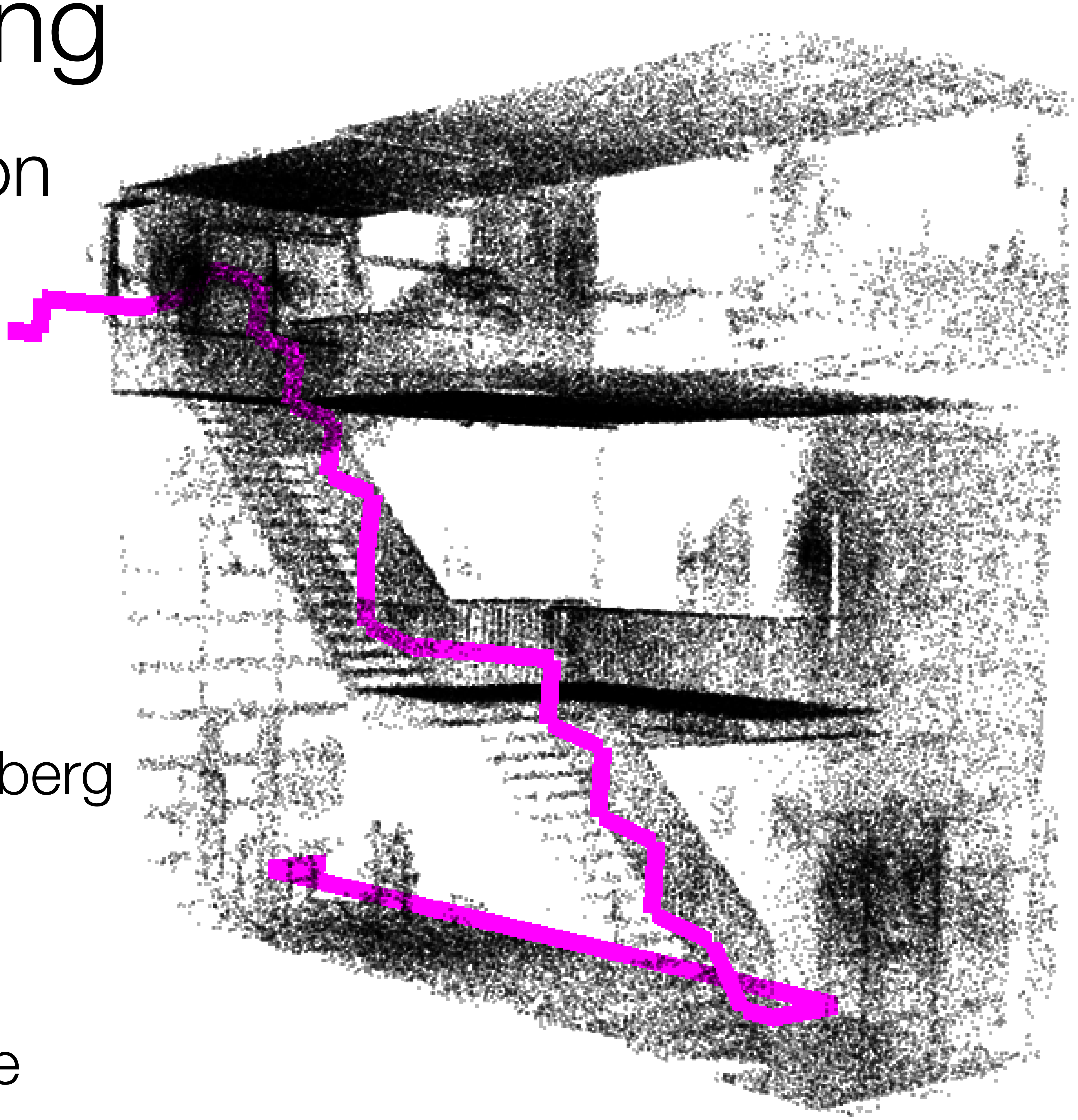
From indoor point cloud to path finding

P5 presentation



By: Olivier Rodenberg

Supervisors:
Dr. Sisi Zlatanova
ir. Edward Verbree

Contents

1. Introduction
2. Research objectives
3. Methodology
4. Results
5. Conclusions & Future work

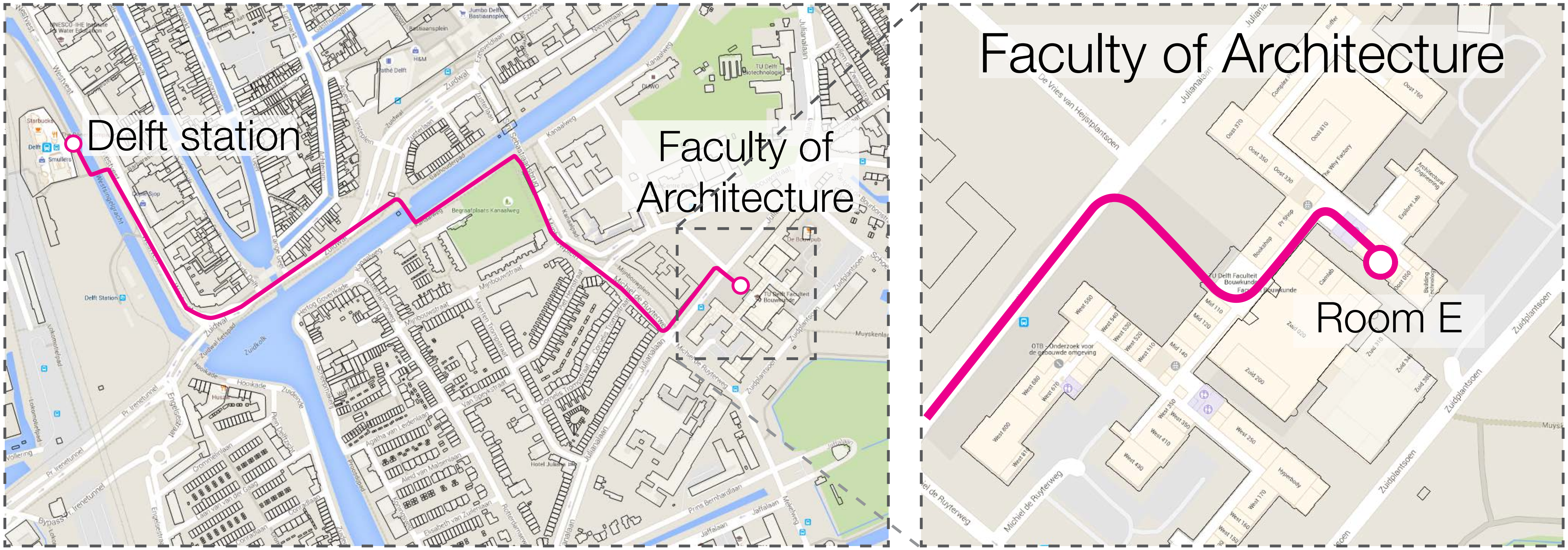
1. Introduction

1.1. Use case
1.2. Point cloud
1.3. Octree
1.4. Path finding

1.1. Use case

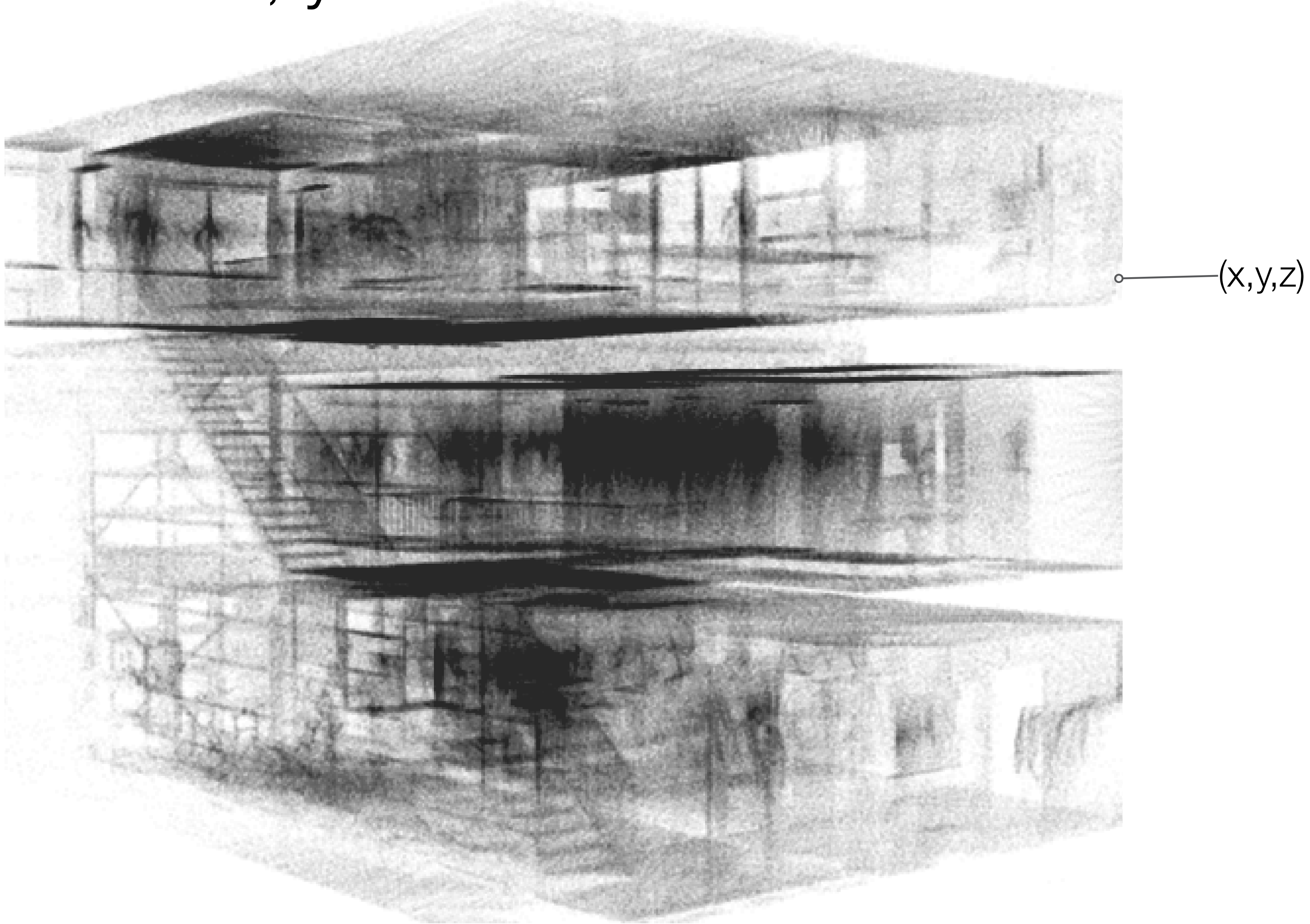
Indoor meets outdoor pathfinding

Path finding from Delft station to room E in the Faculty of Architecture

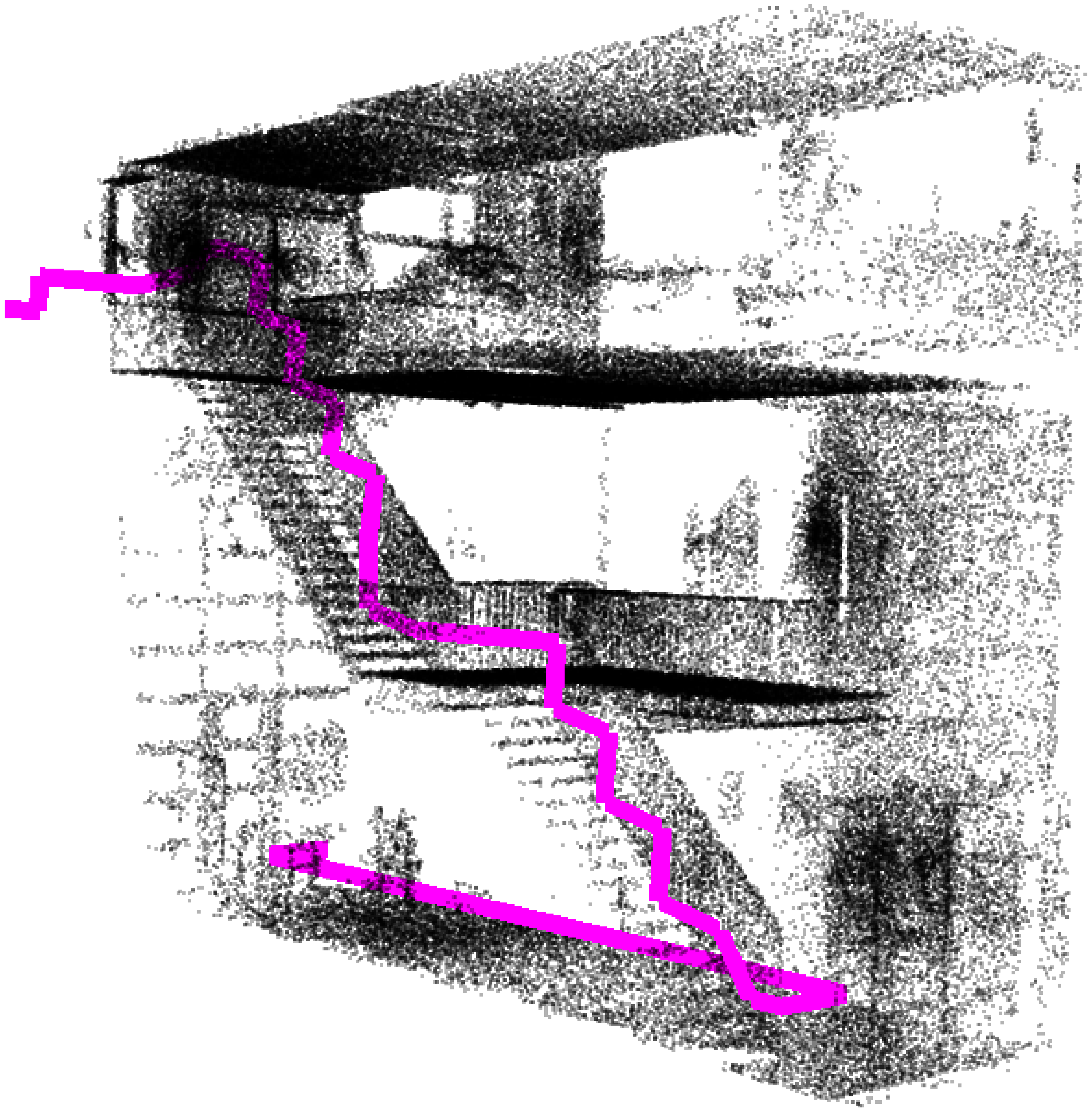


1.2. Point cloud

A large collection of points with at least an x, y and z coordinate.



1.2. Point cloud
for path finding



Advantages:

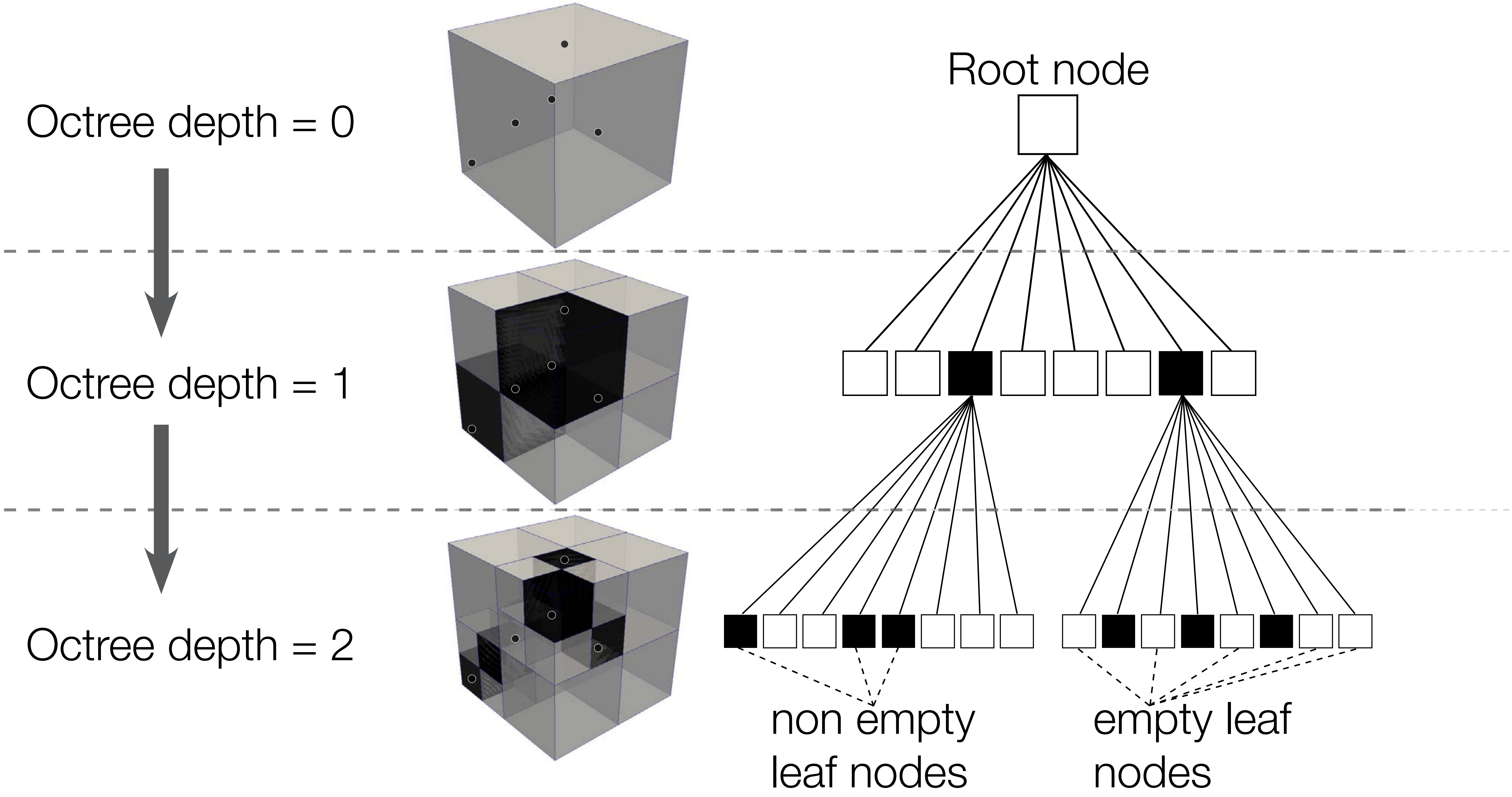
- Fast acquisition of 3D model

Difficulties:

- Only information about the boundaries
- What is the empty space?
- A point cloud is unstructured

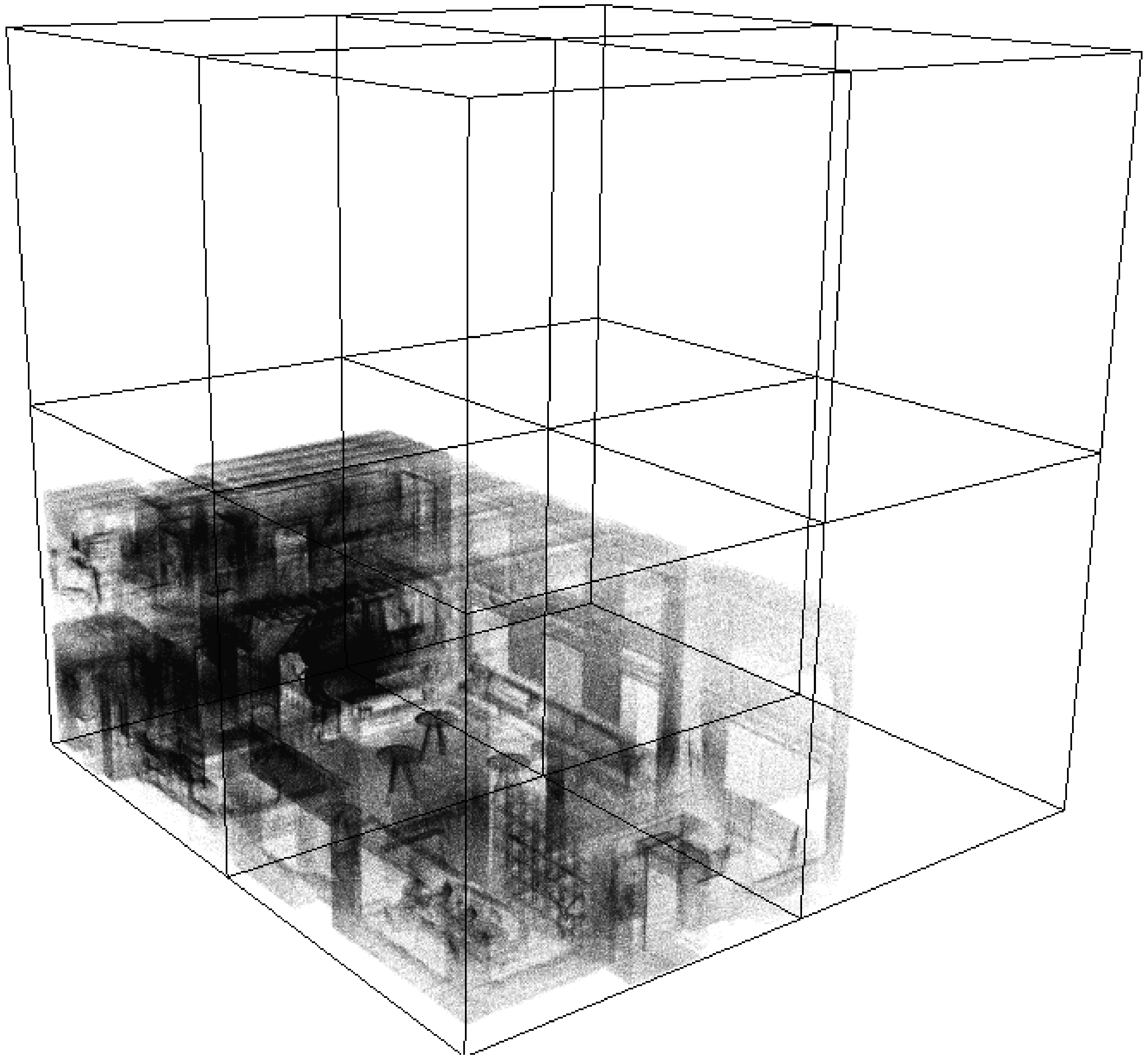
1.3. Octree

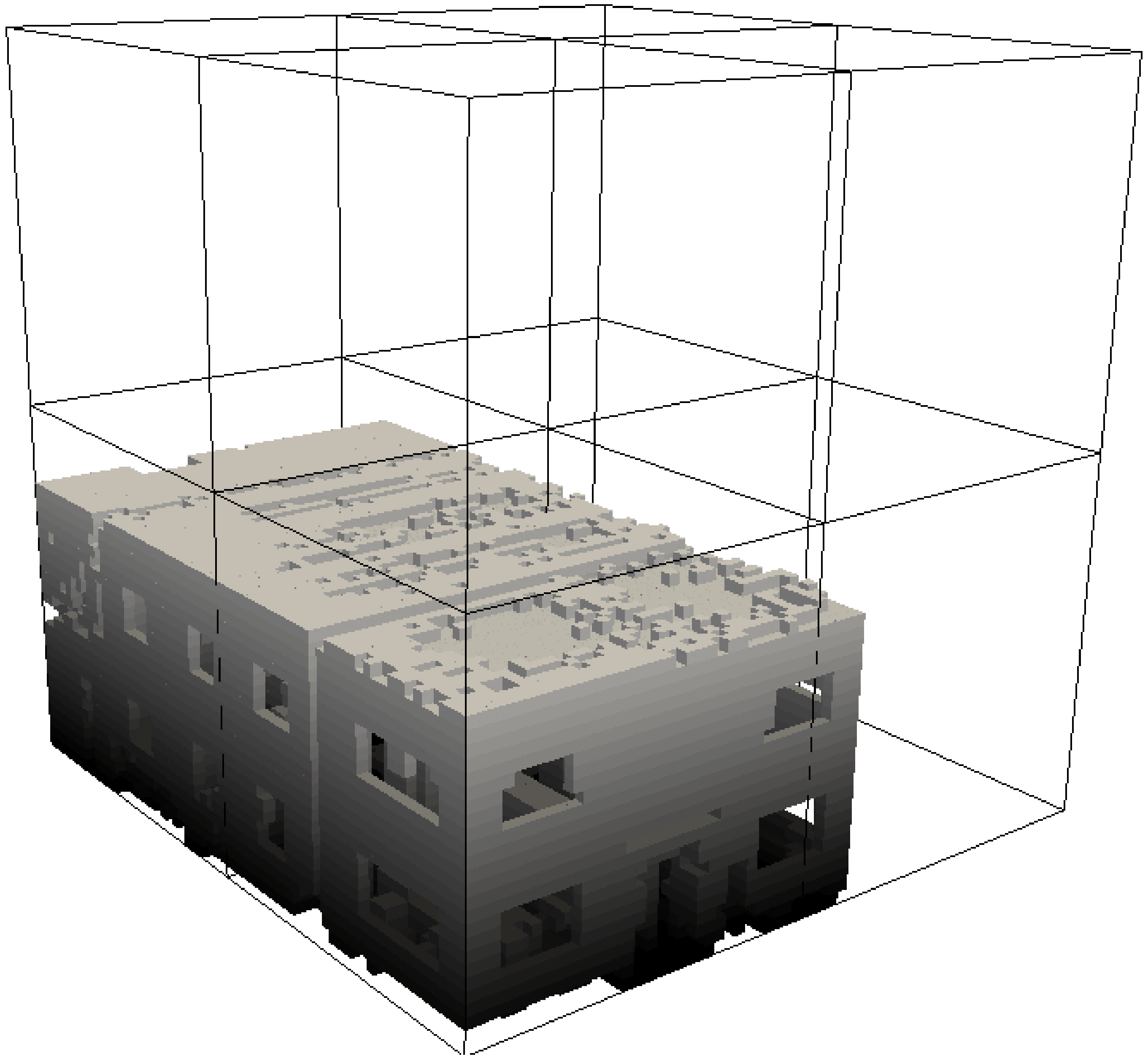
Structure and classify a point cloud



1.3. Octree

Generation: 1) Identify the non empty space

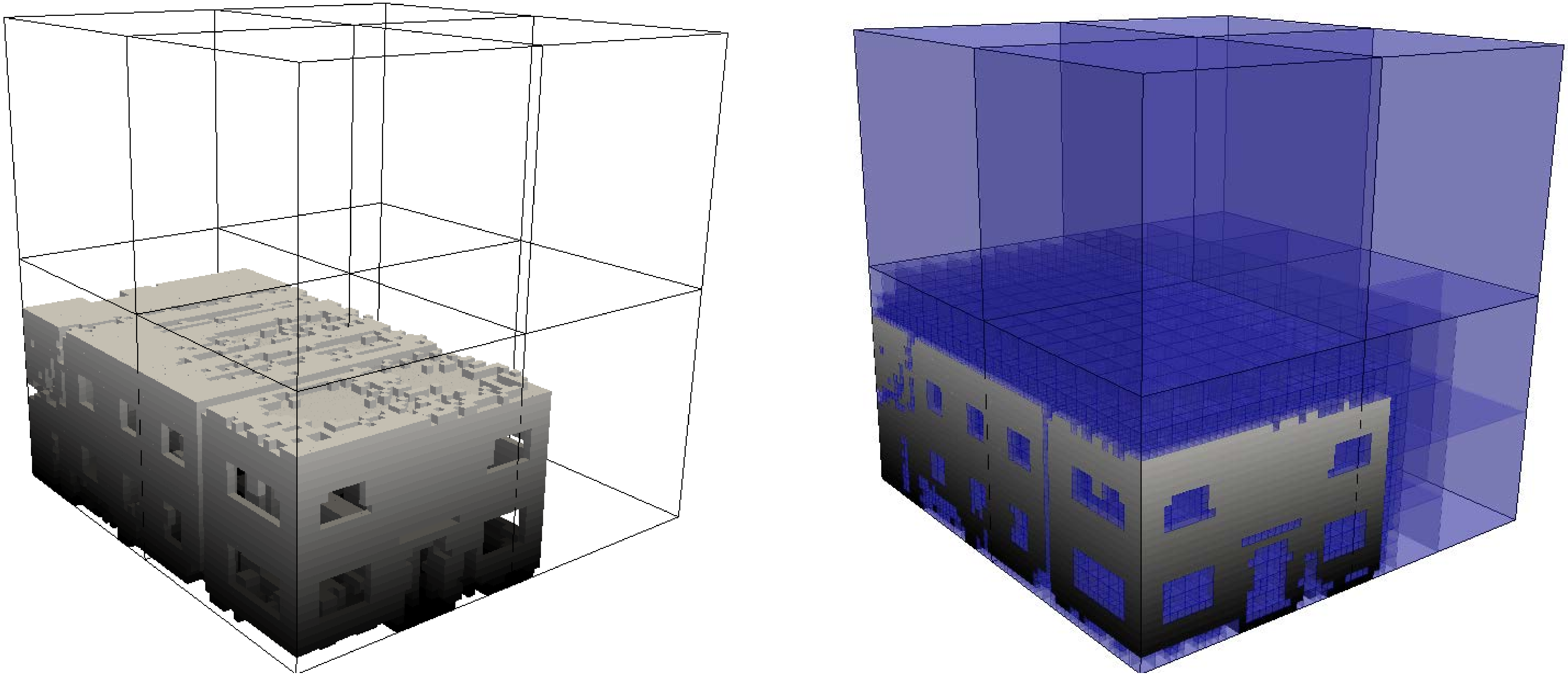




point cloud → non empty space

1.3. Octree

Generation: 2) Derive the empty space



non empty space → empty space

1.3. Octree

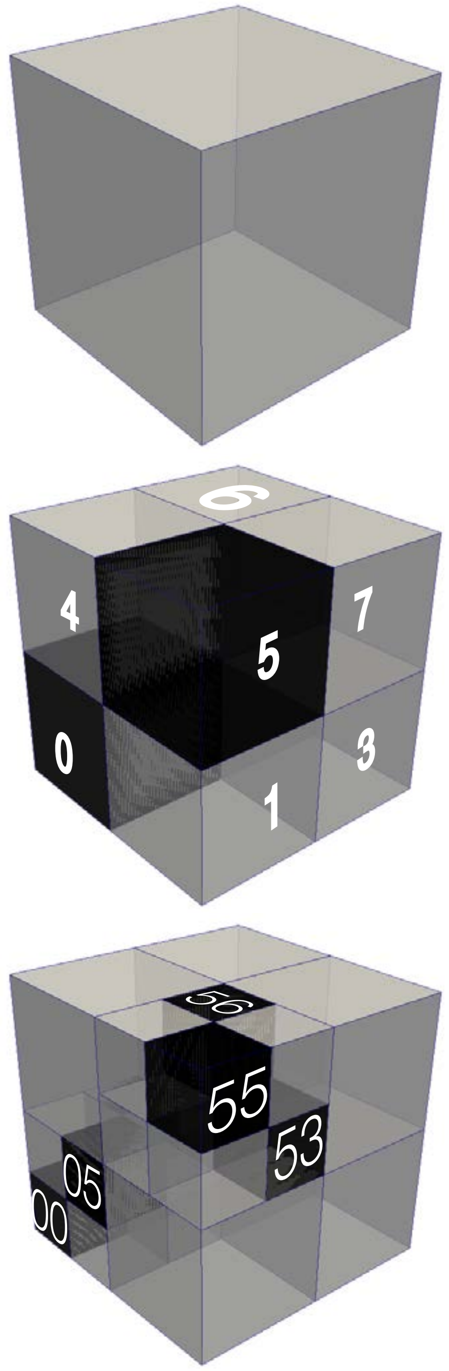
Spatial location code

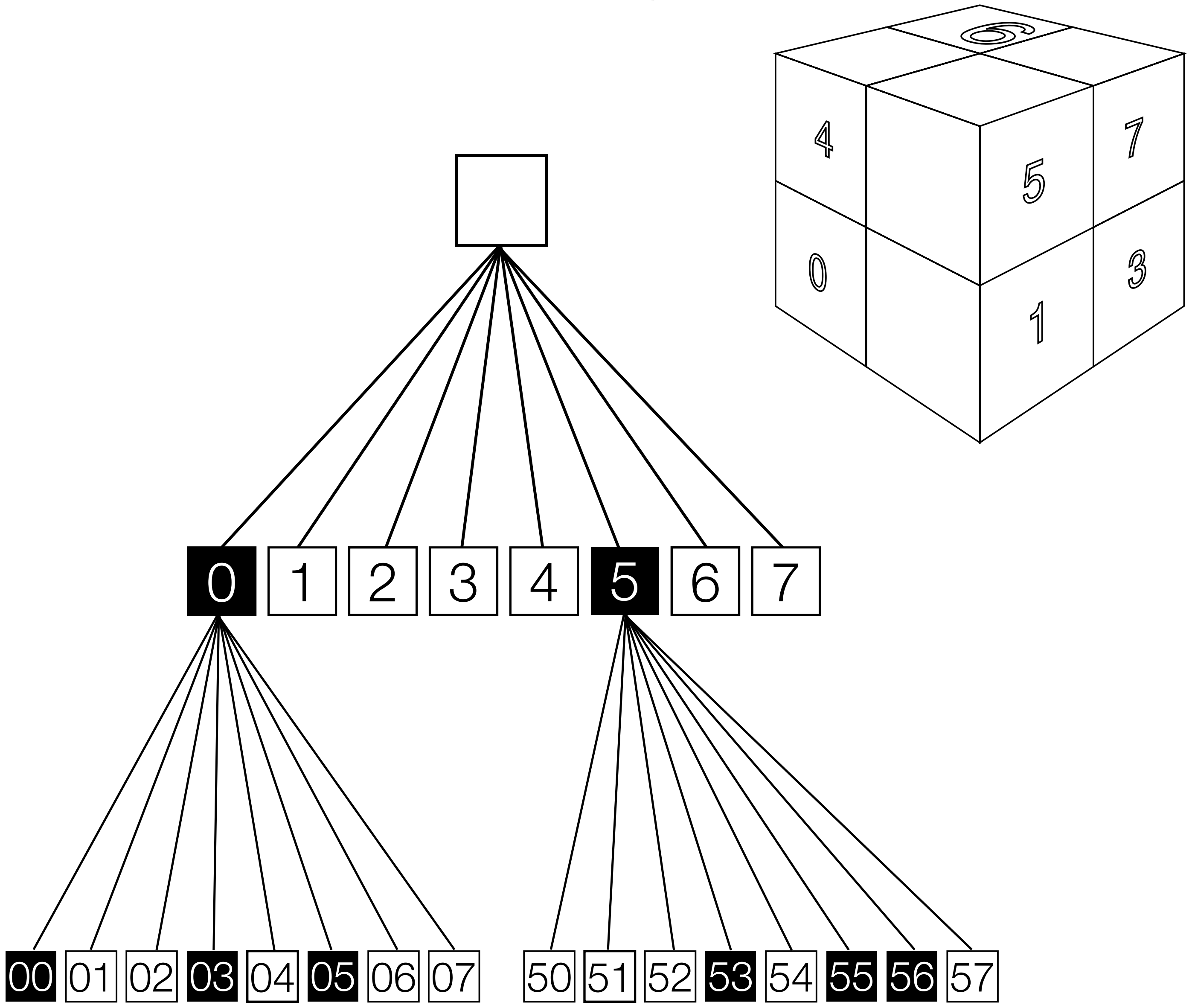
Path from the root node to any leaf node

Octree depth = 0

Octree depth = 1

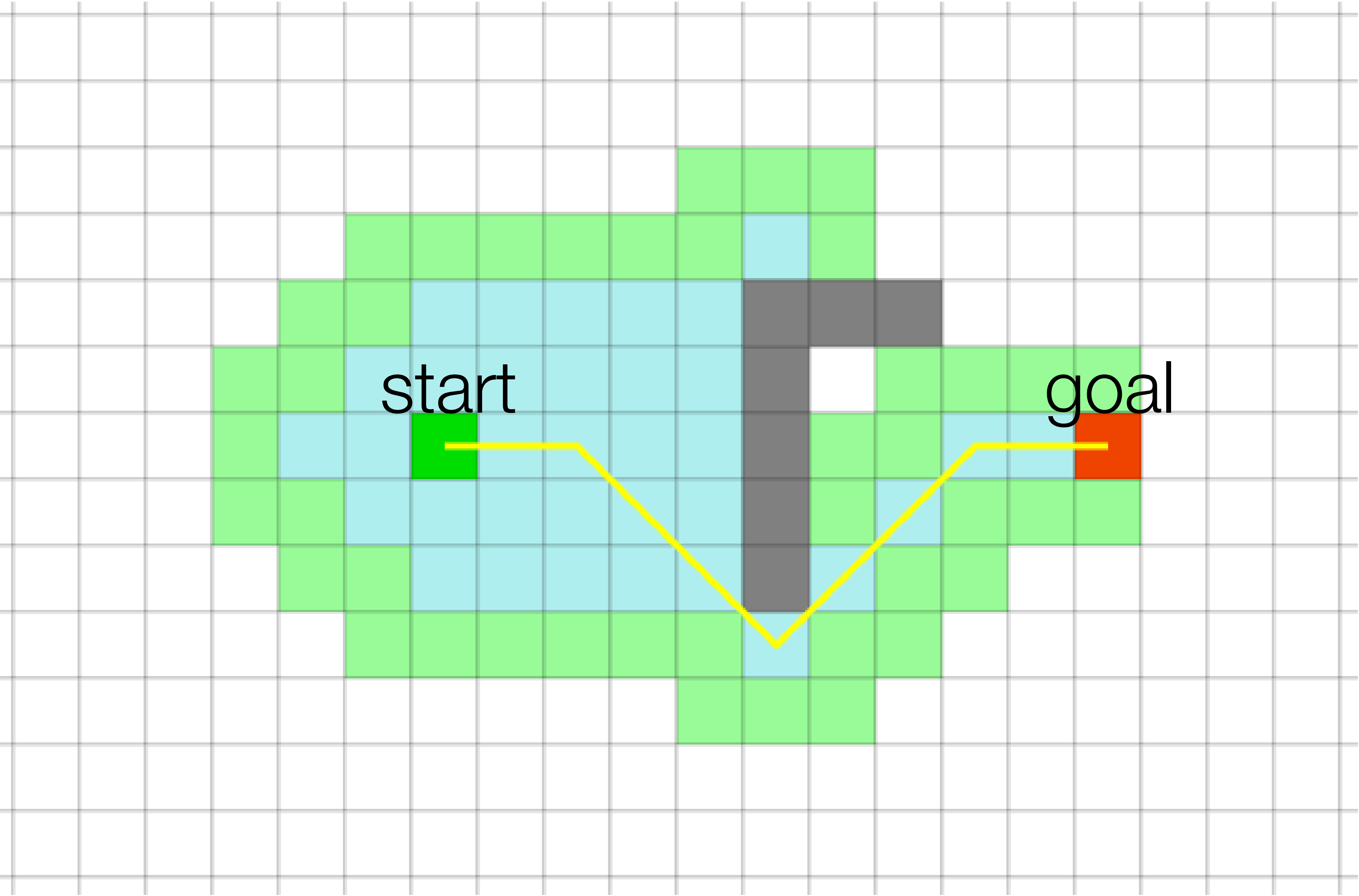
Octree depth = 2



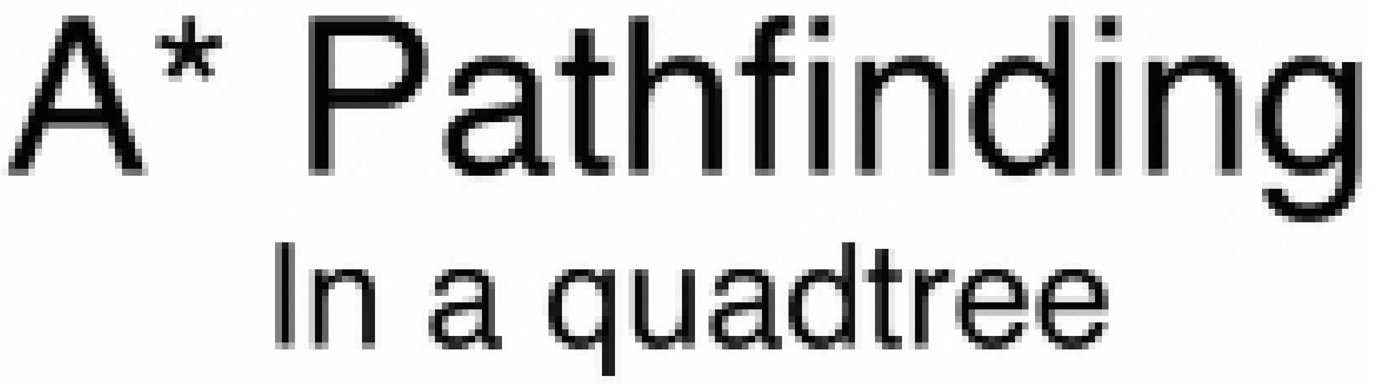


1.4. Path finding

Finding an optimal (collision free) path between a start and goal point.



1.4. A* path finding



A* Pathfinding

In a quadtree

2. Research objectives
Objectives

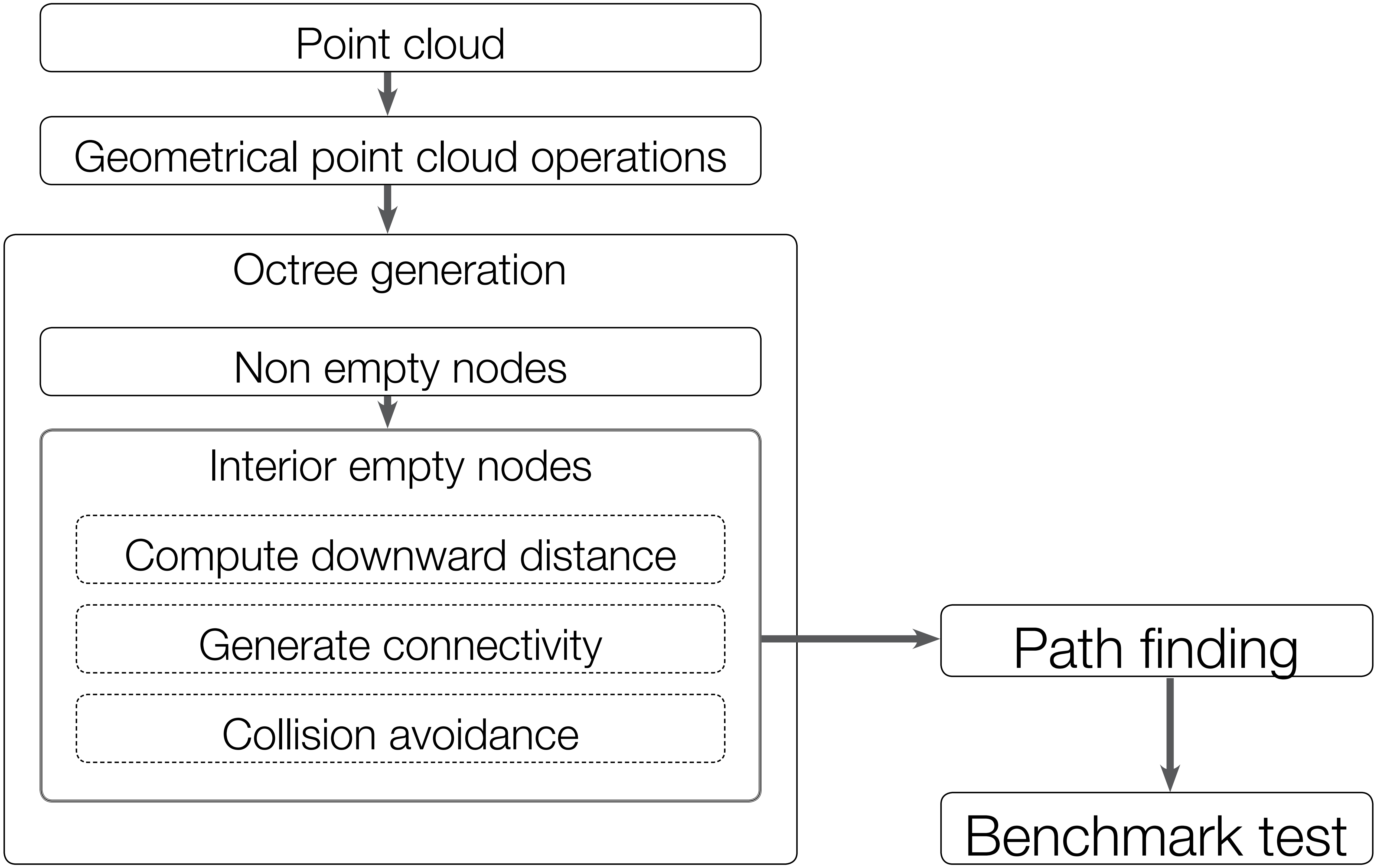
1. Workflow for path finding through an octree representation of an indoor point cloud

2. Identify the effects of:
   2.1. Geometrical point cloud operations on octree generation
   2.2. Octree operators on A* path finding.
   2.3. A* operations on A* path finding.

3. Methodology

3.1. Geometric point cloud processing operations
3.2. Interior empty space
3.3. Connectivity generation
3.4. Collision avoidance
3.5. Distance types
3.6. Benchmark tests
3.7. Point cloud datasets

3. Methodology



Point cloud

Geometrical point cloud operations

Octree generation

Non empty nodes

Interior empty nodes

Compute downward distance

Generate connectivity

Collision avoidance

Path finding

Benchmark test

3.1. Geometric point cloud processing operations

- Maximizing the spatial resolution
  - Spatial resolution: the size in a point cloud represented by the smallest leaf nodes

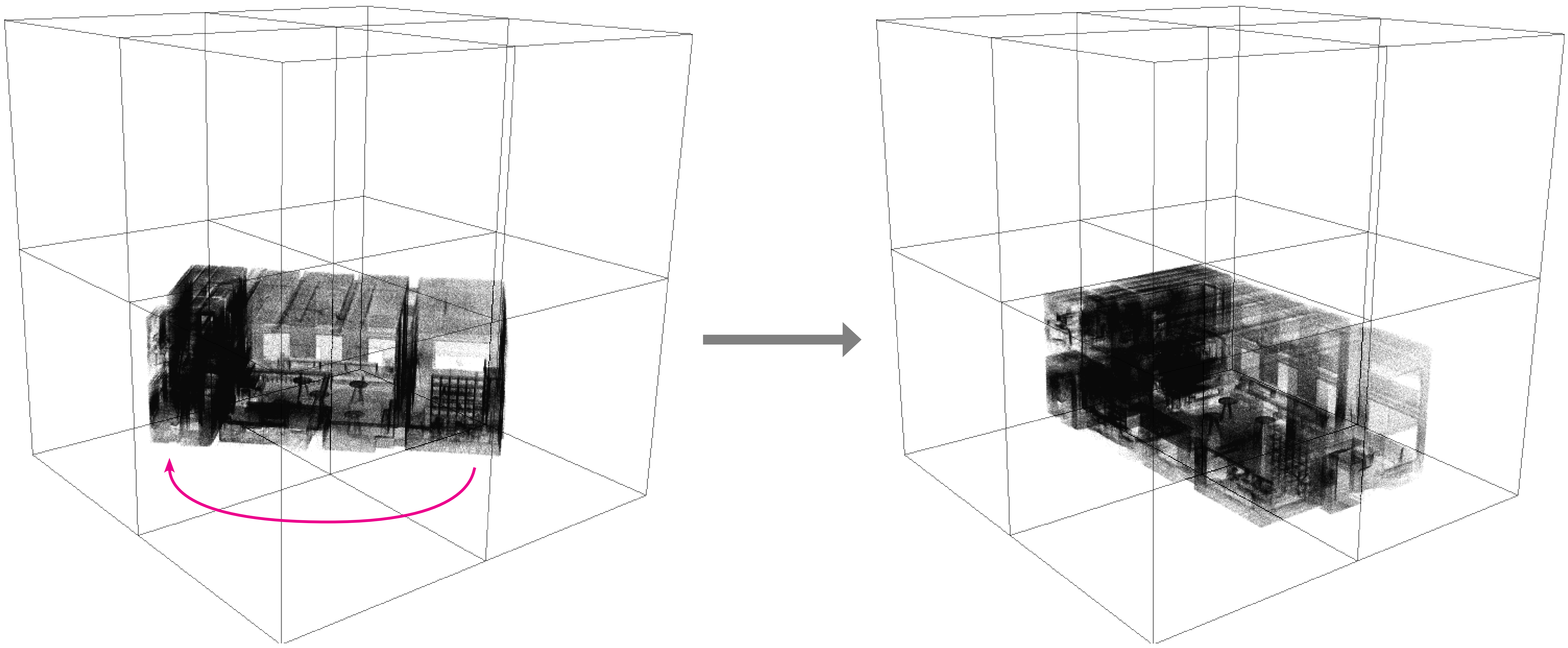
- Minimize the octree depth needed for path finding

Work flow:

1. Align point cloud to octree axis
2. Translated point cloud so origin is in coordinate: (0,0,0)
3. Scale so it fits in a grid of 2^n * 2^n * 2^n, where n is the ocree depth.

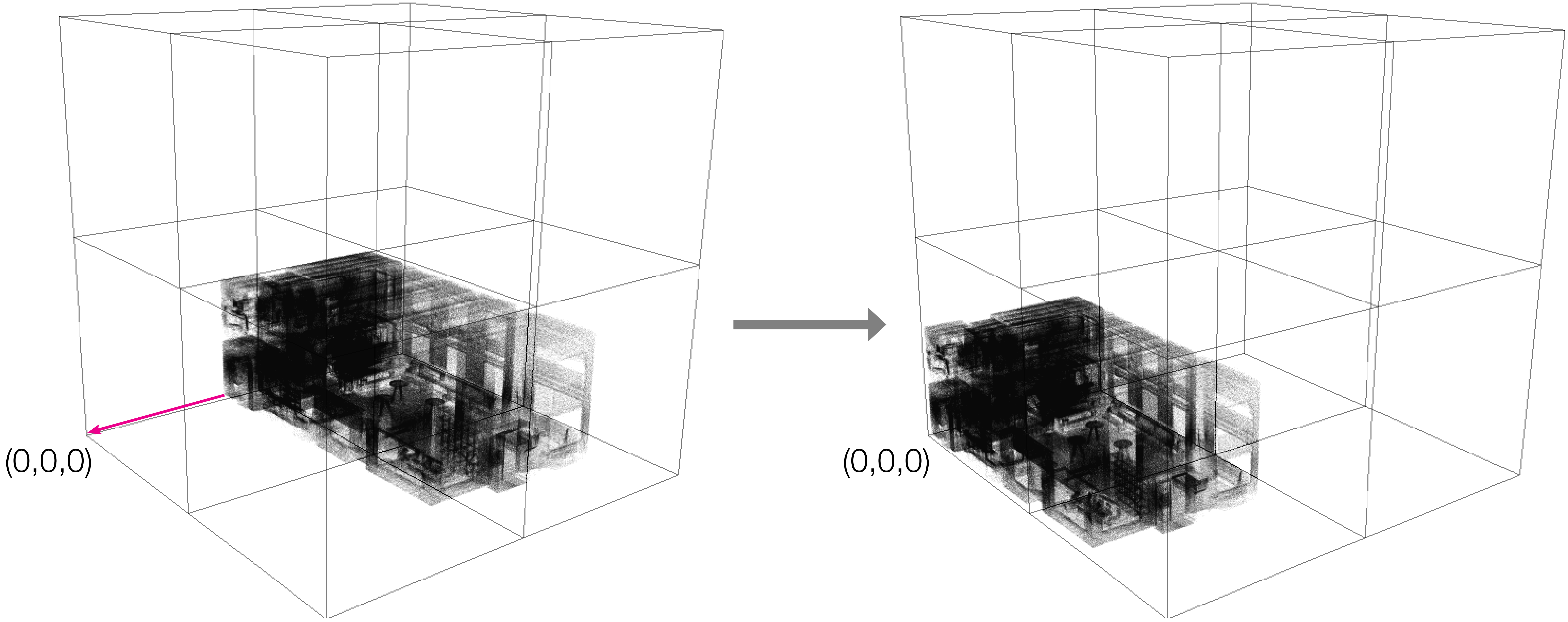
3.1. Geometric point cloud processing operations

Align point cloud to octree axis to minimize the axis-aligned bounding box



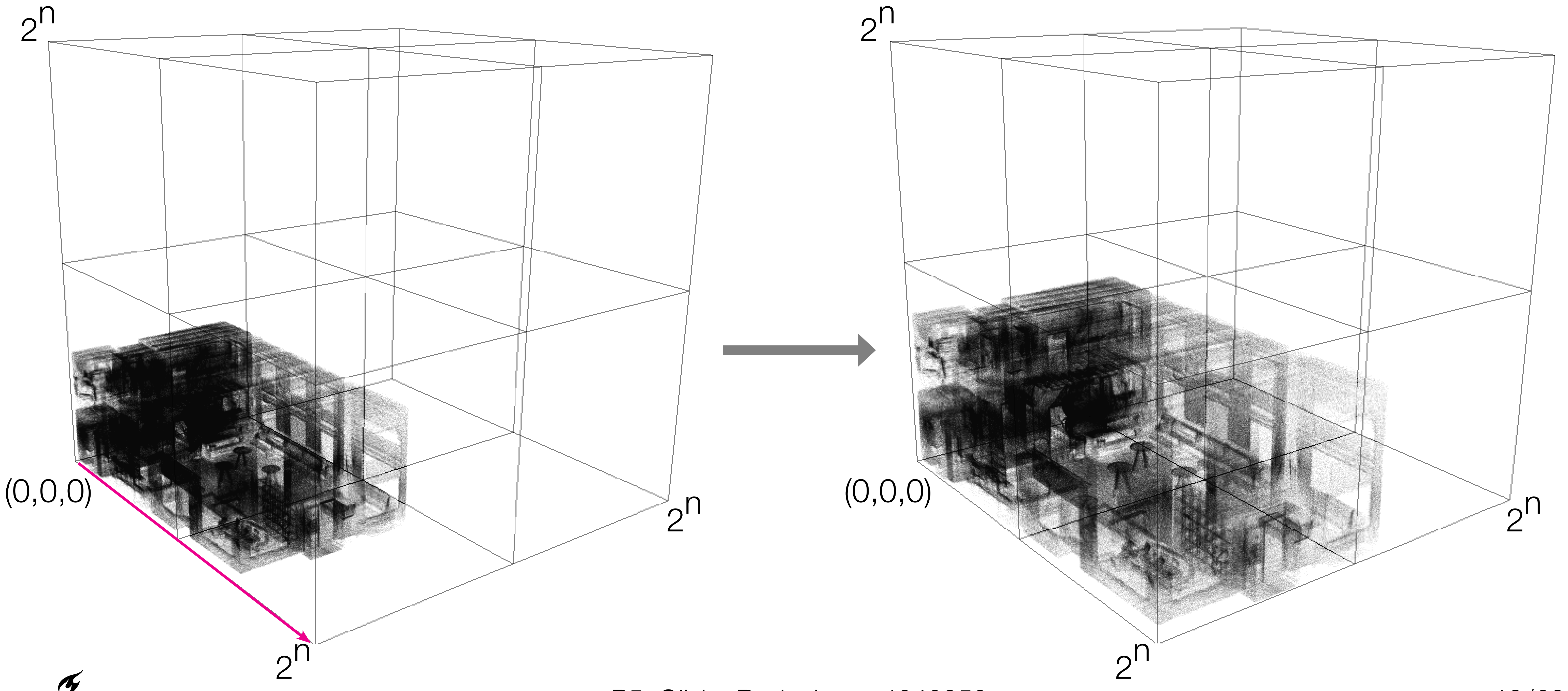
3.1. Geometric point cloud processing

Translate point cloud to origin of octree, as this is the origin of scaling



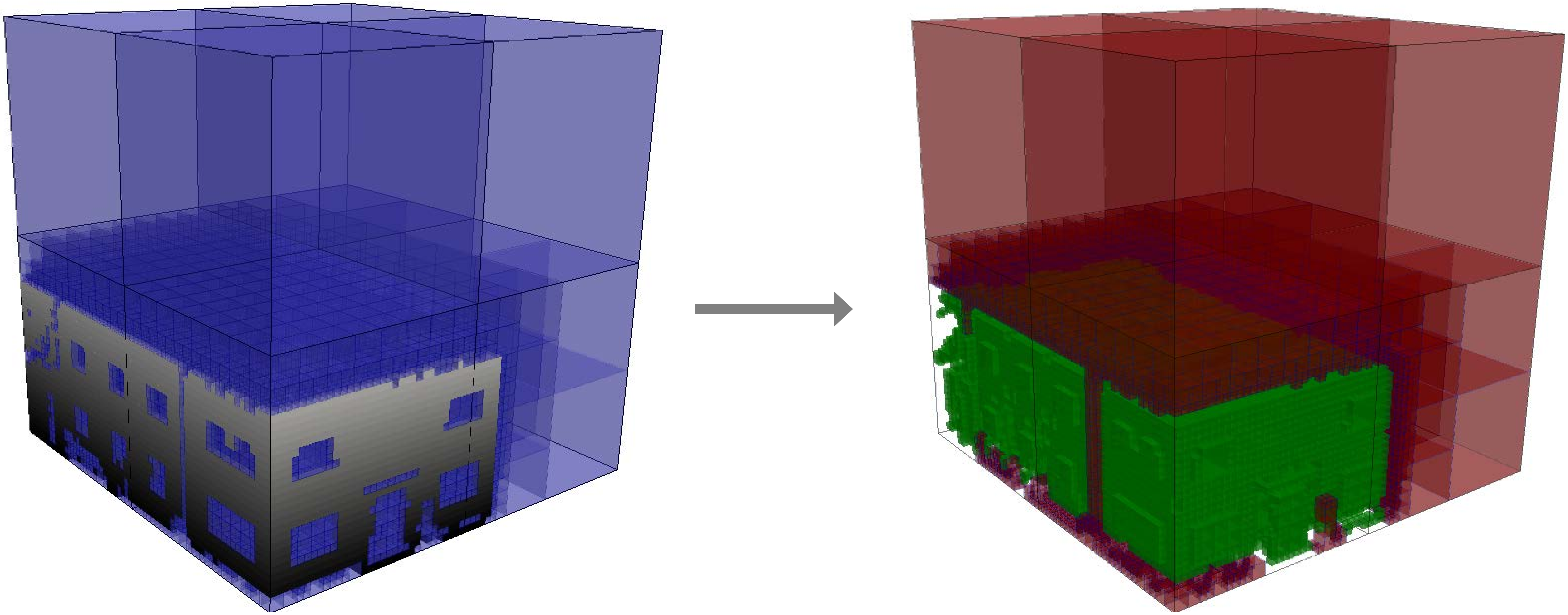
3.1. Geometric point cloud processing

Scale point cloud so it fits in a grid of $2^n * 2^n * 2^n$



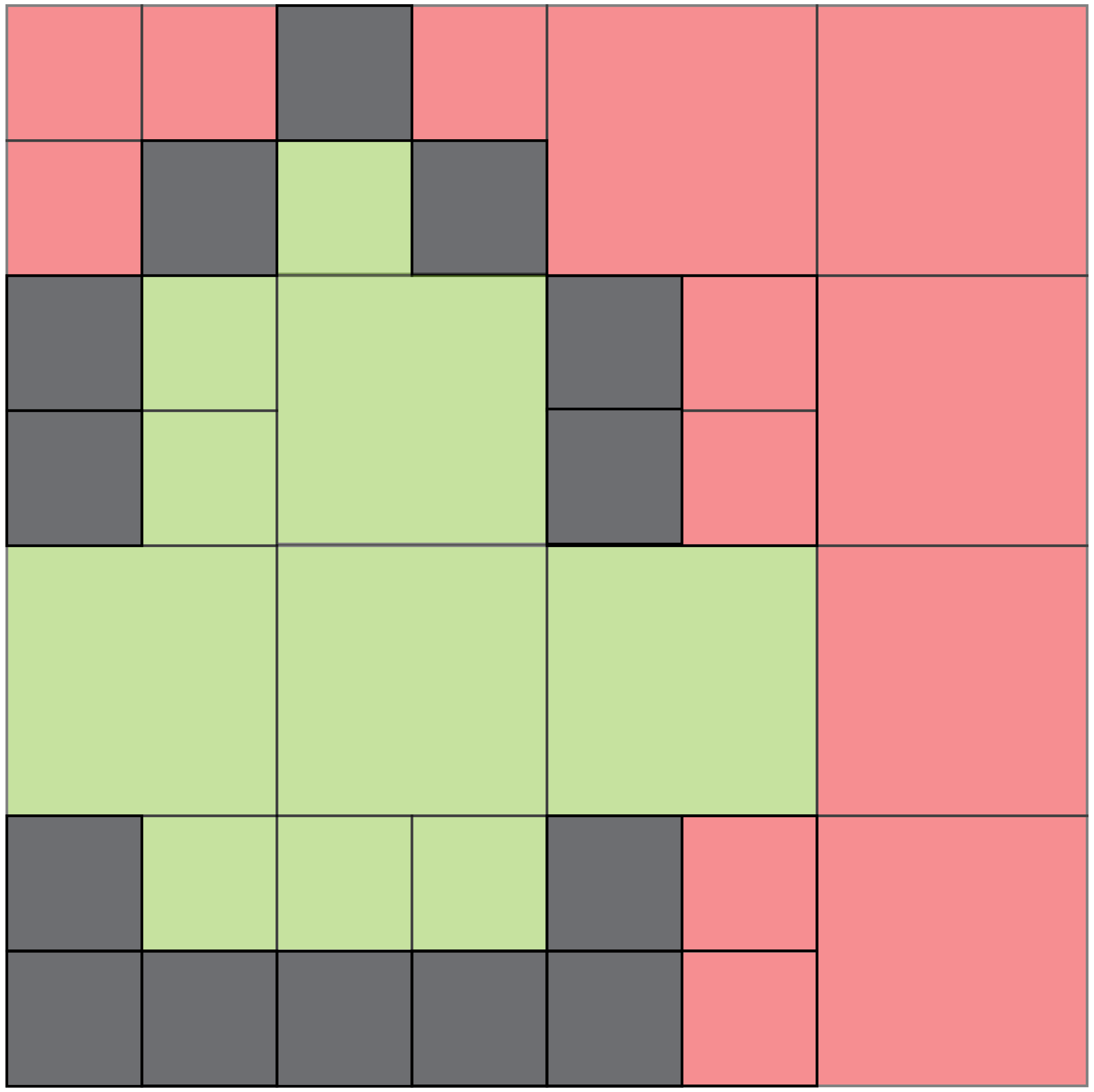
3.2. Interior empty space

Separate the exterior and interior empty space



3.2. Interior empty space

2D section of a building



An empty node is interior if:

- it has a non empty node directly above it
- it has a non empty node directly under it

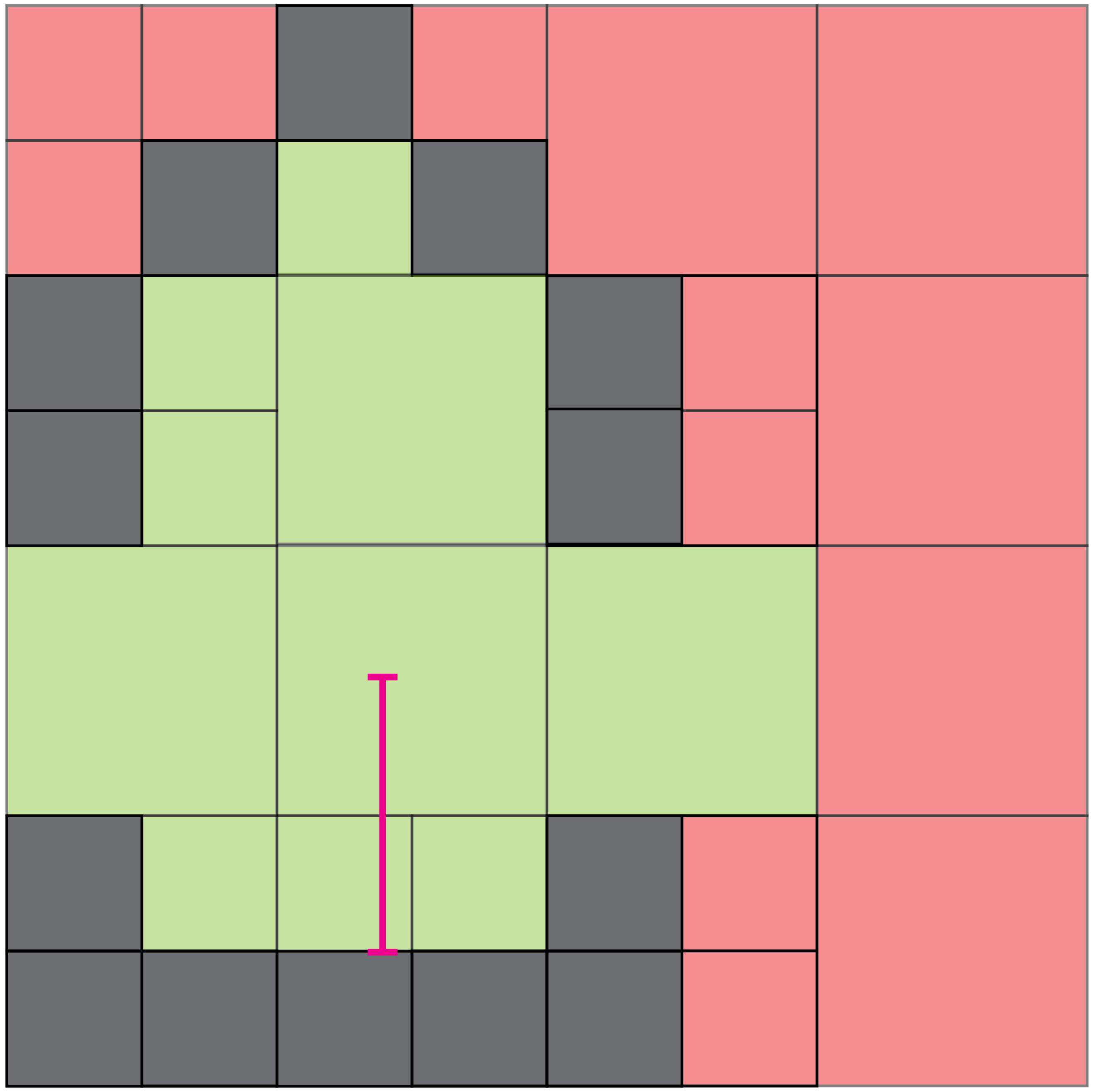
🟩 Interior

🟥 Exterior

3.2. Interior empty space

Downward distance to non empty node

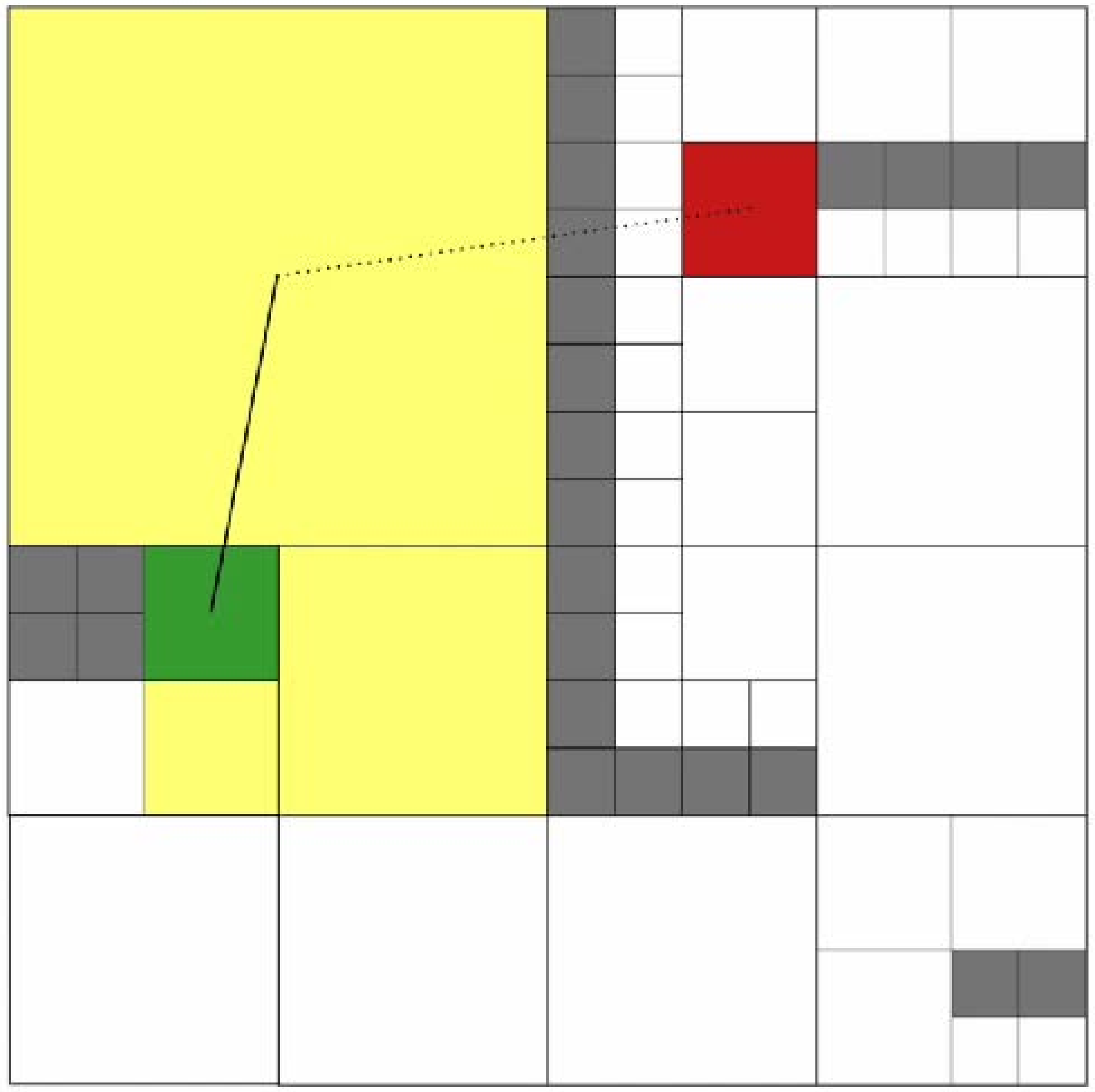
2D section of a building

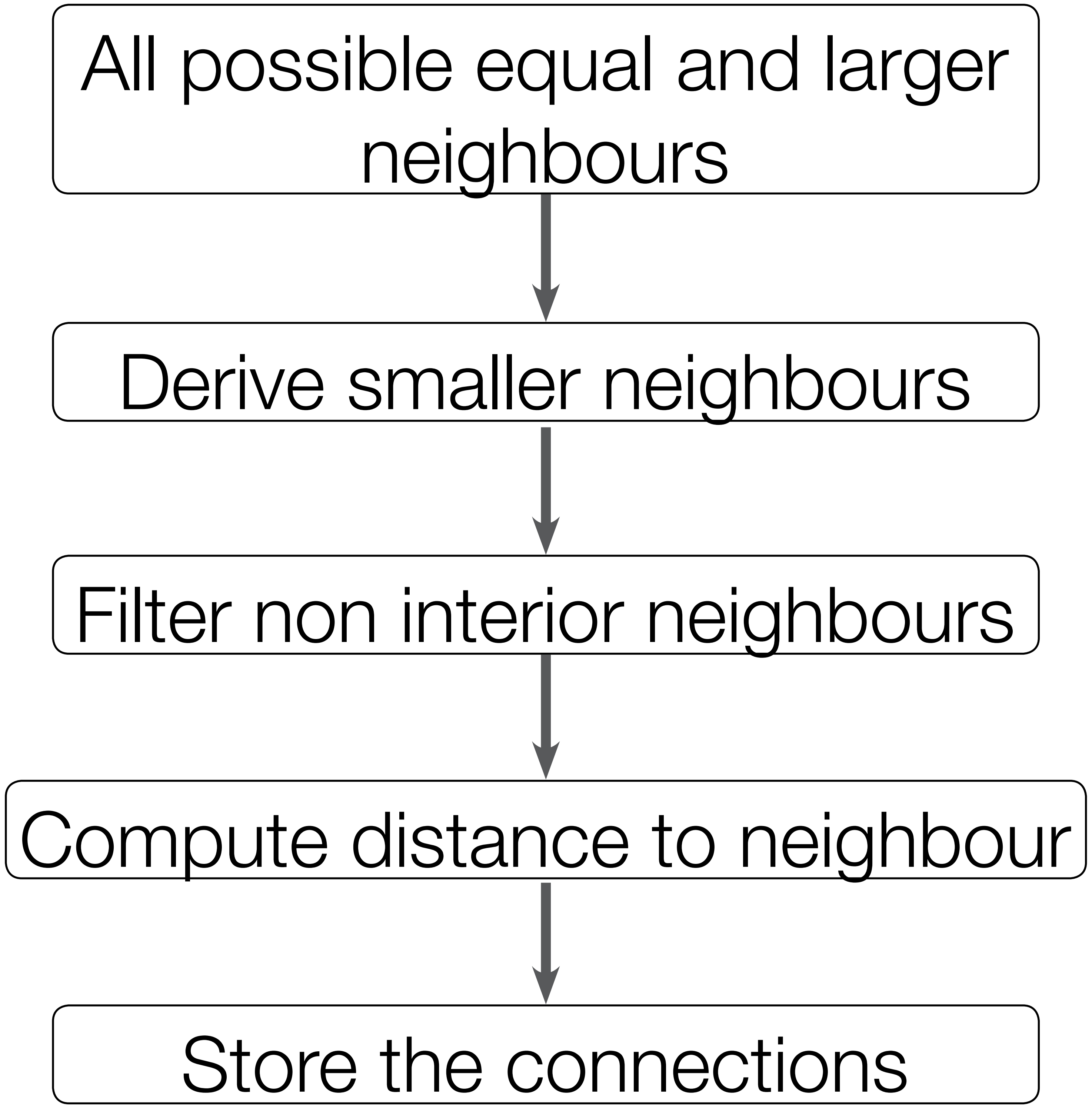


For each interior empty node:

- Compute the downward distance to the closest non empty node
- Use as constraint in path finding

3.3. Connectivity generation





All possible equal and larger neighbours

Derive smaller neighbours

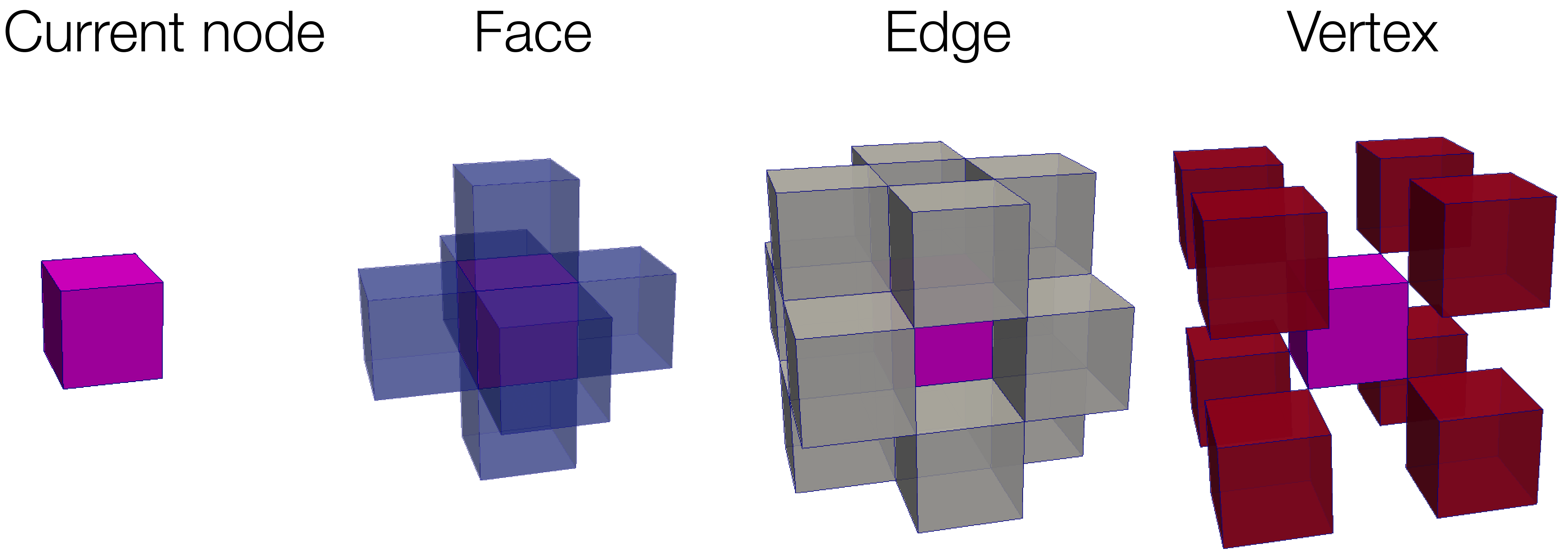
Filter non interior neighbours

Compute distance to neighbour

Store the connections

3.3. Neighbour finding

connectivity between nodes



Current node

Face

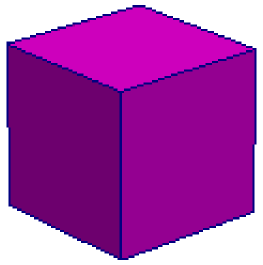
Edge

Vertex

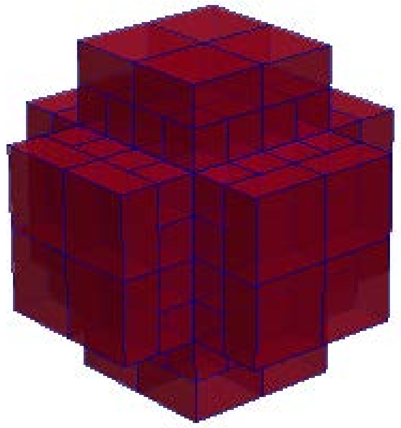
3.3. Neighbour finding

Size

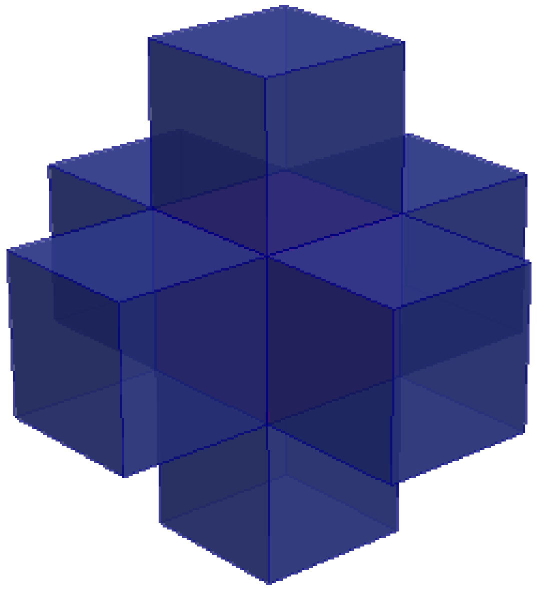
Current node



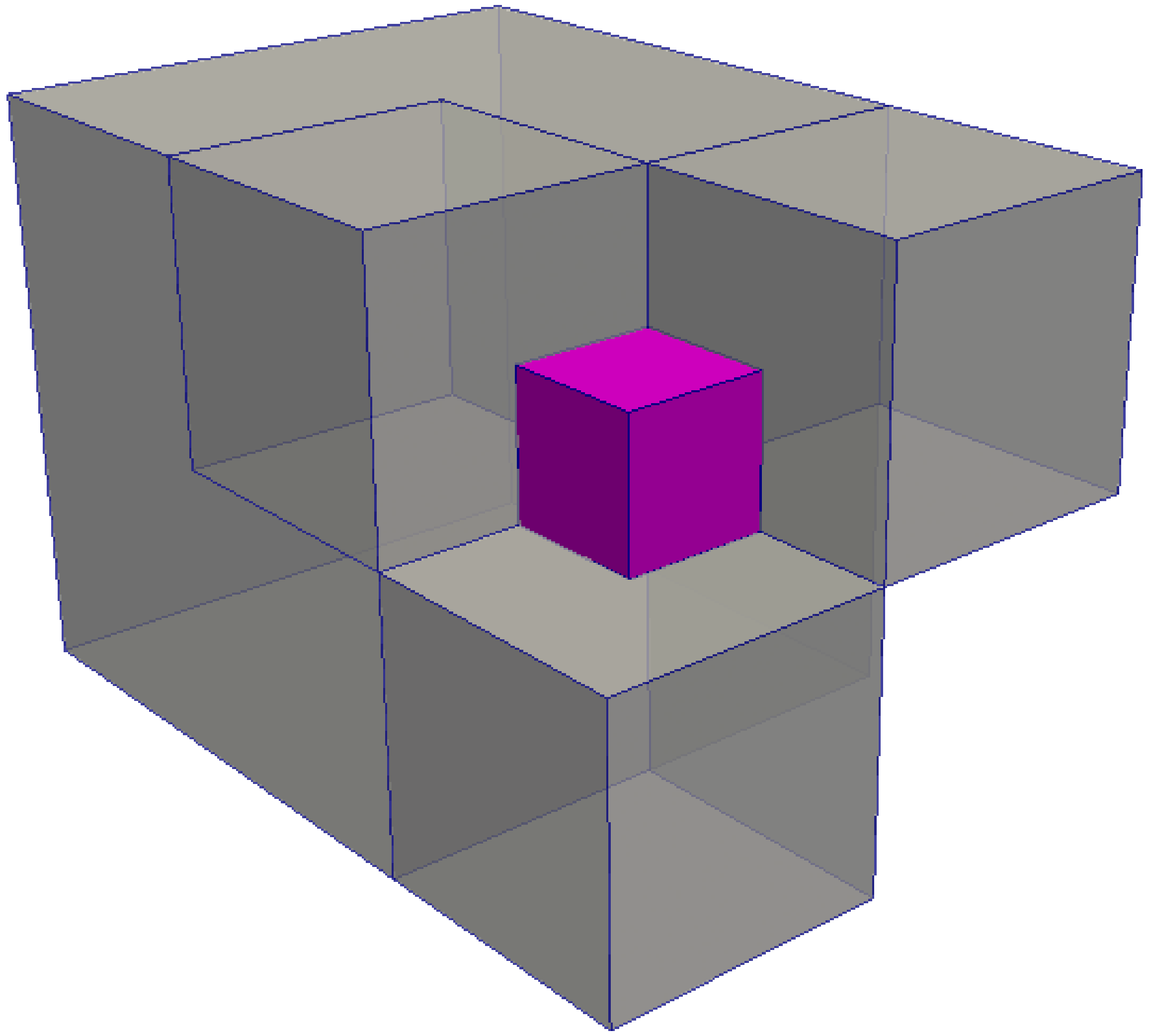
Smaller



Equal

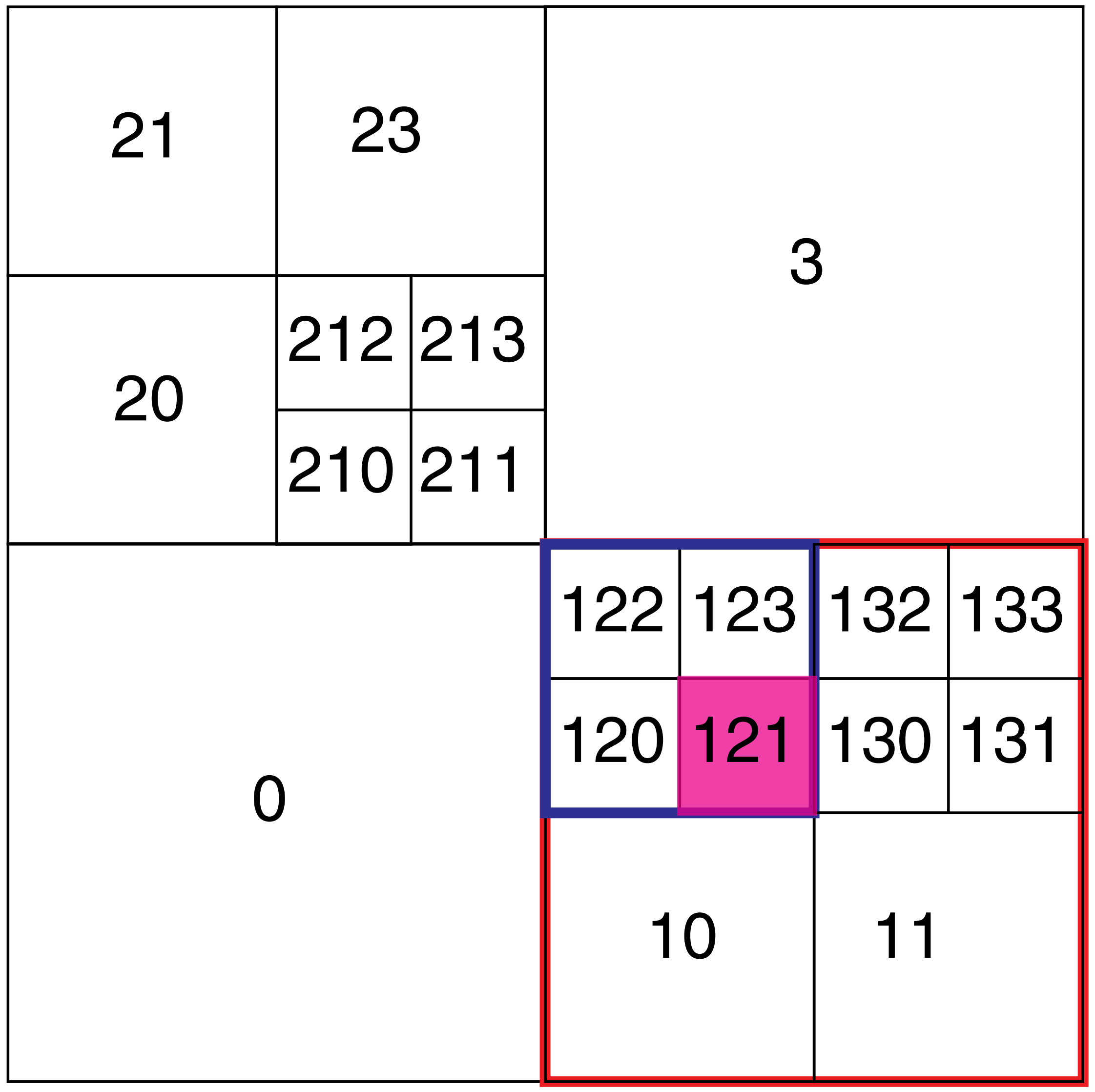


Larger



3. 3 Neighbour finding

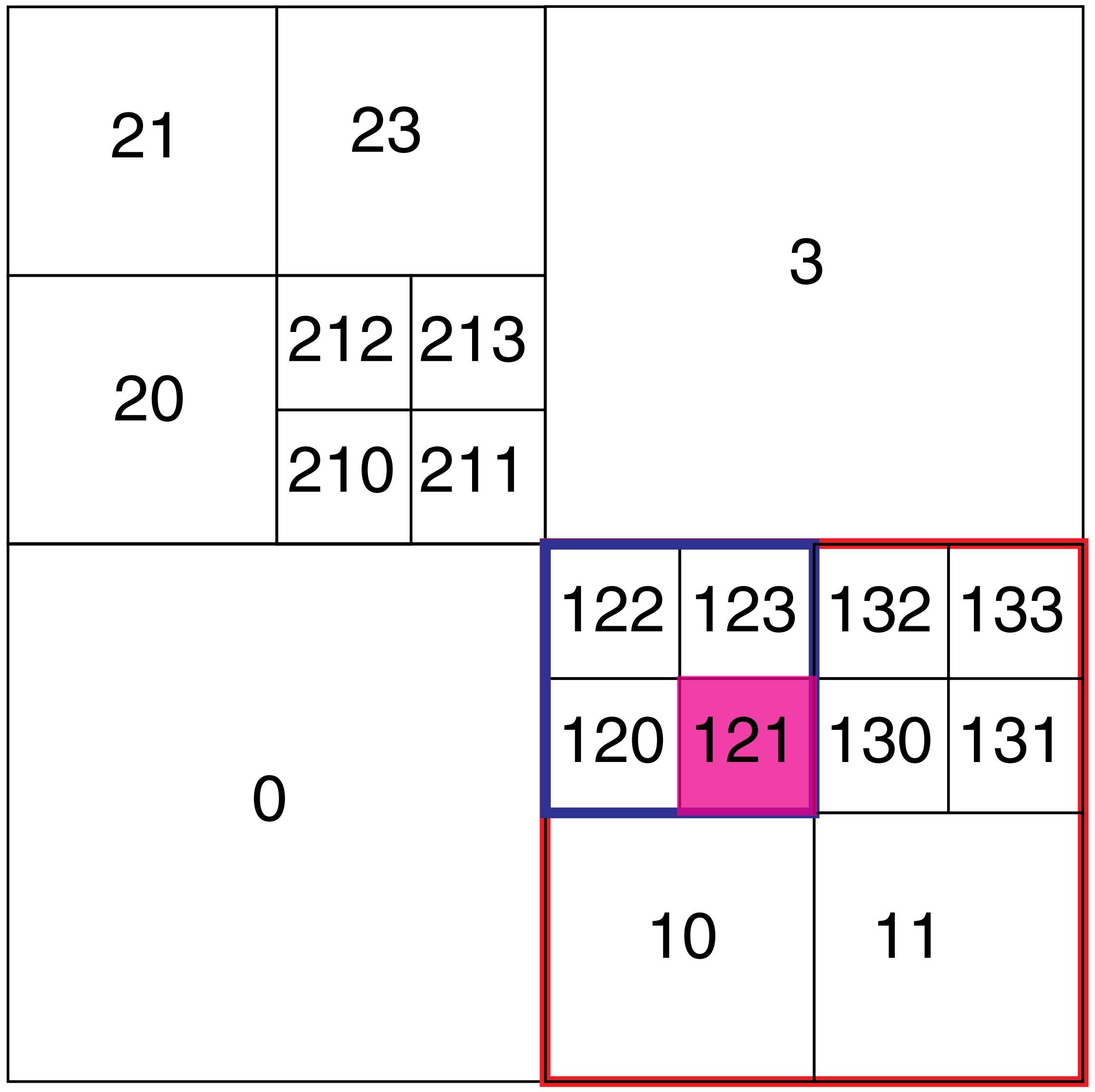
Bases on method of Vörös (2000)



- All smaller, equal and larger face neighbours
- Finds neighbours based on their spatial location code

3.3. Neighbour finding

Equal inner face neighbours



Equal inner face neighbours of node 121:

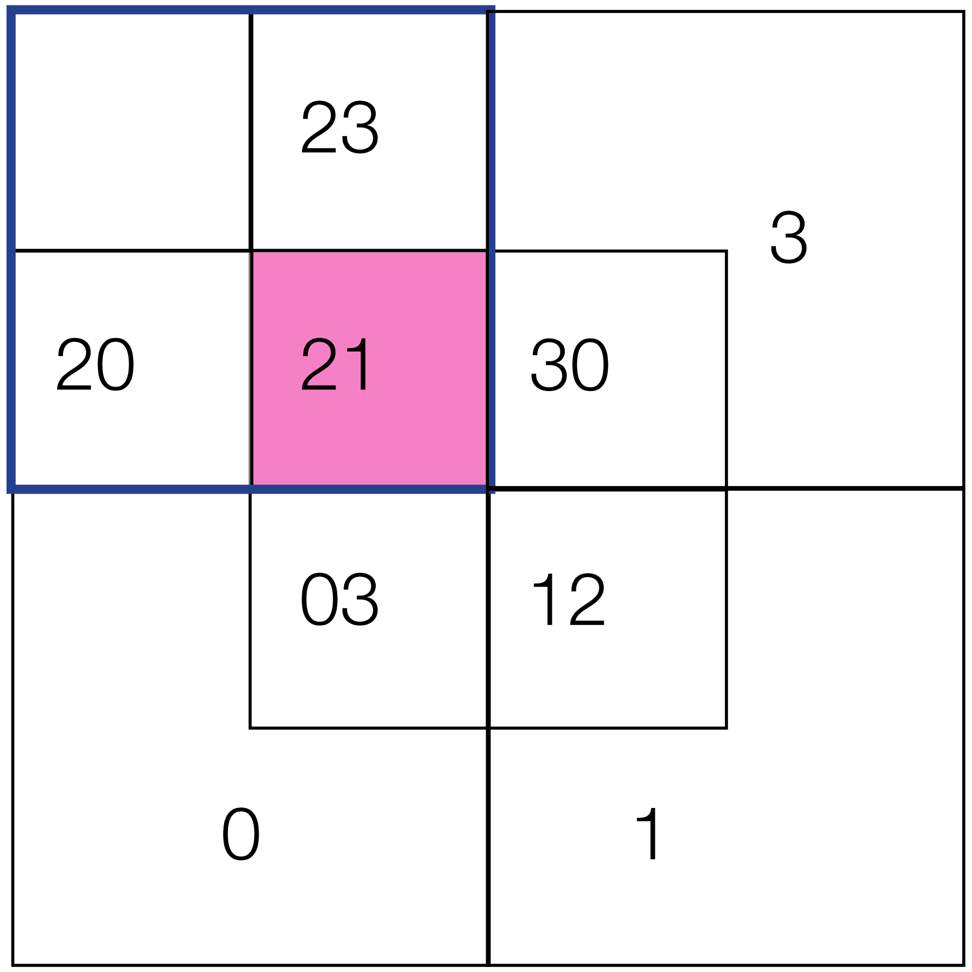
121

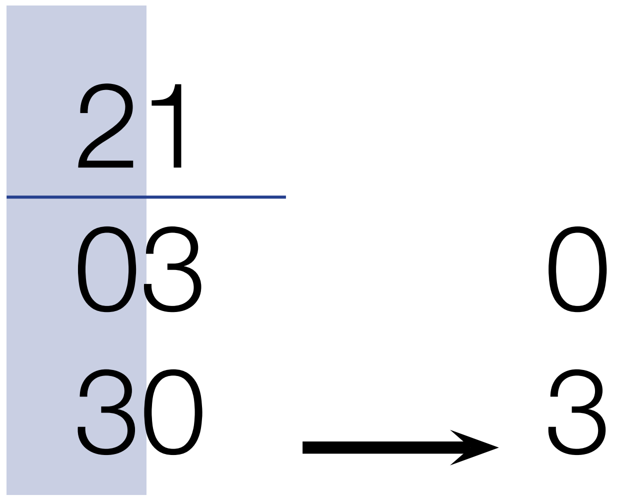
120

123

3.2. Neighbour finding

Larger neighbours





21

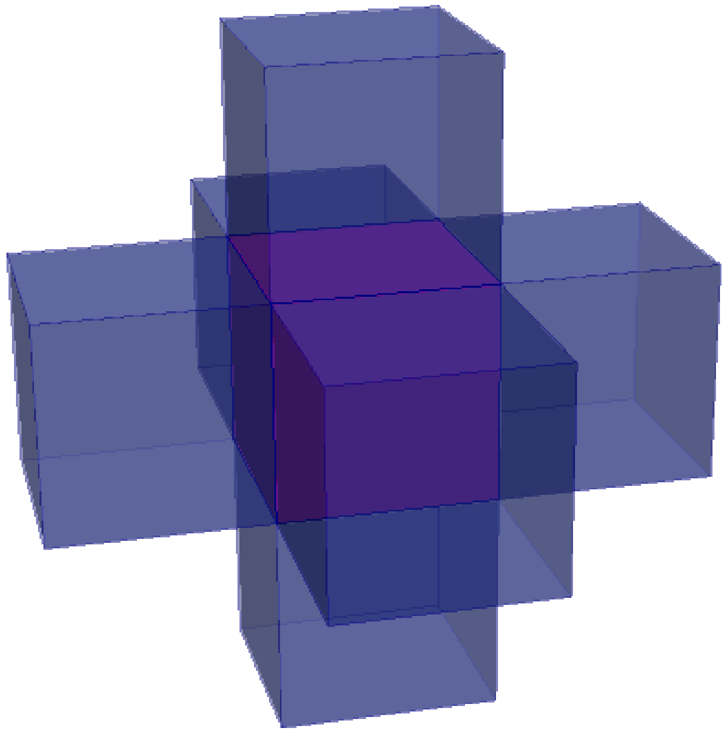
03 0

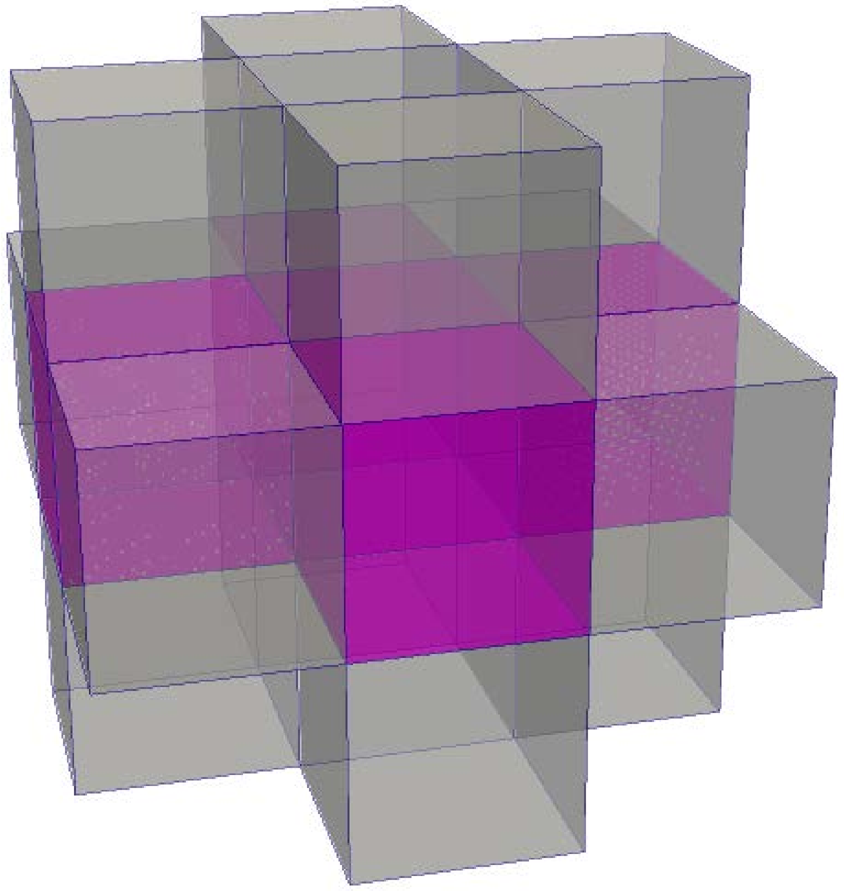
30 → 3

- Larger neighbour are computed by deleting digits from the location code of the equal neighbours

3.3. Neighbour finding

Equal edge neighbours

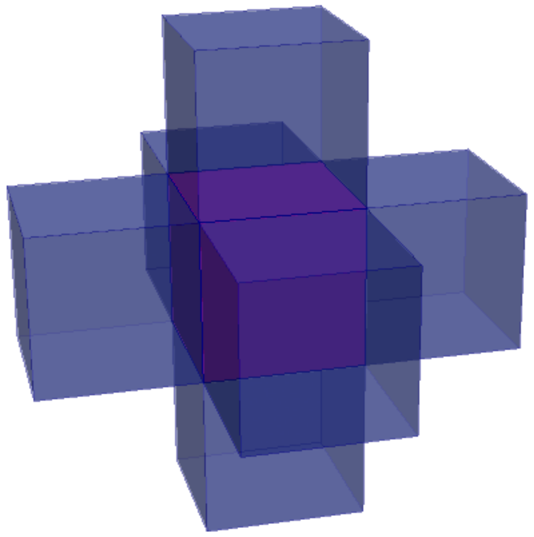




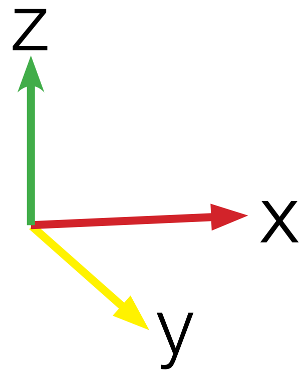
- Based on a face connection between face and edge neighbours

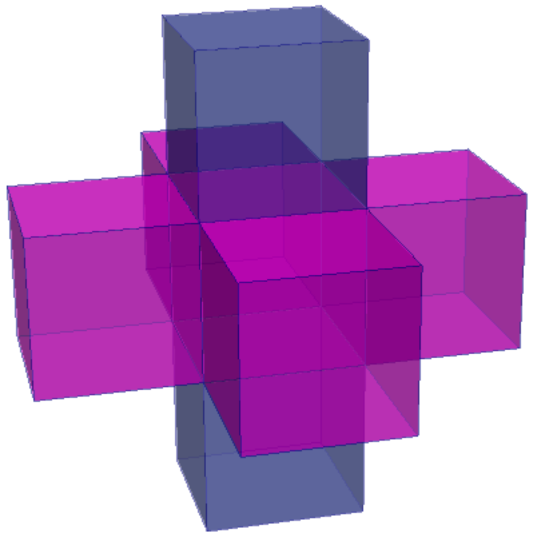
3.3. Neighbour finding

Equal edge neighbours

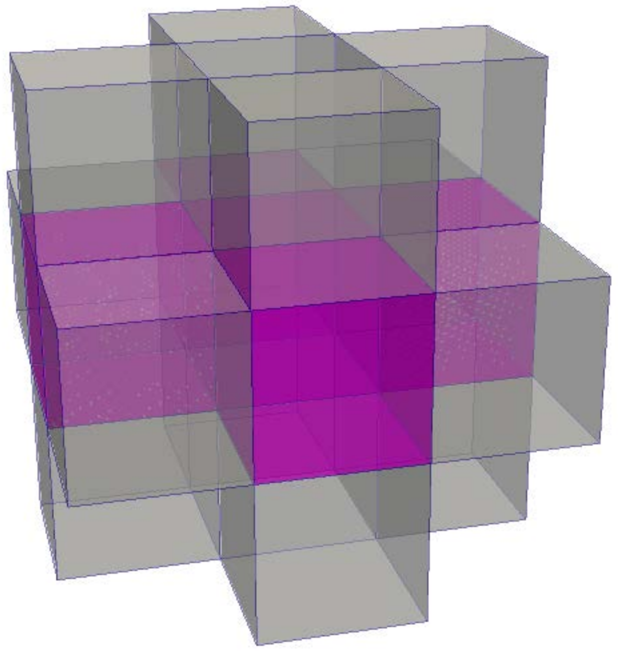


Compute all face neighbours

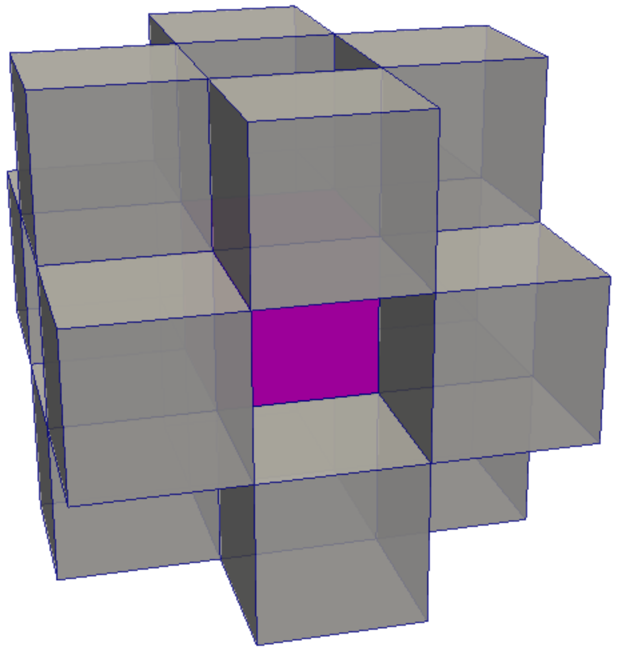




Select face neighbours in x and y direction



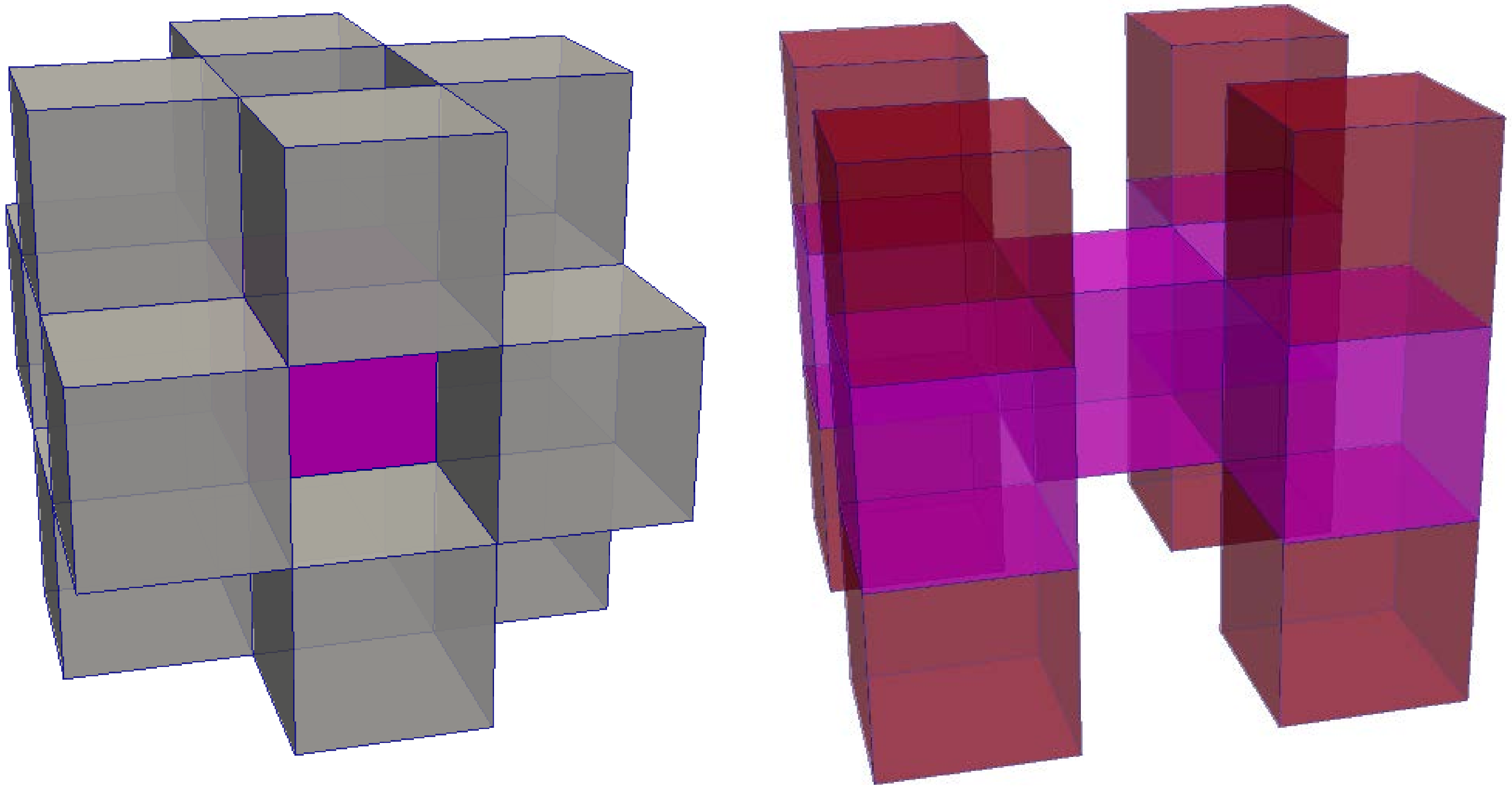
- Compute the y and z neighbours of x face neighbours
- Compute z neighbours of y face neighbour



Edge neighbours

3.3. Neighbour finding

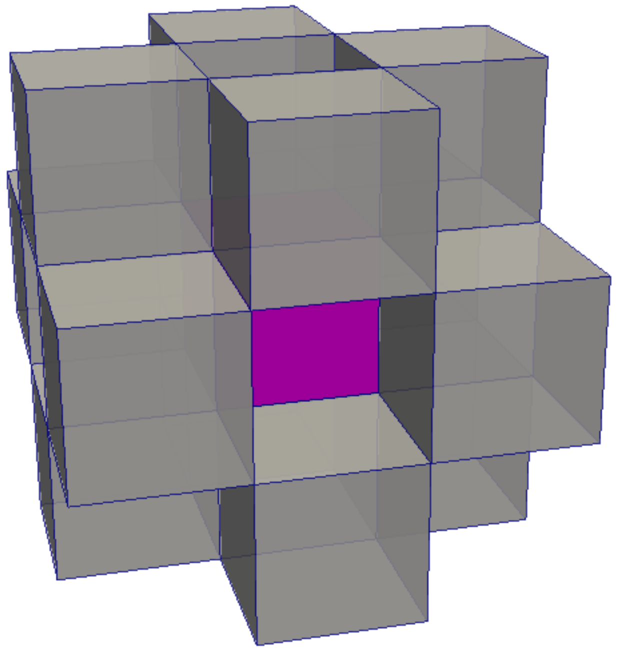
Equal vertex neighbours



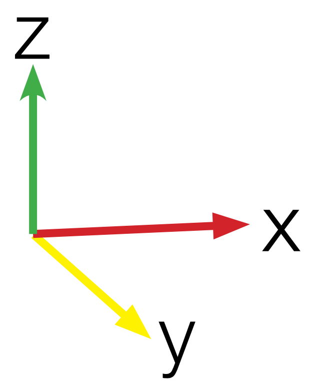
- Based on a face connection between edge and vertex neighbours

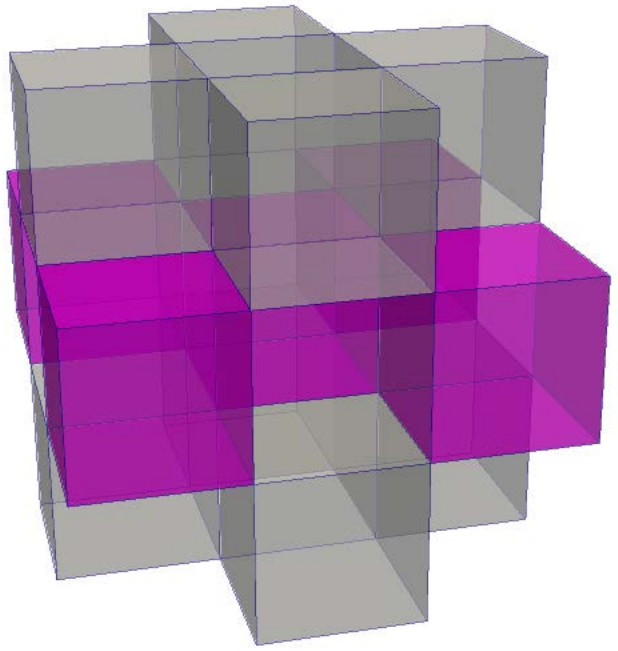
3.3. Neighbour finding

Equal vertex neighbours

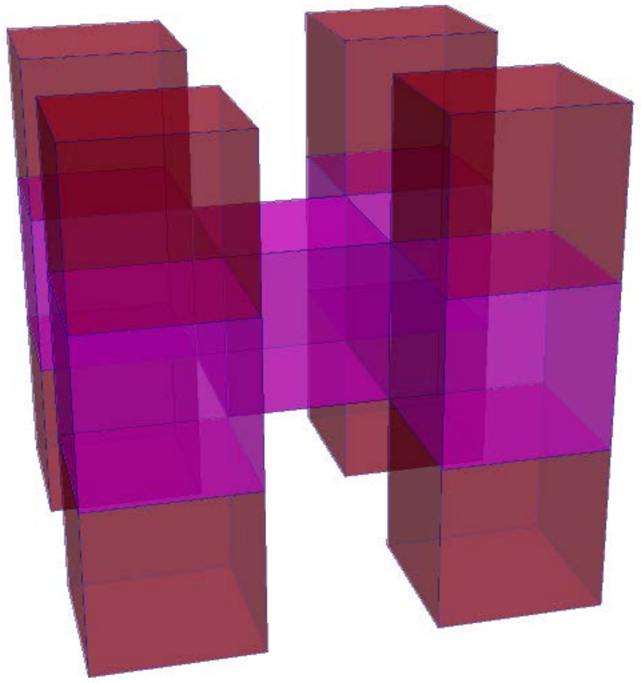


Edge neighbours

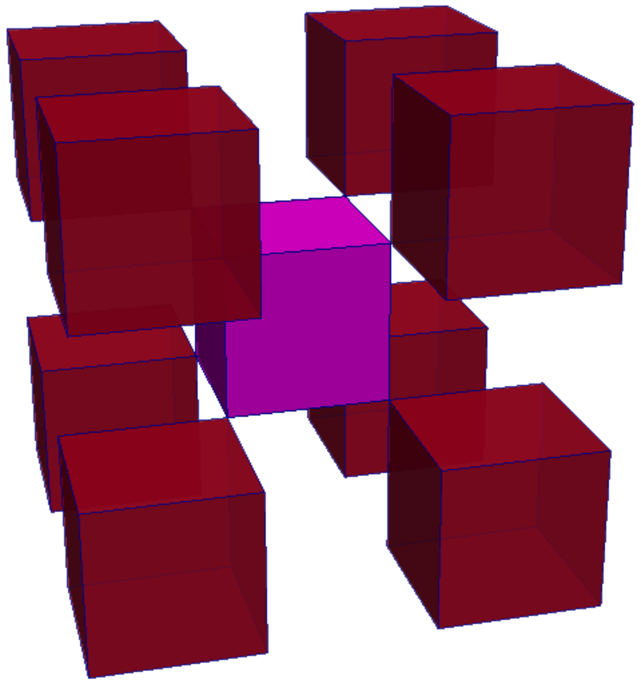




Select edge neighbours in x and y direction

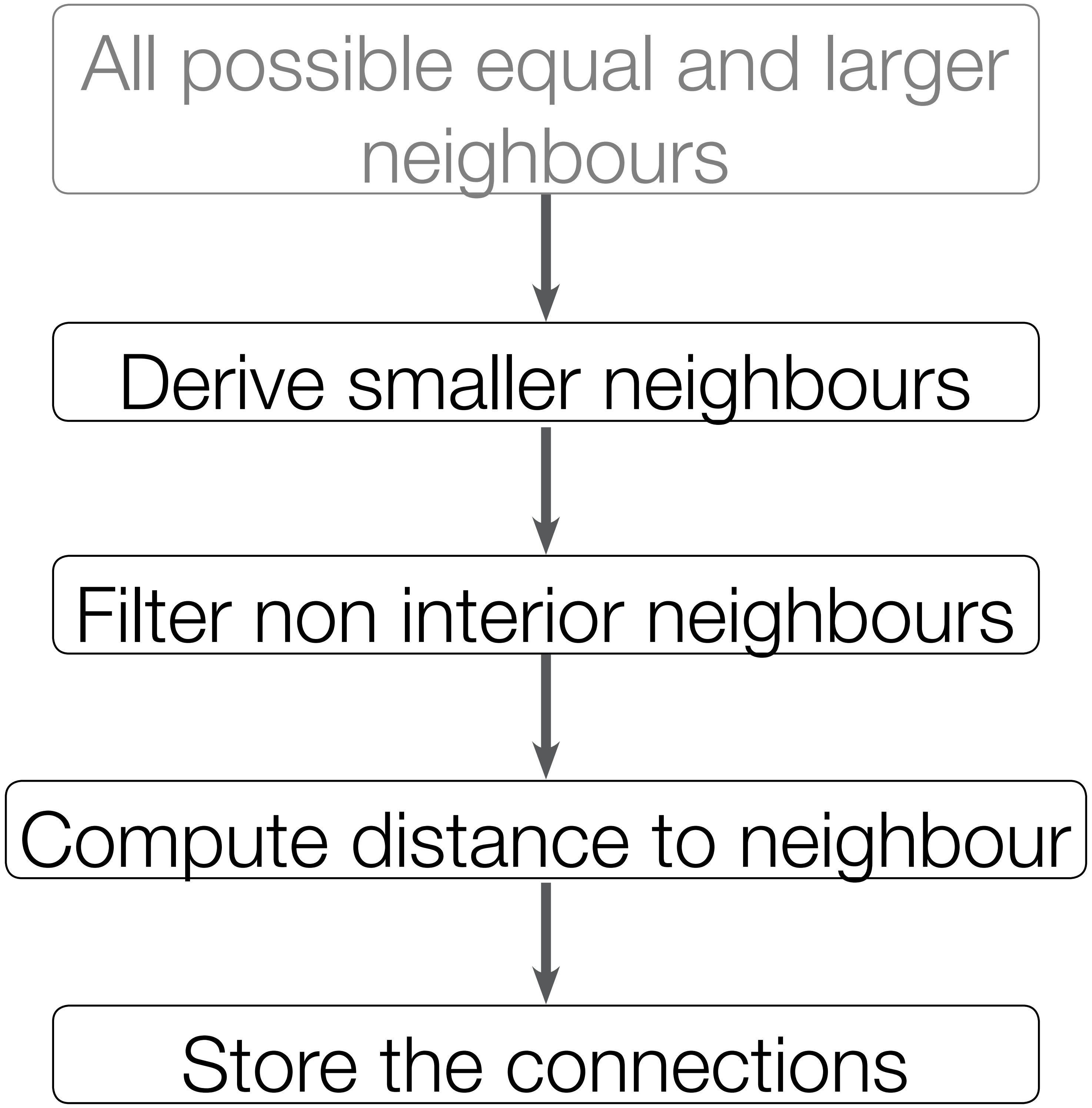


Compute the face neighbours in z direciton of the selected edge neighbours



Vertex neighbours

3.3. Connectivity generation



All possible equal and larger neighbours

Derive smaller neighbours

Filter non interior neighbours

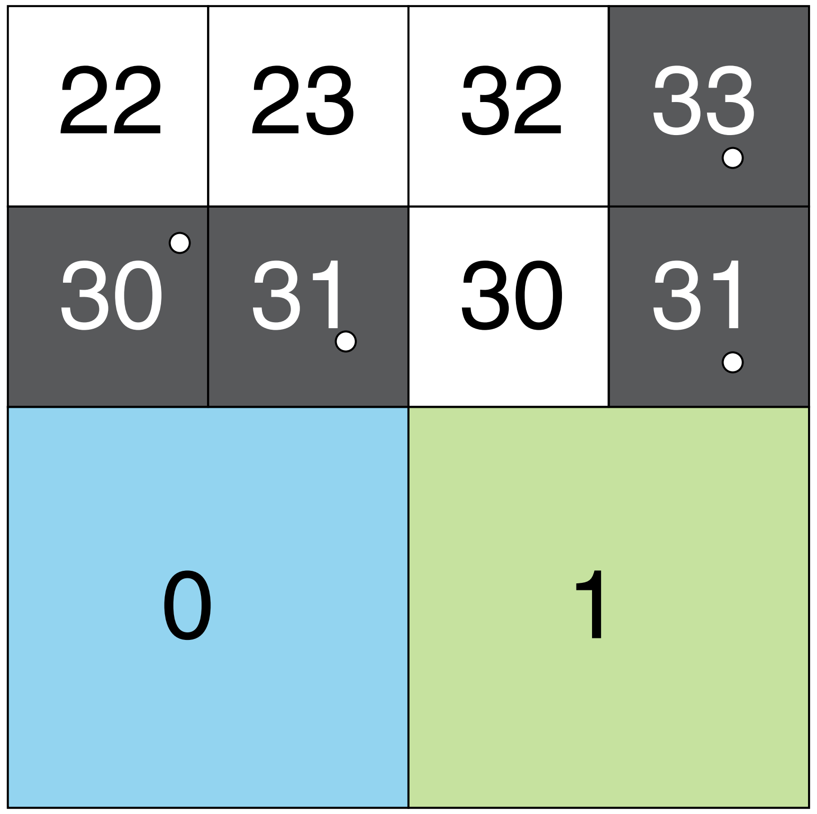
Compute distance to neighbour

Store the connections

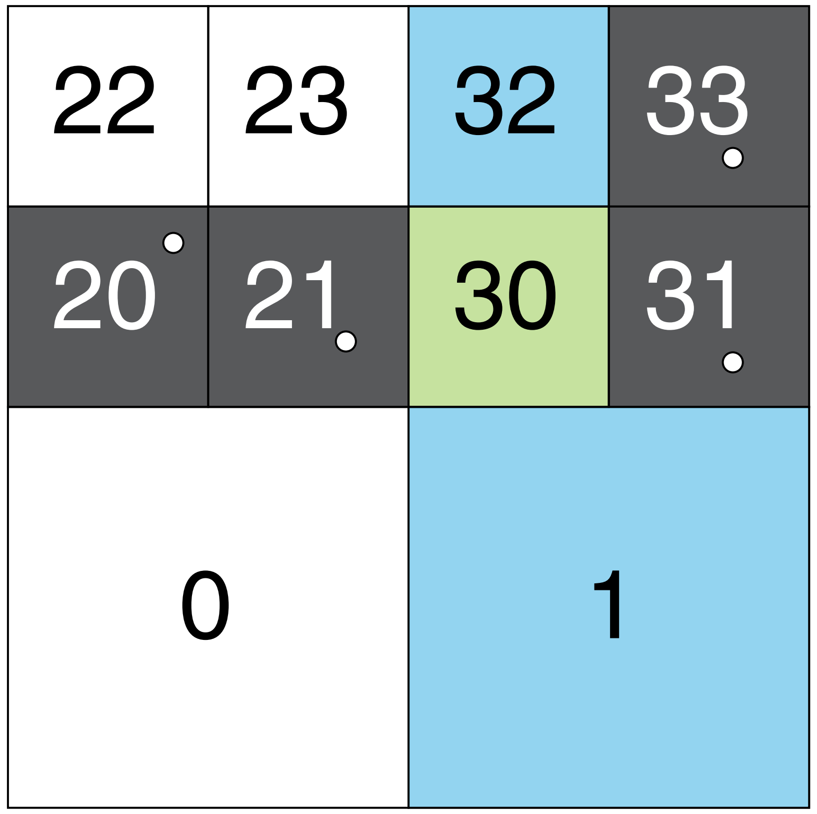
3.3. Connectivity generation

Smaller neighbours

Larger and equal neighbours are recursively computed from the largest node to the smallest nodes (Namdari et al 2015)

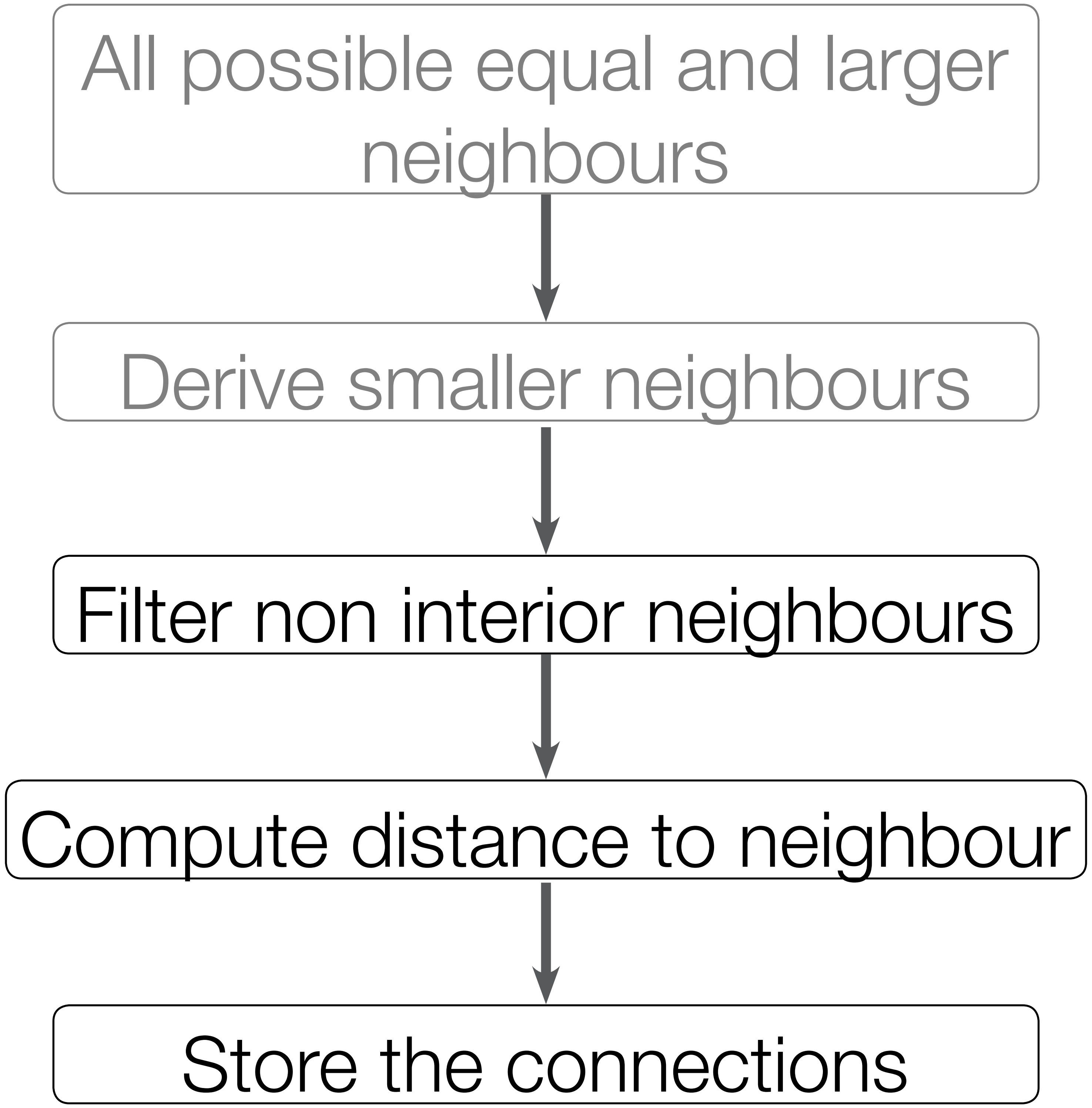


Node:	Neighbour(s):
0	1
1	0



Node:	Neighbour(s):
0	1
1	0, 30
30	1,32

3.3. Connectivity generation



All possible equal and larger neighbours

Derive smaller neighbours

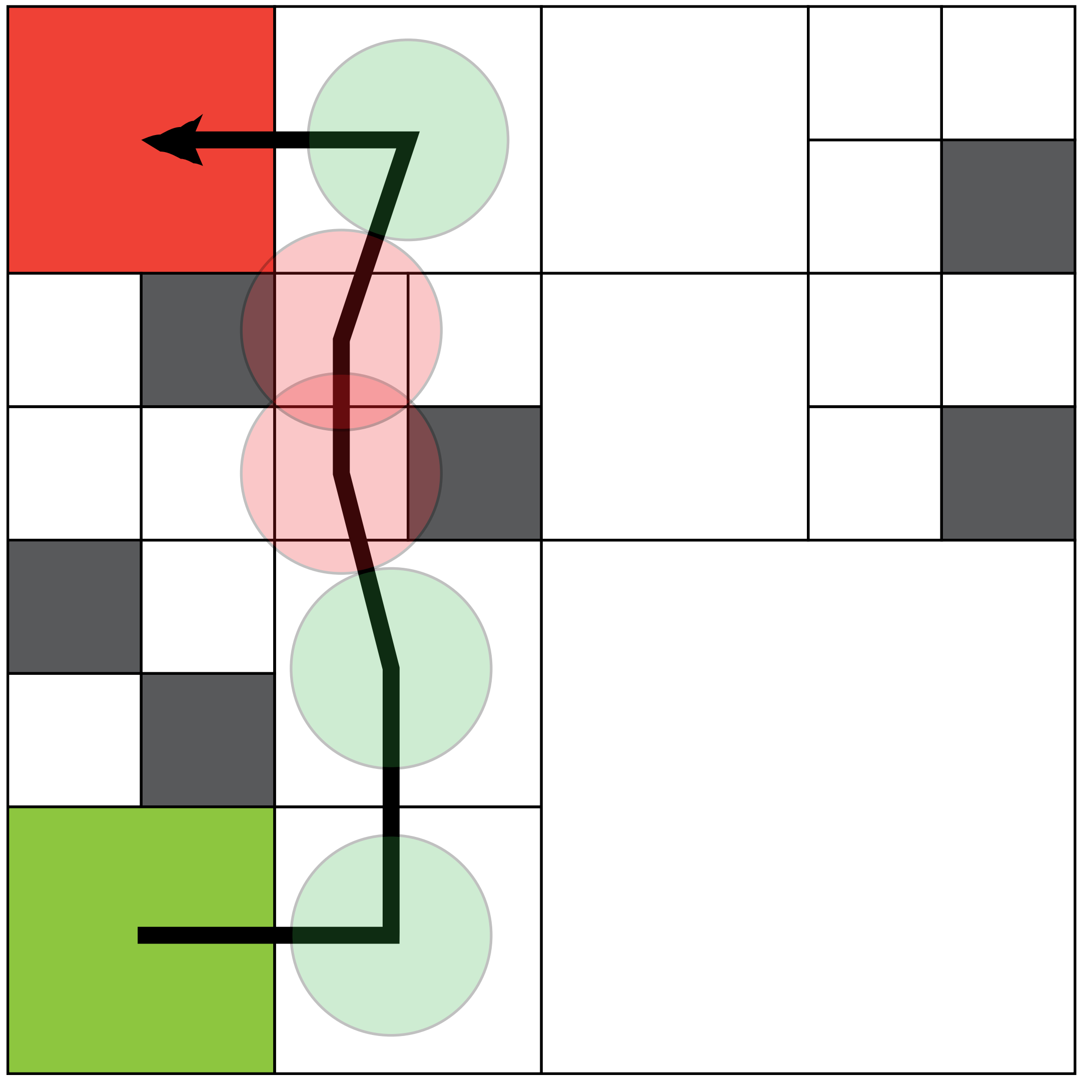
Filter non interior neighbours

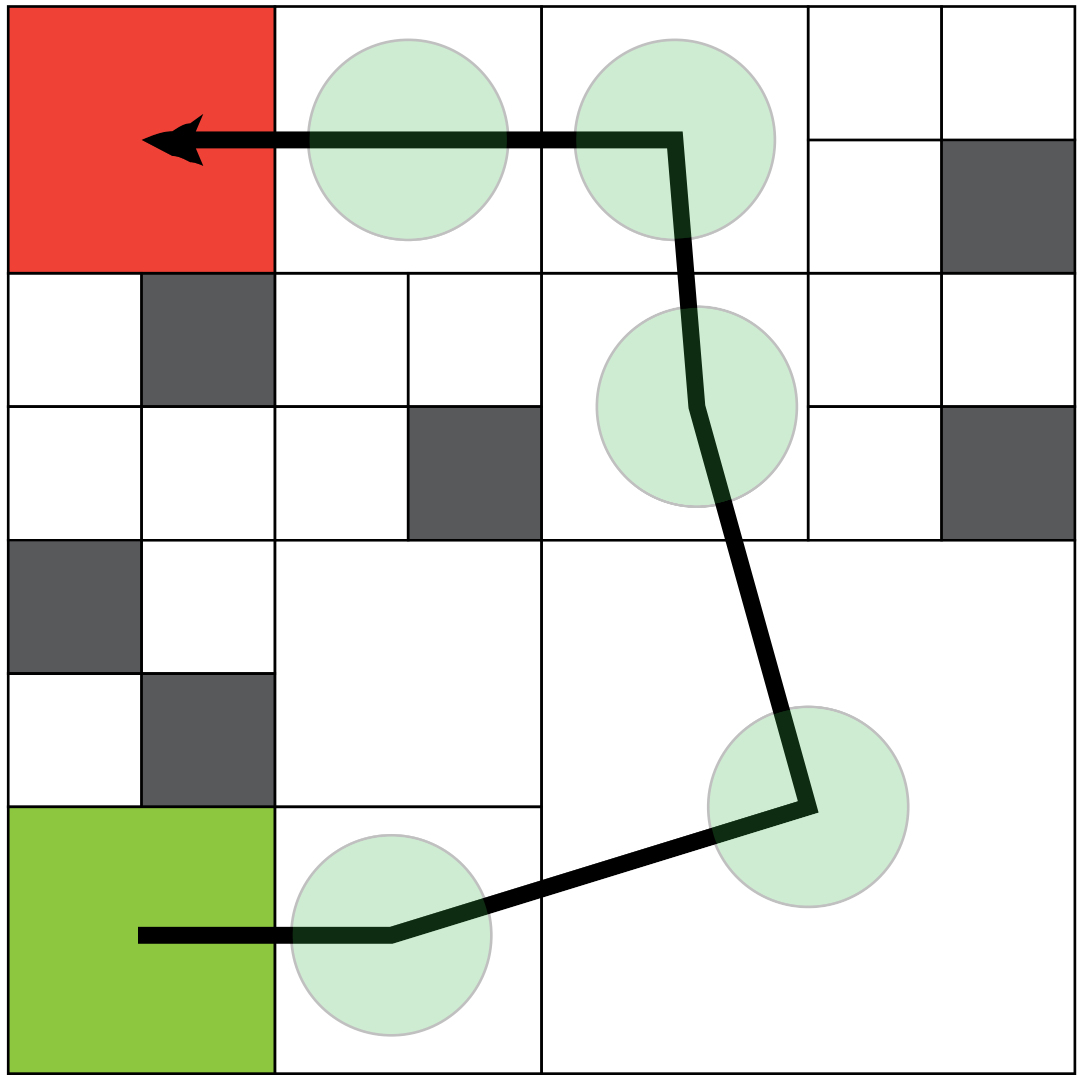
Compute distance to neighbour

Store the connections

3.4. Collision avoidance

A path is collision free if an object cannot intersect with any non interior empty node along the path.

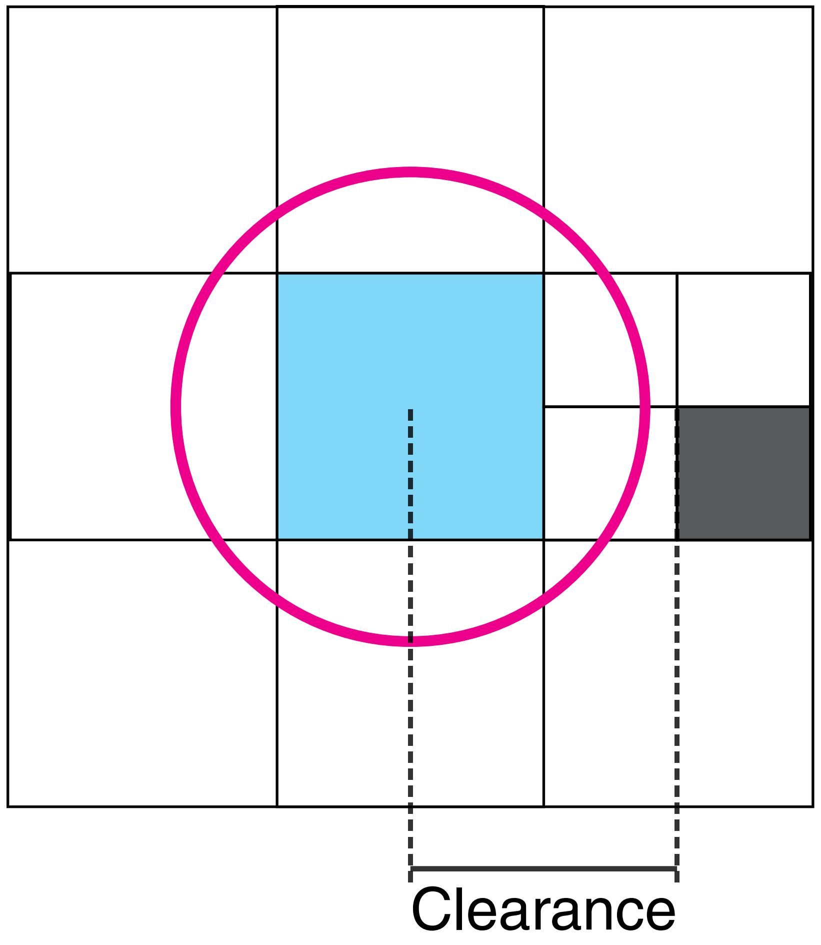




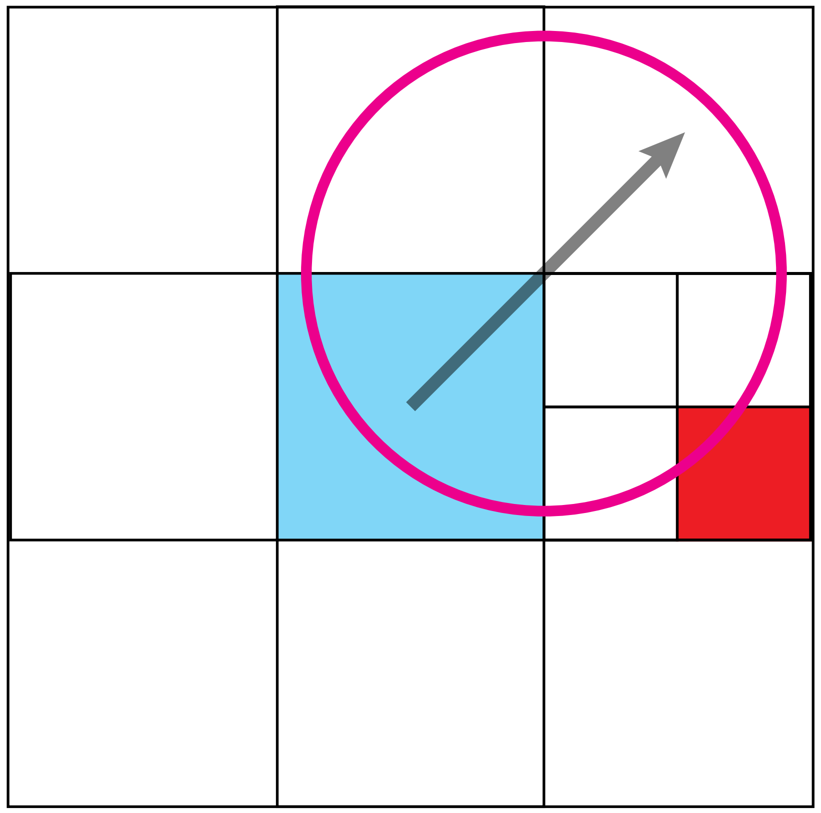
3.4. Collision avoidance

1. No intersection in centre points of empty nodes
2. No intersection in crossing point between two empty nodes

Clearance



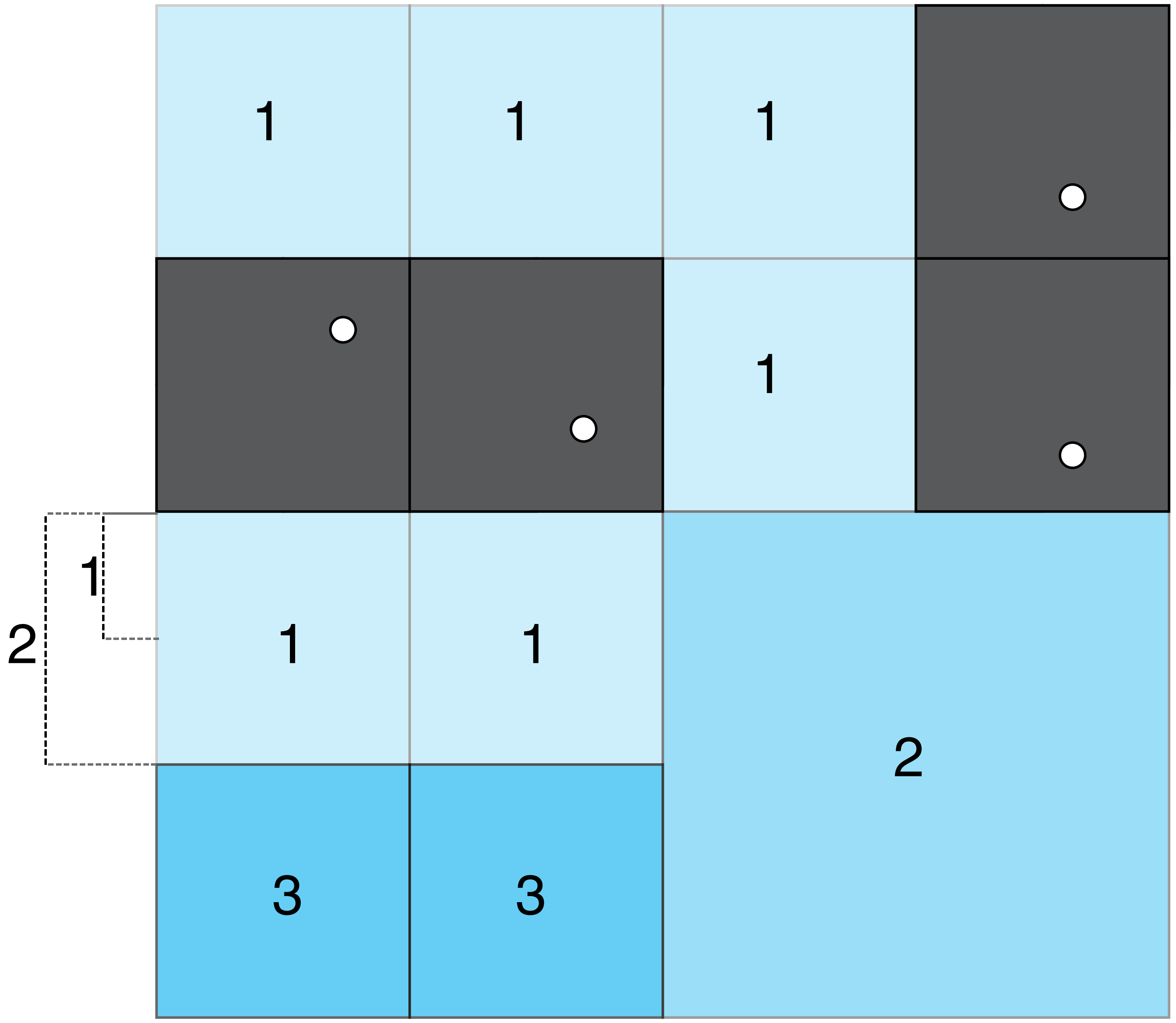
Maximal crossing value



3.4. Collision avoidance

Clearance map

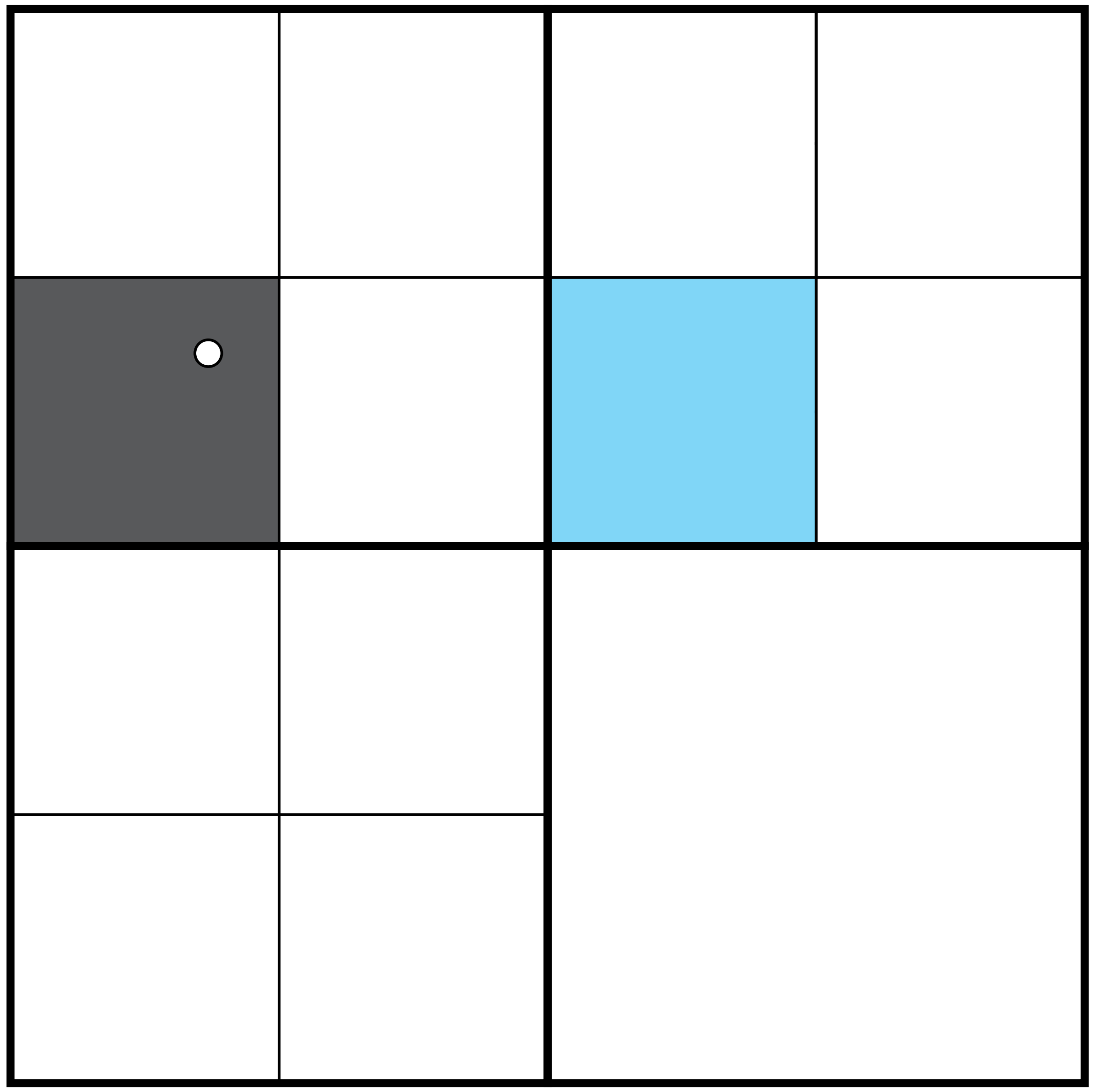
The distance between the centre of an interior empty node and the border of the closest non empty node

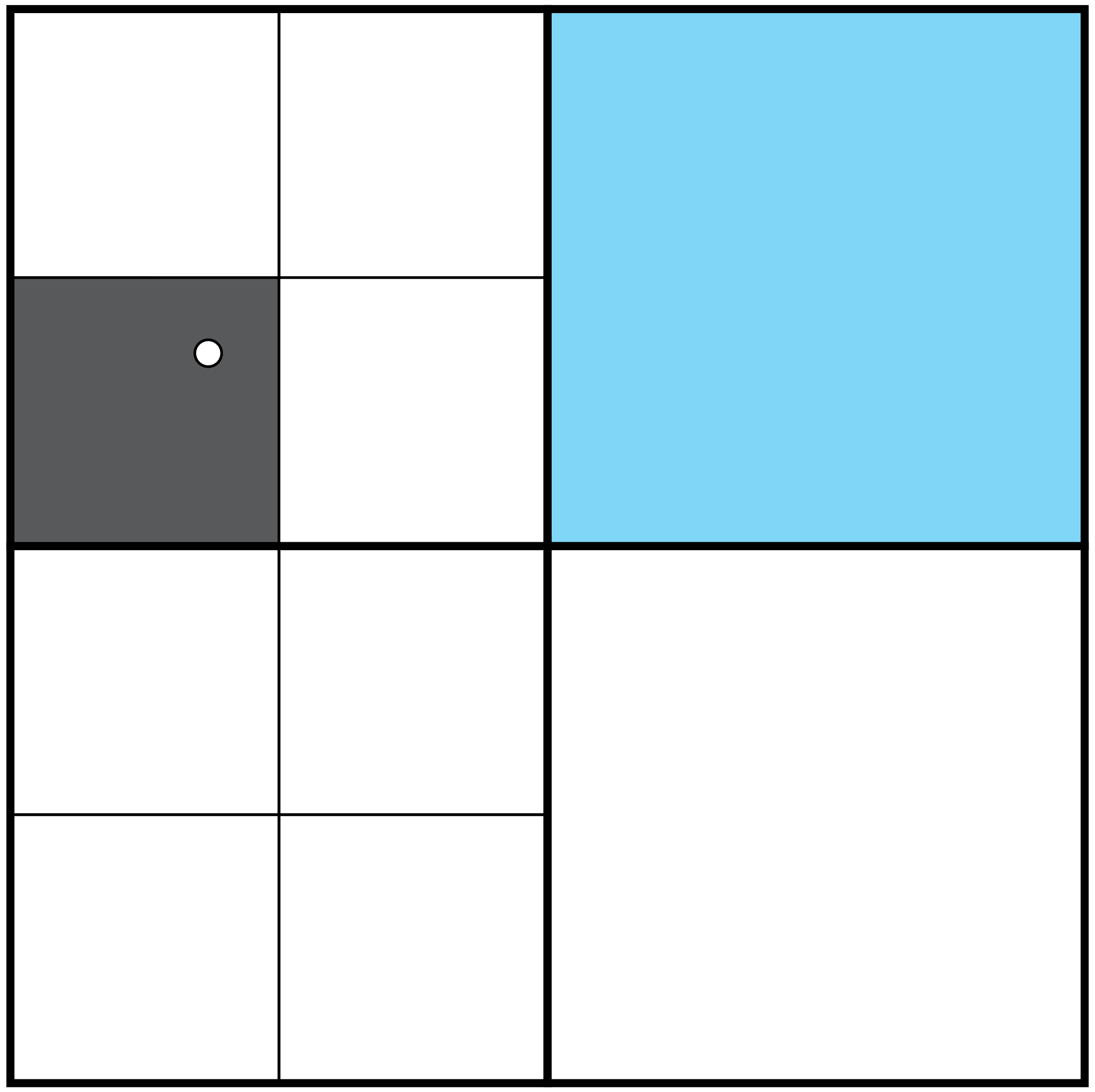


3.4. Collision avoidance

Clearance

For any white node, its equal sized neighbours cannot all be black. Otherwise merging would take place (Samet, 1982).

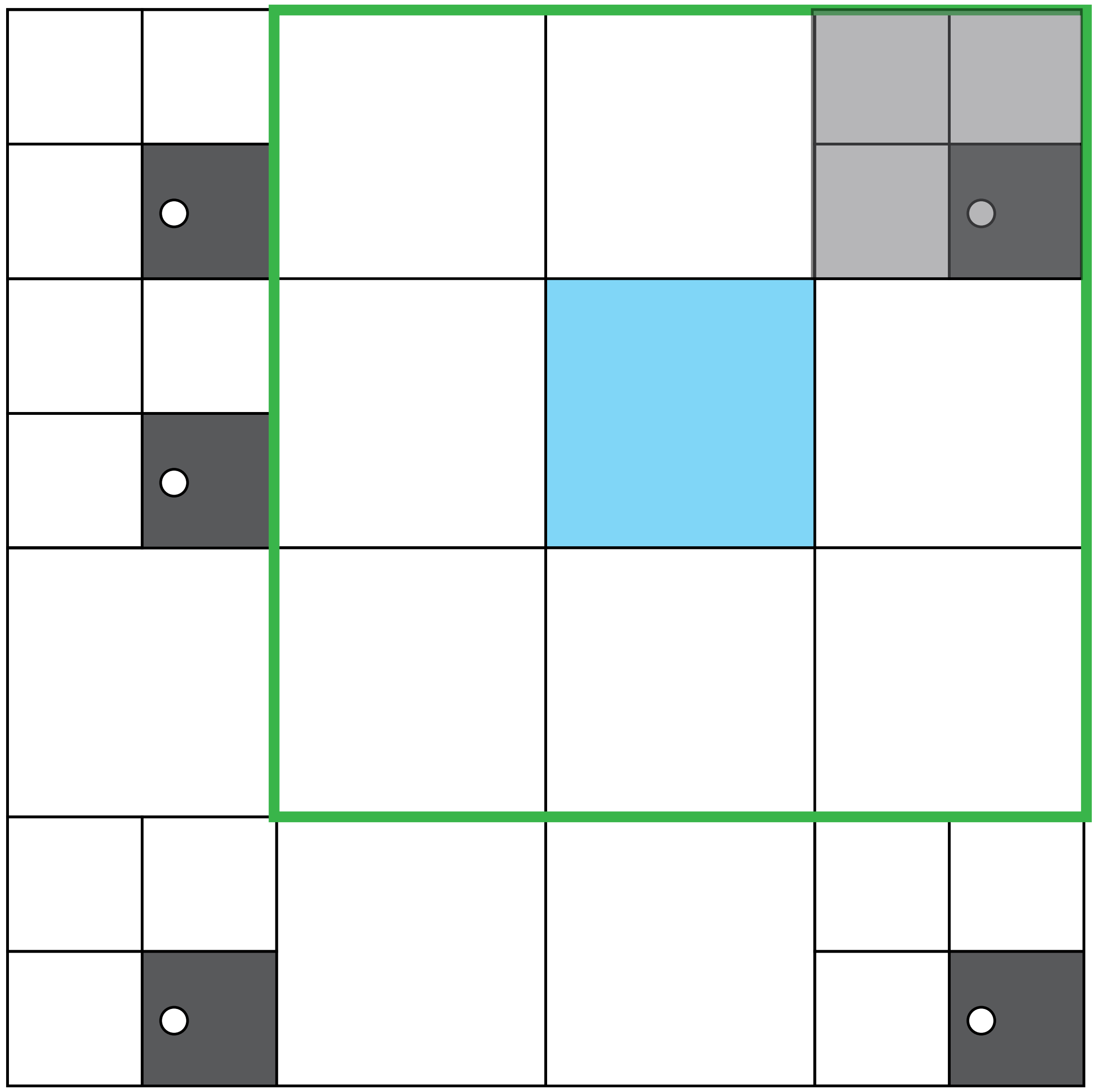




3.4. Collision avoidance

Clearance

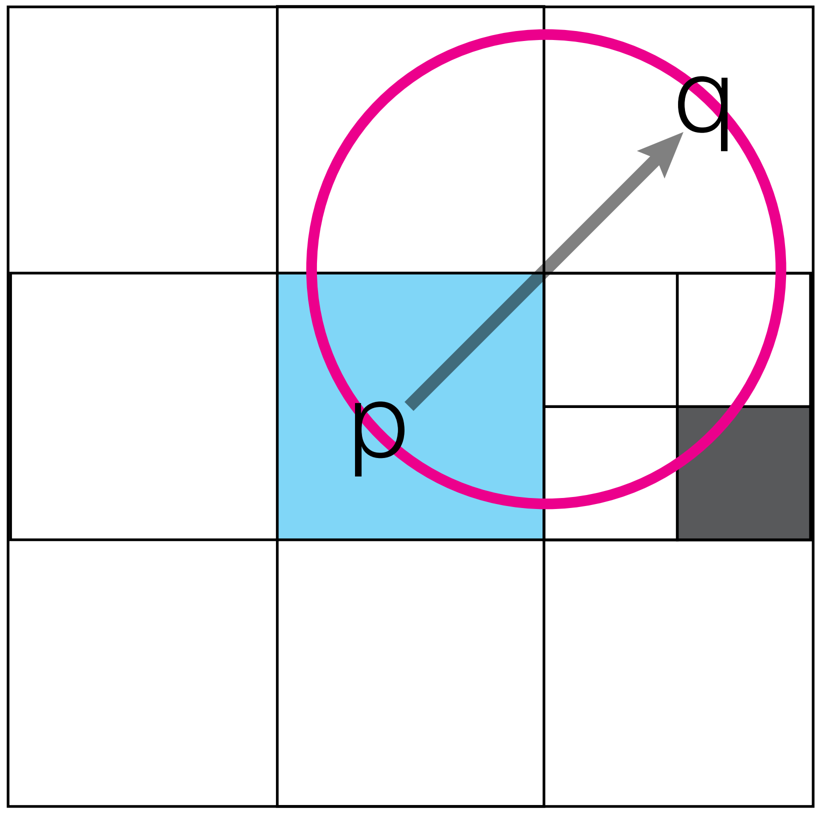
Leaf children of 8 equal sized neighbours need to be checked in an quadtree (26 in octree).

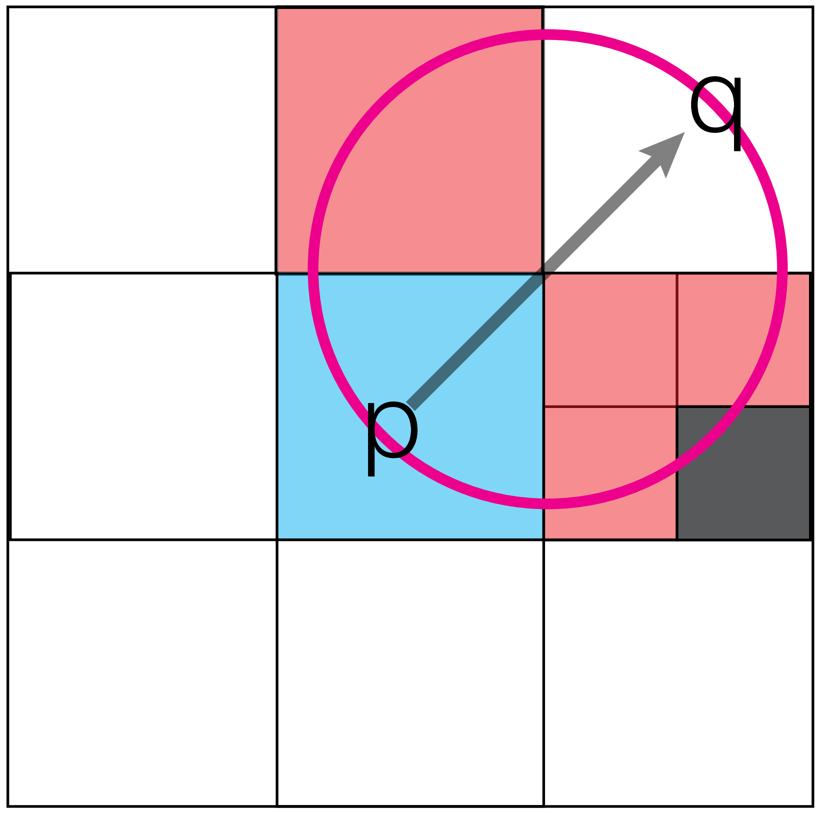


3.4. Collision avoidance

Maximal crossing value

Compute the minimal distance to a non empty node for each connection between two interior empty nodes.

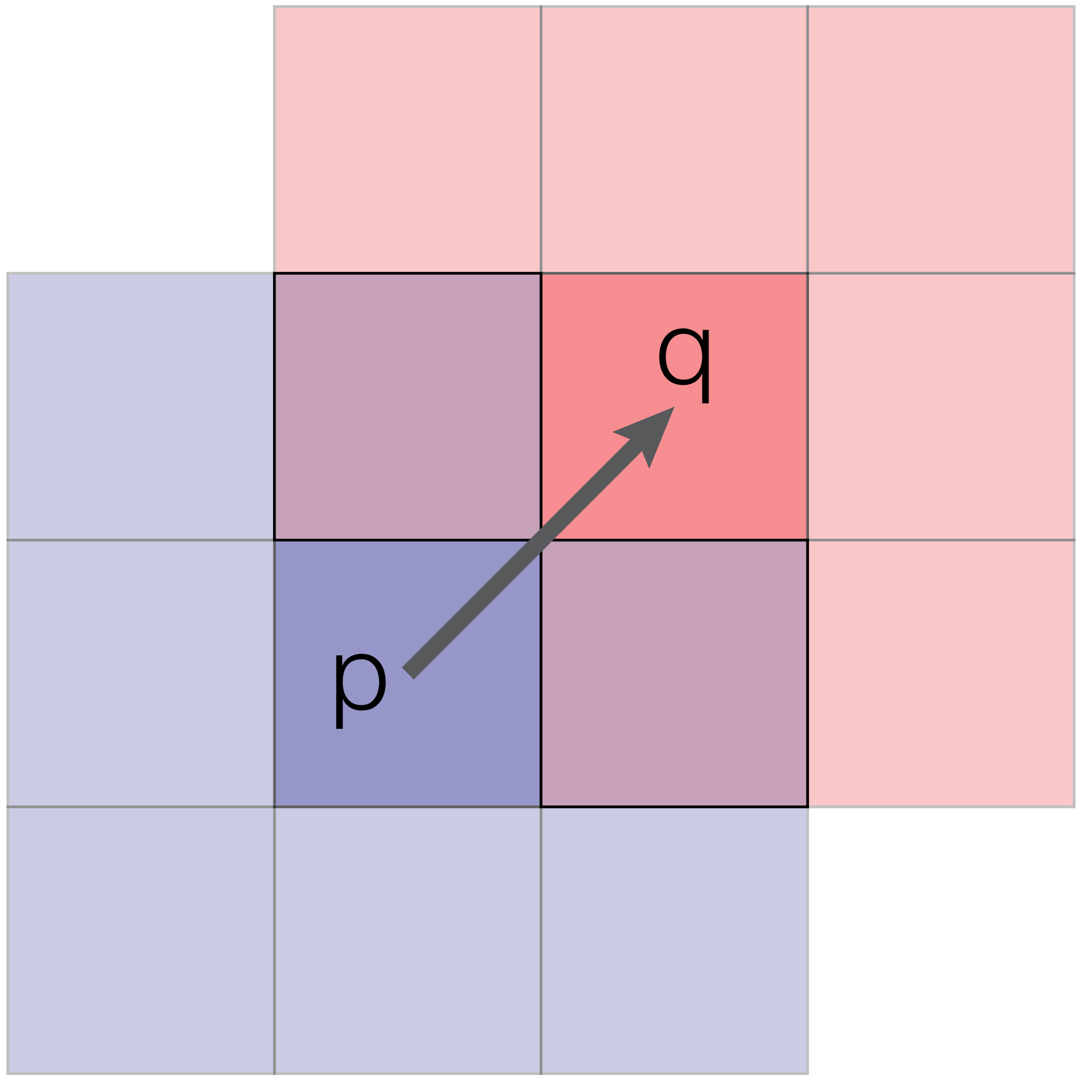




3.4. Collision avoidance

Maximal crossing value

Explore only common neighbours of node p and q for non empty nodes



3.5. Distance types

1	1	1
1	0	1
1	1	1

Chessboard

√2	1	√2
1	0	1
√2	1	√2

Euclidean

2	1	2
1	0	1
2	1	2

Manhattan

- Chessboard
  - » maximum of the x, y and z components
- Manhattan
  - » sum of the x, y and z components
- Euclidean
  - » ‘real’ distance ($a^2 + b^2 = c^2$).

3.6. Benchmark tests

Identify the effects on path length and computation time in A* path finding:

- Octree depth
  - 6-8

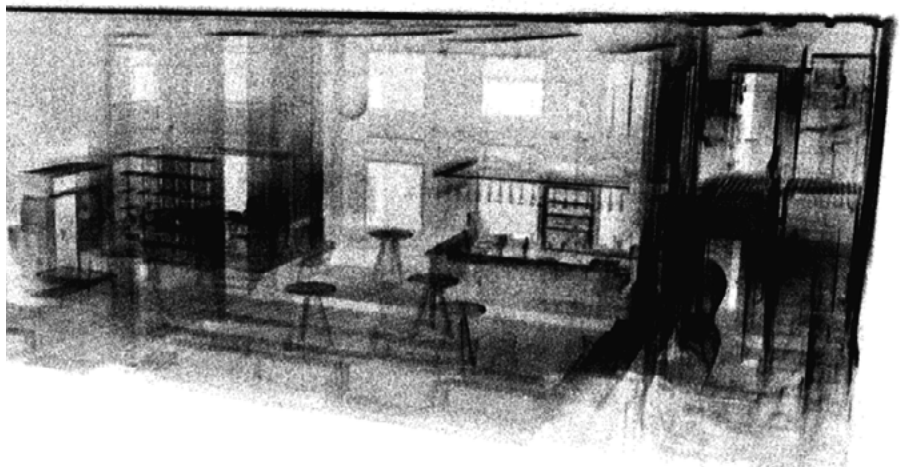
- pre-processing connectivity versus on the fly
  - the effect of pre-processed connectivity

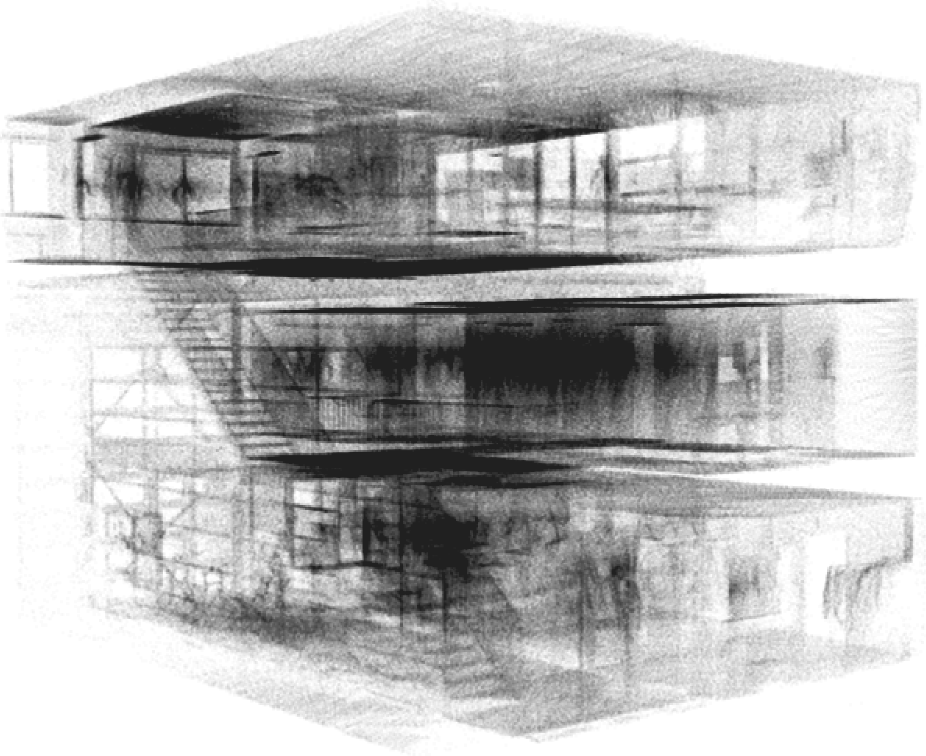
- Path connectivity
  - face, edge and vertex

- Distance type
  - Euclidean, chessboard and Manhattan

3.7. Point cloud datasets

Name	Points	Bounding box
Bouwpub	2.196.903	16m *9m *6m
Fire department	2.266.067	11m *14m *13m,
Test	3000	64*64*64



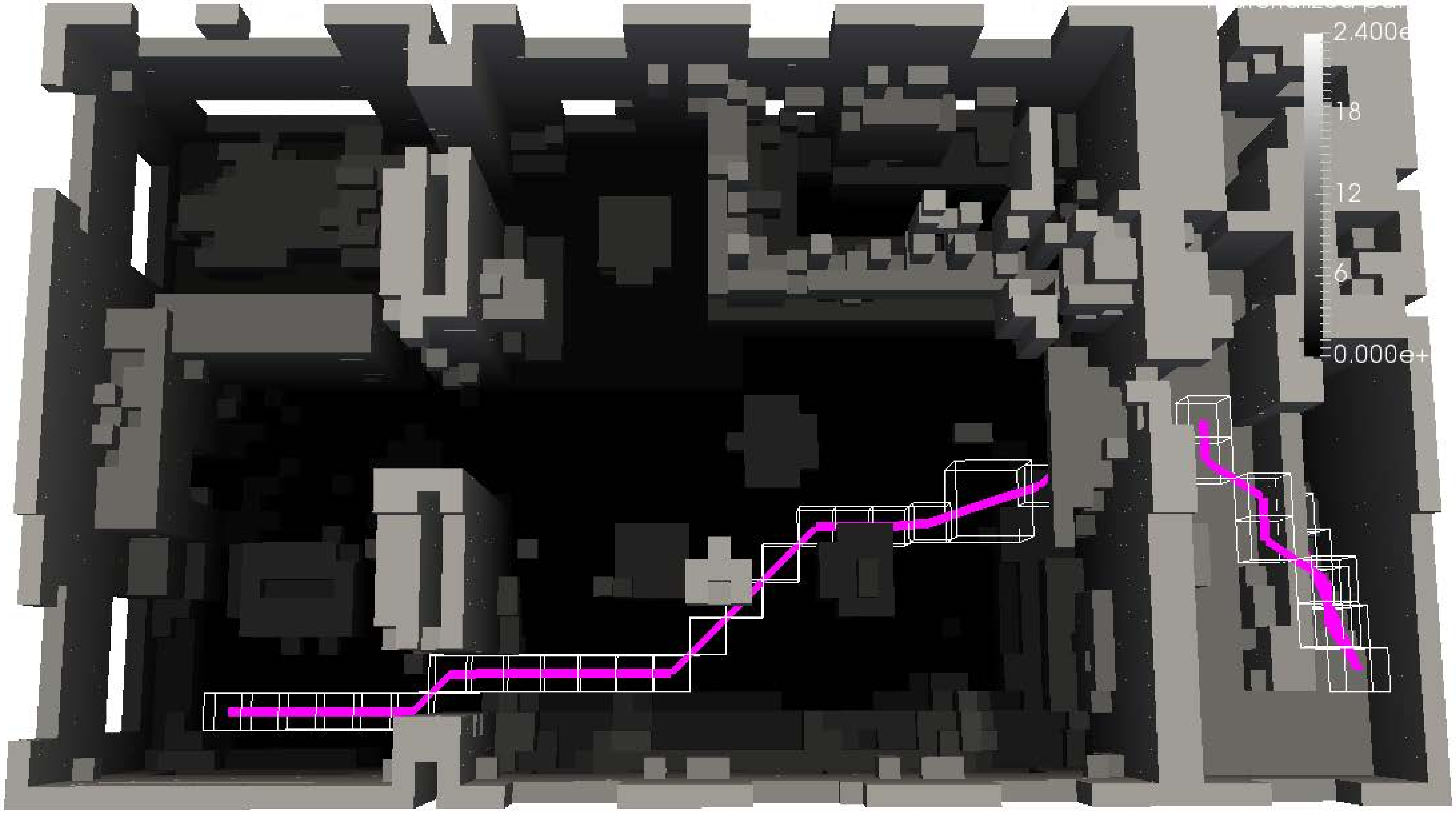


5. Results

5.1. A* path finding
5.2. Interior empty nodes
5.3. Downward distance
5.4. Connectivity generation
5.5. Collision avoidance
5.6. Benchmark results

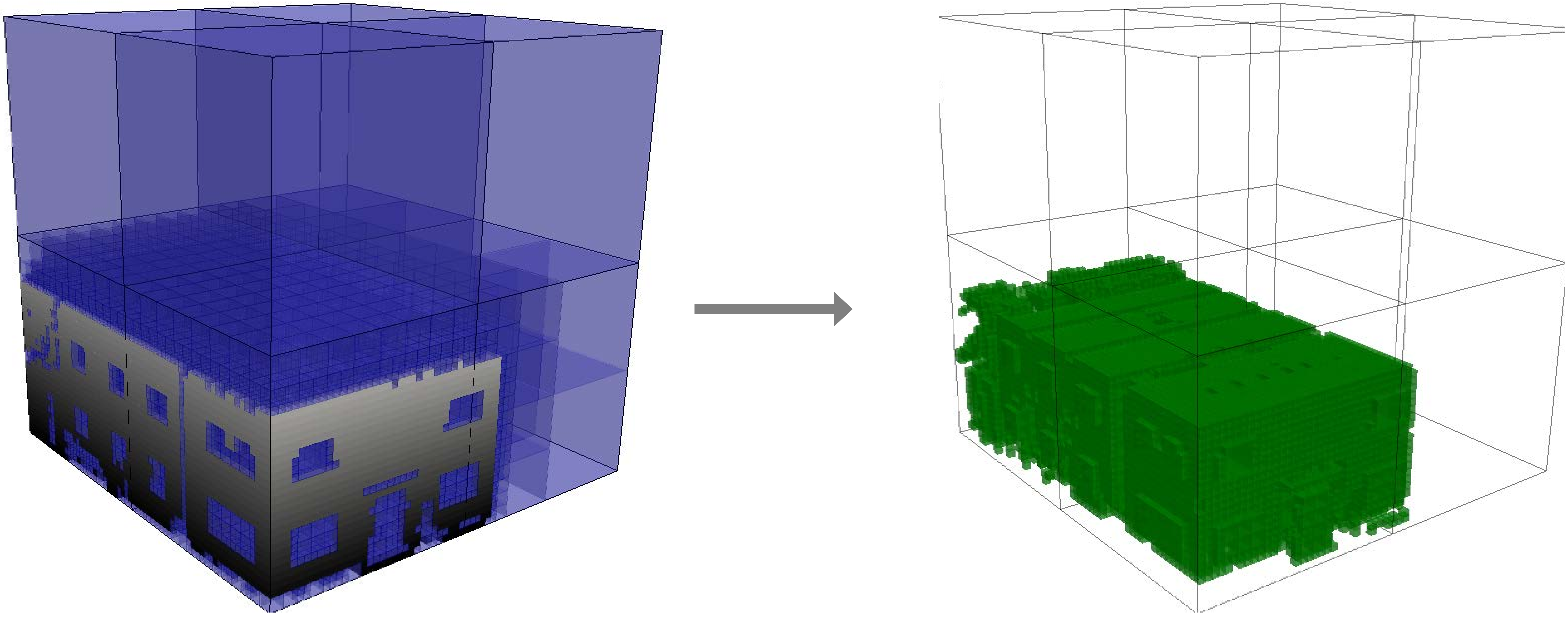
5.1. Path finding

Path goes through centre points of interior empty nodes



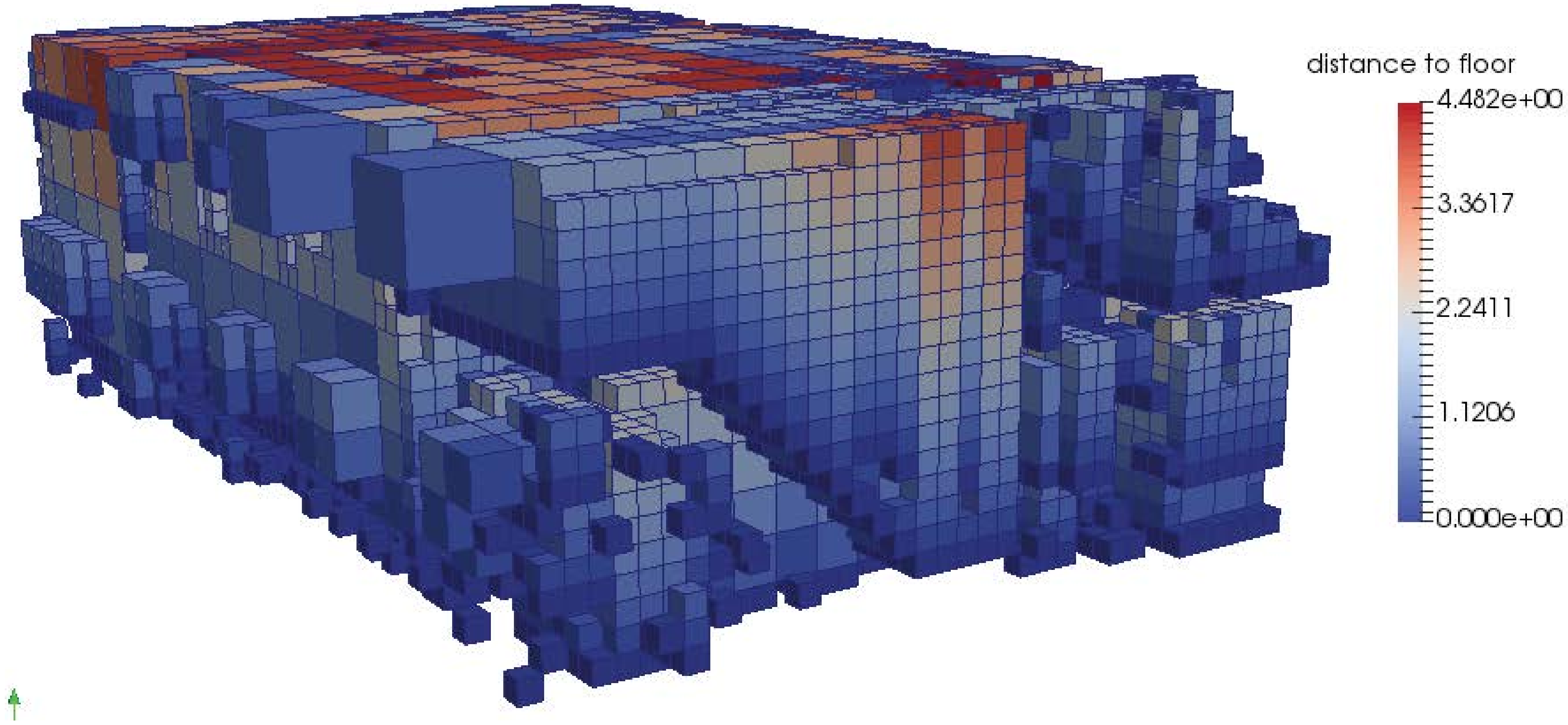
5.2. Interior empty space

1/3 of the empty nodes are filtered



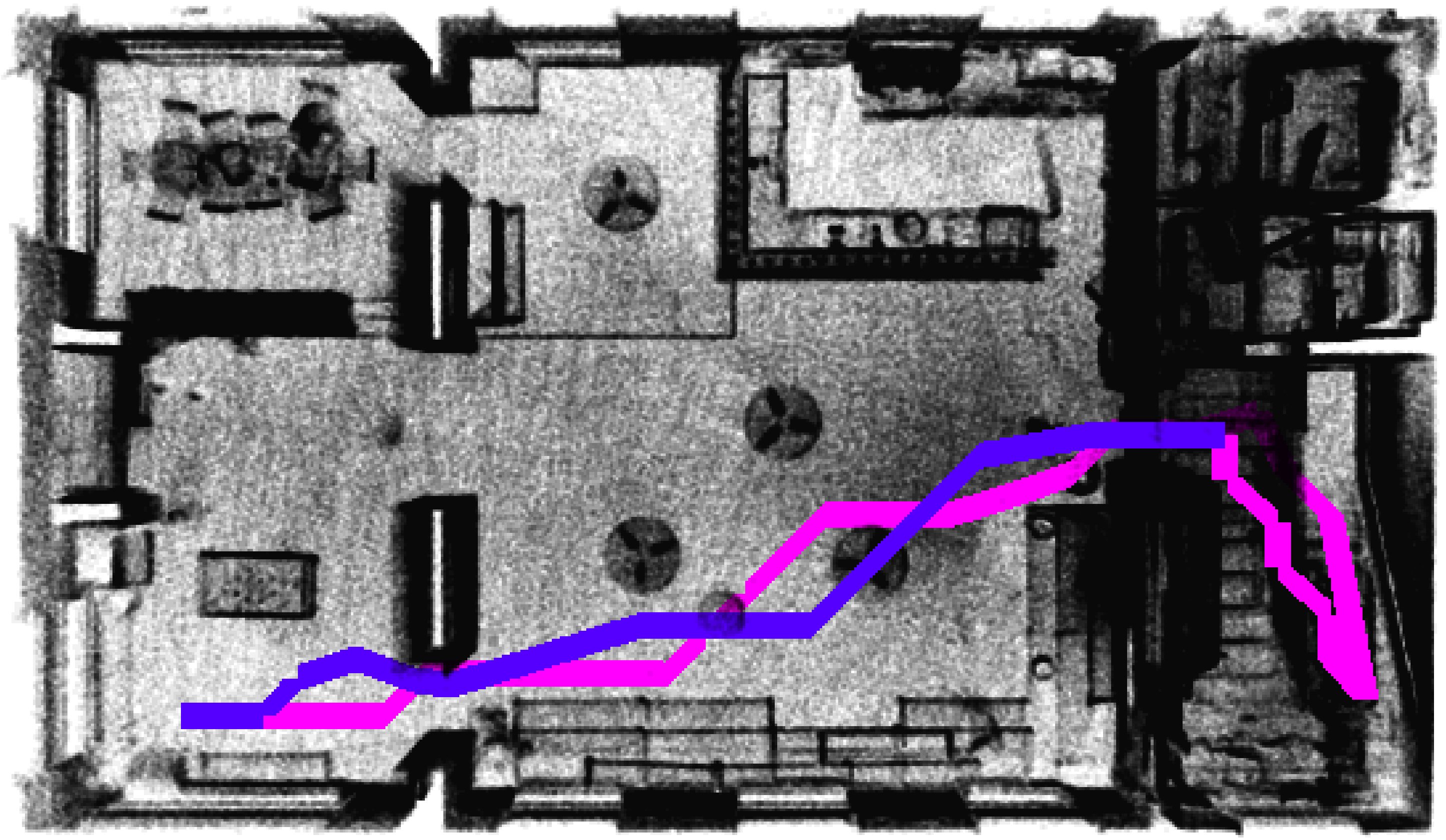
5.3. Downward distance

to non empty node

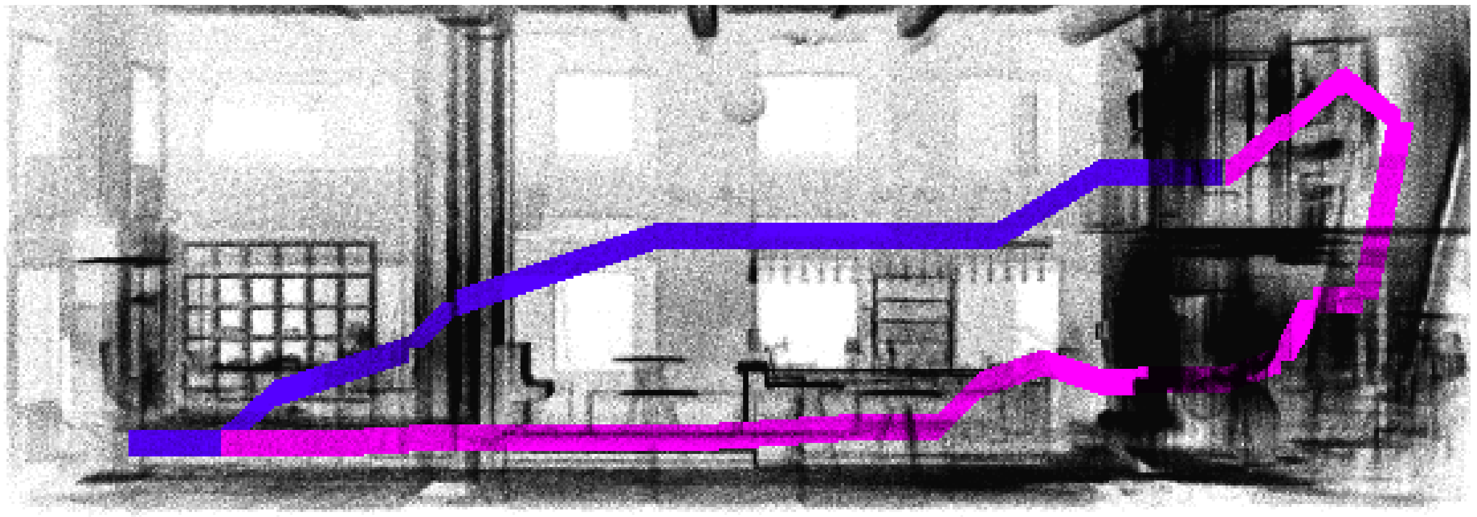


5.3. Downward distance
for path finding

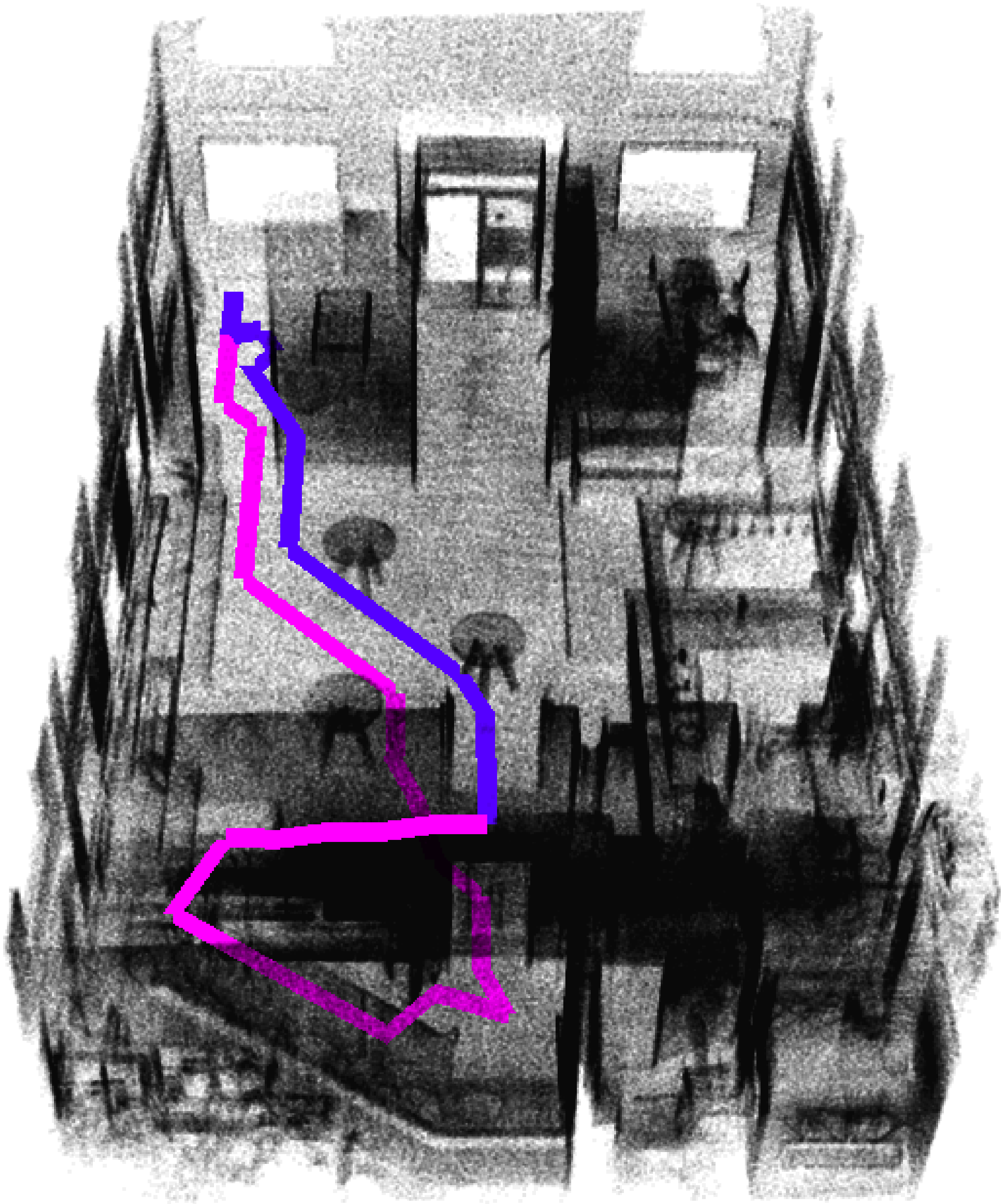
Top view



Side view

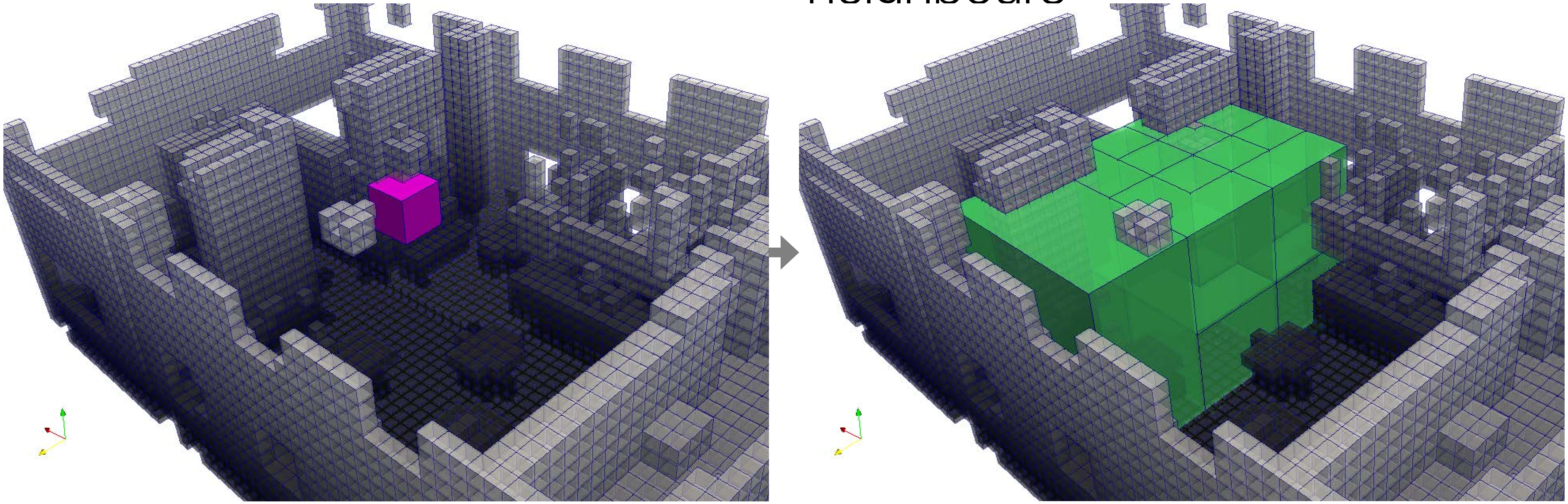


3D view



5.4. Connectivity generation

Neighbour finding

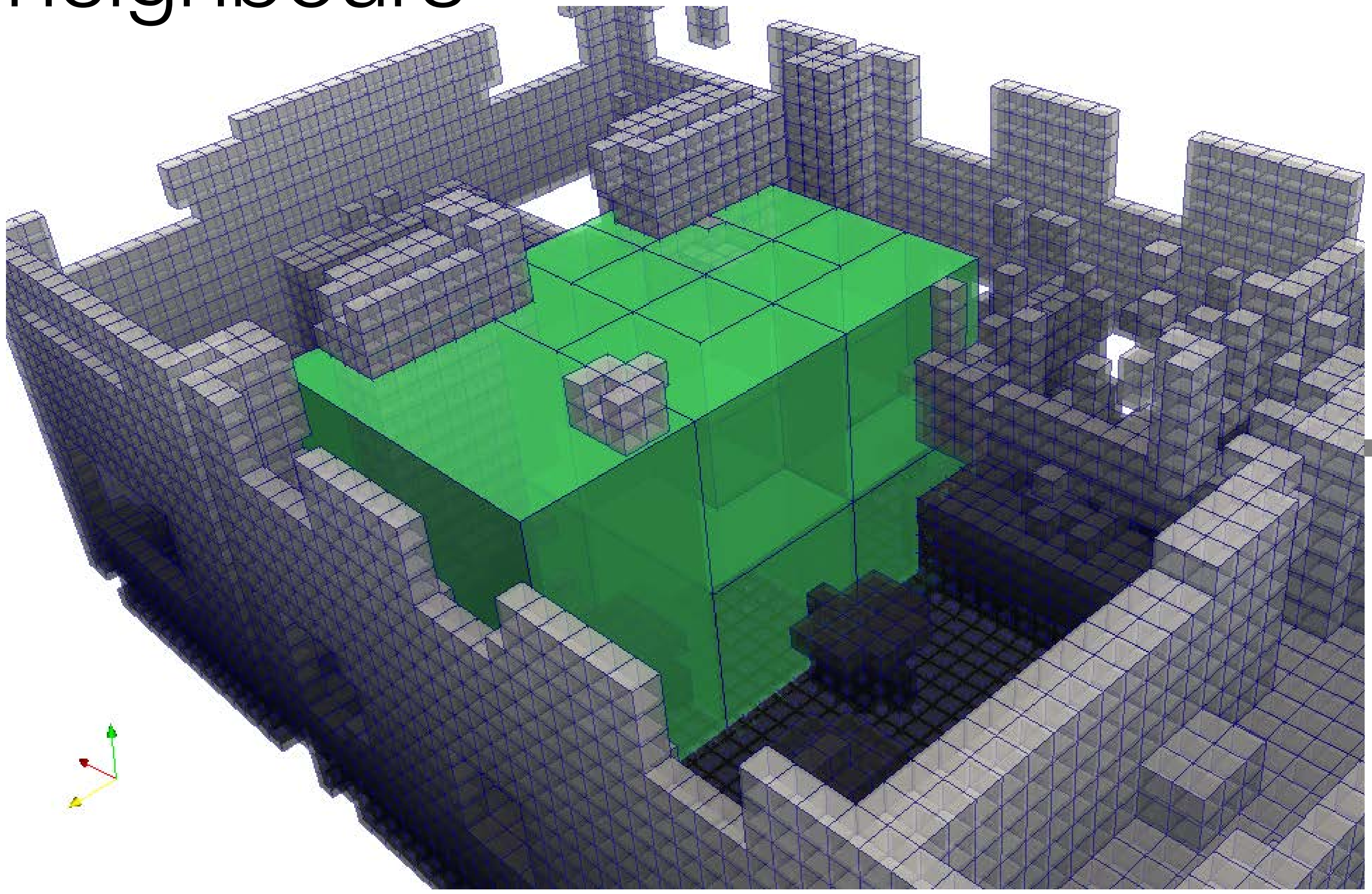


Current node

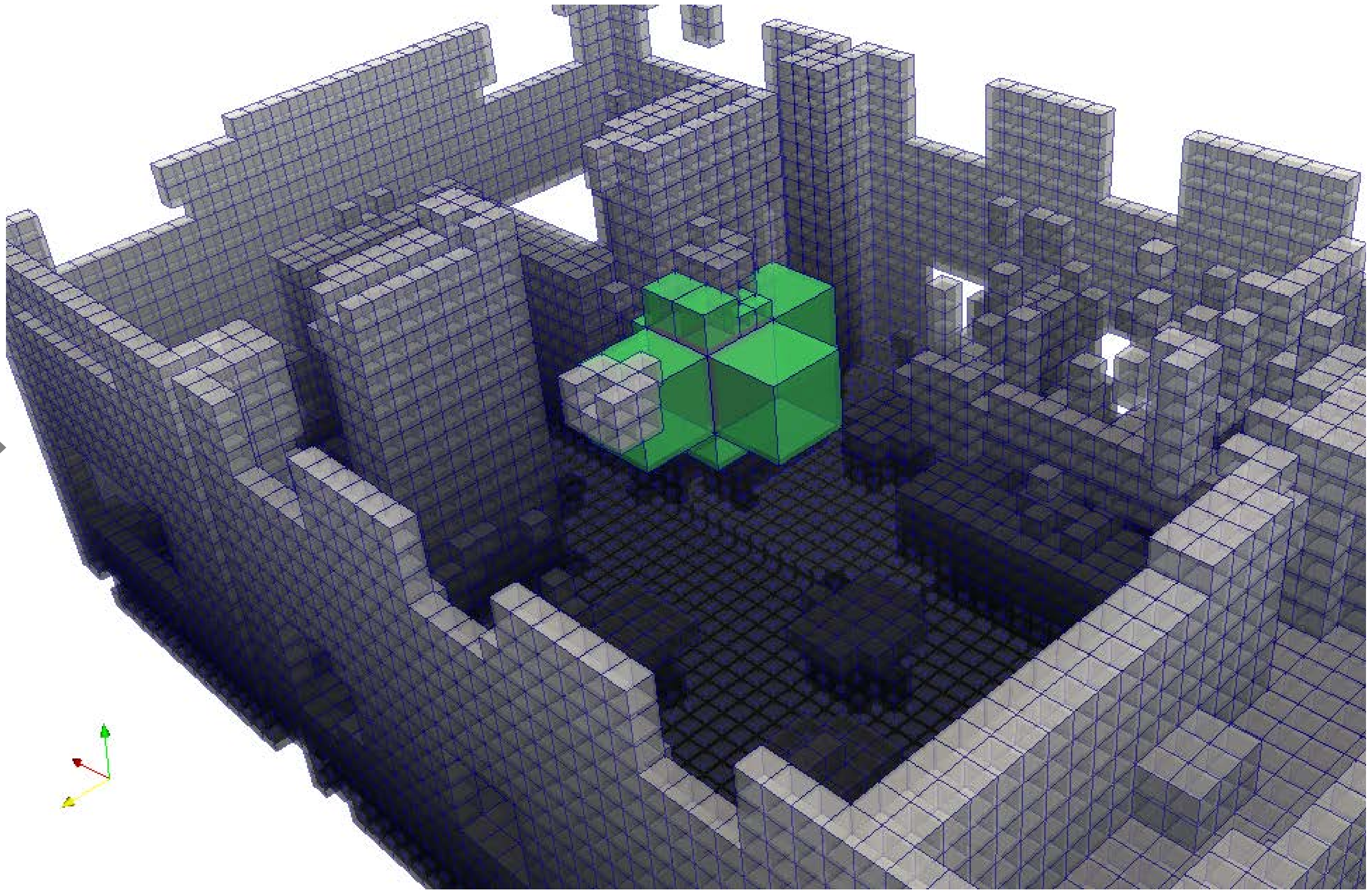
All possible equal and larger neighbours

5.4. Connectivity generation

All possible equal, larger neighbours

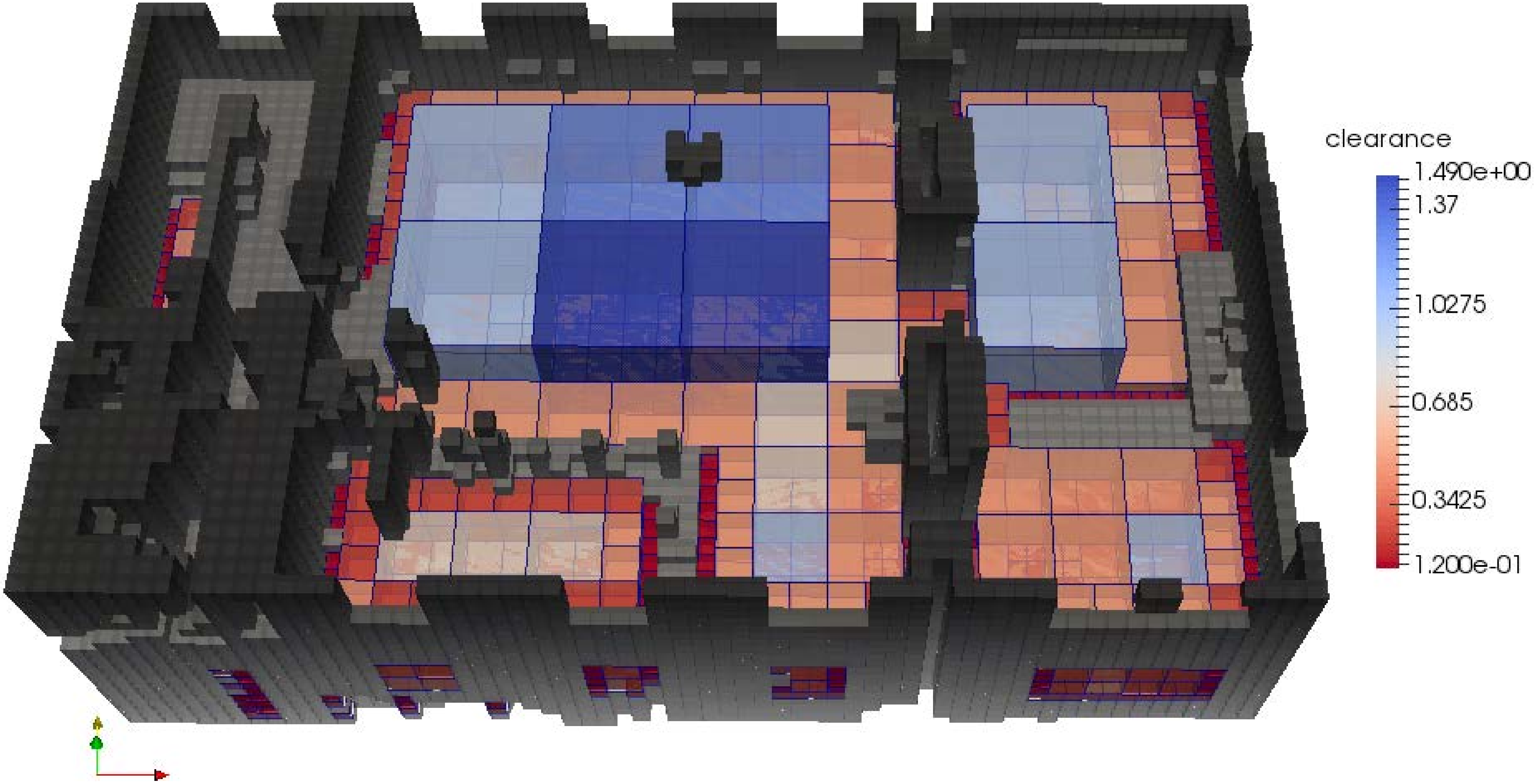


All interior empty neighbours



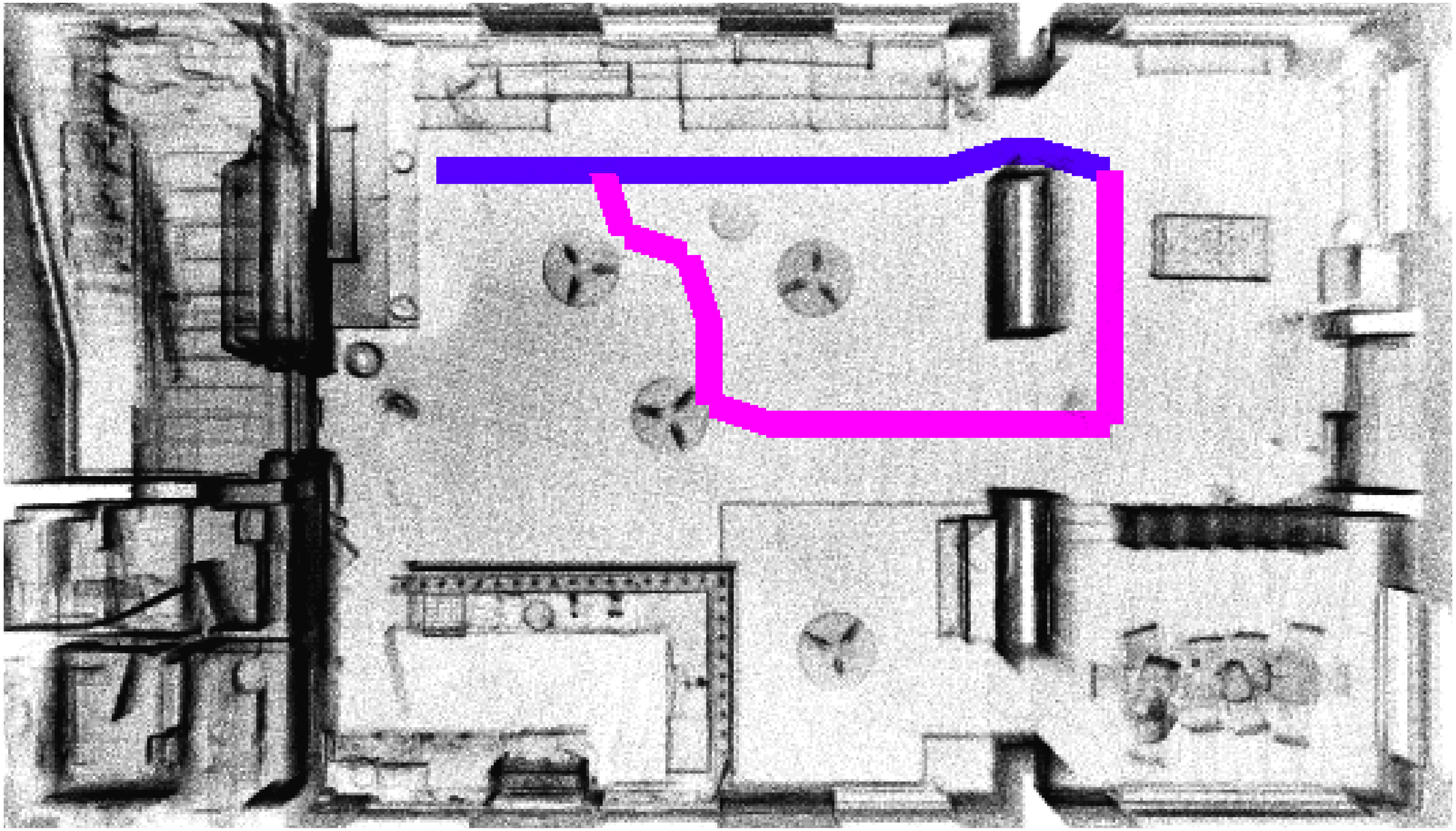
5.5. Collision avoidance

Clearance map



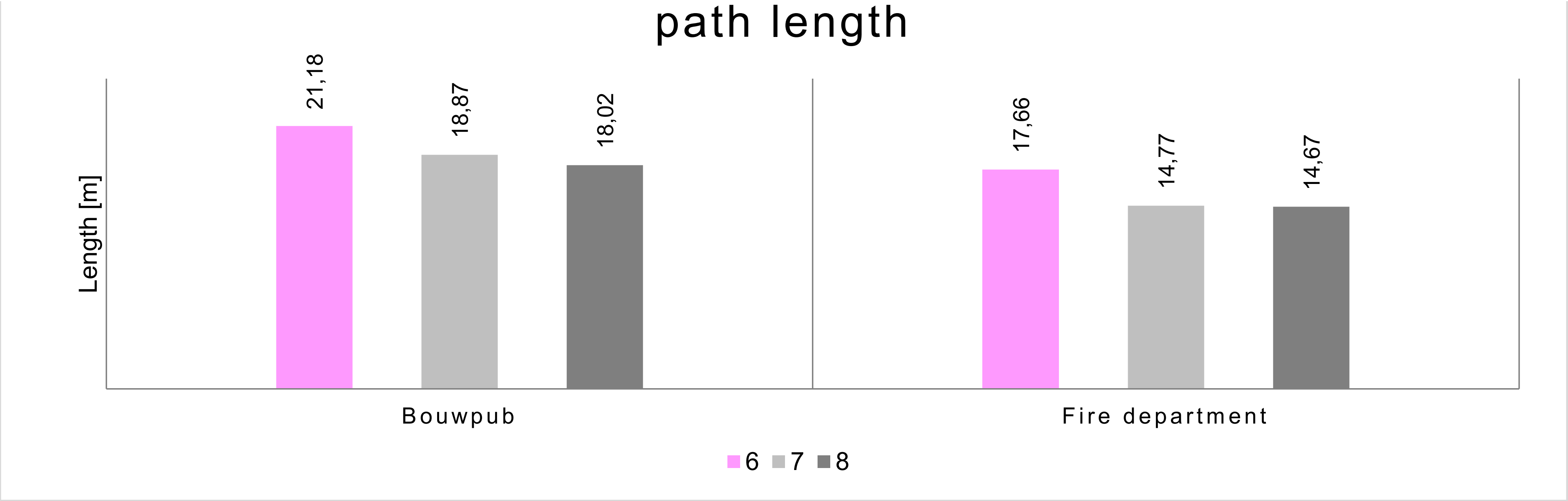
5.5. Collision avoidance

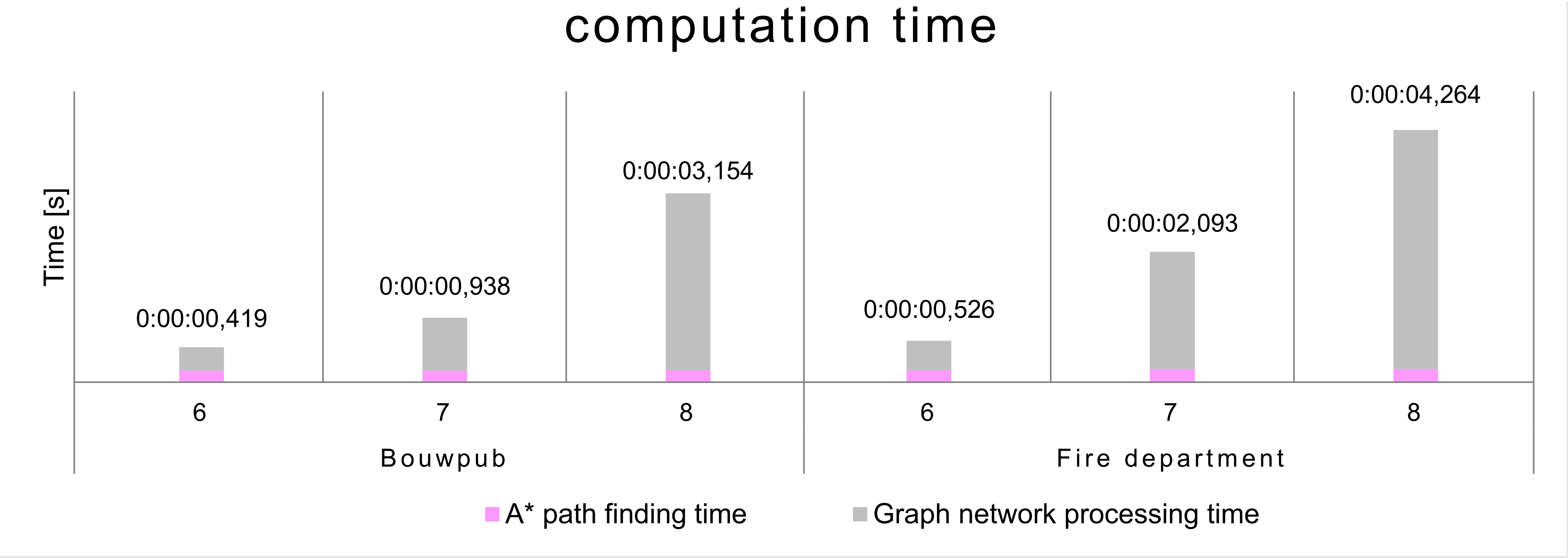
Clearance map for path finding



5.6. Benchmark results

Octree depth

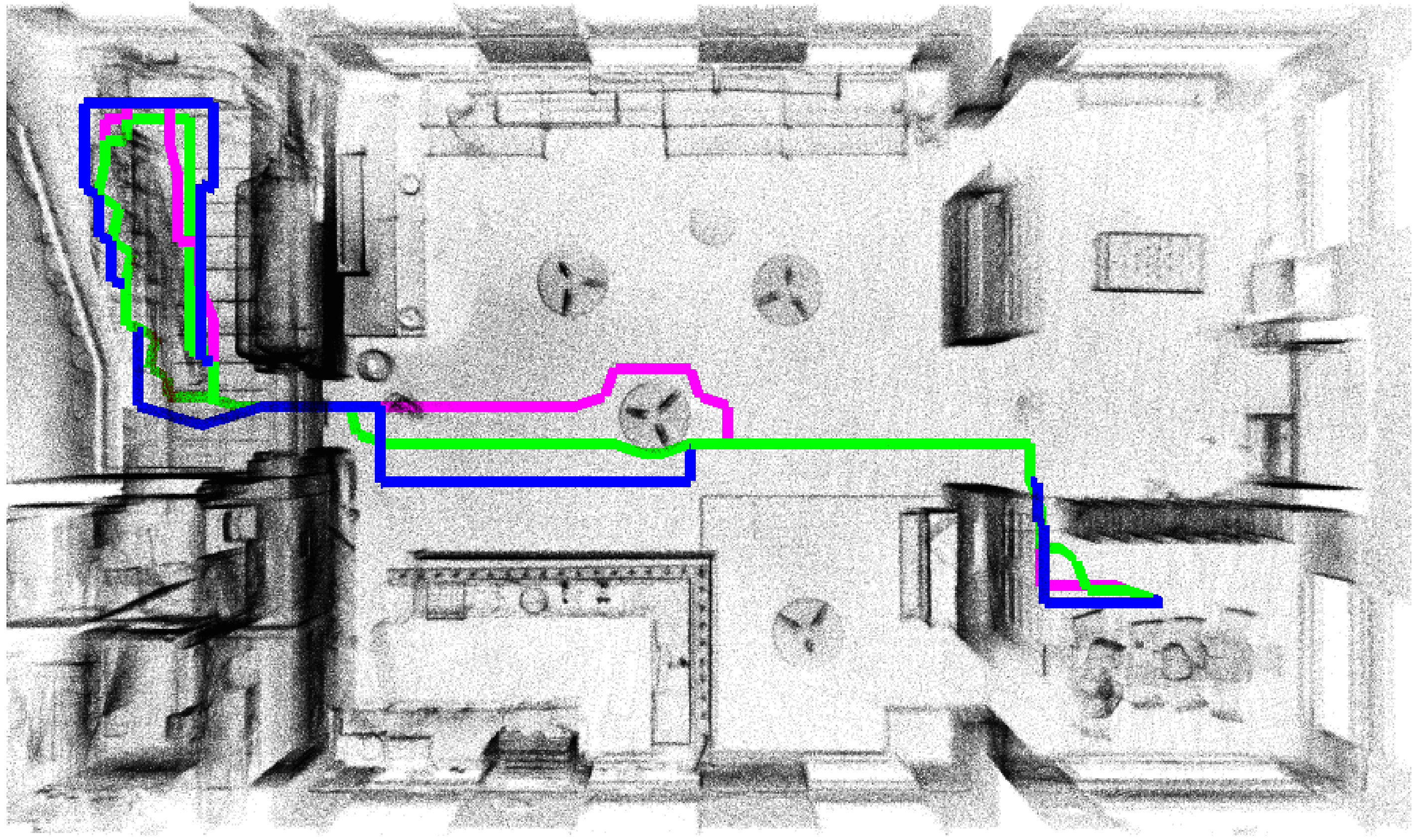




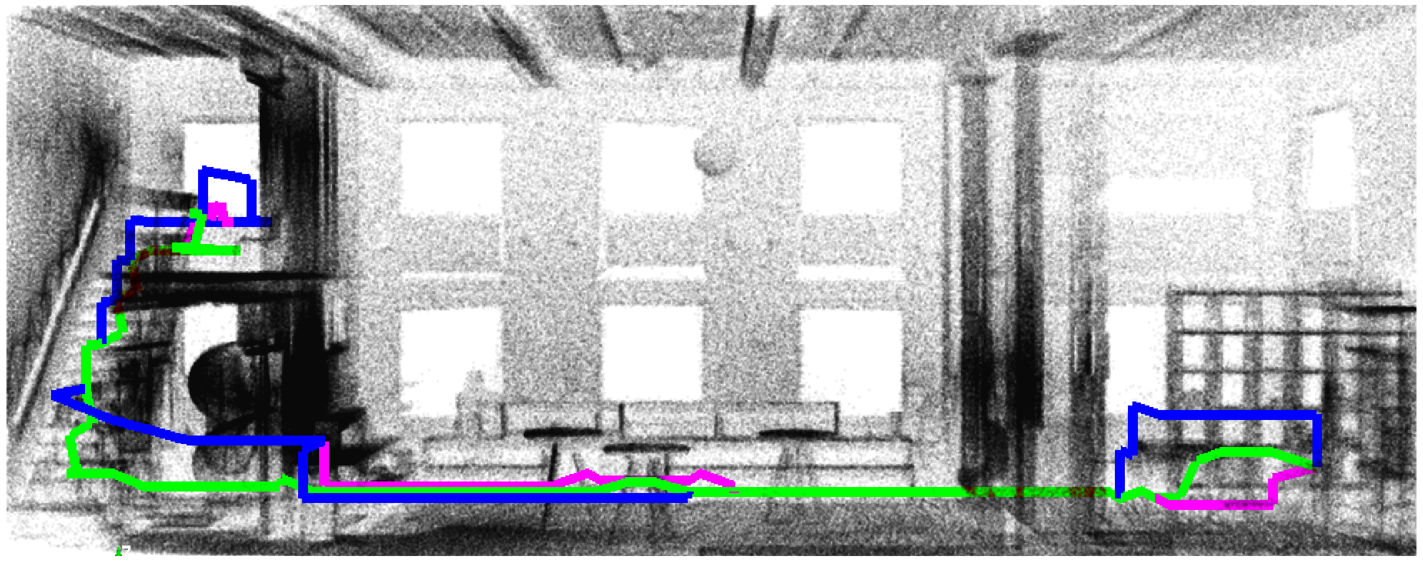
5.6. Benchmark results

Octree depth

Top view



Side view



1. Path length
   - 1.1. Spatial resolution between 0,43 m and 0,15 m.
2. Path finding computation time
   - 2.1. Increases with octree dept
   - 2.2. Mainly due to size of network graph

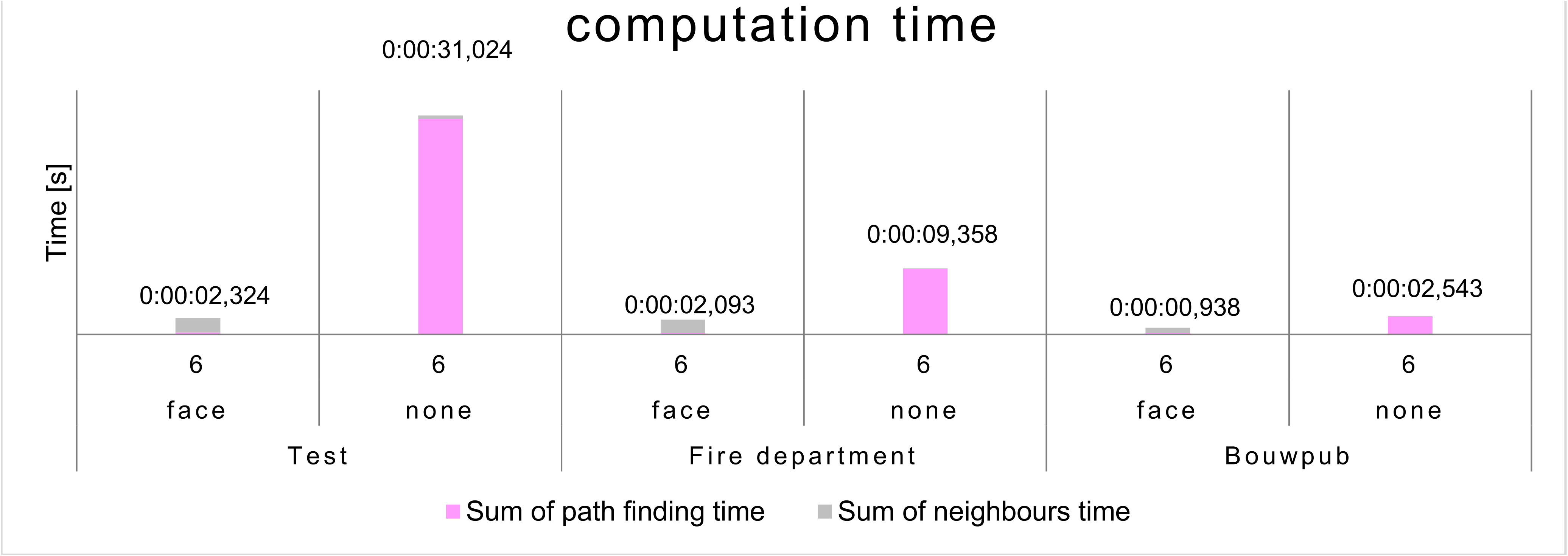
🟦 6 levels

🟪 7 levels

🟩 8 levels

5.6. Benchmark results

Pre-processing connectivity



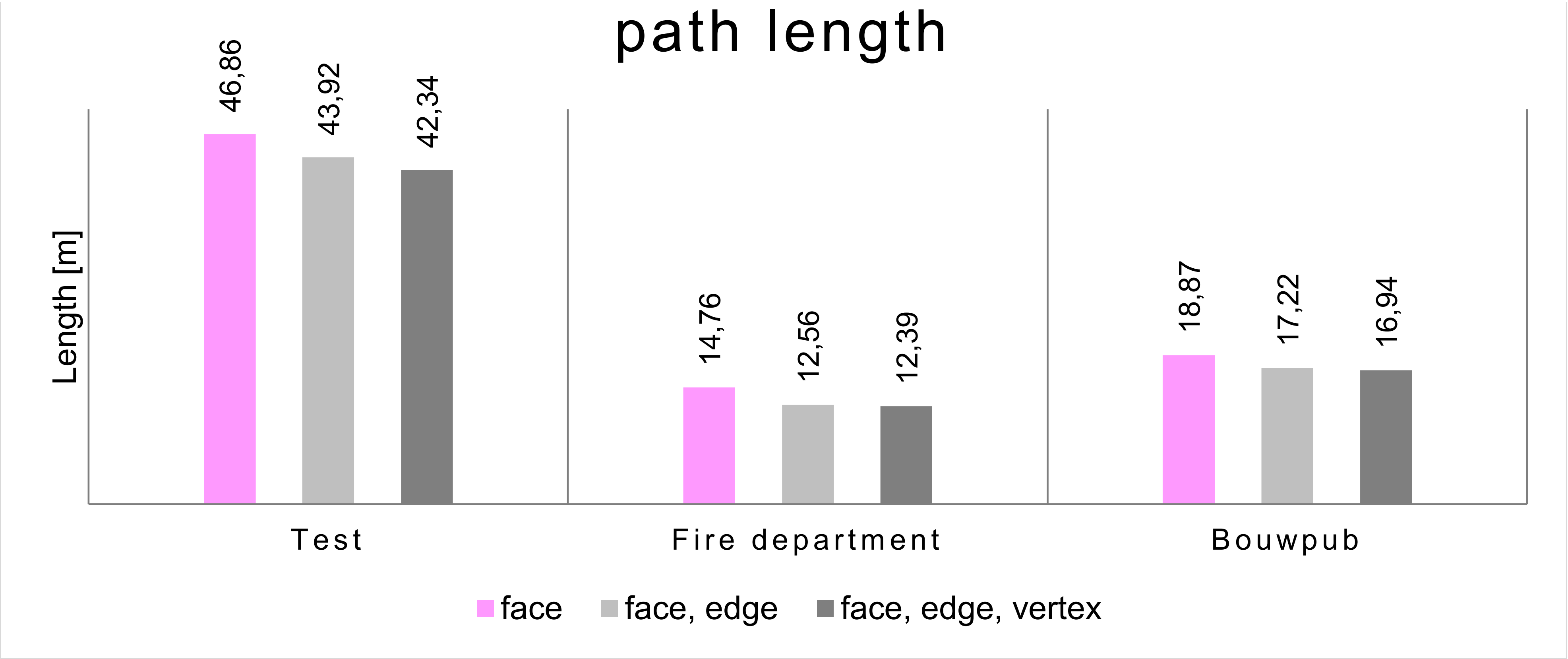
5.6. Benchmark results

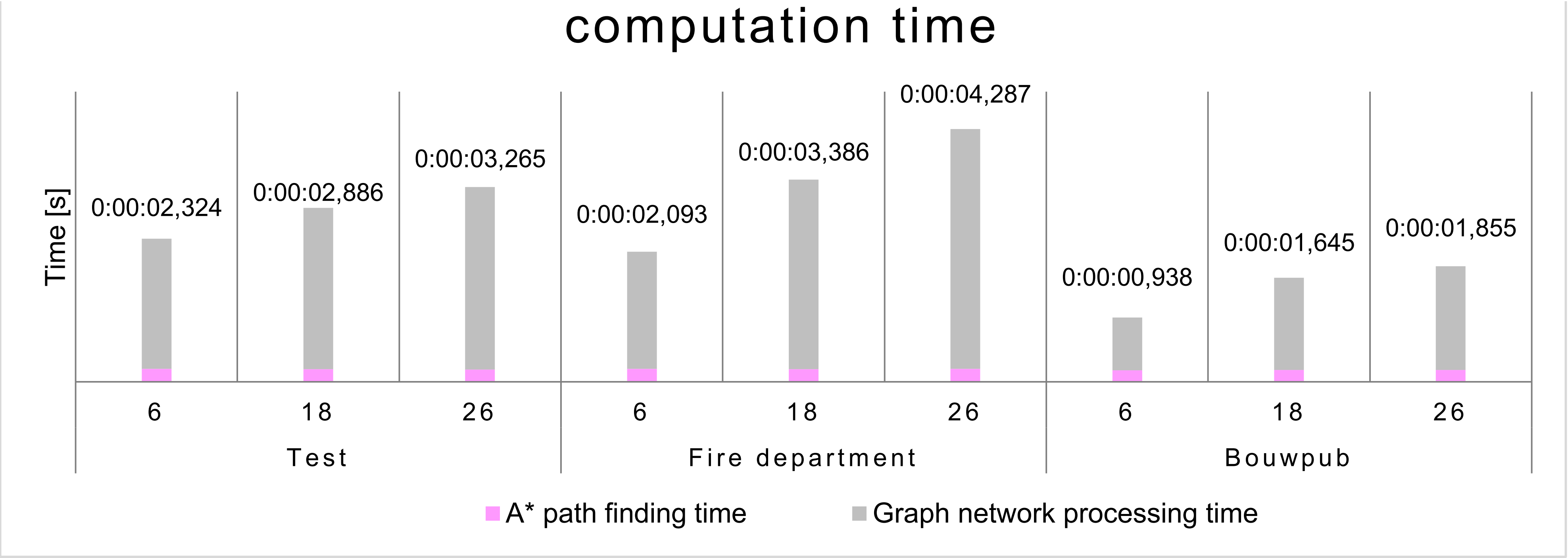
Pre-processing connectivity

1. Path length
   1.1. Not effected
2. Path finding computation time
   2.1. Improvement in computation time
   2.2. Bottle neck is loading and processing network graph

5.6. Benchmark results

Path connectivity

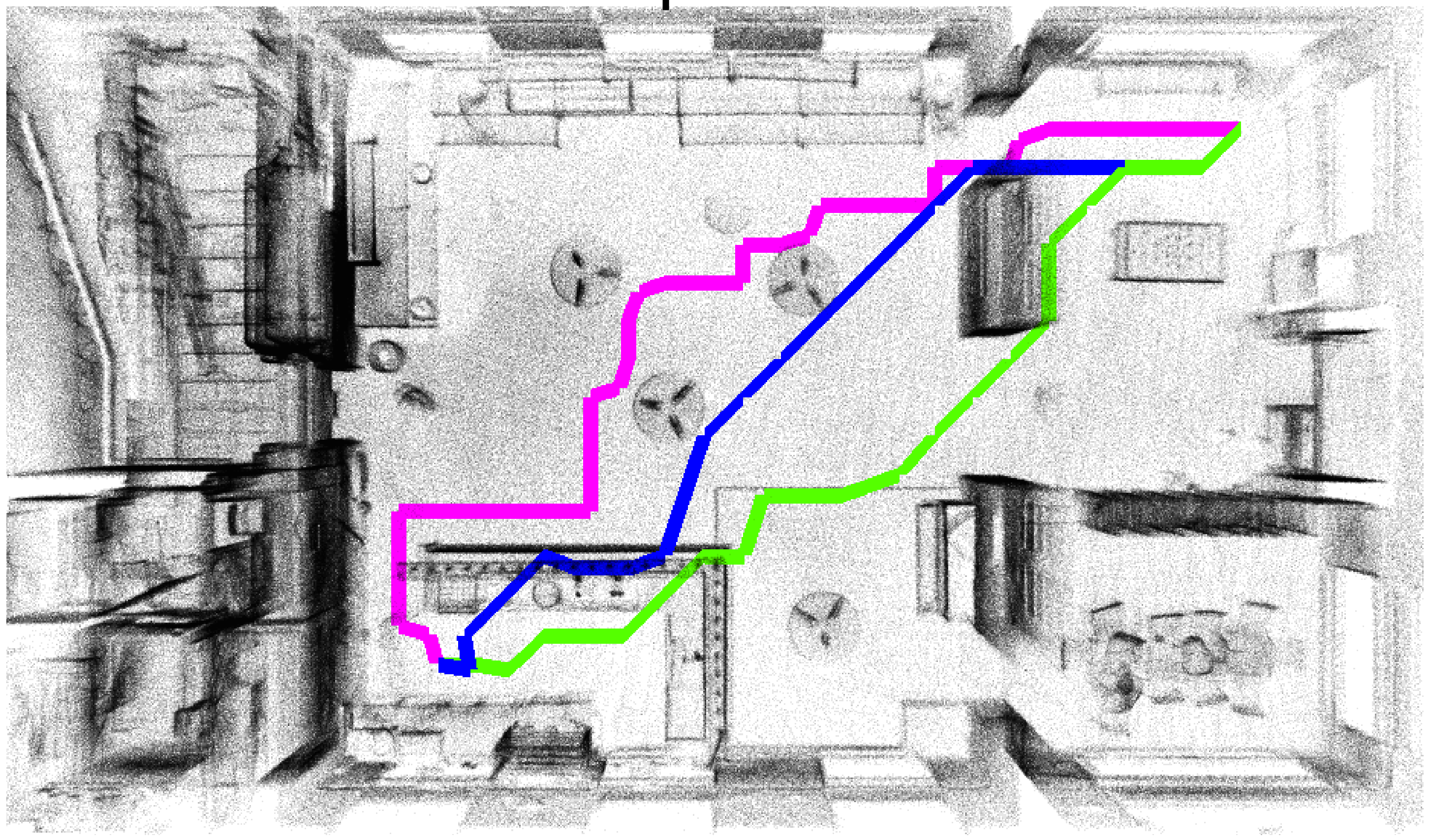




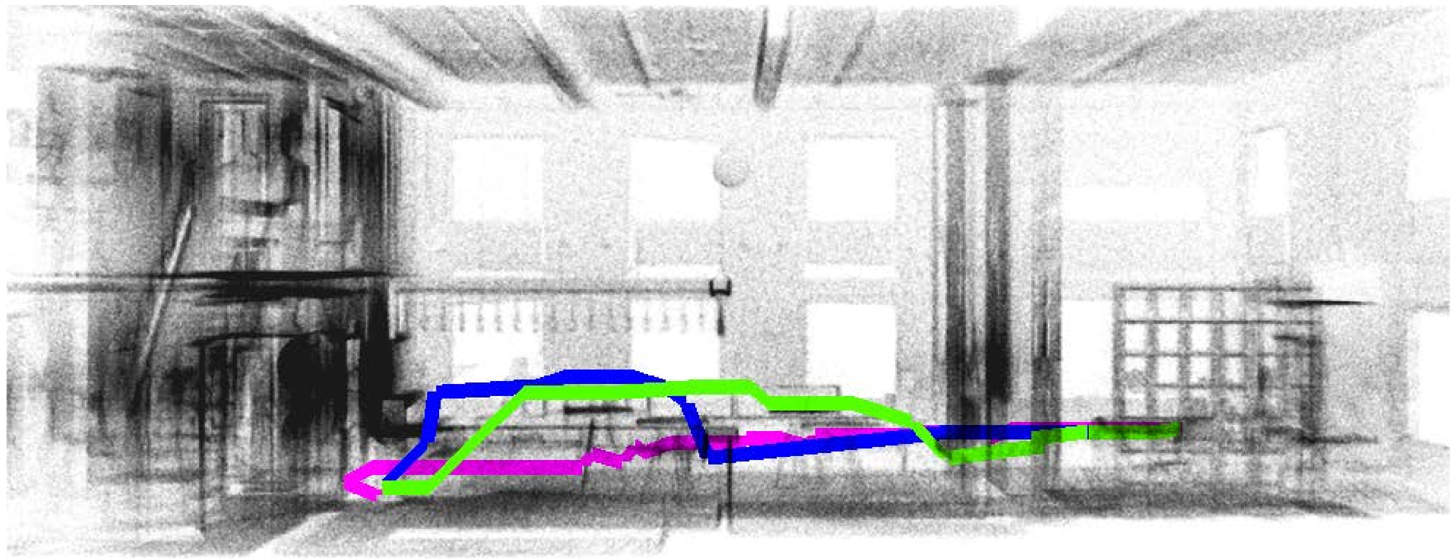
5.6. Benchmark results

Path connectivity

Top view



Side view



1. Path length
   1.1. Between 10% and 12% reduction in path length
2. Path finding computation time
   2.1. Increases as the connectivity is extended
   2.2. Mainly due to size of network graph

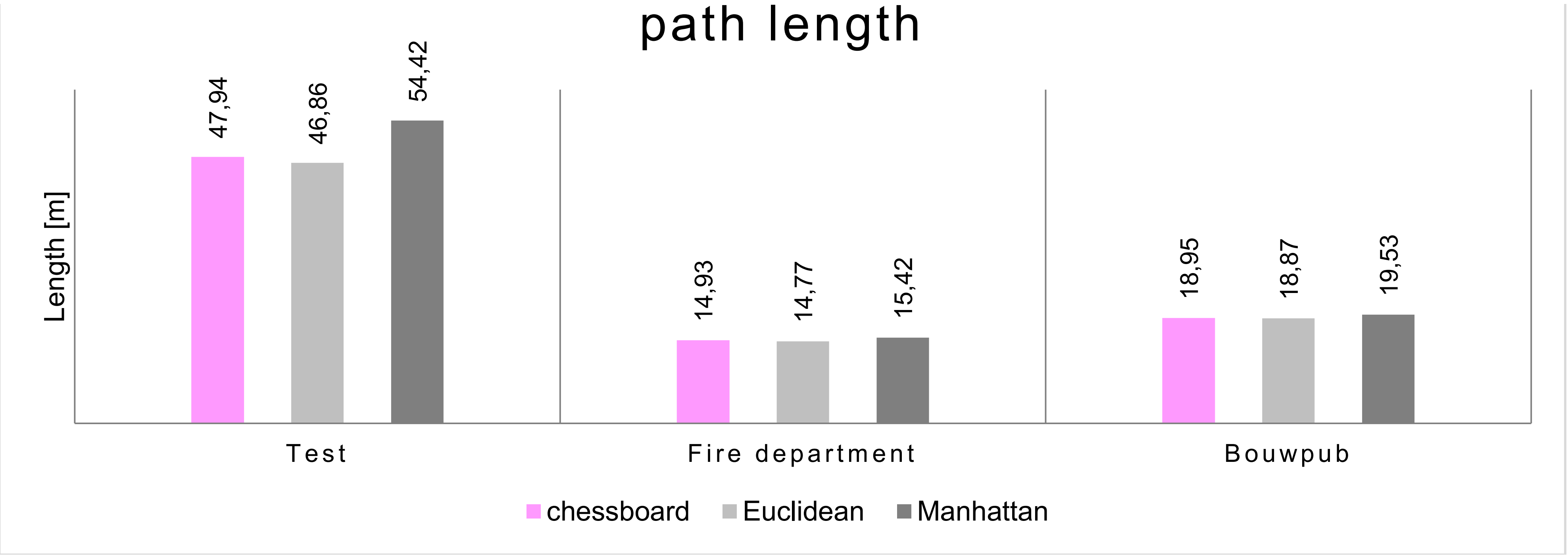
🟪 Face

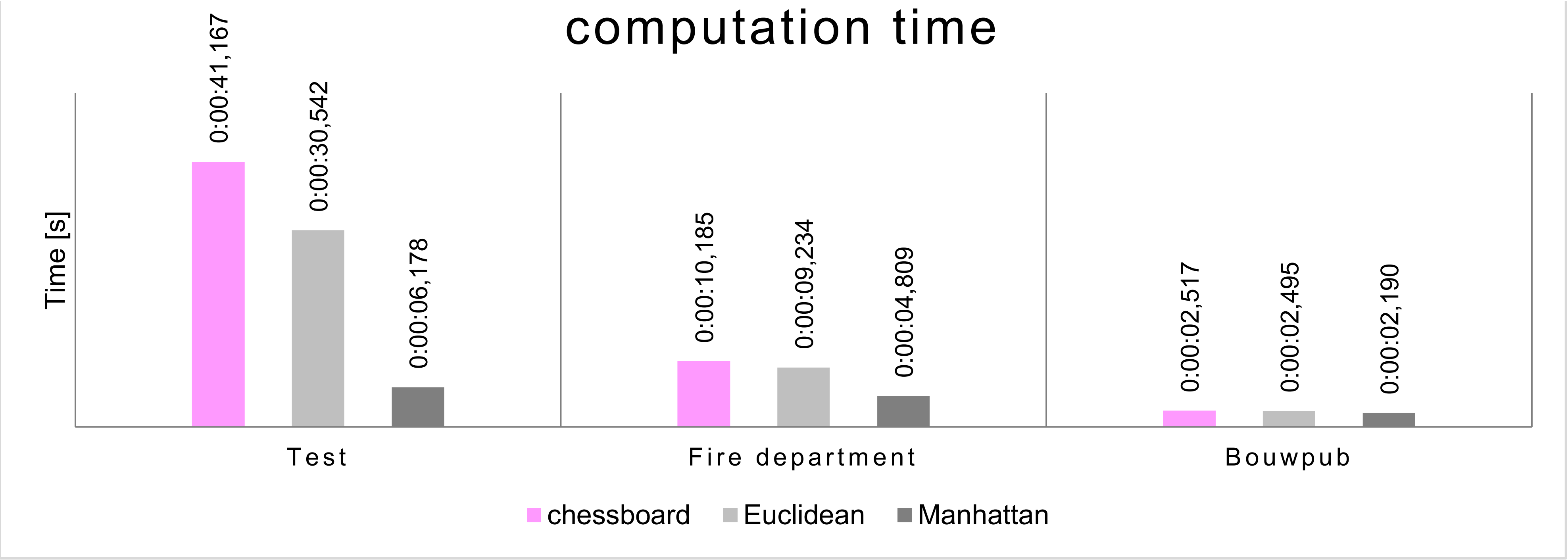
🟦 Face and edge

🟩 Face, edge and vertex

5.6. Benchmark results

Distance type

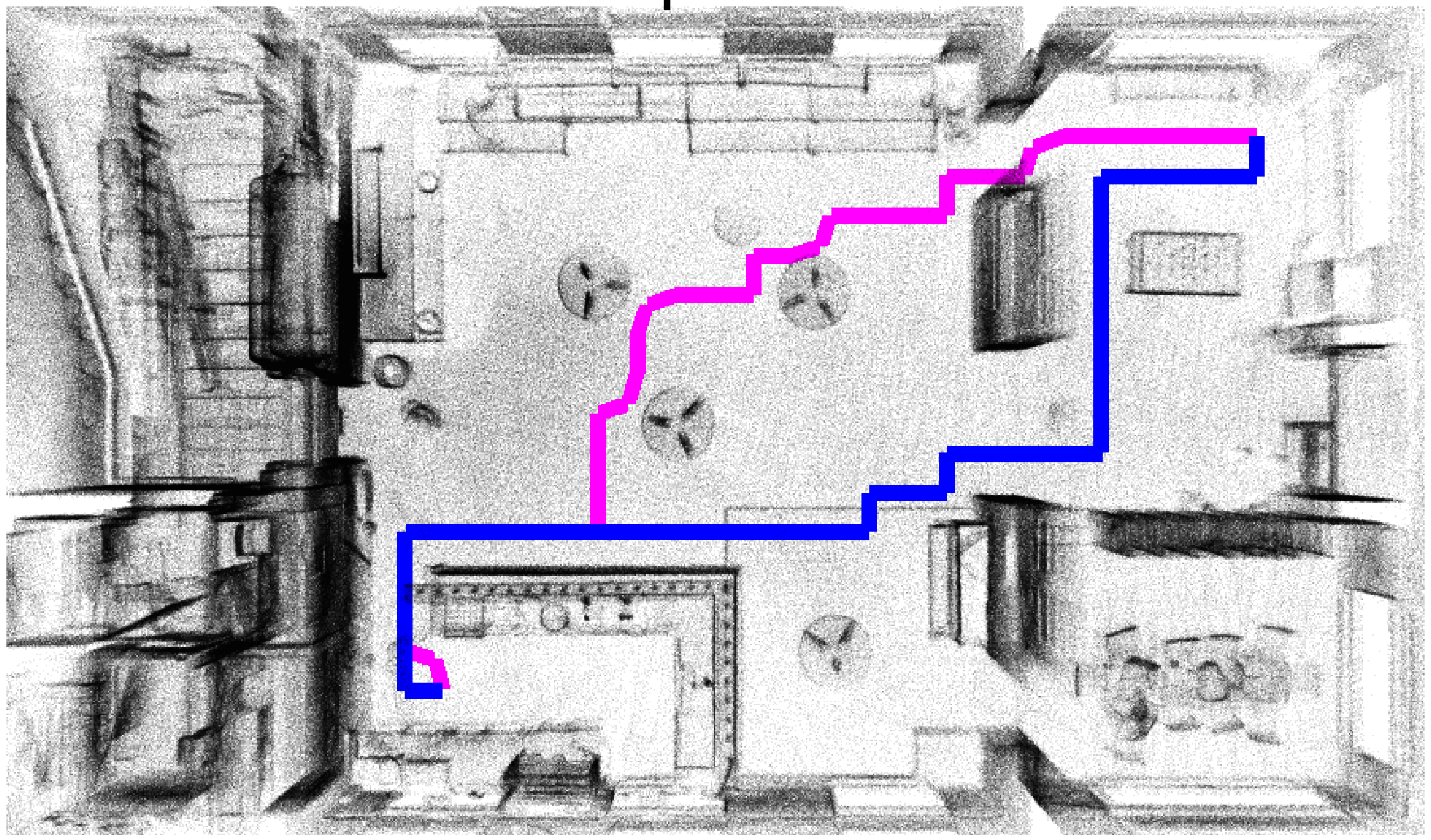




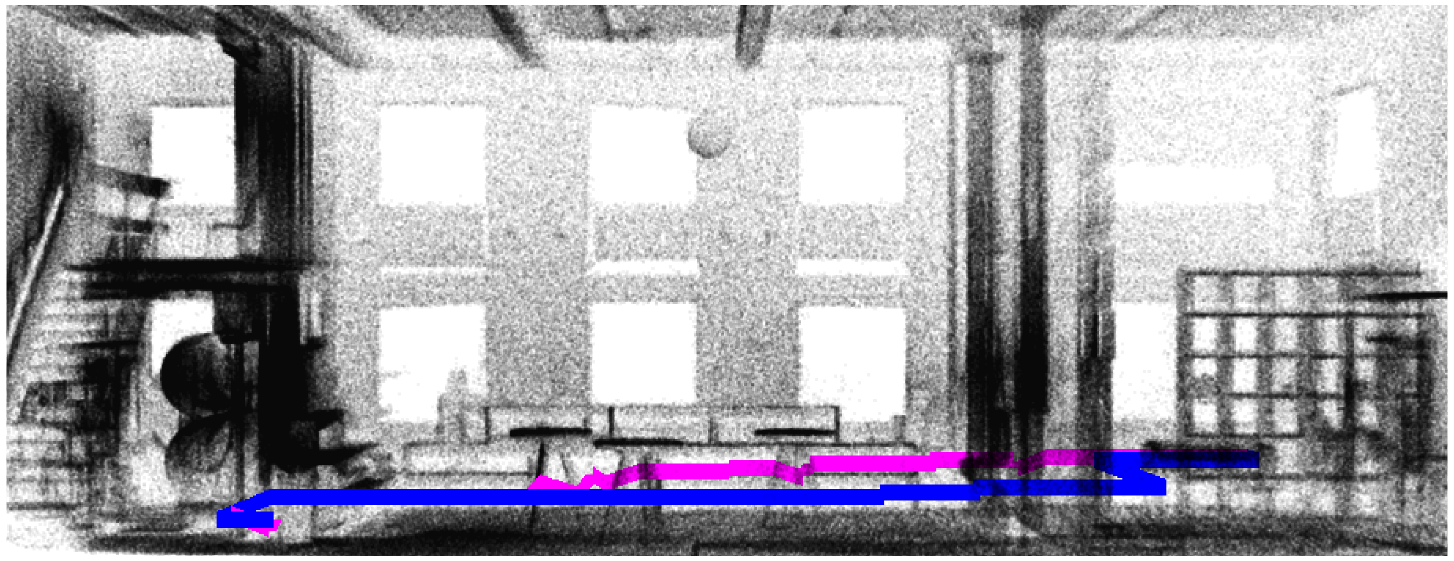
5.6. Benchmark results

Distance type

Top view



Side view



1. Path length
   1. 1.1. Euclidean distance shortest
   2. 1.2. Manhattan distance longest
2. Path finding computation time
   1. 2.1. Manhattan distance finds path fastest

▬ Euclidean

▬ Manhattan

6. Conclusions and Future work

6.1. Conclusion
6.2. Future work

6.1. Conclusions

- I created a work flow to pre process a dataset usable for collision free path finding
- Necessary octree depth depends on size of the point cloud
- Spatial resolution can be improved by geometrical point cloud operations
- Pre processing network graph is beneficial for computation time
    - Bottleneck in computation time is loading and processing network graph
- Extending path connectivity reduces the path length
- Manhattan distance is most suitable for computation time and Euclidean distance for path length

6.2. Future work

- Improve efficiency of storing and accessing network graph
- Automatic alignment of point cloud
- Integrate interior and exterior
- Research larger and more complex buildings
- Create web service

Thank you for your attention!