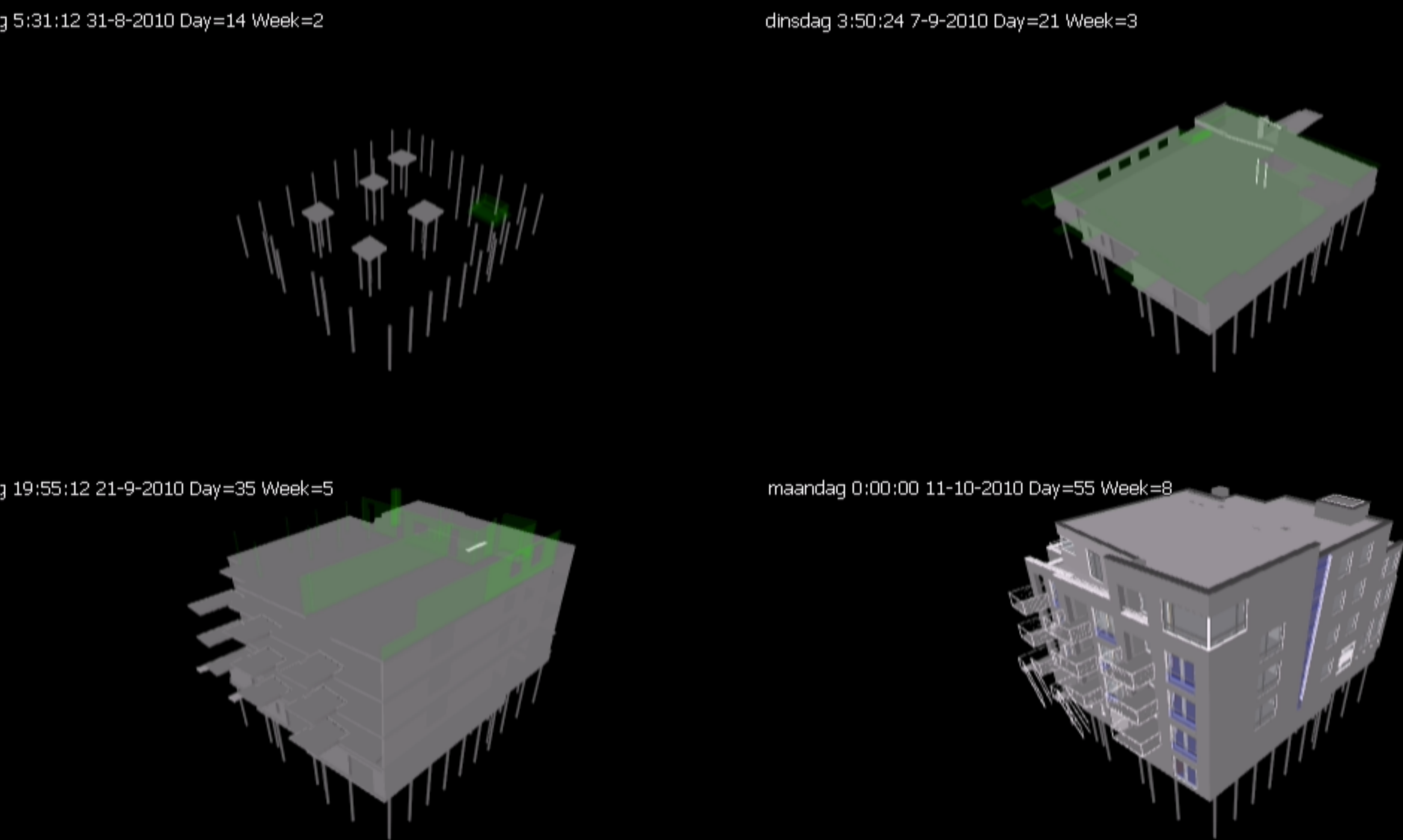
Constructing an *n*-dimensional cell complex from a soup of (*n*-1)-dimensional faces

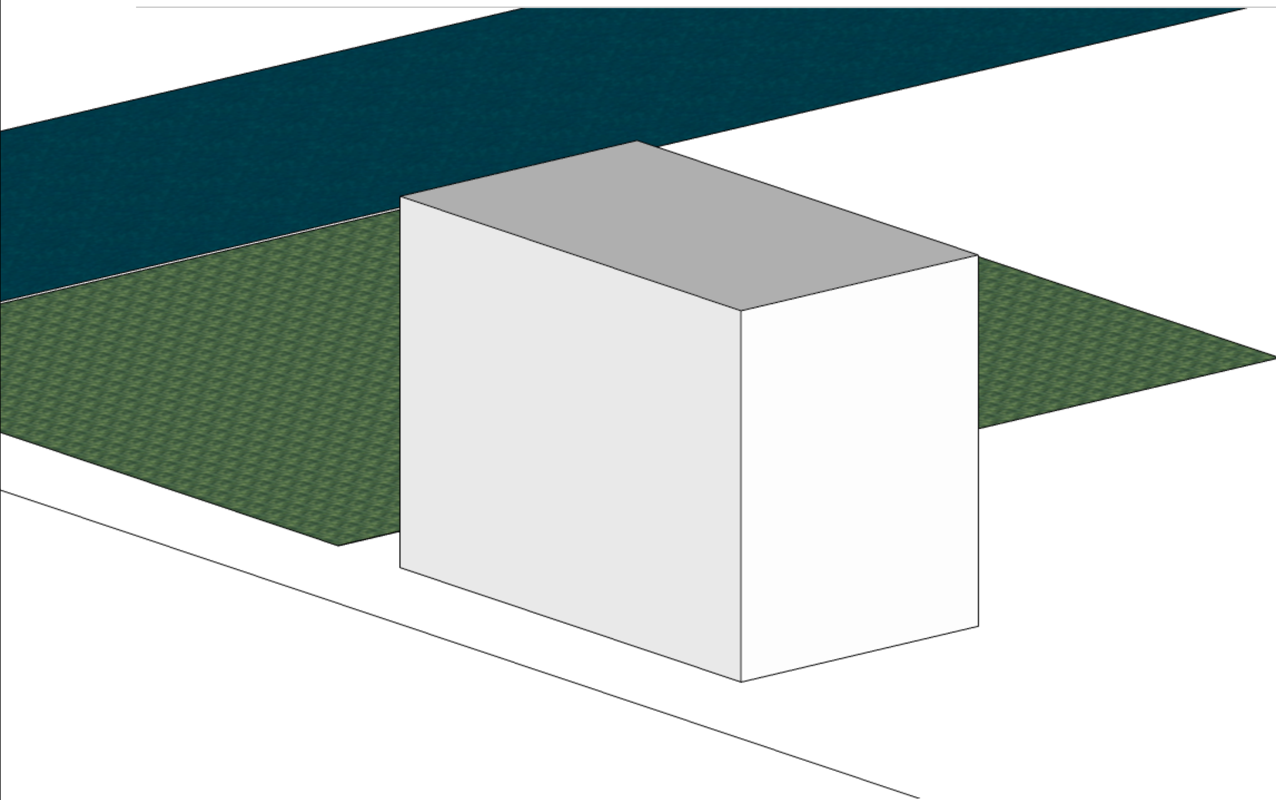
Ken Arroyo Ohori
Guillaume Damiand
Hugo Ledoux

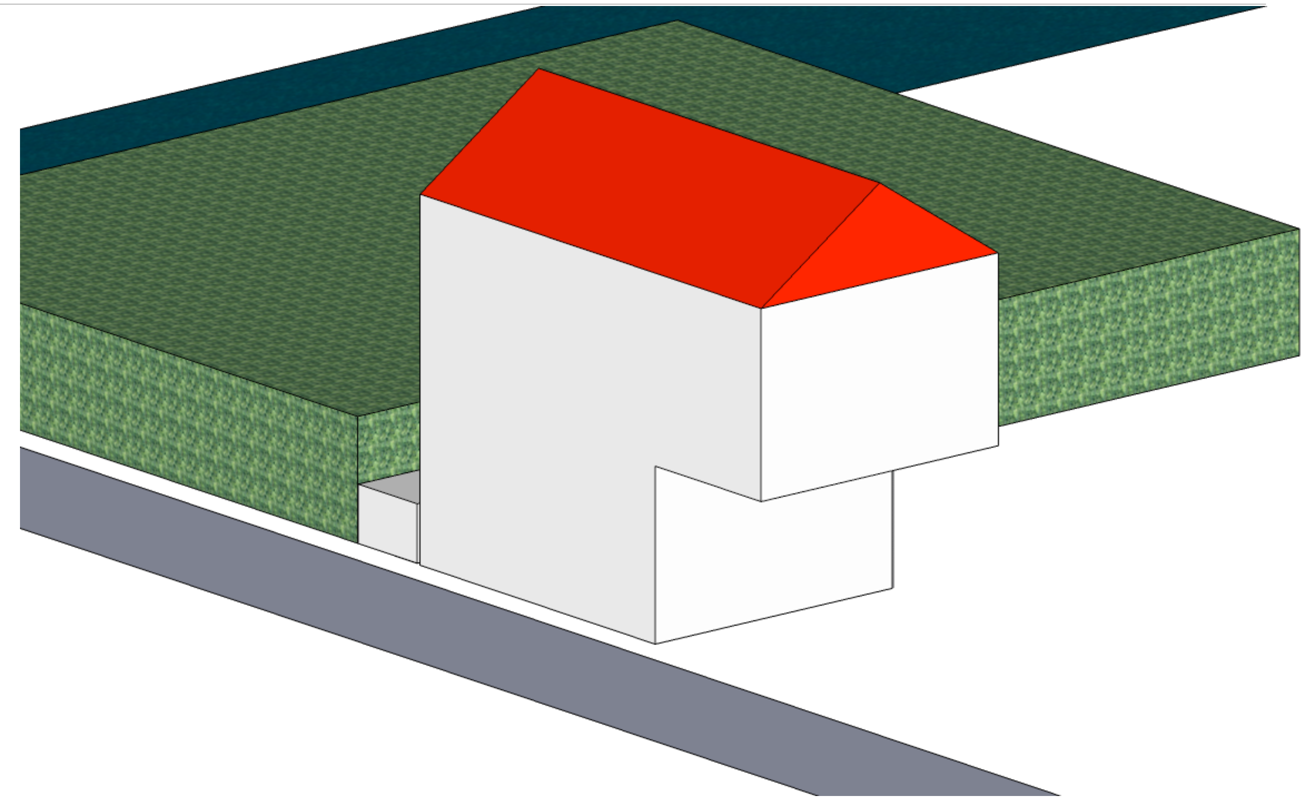
January 13, 2014
ICAA 2014

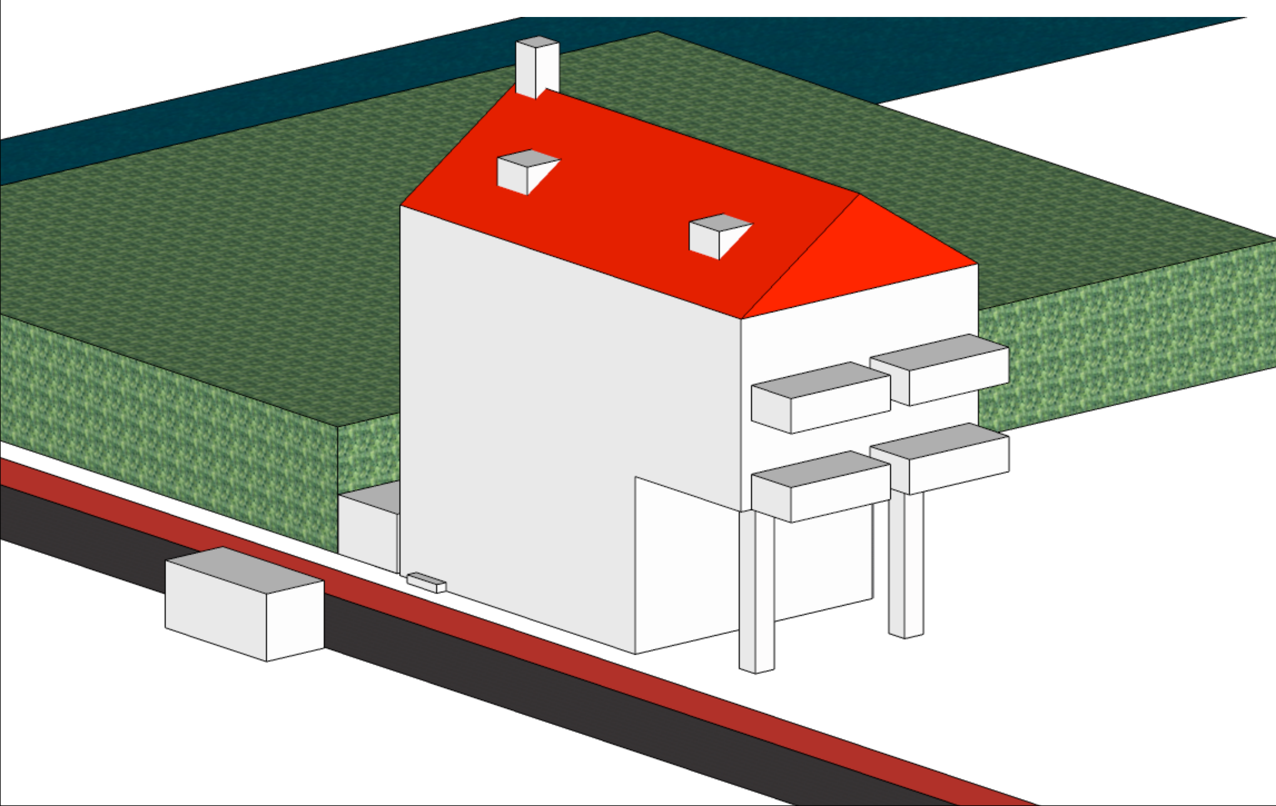
Motivation: time

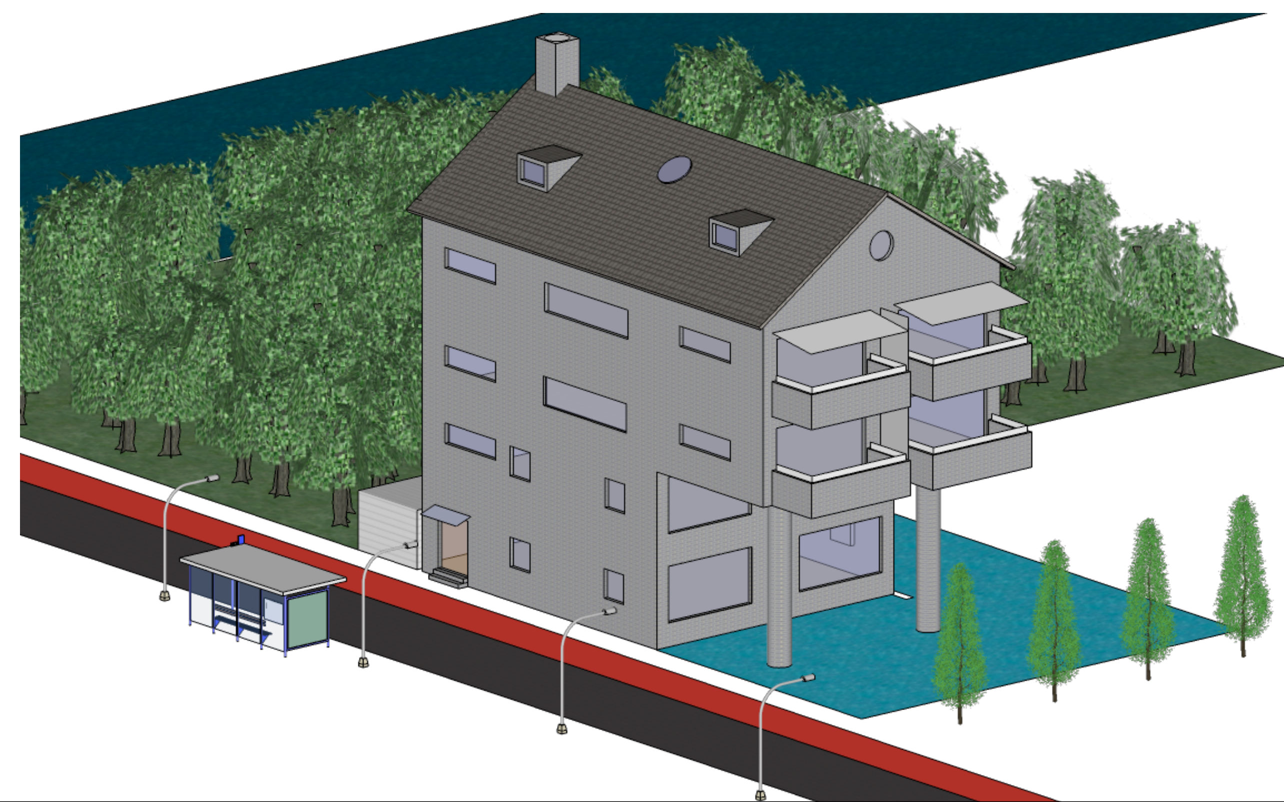


Motivation: scale



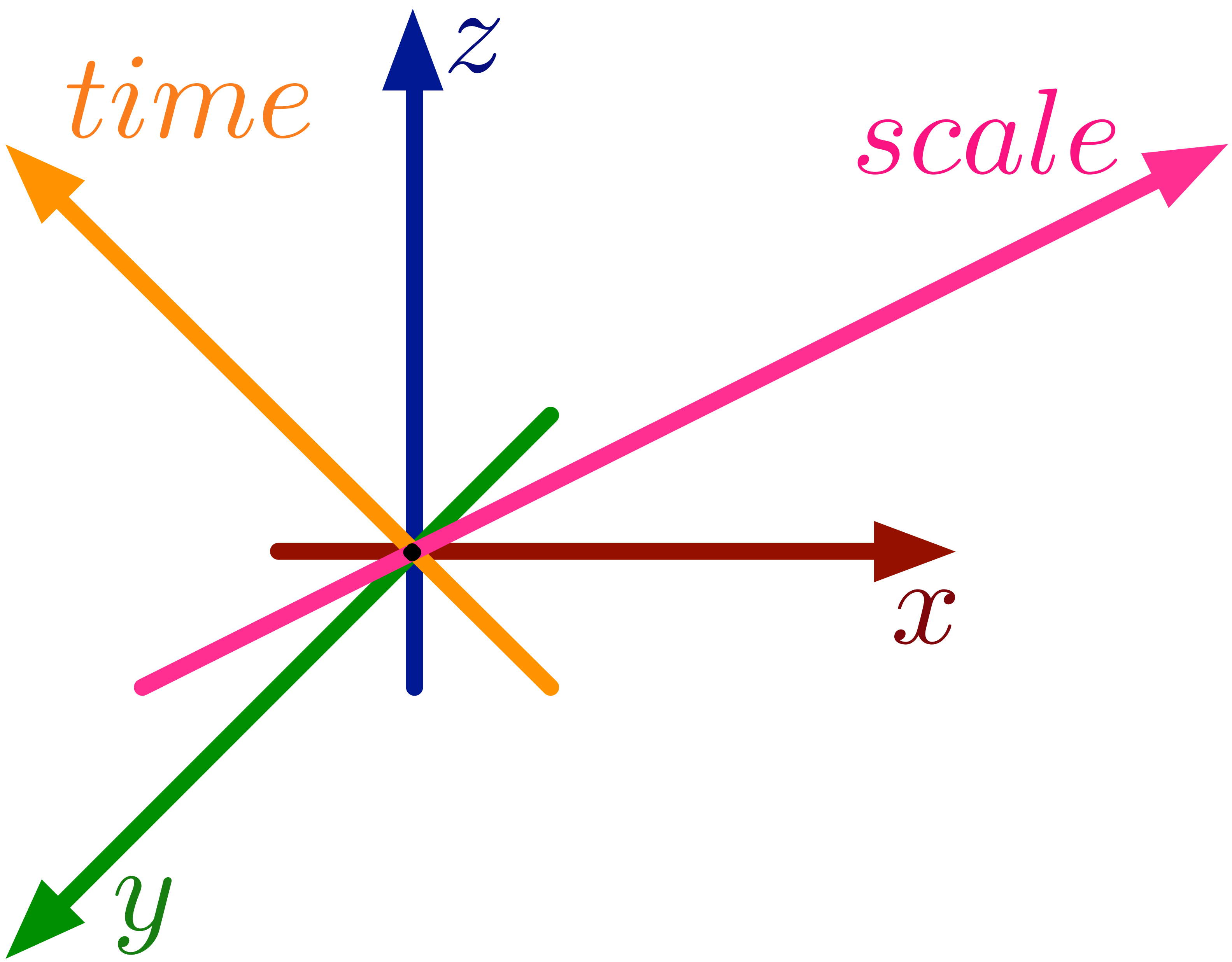




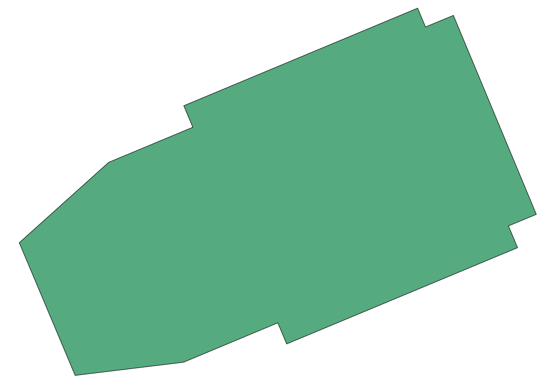


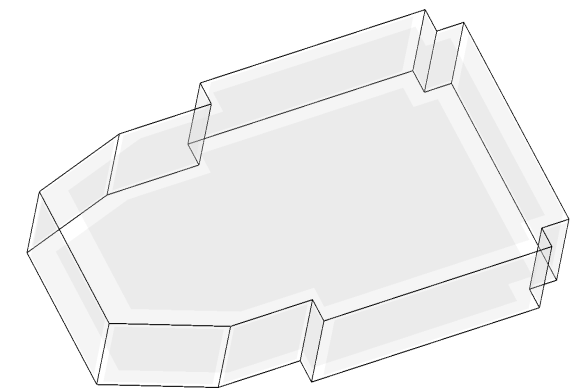
Higher dimensional objects

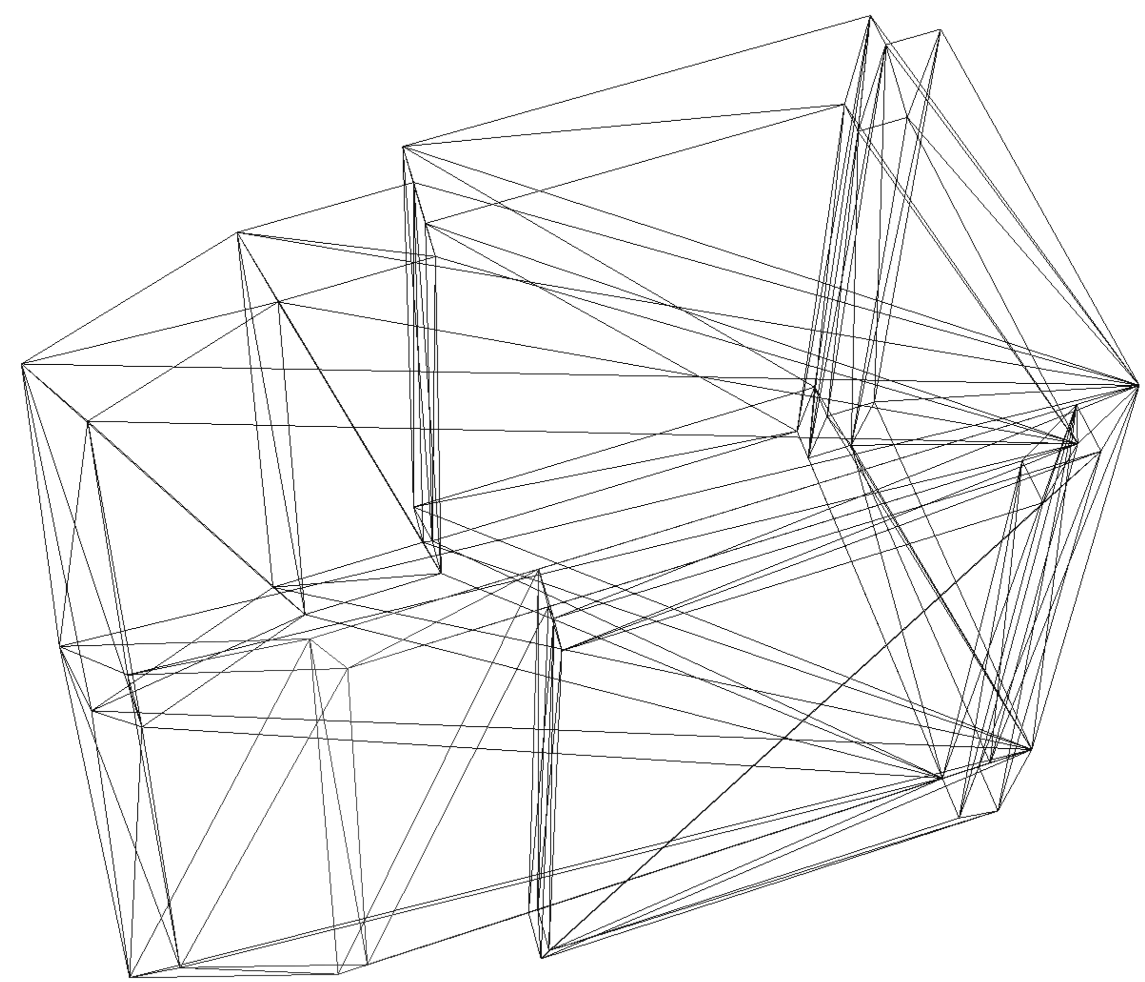
- Mathematically simple but difficult to describe intuitively



Example

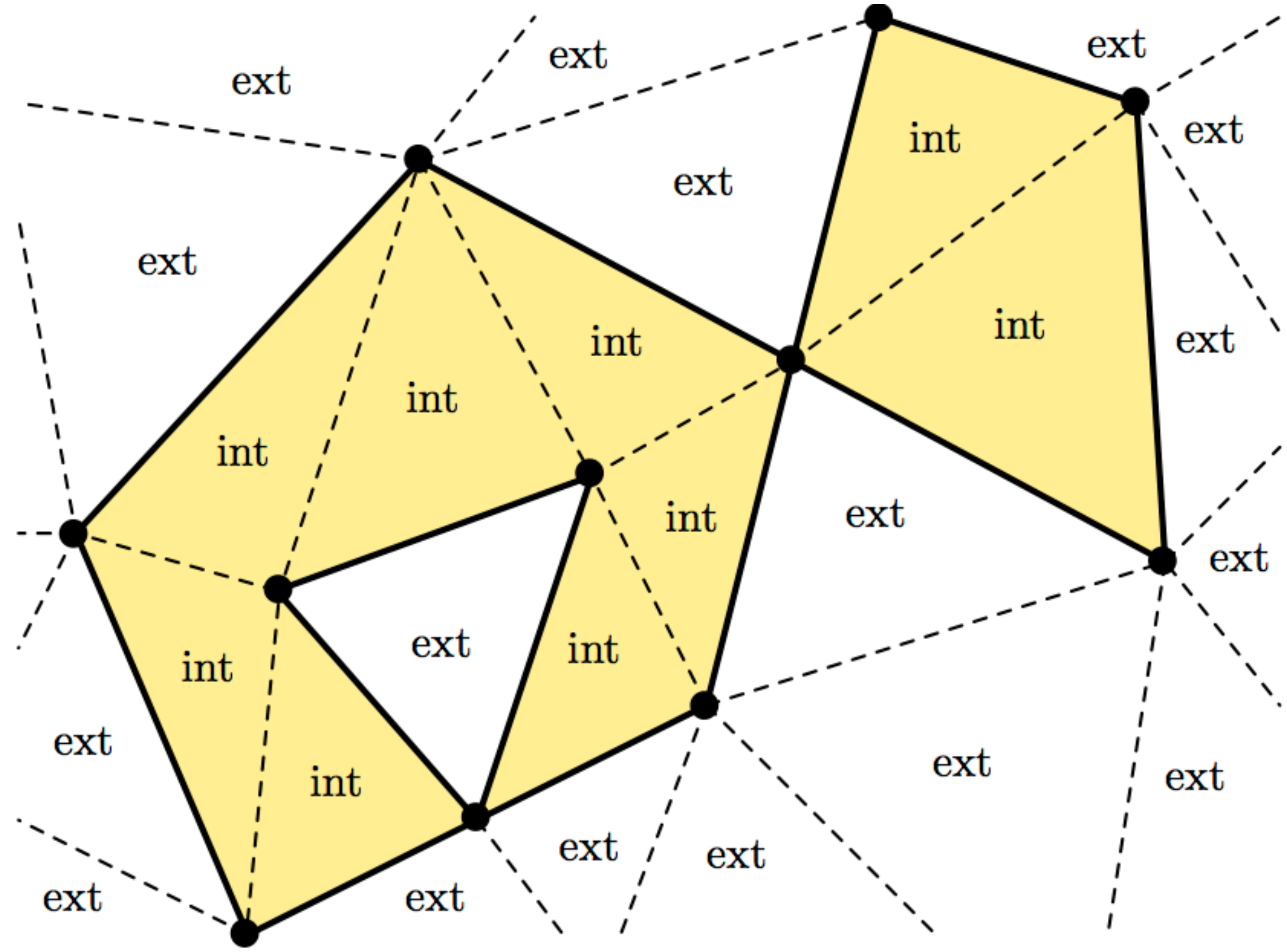






Boundary representation

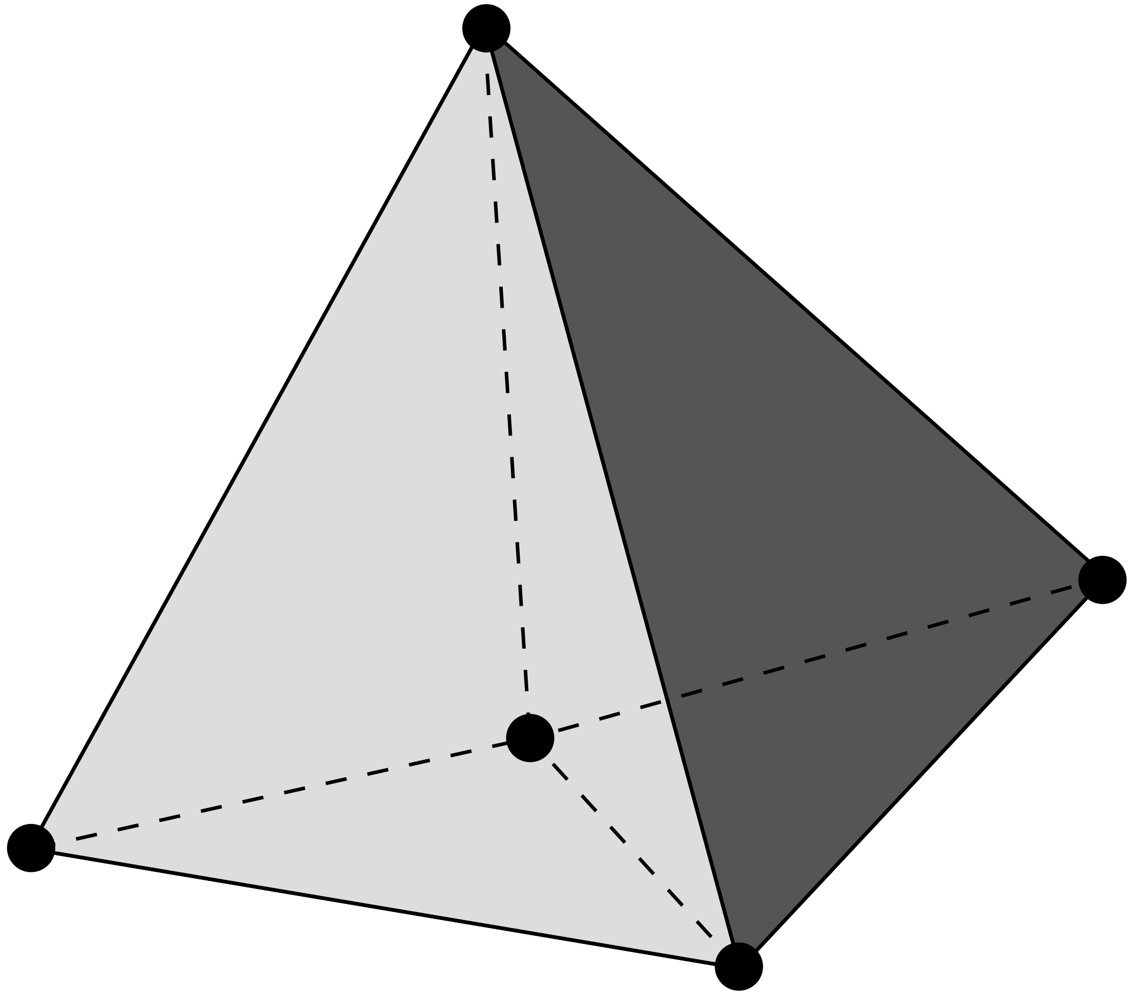
- In 2D: Jordan curve theorem (1887)
- In 3D: b-rep (informally)
- Representing an *n*-dimensional object by its (*n*-1)-dimensional boundary

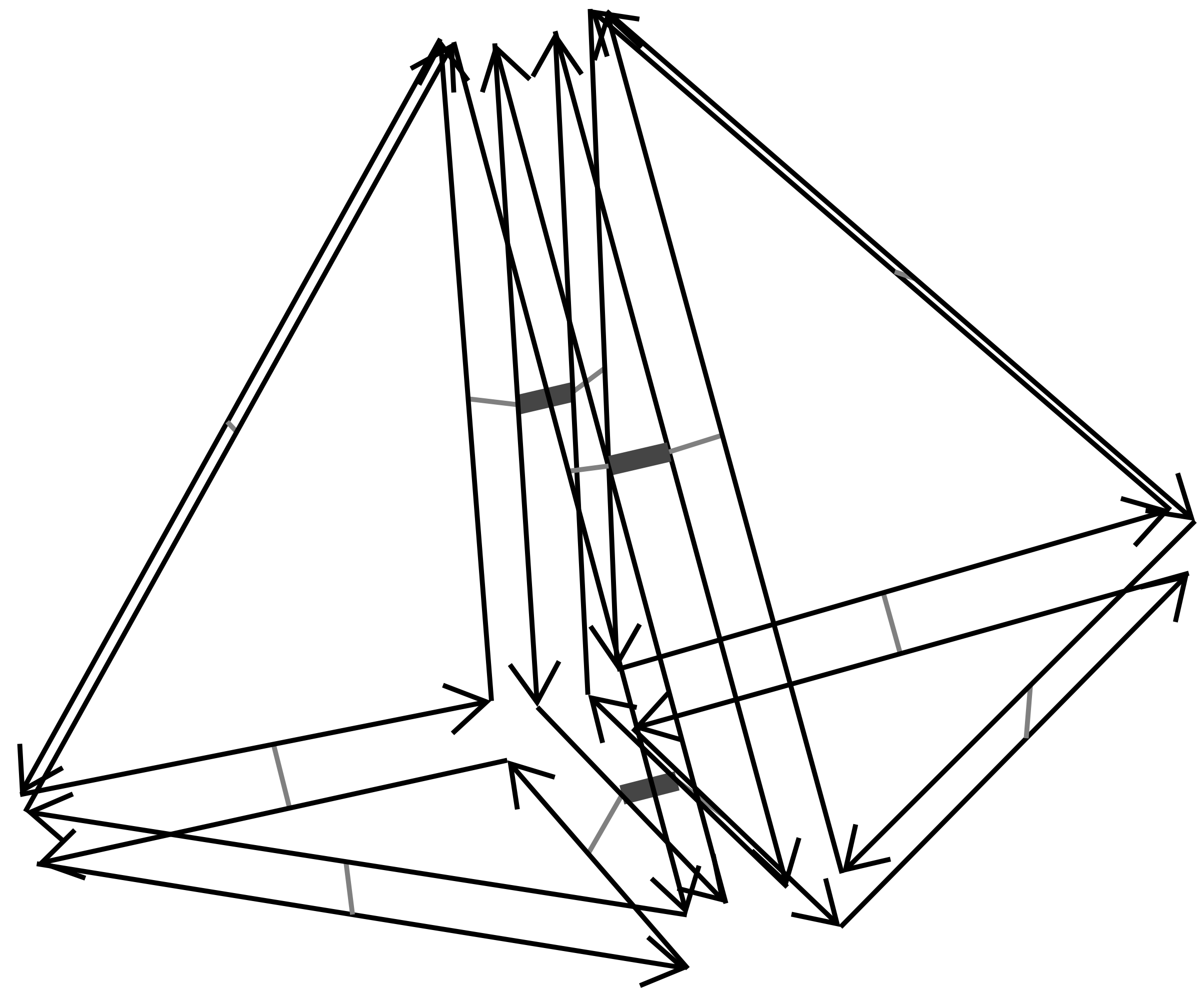


Incremental construction

- Build objects based on their boundary in increasing dimension
- 0D, (1D), 2D, 3D, 4D, …
- Connecting adjacent cells
- Equivalent to the generation of topology

Combinatorial maps





Combinatorial maps

```
struct Dart {
    Dart *involutions[n+1];
    Embeddings *embeddings[n+1];
};

struct Embedding {
    Dart *referenceDart;
    Embedding *holes[];
    int dimension;
    ...
    float red, green, blue;
};

struct PointEmbedding : Embedding {
    float x, y, z;
};
```

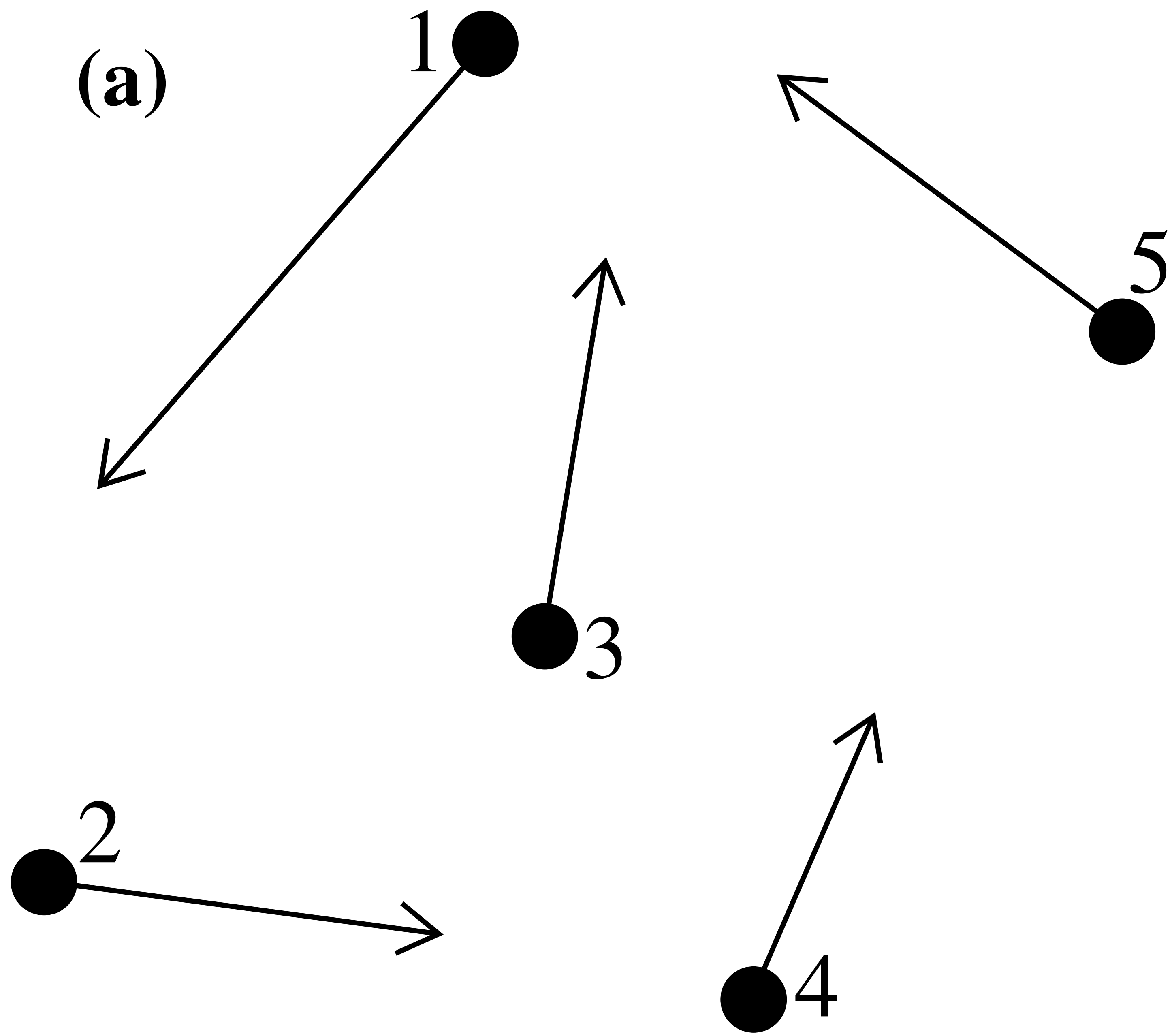
Related work

- 2D combinatorial maps [Edmonds'60]
- *n*D combinatorial maps [Lienhardt'94]
- Open combinatorial maps [Poudret'07]
- Search using signatures [Gosselin'11]
  - Test for isomorphism in quadratic time

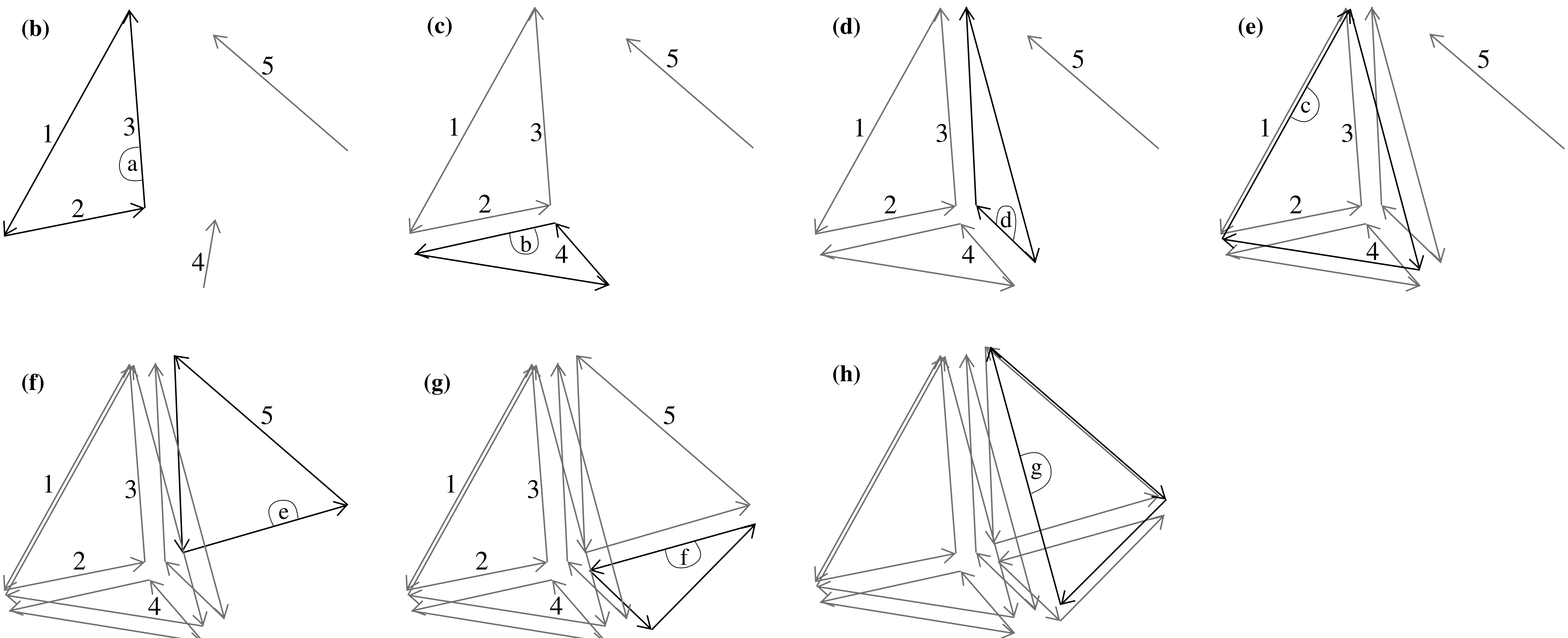
How it’s done

- Assume unique vertices
- Indices on (n-1)- and (n-2)-cells
  - Lexicographically smallest vertex
- Re-use or copy combinatorial structures (with reversed orientation)

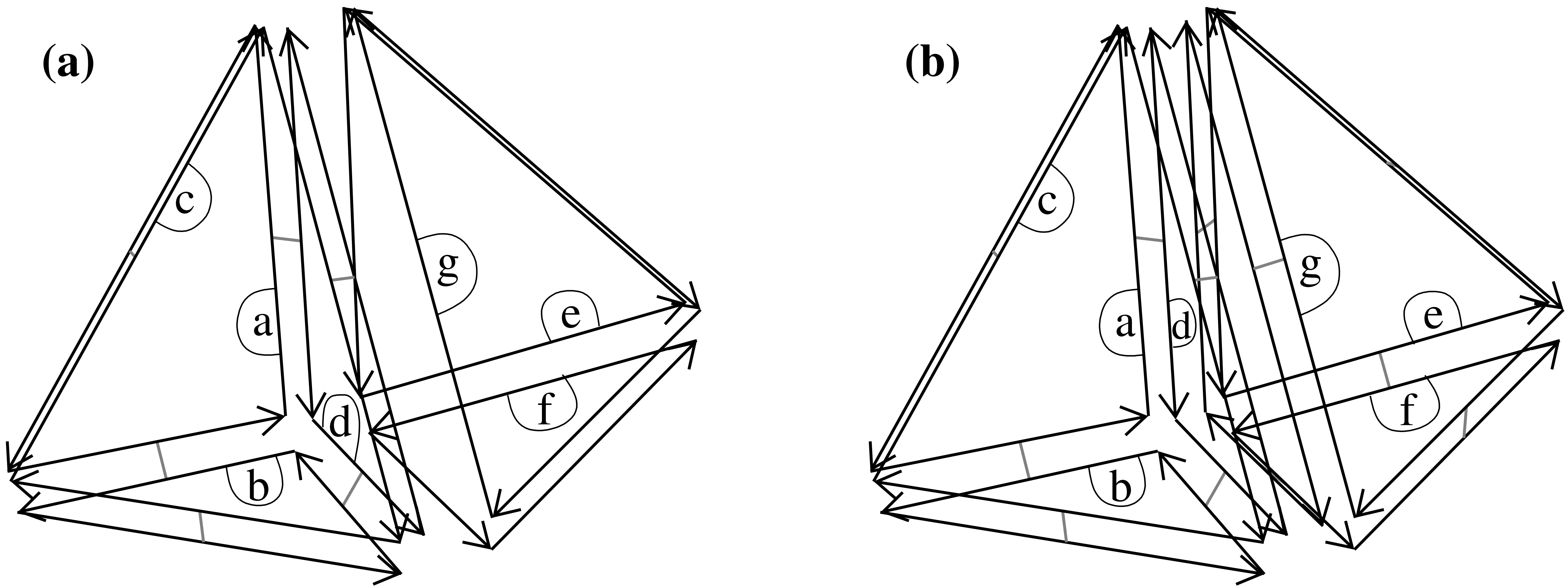
Incremental construction: 0D



Incremental construction: 2D



Incremental construction: 3D

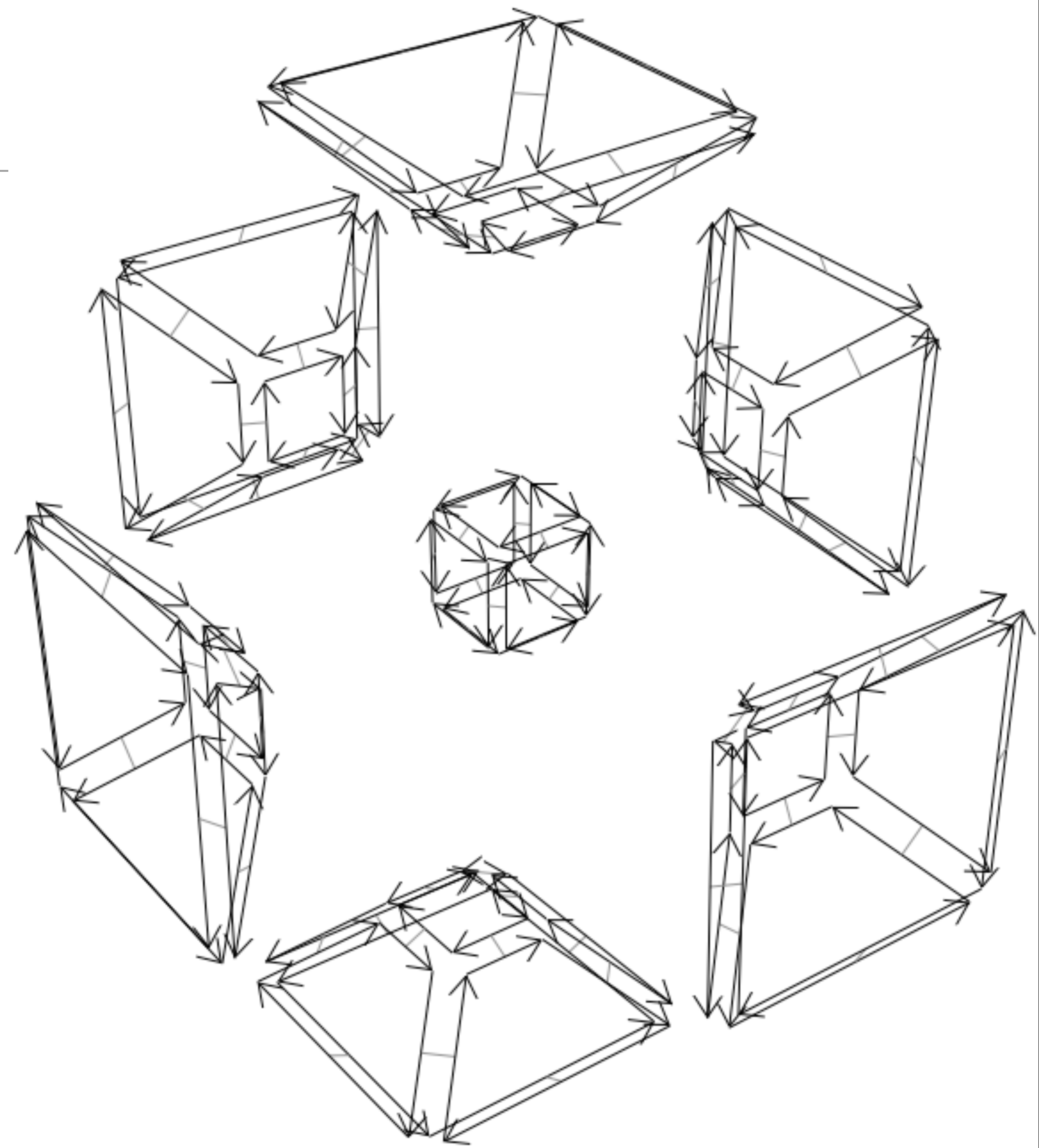


Implementation

- C++11 with recursive templates
- CGAL Combinatorial Maps and Linear Cell Complex
- **`std::map`** for isomorphism checks

Tests

- 2D, 3D and relatively simple 4D objects
- Compared with objects created with CGAL functions
- Manually by verifying β-links



Conclusions

- Intuitive method to create arbitrary *n*D cell complexes
- Fully dimension-independent method
- Quadratic time, but much better in practice
- Keep two indices only

Thank you!

`ken.mx`
`g.a.k.arroyoohori@tudelft.nl`