Defining simple nD operations based on prismatic nD objects

Ken Arroyo Ohori Hugo Ledoux Jantien Stoter

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

Defining simple nD operations based on prismatic nD objects. Ken Arroyo Ohori, Hugo Ledoux and Jantien Stoter. In E. Dimopoulou and P. van Oosterom (eds.), *11th 3D Geoinfo Conference*, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences IV-2/W1, ISPRS, Athens, Greece, October 2016, pp. 155–162. ISSN: 2194-9042 (Print), 2194-9050 (Internet and USB).
DOI: 10.5194/isprs-annals-IV-2-W1-155-2016

An alternative to the traditional approaches to model separately 2D/3D space, time, scale and other parametrisable characteristics in GIS lies in the higher-dimensional modelling of geographic information, in which a chosen set of non-spatial characteristics, e.g. time and scale, are modelled as extra geometric dimensions perpendicular to the spatial ones, thus creating a higher-dimensional model. While higher-dimensional models are undoubtedly powerful, they are also hard to create and manipulate due to our lack of an intuitive understanding in dimensions higher than three. As a solution to this problem, this paper proposes a methodology that splits the creation and manipulation of an nD objects in three steps: (i) constructing simple nD objects based on nD prismatic polytopes—analogous to prisms in 3D—, (ii) defining simple modification operations at the vertex level, and (iii) simple postprocessing to fix errors introduced in the model. In particular, we show how two sets of operations can be defined and implemented in a dimension-independent manner using this methodology: the most common transformations (i.e. translation, scaling and rotation) and the collapse of objects.

1 Introduction

The traditional approaches to model 2D/3D space, time and scale in GIS are mostly based on adaptations to well-known 2D data structures, such as the DCEL [Muller and Preparata, 1978] and the quad-edge [Guibas and Stolfi, 1985]. Many '3D' GIS internally represent objects using a 2.5D structure, essentially treating the third dimension as an attribute, or represent individual 3D objects only implicitly through the 2D surface that separates their interior from their exterior. Spatiotemporal GIS keep multiple representations of 2D structures [Armstrong, 1988], each at a different point in time, or a list of changes per object [Worboys, 1992, Peuquet, 1994], while multi-scale datasets generally consist of independent datasets at each scale with some identifiers that link equivalent objects between datasets [Friis-Christensen and Jensen, 2003, Stoter et al., 2014].

An alternative to this approach lies in the *higher-dimensional modelling of geographic information* [Arroyo Ohori, 2016], where a chosen set of non-spatial characteristics, e.g. time and scale [van Oosterom and Stoter, 2010], are modelled as true geometric dimensions in addition to the spatial ones. For instance, the complete history of a set of 3D objects in time and all of their possible representations at various scales can be modelled as a single 5D model. Mathematically, a higher-dimensional model corresponds to the definition of an n-dimensional cell complex (Section 2.1), which can be directly implemented in a computer using a variety of data structures.

Higher-dimensional models are undoubtedly space-intensive, but they are also very powerful: they provide a simple and consistent way to store the geometry, attributes and topological relationships between any objects of any dimension. However, *one of the main problems of higher-dimensional models is that they are not intuitive.* While we are used to solving problems in 2D and 3D, and we thus have an intuitive understanding of 2D and 3D space, and of operations on 2D and 3D objects (e.g. the Euler operators typically used to create polyhedra), we do not have similar intuitive notions and experiences for nD objects.

As a partial solution to this problem, this paper proposes to use nD prismatic polytopes—analogous to prisms in 3D—as a base for the creation and manipulation of nD models. Prismatic polytopes essentially represent objects that are unchanged along a single dimension. By applying a modification operation to the vertices of the top or bottom facet of a prismatic polytope, they can be used to model many common geographic phenomena, such as objects that are moving and/or changing shape in time, or being generalised as their LOD is reduced. Moreover, unlike arbitrary nD objects, nD prismatic polytopes can be easily created based on nD extrusion, as is explained in Section 2.2.

Our methodology (Section 3) splits the creation or modification of a set of nD objects into three steps: (i) creating simple nD objects based on nD prismatic polytopes, (ii) defining simple modification operations at the vertex level, and (iii) simple postprocessing to fix the errors introduced in the model.

Within this paper, we describe in detail two concrete examples of operations defined based on our general methodology: the most common transformations applied to GIS objects (i.e. translation, scaling and rotation) in Section 4.1, and collapsing cells in Section 4.2. We finish the paper with a discussion on the possibilities of these and other nD operations based on the same methodology in Section 5.

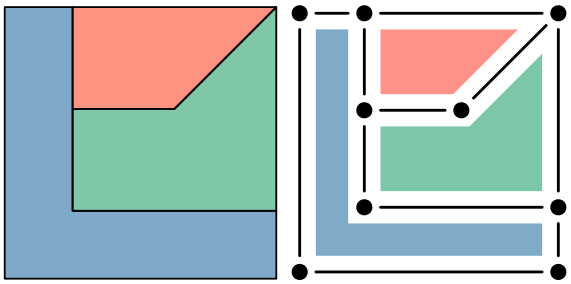
2 Related work

2.1 nD cell complexes and their implementation

Hereafter follows a simple intuitive definition of n-dimensional cell complexes and their related terms as used in this paper. More correct (but harder) definitions

are usually based on induction. See e.g. Fomenko [1990] or Hatcher [2002].

An n-dimensional cell complex is a structure made of connected *cells* of dimensions from zero up to n, where an i-dimensional cell (i-cell), $0 \leq i \leq n$, is an object homeomorphic to an open i-ball (i.e. a 0D point, 1D open arc, 2D open disk, 3D open ball, etc.)[1]. 0-cells are commonly known as vertices, 1-cells as edges, 2-cells as faces and 3-cells as volumes. For GIS purposes and considering only linear geometries, 0-cells are used to model points, 1-cells to model line segments, 2-cells to model polygons, 3-cells to model polyhedra, and so on. Figure 1 shows a diagrammatic description of a set of three simple polygons as a 2D cell complex.



(a) Polygons (b) Cell complex

Figure 1: (a) A set of polygons can be represented as (b) a 2D cell complex consisting of 0-cells (black disks), 1-cells (black lines), and 2-cells (coloured polygons).

An i-cell ($i > 0$) is bounded by a structure of j-cells, $j < i$, which are collectively known as its *boundary*. A j-dimensional *face* (j-face) of an i-cell is a j-cell, $j \leq i$, that is part of the boundary of the i-cell. A *facet* of an i-cell is an $(i-1)$-face of the i-cell, and a *ridge* of an i-cell is an $(i-2)$-face of the i-cell. A facet of a polyhedron is thus one of the polygons on its boundary, and a ridge of the polyhedron is one of the line segments on its boundary.

Various surveys describe the data structures that can be used to represent n-dimensional cell complexes [Čomić and de Floriani, 2012, Arroyo Ohori et al., 2015b]. Possible data structures include incidence graphs [Rossignac and O'Connor, 1989, Masuda, 1993, Sohanpanah, 1989], Nef polyhedra [Bieri and Nef, 1988], and ordered topological models [Brisson, 1993, Lienhardt, 1994]. nD combinatorial maps [Damiand and Lienhardt, 2014] are particularly promising [Arroyo Ohori et al., 2015b], as they are reasonably compact, can elegantly handle attributes for the cells of every dimension and have an excellent freely available implementation in CGAL[2]. They are thus used in order for the implementations developed in this paper.

2.2 Prismatic polytopes and their creation using extrusion

A *prismatic polytope*[3] is the higher-dimensional analogue of a 2D rectangle or a 3D prism. Using the terminology of a cell complex and just like in rectangles and prisms, a prismatic polytope can be defined intuitively as a cell that is bounded by a set of facets: two identical 'top' and 'bottom' facets, and a set of other facets that join corresponding ridges of the top and bottom facets. Figure 2 shows an icosahedral prism, an example of a 4D prismatic polytope with icosahedron-shaped (equivalent to Figure 2b) top and bottom facets.

Extrusion is a widely used technique in GIS to construct simple 3D models from 2D+height data [Ledoux and Meijers, 2011]. Starting from a set of non-overlapping polygons and a height interval associated to each of them, it generates a set of non-overlapping prisms by considering that each polygon exists all along its related interval. As shown in Figure 3, a set of building footprints and associated heights can thus be extruded into a set of simple prismatic buildings. Figure 4 shows a different 3D use case in which the third dimension represents time. The history of a set of building footprints is thus represented by a set of extruded polyhedra.

[1] Note that these objects do not contain their boundary, unlike closed i-balls which do contain it.

[2] `http://www.cgal.org`

[3] A polytope is analogous to a polygon in 2D or polyhedron in 3D.

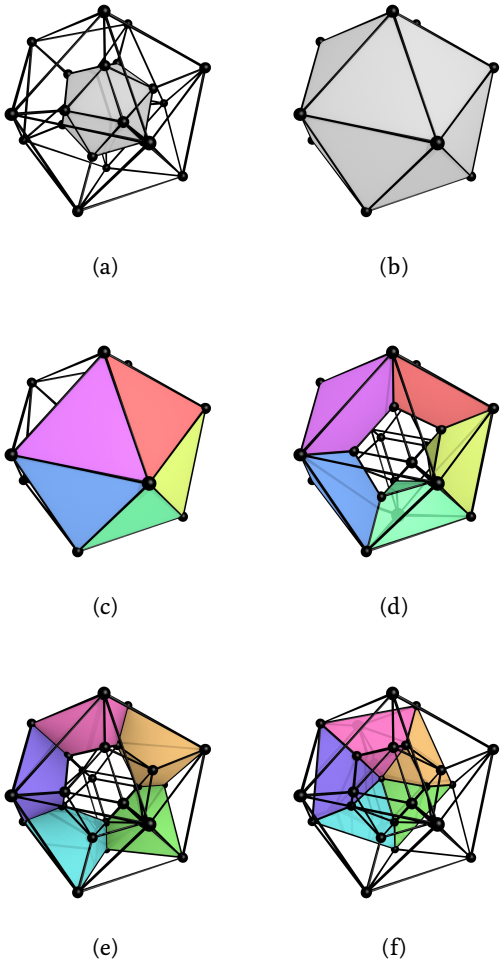


Figure 2: The 24 vertices, 72 edges, 70 faces and 22 volumes bounding an icosahedral prism in a stereographic projection from 4D to 3D, shown here in parts for clarity. Due to the projection used, the fourth dimension corresponds to an inwards-outwards axis. In this manner, the 'top' and 'bottom' facets (i.e. volumes in this 4D case) of the prismatic polytope are visualised as (a) the inner and (b) the outer icosahedra. Meanwhile, (c–f) the other facets are triangular prisms that join corresponding ridges (i.e. faces in this 4D case) of the top and bottom facets.

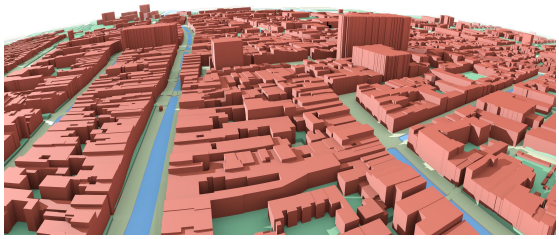


Figure 3: A simple 3D city model created by extruding buildings.

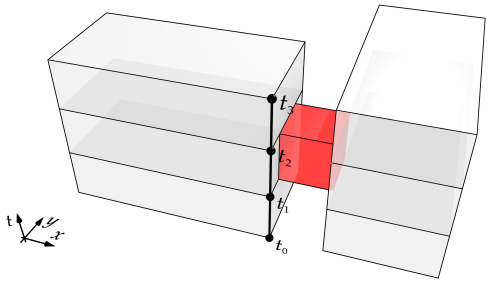


Figure 4: A 2D+time model generated using extrusion. The red volume represents the footprint of a corridor that existed between times t_1 and t_2. The left and right building footprints were thus extruded along $[t_0, t_3]$ and the corridor footprint along $[t_1, t_2]$.

While extrusion is usually used in GIS only in this 2D-to-3D form, it can be straightforwadly generalised to higher dimensions [Arroyo Ohori et al., 2015c]: given a set of $(n-1)$-dimensional objects in the form of a $(n-1)$-dimensional cell complex, and a set of intervals per $(n-1)$-cell in the complex, it is possible to extrude the cells along the n-th dimension, thus creating a n-dimensional cell complex. Considering linear geometries only, this perfectly corresponds to the creation of a set of n-dimensional prismatic polytopes from a set of $(n-1)$-dimensional polytopes.

When all the $(n-1)$-cells are extruded along a single identical interval, this operation can be computed purely combinatorially (i.e. without any geometric tests) and is equivalent to the Cartesian product of the cells in the complex with an edge. This was described for the case of generalised maps—an ordered topological model—by

Lienhardt et al. [2004]. Moreover, such a procedure can be repeated in order to support more than one interval, and given some preprocessing of the intervals by subdividing them into non-overlapping parts, it can also be used with different intervals per cell, as was suggested by Ferrucci [1993]. Arroyo Ohori et al. [2015c] describes a more complex method to do so in the context of a generalised or combinatorial map, but one that minimises the total number of generated cells. For the purposes of this paper, any of these methods can be used as a base to generate the required prismatic polytopes.

3 Methodology

We propose a simple methodology for the creation and manipulation of nD objects based on three steps: (i) an $(n-1)$-dimensional object or set of objects is extruded into a set of n-dimensional prismatic polytopes, (ii) a modification operation is applied to the vertices of the top or bottom facets of each prismatic polytope, and (iii) simple postprocessing is used to fix the errors that were introduced in the model. The manipulation of an nD object corresponds to the last two steps only. If a topological data structure is used, the last step includes recomputing the topological relationships between the cells that have changed. The reasoning behind each of these steps is described in more detail below.

A n-dimensional prismatic polytope, where the n-th dimension represents a non-spatial characteristic such as time or scale, essentially represents *a lack of change*. It can be thus equivalent to an $(n-1)$-dimensional object whose geometry does not change along a period of time or whose representation is valid along an interval of levels of detail (LODs) [Arroyo Ohori et al., 2015a]. While extrusion might seen unduly restrictive, it is able to produce a large set of useful objects (see e.g. the complex examples in Ferrucci [1993]), and it remains simple and intuitive even in higher dimensions.

Moreover, the facets of a prismatic polytope have a few properties that make them appealing as a base for further operations:

- the top and bottom facets are parallel to the $(n-1)$-dimensional subspace defined by the axes of the $n-1$ coordinate system of the original $(n-1)$-dimensional cell complex—or alternatively, the top and bottom facets are orthogonal to the newly defined n-th axis;
- the 'side' facets connecting corresponding ridges of the top and bottom facets are orthogonal to the $(n-1)$-dimensional subspace defined by the axes of the $n-1$ coordinate system of the original $(n-1)$-dimensional cell complex—or alternatively, the side facets are parallel to the newly defined n-th axis.

Together, these properties mean that it is possible to apply various modification operations by applying them directly to the vertices of the top or bottom facet of a prismatic polytope. Or in the case of multiple extrusions (i.e. multiple prisms stacked on top of each other), to the vertices of any facet that is orthogonal to the n-th axis. As long as these modifications do not move the vertices of the facet out of the hyperplane[4] where it lies, and the modifications do not cause the side facets to intersect (e.g. a 180° rotation), the resulting polytope should be a reasonably well behaved object. That is, it should contain only minor errors that are relatively easy to fix, either by minor modifications to the process or by using simple preprocessing or postprocessing steps (e.g. moving some vertices to remove self-intersections).

For instance, a smooth non-self intersecting rotation of any angle can be obtained by applying several small extrusions and rotations that together define the complete rotation—thus generating a screw shape rather than a twisted prism. In this manner, even rotations of more than 360° are possible. Meanwhile, many other types of operations causing self-intersections can be fixed by first refining the input cells into smaller

[4] i.e. an $(n-1)$-dimensional unbounded linear subspace in an n-dimensional setting, such as a line in 2D and a plane in 3D.

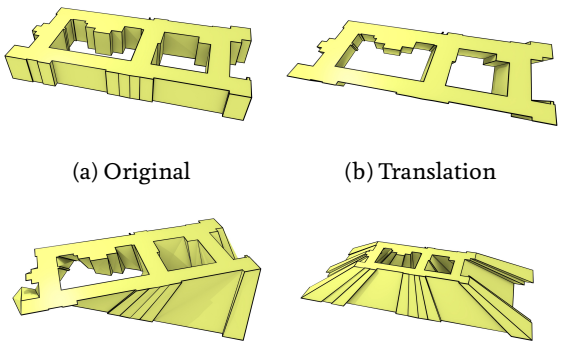
cells whose neighbourhood is expected to be well-behaved. Among other examples, this preprocessing step can solve problems with flexible objects that do not morph uniformly (such as animated 3D models, which might otherwise have self-intersections), as well as non-linear transformations.

4 Two sets of operations on prismatic polytopes

In order to provide clear examples of nD operations that can be defined using the methodology described above, this section describes two such operations in detail and in a dimension-independent form: the most common transformations in Section 4.1, and collapsing cells in Section 4.2. The former represents the base of most interactive geometric modellers, while the latter together with the former provide a simple way to define a set of simple dimension-independent map generalisation operators.

4.1 Simple transformations

Starting from a set of n-dimensional prismatic polytopes, where every vertex is embedded in a location in $\mathbb{R}^n$, it is possible to define a set of basic transformations to manipulate them or parts of them (e.g. their top or bottom $(n-1)$-dimensional facets) simply by applying these transformations to the coordinates of their corresponding vertices. This section thus gives a simple dimension-independent formulation of the most important transformations that are typically applied to 2D/3D objects in GIS: translation, scaling and rotation. In this way, objects that are moving or changing in shape (in certain ways) can be modelled. Figure 5 shows the result of these transformations being applied to the vertices of the top facet of a prismatic polyhedron, which was generated by extruding a building footprint with two holes. Note how the two holes in the original polygon become *genera*[5] of the extruded polyhedron's surface.



(a) Original (b) Translation

(c) Rotation (d) Scaling

Figure 5: Applying transformations to the top face of (a) a prism results in: (b) a parallelepiped in the case of a translation, (c) a *twisted prism* in the case of a rotation, and (d) a *frustum* in the case of scaling.

Translating of a set of points in $\mathbb{R}^n$ can be easy expressed as a sum with a vector $t = [t_0, \dots, t_n]$, or alternatively as a multiplication with a matrix using homogeneous coordinates, which is defined as:

$$T = \begin{bmatrix} 1 & 0 & \cdots & 0 & t_0 \\ 0 & 1 & \cdots & 0 & t_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & t_n \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

For instance, it is often useful to apply a multiplication with a *centering matrix* [Marden, 1996, §3.2], which moves a dataset to a position around the origin. Such a matrix would be defined as $\mathbb{I}_n - \frac{1}{n}M_1$, where $\mathbb{I}_n$ is an $n \times n$ identity matrix and M_1 is an $n \times n$ matrix where all entries are set to 1.

Scaling is similarly simple. Given a vector $s = [s_0, s_1, \dots, s_n]$ that defines a scale factor per axis (which in the simplest case can be the same for all axes), it is possible to define a matrix to scale an object as:

[5] Plural of *genus*. Intuitively, the genus of a surface indicated how many holes or handles it has. For instance, a donut has genus 1.

$$S = \begin{bmatrix} s_0 & 0 & \cdots & 0 \\ 0 & s_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_n \end{bmatrix}$$

Rotation is somewhat more complex. Rotations in 3D are often conceptualised intuitively as rotations *around an axis*. As there are three degrees of rotational freedom in 3D, combining three such elemental rotations can be used to describe any rotation in 3D space. Most conveniently, these three rotations can be performed respectively around the x, y and z axes, such that a point's coordinate on the axis being rotated remains unchanged. This is a very elegant formulation, but this view of the matter is only valid in 3D.

A more correct way to conceptualise rotations is to consider them as rotations *parallel to a given plane* [Hollasch, 1991], such that a point that is continuously rotated (without changing rotation direction) will form a circle that is parallel to that plane. This view is valid in 2D (where there is only one such plane), in 3D (where a plane is orthogonal to the usually defined axis of rotation) and in any higher dimension. Incidentally, this shows that the degree of rotational freedom in nD is given by the number of possible combinations of two axes (which define a plane) on that dimension [Hanson, 1994], i.e. $\binom{n}{2}$. A general rotation in any dimension can also be seen as a sequence of elementary rotations, although the total number of these rotations that need to be performed increases significantly.

Consider the 2D rotation matrix R_{xy} that rotates points in $\mathbb{R}^2$ parallel to the xy plane:

$$R_{xy} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Based on it, it is possible to obtain the three 3D rotation matrices to rotate points in $\mathbb{R}^3$ around the x, y and z axes, which correspond to the rotations parallel to the yz, zx and xy planes[6]. These would consist of an identity row and column that preserves the coordinate of a particular axis and rotates the coordinates of the other two, resulting in the following three 3D rotation matrices:

$$R_{yz} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_{zx} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_{xy} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Similarly, in a 4D coordinate system defined by the axes x, y, z and w, it is possible to define six 4D rotation matrices, which correspond to the six rotational degrees of freedom in 4D [Hanson, 1994]. These respectively rotate points in $\mathbb{R}^4$ parallel to the xy, xz, xw, yz, yw and zw planes:

$$R_{xy} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{xz} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{xw} = \begin{bmatrix} \cos\theta & 0 & 0 & -\sin\theta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\theta & 0 & 0 & \cos\theta \end{bmatrix} \quad R_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{yw} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & -\sin\theta \\ 0 & 0 & 1 & 0 \\ 0 & \sin\theta & 0 & \cos\theta \end{bmatrix} \quad R_{zw} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & \sin\theta & \cos\theta \end{bmatrix}$$

This scheme of a set of elementary rotations can be easily extended to any dimension, always considering a rotation matrix as a transformation that rotates two coordinates of every point and maintains all other coordinates. An alternative to this could be to apply more than one rotation at a time [van Elfrinkhof, 1897]. However, for an application expecting user interaction, it might be more intuitive to rely on an arbitrarily

[6] Note the order of the three rotation planes given here, which results from *omitting* the axes x, y and z (in that order). Note also the order of the two axes in zx, which follows the right hand rule and defines the signs of the sines in the rotation matrices.

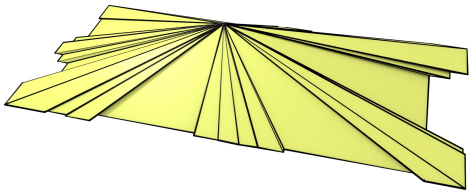
defined rotation plane that does not correspond to specific axes, e.g. by defining such a plane through a triplet of linearly independent points [Hanson, 1994].

Unlike the cases of translation and scaling presented above, a rotation applied to the vertices of the top or bottom facet of a prismatic polytope causes its side facets to deform. The vertices of such side facets thus do not lie on a hyperplane (i.e. its vertices become non-collinear, non-coplanar, etc.). If this is a problem, it is then necessary to apply a simple postprocessing step that subdivides the side facets into simplices (i.e. triangles in a 3D model, tetrahedra in a 4D model). Since the side facets of model are combinatorially equivalent to $(n-1)$-cubes, they can be decomposed into simplices with ease. For instance, a square is split into two triangles by an edge that joins two of its opposite vertices, and a cube can be split into 5 tetrahedra. Similar optimal simplicial decompositions are known up to 7D [Hughes and Anderson, 1996], while non-optimal solutions are known for any dimension [Orden and Santos, 2003, Haiman, 1991].

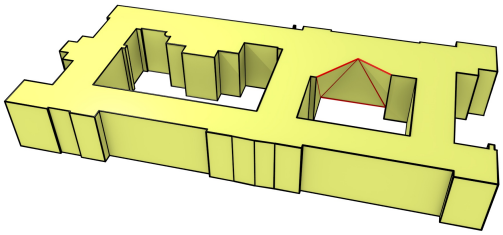
4.2 Collapsing cells

Collapsing cells is perhaps the most common simplification operation used in both geometric processing and in cartographic generalisation [Weibel, 1997]. For instance, collapsing edges and faces in a triangulation are the two most important fundamental operations used in the simplification of a GIS TIN, or more generally any triangular or polygonal mesh [Hoppe, 1996]. Similarly, roads are collapsed from areas to lines and cities are collapsed from areas to points at appropriate scales according to various cartographic rules [SGK, 1975]. For instance, Goodchild [2001] suggests a minimal feature size in paper maps of 0.5 mm—arguably, objects smaller than this value should be either increased in size or collapsed to points and represented instead using appropriate symbols.

Within the methodology described in this paper, implementing the collapse of a cell of any dimension is quite simple and can be performed in three steps: (i) all the vertices of the cell are moved to a single location (e.g. the centroid of the cell or a point known to lie in its interior), (ii) the degenerate cells produced by this process are removed, and (iii) any non linear geometries that were generated are subdivided. Figure 6 shows an example with the collapse operation applied to the top face and an edge of the top face of a polyhedron. Figure 7 shows a 4D example where the collapse operation is applied to a volume.



(a) Face collapse



(b) Edge collapse

Figure 6: Cells can be collapsed by moving all of their vertices to the same location. Shown here are the collapse of (a) a face and (b) an edge on the top facet of the prism. The modified edges in (b) are highlighted in red. Note how among the modified edges, one edge that is not incident to the collapse vertex (plus another one hidden from this view) is added in order to triangulate a non-planar facet.

As in the rotation example from Section 4.1, it is also important to note that if a lower-dimensional cell that is collapsed is a face of

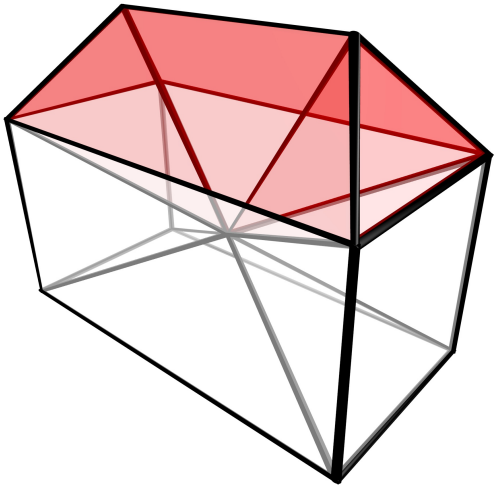
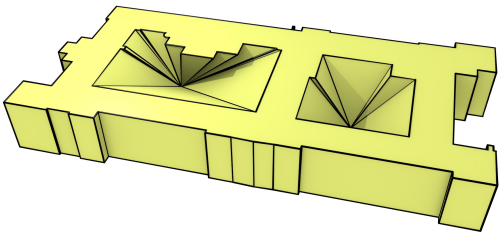


Figure 7: Collapsing a facet consisting of the volume of a simple 3D model of a house in 4D. As in Figure 2, the fourth dimension corresponds to an inwards-outwards axis due to the stereographic projection that is used.

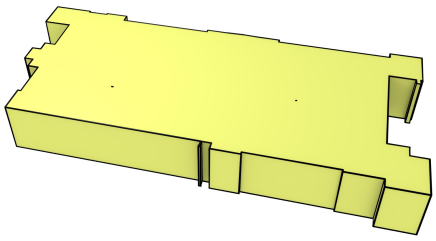
some higher-dimensional cells, the higher-dimensional cells might deform so that their vertices will not lie on a hyperplane. Whether this happens depends on the location of the newly collapsed vertex. As before, if this is a significant problem, such higher-dimensional cells can be split into simplices.

Collapse operations can be used in other ways as well. As shown in the edge collapse example in Figure 6b, it is possible to define operations that are applied only to certain vertices of a facet. However, these vertices do not have to correspond to the vertices of any given cell. For instance, they can be applied to the vertices surrounding a hole in the top or bottom facet. This makes it possible to define operations that correspond to the removal of a hole of any dimension, such as in the example in Figure 8.

Once a collapse operation has been performed, it is then often desirable to remove the degenerate cells from the model, i.e. those that are infinitesimally small but still exist in the model. Such cells are those that have been explicitly collapsed, as well as



(a) Viewed from the top



(b) Viewed from the bottom

Figure 8: Collapsing the vertices surrounding both of the holes on the bottom facet of a prism. Note how in (b) the view from the bottom, the holes are collapsed into single vertices. If a subsequent extrusion was applied to this face, they could then be ignored and reflect a smoothly changing geometry.

the lower-dimensional ones that lie on their boundaries. However, even if this information is lost in the process, detecting the degenerating cells created by this method is straightforward: such cells are those that have all their vertices at the same location in $\mathbb{R}^n$. An algorithm can thus iterate over all the vertices of all the cells of the model and easily find those that are degenerate.

When a topological data structure is not used, the degenerate cells can be deleted directly. However, when a topological data structure is used, it is better to remove these cells using algorithms that operate on a combinatorial level and recomputes only the topological relationships that have changed. The exact form of such operations depends on the specific data structure that is used. For instance, n-dimensional combi-

natorial and generalised maps already have a defined dimension-independent *removal* operator [Damiand and Lienhardt, 2003, 2014] that corresponds exactly to this requirement.

If such an operation has not been defined, an alternative approach would be to remove all the combinatorial primitives belonging to the deleted cells (and are not used elsewhere), and then proceed to recompute reconstruct an object from its boundary, e.g. using the incremental construction method presented in Arroyo Ohori et al. [2014].

5 Conclusions

Defining operations that are both dimension-independent and intuitive to use can be difficult, as we do not have the same intuitive understanding of the manipulation of higher-dimensional objects that we have in 2D and 3D. Our proposed solution to this problem is to define some of such operations on the basis of *prismatic polytopes*, as these nD objects are general enough to represent many phenomena commonly modelled in GIS but still have a simple geometry that is analogous to familiar shapes—2D rectangles and 3D prisms.

Prismatic polytopes can be easily generated using dimension-independent extrusion [Arroyo Ohori et al., 2015c], and much as in 2D, extrusion is particularly appealing because it has a simple definition and a relatively easy implementation in arbitrary dimensions. Starting from a set of non-overlapping $(n-1)$-dimensional objects, extrusion guarantees that its output consists of a set of valid non-intersecting n-dimensional objects.

Our proposed methodology to define operations on the basis of prismatic polytopes is straightforward: (i) an $(n-1)$-dimensional object or set of objects is extruded into a set of n-dimensional prismatic polytopes, (ii) a modification operation is applied to the vertices of the top or bottom facets of each prismatic polytope, and (iii) simple postprocessing is applied to fix errors introduced in the model. Within this paper, we have shown the application of this methodology to define two sets of dimension-independent operations: the most common transformations applied to GIS objects (i.e. translation, scaling and rotation) and collapsing cells. The former can be used to provide object manipulation in an interactive environment or to model objects that move and change shape, while the latter can be used as a basis for nD generalisation, or when used together with time to model smooth animations between various timestamps.

While we have only shown in this paper three concrete examples of nD transformations (translation, rotation and scale), it is good to point out that the same scheme readily extends to other affine transformations, such as shears, reflections and homothetic transformations.

Within this paper, we limit the description of the operations to the top and bottom facets of a prismatic polytope, as this type of operations has a clear and intuitive meaning. However, the general methodology can be applied to any given facet of a prismatic polytope, as well as to any bounding cell of a general polytope as long as more strict constraints are kept, e.g. preserving certain relationships to the higher-dimensional cells bounded by it and avoiding intersections caused by misplacing a collapsed point.

An interesting future possibility is to consider how to intuitively define various operations as types of *reverse collapses*. In addition to collapsing a set of vertices to a single point, it is also possible to *expand* a single vertex into an arbitrary cell of any dimension higher than zero. In many instances, an expansion of a vertex into an i-cell is trivial to build, as it geometrically results in an $(i+1)$-dimensional pyramid with the i-cell as its base and the vertex as its apex, as can be seen by considering an operation that undoes the collapses in Figures 6a, 7 and 8. This is always true when the resulting pyramid lies completely inside or completely outside the model (i.e. the cell complex).

However, if the vertex and the expansion i-cell lie on the boundary of the model[7], the operation is much more complex to implement and in some cases can be ambiguous—thus requiring more information from the user is needed in order to form the desired combinatorial structure. This can be seen by trying to define an operation that reverses the collapse in Figure 6b. Such an operation would expand the collapse vertex into an edge, but in order to do so, it would need to know which of the three triangles should be transformed into a quadrangle.

In the future, we plan to devise a more complete set of operations that provide all basic functions required in 3D+time and 3D+scale modelling while remaining intuitive, as well as to implement proofs of concept based on CGAL combinatorial maps.

Acknowledgements

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).

References

Marc P. Armstrong. Temporality in spatial databases. In *GIS/LIS '88 : proceedings : accessing the world : third annual International Conference, Exhibits, and Workshops*, pages 880–889. American Society for Photogrammetry and Remote Sensing, 1988.

Ken Arroyo Ohori. *Higher-dimensional modelling of geographic information*. PhD thesis, Delft University of Technology, apr 2016.

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

Ken Arroyo Ohori, Hugo Ledoux, Filip Biljecki, and Jantien Stoter. Modelling a 3D city model and its levels of detail as a true 4D model. *ISPRS International Journal of Geo-Information*, 4(3):1055–1075, September 2015a.

Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. An evaluation and classification of nD topological data structures for the representation of objects in a higher-dimensional GIS. *International Journal of Geographical Information Science*, 29(5):825–849, May 2015b.

Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. A dimension-independent extrusion algorithm using generalised maps. *International Journal of Geographical Information Science*, 29(7):1166–1186, July 2015c.

H. Bieri and W. Nef. Elementary set operations with d-dimensional polyhedra. In Hartmut Noltemeier, editor, *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. *Discrete & Computational Geometry*, 9: 387–426, 1993.

Lidija Čomić and Leila de Floriani. *Modeling and Manipulating Cell Complexes in Two, Three and Higher Dimensions*, volume 2 of *Lecture Notes in Computational Vision and Biomechanics*, chapter 4, pages 109–144. Springer, 2012.

Guillaume Damiand and Pascal Lienhardt. Removal and contraction for n-dimensional generalized maps. In *Proceedings of the 11th Discrete Geometry for Computer Imagery*, volume 2886, pages 408–419, 2003.

[7] This happens when the vertex and i-cell are collinear for $i = 1$, coplanar for $i = 2$ and more generally when the vertices of the i-cell and the other vertex are not linearly independent.

Guillaume Damiand and Pascal Lienhardt. *Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing*. CRC Press, 2014.

Vincenzo Ferrucci. Generalised extrusion of polyhedra. In *2nd ACM Solid Modelling '93*, pages 35–42. ACM, 1993.

A.T. Fomenko. *Variational Problems in Topology: The Geometry of Length, Area and Volume*. CRC Press, 1990.

Anders Friis-Christensen and Christian S. Jensen. Object-relational management of multiply represented geographic entities. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*, pages 150–159. IEEE Computer Society, 2003.

Michael F. Goodchild. Metrics of scale in remote sensing and GIS. *International Journal of Applied Earth Observation and Geoinformation*, 3(2):114–120, 2001.

Leonidas J. Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

Mark Haiman. A simple and relatively efficient triangulation of the n-cube. *Discrete & Computational Geometry*, 6:287–289, 1991.

Andrew J. Hanson. Geometry for *n*-dimensional graphics. In Paul S. Heckbert, editor, *Graphics Gems IV*, chapter II.6, pages 149–170. Academic Press Professional, 1994.

Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

Steven Richard Hollasch. Four-space visualization of 4D objects. Master's thesis, Arizona State University, 1991.

Hughes Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH 1996*, pages 99–108, 1996.

Robert B. Hughes and Michael R. Anderson. Simplexity of the cube. *Discrete Mathematics*, 158:99–150, 1996.

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.

John I. Marden. *Analyzing and Modeling Rank Data*, volume 64 of *CRC Monographs on Statistics & Applied Probability*. Chapman & Hall, August 1996.

Hiroshi Masuda. Topological operators and Boolean operations for complex-based non-manifold geometric models. *Computer-Aided Design*, 25(2), 1993.

D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7(2):217–236, 1978.

David Orden and Francisco Santos. Asymptotically efficient triangulations of the d-cube. *Discrete & Computational Geometry*, 30:509–528, 2003.

Donna J. Peuquet. It's about time: A conceptual framework for the representation of temporal dynamics in geographic information systems. *Annals of the Association of American Geographers*, 84(3):441–461, 1994.

J. Rossignac and M. O'Connor. SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries. In M. Wosny, J. Turner, and K. Preiss, editors, *Proceedings of the IFIP Workshop on CAD/CAM*, pages 145–180, 1989.

SGK. *Kartographische Generalisierung*. Der Schweizerischen Gesellschaft für Kartographie, 1975.

Cathy Sohanpanah. Extension of a boundary representation technique for the description of n dimensional polytopes. *Computers & Graphics*, 13(1):17–23, 1989.

J.E. Stoter, M. Post, V. van Altena, R. Nijhuis, and B. Bruns. Fully automated generalisation of a 1:50k map from 1:10k data. *Cartography and Geographic Information Science*, 41(1):1–13, January 2014.

L. van Elfrinkhof. Eene eigenschap van de orthogonale substitutie van de vierde orde. In *Handelingen van het 6e Nederlandsch Natuurkundig en Geneeskundig Congres*, pages 237–240, Delft, 1897.

Peter van Oosterom and Jantien Stoter. 5D data modelling: Full integration of 2D/3D space, time and scale dimensions. In Sara Irina Fabrikant, Tumasch Reichenbacher, Marc van Kreveld, and Christoph Schlieder, editors, *Geographic Information Science: 6th International Conference, GIScience 2010, Zurich, Switzerland, September 14-17, 2010. Proceedings*, pages 311–324. Springer Berlin Heidelberg, 2010.

Robert Weibel. Generalization of spatial data: Principles and selected algorithms. In Marc van Kreveld, Jürg Nievergelt, Thomas Roos, and Peter Widmayer, editors, *Algorithmic Foundations of Geographic Information Systems*, volume 1340 of *Lecture Notes in Computer Science*, pages 99–152. Springer Berlin Heidelberg, 1997.

M.F. Worboys. A model for spatio-temporal information. In *Proceedings of the 5th International Symposium on Spatial Data Handling*, pages 602–611, 1992.