

Voxels and voxelisation

GEO1004:
3D modelling of the built environment

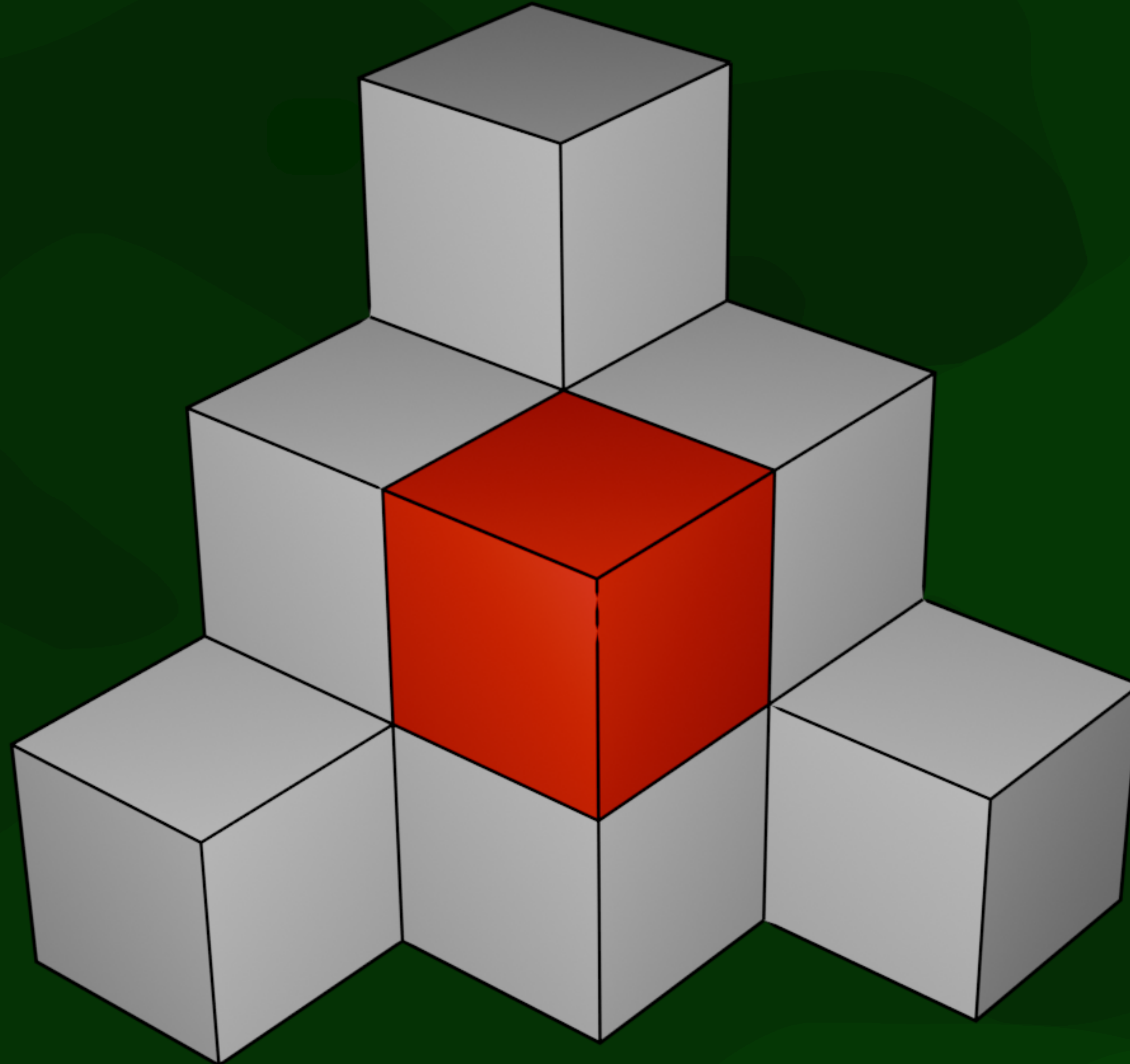
<https://3d.bk.tudelft.nl/courses/geo1004>



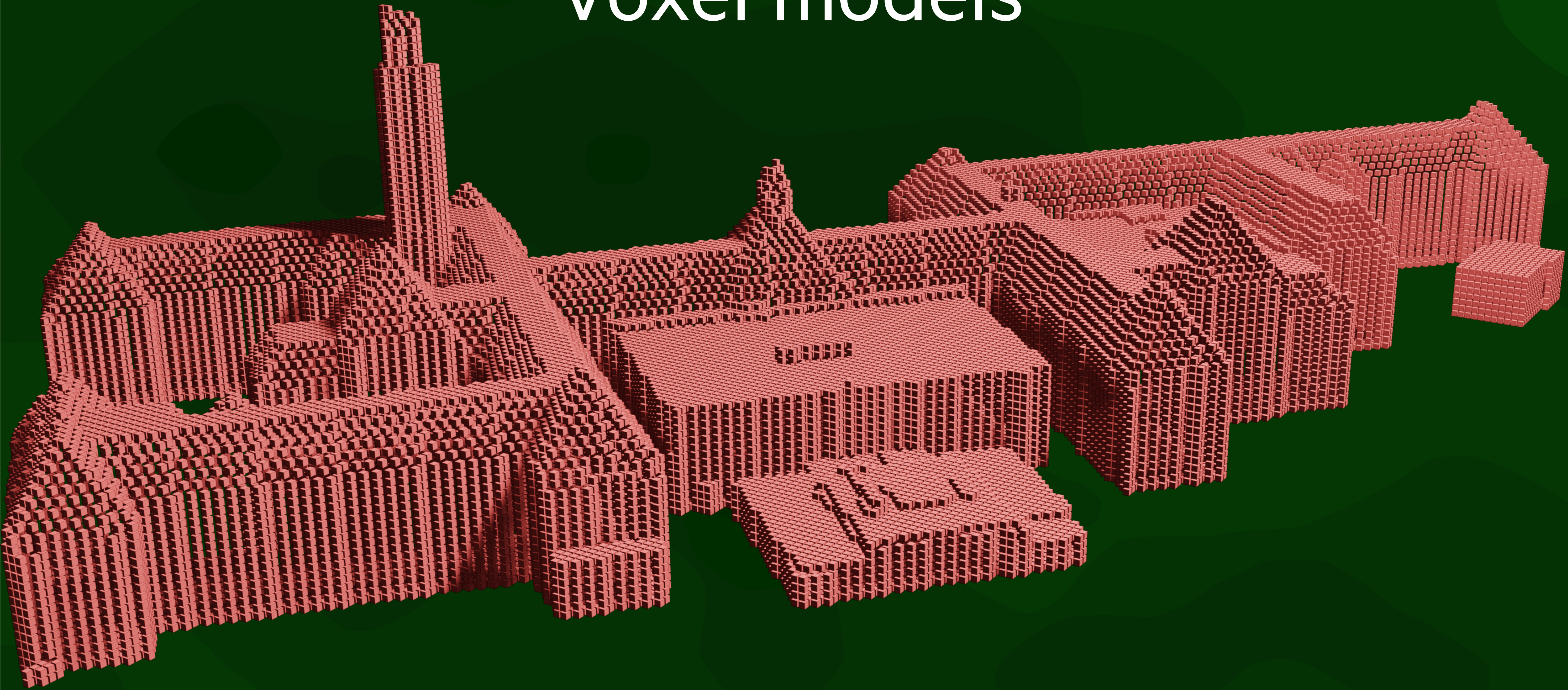
3D geoinformation

Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

Voxels: 3D pixels



Voxel models



What is the volume of BK?
(brep vs. voxel)

Volume of BK

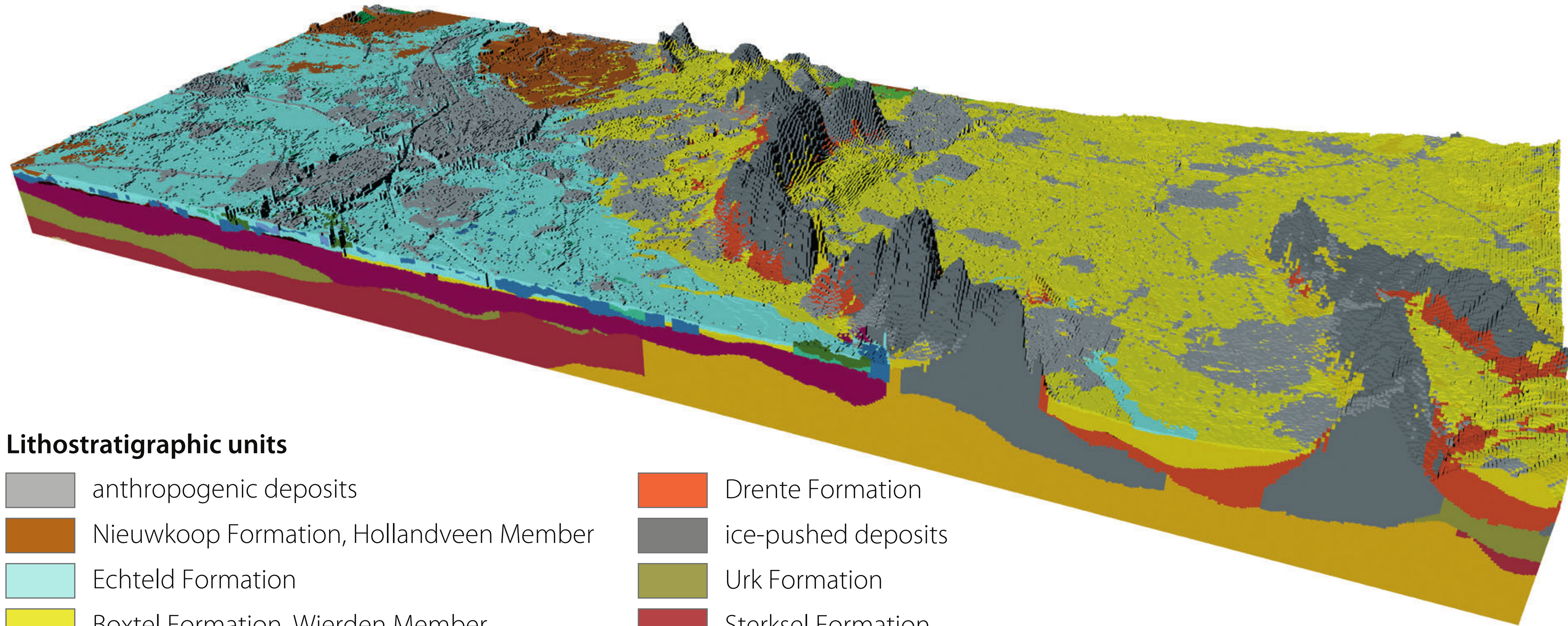
- B-rep model:
 - Constrained tetrahedralisation of interior? -> requires watertight geometry
 - Oriented volume using triangulated faces -> requires correctly oriented faces
 - Voxels: count voxels, multiply by volume of voxel

Why?

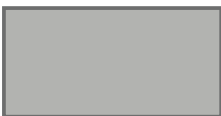
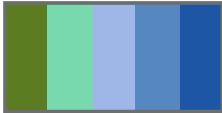
- Simplicity: simple storage, simple processing methods, simple to use in applications
- Robustness: many issues solved automatically or at least more easily, few degenerate cases
- Speed: easy to do parallel processing by processing voxels independently
- A good match for (interpolated) fields data

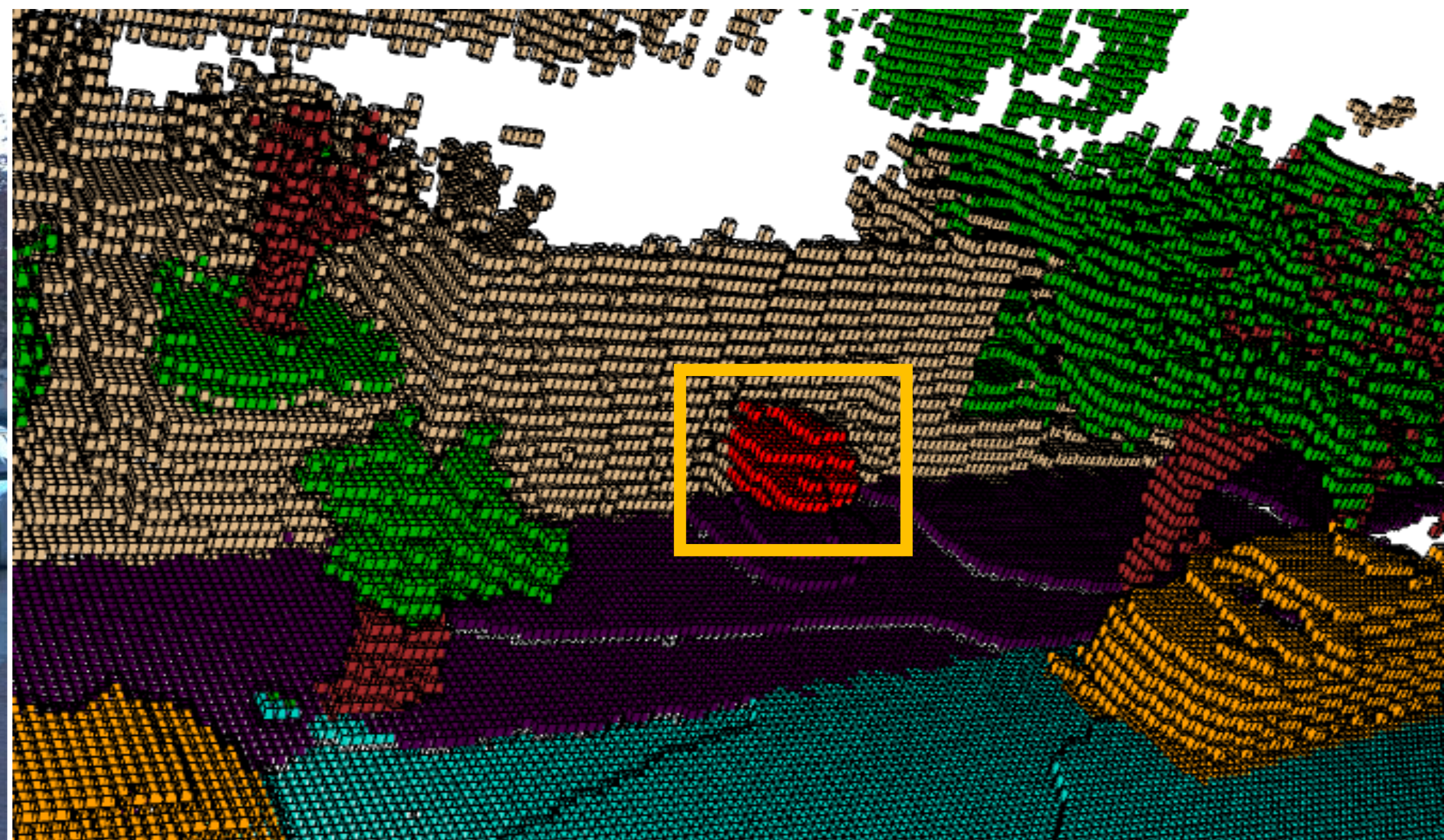
Where?

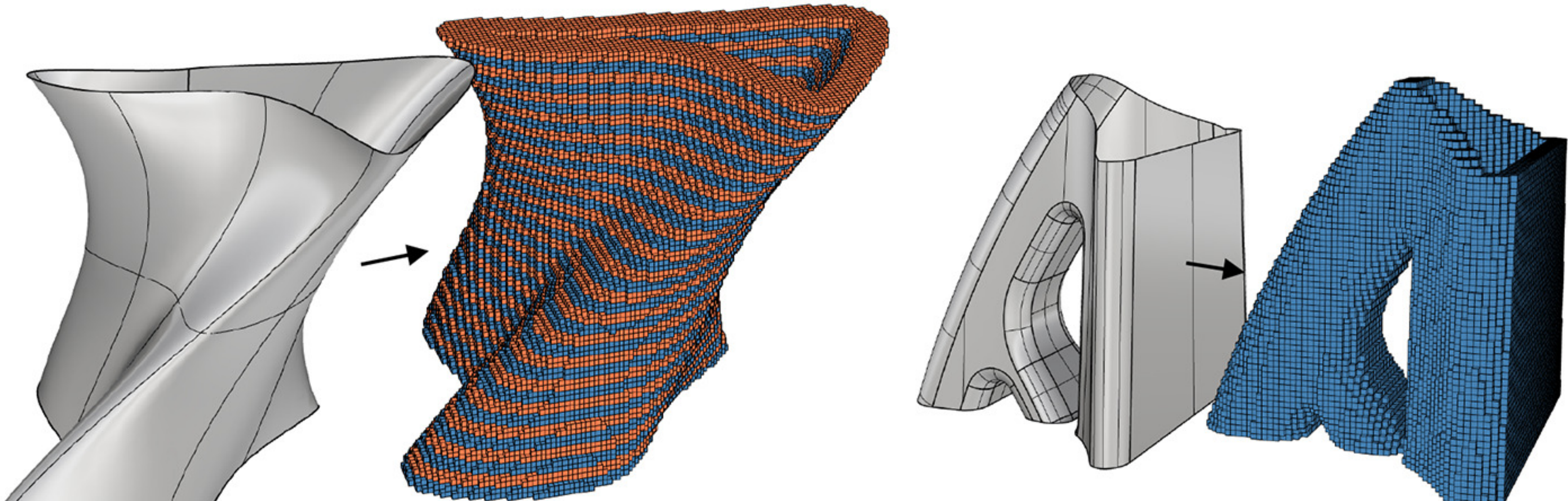
- Visibly: some datasets (e.g. most geological data) and software (e.g. voxel layers in GRASS and ArcGIS)
- But mostly invisibly: simulations, navigation, DL models, optimisation, computer graphics and rendering, etc.
- Plus related fields: games (e.g. visibly in Minecraft), medical imaging (CT/MRI scans), 3D printing and manufacturing



Lithostratigraphic units

- | | | | |
|--|---|---|--|
|  | anthropogenic deposits |  | Drente Formation |
|  | Nieuwkoop Formation, Hollandveen Member |  | ice-pushed deposits |
|  | Echteld Formation |  | Urk Formation |
|  | Boxtel Formation, Wierden Member |  | Sterksel Formation |
|  | Boxtel Formation |  | Peize and Waalre formations |
|  | Kreftenheye Formation |  | Echteld Formation, fluvial channel belts |





Nomenclature

- 2D:

- pixel
- image / 2D raster
- (2D) rasterisation

- 3D:

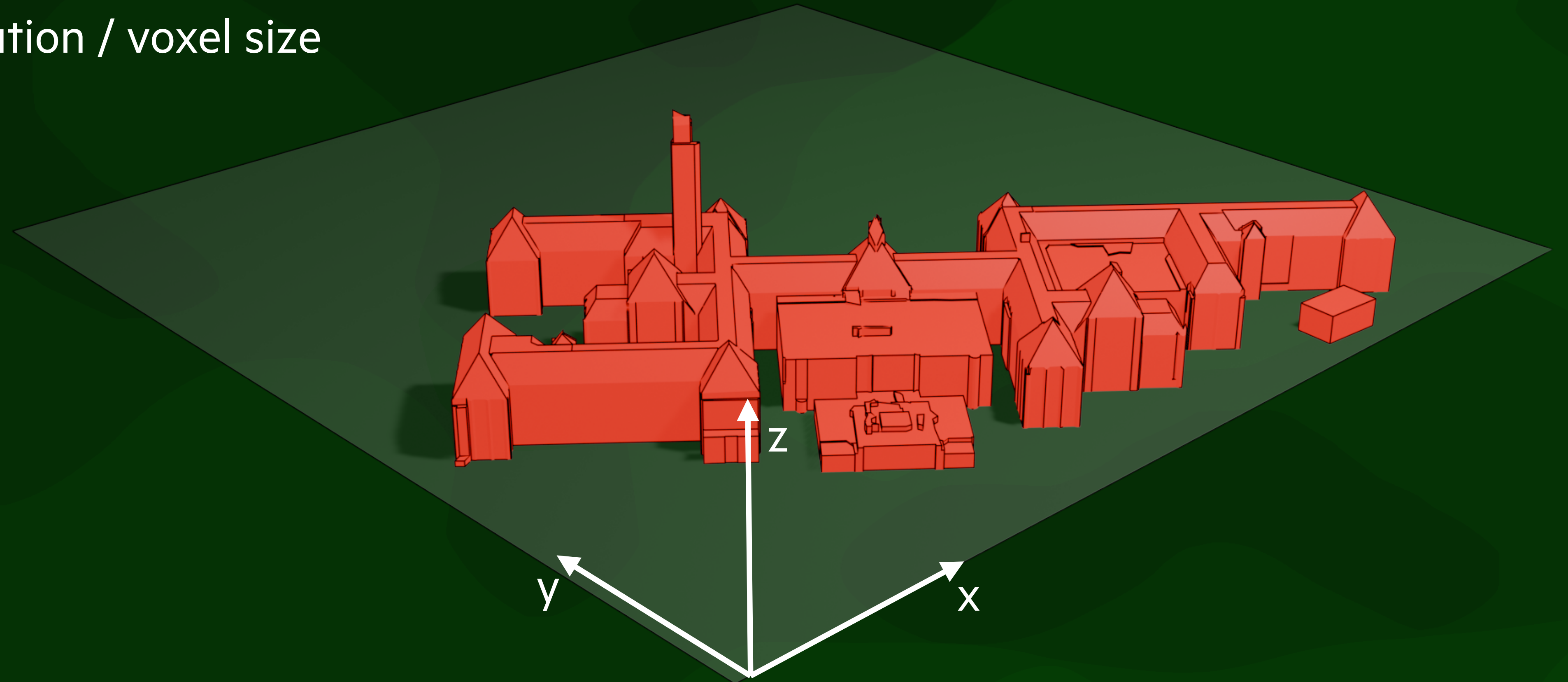
- voxel
- voxel model / 3D raster
- voxelisation / 3D rasterisation

Today: voxel-related representations

- Voxels: variations, compression, different encodings
- Hierarchical representations (e.g. octrees) and their variations
- Voxelisation

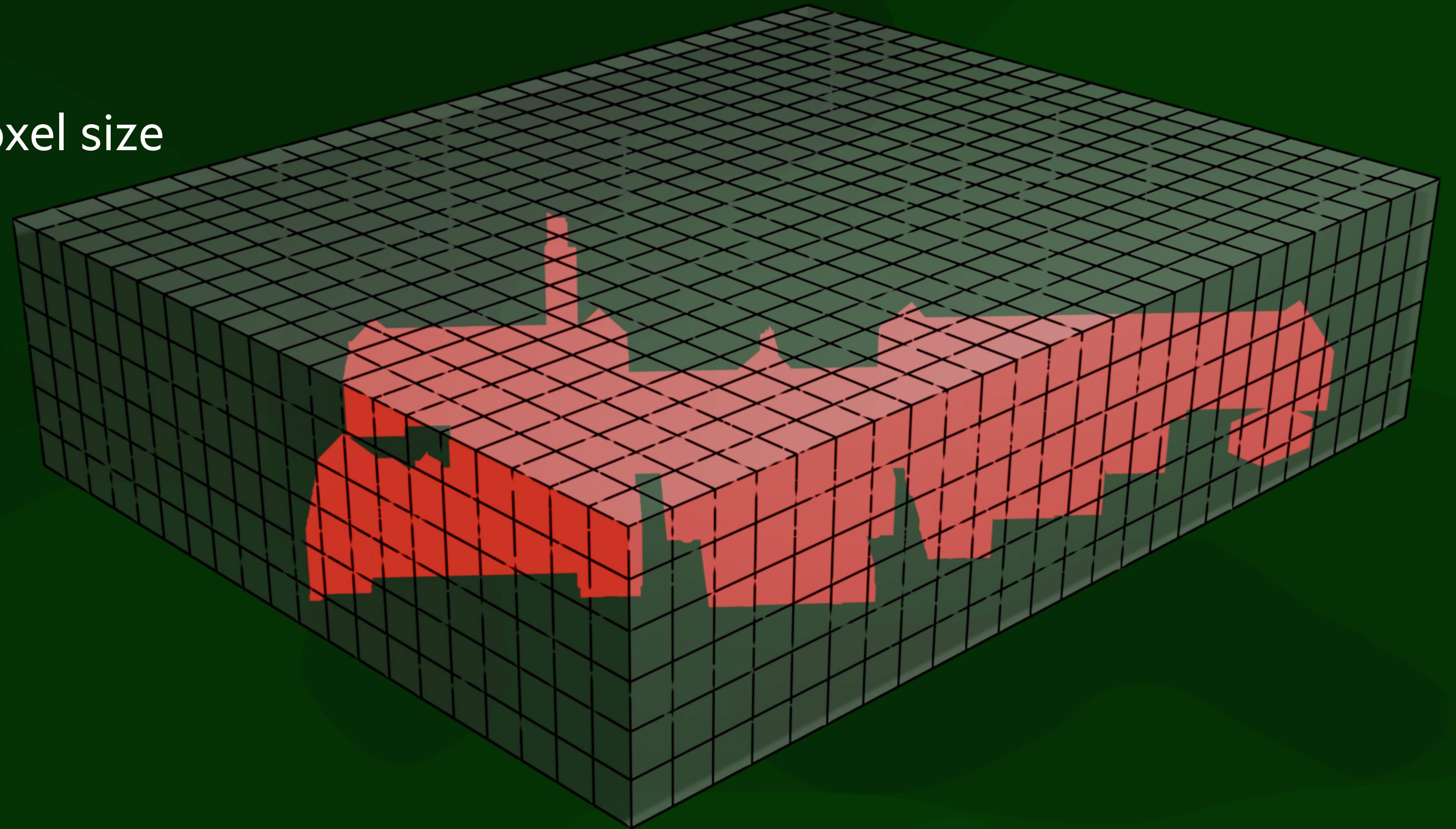
Setup

1. Define domain (origin and orientation / axes)
2. Define resolution / voxel size



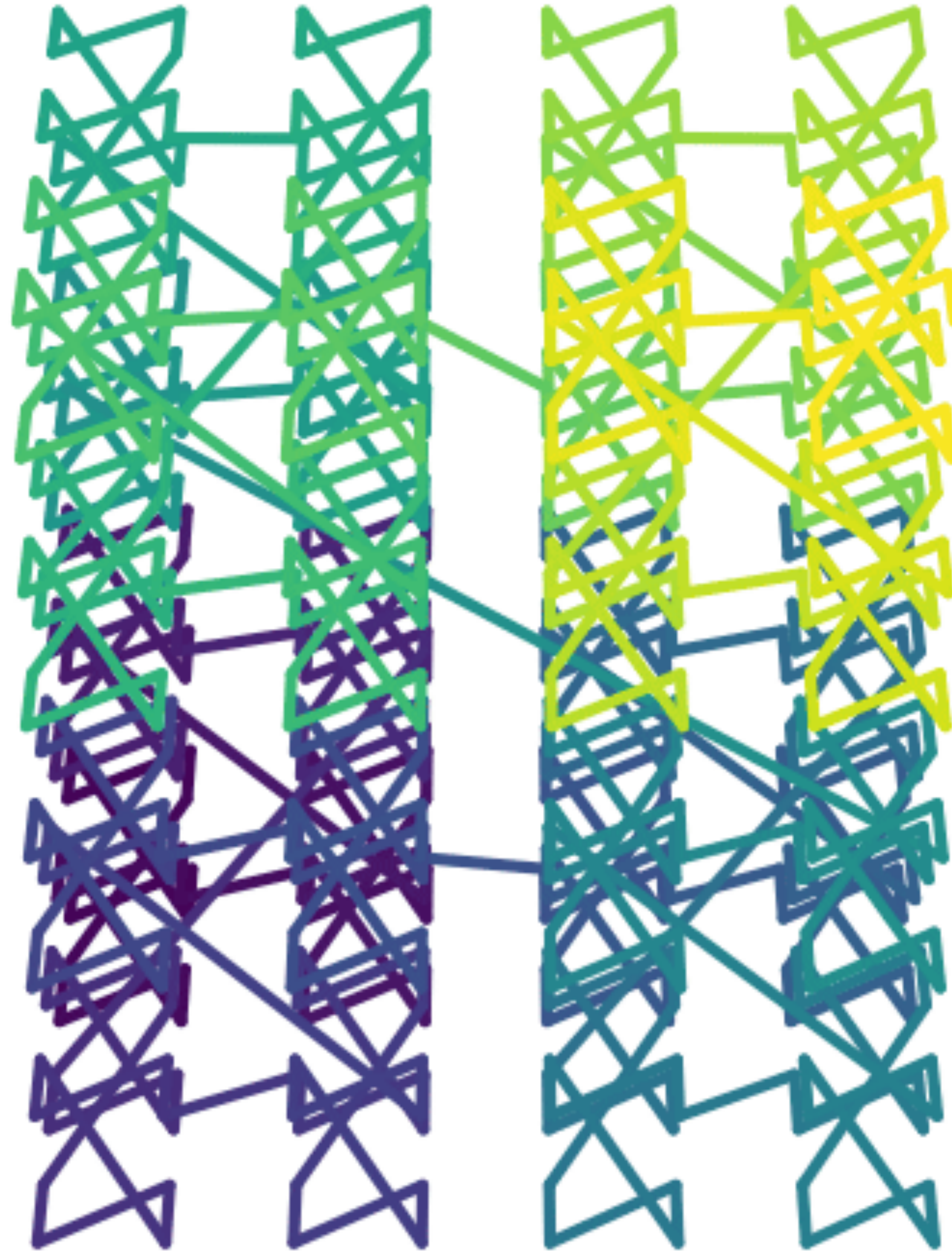
Setup

1. Define domain (origin and orientation / axes)
2. Define resolution / voxel size



Usual (dense) encoding

1. Define order for storage / traversal (first x $0\dots i$, then y $0\dots j$, then z $0\dots k$)
2. Following the order: specify what is in each voxel, e.g. occupied or not, object IDs, field values, etc.
 - [..., false, true, true, false, ...]
 - [..., 1, 1, 3, 3, 1, ...]
 - [..., 3.45, 3.56, 3.78, 3.99, ...]



Sparse encoding

- List of entries containing for each voxel:
 - location (along x, y, z)
 - contents
- e.g. $(x, y, z, ID) = \dots, [100, 105, 0, 3], [101, 105, 0, 3], \dots$

Other forms of compression

- Run-length encoding: repeated entries $\rightarrow$ value + count
- Differential encoding: (large) values $\rightarrow$ difference w.r.t. last voxel

What encoding is better?

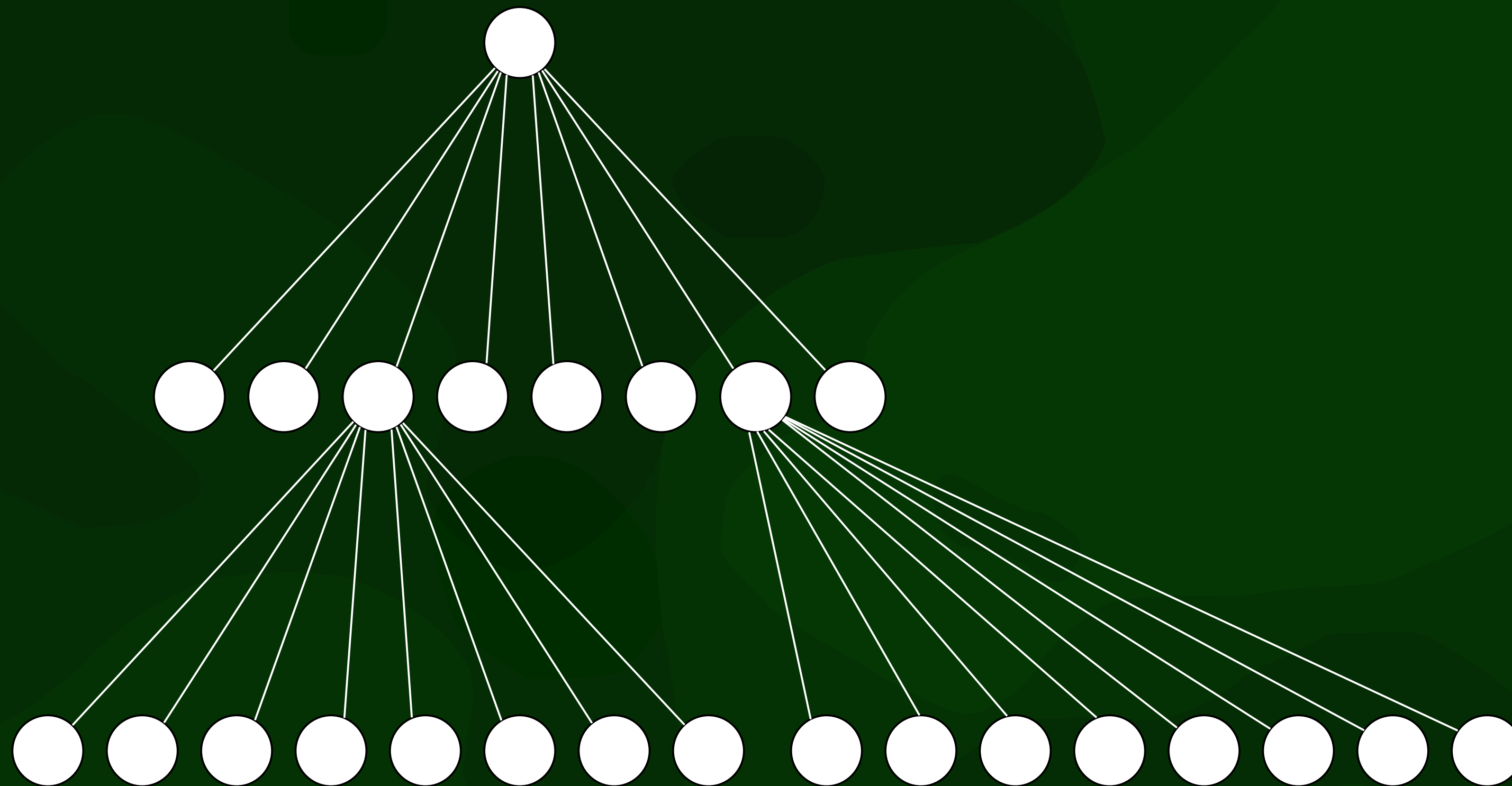
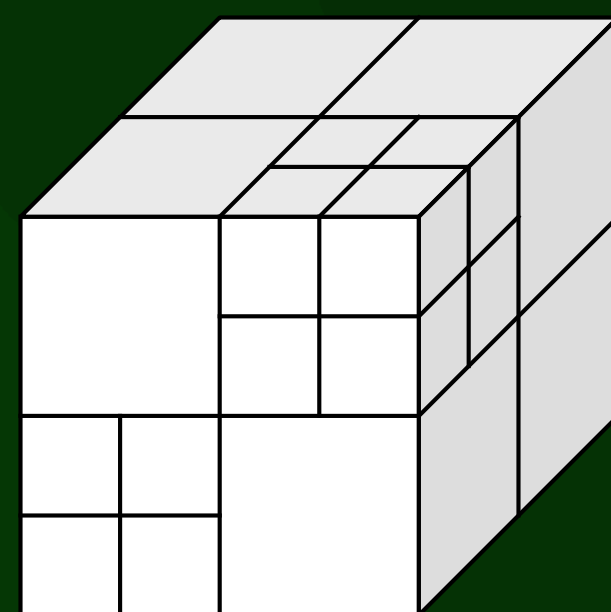
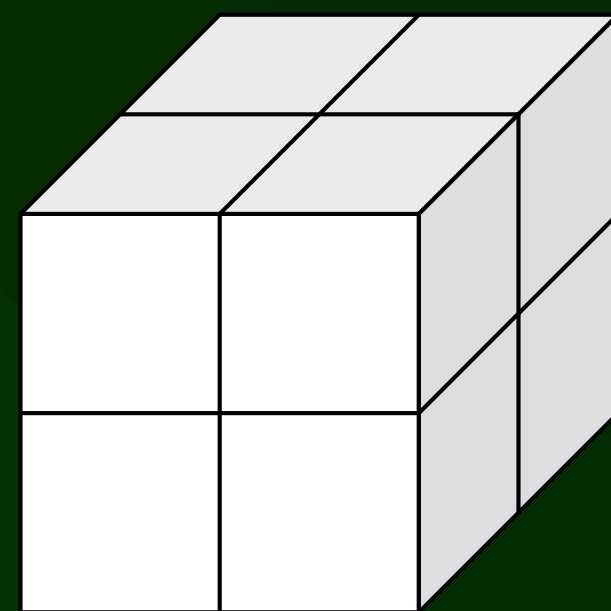
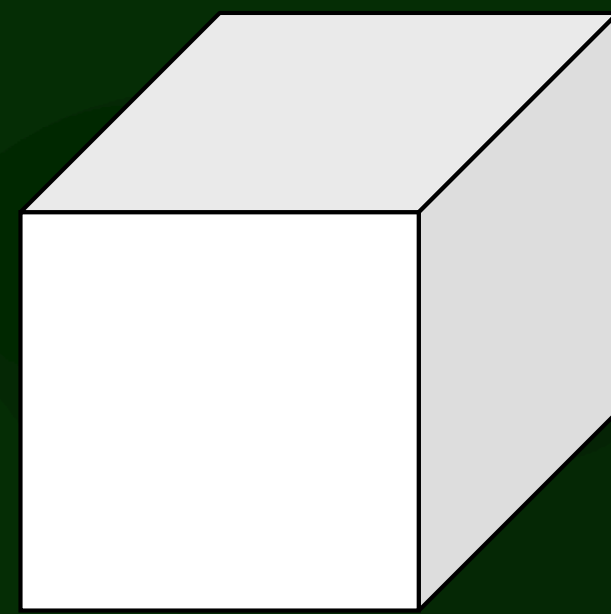
What encoding is better?

- Storage?
- Access time?
- Ease of implementation?
- Speed of different operations?

Variations

- Varying sizes, e.g.
 - anisotropic voxels - different size along each dimension (e.g. better resolution for z for geologic data, lower resolution in medical images across slices)
 - adaptive resolution - smaller size around particular areas
- Differently shaped cells, e.g. tetrahedra or octahedra in a regular pattern
- Dual voxels (data in vertices), used in dual contouring

Hierarchical subdivisions

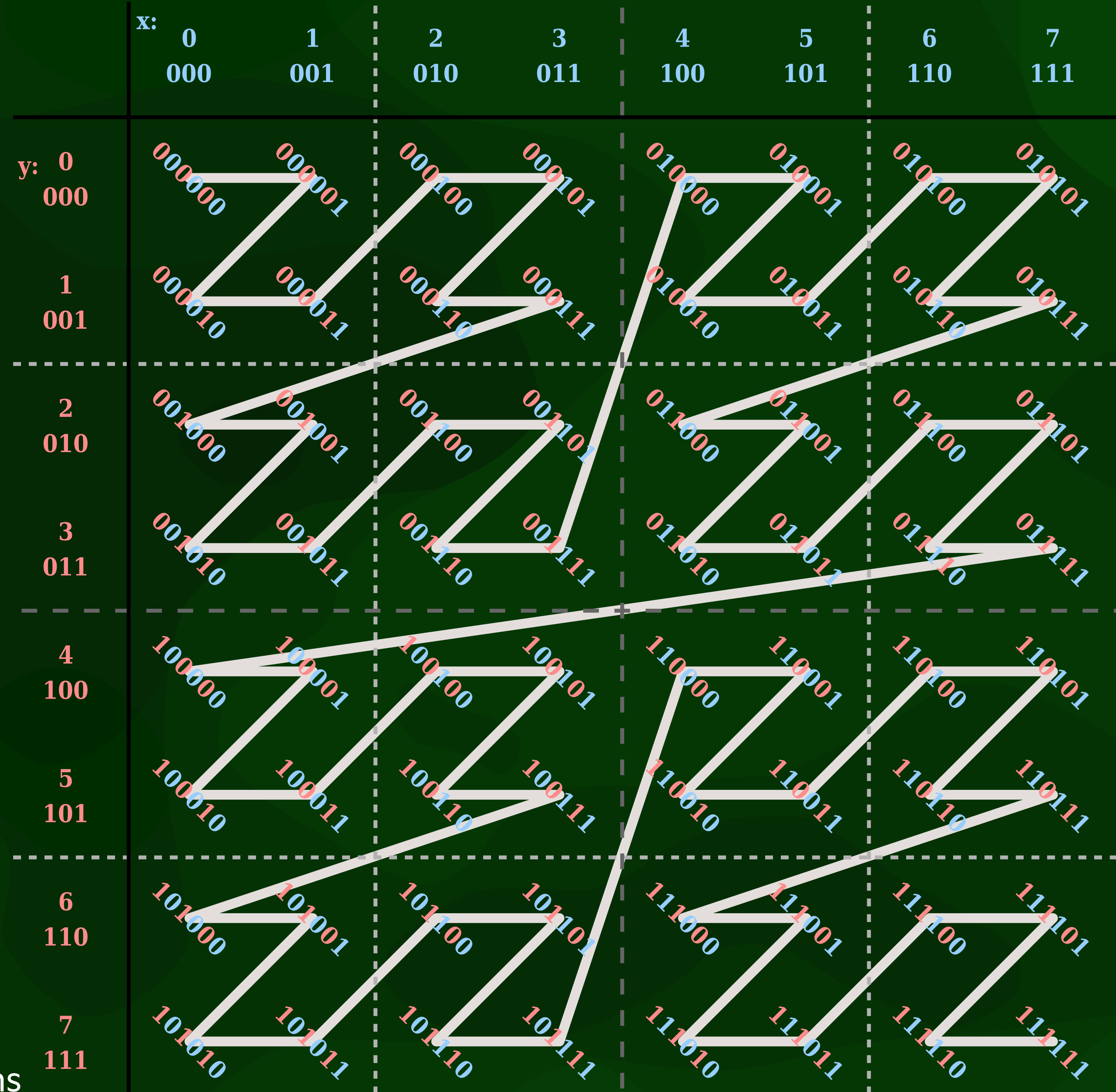


Hierarchical subdivisions

- Octrees: in half along every axis
- Bintrees: in half along a sequence of axes (typically x, then y, then z, then x...)
- k-d trees: using a stored value along alternating axes

Octrees

- Construction: top-down, rules for splitting and for stopping
- Pointer-based vs. linear



Bintrees

- Two children per node -> more space-efficient but deeper trees
- Can adapt to anisotropic distribution along axes (e.g. geological data with more resolution along z axis)

kd-trees

- Choose a splitting value rather than just halves -> greater adaptability, e.g.
 - Median -> half of objects in each side, balanced tree
 - Disjoint sets -> easy and fast queries

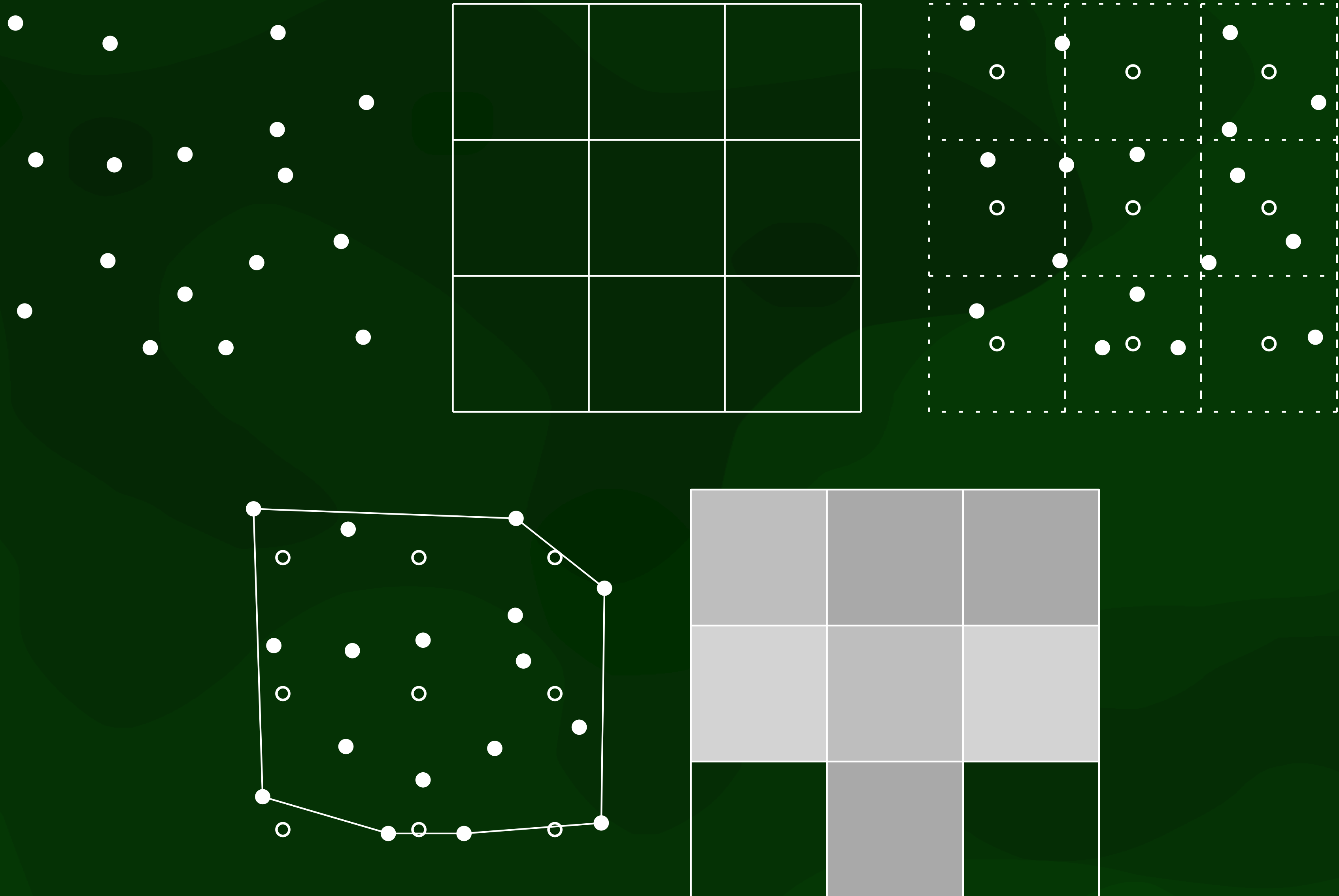
Don't want visible voxels?

Use voxels for operations in the background but...

show brep instead

Voxelisation

Voxelisation of fields



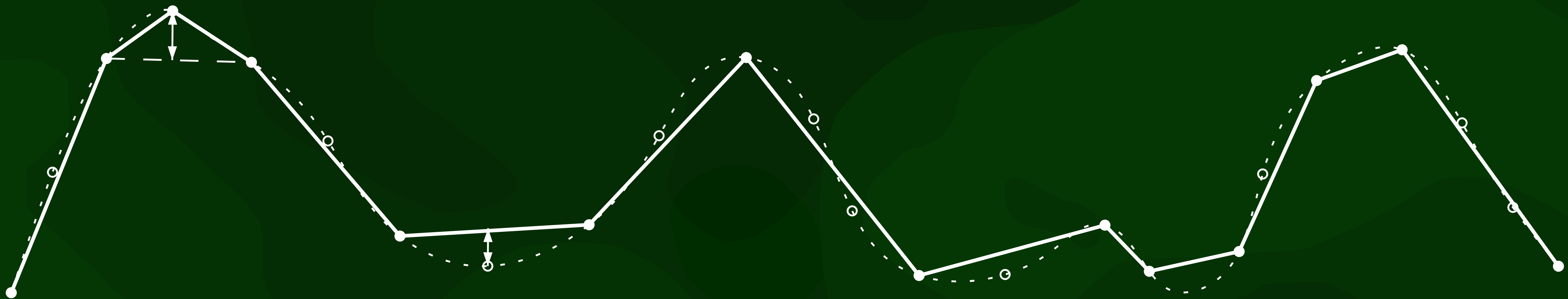
Voxelisation of fields

- Nearest neighbour
- IDW
- Linear in tetrahedra
- Natural neighbour
- Kriging
- etc.

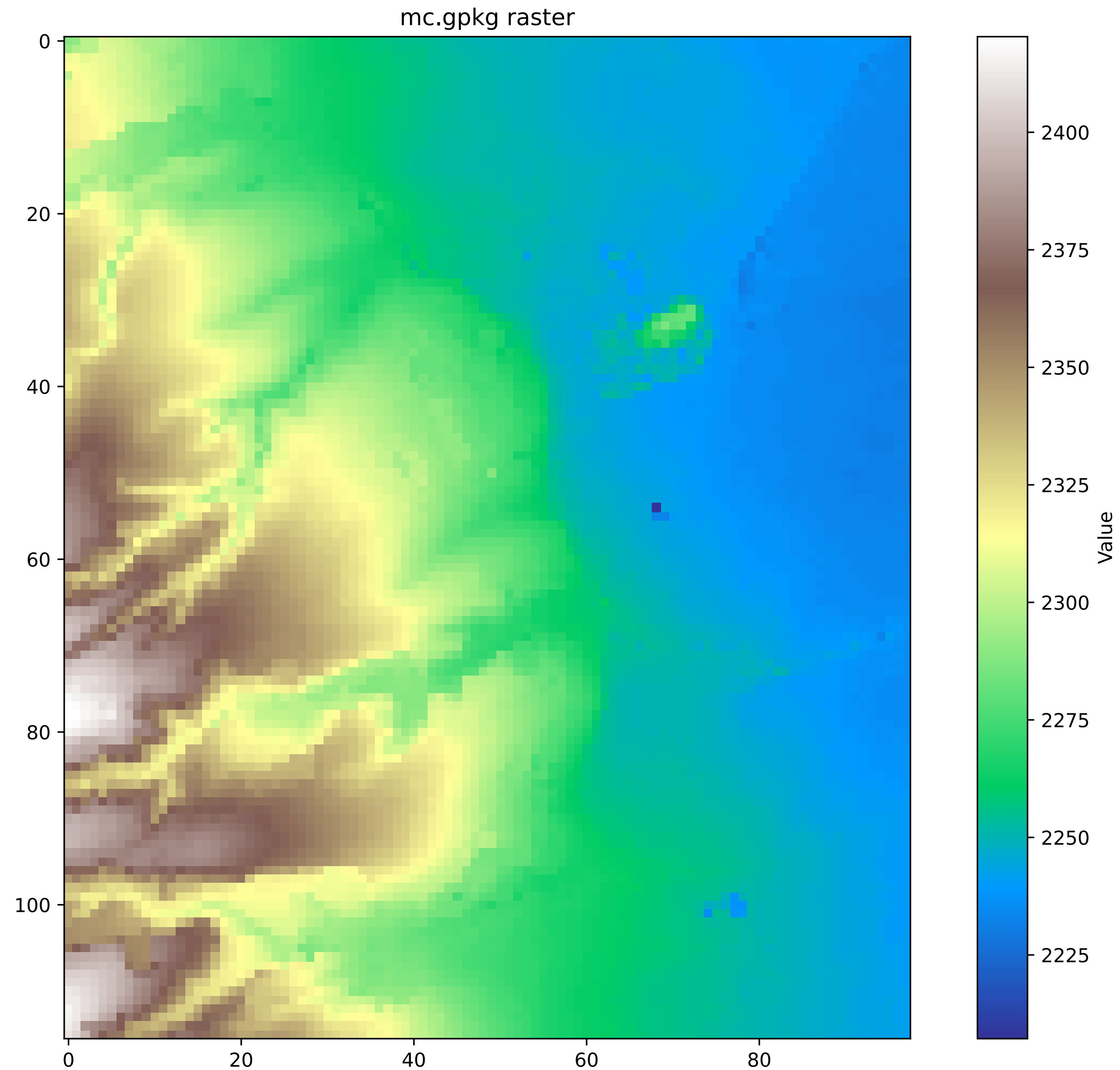
Reverse process

- Simplest approach:
 - Create line segments (1D) or triangles/quads (2D/brep 3D) using every voxel center
 - Simplify using decimation
- Plenty of more advanced methods, e.g. marching cubes, dual contouring

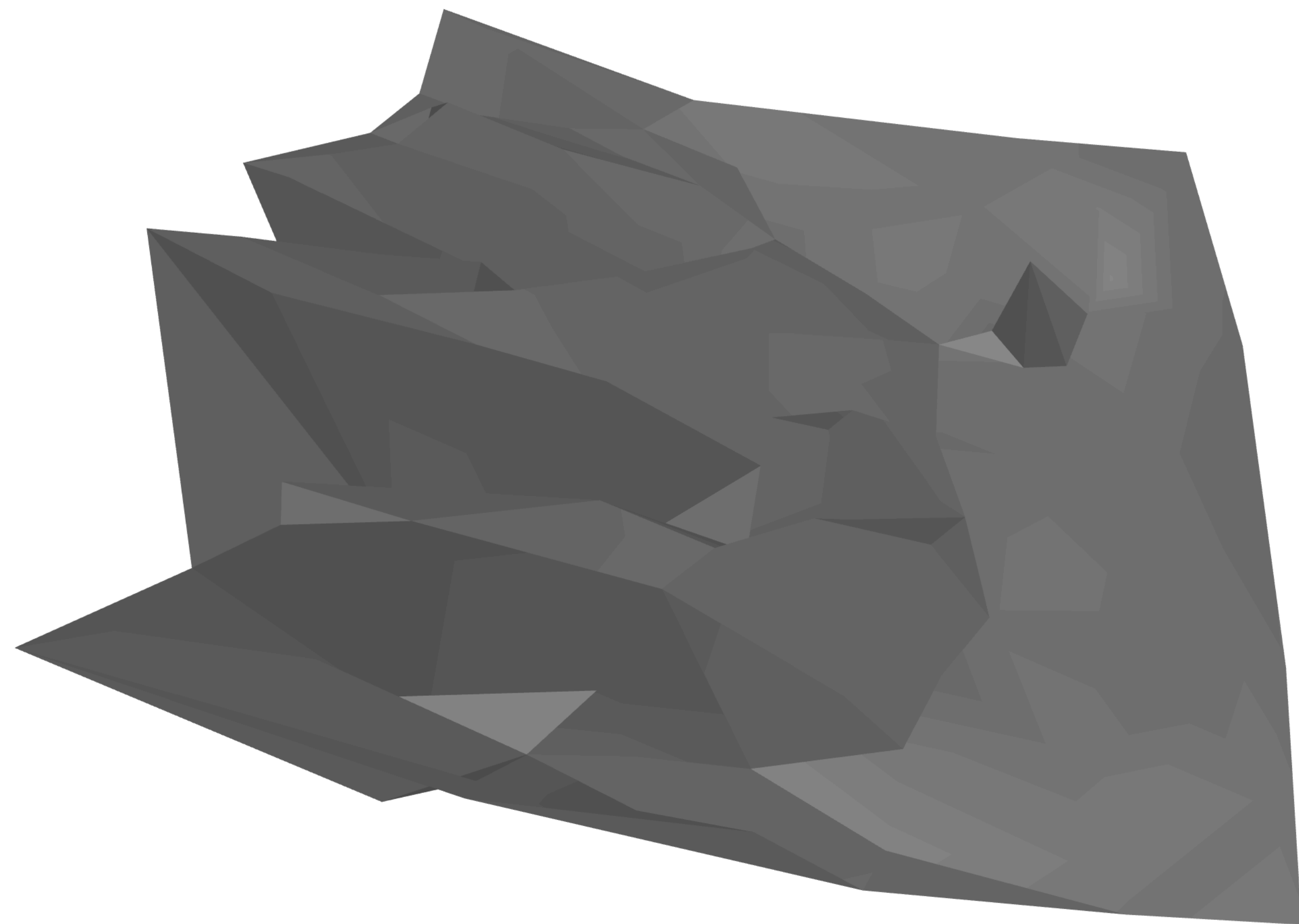
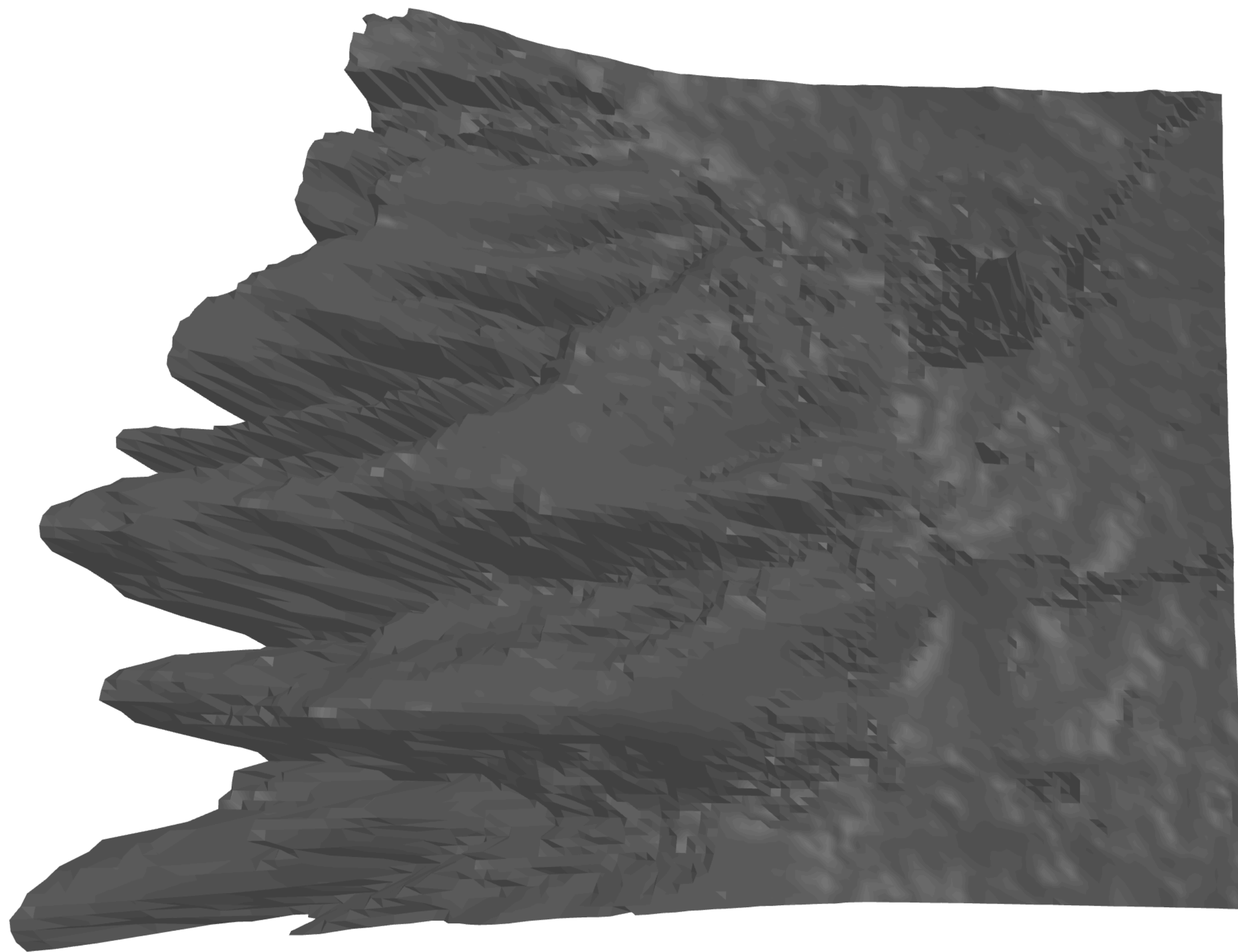
Reverse process (profile)



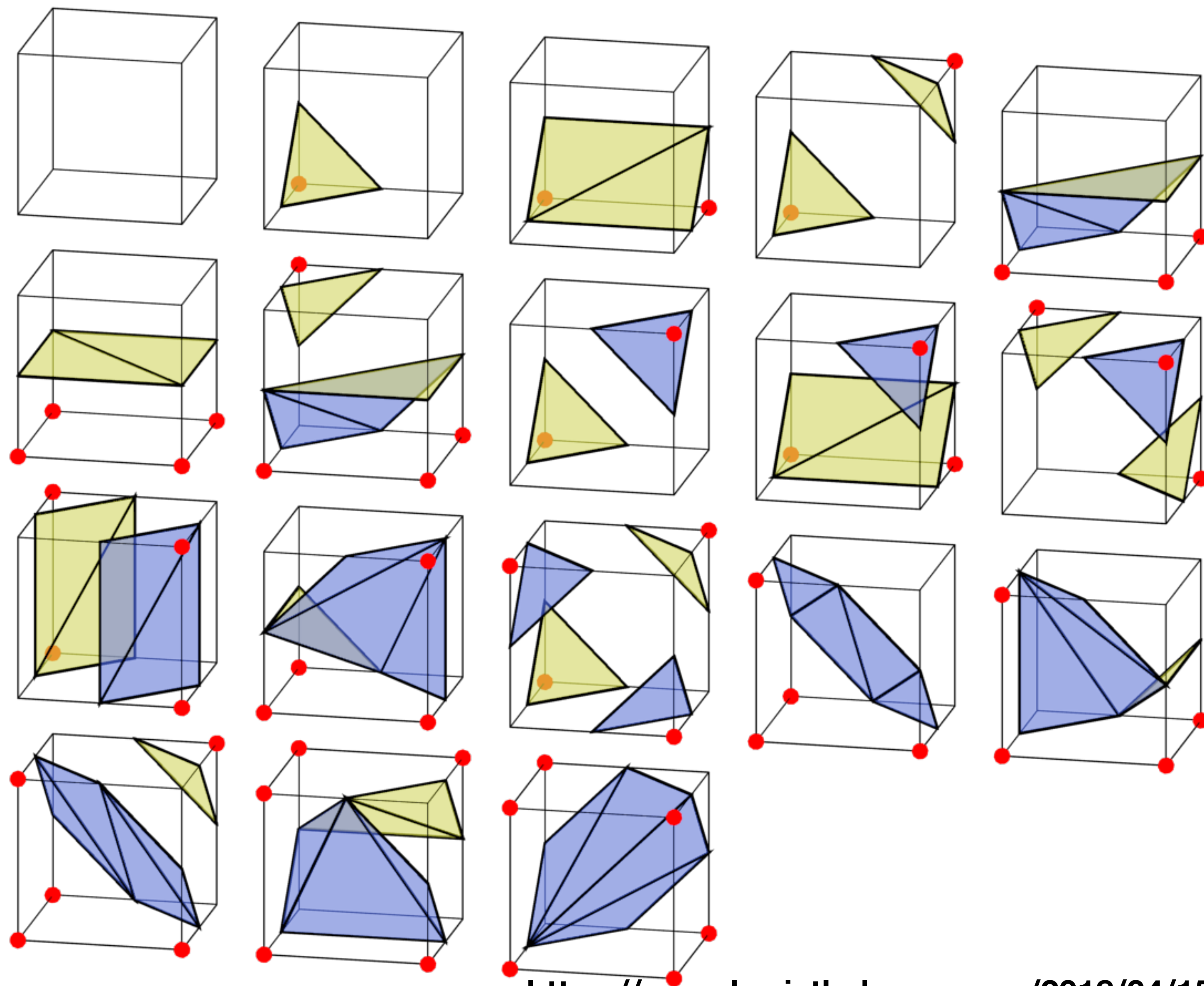
Reverse process (2D/3D)



Reverse process (2D/3D)

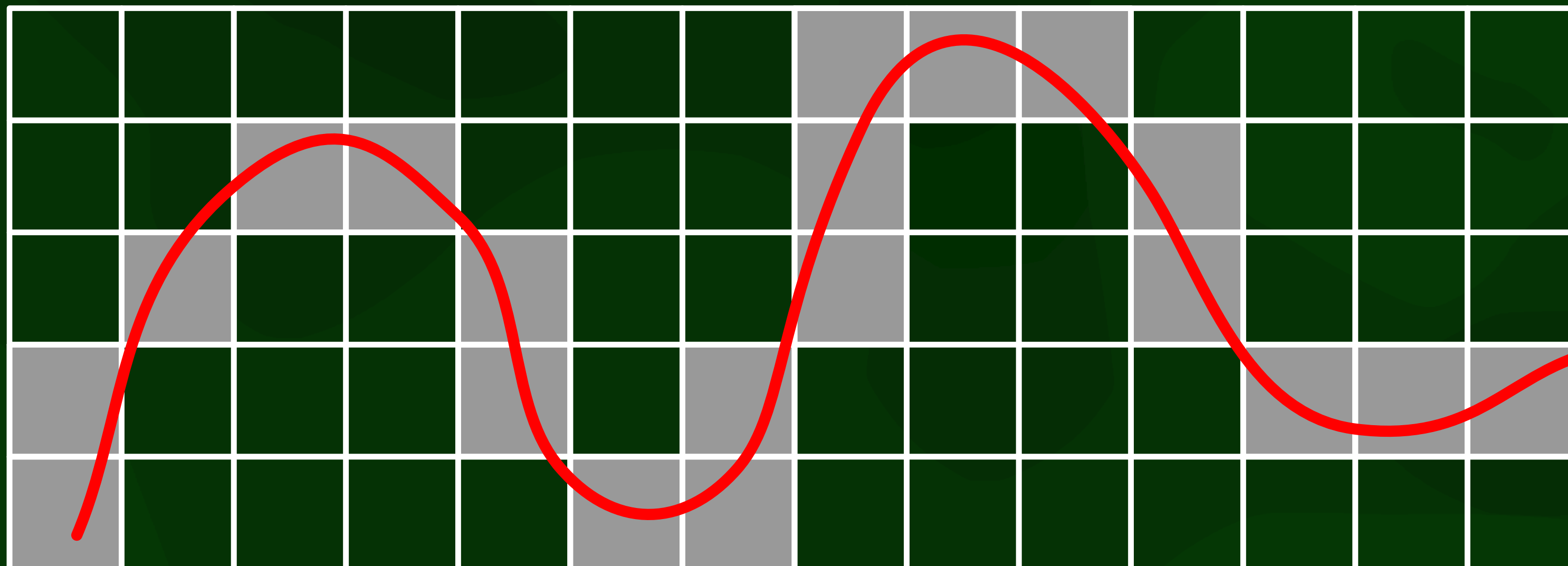
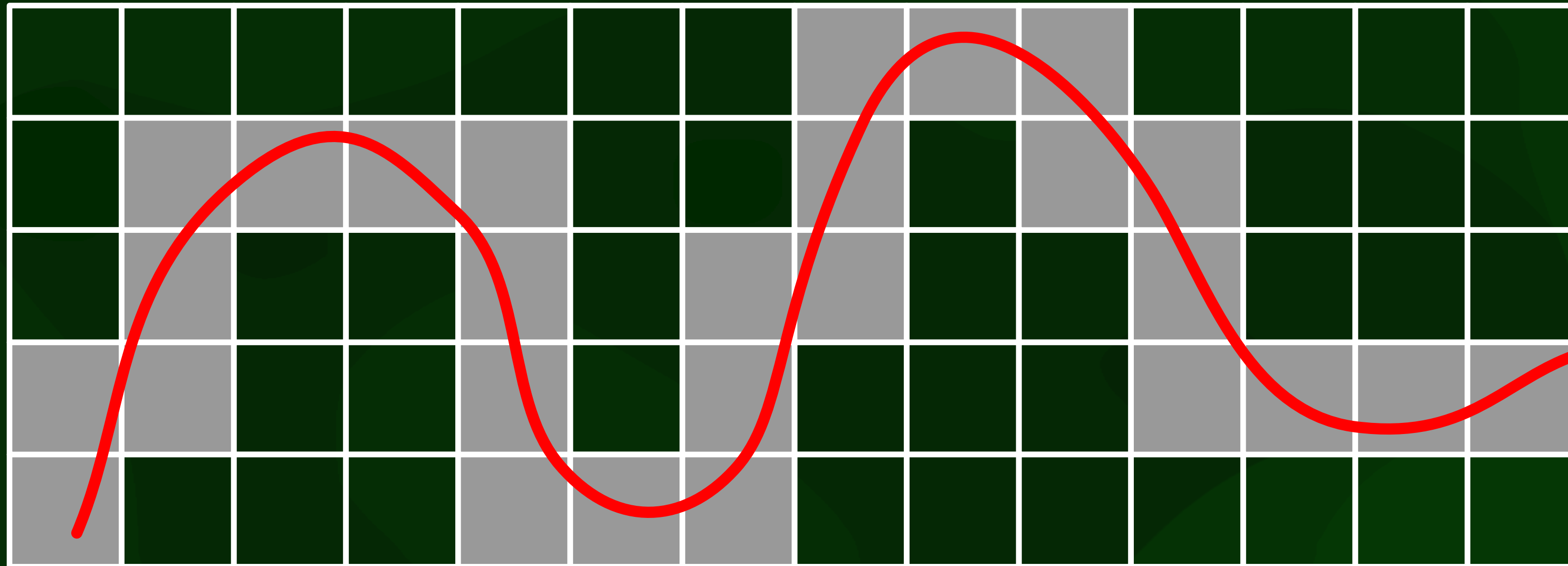


Marching cubes

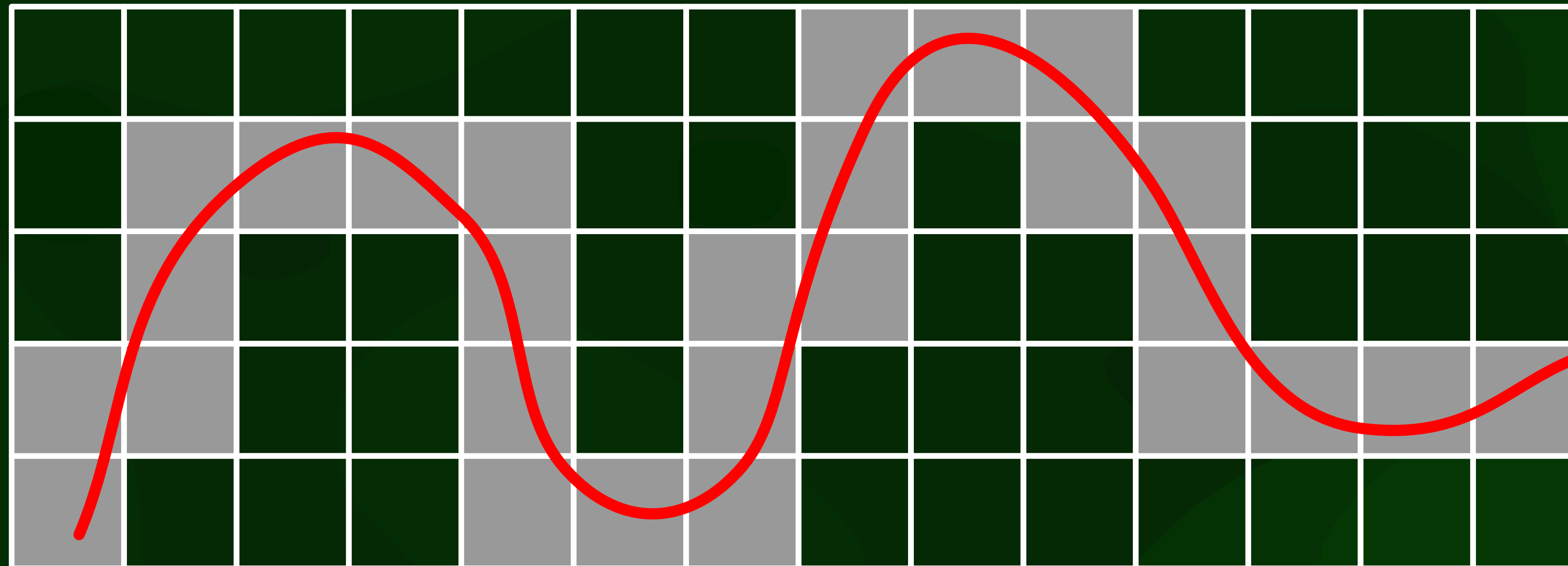


Voxelisation of objects

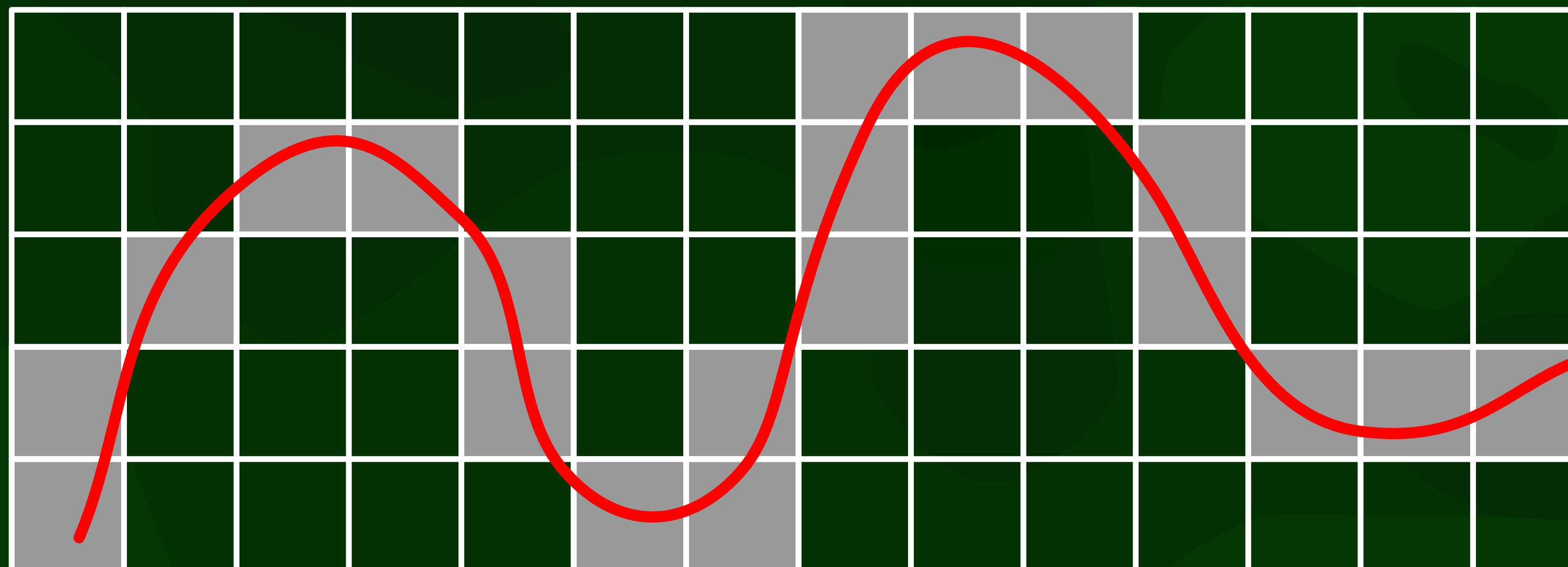
In 2D: what is the difference?



In 2D: what is the difference?

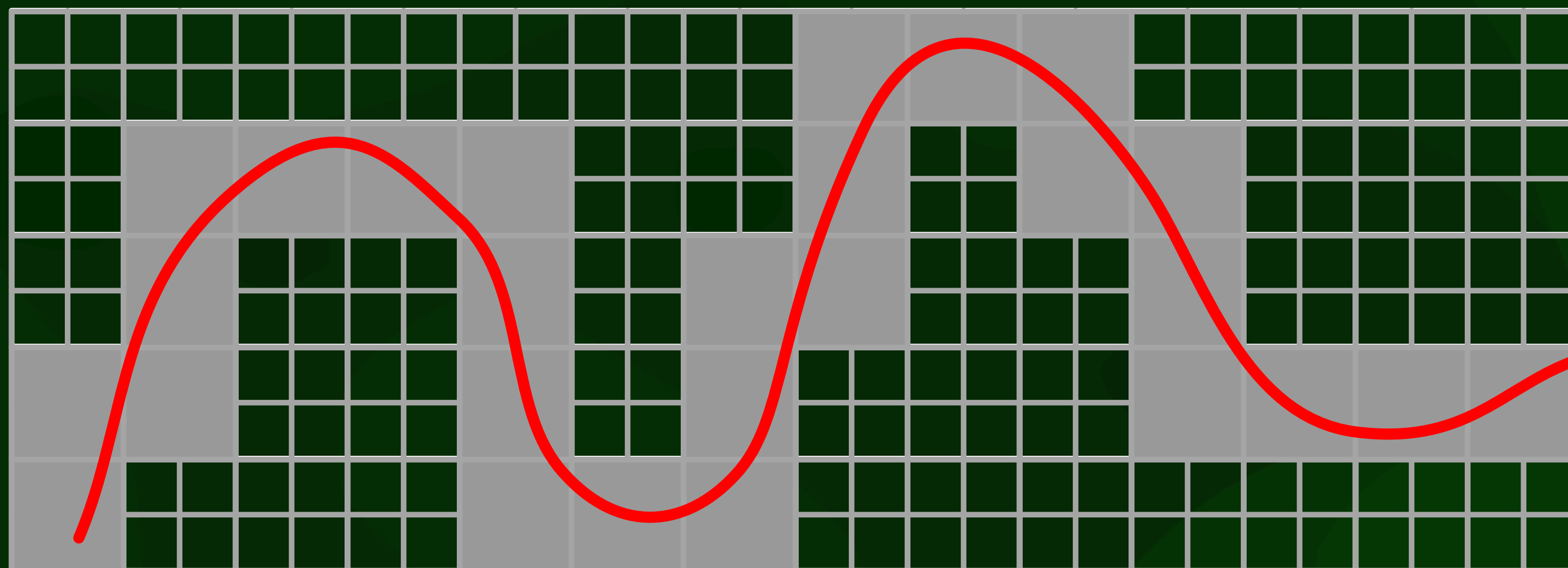


4-connected

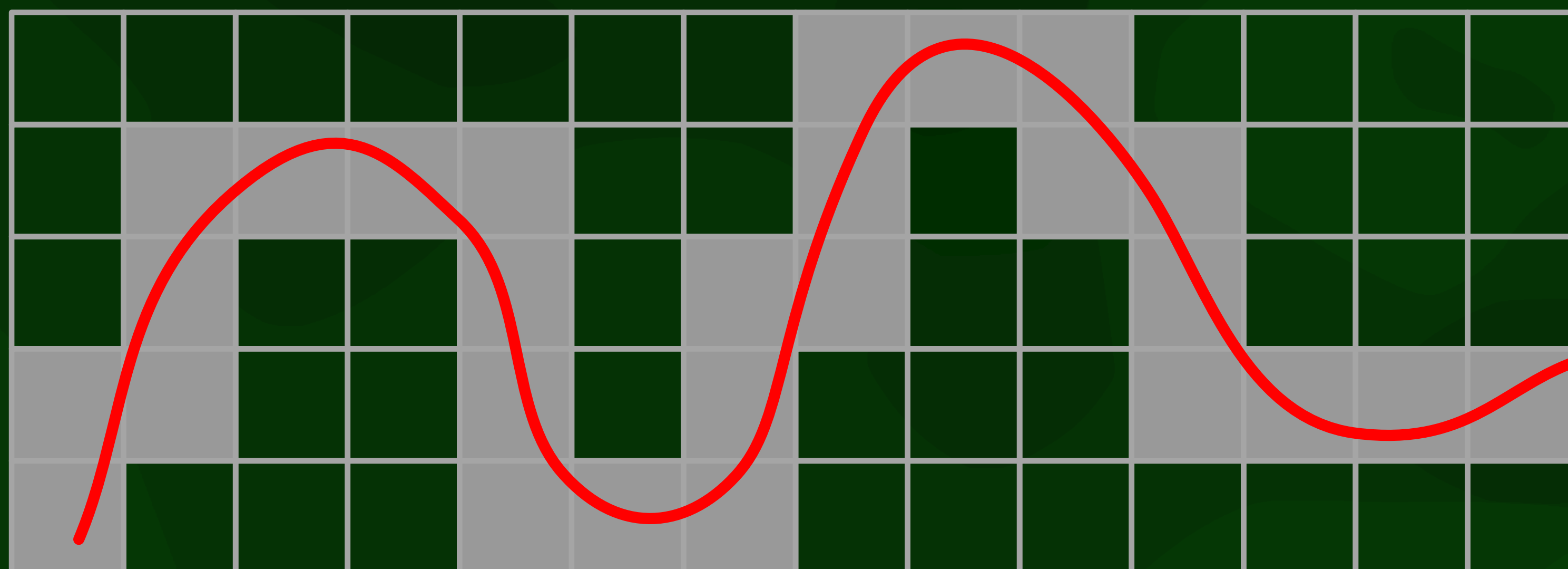


8-connected

2D targets

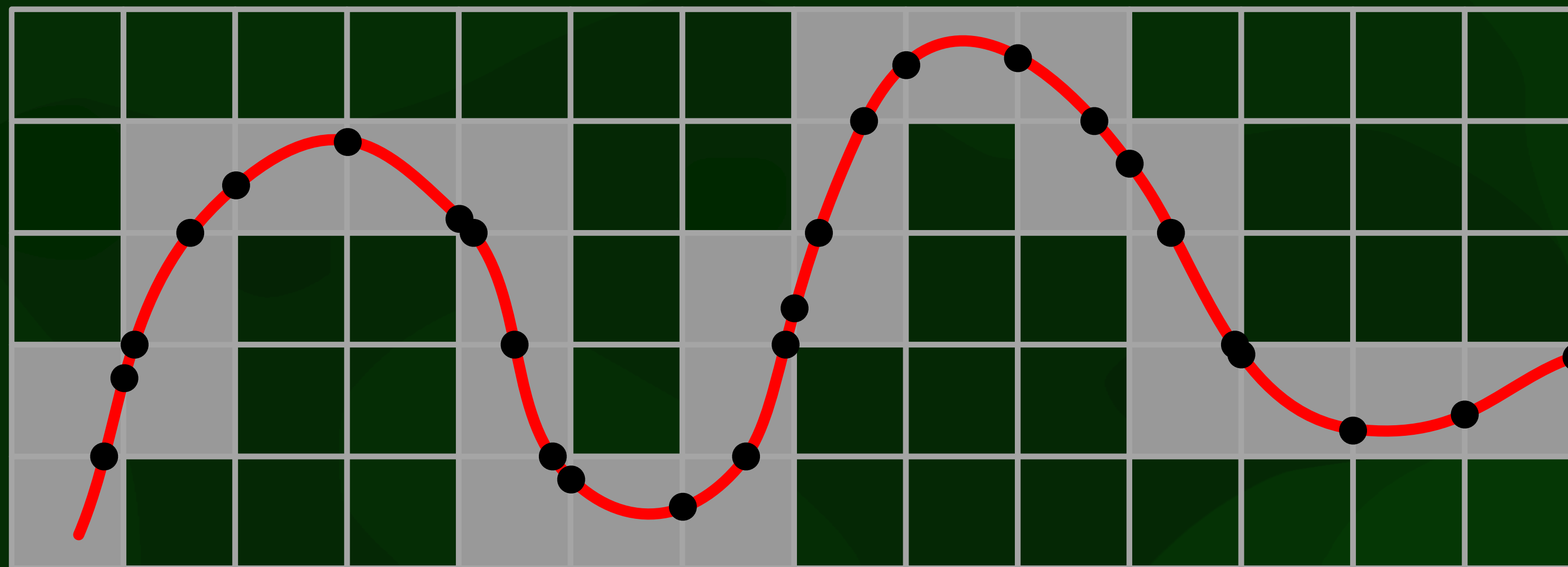


4-connectivity

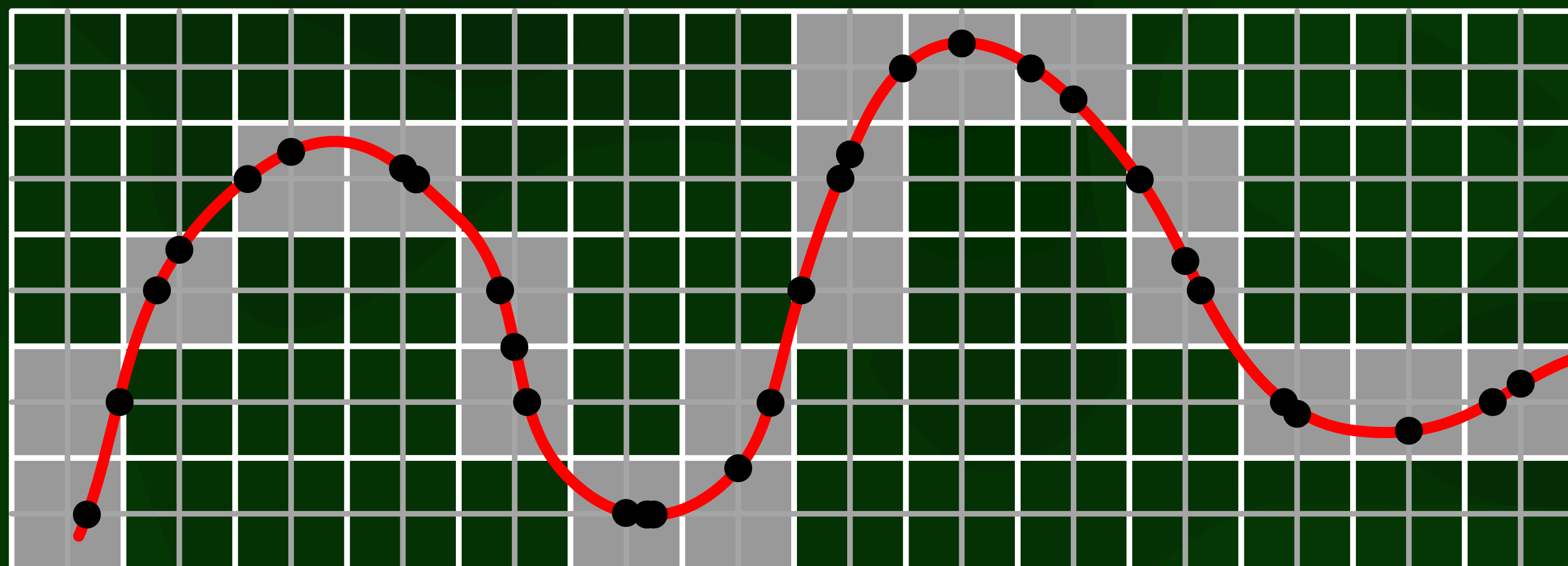


8-connectivity

2D targets

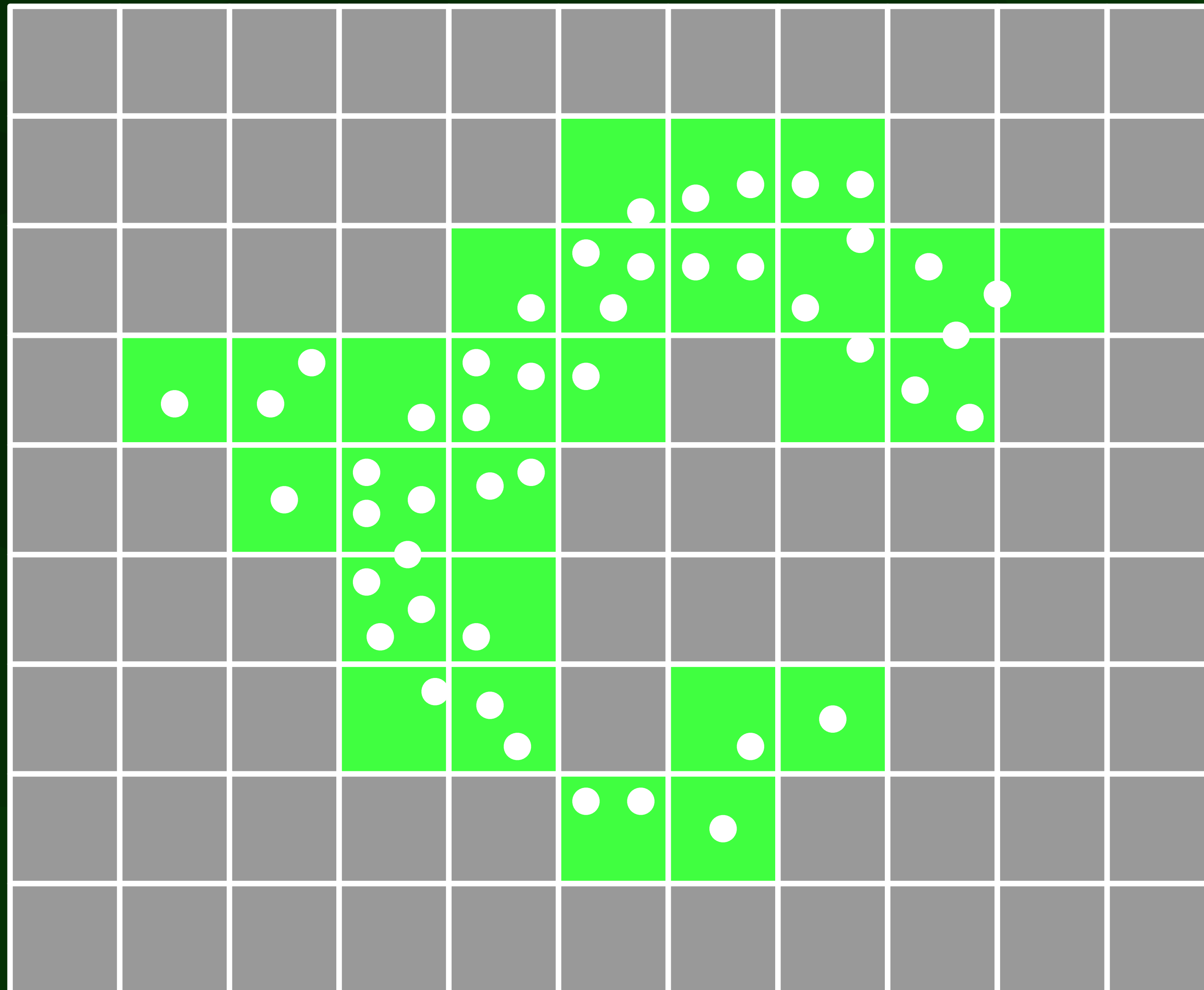


4-connectivity

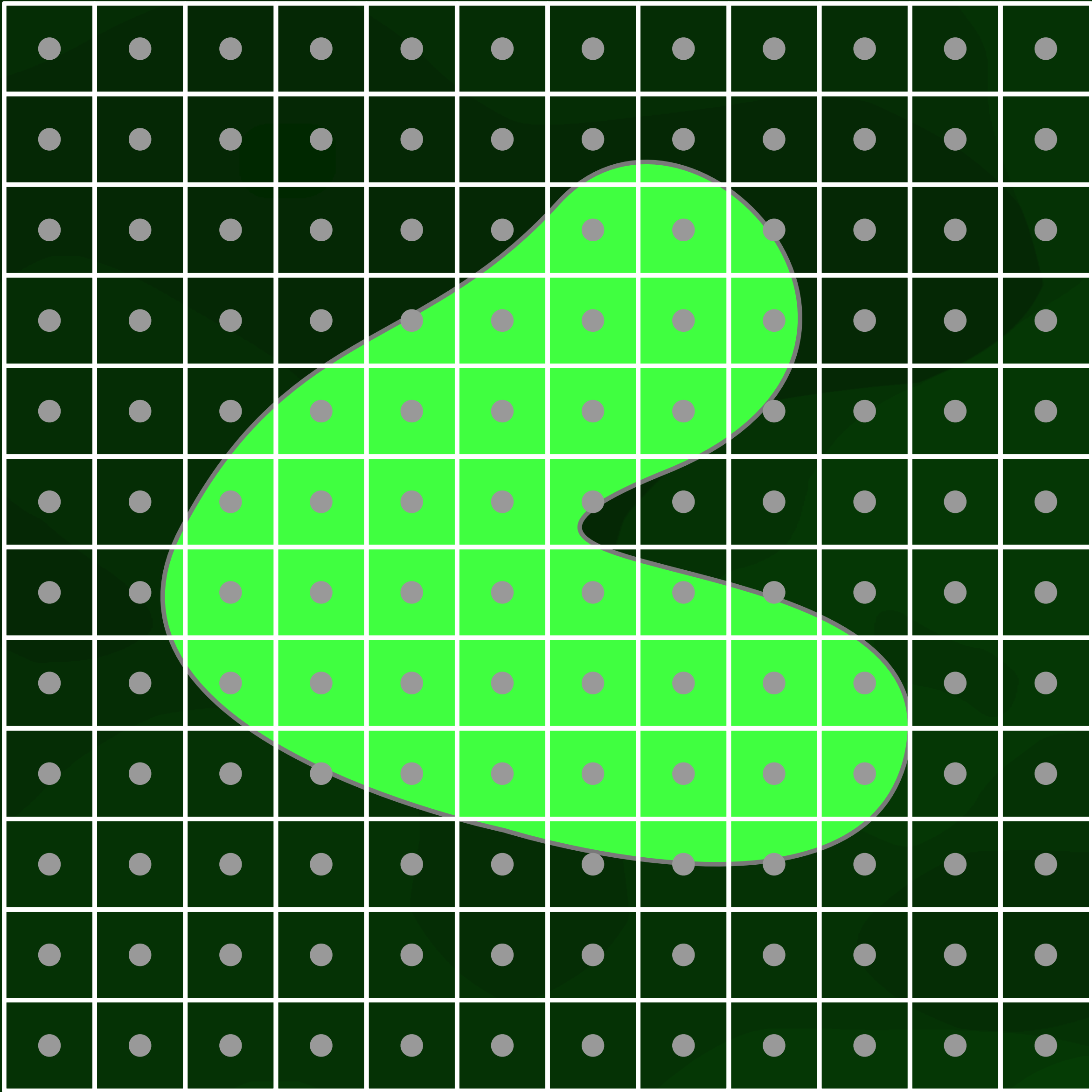


8-connectivity

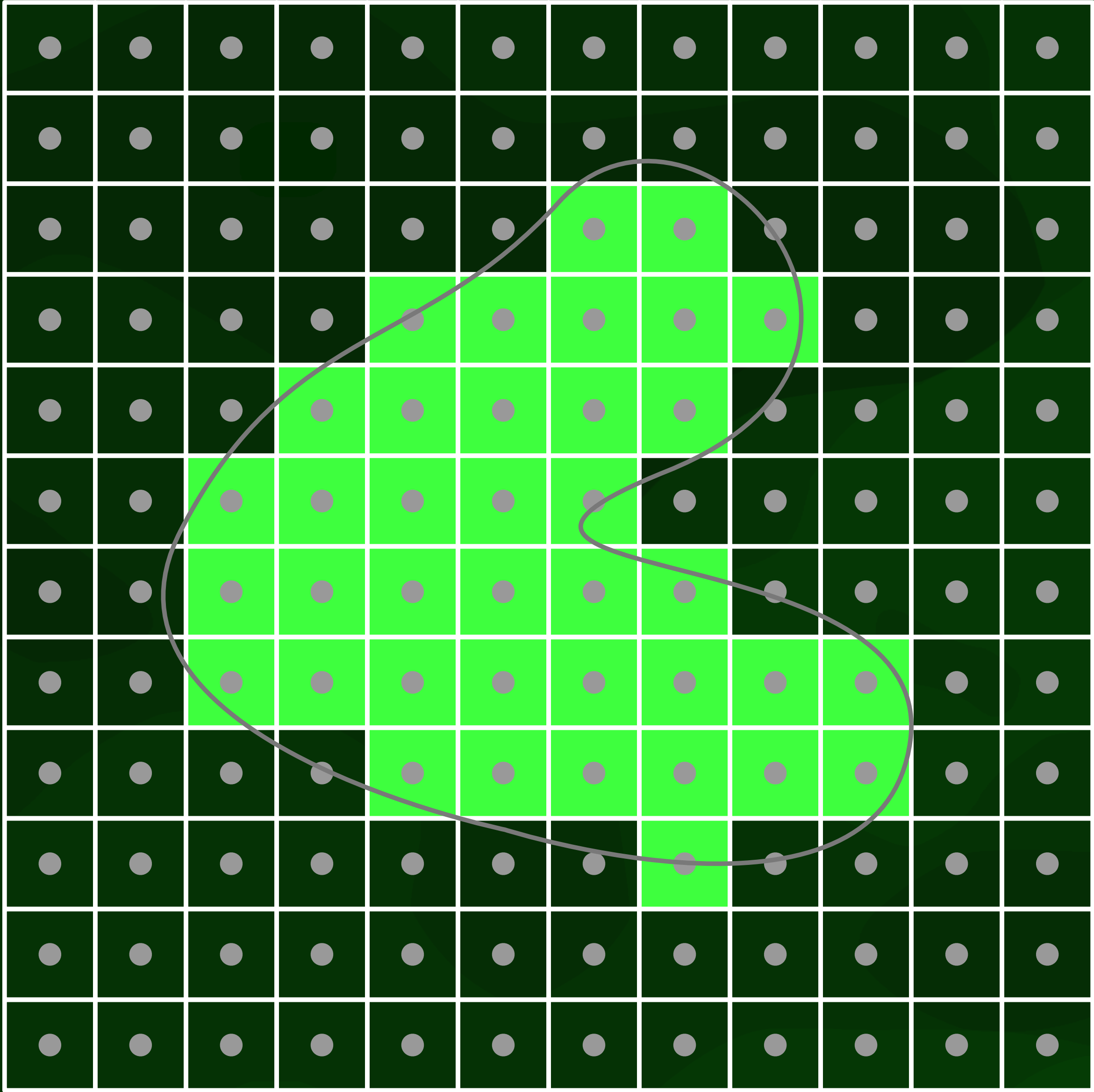
Targets for points



Targets for areas



Targets for areas



Duality of targets in 2D

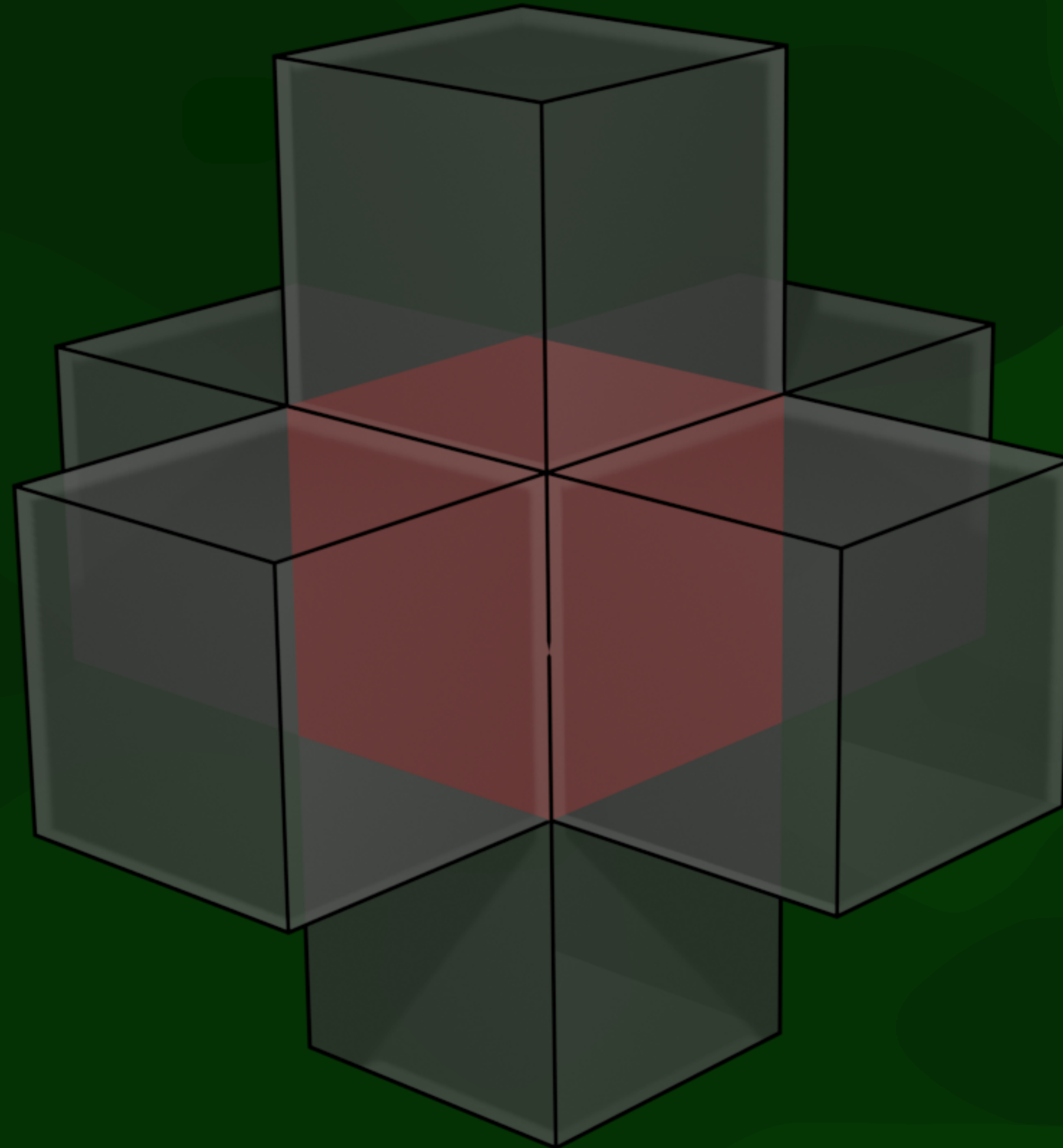
- To detect:
 - points (0D) $\rightarrow$ squares (2D)
 - lines (1D) $\rightarrow$ line segments (1D)
 - areas (2D) $\rightarrow$ centre points (0D)

Duality of targets in 3D

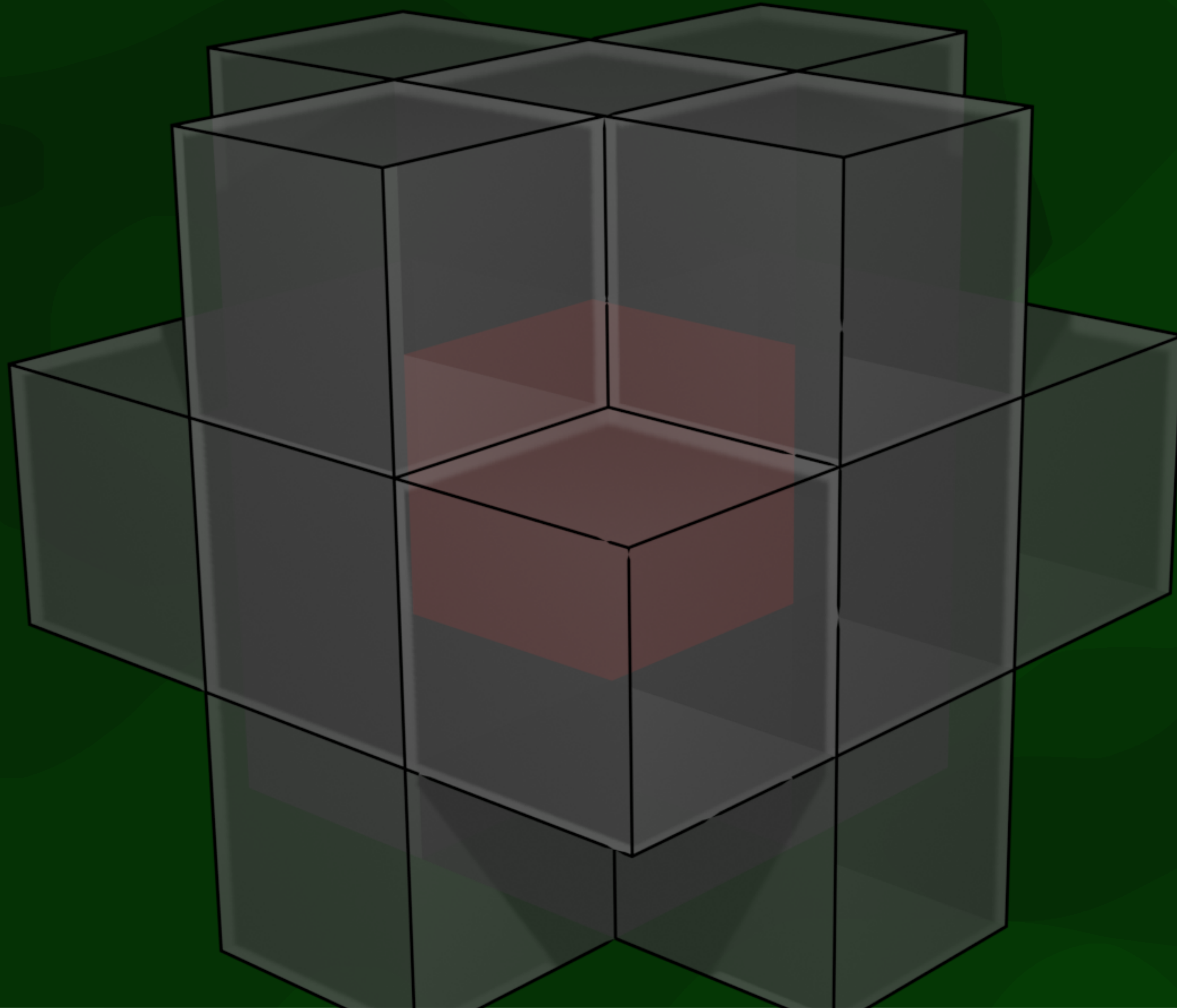
- To detect:
 - points (0D) $\rightarrow$ cubes (3D)
 - lines (1D) $\rightarrow$ surfaces (2D)
 - surfaces (2D) $\rightarrow$ line segments (1D)
 - volumes (3D) $\rightarrow$ centre points (0D)

Connectivity in 3D

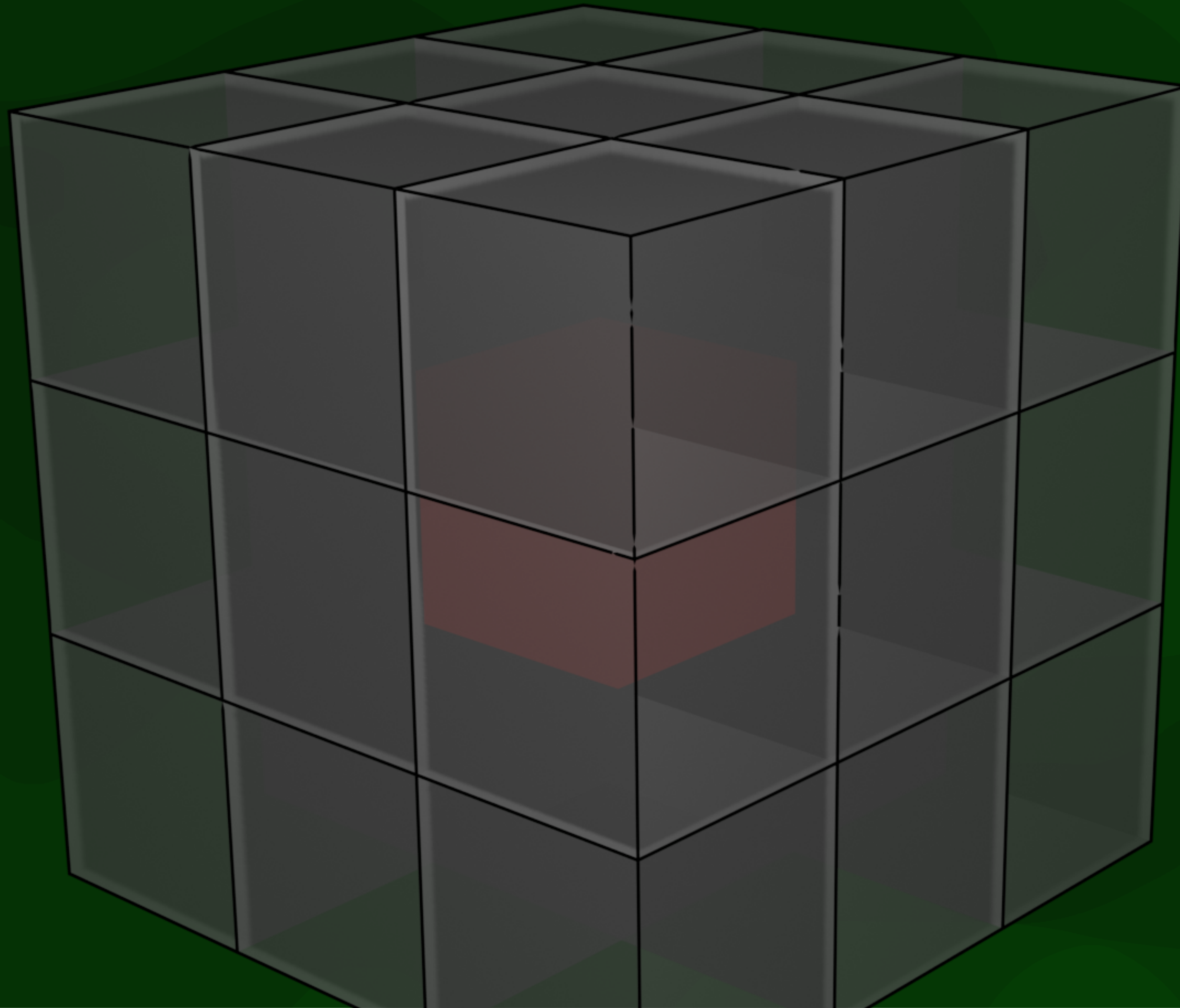
6-connectivity



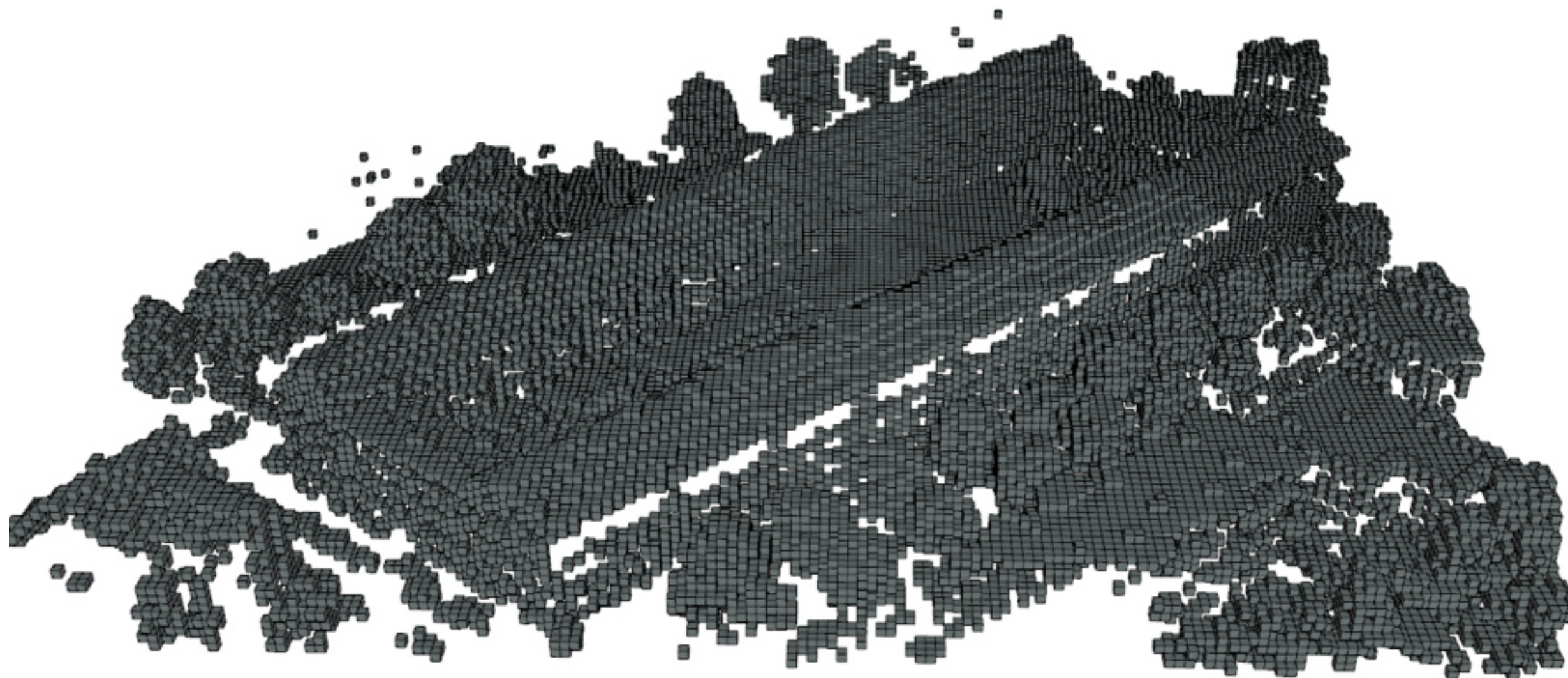
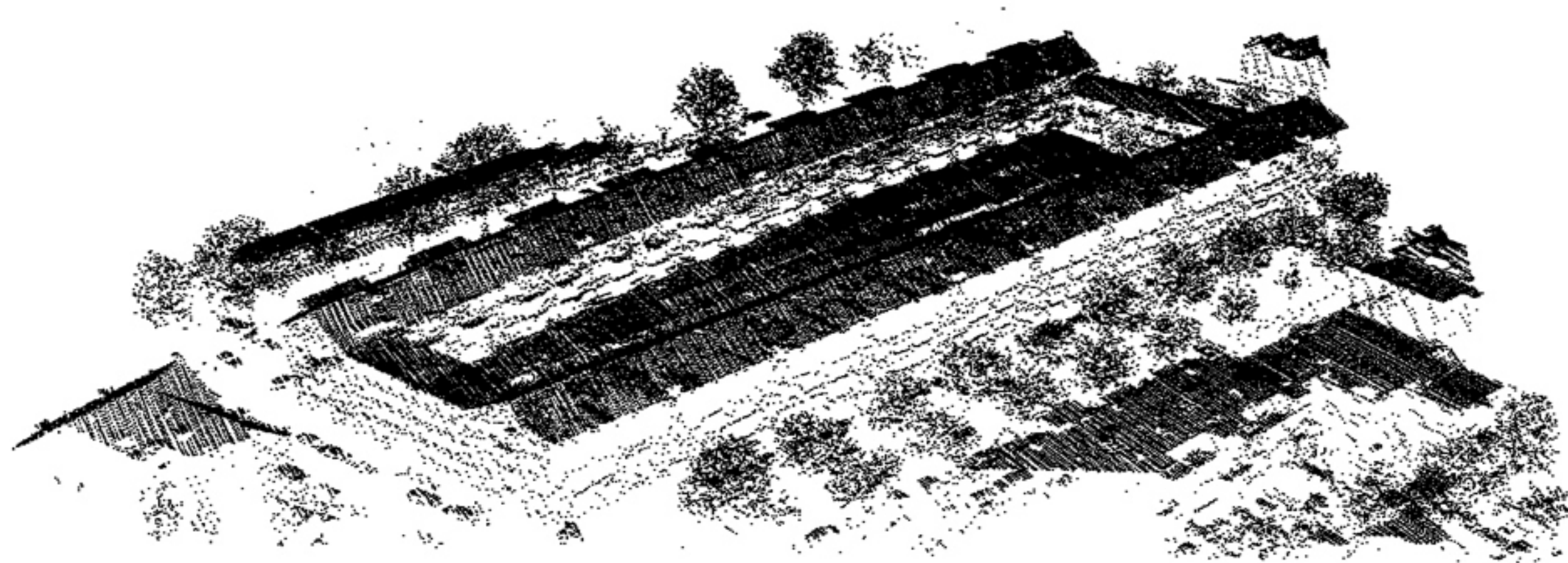
18-connectivity



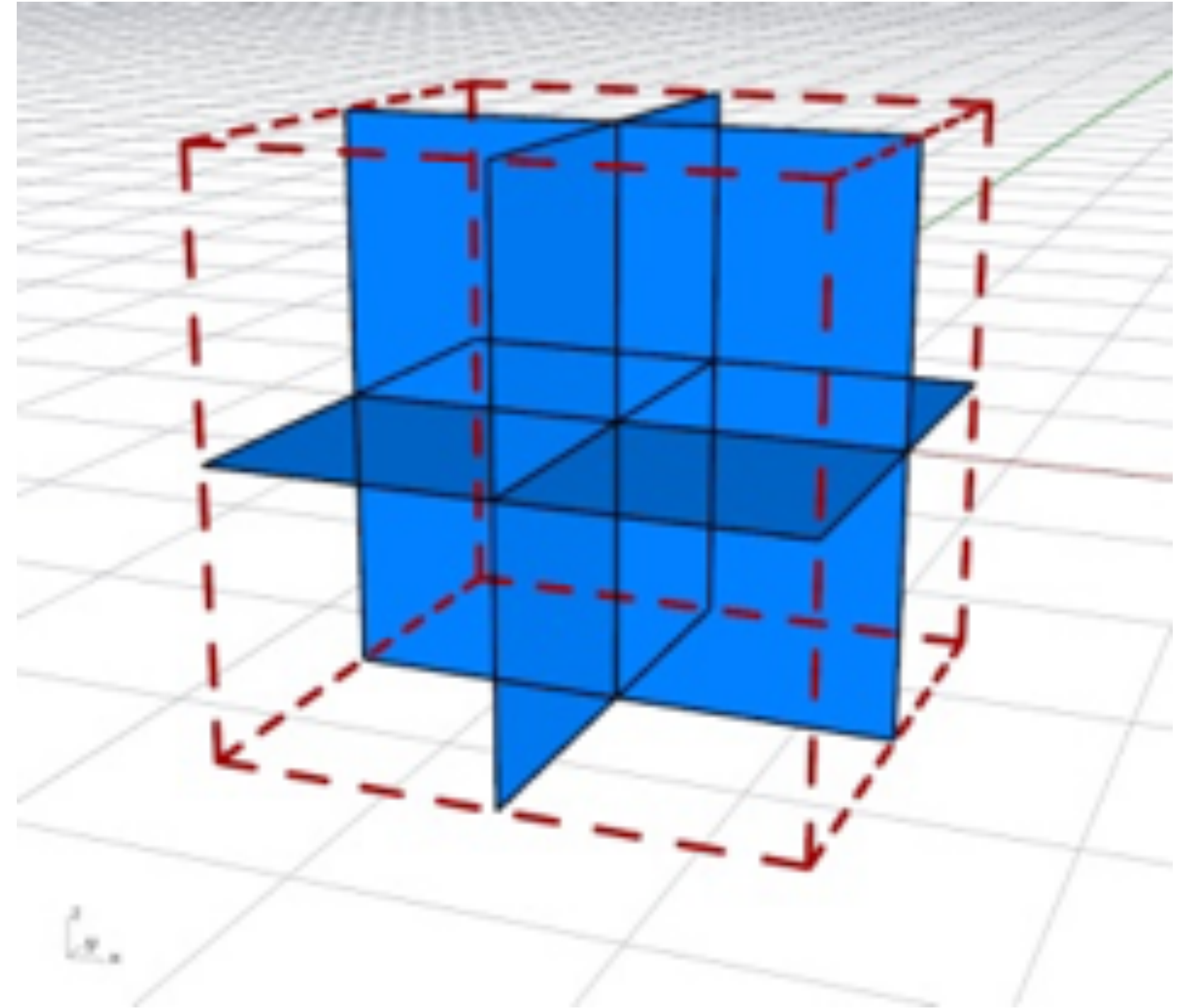
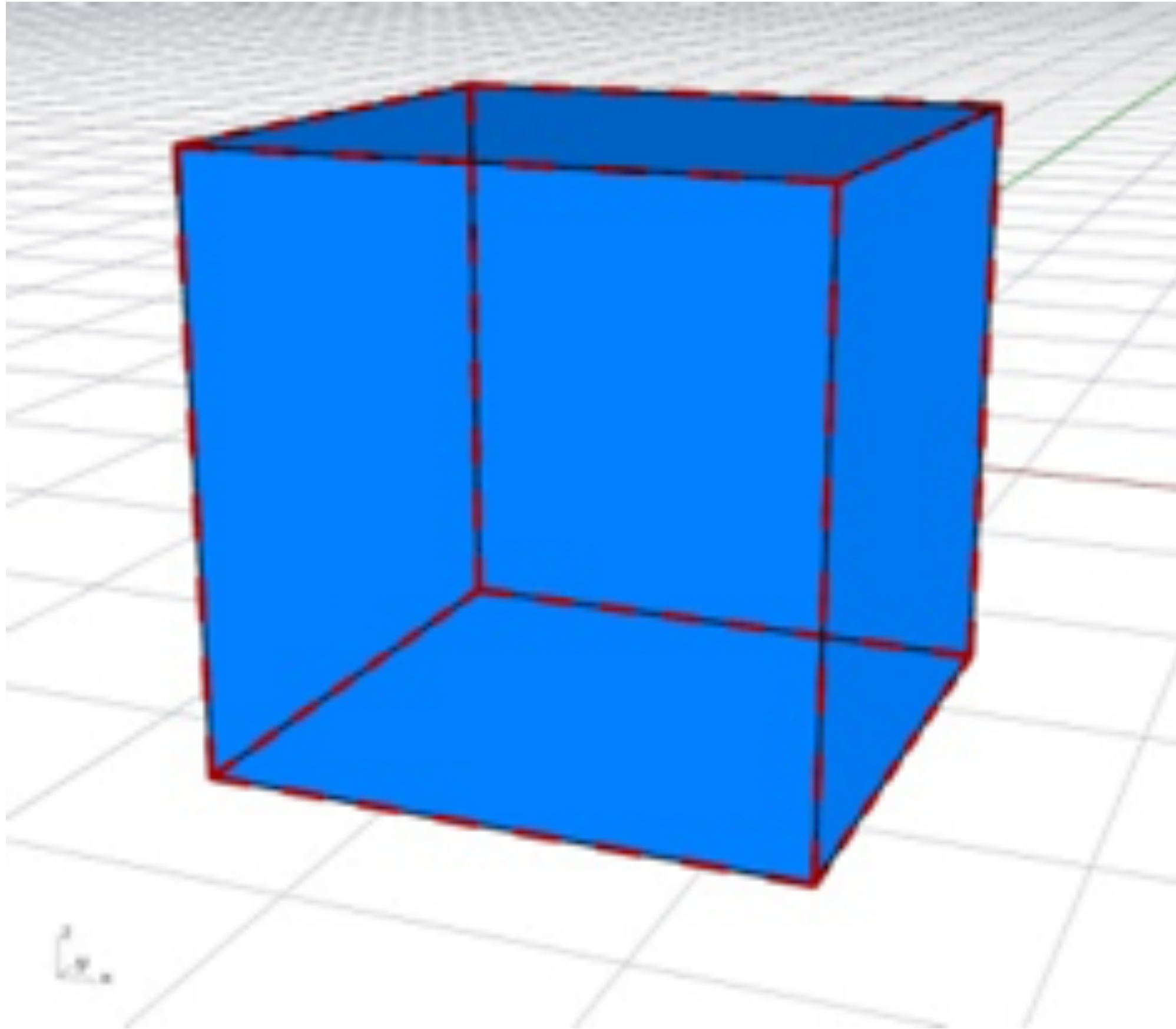
26-connectivity



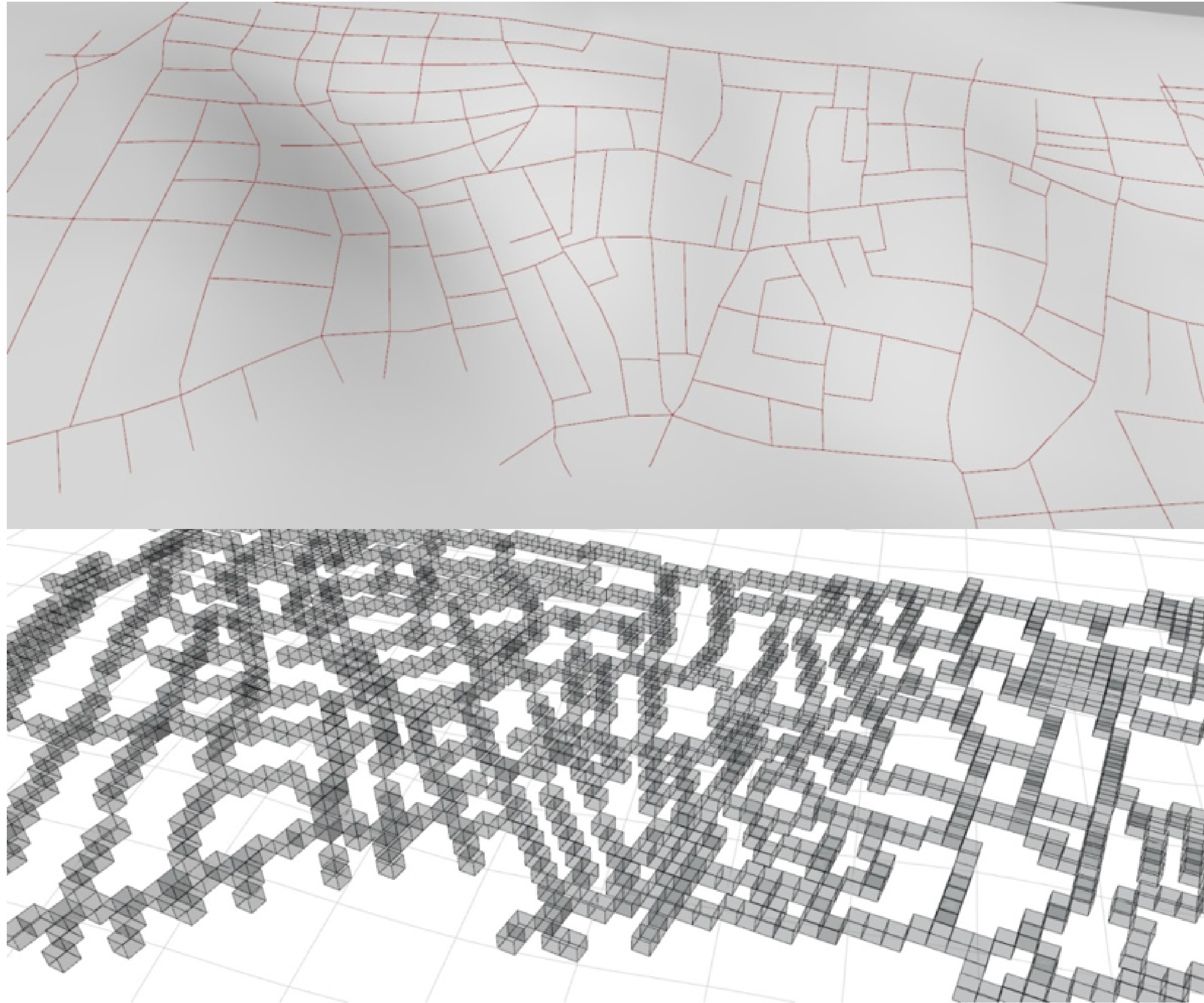
Voxelising points



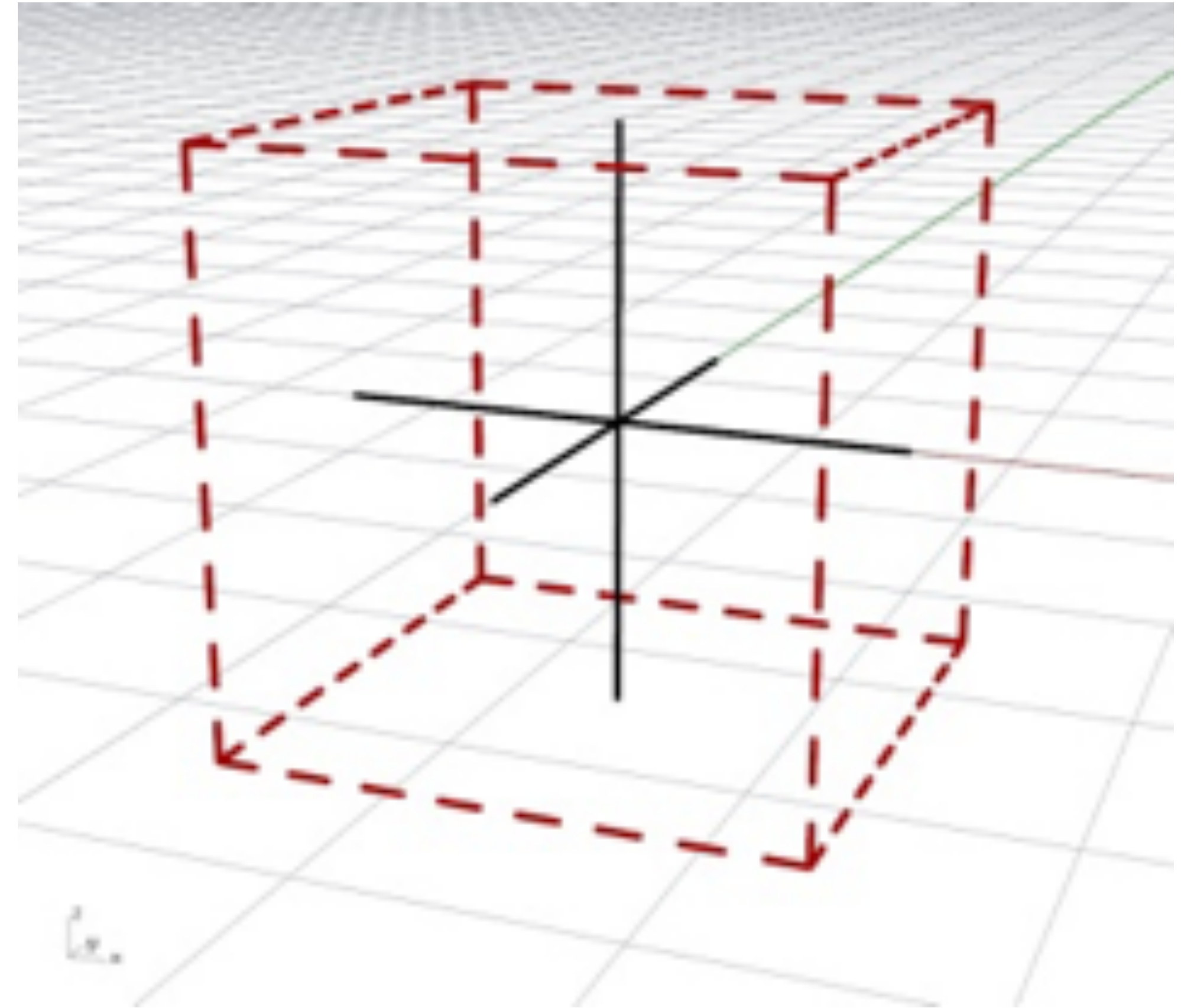
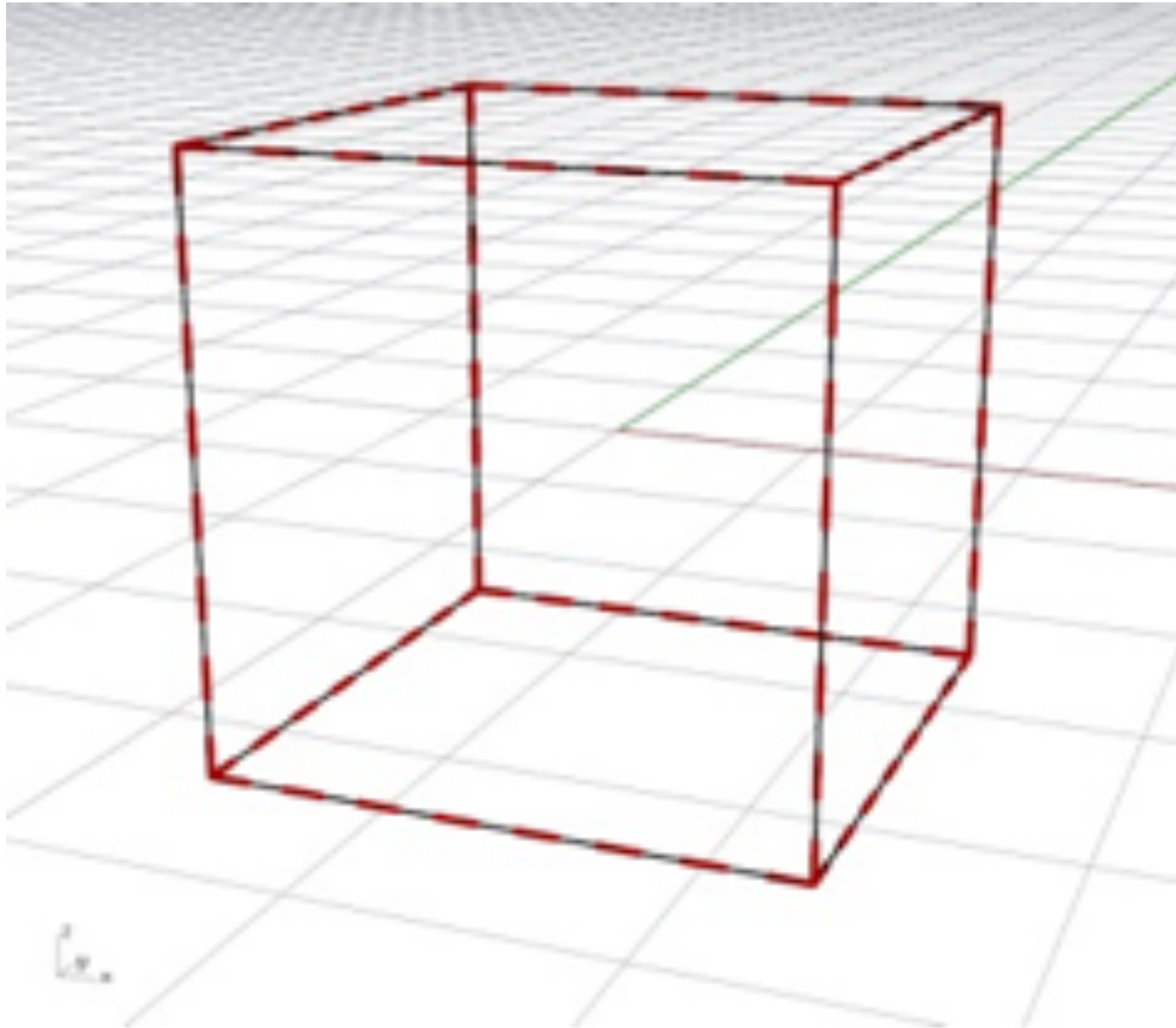
Voxelising lines



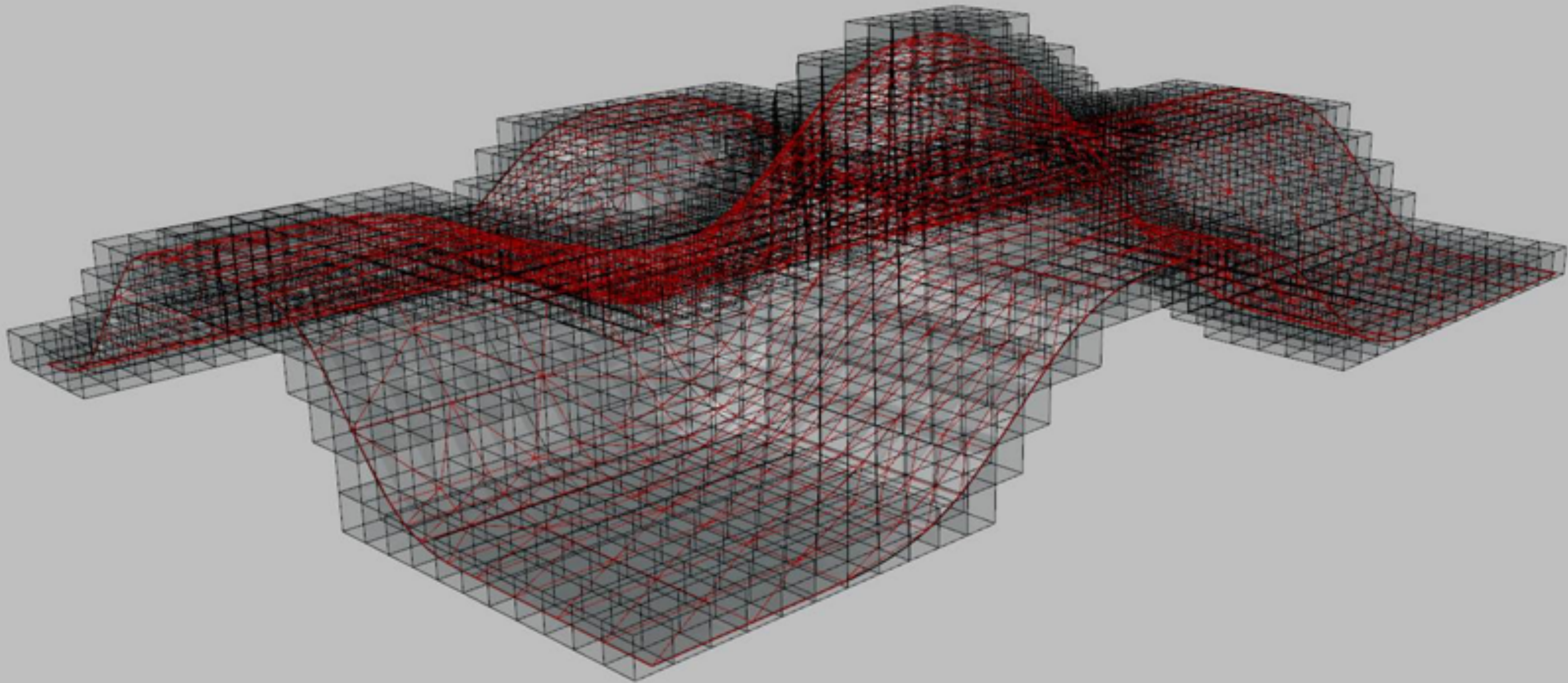
Voxelising lines



Voxelising surfaces



Voxelising surfaces



Voxelising volumes?

Voxelising volumes?

Assuming b-rep:

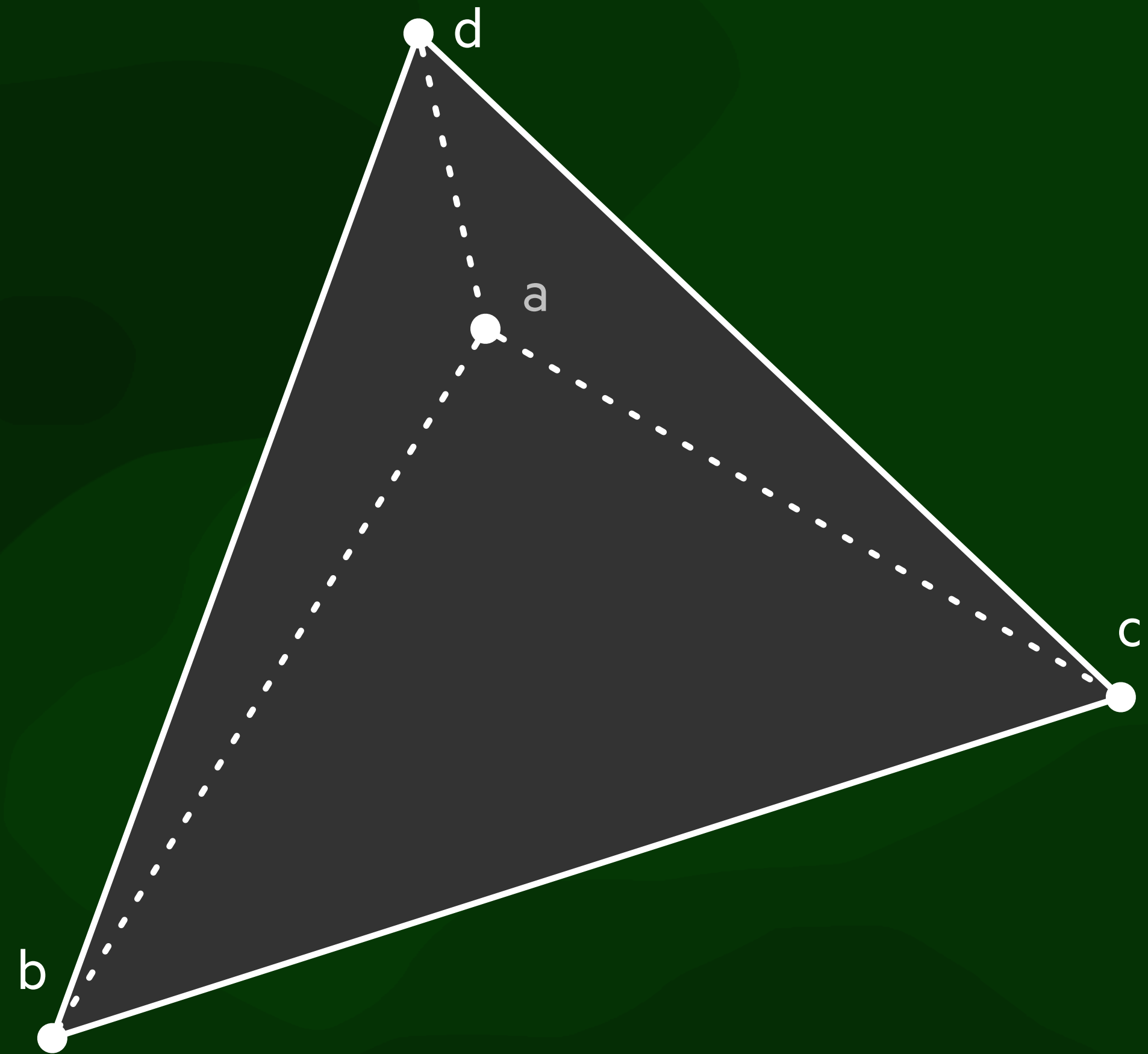
1. Voxelise surface
2. Fill interior

Is this accurate?

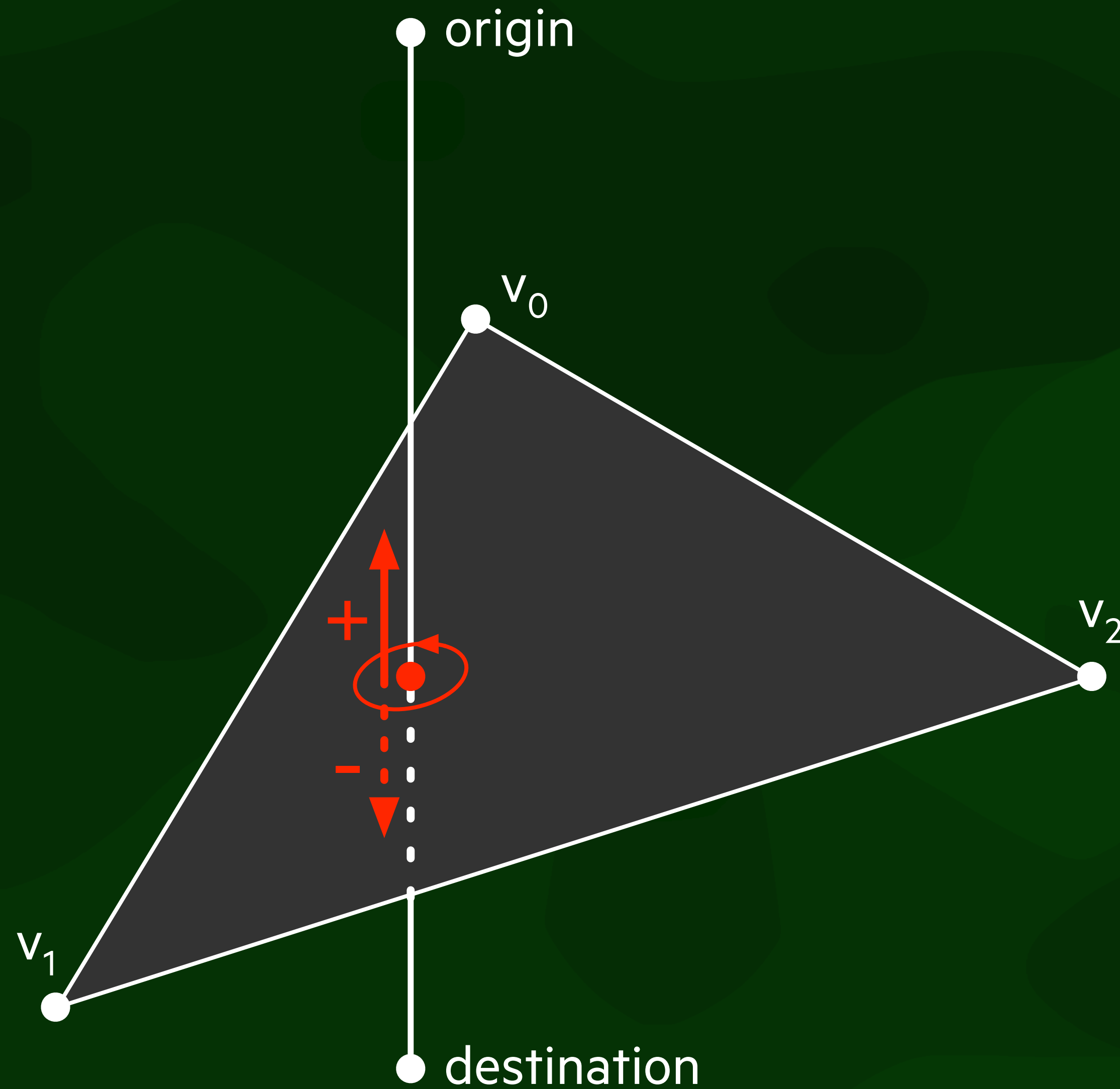
In detail: voxelising lines / surfaces

- Same case!
- Can be computed as line segment - triangle intersection
- Possible with max 4 orientation tests, e.g. with signed volume:

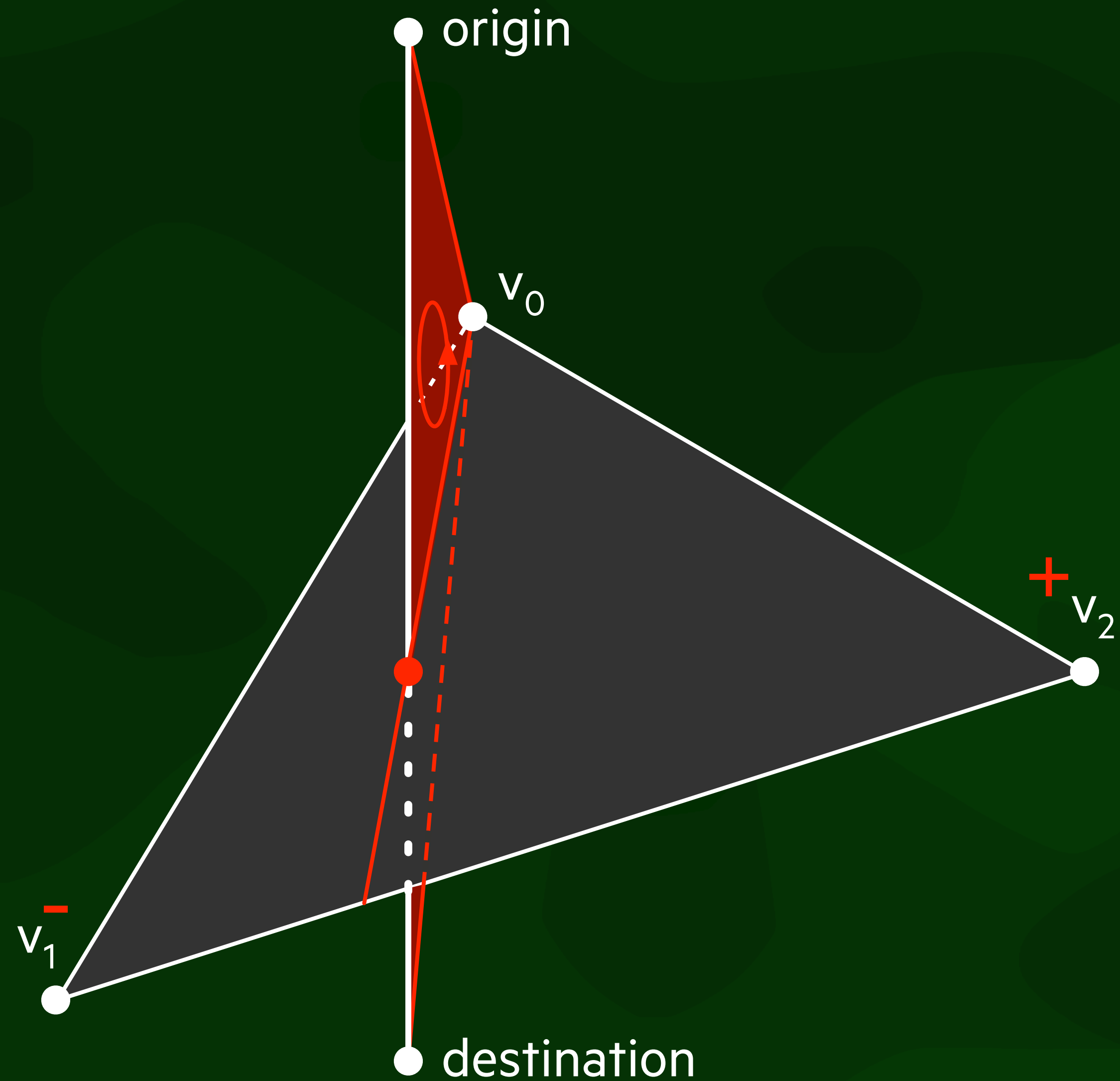
$$\frac{(\overrightarrow{a-d})(\overrightarrow{b-d}) \times (\overrightarrow{c-d})}{6}$$



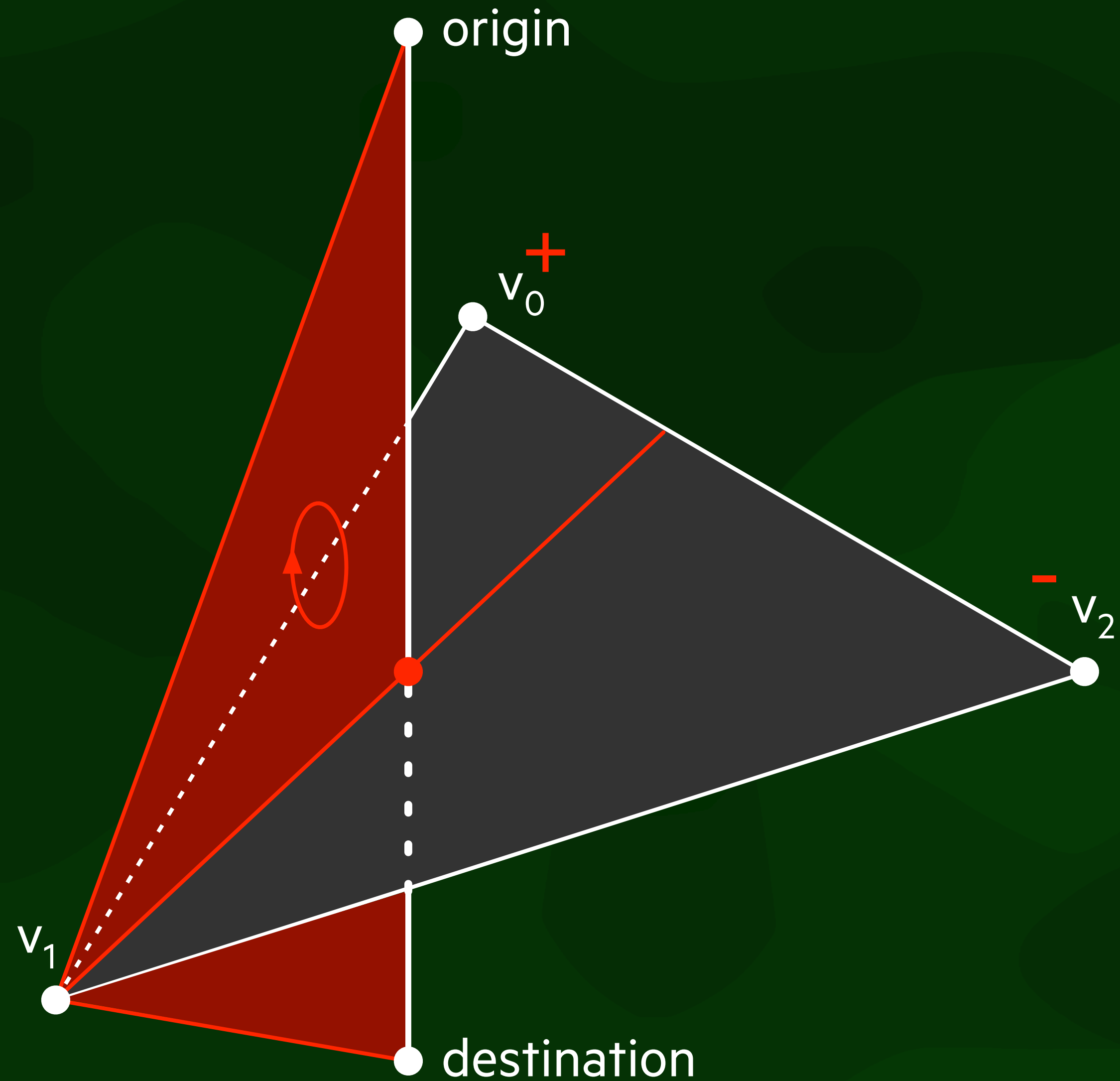
Voxelising lines / surfaces



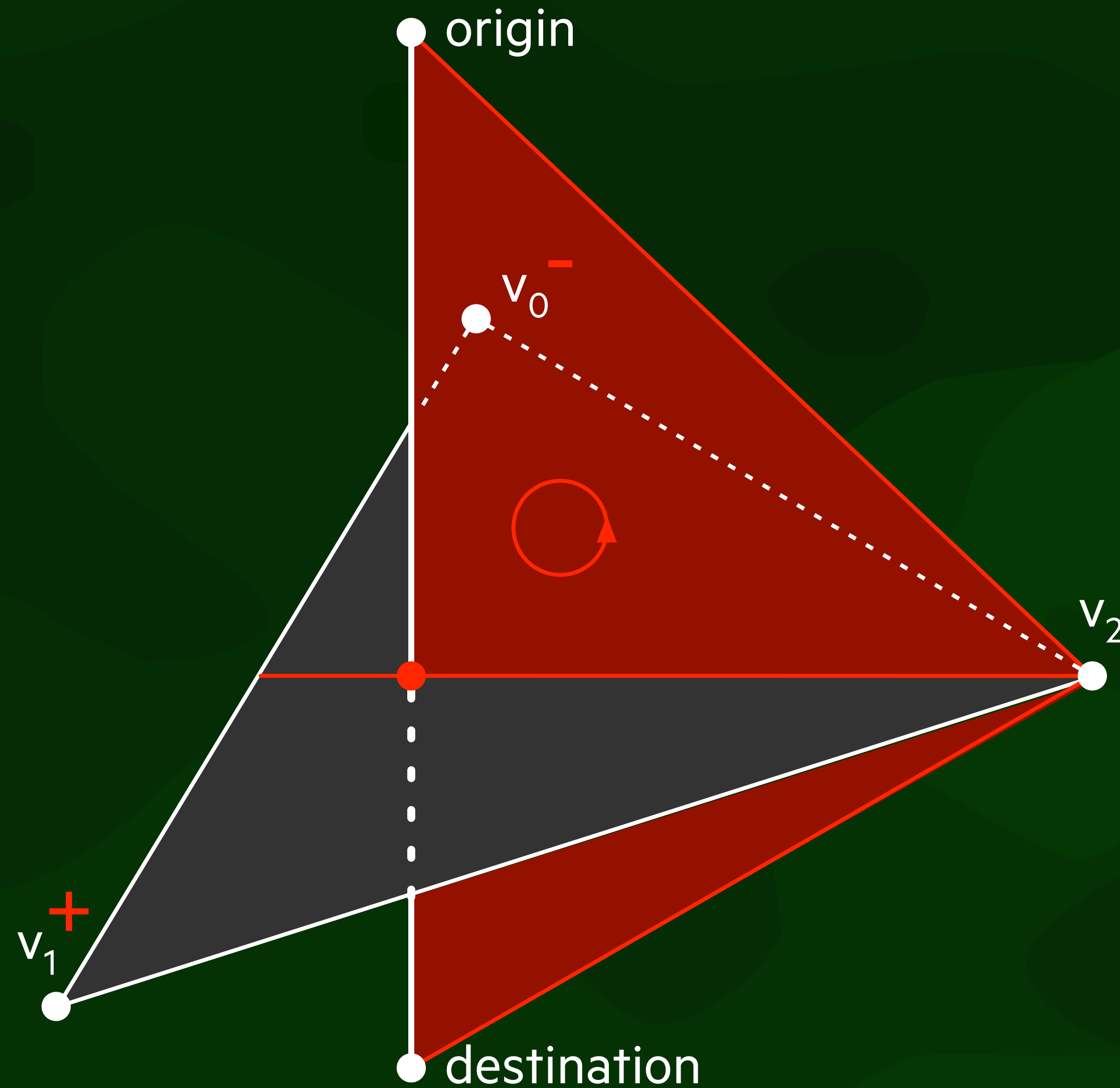
Voxelising lines / surfaces



Voxelising lines / surfaces



Voxelising lines / surfaces



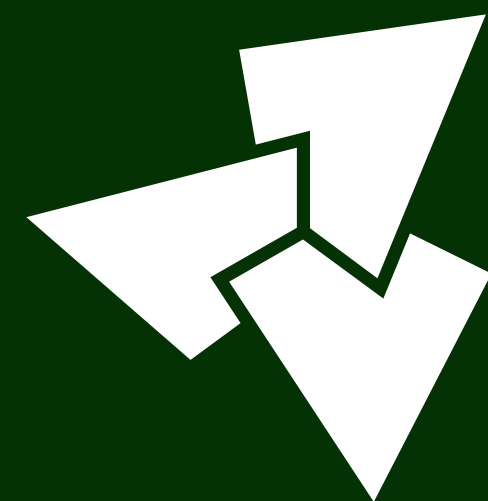
What to do next?

1. Today:

- Continue with Homework 1 (due May 8)
- Go to geo1004 website and study today's lesson (**updated** 3D book Chapter 4)
- If you have extra time: study Wednesday's lesson (3D book Chapter 9)

2. Wednesday: val3dity demos, then help with lessons or Hw 1

<https://3d.bk.tudelft.nl/courses/geo1004>



3D geoinformation

Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

References

- [8]: Stafleu, Jan, et al. "Geological subsurface models of the Netherlands." *Geology of the Netherlands: Second Edition*. Amsterdam University Press (2025): 849-894.
- [9]: Tian, Xiaoyu, et al. "Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving." *Advances in Neural Information Processing Systems* 36 (2023): 64318-64330.
- [10]: Vantyghem, Gieljan, Ticho Ooms, and Wouter De Corte. "VoxelPrint: A Grasshopper plug-in for voxel-based numerical simulation of concrete printing." *Automation in Construction* 122 (2021): 103469.