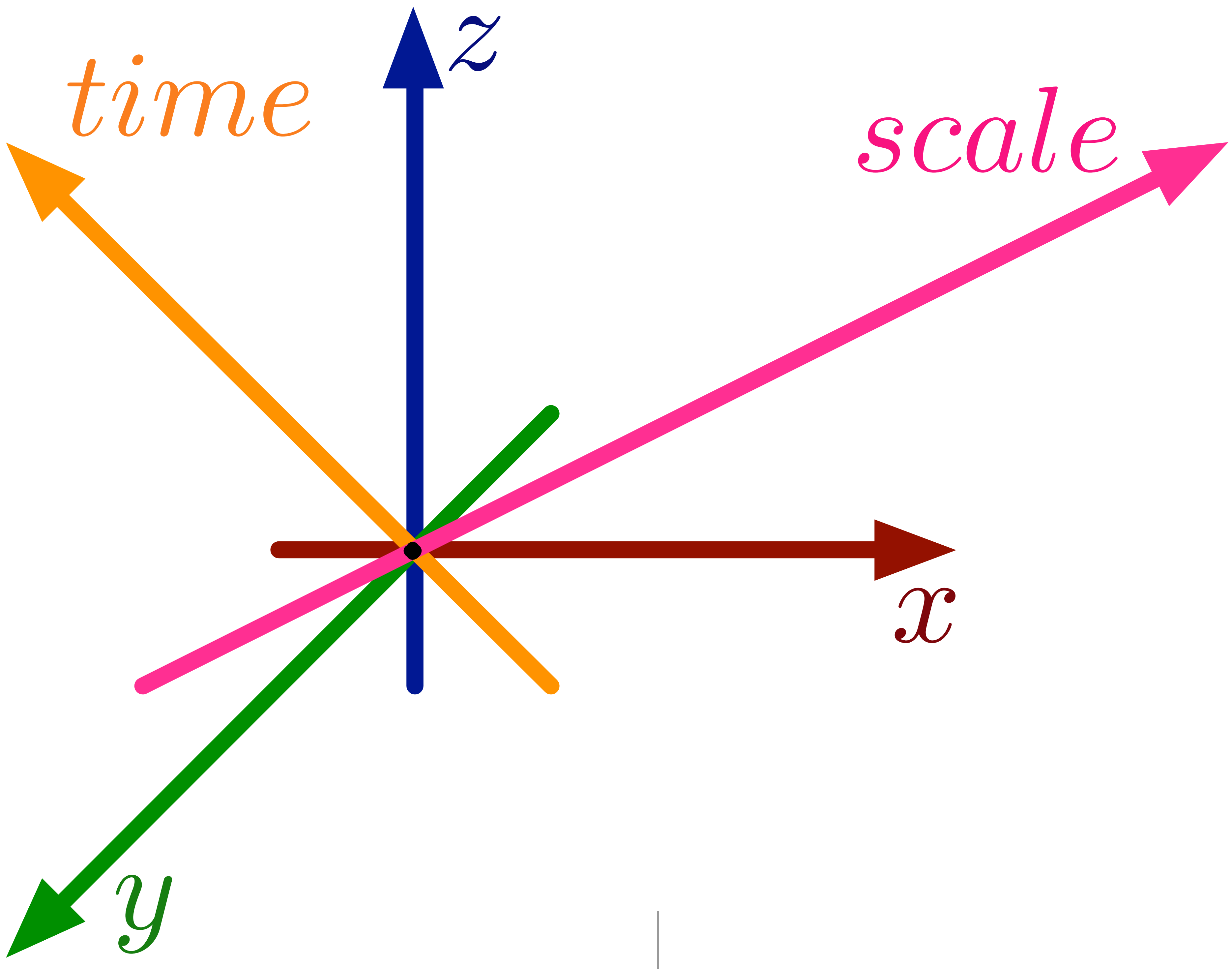
Constructing an *n*D topological representation from a soup of (*n*-1)D faces

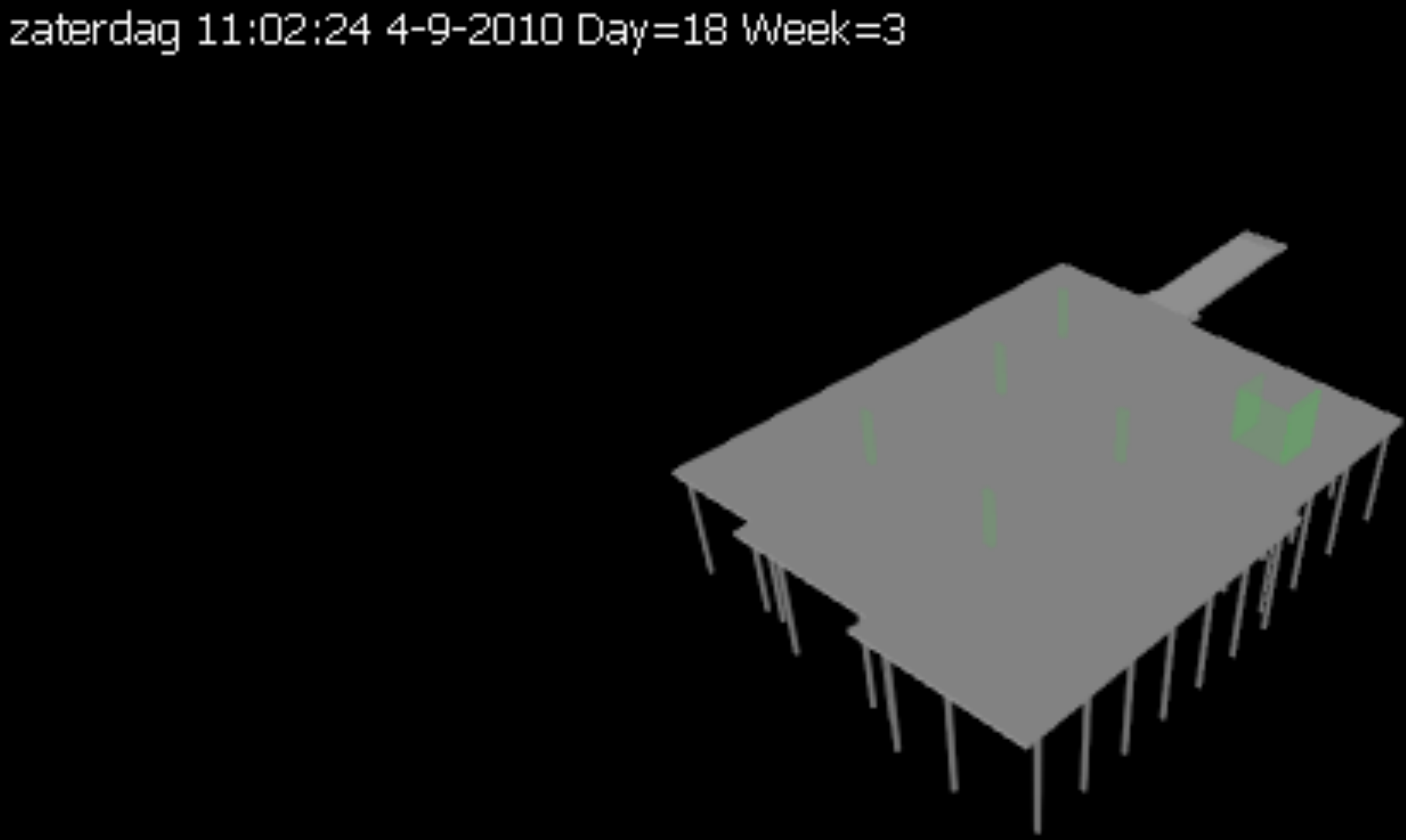
Ken Arroyo Ohori

10th Dutch Computational Geometry Day
November 19, 2015

TU Delft Delft University of Technology

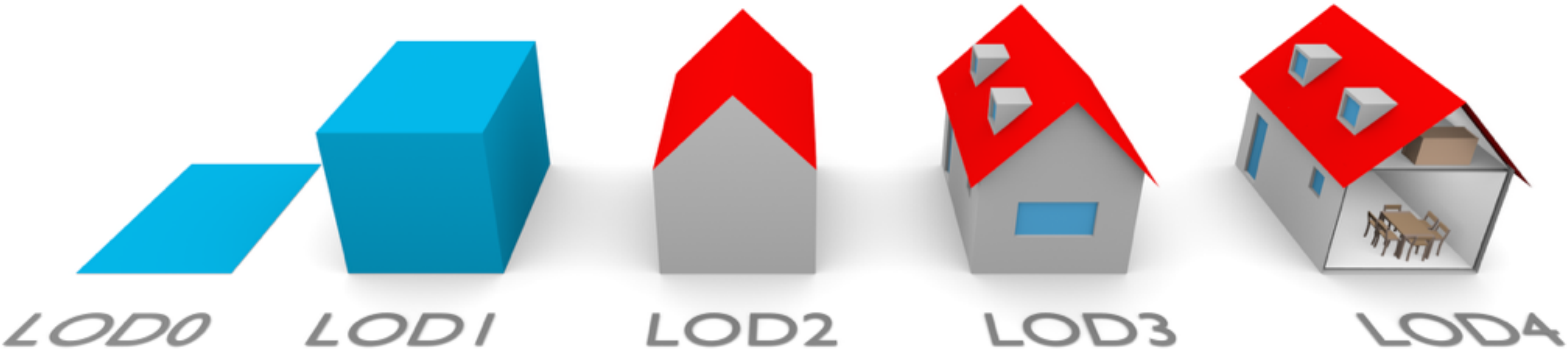


z
$time$
$scale$
x
y



Motivation

3D+time data



Filip Biljecki

Motivation

3D+scale data

But…

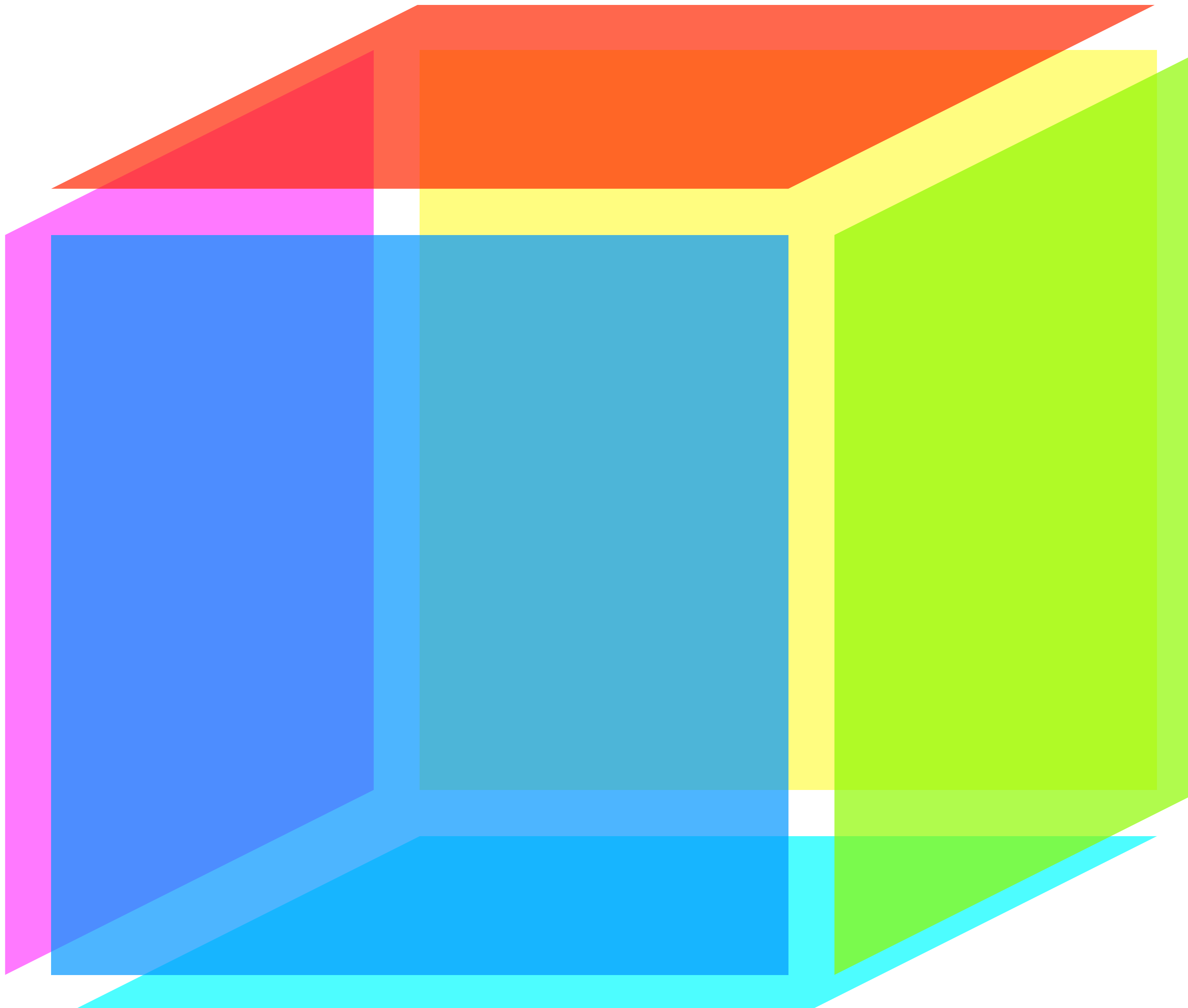
how can *n*D objects be constructed in practice?

Background

- The Jordan-Brouwer separation theorem (Lebesgue 1911, Brouwer 1911): a subset of space homeomorphic to an (n-1)D sphere S^{n-1} in $\mathbf{R}^n$ divides the space into two connected components: the *interior*, which is the region bounded by the sphere, and the *exterior*.

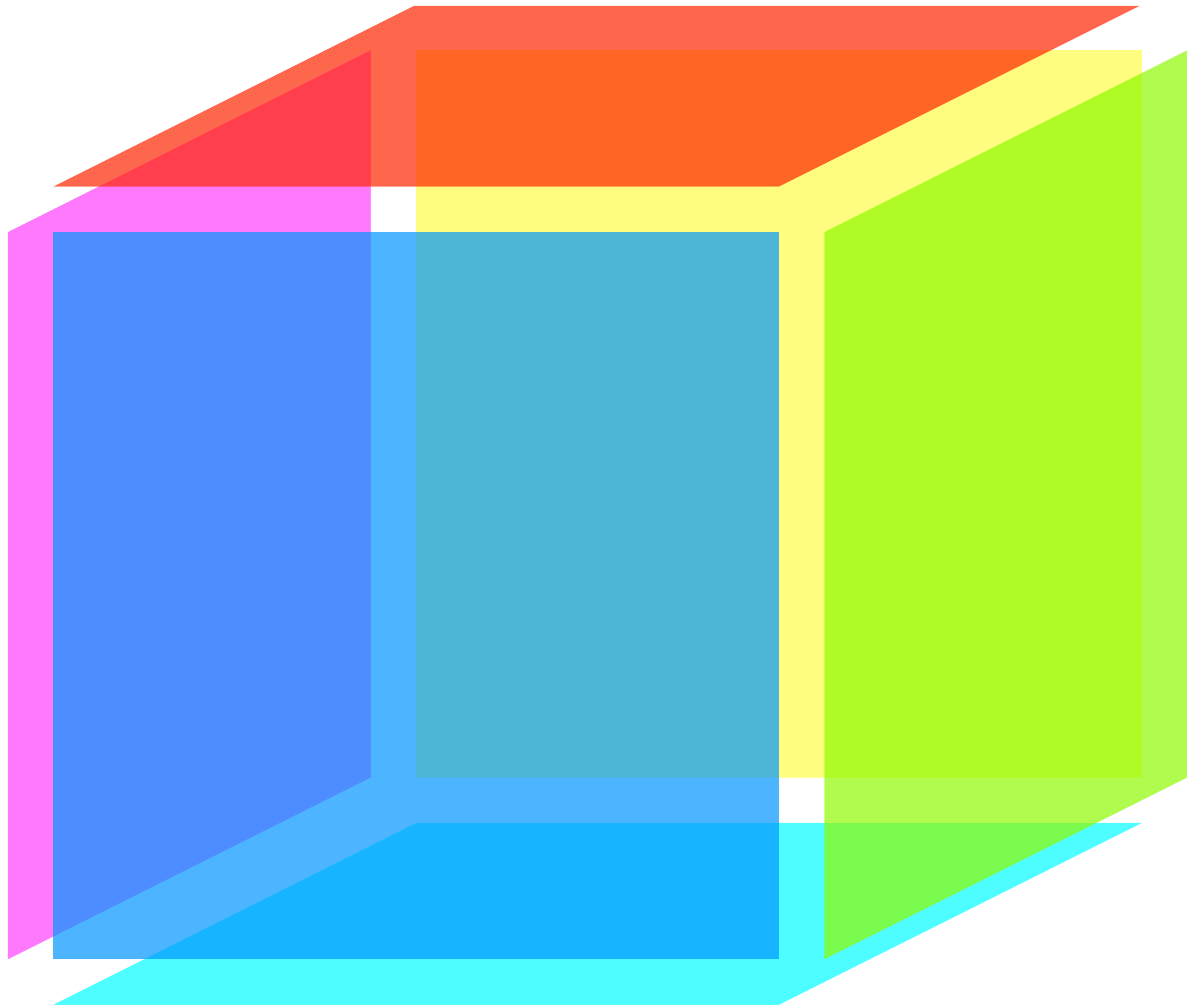
- Thus, an n-cell in a cell complex can be described (and therefore constructed) based a set of (n-1)-cells that are known to form its complete (closed) boundary.

Constructing an *n*D topological representation from a soup of (*n*-1)D faces



Alternative formulations of the same problem

- Do these (n-1)-cells form a closed (quasimanifold) object?
- Compute the adjacency relations between a set of (n-1)-cells, i.e. the common pairs of (n-2)D ridges.

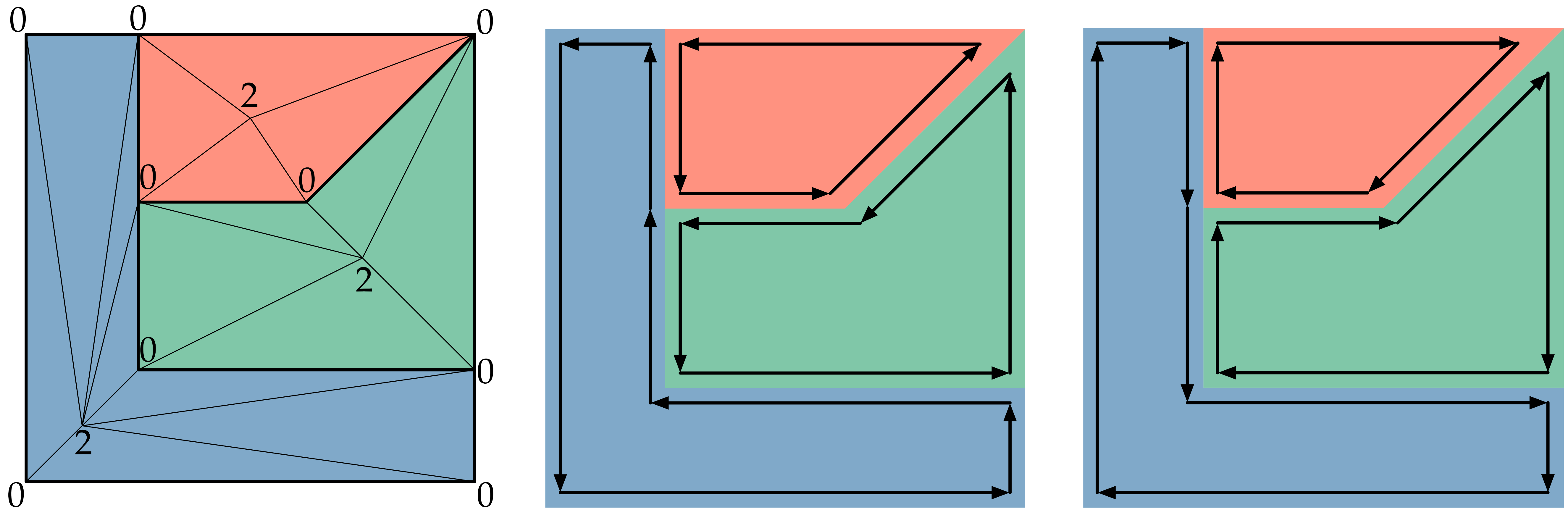


One difficulty

Boundary elements can have different orders, starting points and orientations

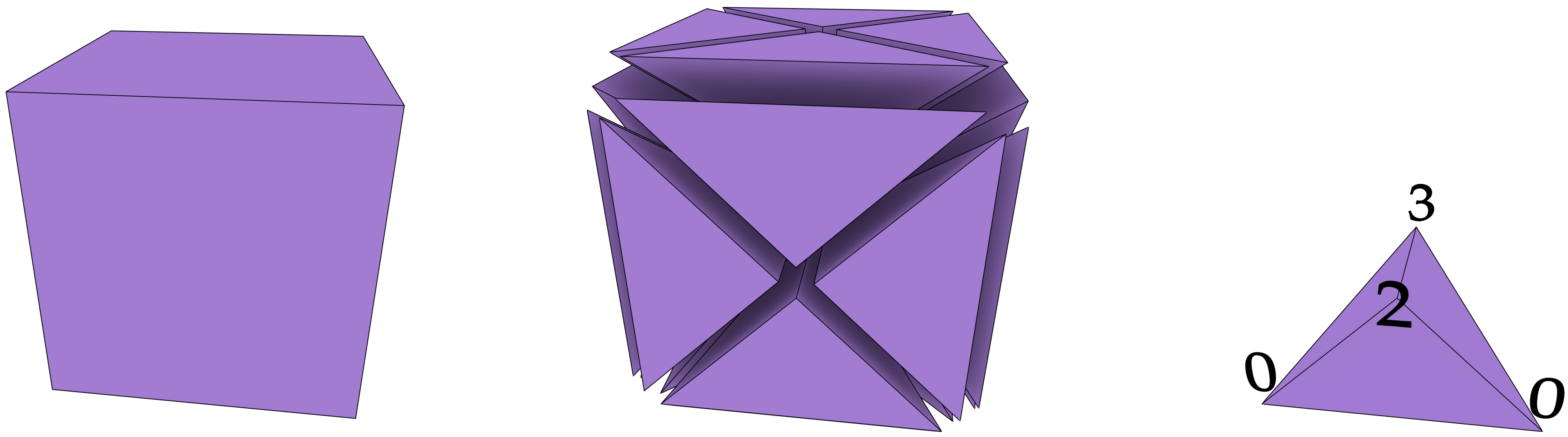
Here, a tesseract (4-cube)

```
[[[[0, 0, 0, 0], [0, 1, 0, 0], [1, 1, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0]],
  [[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 1, 0], [0, 0, 1, 0], [0, 0, 0, 0]],
  [[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 1, 0], [0, 1, 1, 0], [0, 1, 0, 0]],
  [[1, 1, 0, 0], [1, 0, 0, 0], [1, 0, 1, 0], [1, 1, 1, 0], [1, 1, 0, 0]],
  [[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 1, 0], [1, 0, 1, 0], [1, 0, 0, 0]],
  [[0, 0, 1, 0], [0, 1, 1, 0], [1, 1, 1, 0], [1, 0, 1, 0], [0, 0, 1, 0]]],
 [[[0, 0, 0, 0], [0, 1, 0, 0], [1, 1, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0]],
  [[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 0, 1], [0, 0, 0, 1], [0, 0, 0, 0]],
  [[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 0, 1], [0, 1, 0, 1], [0, 1, 0, 0]],
  [[1, 1, 0, 0], [1, 0, 0, 0], [1, 0, 0, 1], [1, 1, 0, 1], [1, 1, 0, 0]],
  [[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1], [1, 0, 0, 1], [1, 0, 0, 0]],
  [[0, 0, 0, 1], [0, 1, 0, 1], [1, 1, 0, 1], [1, 0, 0, 1], [0, 0, 0, 1]]],
 [[[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 1, 0], [0, 0, 1, 0], [0, 0, 0, 0]],
  [[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 0, 1], [0, 0, 0, 1], [0, 0, 0, 0]],
  [[0, 1, 0, 0], [0, 1, 1, 0], [0, 1, 1, 1], [0, 1, 0, 1], [0, 1, 0, 0]],
  [[0, 1, 1, 0], [0, 0, 1, 0], [0, 0, 1, 1], [0, 1, 1, 1], [0, 1, 1, 0]],
  [[0, 0, 1, 0], [0, 0, 0, 0], [0, 0, 0, 1], [0, 0, 1, 1], [0, 0, 1, 0]],
  [[0, 0, 0, 1], [0, 1, 0, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]]],
 [[[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 1, 0], [0, 1, 1, 0], [0, 1, 0, 0]],
  [[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 0, 1], [0, 1, 0, 1], [0, 1, 0, 0]],
  [[1, 1, 0, 0], [1, 1, 1, 0], [1, 1, 1, 1], [1, 1, 0, 1], [1, 1, 0, 0]],
  [[1, 1, 1, 0], [0, 1, 1, 0], [0, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 0]],
  [[0, 1, 1, 0], [0, 1, 0, 0], [0, 1, 0, 1], [0, 1, 1, 1], [0, 1, 1, 0]],
  [[0, 1, 0, 1], [1, 1, 0, 1], [1, 1, 1, 1], [0, 1, 1, 1], [0, 1, 0, 1]]],
 [[[1, 1, 0, 0], [1, 0, 0, 0], [1, 0, 1, 0], [1, 1, 1, 0], [1, 1, 0, 0]],
  [[1, 1, 0, 0], [1, 0, 0, 0], [1, 0, 0, 1], [1, 1, 0, 1], [1, 1, 0, 0]],
  [[1, 0, 0, 0], [1, 0, 1, 0], [1, 0, 1, 1], [1, 0, 0, 1], [1, 0, 0, 0]],
  [[1, 0, 1, 0], [1, 1, 1, 0], [1, 1, 1, 1], [1, 0, 1, 1], [1, 0, 1, 0]],
  [[1, 1, 1, 0], [1, 1, 0, 0], [1, 1, 0, 1], [1, 1, 1, 1], [1, 1, 1, 0]],
  [[1, 1, 0, 1], [1, 0, 0, 1], [1, 0, 1, 1], [1, 1, 1, 1], [1, 1, 0, 1]]],
 [[[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 1, 0], [1, 0, 1, 0], [1, 0, 0, 0]],
  [[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1], [1, 0, 0, 1], [1, 0, 0, 0]],
  [[0, 0, 0, 0], [0, 0, 1, 0], [0, 0, 1, 1], [0, 0, 0, 1], [0, 0, 0, 0]],
  [[0, 0, 1, 0], [1, 0, 1, 0], [1, 0, 1, 1], [0, 0, 1, 1], [0, 0, 1, 0]],
  [[1, 0, 1, 0], [1, 0, 0, 0], [1, 0, 0, 1], [1, 0, 1, 1], [1, 0, 1, 0]],
  [[1, 0, 0, 1], [0, 0, 0, 1], [0, 0, 1, 1], [1, 0, 1, 1], [1, 0, 0, 1]]],
 [[[0, 0, 1, 0], [0, 1, 1, 0], [1, 1, 1, 0], [1, 0, 1, 0], [0, 0, 1, 0]],
  [[0, 0, 1, 0], [0, 1, 1, 0], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 1, 0]],
  [[0, 1, 1, 0], [1, 1, 1, 0], [1, 1, 1, 1], [0, 1, 1, 1], [0, 1, 1, 0]],
  [[1, 1, 1, 0], [1, 0, 1, 0], [1, 0, 1, 1], [1, 1, 1, 1], [1, 1, 1, 0]],
  [[1, 0, 1, 0], [0, 0, 1, 0], [0, 0, 1, 1], [1, 0, 1, 1], [1, 0, 1, 0]],
  [[0, 0, 1, 1], [0, 1, 1, 1], [1, 1, 1, 1], [1, 0, 1, 1], [0, 0, 1, 1]]],
 [[[0, 0, 0, 1], [0, 1, 0, 1], [1, 1, 0, 1], [1, 0, 0, 1], [0, 0, 0, 1]],
  [[0, 0, 0, 1], [0, 1, 0, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]],
  [[0, 1, 0, 1], [1, 1, 0, 1], [1, 1, 1, 1], [0, 1, 1, 1], [0, 1, 0, 1]],
  [[1, 1, 0, 1], [1, 0, 0, 1], [1, 0, 1, 1], [1, 1, 1, 1], [1, 1, 0, 1]],
  [[1, 0, 0, 1], [0, 0, 0, 1], [0, 0, 1, 1], [1, 0, 1, 1], [1, 0, 0, 1]],
  [[0, 0, 1, 1], [0, 1, 1, 1], [1, 1, 1, 1], [1, 0, 1, 1], [0, 0, 1, 1]]]]
```



Background

2D combinatorial maps



Background

3D combinatorial maps

Background

- Gosselin et al. (2011) describe a method to check if two combinatorial maps are isomorphic using *signatures*.
- Based on the ordering properties of a combinatorial map, it is possible to traverse the darts of a cell in a manner that is always consistent, yielding a canonical representation.
- By following parallel traversals of this type, an algorithm can verify that two cells or maps are isomorphic in $O(n^2)$ time on the number of darts in a cell/map.

The incremental construction method

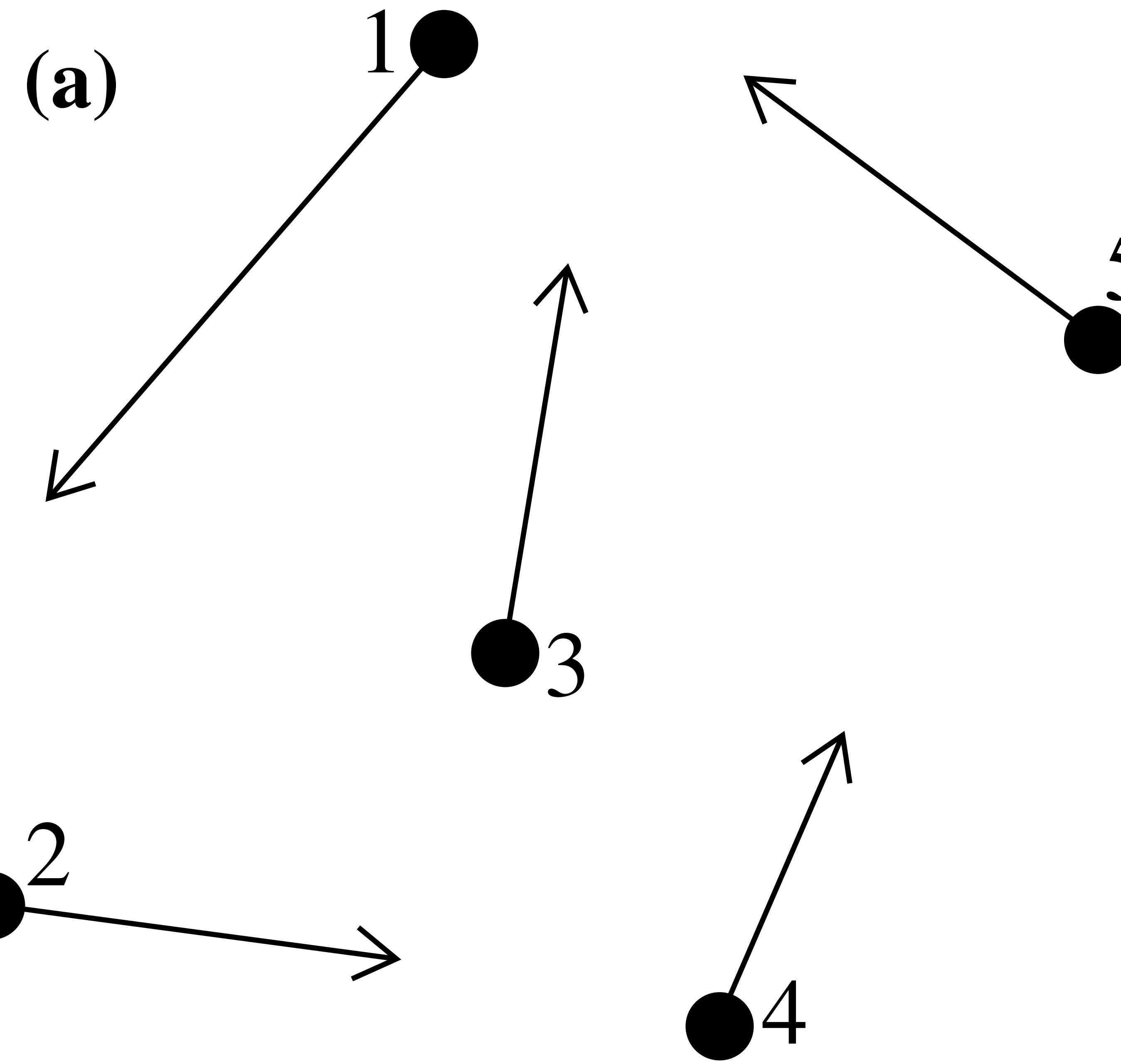
- Ensure that every element (as defined by its geometry) is only created once
- For the construction of a given n-cell based on a set of (n-1)-cells (faces), quickly find their common (n-2)-cells (ridges)
- For a quasi-manifold, ridges should form pairs, which are linked
- Reverse orientations whenever needed

As an algorithm

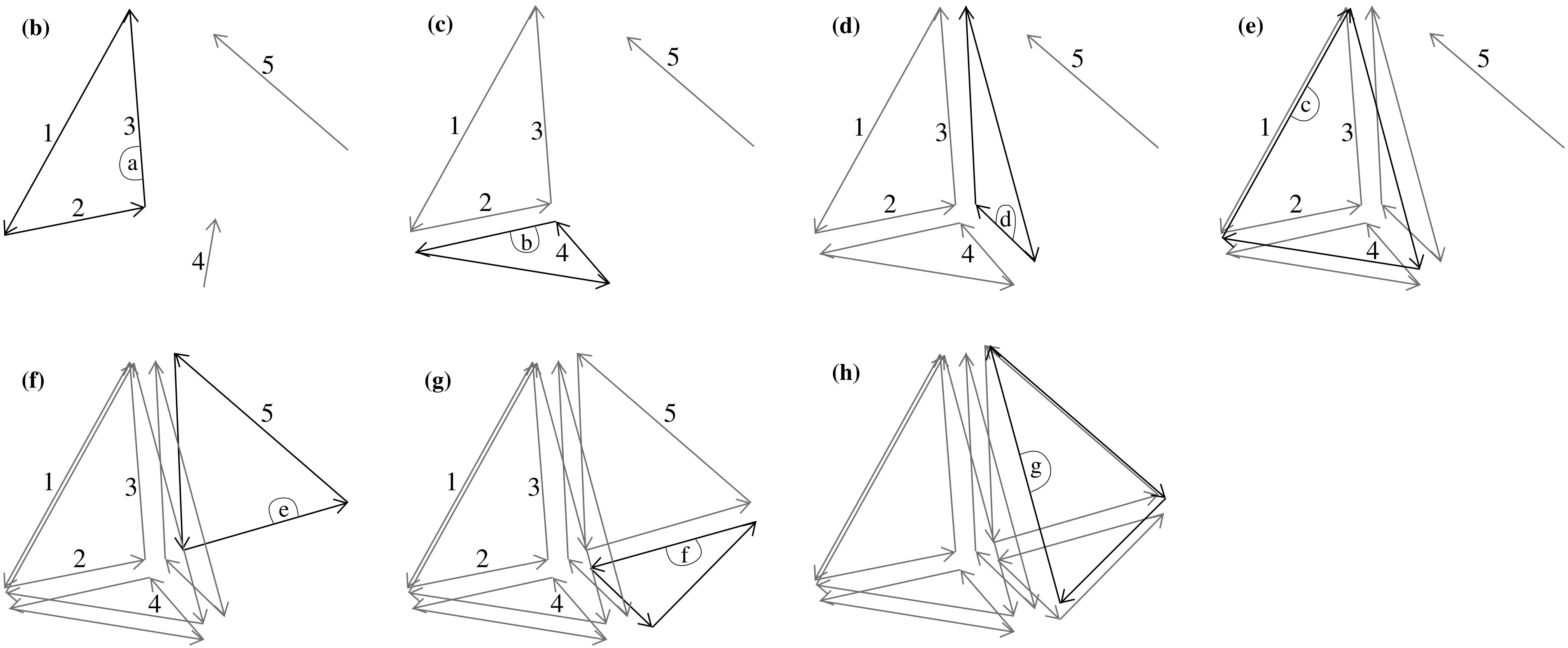
- Check for element equality by testing for isomorphism and equal vertex embeddings in $\mathbf{R}^n$
- Simple index on the lexicographically smallest vertex of every cell
- Complexity is $O(n^2)$ with *n* the number of darts per cell, but close to linear in practice

Incremental construction: 0D

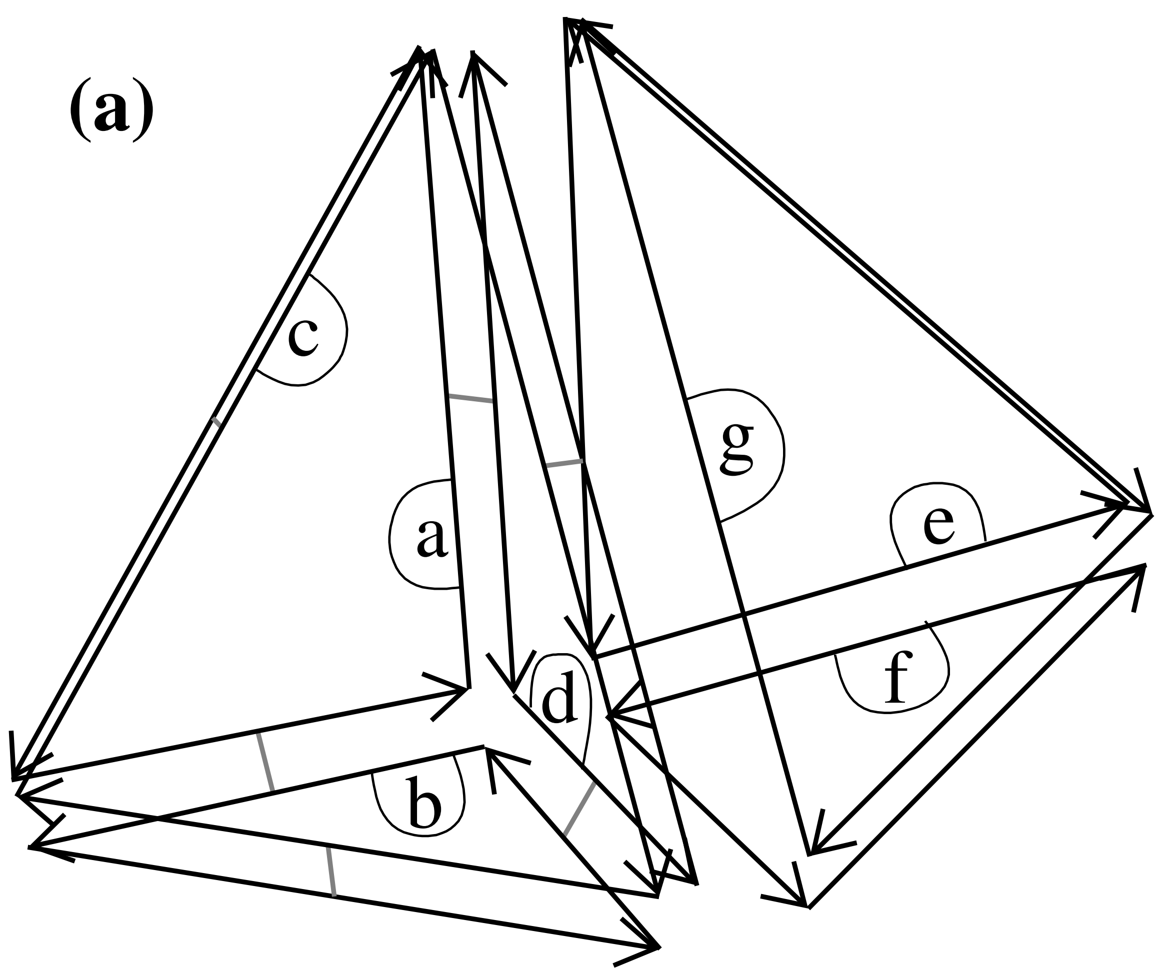
Unique points



Incremental construction: 2D

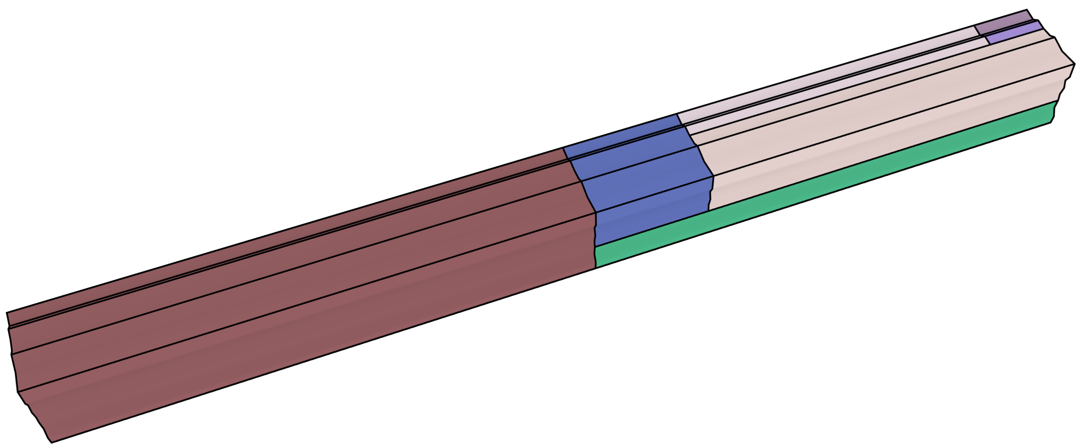


Incremental construction: 3D and higher



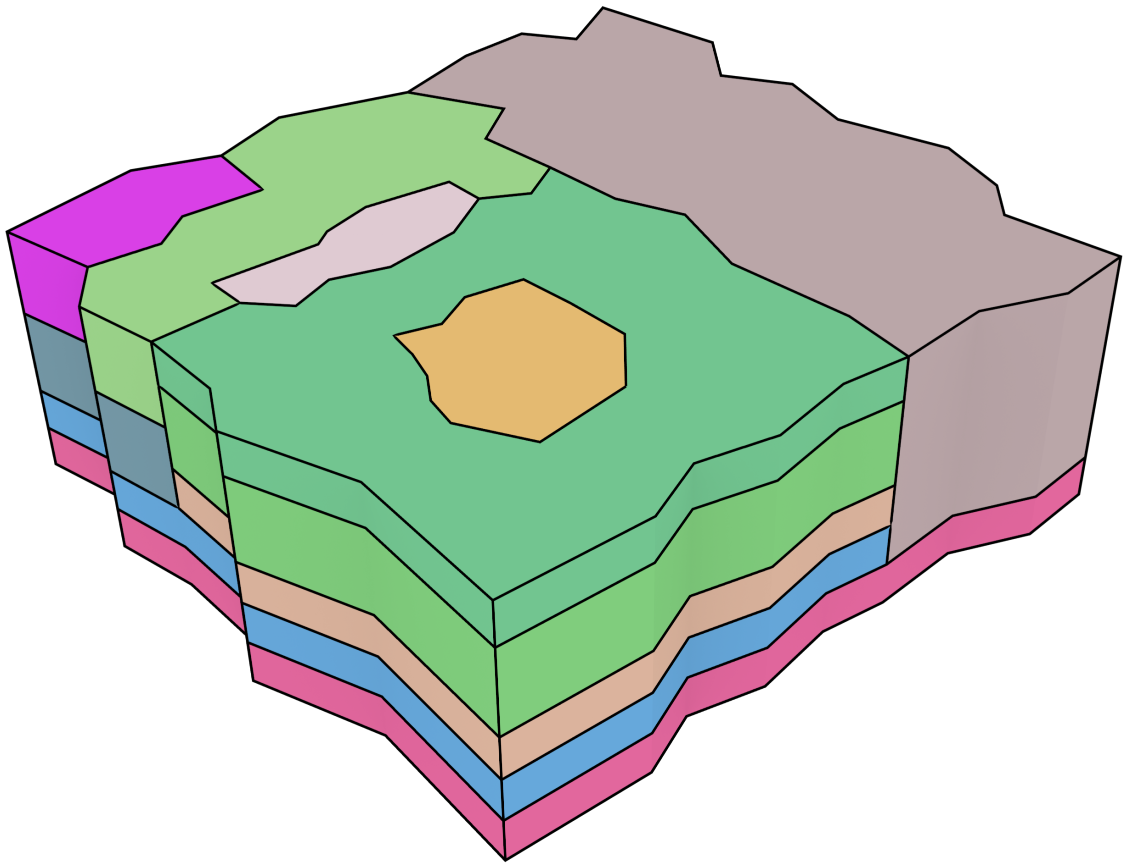
Implementation

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



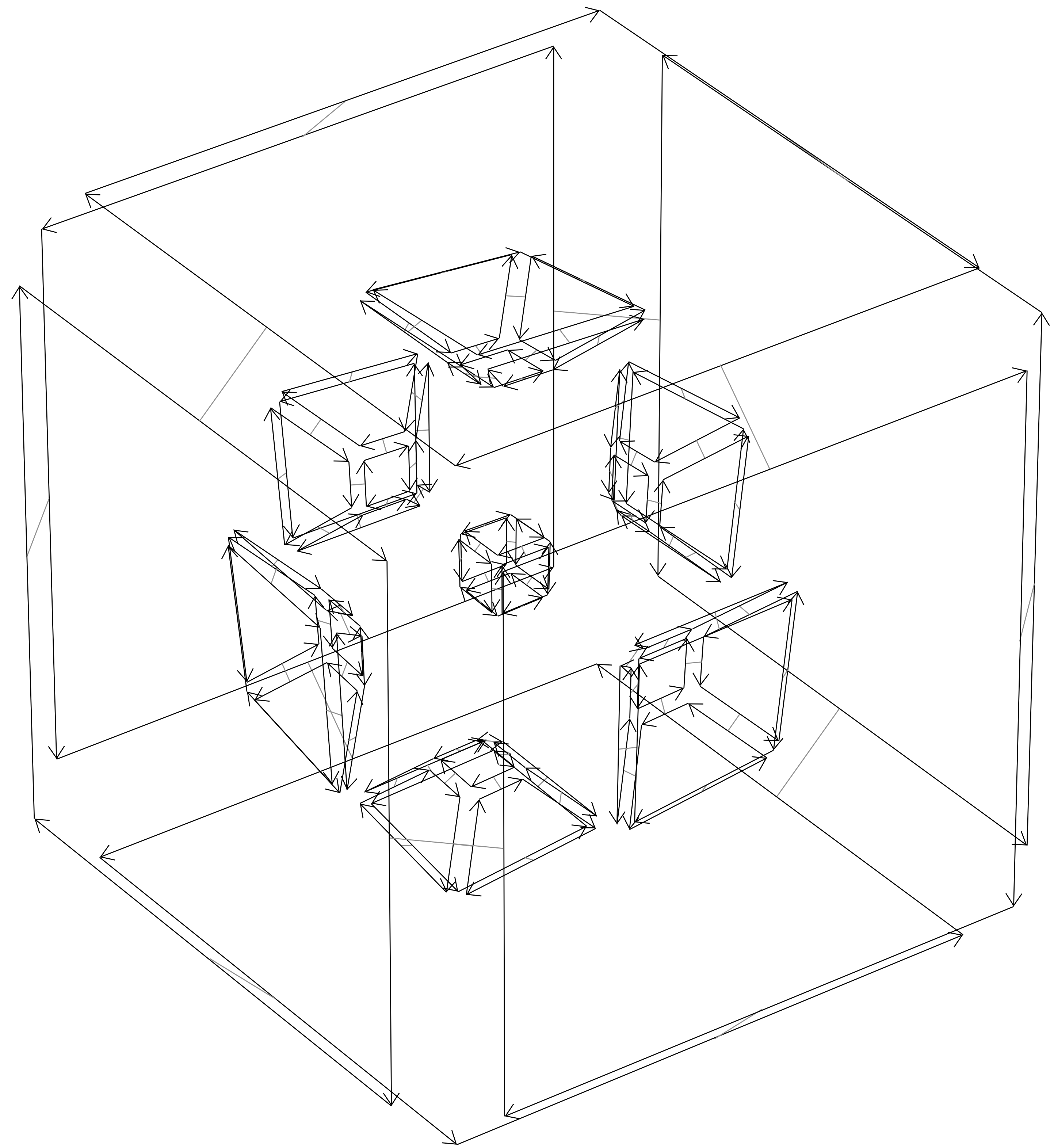
Tests

2D+scale as 3D



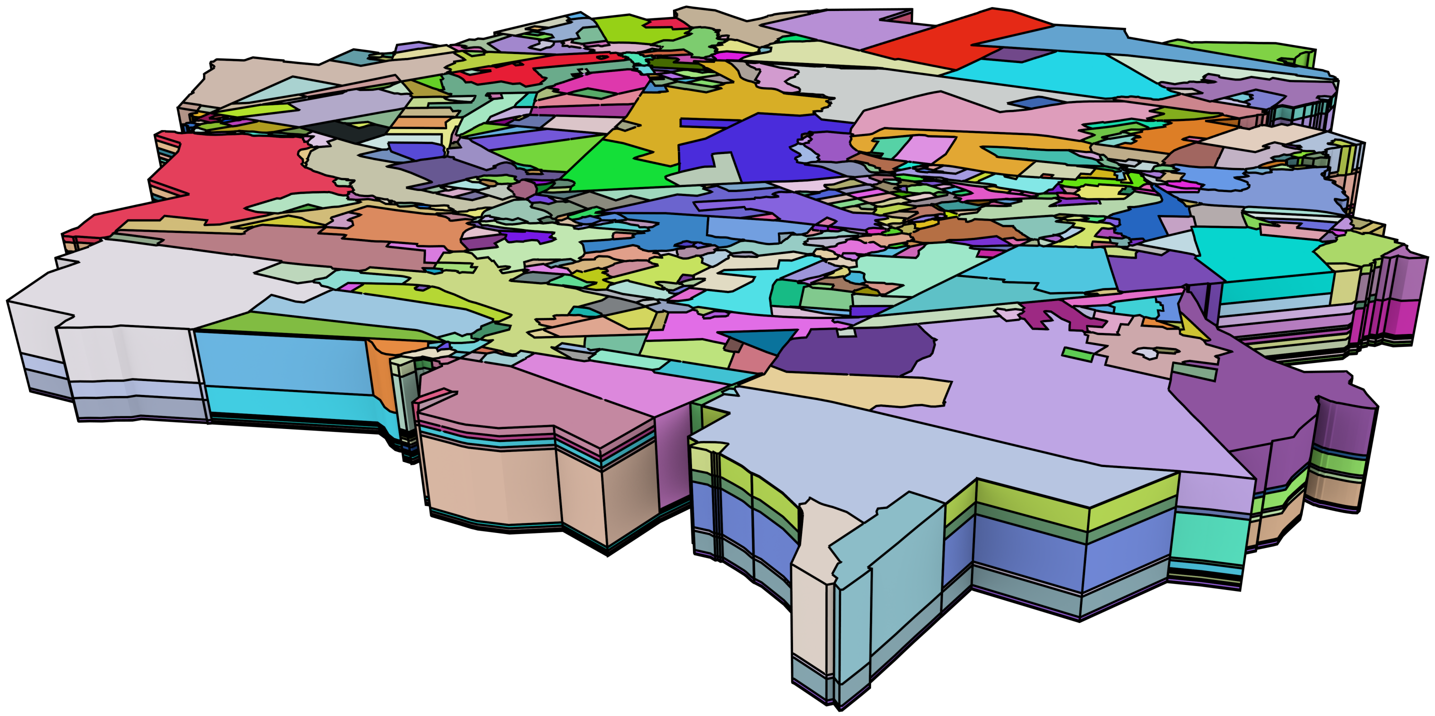
Tests

Simple 4D objects



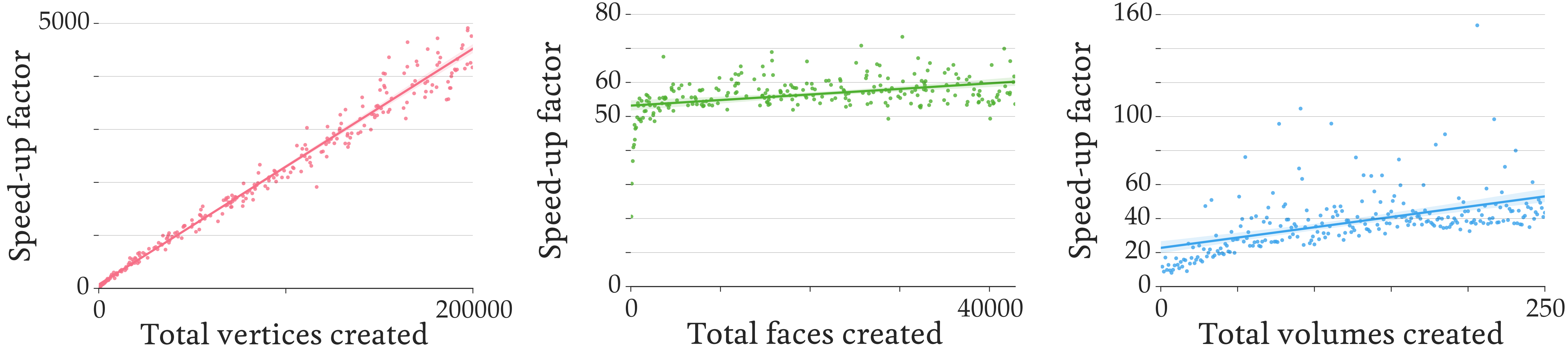
Tests

Large 2D+scale datasets



Here, 698 polyhedra with 457 185 faces

Speed gains from the use of indices



Read more

- **Constructing an *n*-dimensional cell complex from a soup of (*n*-1)-dimensional faces.** Ken Arroyo Ohori, Guillaume Damiand and Hugo Ledoux. In Prosenjit Gupta and Christos Zaroliagis (eds.), *Applied Algorithms*, ICAA 2014, Kolkata, India, January 13-15, 2014. Lecture Notes in Computer Science 8321, Springer International Publishing Switzerland, January 2014, pp. 37–48.

Thank you!

Ken Arroyo Ohori
tudelft.nl/kenohori

References

- L.E.J. Brouwer. **Beweis des Jordanschen Satzes für den *n*-dimensionalen Raum**. *Mathematische Annalen*, 71:314–319, 1911.

- **Sur l'invariance du nombre de dimensions d'un espace et sur le theorème de M. Jordan relatif aux varieté fermées**. *Comptes rendus de l'Académie des Sciences*, 152:841–844, 1911.

- Stéphane Gosselin, Guillaume Damiand, and Christine Solnon. **Efficient search of combinatorial maps using signatures**. *Theoretical Computer Science*, 412 (15):1392–1405, March 2011.