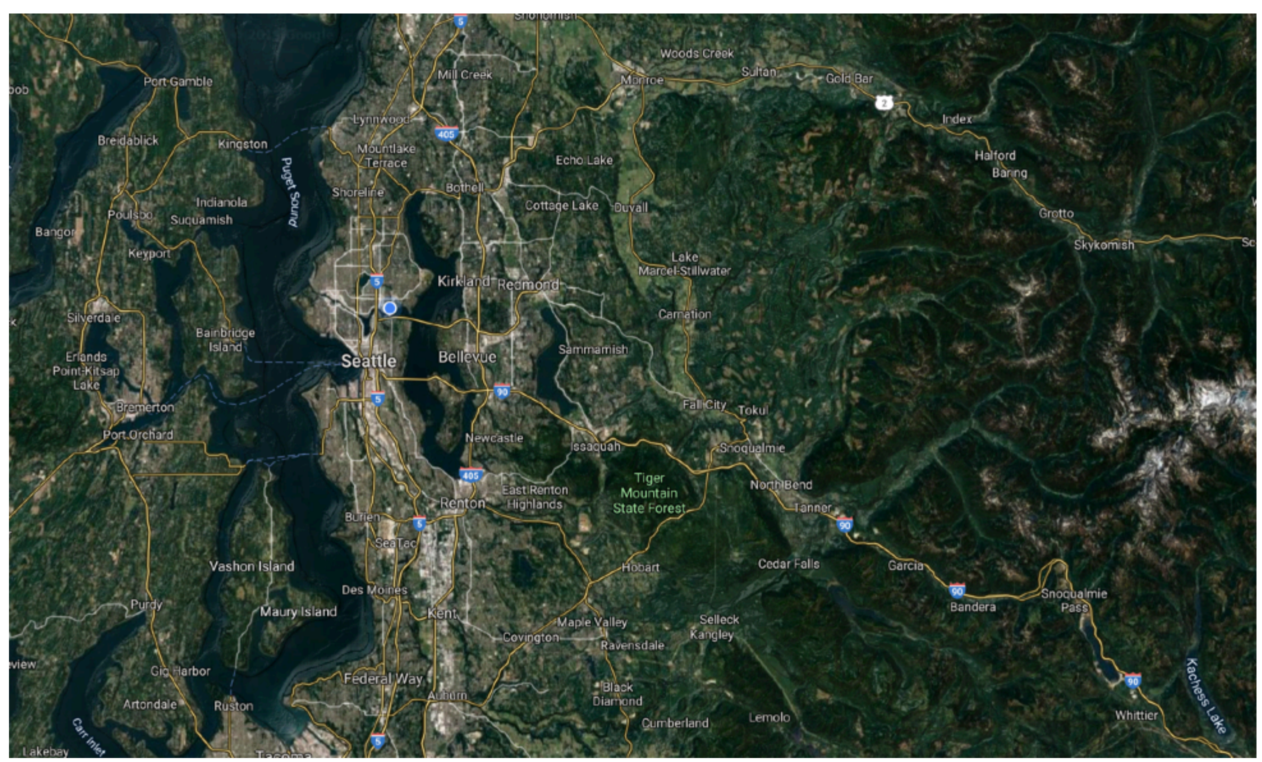


Do all maps convey the same information?



Do all maps convey the same information?





Maps are a summary of information about the world

Maps are a summary of information about the world

What sort of information? Depends on the task

Maps are a summary of information about the world

What sort of information? Depends on the task

Task also determines how we
query, update, store maps

Today's objective

1. Framework / taxonomy to think about maps

2. Look at various maps and the underlying tasks they serve

3. Distance map

What do we want from maps?

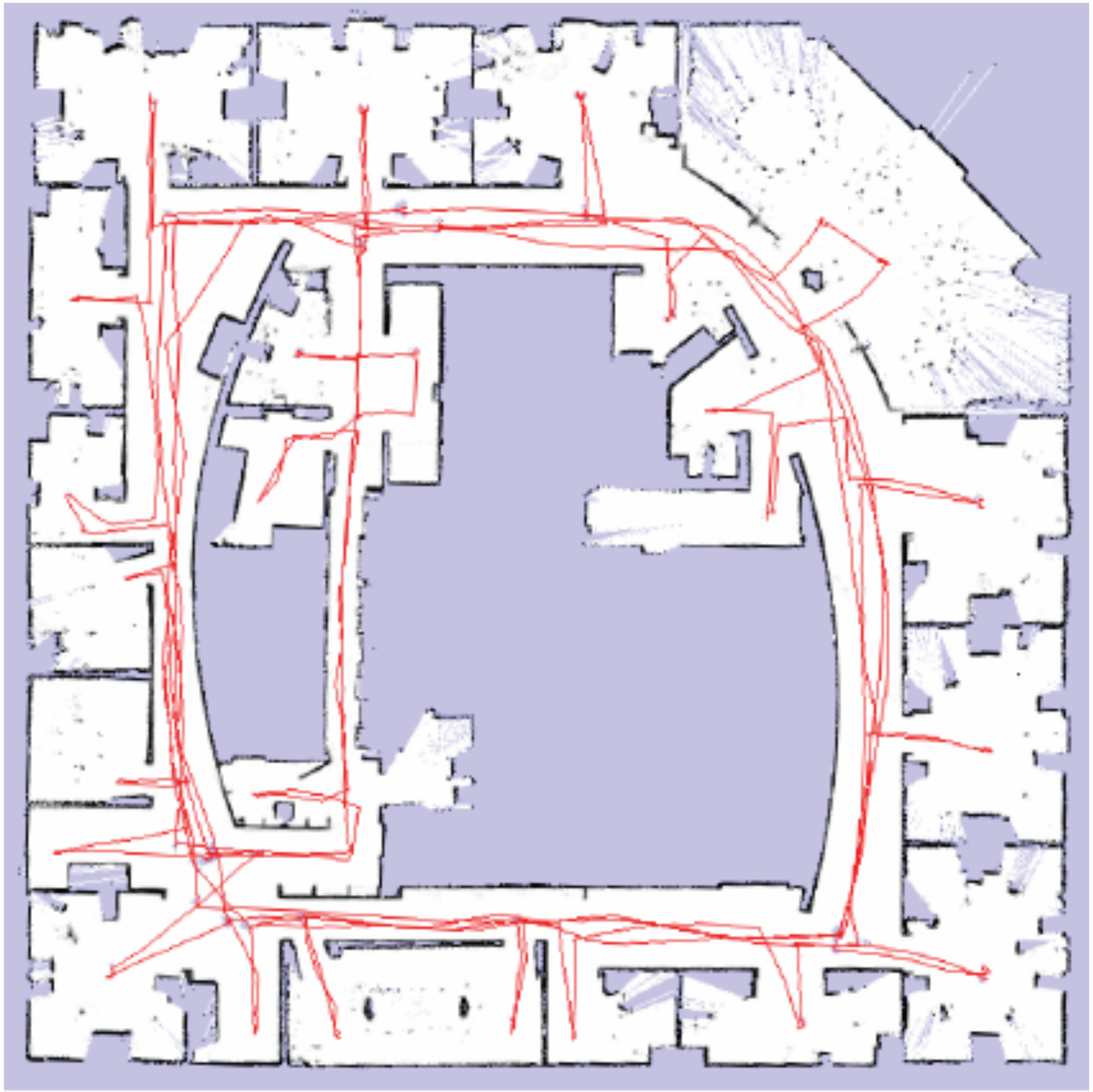
1. Information - What task does it help me solve? (Help me localize, help me navigate, help humans navigate / plan their lives etc)

2. Query - Can we query it online? How often?

3. Updatable - Can we update it online? Can it deal with noisy measurements?

4. Memory - How much storage does it need? Is it transportable? How does it scale with time? Scale with amount of stuff we see ?

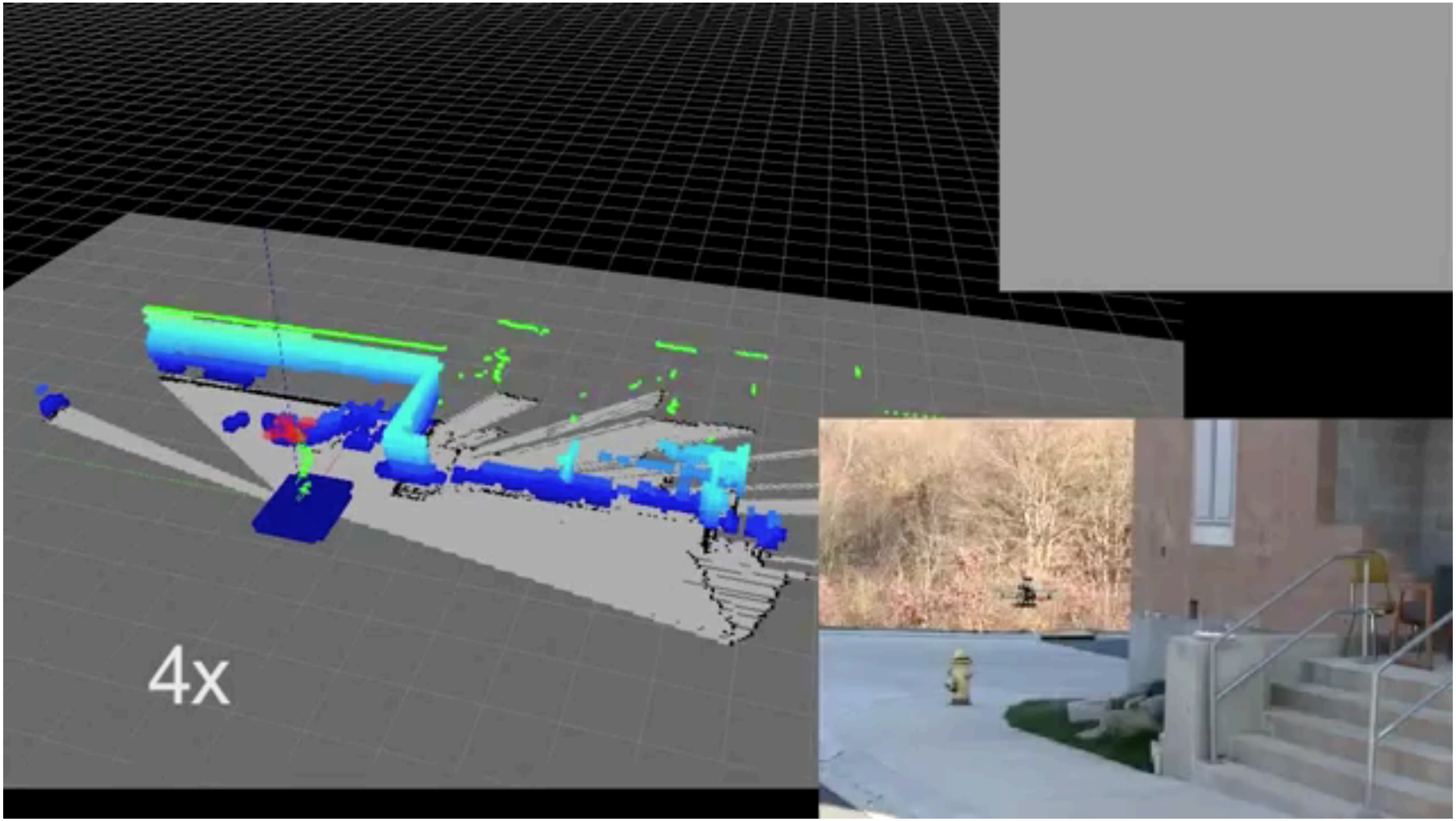
Example 1: Occupancy grids



Example 1: Occupancy grids

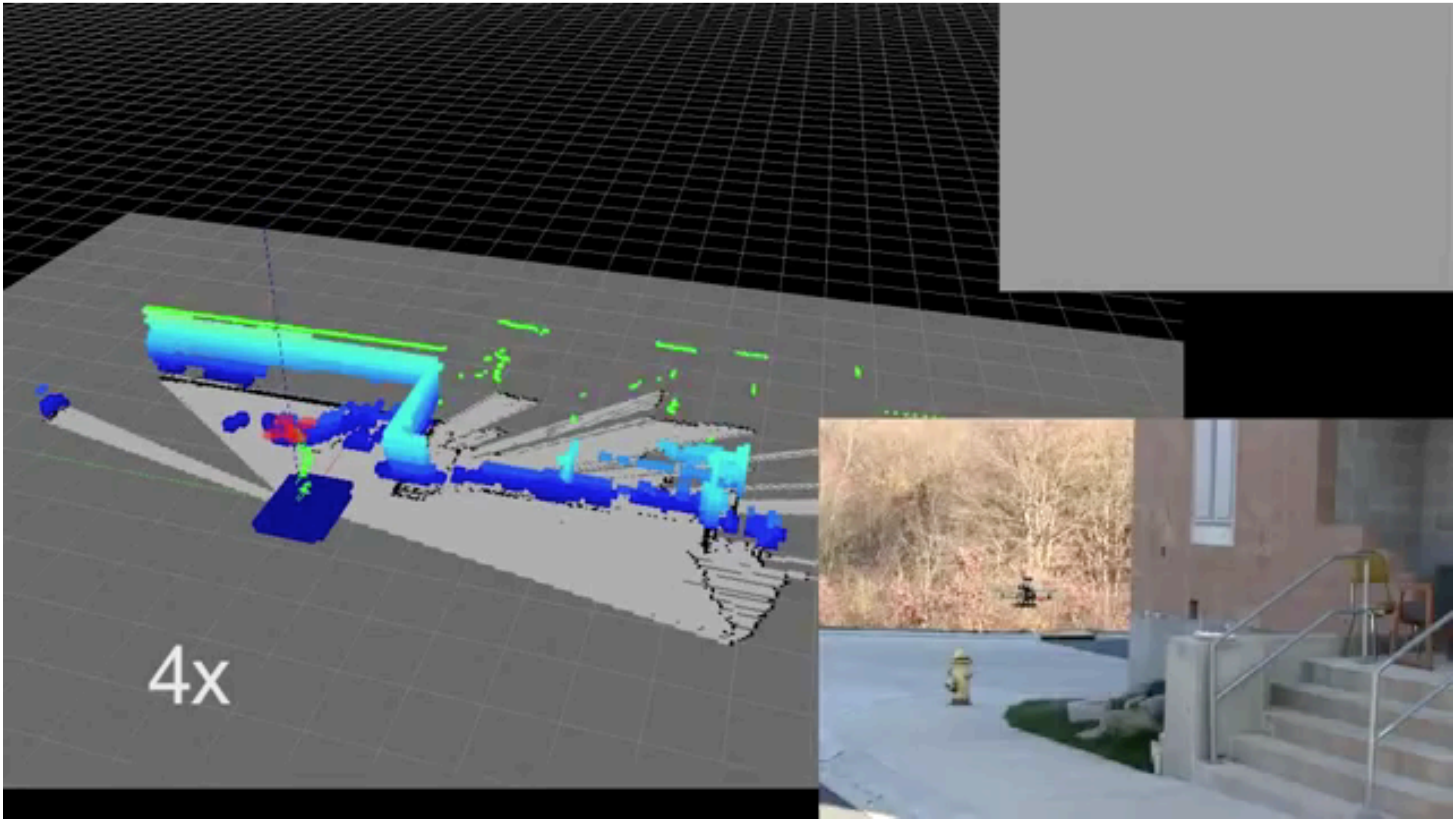
Category	Details
Information	
Query	
Update	
Memory	

Occupancy grids in action



"Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV", S. Shen, N. Michael, V.Kumar, 2010

Occupancy grids in action



"Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV", S. Shen, N. Michael, V.Kumar, 2010

Example 1: Occupancy grids

Category	Details
Information	
Query	
Update	
Memory	

Example 1: Occupancy grids

Category	Details
Information	Discretized likelihood of occupancy (free/occ/unknown) Useful for exploration (go to unknown areas) Useful for safe navigation (keep robot in known free space)
Query	
Update	
Memory	

Example 1: Occupancy grids

Category	Details
Information	Discretized likelihood of occupancy (free/occ/unknown) Useful for exploration (go to unknown areas) Useful for safe navigation (keep robot in known free space)
Query	Cheap: O(1)
Update	
Memory	

Example 1: Occupancy grids

Category	Details
Information	Discretized likelihood of occupancy (free/occ/unknown) Useful for exploration (go to unknown areas) Useful for safe navigation (keep robot in known free space)
Query	Cheap: O(1)
Update	Can deal with noisy sensors (log likelihood update) Updates equal ray-casting (O(l) where l is length of ray)
Memory	

Example 1: Occupancy grids

Category	Details
Information	Discretized likelihood of occupancy (free/occ/unknown) Useful for exploration (go to unknown areas) Useful for safe navigation (keep robot in known free space)
Query	Cheap: O(1)
Update	Can deal with noisy sensors (log likelihood update) Updates equal ray-casting (O(l) where l is length of ray)
Memory	Bounded Can still be large if we want really fine resolution Need to allocate all the memory upfront

Problems with occupancy grids

Problems with occupancy grids

1. Memory scales with distance travelled in any one direction

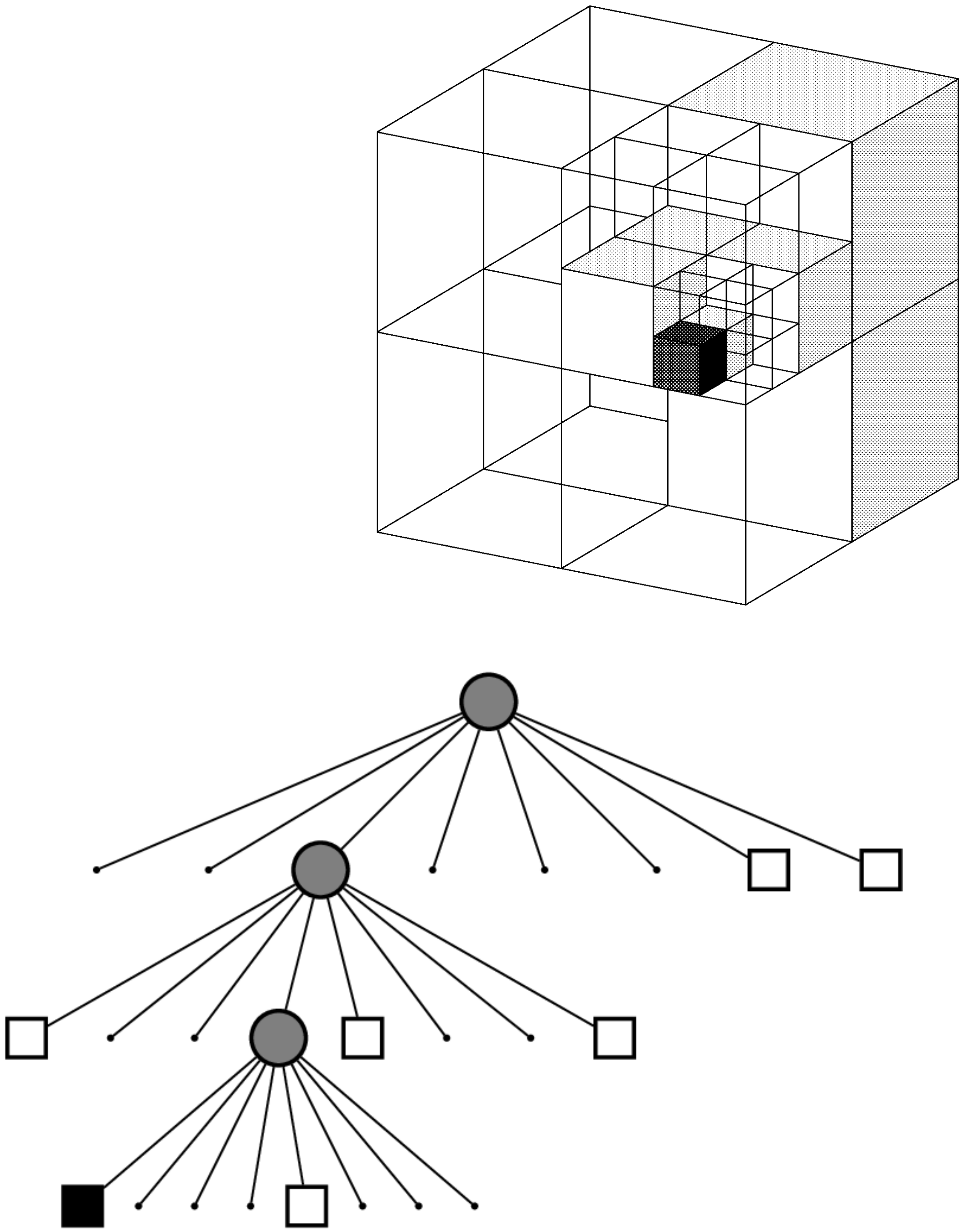
Problems with occupancy grids

1. Memory scales with distance travelled in any one direction

2. Do I need high resolution information everywhere?

Example 2: Occupancy Trees (OctoMap)

Hornung et al. 2013



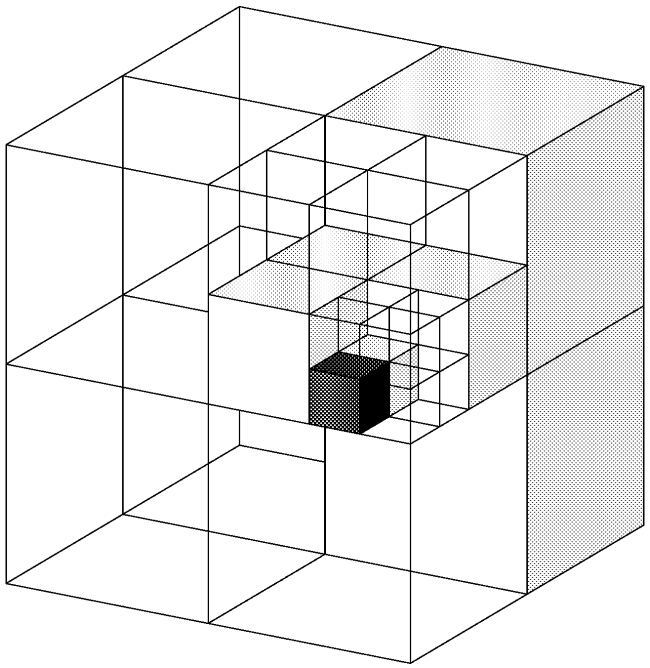
Tree-based data structure

Recursive sub-division of space

Query maps at multiple-resolutions!

Example 2: Occupancy Trees (OctoMap)

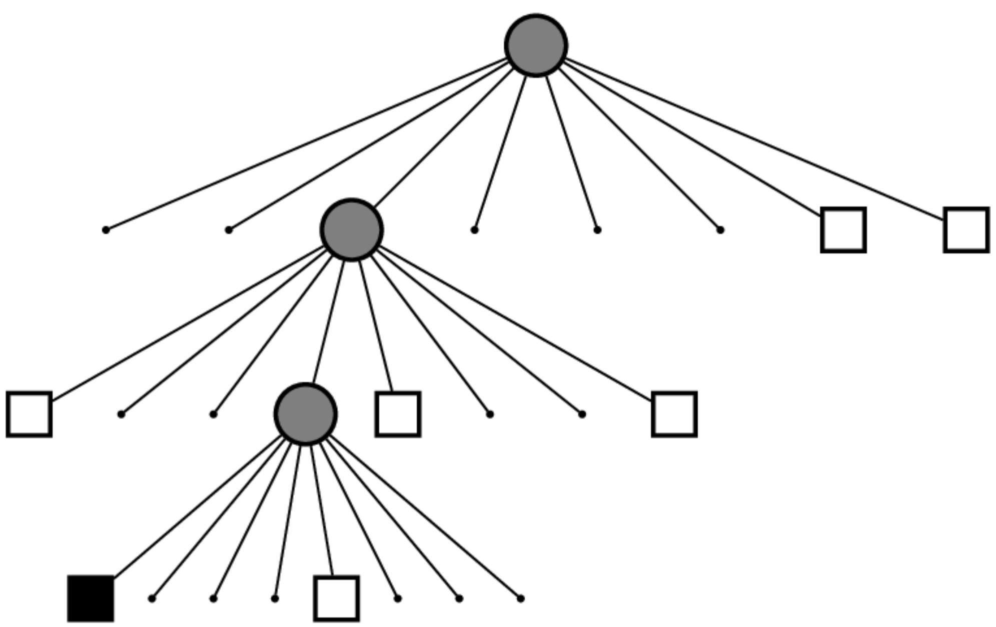
Hornung et al. 2013

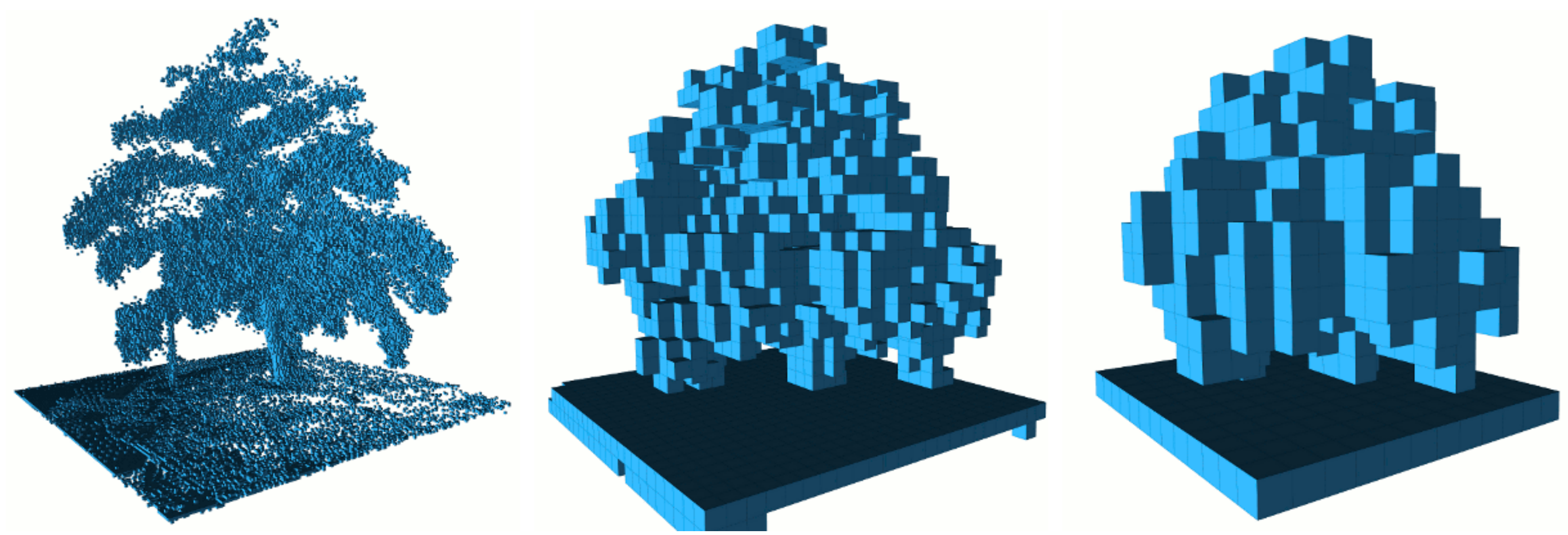


Tree-based data structure

Recursive sub-division of space

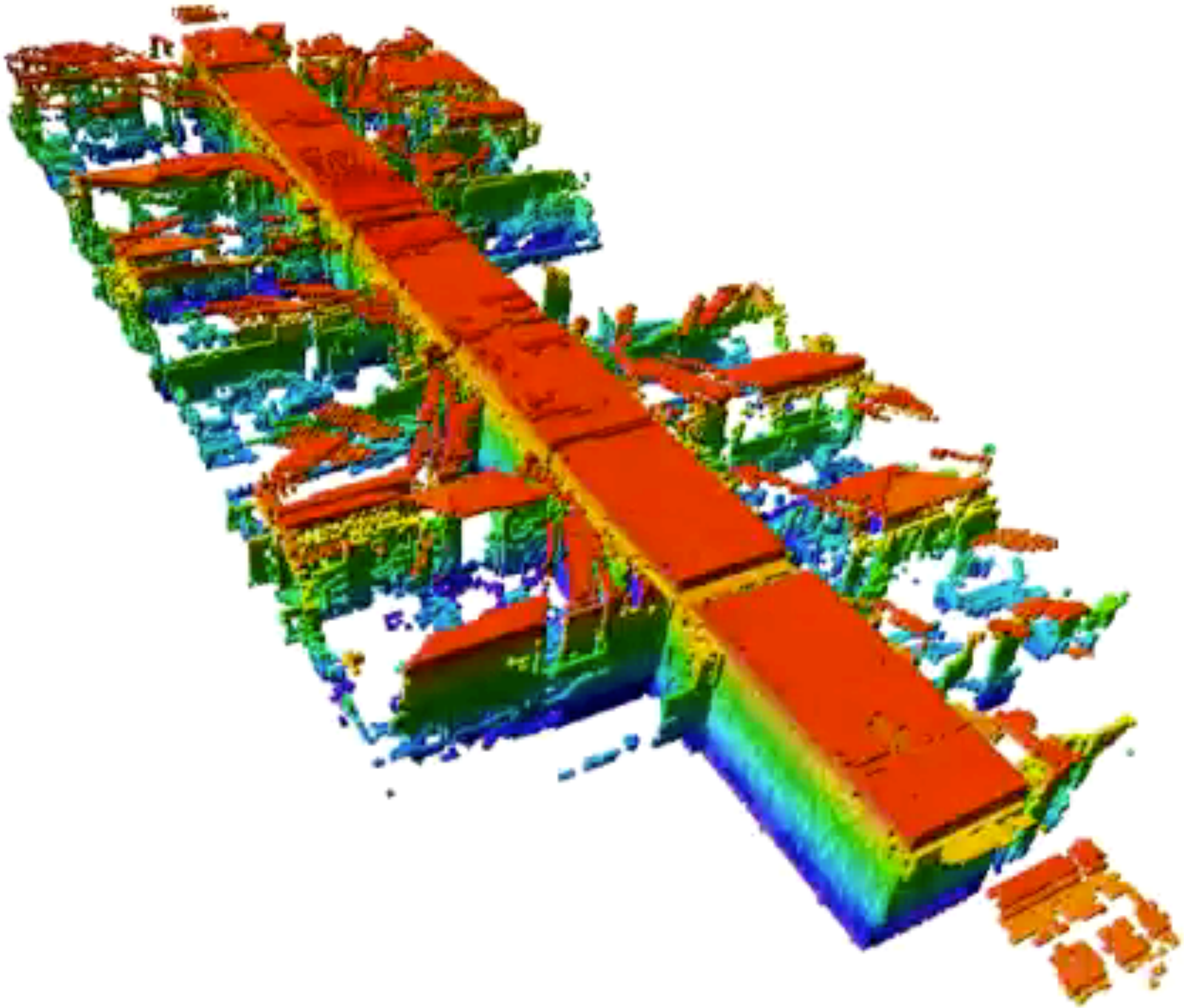
Query maps at multiple-resolutions!



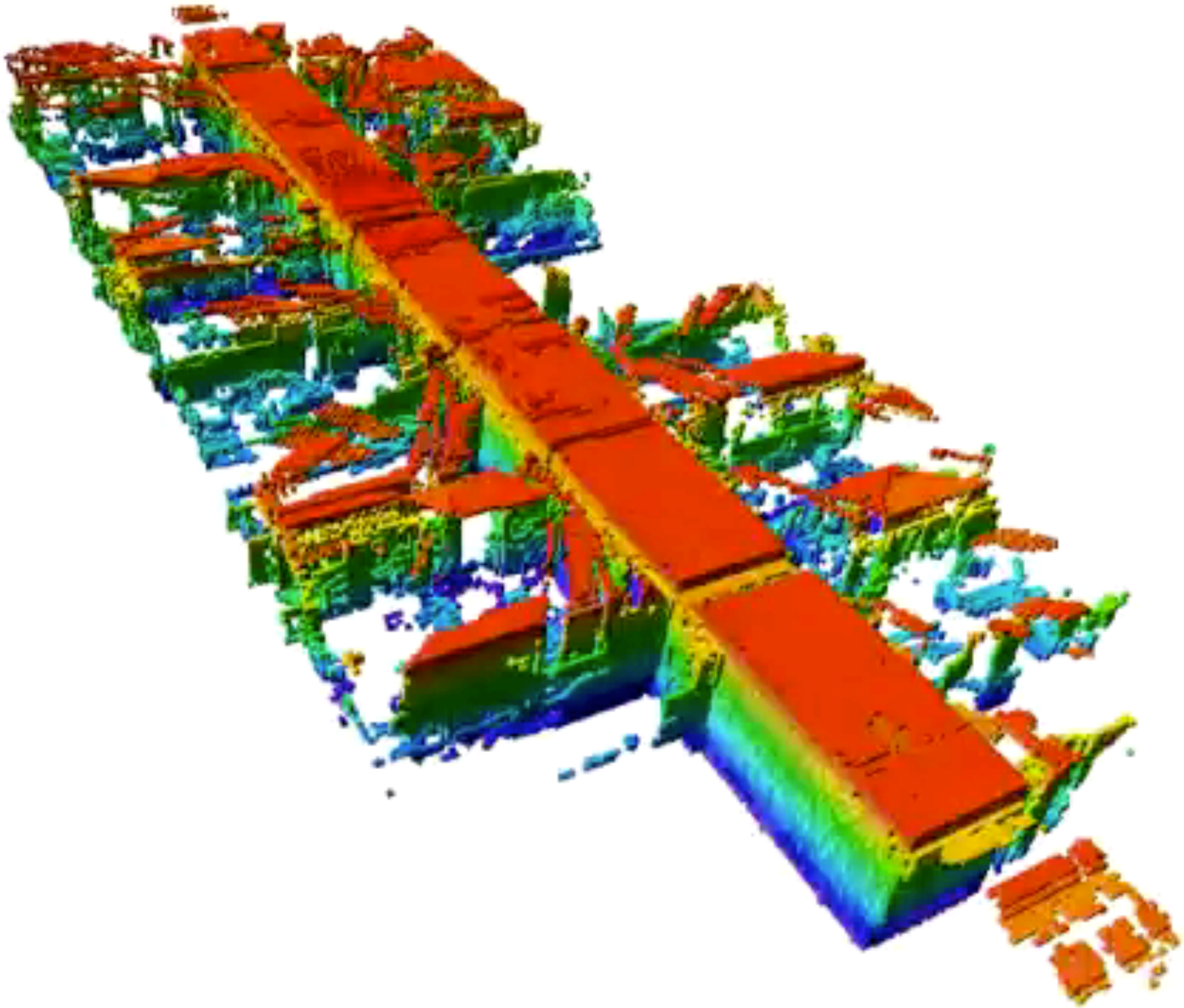


https://octomap.github.io/

Example 2: OctoMap



Example 2: OctoMap



Example 2: OctoMap

Category	Details
Information	
Query	
Update	
Memory	

Example 2: OctoMap

Category	Details
Information	Same as occupancy grids Stores information at multiple resolutions. Useful for large scale exploration, multi-res planning.
Query	
Update	
Memory	

Example 2: OctoMap

Category	Details
Information	Same as occupancy grids Stores information at multiple resolutions. Useful for large scale exploration, multi-res planning.
Query	Little expensive : O(log n), where n is the number of nodes in tree
Update	
Memory	

Example 2: OctoMap

Category	Details
Information	Same as occupancy grids Stores information at multiple resolutions. Useful for large scale exploration, multi-res planning.
Query	Little expensive : O(log n), where n is the number of nodes in tree
Update	Similar to occupancy grids, extra O(log n) complexity
Memory	

Example 2: OctoMap

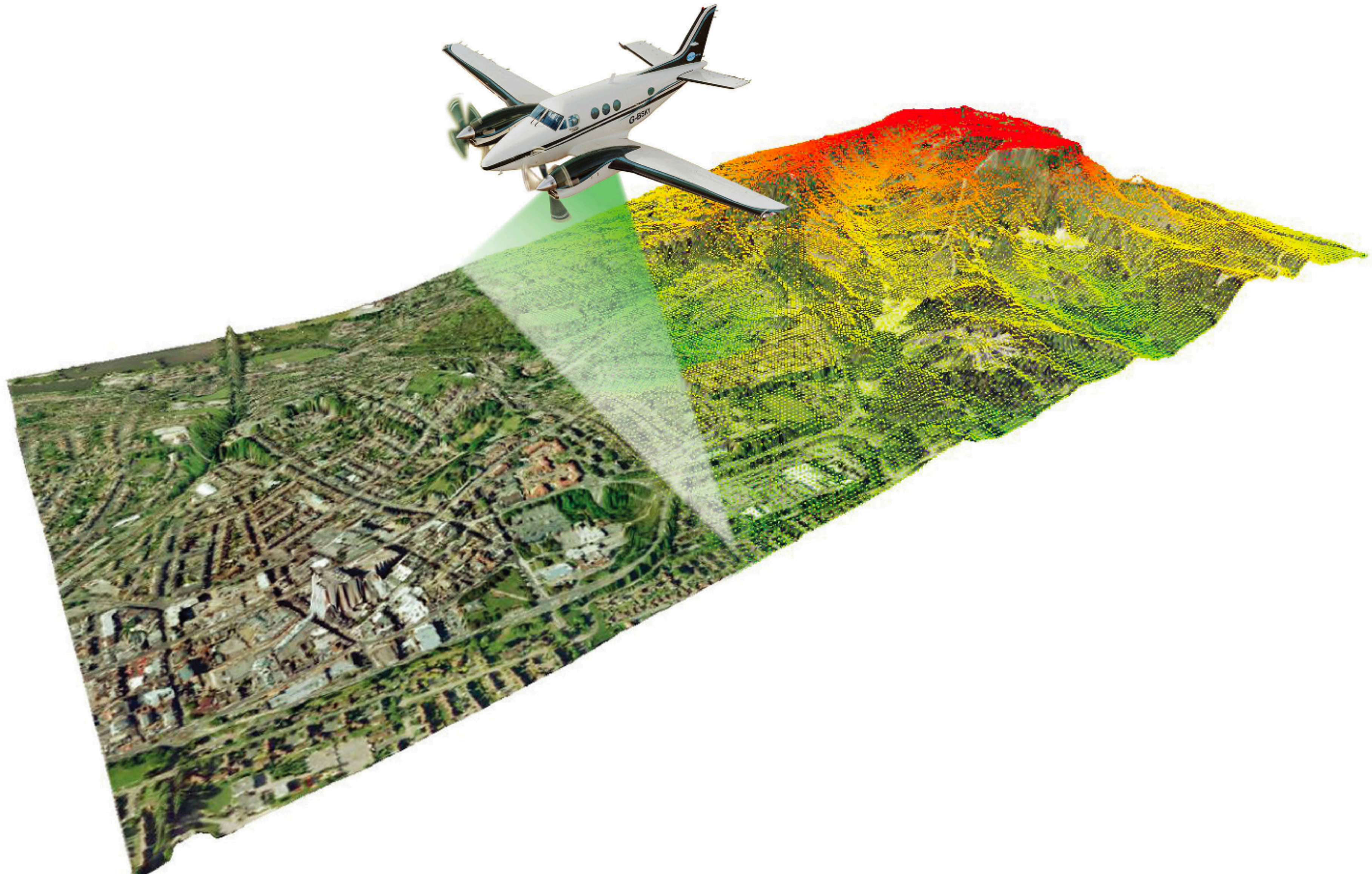
Category	Details
Information	Same as occupancy grids Stores information at multiple resolutions. Useful for large scale exploration, multi-res planning.
Query	Little expensive : O(log n), where n is the number of nodes in tree
Update	Similar to occupancy grids, extra O(log n) complexity
Memory	Much smaller than occupancy grids (proportional to amount of stuff in the world)

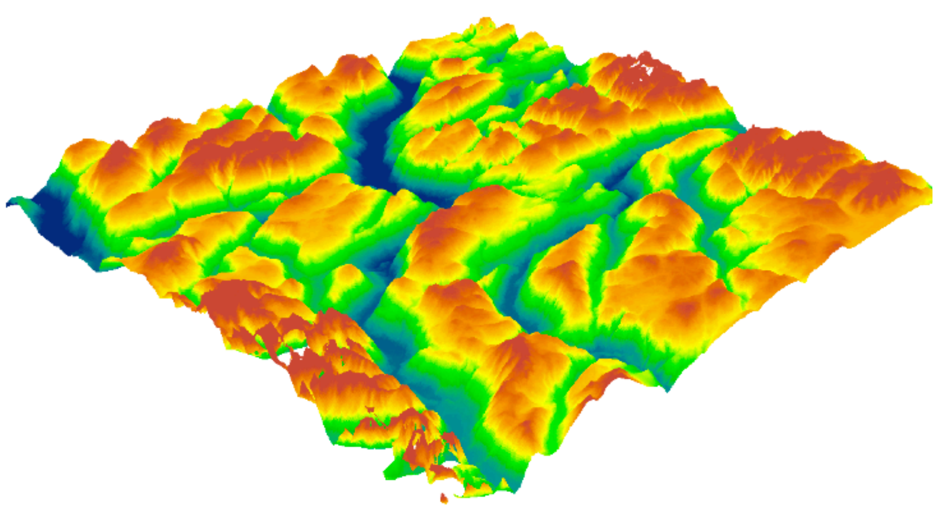
Is the world always 3D?

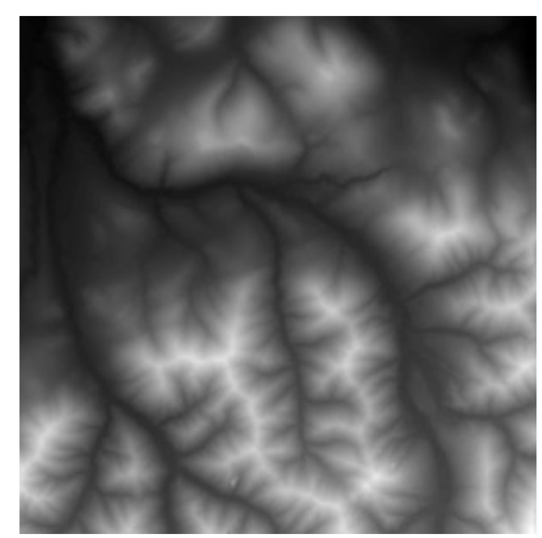
~~Is the world always 3D?~~

Do we care about 3D?

Example 3: 2.5D height map







Example 3: 2.5D height map

Category	Details
Information	
Query	
Update	
Memory	

Example 3: 2.5D height map

Category	Details
Information	Image where each pixel denotes height. Useful for mapping terrain where for overhead flight. Don't use when flying underneath objects
Query	
Update	
Memory	

Example 3: 2.5D height map

Category	Details
Information	Image where each pixel denotes height. Useful for mapping terrain where for overhead flight. Don't use when flying underneath objects
Query	O(1)
Update	
Memory	

Example 3: 2.5D height map

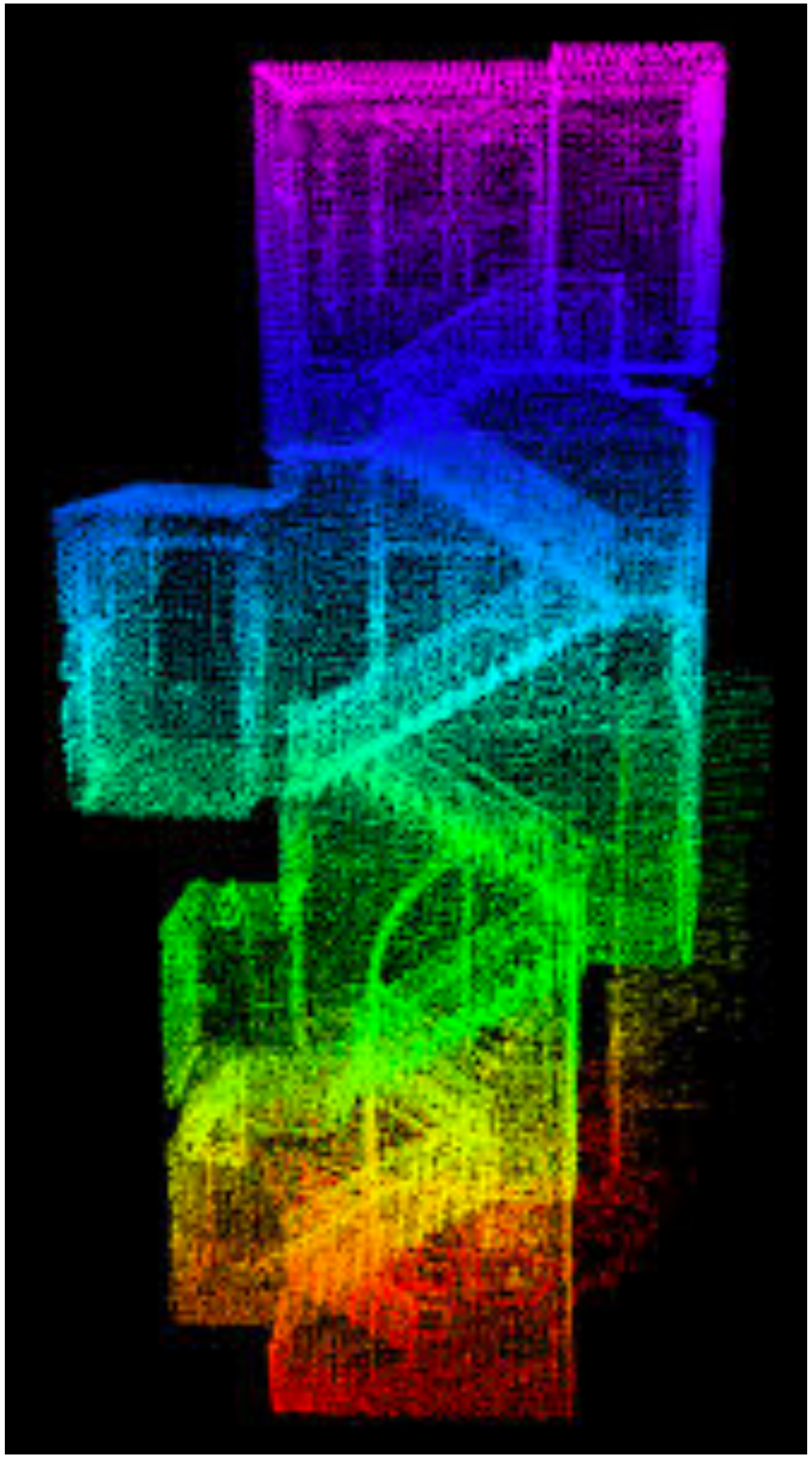
Category	Details
Information	Image where each pixel denotes height. Useful for mapping terrain where for overhead flight. Don't use when flying underneath objects
Query	O(1)
Update	Can handle noisy measurements by defining a Bayes filter for height of each cell.
Memory	

Example 3: 2.5D height map

Category	Details
Information	Image where each pixel denotes height. Useful for mapping terrain where for overhead flight. Don't use when flying underneath objects
Query	O(1)
Update	Can handle noisy measurements by defining a Bayes filter for height of each cell.
Memory	Very cheap! (2D grid)

What are my options if I don't want to discretize?

Example 4: Point cloud



courtesy Ji Zhang

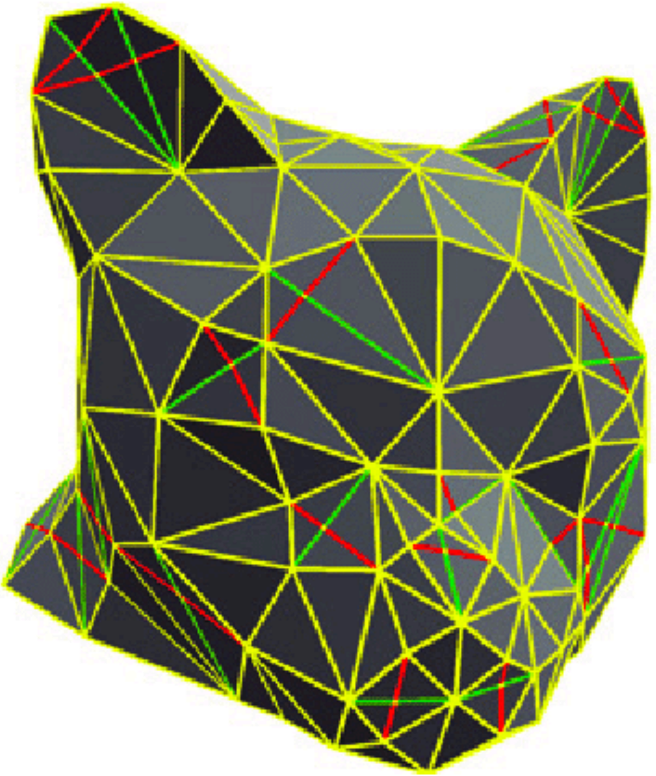
Example 4: Point cloud

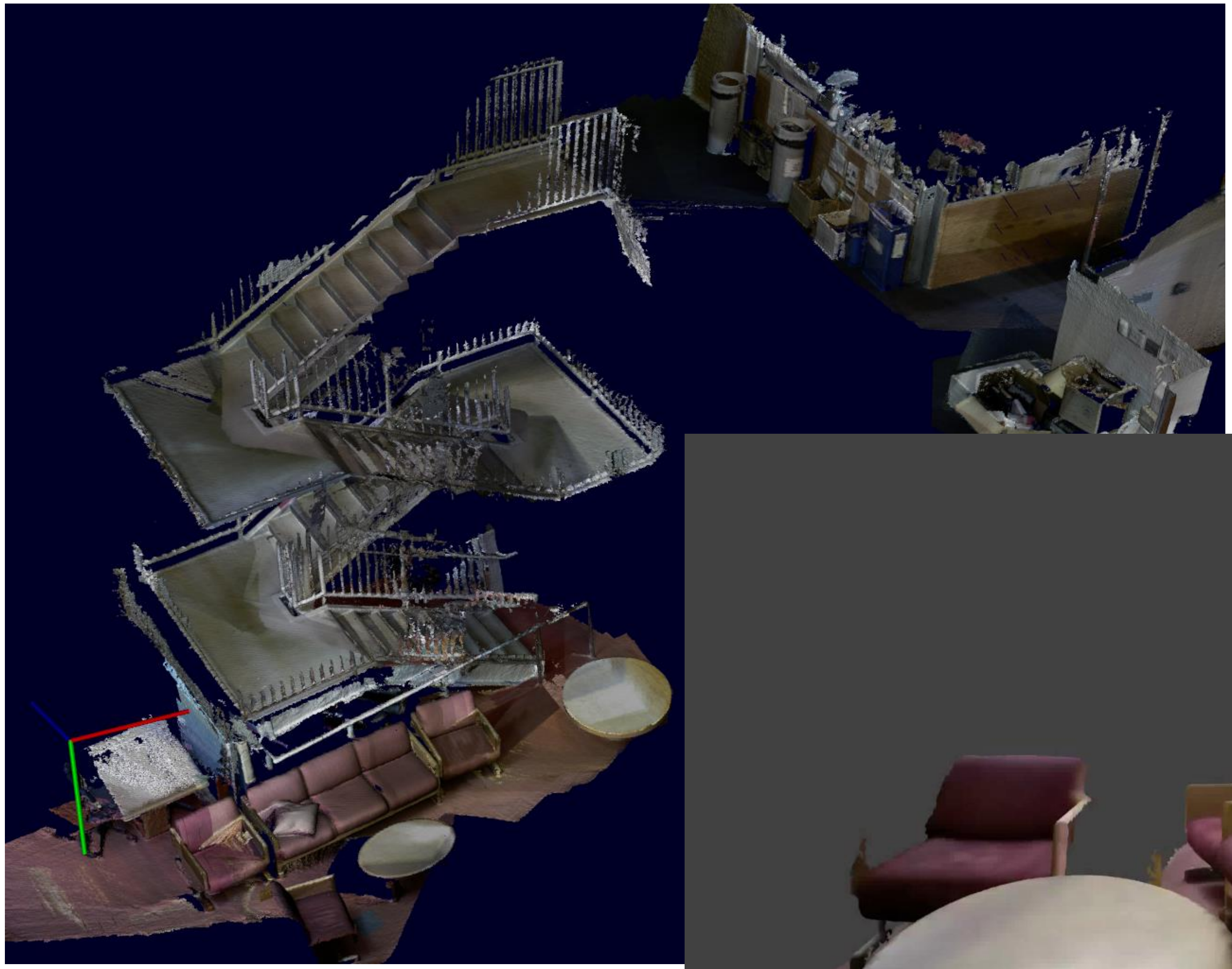
Example 4: Point cloud

Example 4: Point clouds

Category	Details
Information	Surface of obstacles (no discretization) Useful for 3D reconstruction Very accurate laser based odometry.
Query	Typical query - give me the closest point / set of points Naive query is O(N) (remember N is huge!!!)
Update	Easy to update (just dump points) Cannot deal with noisy measurements
Memory	Unbounded - can always keep adding points on top of each other indefinitely.

Example 5: Surface representations





- Handheld RGB-D sensor ($180)
- Real-time with GPU processing

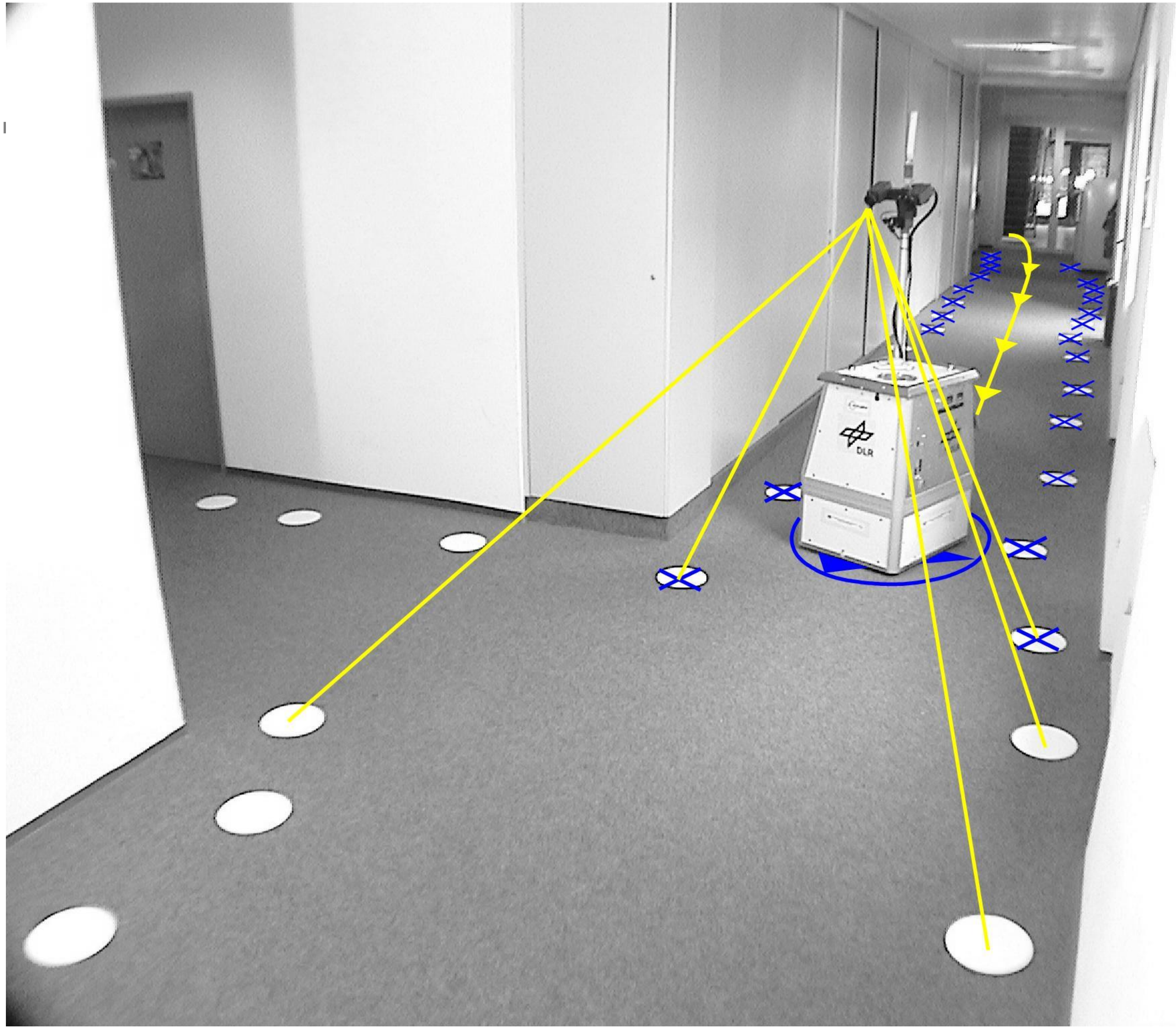
courtesy M.Kaess

Example 5: Surface representations

Category	Details
Information	List of triangles representing surface No discretization, arbitrary surfaces Used for computing object object interactions
Query	Find the closest surface. Very naively O(N) but can get massive speedups
Update	Can be updated online (albeit non-trivial) Very susceptible to noisy sensors
Memory	Proportional to amount of surface

Maps that help robots localize

Example 6: Landmark maps

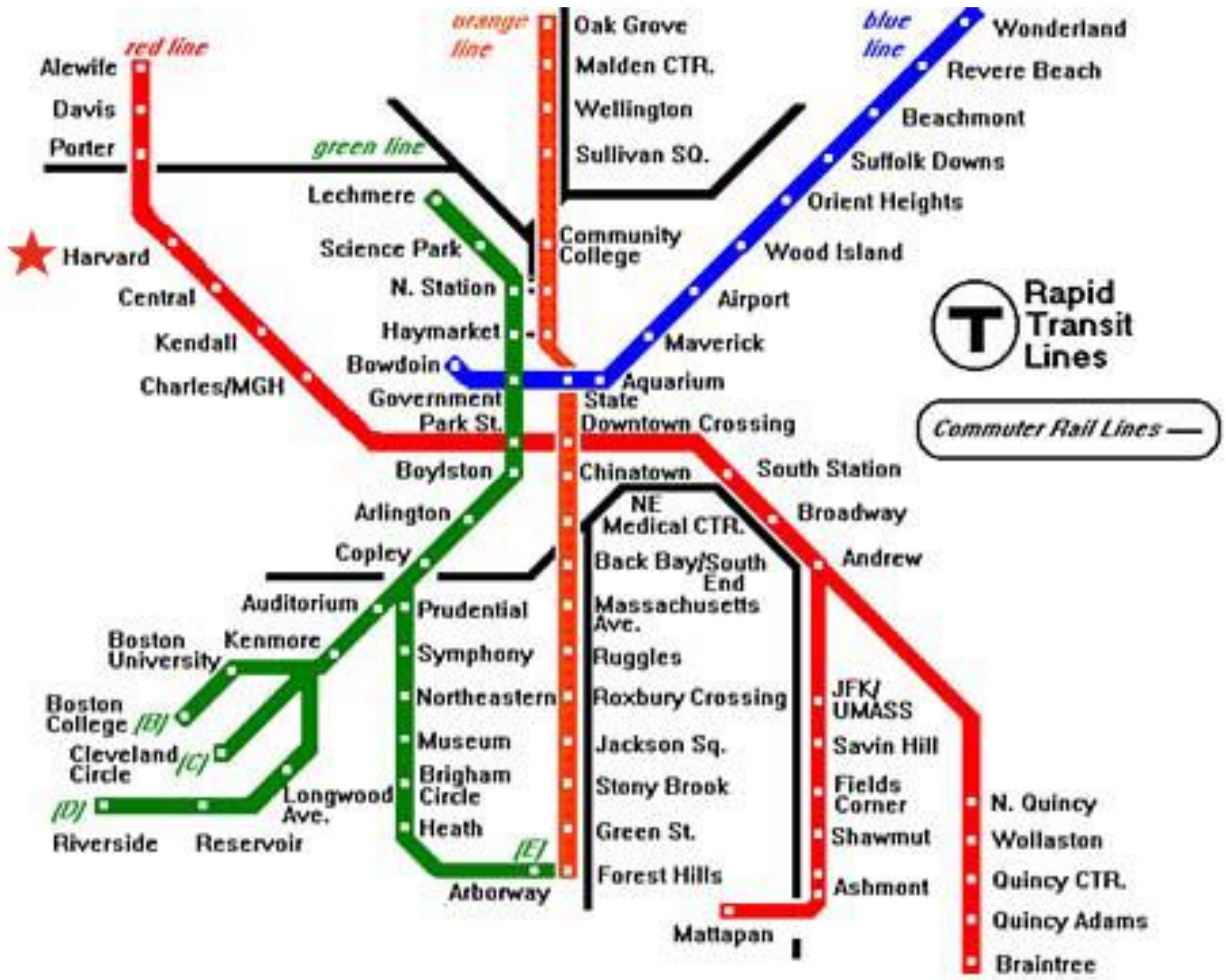


http://www.informatik.uni-bremen.de/~ufrese/slamvideos1_e.html Courtesy U. Frese

Example 6: Landmark maps

Category	Details
Information	Localization (correspondence between images at different timesteps)
Query	Typical query - give me the closest landmark Naive query is O(N)
Update	Easy to update (just dump landmarks) Need outlier rejection
Memory	Unbounded (but usually small as landmarks are sparse)

Example 7: Topological representations

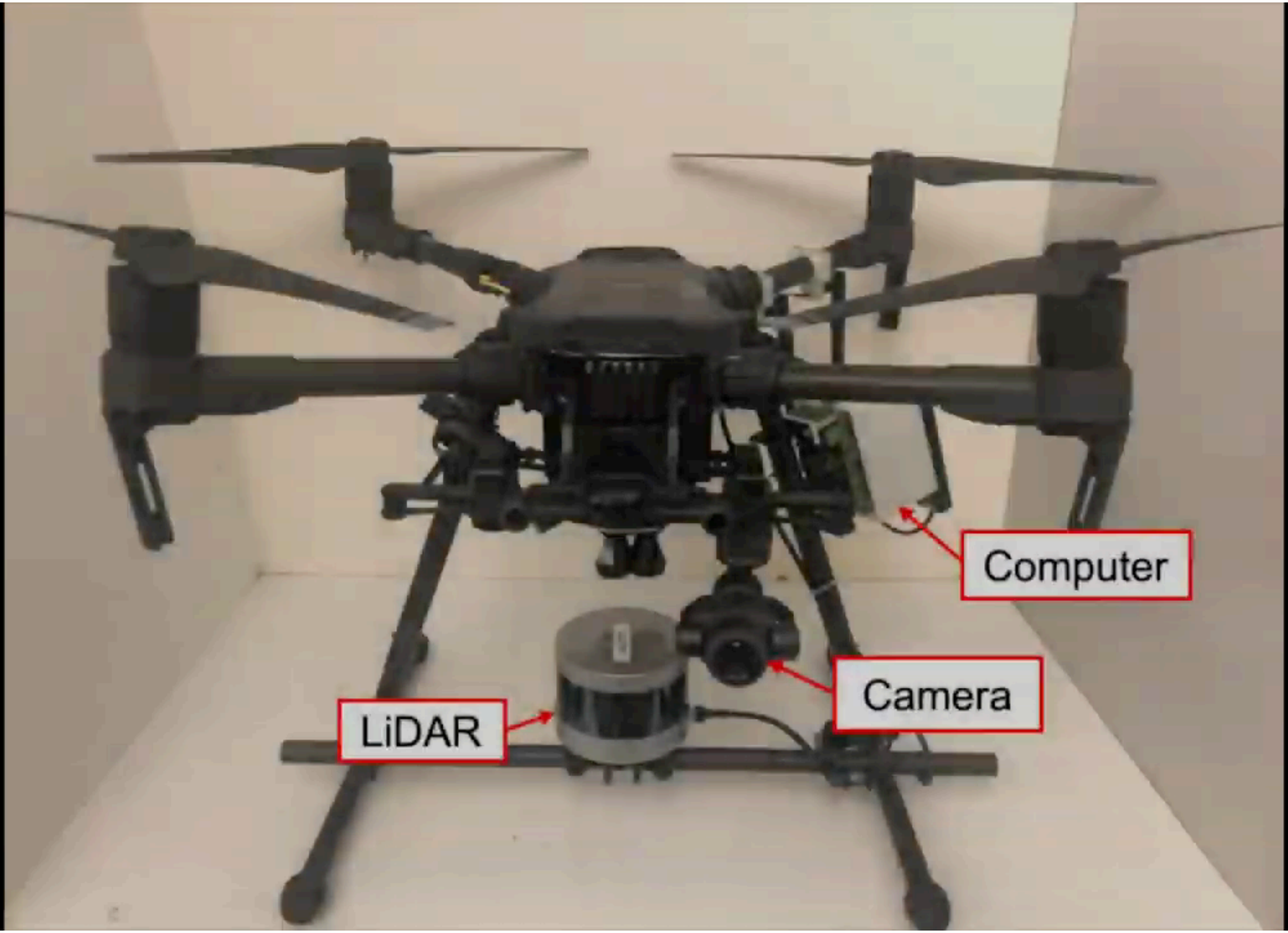


Courtesy M.Kaess

Example 7: Topological representations

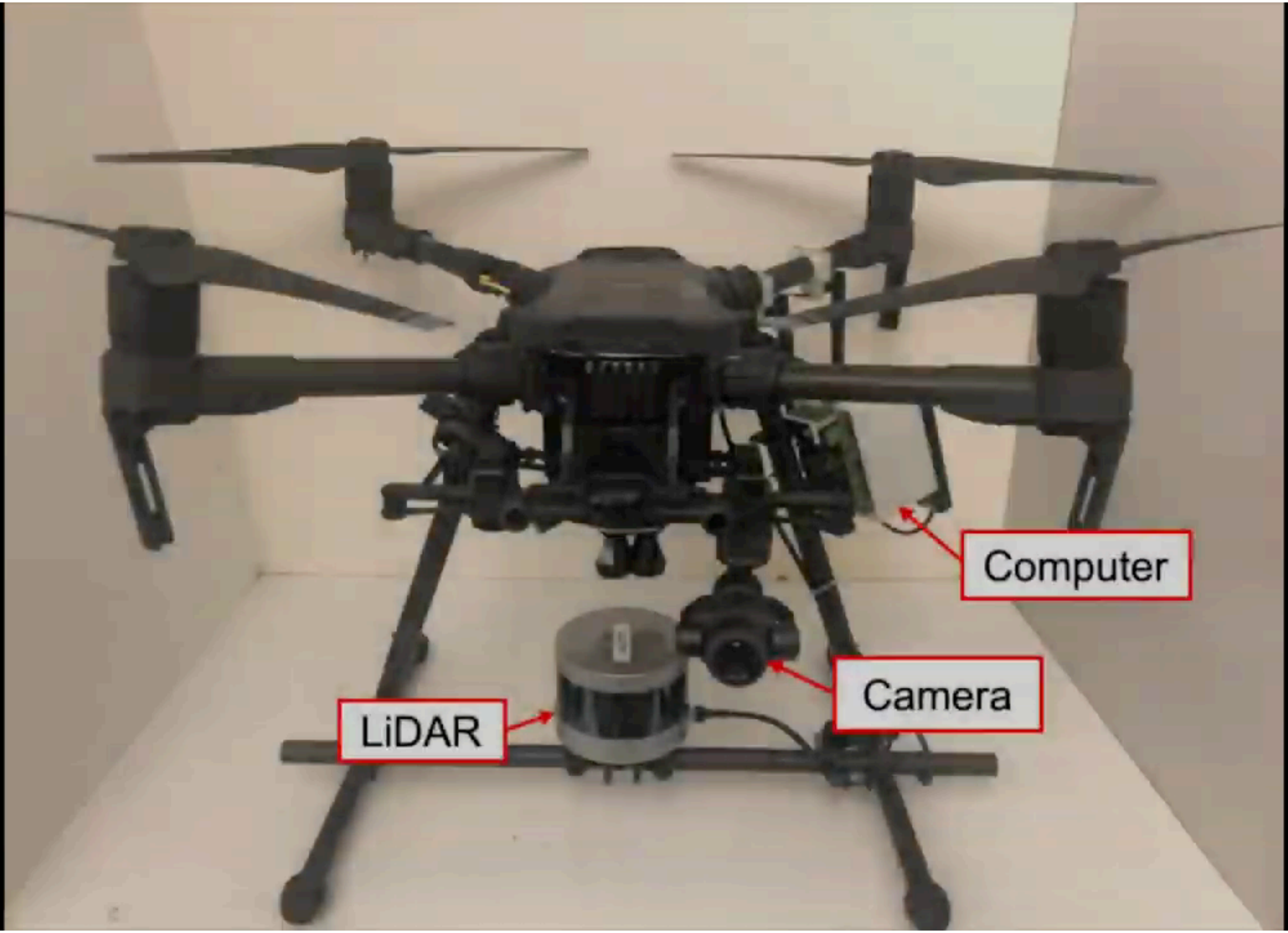
Category	Details
Information	Graph where vertices are landmarks (e.g. rooms in a building), and edges represent relationships (connections) High level navigation tasks which are specified on the topomap. Localize robot on the map by finding correspondence with vertices.
Query	Cheap graph query
Update	Non-trivial / mostly done offline
Memory	Low

Applications with multiple map representations



Bonnatti et al. 2019

Applications with multiple map representations



Bonnatti et al. 2019

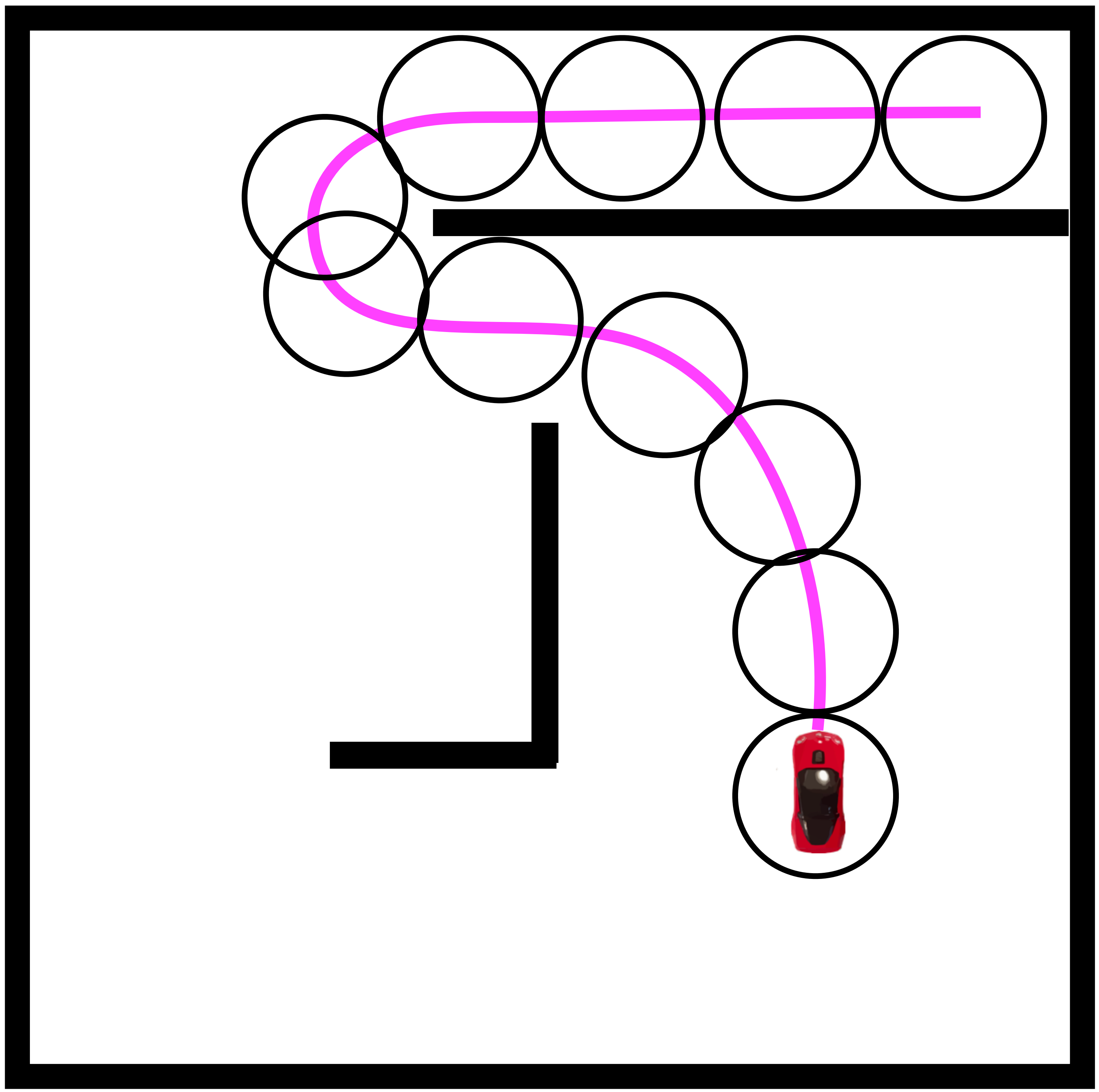
Maps are not just ways of storing sensor data

Some maps are computational operations on other maps

Distance map

Why do we need distance?

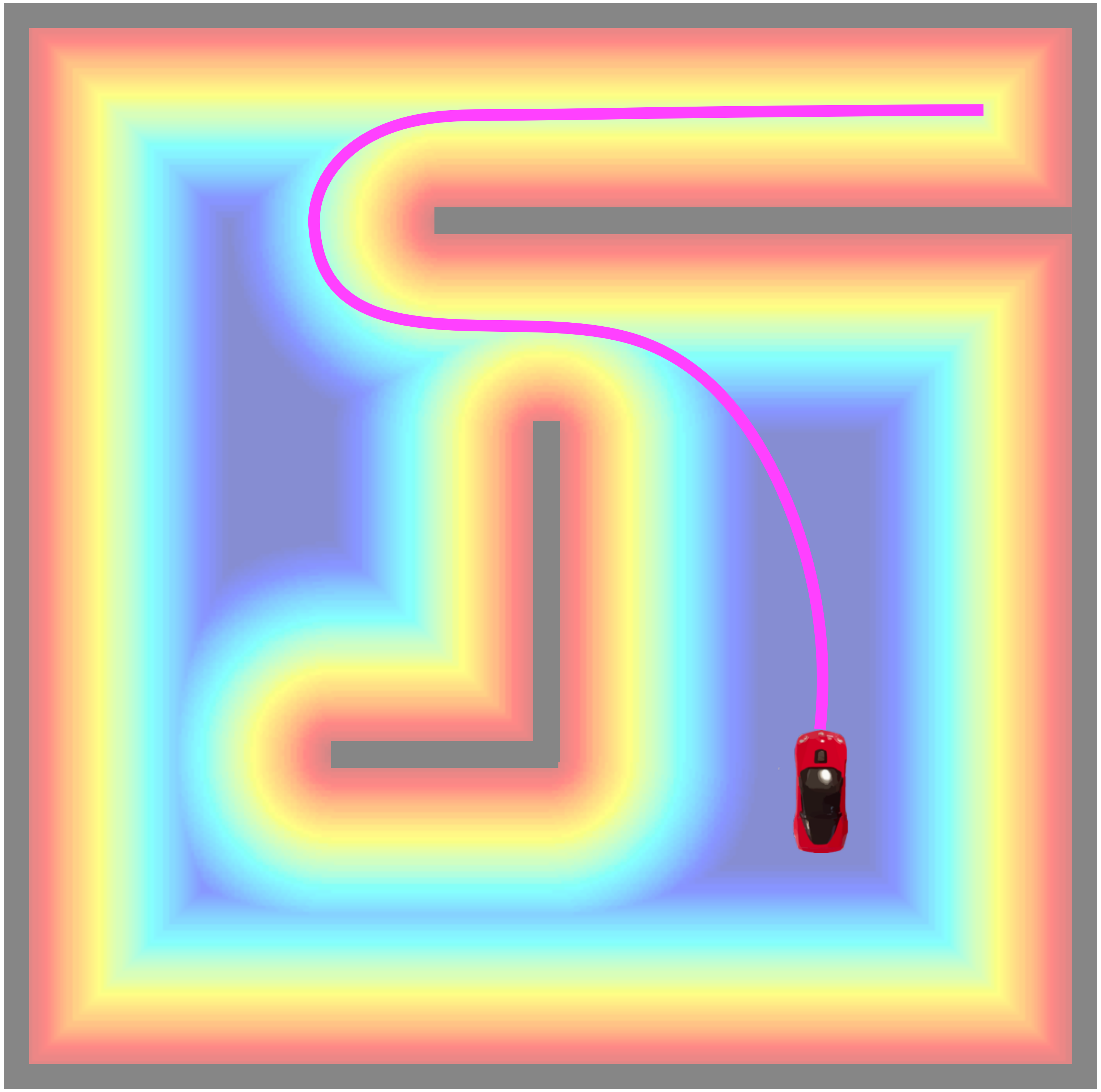
Plan a path that penalizes proximity to obstacles



Why do we need a map?

Desiderata: Map storing (truncated) distance

Input:
Binary map
of the world



Output:
Map of
same size
storing
truncated
distance

Example 8: Distance map

Category	Details
Information	Truncated distance to obstacles
Query	O(1)
Update	We want to incrementally update this map Ideally O(k) where k is the number of cells which changed distance value
Memory	Same as the underlying occupancy grid

How do we efficiently calculate distance map?

Dynamic programming to the rescue!

Initialize distance d(i) for free cells to Inf

Insert all boundary pixels to queue Q

While Q not empty

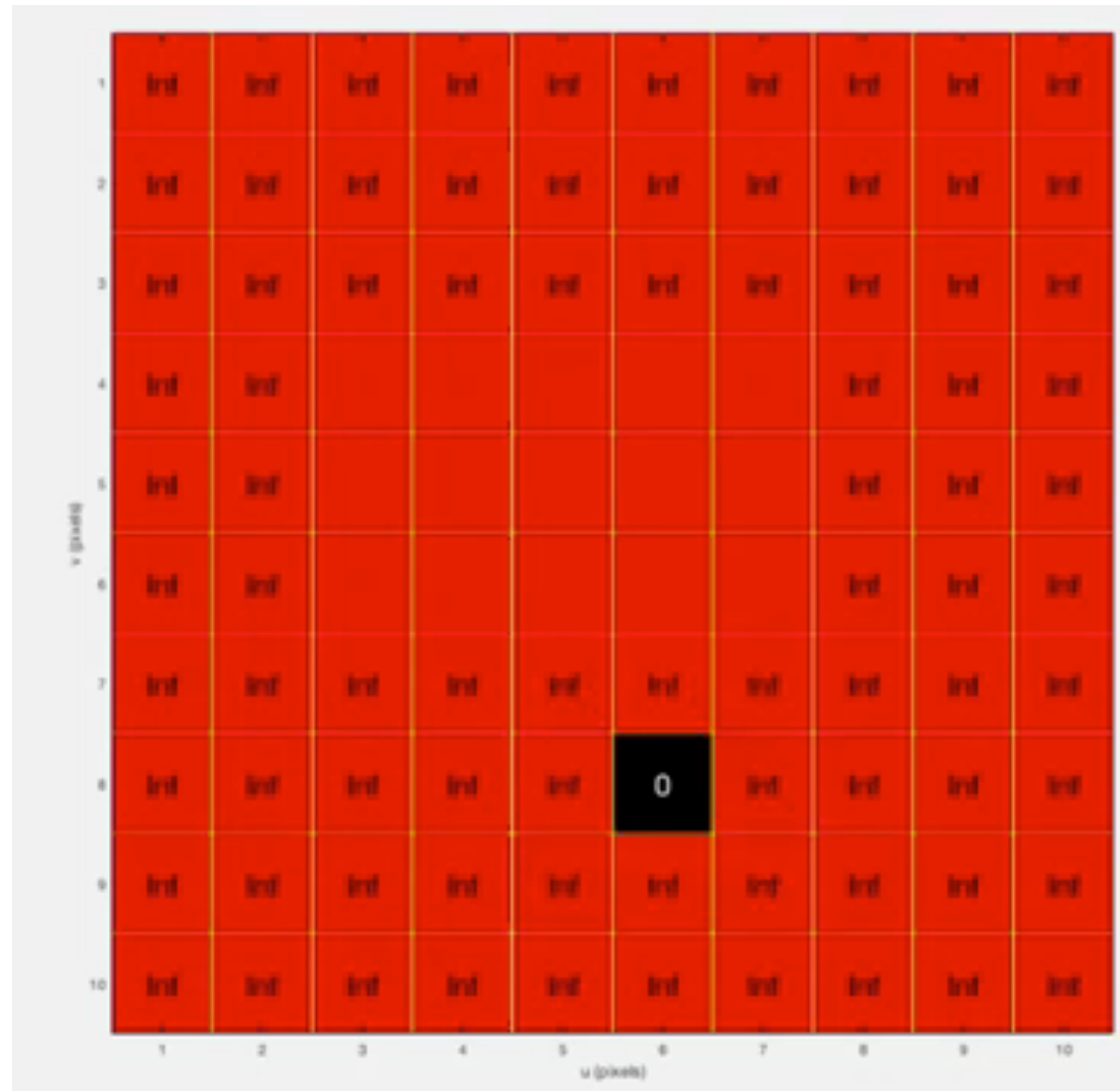
    $x = Q.pop()$

    for each n **in** Neigbour(x)

        $d(n) = \min(d(n), v(x) + dis(x,n))$

        if $d(n) <= dmax$

            Q.insert(n)



Dynamic programming to the rescue!

Initialize distance d(i) for free cells to Inf

Insert all boundary pixels to queue Q

While Q not empty

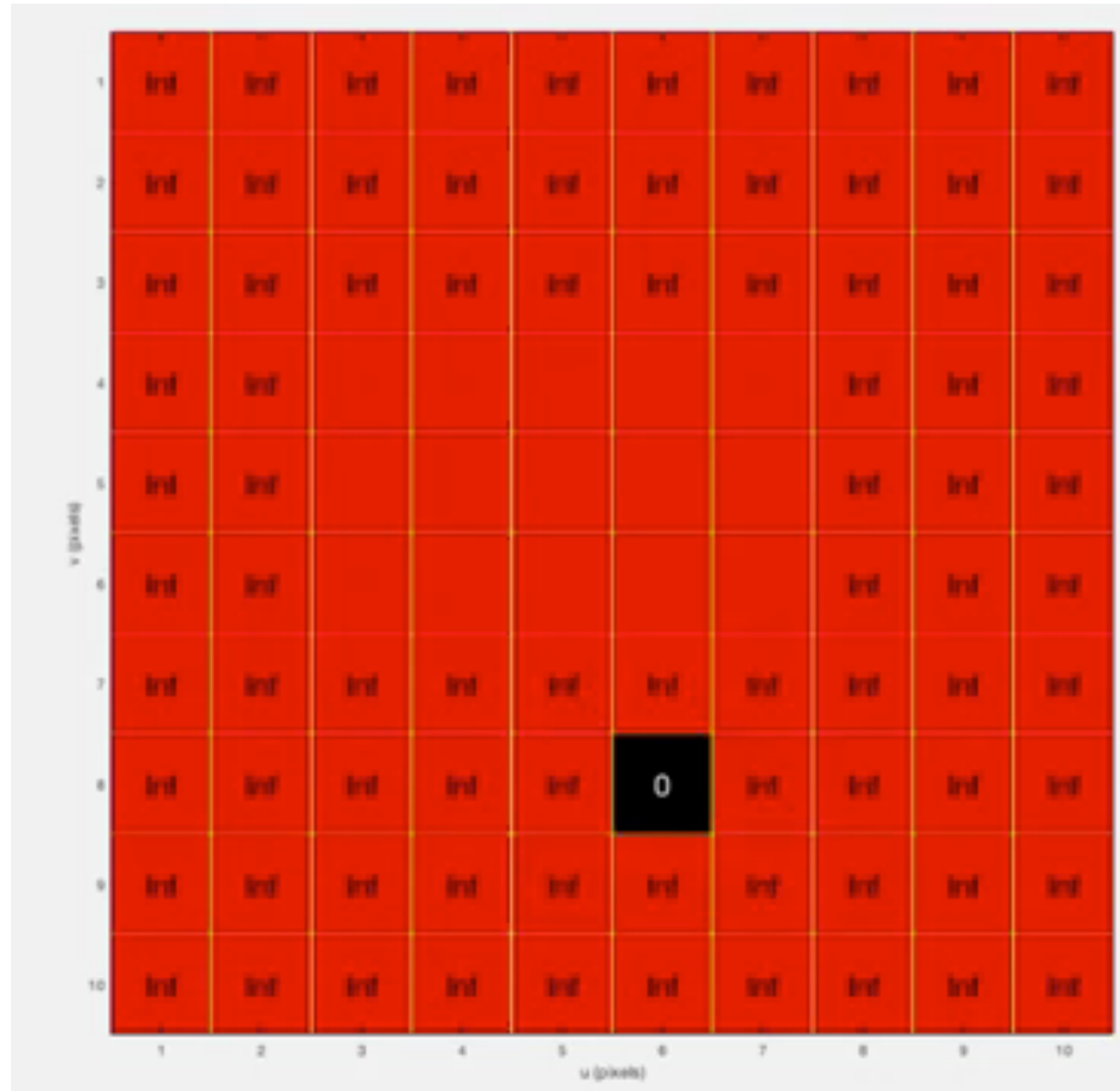
    x = Q.pop()

    for each n **in** Neigbour(x)

        d(n) = min(d(n), v(x) + dis(x,n))

        if d(n) <= dmax

            Q.insert(n)



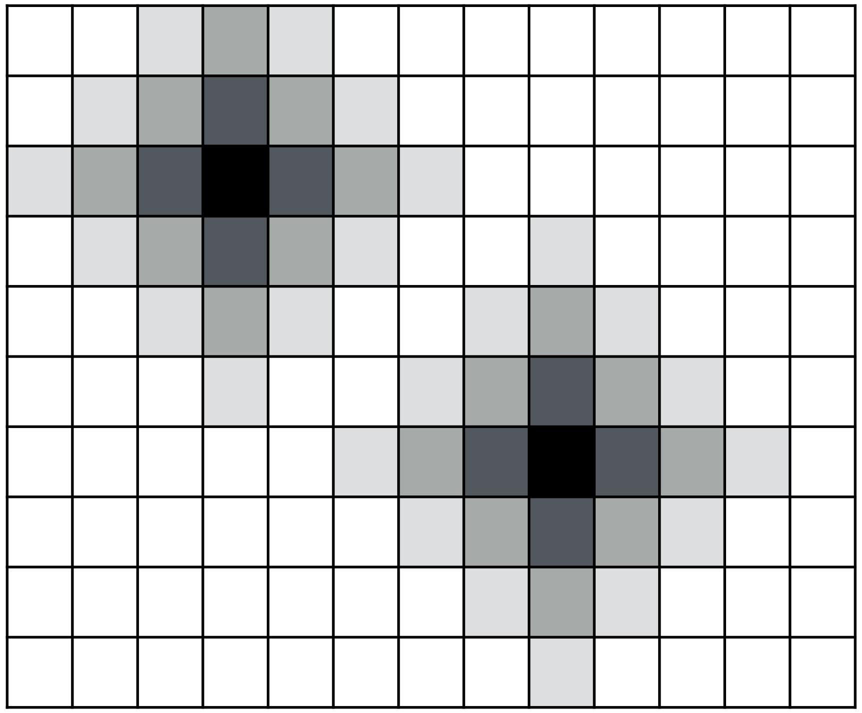
How can we incrementally update this map?

Tale of two wavefronts

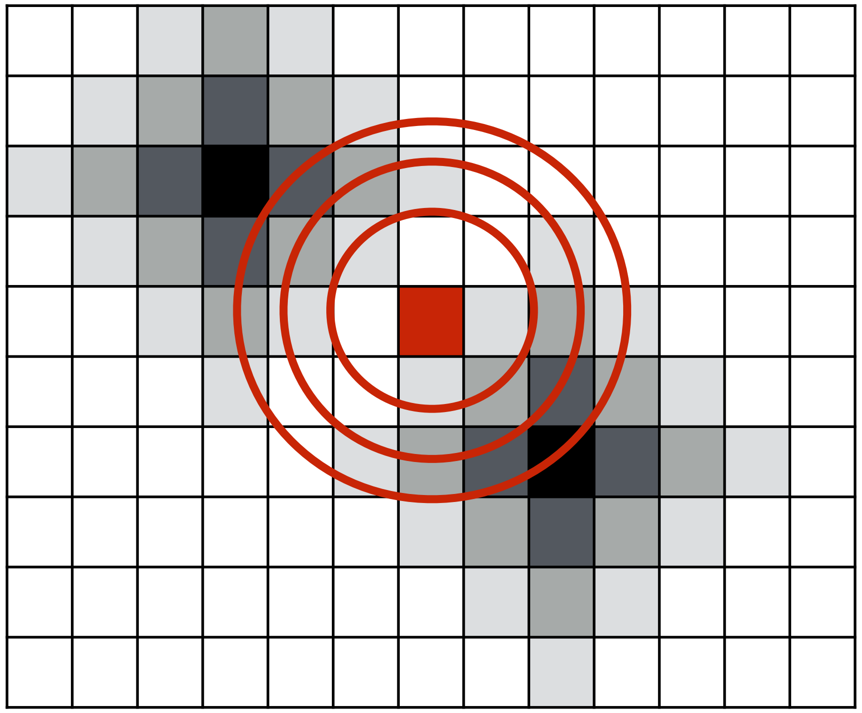
LOWER (when you add obstacle)

RAISE (when you delete obstacle)

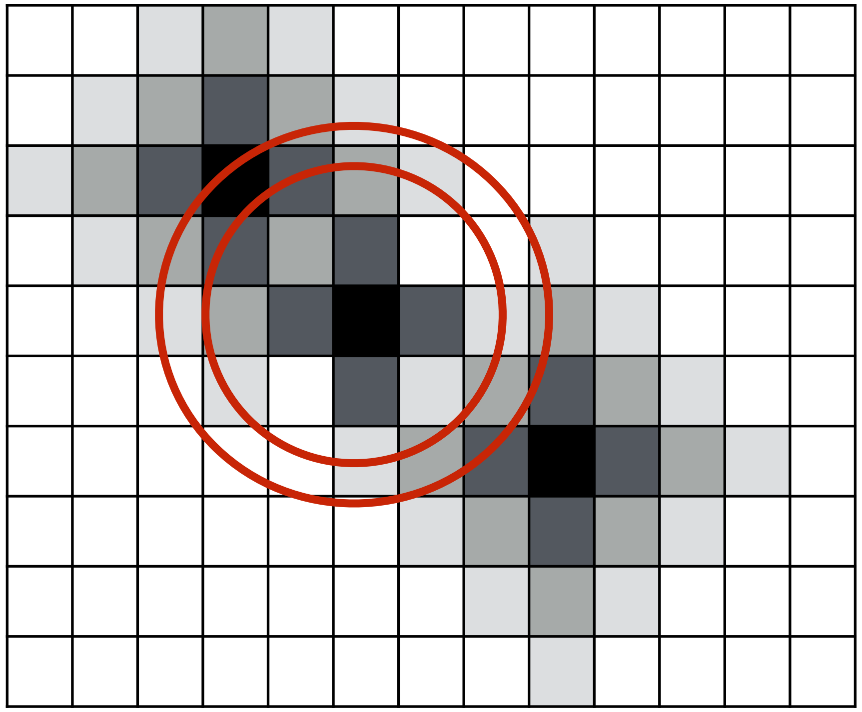
When obstacle is added



Existing distance map

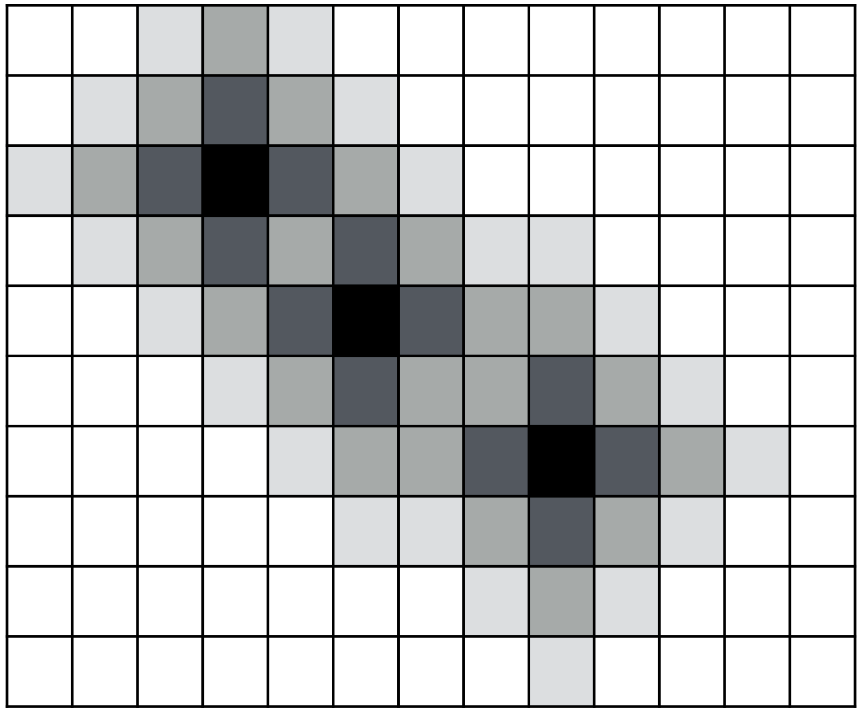


New obstacle added. LOWER wavefront started



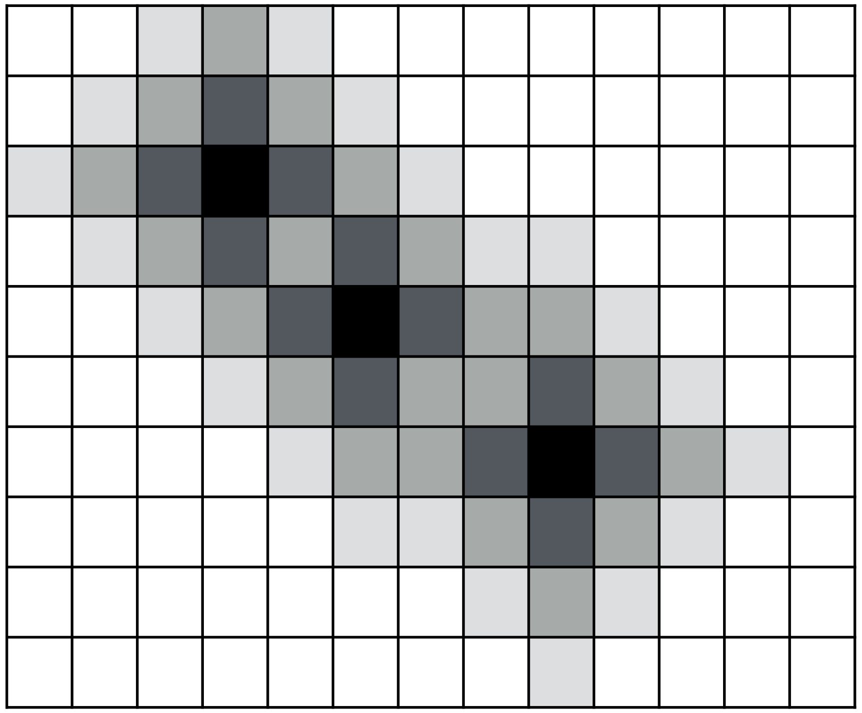
Overwrite distances if smaller value.

Remember closest obstacle

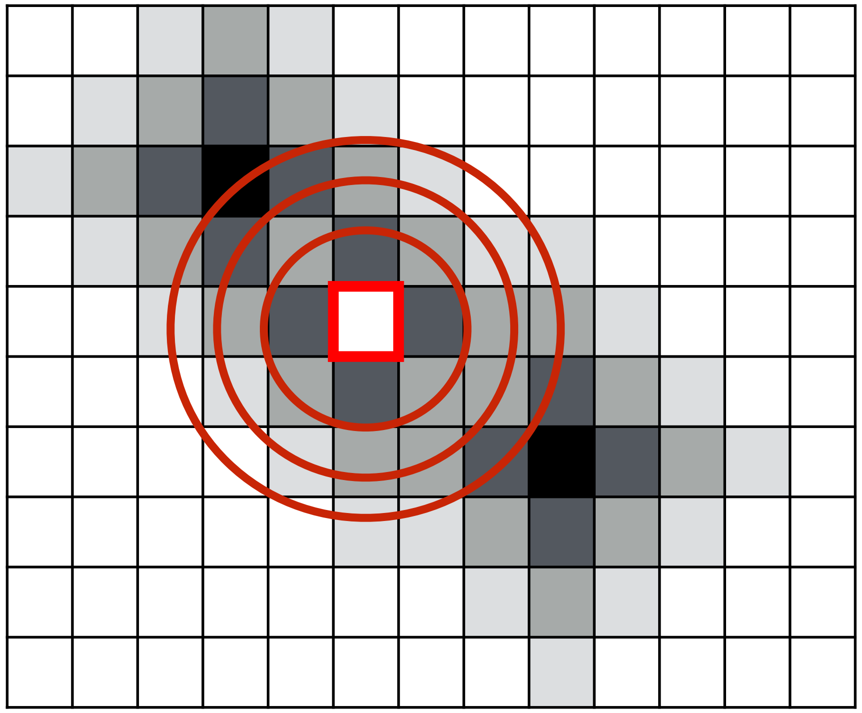


Stop wavefront whenever you meet higher distance

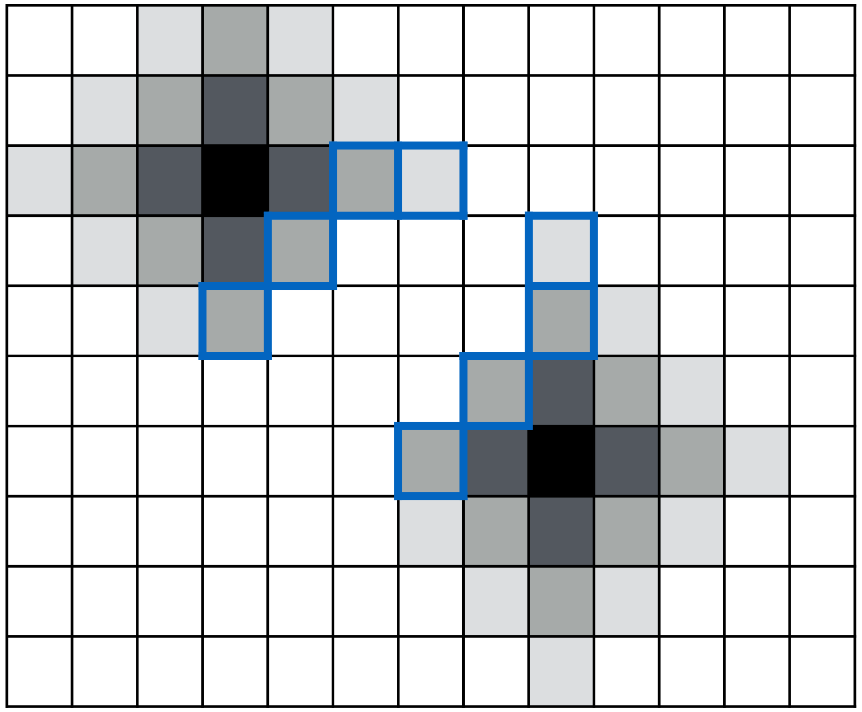
When obstacle is deleted



Existing distance map

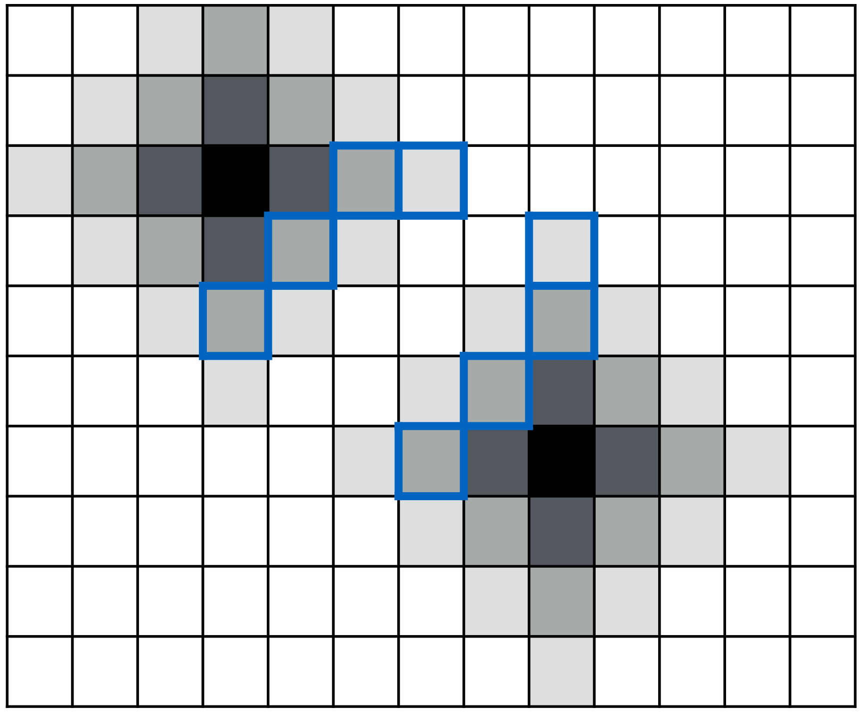


Obstacle deleted RAISE wavefront started



Set d=dmax if closest obstacle was deleted.

Stop wavefront otherwise.



Boundary cells trigger LOWER wavefront

Template for incremental dynamic programming

Input: Cells which changed status (obstacles added / removed)

Insert all changed cells into a queue Q

While Q not empty

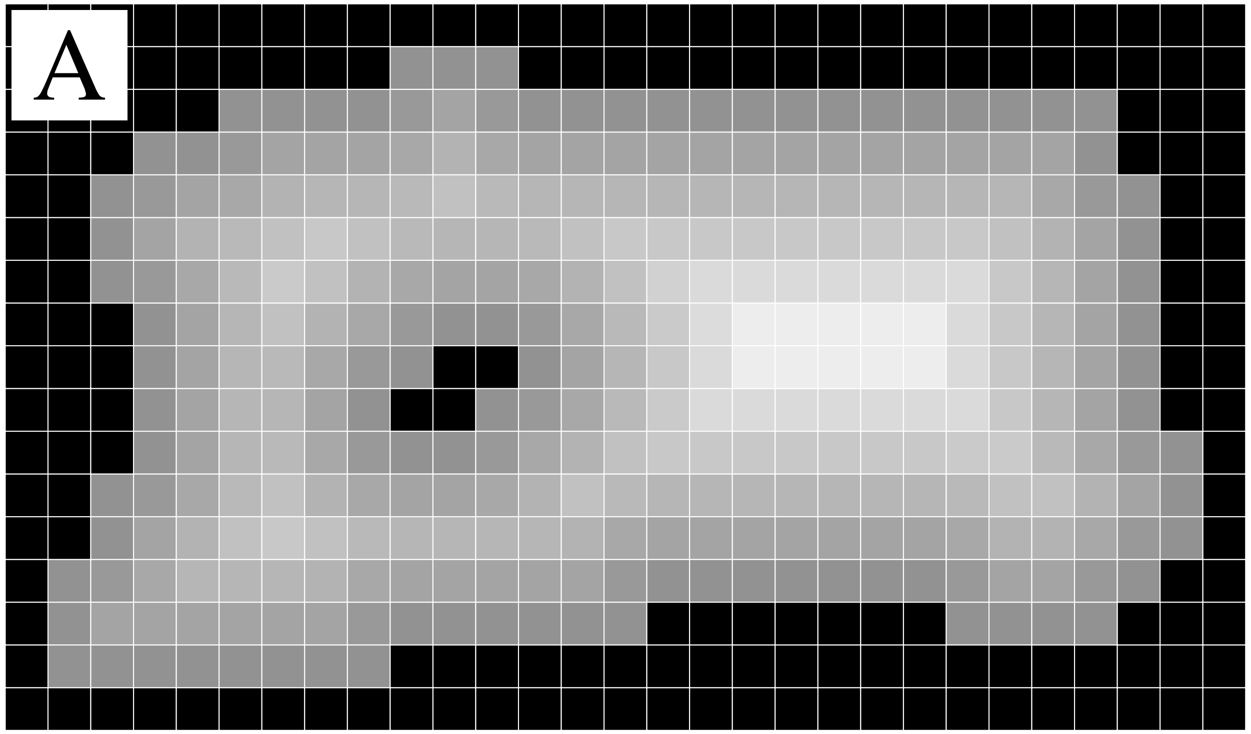
Node n = Q.pop()

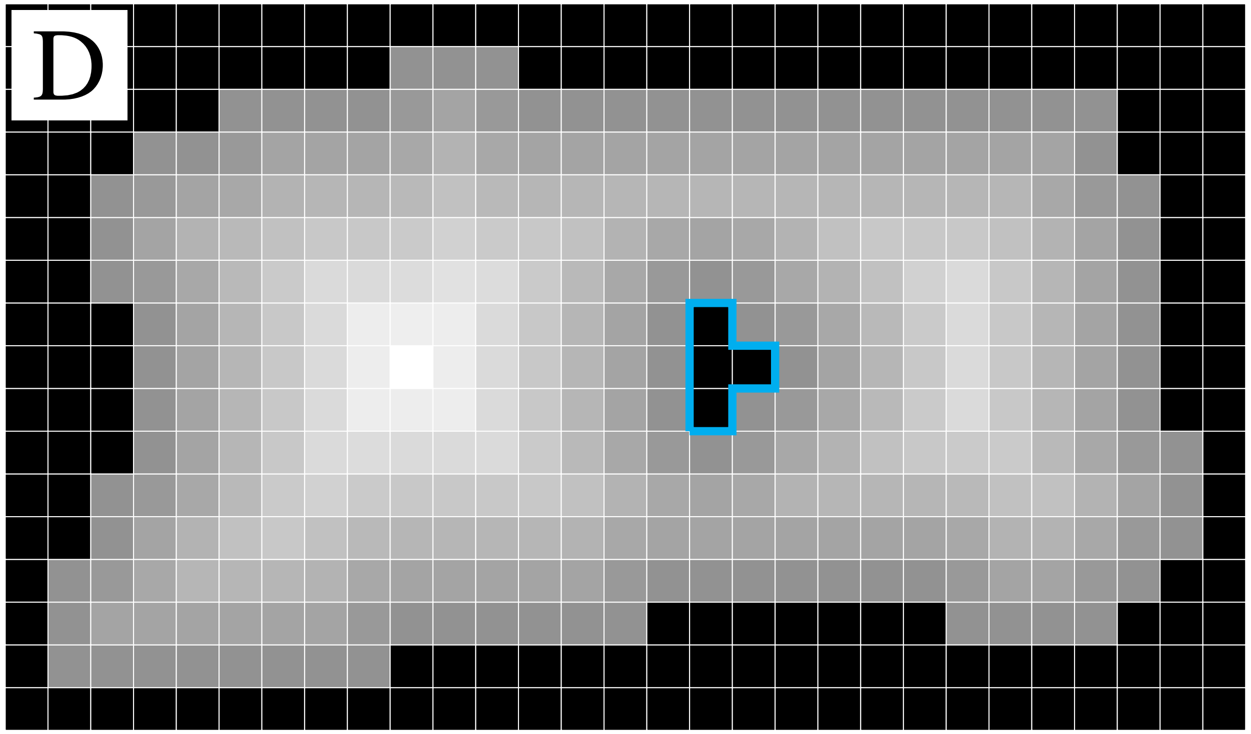
If n is over consistent (d_old > d_new), lower value

If n is under consistent (d_old < d_new), raise value

Add neighbors whose values need to be changed.

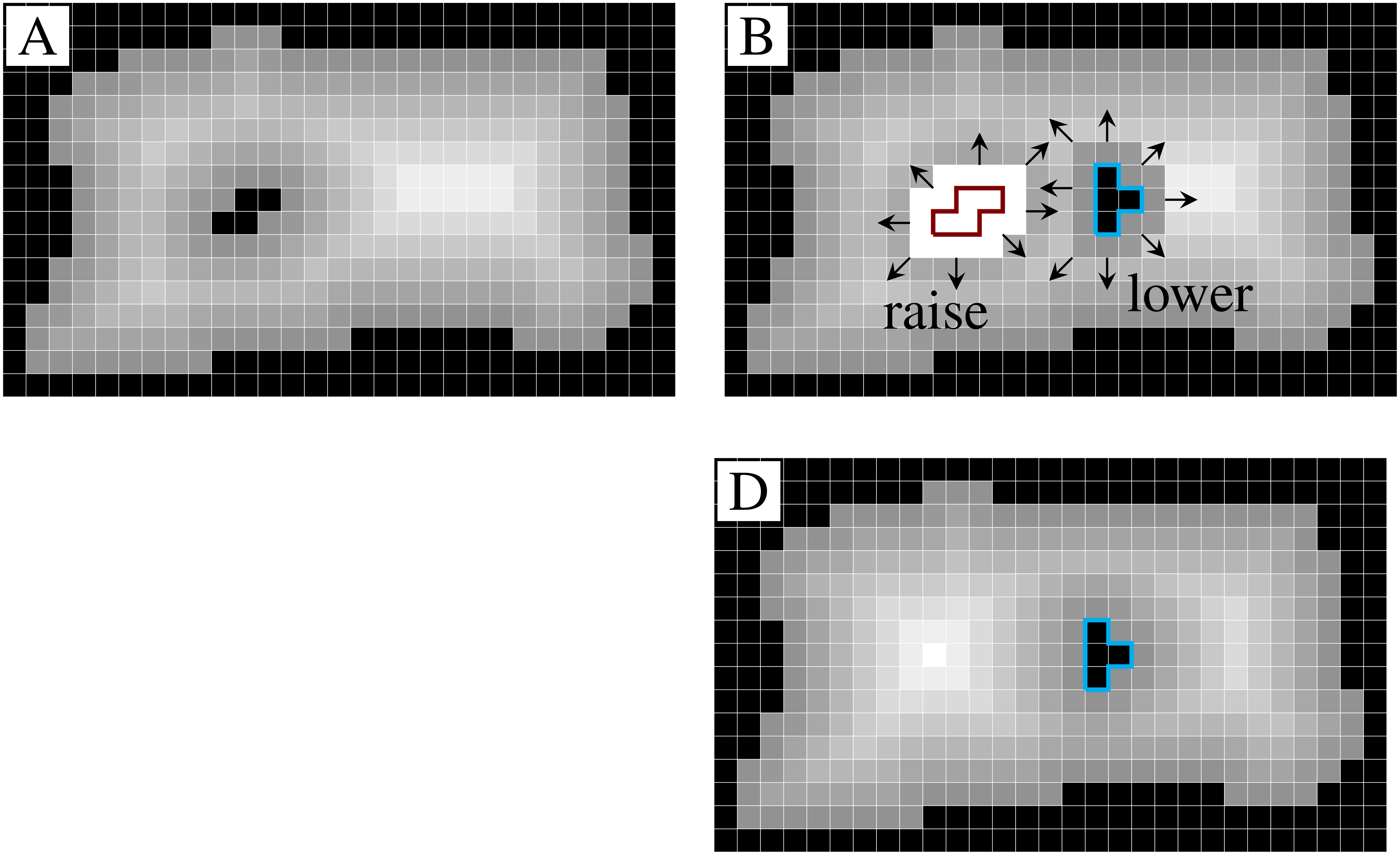
Incremental Euclidean Distance Mapping





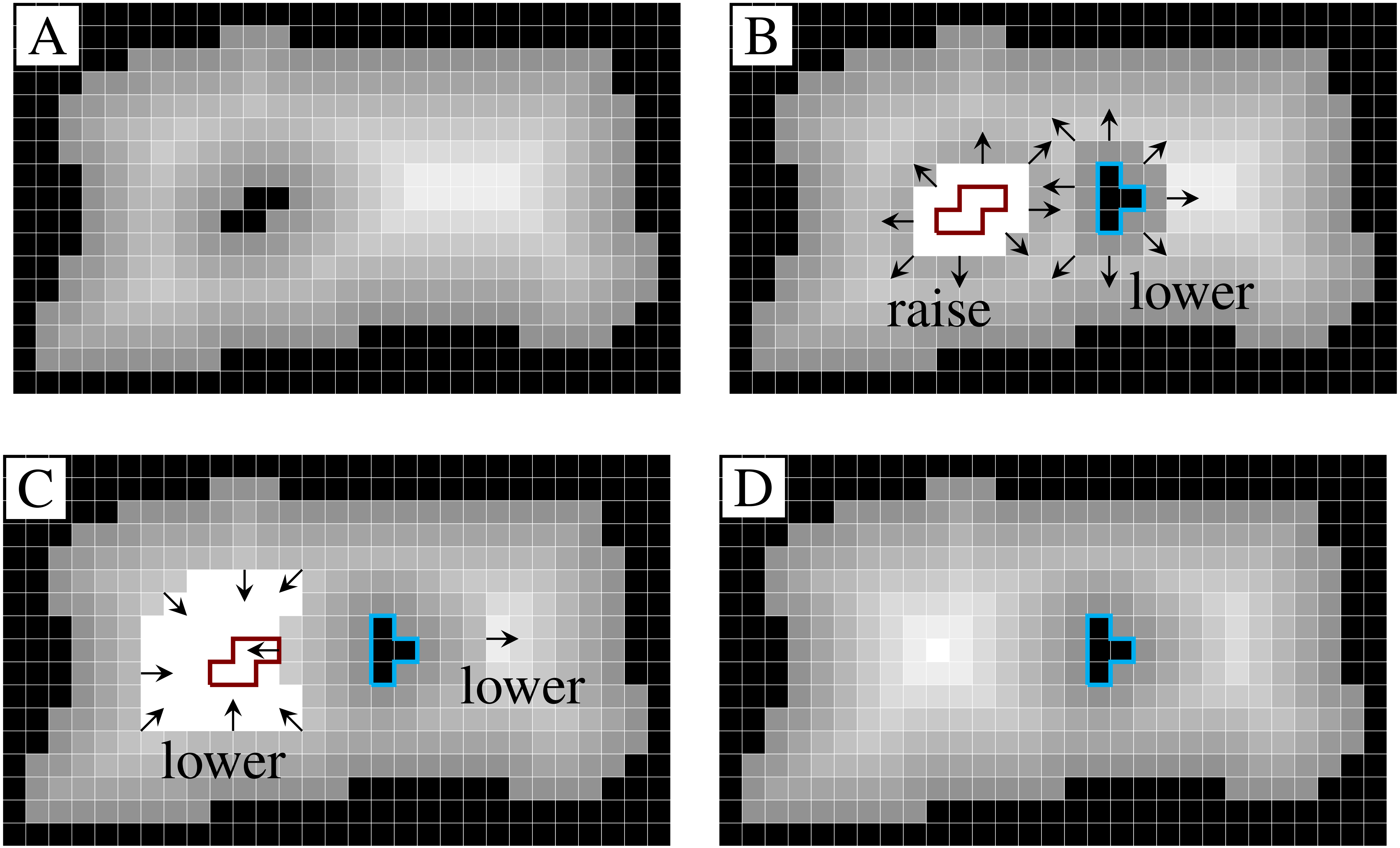
"Improved updating of Euclidean distance maps and Voronoi diagrams", Lau et al. 2010

Incremental Euclidean Distance Mapping



“Improved updating of Euclidean distance maps and Voronoi diagrams”, Lau et al. 2010

Incremental Euclidean Distance Mapping



“Improved updating of Euclidean distance maps and Voronoi diagrams”, Lau et al. 2010

Truncated signed distance map

We can easily modify this algorithm to tell us distance inside an object.

Signed distance - negative inside object, positive outside

Signed distance important in motion planning!