Using extrusion to generate higher-dimensional GIS datasets

Ken Arroyo Ohori
Hugo Ledoux

November 6, 2013
ACM SIGSPATIAL GIS 2013

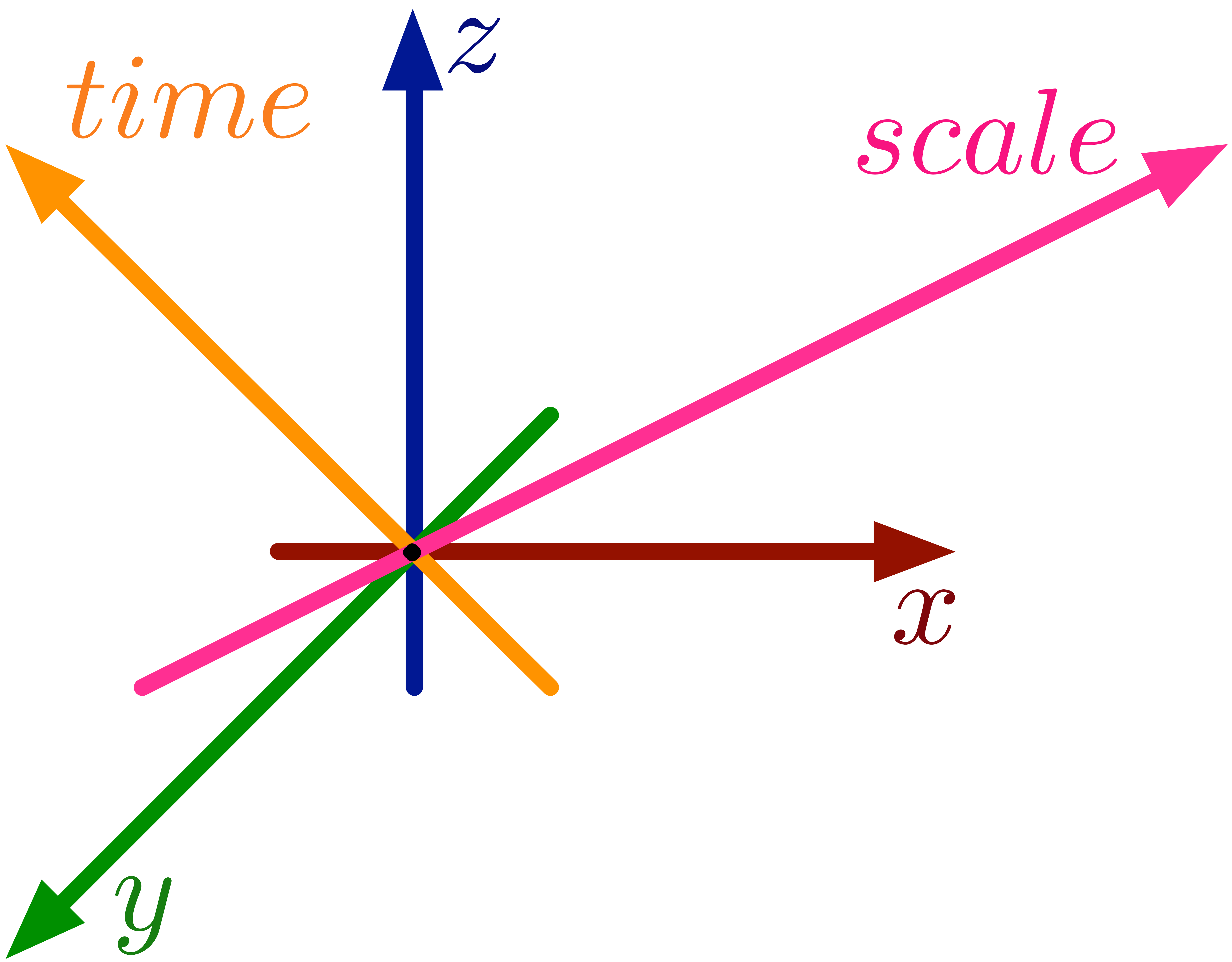
TU Delft Delft University of Technology

From multiple representation to higher dimensional GIS

From multiple representation to higher dimensional GIS

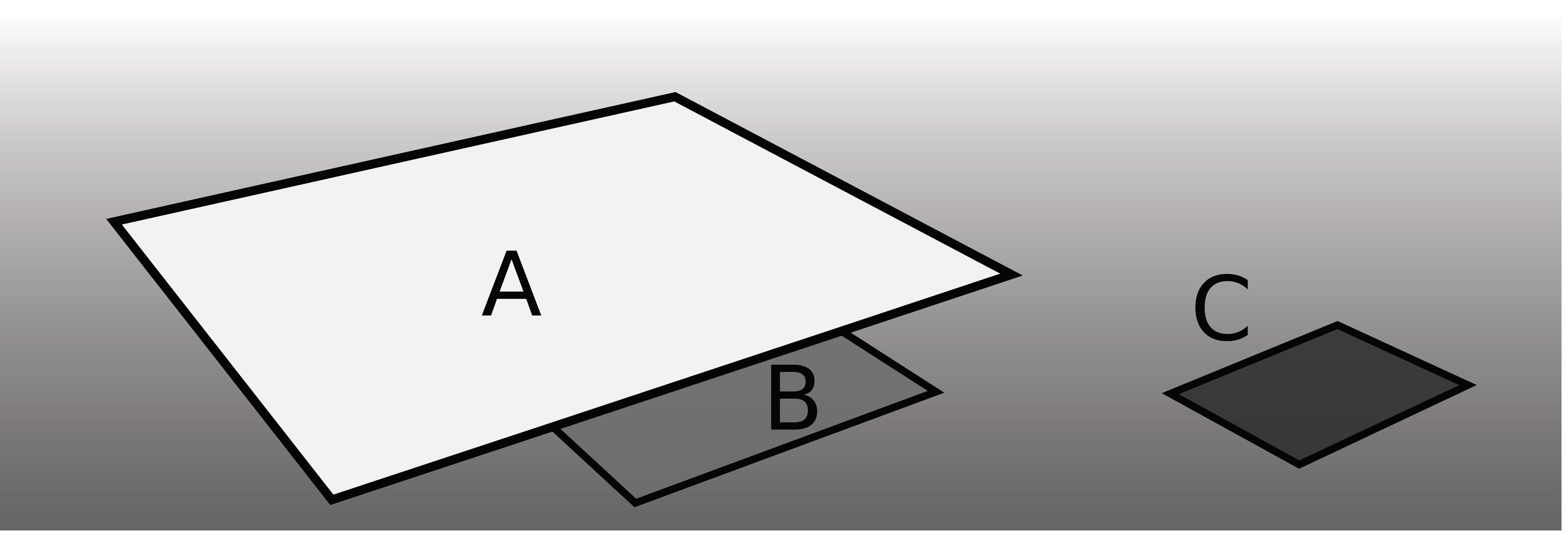
From multiple representation to higher dimensional GIS

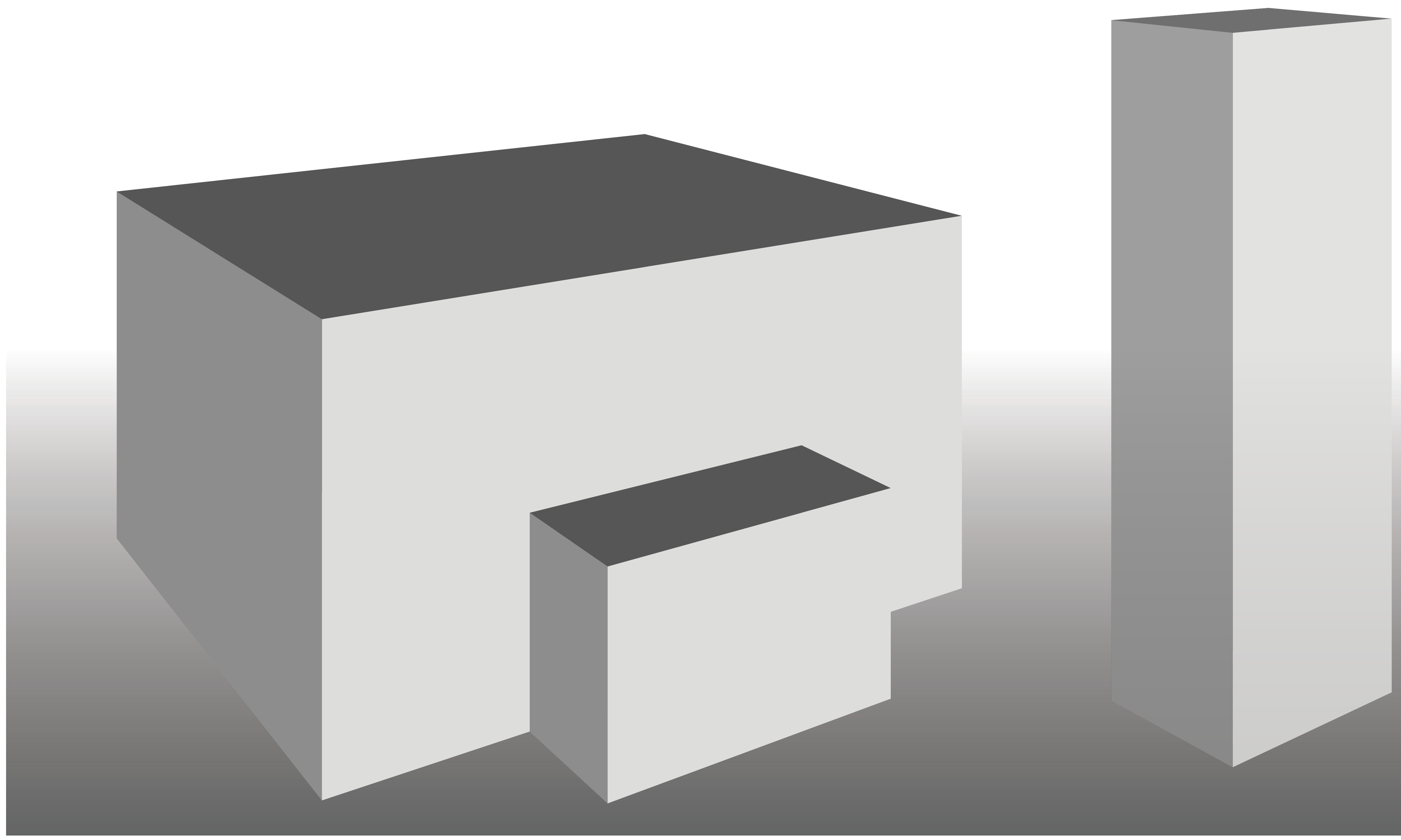
- Mathematically easy but tools are needed



2D to 3D extrusion

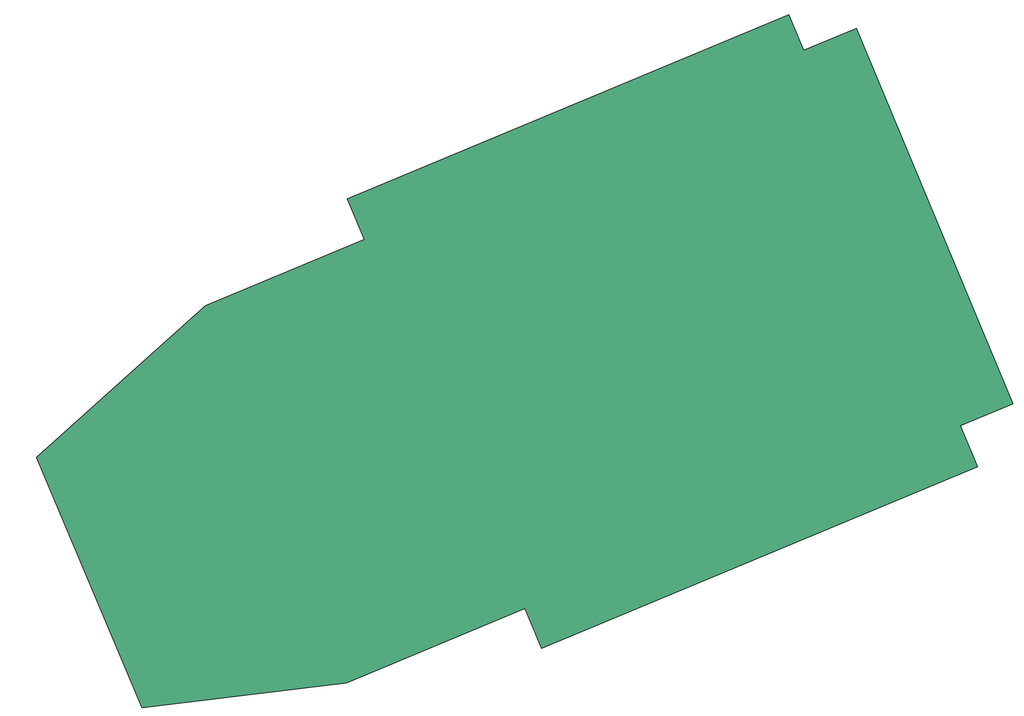
- 2D footprint + height → 3D

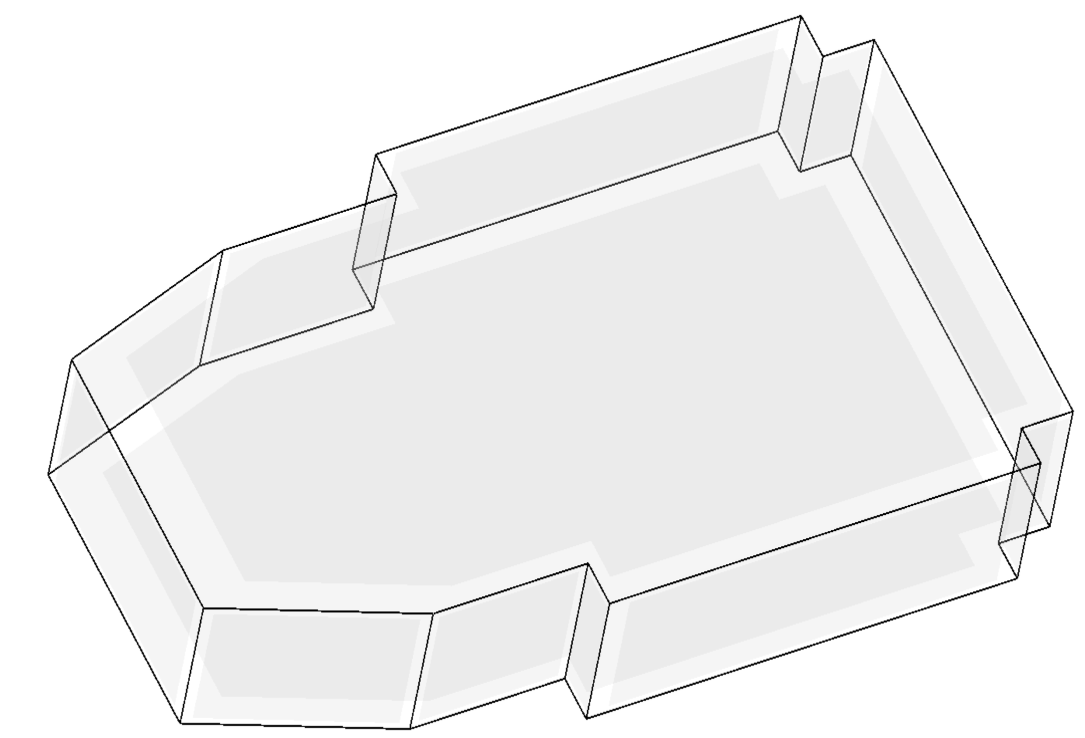


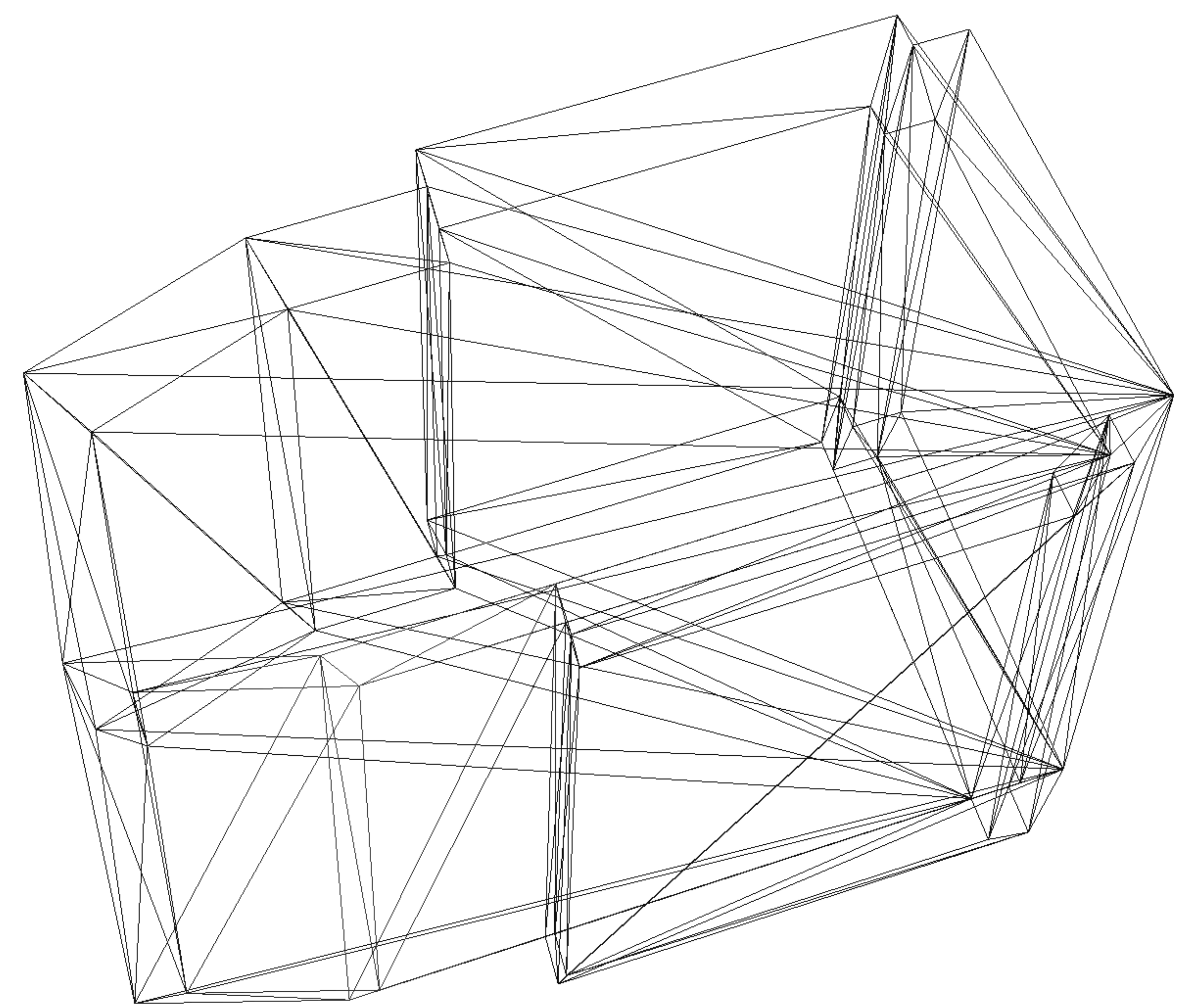


n-D extrusion

- (n-1)-D + range → n-D



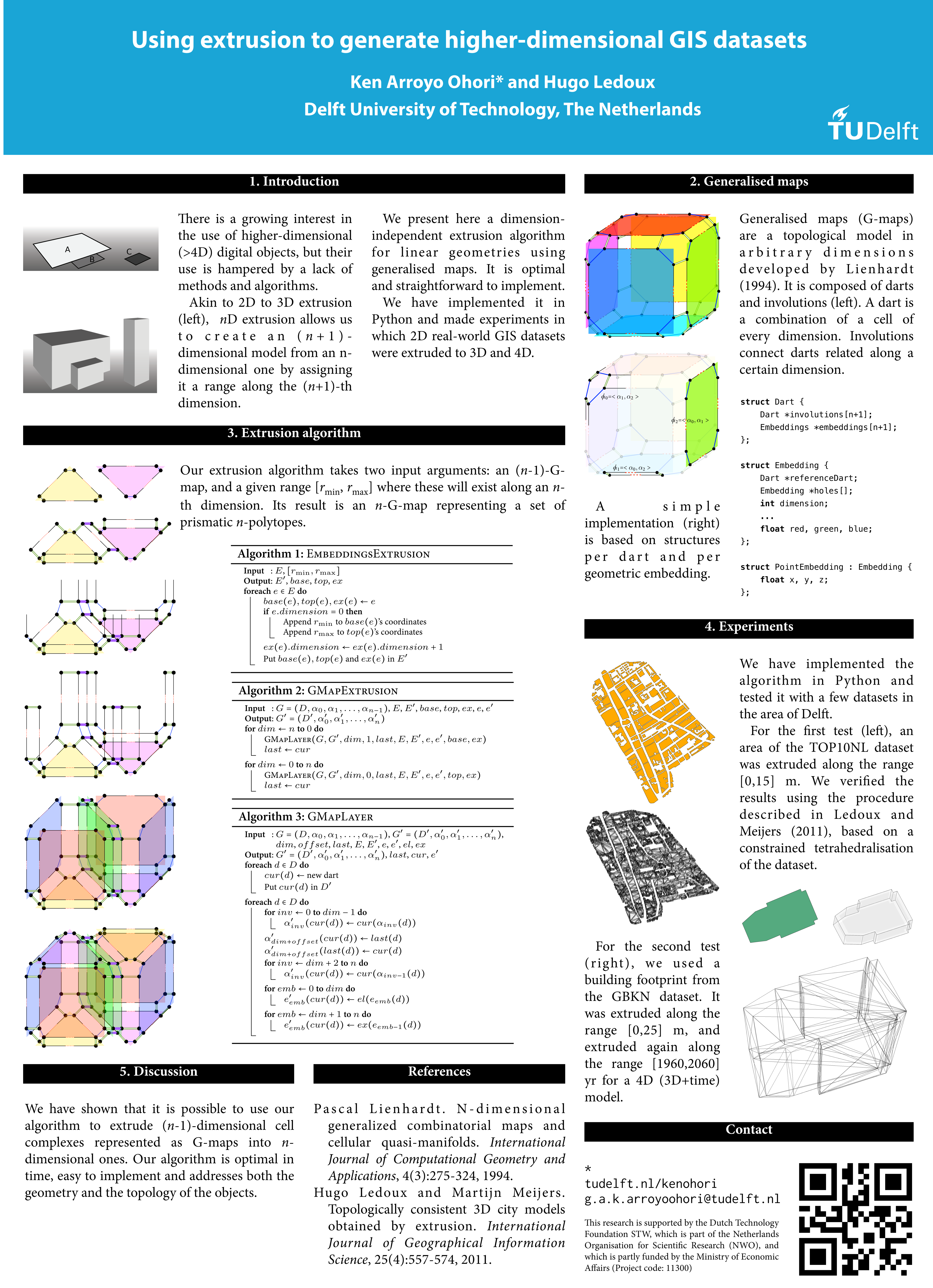




Thank you!

g.a.k.arroyoohori@tudelft.nl

TU Delft — Delft University of Technology



Using extrusion to generate higher-dimensional GIS datasets

Ken Arroyo Ohori* and Hugo Ledoux
Delft University of Technology, The Netherlands

TU Delft

1. Introduction

There is a growing interest in the use of higher-dimensional (>4D) digital objects, but their use is hampered by a lack of methods and algorithms.

Akin to 2D to 3D extrusion (left), nD extrusion allows us to create an $(n+1)$-dimensional model from an n-dimensional one by assigning it a range along the $(n+1)$-th dimension.

We present here a dimension-independent extrusion algorithm for linear geometries using generalised maps. It is optimal and straightforward to implement.

We have implemented it in Python and made experiments in which 2D real-world GIS datasets were extruded to 3D and 4D.

2. Generalised maps

Generalised maps (G-maps) are a topological model in arbitrary dimensions developed by Lienhardt (1994). It is composed of darts and involutions (left). A dart is a combination of a cell of every dimension. Involutions connect darts related along a certain dimension.

$\phi_0 = \langle \alpha_1, \alpha_2 \rangle$, $\phi_2 = \langle \alpha_0, \alpha_1 \rangle$, $\phi_1 = \langle \alpha_0, \alpha_2 \rangle$

A simple implementation (right) is based on structures per dart and per geometric embedding.

```
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;
};
```

3. Extrusion algorithm

Our extrusion algorithm takes two input arguments: an $(n-1)$-G-map, and a given range $[r_{min}, r_{max}]$ where these will exist along an n-th dimension. Its result is an n-G-map representing a set of prismatic n-polytopes.

Algorithm 1: EmbeddingsExtrusion

Input : E, $[r_{min}, r_{max}]$
Output: E', $base$, top, ex
foreach $e \in E$ **do**
  $base(e), top(e), ex(e) \leftarrow e$
  if $e.dimension = 0$ **then**
    Append r_{min} to $base(e)$'s coordinates
    Append r_{max} to $top(e)$'s coordinates
  $ex(e).dimension \leftarrow ex(e).dimension + 1$
  Put $base(e), top(e)$ and $ex(e)$ in E'

Algorithm 2: GMapExtrusion

Input : $G = (D, \alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, E, E', $base$, top, ex, e, e'
Output: $G' = (D', \alpha'_0, \alpha'_1, \ldots, \alpha'_n)$
for $dim \leftarrow n$ **to** 0 **do**
  GMapLayer$(G, G', dim, 1, last, E, E', e, e', base, ex)$
  $last \leftarrow cur$
for $dim \leftarrow 0$ **to** n **do**
  GMapLayer$(G, G', dim, 0, last, E, E', e, e', top, ex)$
  $last \leftarrow cur$

Algorithm 3: GMapLayer

Input : $G = (D, \alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, $G' = (D', \alpha'_0, \alpha'_1, \ldots, \alpha'_n)$, dim, $offset$, $last$, E, E', e, e', el, ex
Output: $G' = (D', \alpha'_0, \alpha'_1, \ldots, \alpha'_n)$, $last$, cur, e'
foreach $d \in D$ **do**
  $cur(d) \leftarrow$ new dart
  Put $cur(d)$ in D'
foreach $d \in D$ **do**
  for $inv \leftarrow 0$ **to** $dim - 1$ **do**
    $\alpha'_{inv}(cur(d)) \leftarrow cur(\alpha_{inv}(d))$
  $\alpha'_{dim+offset}(cur(d)) \leftarrow last(d)$
  $\alpha'_{dim+offset}(last(d)) \leftarrow cur(d)$
  for $inv \leftarrow dim + 2$ **to** n **do**
    $\alpha'_{inv}(cur(d)) \leftarrow cur(\alpha_{inv-1}(d))$
  for $emb \leftarrow 0$ **to** dim **do**
    $e'_{emb}(cur(d)) \leftarrow el(e_{emb}(d))$
  for $emb \leftarrow dim + 1$ **to** n **do**
    $e'_{emb}(cur(d)) \leftarrow ex(e_{emb-1}(d))$

4. Experiments

We have implemented the algorithm in Python and tested it with a few datasets in the area of Delft.

For the first test (left), an area of the TOP10NL dataset was extruded along the range [0,15] m. We verified the results using the procedure described in Ledoux and Meijers (2011), based on a constrained tetrahedralisation of the dataset.

For the second test (right), we used a building footprint from the GBKN dataset. It was extruded along the range [0,25] m, and extruded again along the range [1960,2060] yr for a 4D (3D+time) model.

5. Discussion

We have shown that it is possible to use our algorithm to extrude $(n-1)$-dimensional cell complexes represented as G-maps into n-dimensional ones. Our algorithm is optimal in time, easy to implement and addresses both the geometry and the topology of the objects.

References

Pascal Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3):275-324, 1994.

Hugo Ledoux and Martijn Meijers. Topologically consistent 3D city models obtained by extrusion. *International Journal of Geographical Information Science*, 25(4):557-574, 2011.

Contact

*
tudelft.nl/kenohori
g.a.k.arroyoohori@tudelft.nl

This research is supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs (Project code: 11300)