Using Extrusion to Generate Higher-dimensional GIS Datasets

Ken Arroyo Ohori Hugo Ledoux

This is an author's version of the paper. The authoritative version is:

Using extrusion to generate higher-dimensional GIS datasets. Ken Arroyo Ohori and Hugo Ledoux. In Craig Knoblock, Peer Kröger, John Krumm, Markus Schneider and Peter Widmayer (eds.), *SIGSPATIAL'13: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, November 2013, pp. 398–401. ISBN: 978-1-4503-2521-9.
DOI: 10.1145/2525314.2525447

Related source code is available at:
`https://github.com/kenohori/lcc-tools`

While there is a growing interest in the use of higher-dimensional (≥4D) digital objects built from complex real-world data, their construction is in practice hampered by a lack of methods and algorithms. We present in this paper a dimension-independent extrusion algorithm for linear geometries using generalised maps. It permits 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 our algorithm, which both optimal and straightforward to implement, using generalised maps as a base, and report on experiments in which 2D real-world GIS datasets were extruded to 3D and 4D.

1 Introduction

There is substantial and growing interest in the use of higher-dimensional (≥4D) digital objects that are built from complex real-world data. This form of representation is popular in the literature as a conceptually straightforward extension of 2D/3D models. In geographic information systems (GIS), these can be generated through the integration of 2D/3D space with time [Worboys, 1994] and/or scale [van Oosterom and Meijers, 2011], among others. However, actual use of these higher-dimensional objects has not yet been widely realised.

From a modeller's perspective, the first obstacle that is found is a general lack of high-level techniques to create these higher dimensional objects. For construction, existing techniques for higher-dimensional data are mostly limited to data composed of points, or geometric simplicial complexes [Karimipour et al., 2010] that are built with only points as input. This significantly restricts the class of objects that can be modelled.

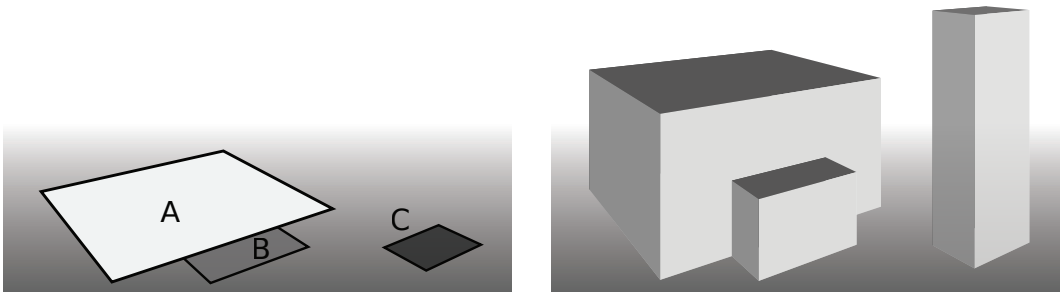
We are therefore interested in operations that allow the representation of cellular complexes, which are far less restrictive. A cellular complex, or cell complex, is a subdivision of space into cells, so that an i-dimensional cell (i-cell) is an object homeomorphic to an i-ball (e.g. point, segment, disk, ball, etc.). Every i-cell in the complex has a number of $(i-1)$-cells (faces) as its boundary, and these faces are also part of the complex.

Within the range of possible data structures to represent cell complexes, ordered topological models are particularly useful since the ordering information kept within them can be used to efficiently traverse the structure. In particular, we use here one such model: generalised maps [Lienhardt, 1994], which are briefly presented in Section 2. These allow us to represent a wider range of objects than cell complexes: objects of heterogeneous dimensions can be modelled. While there are other structures that are able to represent even larger classes of objects, such as non-manifolds [Bieri and Nef, 1988], these are significantly more complex, and are very difficult to implement in a dimension-independent manner. Moreover, we limit ourselves to cellular complexes consisting solely of *unique linear geometries* (i.e. points, line segments, polygons, polyhedrons, etc., from which no two are identical), which allow us to make assumptions that simplify their use significantly.

For GIS datasets, we believe that two construction techniques cover the majority of the cases found in practice: incremental construction and extrusion. Both are part of our ongoing work. The first one, incremental construction, builds individual i-cells in a cell complex from their unconnected $(i-1)$-dimensional faces, which involves correctly generating the topological relationships that exist between the different dimensional objects.

The second one, extrusion, is the focus of this paper and is presented in Section 3. In GIS, extrusion 'lifts' a n-dimensional model into $(n+1)$-dimensional space by assigning *each* n-dimensional object a range along which it exists, generating a set of prismatic polytopes. A common use is in 3D city modelling, where a 3D model of a building is commonly constructed from its footprint and a height value [Ledoux and Meijers, 2011], data that are easy to obtain in practice (e.g. cadastral parcels + airborne laser). An example of this is shown in Figure 1. If other dimensions are added, such as the construction and the demolition dates, the 3D models can be extruded into a four- or higher-dimensional model.

Within this paper, we limit ourselves to extruding all objects along the same range, something that in generalised maps is combinatorially equivalent to a cartesian product with an edge [Lienhardt et al., 2004]. However, unlike a cartesian product, our approach works on both geometry and topology, is simple to implement, and is designed to be extended to support extrusion to different ranges (per object), as expected in 2D/3D GIS. Like the cartesian product, our method is optimal in time and works in any dimension.



(a) Building footprints (b) After extrusion

Figure 1: Extruding a set of three building footprints by their heights is used to generate a 3D city model [Ledoux and Meijers, 2011].

We report on experiments that demonstrate how our algorithm works in Section 4, creating topologically consistent objects in any dimension. We finish with our conclusions and discussion in Section 5.

2 Generalised maps

Generalised maps (G-maps) are an ordered topological model in arbitrary dimensions developed by Lienhardt [1994] based on the concept of a combinatorial map, which is used to describe the 2D surfaces of 3D objects. They are roughly equivalent to the cell-tuple structure of Brisson [1989], but have been shown to be able to represent a wider class of objects known as cellular quasi-manifolds—a *combinatorial* simplicial decomposition of a quasi-manifold[1] which has been structured into cells. Note that unlike in a geometric simplicial decomposition (i.e. nD triangulation), constraints are not needed to make the decomposition conform to the shape of the object.

A generalised map is composed of two elements: *darts* and *involutions* between them (α). The precise definition of a dart is related to the underlying simplicial decomposition of the object, each dart being equivalent to a simplex in it. However, a dart can be intuitively seen as a unique combination of a specific cell of each dimension, all of which are incident to each other. Involutions are bijective operators connecting darts that are related along a certain dimension. In this manner, α_0 joins darts into edges, α_1 connects consecutive edges within a facet, α_2 connects adjacent facets within a volume, and so on.

More formally, a n-dimensional generalised map (n-G-map) is defined by a $(n+2)$-tuple $G = (D, \alpha_0, \dots, \alpha_n)$, where D is a non-empty set of darts, α_i is an involution (i.e. $\forall d \in D, \forall 0 \leq i \leq n, \alpha_i(\alpha_i(d)) = d$), and $\forall d \in D, \forall 0 \leq i < i+2 \leq j \leq n$, $\alpha_i(\alpha_j(d))$ is also an involution. These conditions respectively enforce symmetry and completeness when joining darts.

The aforementioned elements are sufficient to represent the combinatorial structure of a generalised map. However, to represent the geometry and other characteristics of the model, additional *embedding* structures are needed. Each of these structures stores the information of one cell of a certain dimension, so that an i-embedding contains the pertinent information of an i-cell. Since only linear geometries are required, only the 0-embeddings (point embeddings) are strictly necessary, which store the coordinates of each vertex [Arroyo Ohori et al., 2013].

3 Extrusion

Our extrusion algorithm takes two input arguments: an $(n-1)$-G-map, representing

[1] A specific combinatorial interpretation of the concept of a manifold. See Lienhardt [1994] for details.

a set of $(n-1)$-polytopes as a cell complex embedded into $(n-1)$-dimensional space, and a given range $[r_{\min}, r_{\max}]$ where these will exist along an n-th dimension, which is orthogonal to all others. Its result is an n-G-map representing a set of prismatic n-polytopes, the n-dimensional analogue of a set of prisms. Intuitively, this new cell complex consists of 'base' and 'top' $(n-1)$-cells (faces) constructed from the original cell complex, with $r_{\min}$ and $r_{\max}$ respectively added as n-th coordinates in their point embeddings. These 'base' and 'top' are joined by a set of additional prismatic faces linking corresponding $(n-2)$-cells (ridges) of the base and top cells.

As stated previously, the use of G-maps makes it possible to separate the combinatorial (topological) and embedding (geometric) aspects of an algorithm. Our extrusion algorithm follows this approach, operating on the combinatorial and embedding structures separately.

Combinatorially, the end result is an n-dimensional cell complex where every i-cell $\forall 0 \leq i < n$ in the input cell complex has been converted into two i-cells, respectively belonging to the base and top, and a prismatic $(i+1)$-cell lying between them. Therefore, extruding an i-th dimensional embedding (i-embedding) results in two i-embeddings (base and top) and one $(i+1)$-embedding. In the case of linear geometries this is simpler: only when extruding a 0-embedding two additional 0-embeddings are created, one with an appended $r_{\min}$ coordinate, and one with an $r_{\max}$ one. The extrusion algorithm for embeddings then receives a set of embeddings E and a range $[r_{\min}, r_{\max}]$, and returns a set of extruded embeddings E', as well as three functions $E \to E'$ linking corresponding unextruded (input) and extruded (output) embeddings: *base*, *top* and *ex*. It is presented in Algorithm 1.

Our algorithm to generate the combinatorial structure then works by iteratively generating 'layers' of darts, assigning them their correct embeddings, and linking them appropriately. The relationships between the darts and their corresponding embeddings are expressed in terms of the input

Algorithm 1: EMBEDDINGSEXTRUSION

```
Input  : E, [r_min, r_max]
Output: E', base, top, ex
foreach e ∈ E do
    base(e), top(e), ex(e) ← 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 ← ex(e).dimension + 1
    Put base(e), top(e) and ex(e) in E'
```

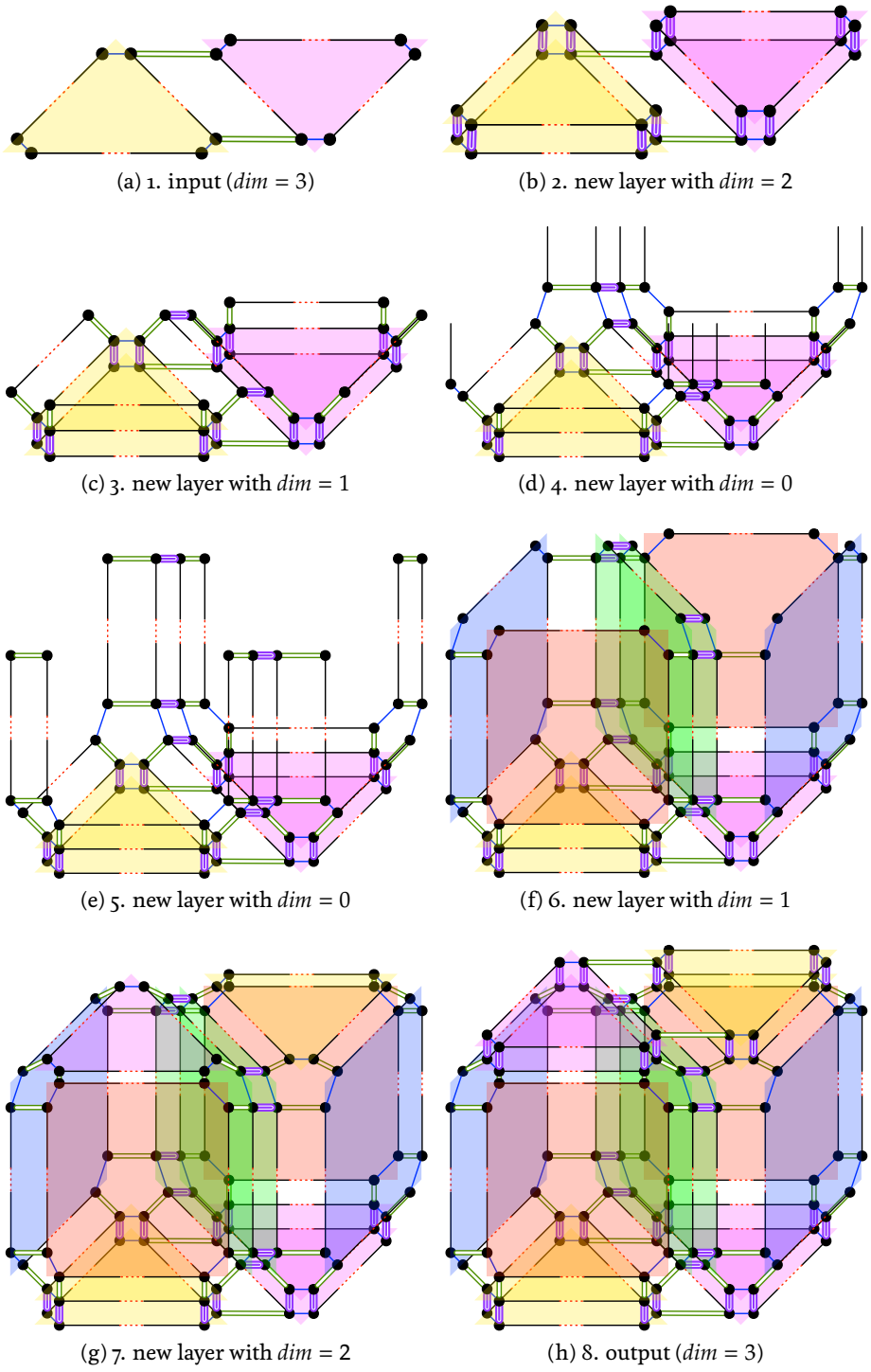
(unextruded) generalised map. In this manner, the input consists of an $(n-1)$-G-map $G = (D, \alpha_0, \ldots, \alpha_{n-1})$, the sets of embeddings E and E', the previously generated functions *base*, *top* and *ex*, and functions $e_i : D \to E$ and $e'_i : D' \to E'$ connecting darts to their corresponding i-embeddings. The output then consists of an n-G-map $G' = (D', \alpha'_0, \ldots, \alpha'_n)$. Observe that G' has one more involution than G. The procedure to extrude the combinatorial structure is presented in Algorithms 2 (complete) and 3 (for one layer), and a layer-by-layer example is shown in Figure 2.

If we denote the embedding of an input i-cell as e, we will identify the base i-embedding of the output as $base(e)$, the top i-embedding as $top(e)$, and the $(i+1)$-embedding between them as $ex(e)$. If we denote an input dart as d, we will denote its corresponding dart in the last layer of the output as $last(d)$, the one in the current layer as $cur(d)$, the dart linked to d by an i-involution as $\alpha_i(d)$, and the i-embedding of d as $e_i(d)$.

Algorithm 2: GMAPEXTRUSION

```
Input  : G = (D, α_0, α_1, ..., α_{n-1}), E, E', base, top, ex, e, e'
Output: G' = (D', α'_0, α'_1, ..., α'_n)
for dim ← n to 0 do
    GMAPLAYER(G, G', dim, 1, last, E, E', e, e', base, ex)
    last ← cur
for dim ← 0 to n do
    GMAPLAYER(G, G', dim, 0, last, E, E', e, e', top, ex)
    last ← cur
```

Extruding an n-G-map into an $(n+1)$-G-map involves the creation of $2n+2$ layers, each of which has the same number of darts as the input generalised map. Also, every embedding is extruded into 3 new embeddings (*top*, *base* and *ex*).



(a) 1. input ($dim = 3$)

(b) 2. new layer with $dim = 2$

(c) 3. new layer with $dim = 1$

(d) 4. new layer with $dim = 0$

(e) 5. new layer with $dim = 0$

(f) 6. new layer with $dim = 1$

(g) 7. new layer with $dim = 2$

(h) 8. output ($dim = 3$)

Figure 2: The steps in the lifting of a 2-G-map of two adjacent triangles into a 3-G-map of two adjacent triangular prisms. Every step represents a new layer of 12 darts (which is the number of darts in the input triangles).

Algorithm 3: GMapLayer

Input : $G = (D, \alpha_0, \alpha_1, \dots, \alpha_{n-1})$, $G' = (D', \alpha'_0, \alpha'_1, \dots, \alpha'_n)$, dim, $offset$, $last$, E, E', e, e', el, ex
Output: $G' = (D', \alpha'_0, \alpha'_1, \dots, \alpha'_n)$, $last$, 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**
    $e'_{emb}(cur(d)) \leftarrow ex(e_{emb-1}(d))$

Since we generate only the required number of darts and embeddings, and since every one of these is generated in constant time (assuming constant time access to specific darts in the base, last and current layers of darts), our algorithm is optimal in time. Indeed, it is linear in the number of darts in the input (and thus output) generalised map.

4 Examples and experiments

We have implemented the extrusion algorithm in Python and tested it with a few 2D datasets in the area of Delft. These were read using the Python Shapefile Library (pyshp)[2] and put into a 2-G-map.

For the first test, an area of the TOP10NL[3] dataset comprising the footprints of the 183 buildings in the Delft city centre with no elevation (Fig. 3a) was extruded along the range $[0, 15]$ (Fig. 3b). This created a pseudo 3D city model of the area, where all buildings have a height of 15 metres. We have experimentally verified that our algorithm and its implementation are correct by using the procedure described in Ledoux and Meijers [2011]. The model was thus written into a TetGen[4] .poly file and tetrahedralised using the same software. Since the tetrahedralisation algorithm [Si and Gärtner, 2005] reports intersecting cells and removes the facets that do not bound a 3-cell, we are able to validate our model using this method. The model was tetrahedralised successfully with no buildings missing.

For the second test, we used the footprint of the Aula Congress Centre in the TU Delft campus from the GBKN[5] dataset (Fig. 4a). It was put into a 2-G-map with 28 darts, 14 0-cells, 14 1-cells and 1 2-cell, which was extruded along the range $[0, 25]m$ to construct a 3D model (Fig. 4c), and extruded again along the range $[1960, 2060]$ years for a 4D (3D+time) model. The model was created with no reported errors, resulting in 1344 ($28 \times 6 \times 8$) darts, 56 0-cells, 112 1-cells, 74 2-cells, 18 3-cells and 1 4-cell. We tested it by verifying that the number of darts and embeddings in it matched the theoretical values, by performing simple validation tests on its geometry and topology, and visually using a 4D to 3D parallel projection.

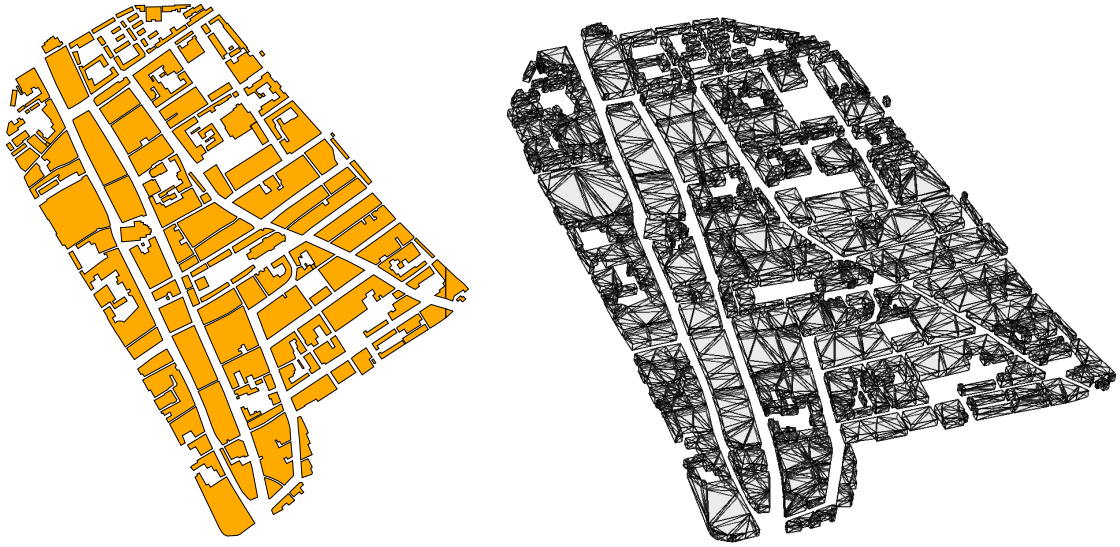
5 Discussion

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 and easy to implement. It has important benefits over the only other known algorithm (a cartesian product with an edge [Lienhardt et al., 2004]): (i) both geometry and topology are addressed; (ii) it can be extended so that multiple ranges are considered when extruding (by detecting overlapping ranges between neighbouring darts); (iii) it can handle datasets that are larger than memory since only three layers of darts need to be kept in main memory at a time.

6 Acknowledgments

This research is supported by the Dutch Technology Foundation STW, which is part

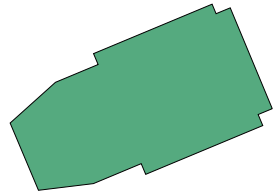
[2] http://code.google.com/p/pyshp/
[3] http://www.kadaster.nl/top10nl/
[4] http://tetgen.berlios.de
[5] http://www.gbkn.nl



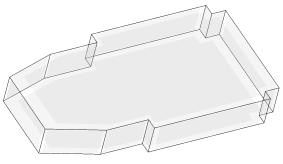
(a) In the TOP10NL dataset

(b) After extrusion and tetrahedralisation (perspective projection of 0-, 1- and 2-cells only)

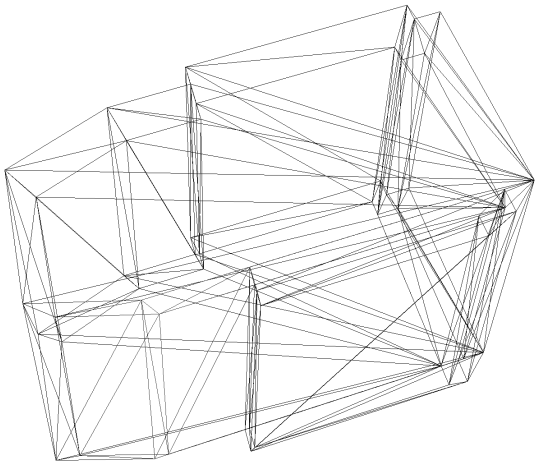
Figure 3: Extruding the Delft city centre.



(a) Input footprint



(c) After the first extrusion



(b) After the second extrusion (double perspective projection of 0- and 1-cells only)

Figure 4: Extruding the Aula Congress Centre in the TU Delft campus.

of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs (Project code: 11300).

References

Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. Modelling higher dimensional data for GIS using generalised maps. In B. Murgante, S. Misra, M. Carlini, C. Torre, H.Q. Nguyen, D. Taniar, B. Apduhan, and O. Gervasi, editors, *Computational Science and Its Applications — ICCSA 2013*, volume 7971 of *Lecture Notes in Computer Science*, pages 526–539. Springer Berlin Heidelberg, Ho Chi Minh City, Vietnam, June 2013.

H. Bieri and W. Nef. Elementary set operations with d-dimensional polyhedra. In *Computational Geometry and its Applications*, volume 333 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin / Heidelberg, 1988.

Erik Brisson. Representing geometric structures in d dimensions: topology and order. In *Proceedings of the 5th annual symposium on Computational geometry*, pages 218–227, New York, NY, USA, 1989. ACM.

Farid Karimipour, Mahmoud R. Delavar, and Andrew U. Frank. A simplex-based approach to implement dimension independent spatial analyses. *Computers & Geosciences*, 36(9):1123–1134, September 2010.

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

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

Pascal Lienhardt, Xavier Skapin, and Antoine Bergey. Cartesian product of simplicial and cellular structures. *International Journal of Computational Geometry and Applications*, 14(3):115–159, 2004.

Hang Si and Klaus Gärtner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, September 2005.

Peter van Oosterom and Martijn Meijers. Towards a true vario-scale structure supporting smooth-zoom. In *Proceedings of the 14th ICA/ISPRS Workshop on Generalisation and Multiple Representation, Paris*, 2011.

Michael F. Worboys. A unified model for spatial and temporal information. *The Computer Journal*, 37(1):26–34, 1994.