STORING A 3D CITY MODEL, ITS LEVELS OF DETAIL AND THE CORRESPONDENCES BETWEEN OBJECTS AS A 4D COMBINATORIAL MAP

K. Arroyo Ohori, H. Ledoux, J. Stoter

3D Geoinformation, Delft University of Technology, Delft, the Netherlands

KEY WORDS: level of detail (LOD), 4D, multi-dimensional GIS, combinatorial maps

ABSTRACT:

3D city models of the same region at multiple LODs are encumbered by the lack of links between corresponding objects across LODs. In practice, this causes inconsistency during updates and maintenance problems. A radical solution to this problem is to model the LOD of a model as a dimension in the geometric sense, such that a set of connected polyhedra at a series of LODs is modelled as a single polychoron—the 4D analogue of a polyhedron. This approach is generally used only conceptually and then discarded at the implementation stage, losing many of its potential advantages in the process. This paper therefore shows that this approach can be instead directly realised using 4D combinatorial maps, making it possible to store all topological relationships between objects.

1. INTRODUCTION

3D city models of the same region are often created at multiple levels of detail (LODs). This allows a user to choose the most appropriate LOD for a given application, balancing the better results that are obtainable using more detailed models with the higher computational requirements that are necessary to obtain them (Biljecki et al., 2014).

However, the creation of these models is a complex task that needs to be performed continuously, as 3D city models need to be kept up to date (Zlatanova and Holweg, 2004; Kolbe et al., 2005). Given the large size and complexity of current 3D city models, it can be very beneficial to have incremental updates to the model which affect only a building and its immediate surrounding area (Döllner et al., 2006). These can take place as buildings and other city objects are built, modified and destroyed.

In order to apply such incremental update processes to 3D city models at multiple LODs, links between related objects are crucial. Given an object at a certain LOD, links usually point to its incident and adjacent objects at the same LOD (i.e. the topological relationships that are most common in GIS), as well as to its *corresponding* objects at other LODs, even when these objects are of different dimension (e.g. when a thin polyhedron in a higher LOD is collapsed to a polygon in a lower LOD). These links can then be used to propagate changes to other LODs (van Oosterom and Stoter, 2010) or to apply consistency checks to new or newly altered objects (Gröger and Plümer, 2011), among other operations that are part of a robust update process.

However, in most of the data models used in GIS, these latter links across LODs are either non-existent or limited to the use of common IDs at the 2D or 3D object level. These simple schemes are sufficient in the cases where the missing links can be deduced geometrically, such as when there are identical geometries across LODs, but as shown in Figure 1, it is possible that there are no common geometries across LODs. Using only common IDs also means that it is difficult to store complex correspondence relationships, such as an aggregation of multiple objects into one, or those connecting the points, line segments and polygons on the boundary of corresponding 2D or 3D objects.

Partly in response to these shortcomings, various authors have proposed to model the LODs of a 3D model as another fully independent dimension in the geometric sense (van Oosterom and Stoter, 2010; Paul et al., 2011; Stoter et al., 2012), resulting in a set of 0D–4D objects that can be modelled mathematically as a 4D cell complex embedded in 4D space. This makes it possible to store *all* correspondences between the objects across all LODs, even in arbitrarily complex situations, such as continuous LODs[1] (Döllner and Buchholz, 2005; van Oosterom and Meijers, 2014) or objects that move and change shape. A 3D building that is normally modelled as different polyhedra across a series of LODs is then modelled as a single *polychoron*, the 4D analogue of a polyhedron, which is embedded in 4D space.

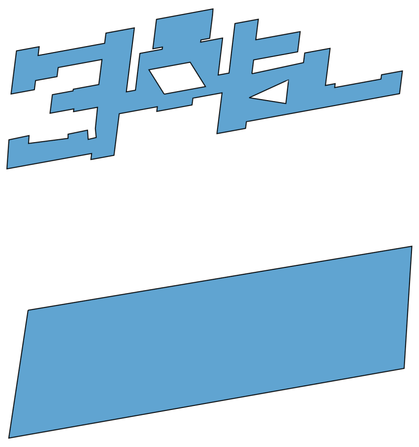
Figure 1: Two LODs of a building footprint. Note that there are no vertices, edges or faces with the same geometry in both LODs, and that many primitives in the higher LOD are equivalent to a single one in the lower LOD.

However, this integrated approach, which is presented in Section 2, is normally only used conceptually and is then discarded at the implementation stage. Most '4D GIS' therefore store the model as a series of minimally linked 2D/3D representations, just as is done when a non-integrated approach is used (Raper, 2000; van Oosterom and Meijers, 2014). Many of the advantages of the integrated approach are thus unfortunately lost in practice.

As an alternative, we argue that it is possible to preserve all cor-

[1] As opposed to a set number of discrete LODs.

respondences between the objects of every dimension directly as a 4D cell complex (Arroyo Ohori et al., 2015c). In this paper we show how this can be done in practice using nD combinatorial maps (Damiand and Lienhardt, 2014). This dimension-independent data structure, introduced briefly in Section 3, is likely the best option for an integrated 4D GIS model at the present time due to its compactness and the availability of efficient libraries implementing them (Arroyo Ohori et al., 2015b), such as CGAL[2] and CGoGN[3] (Kraemer et al., 2014).

Based on CGAL Combinatorial Maps, we show in Section 4 how a 4D GIS using real multidimensional 0D–4D objects can be implemented in practice, describing the main aspects of such a system, including how real-world 4D objects can be created and manipulated. We finish with a discussion of the current and future possibilities of a 4D GIS in practice in Section 5.

2. MODELLING THE LODS OF A 3D MODEL AS A 4D CELL COMPLEX

In the general sense, it is possible to model any number of *parametrisable* characteristics as dimensions in the geometric sense. When the standard (two or three) spatial dimensions are combined with other non-spatial ones modelled in this manner, real-world 0D–3D entities are then modelled as higher-dimensional objects embedded in higher-dimensional space. These can then be directly stored using *higher-dimensional data structures*, such as nD combinatorial maps. Although the approach can be applied with any type of characteristics, it is usually used with characteristics that are closely linked to space, such as time (Raper, 2000) and scale (van Oosterom and Stoter, 2010).

This *higher-dimensional spatial modelling approach* is well grounded in long-standing mathematical theories and offers interesting possibilities in practice. Descartes (1637) already laid the foundation for nD geometry by putting coordinates to space, allowing the numerical description of geometric primitives and the use of algebraic methods on them, theories of nD geometry were developed by Riemann (1868) among others, and Poincaré (1895) developed algebraic topology with a dimension-independent formulation, stating that even if nD objects could not be [then] represented, they do have a precise topological definition, and consequently properties that can be studied. From an application point of view, 4D topological relationships between 4D objects provide insights that 3D topological relationships cannot (Arroyo Ohori et al., 2013), weather and groundwater phenomena cannot be adequately studied in less than four dimensions (McKenzie et al., 2001), and van Oosterom and Stoter (2010) argue that the integration of space, time and scale into a 5D model for GIS can be used to ease data maintenance and improve consistency, as algorithms could detect if the 5D representation of an object is consistent and does not conflict with other objects.

Within this paper, we focus solely on modelling the LOD of a 3D city model as an extra geometric dimension—in so far as it can be used to store all topological relationships between related objects across LODs. A set of connected 2D polygons at multiple LODs are then stored as a single 3D polyhedron[4], as is shown in Figure 2, and a set of connected 3D polyhedra at multiple LODs as a single 4D *polychoron*. Notably, the correspondences between equivalent objects across LODs are modelled as geometric primitives, making it possible to perform geometric operations with

[2]`http://doc.cgal.org/latest/Combinatorial_map`
[3]`http://cgogn.unistra.fr`
[4]Separate polygons might become connected in different situations, such as by being joined into a single one at one or more LODs.

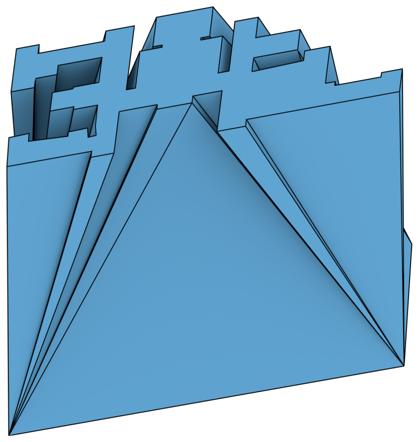
Figure 2: Two LODs of a building footprint are stored as a single polyhedron. Note that the correspondences between vertices, edges and faces between the LODs are clearly indicated by the vertical edges and faces.

them (e.g. extracting an intermediate LOD for visualisation purposes) or to attach attributes to them, just as is done to other geometric primitives.

In order to understand how the polychora in a 4D (3D+LOD) setting look like, it is easier to first consider a 3D setting that consists of two spatial dimensions and the LOD of the model as a third dimension. In this setting, a polygon that is modelled at precisely one LOD (i.e. a point on the LOD axis) is still a polygon, but one that is embedded in 3D space. The simplest volumetric example occurs when a polygon is modelled identically along a range of LODs, as is shown in Figure 3. In this case, the resulting

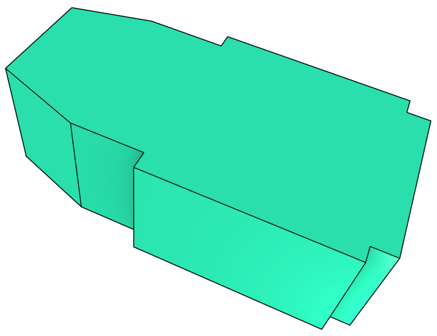
Figure 3: An unchanged building footprint at two LODs forms a prism-shaped polyhedron.

polyhedron is a prism that consists of base and top faces with the same geometry as the original polygon, which are orthogonal to the LOD axis and represent the end points of the range. These faces are joined by lateral quadrilateral faces, which are aligned with the LOD axis and connect corresponding edges of the top and bottom face.

When polygons are modelled differently at different LODs, the resulting polyhedra can be arbitrarily more complex. However, it is worth noting that applying many fundamental operations to a polygon in the same 2D+LOD setting also result in similarly easily-definable volumes. This is the case for all basic transformation operations or collapses of objects of any dimension, as is shown in Figure 4 and Figure 5.

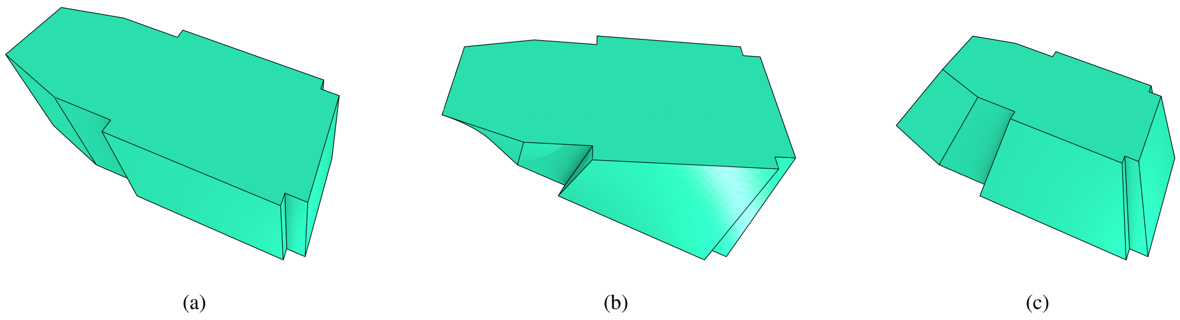

Figure 4: Applying various transformations to a building footprint along the LOD axis: (a) translation, (b) rotation and (c) scale.

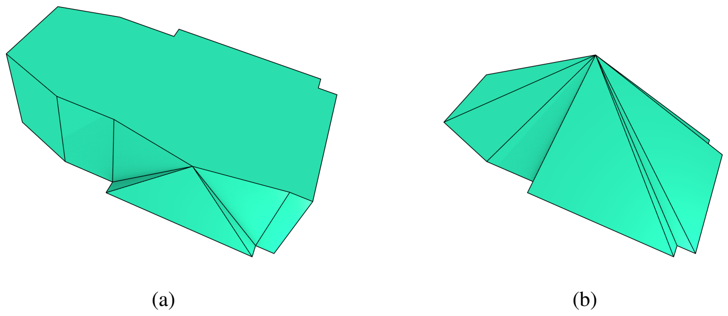

Figure 5: Collapsing (parts of) a building footprint along the LOD axis: (a) an edge, (b) a face.

Moreover, all these volumetric cases (unchanged, transformed and collapsed polygons) can be generated algorithmically in a simple manner by using extrusion as a first step. A polygon that is modelled identically along an LOD range can be created by simply extruding the polygon along the range—a common operation in geometric modelling for which there are various available algorithms (Ledoux and Meijers, 2011). As a second step, the transformations simply require applying the transformation to the extruded vertices[5], while the collapses require moving the unextruded vertices of the collapsed cell to a certain location (e.g. the centroid of the edge or face). Degenerate edges and faces can then be easily detected and removed, as all their vertices are in the same location.

These cases are illustrative because they work in the same manner in the 4D setting[6], being easy to define and to generate algorithmically. A *prismatic polychoron*—the 4D analogue of a prism—can be constructed by extruding a polyhedron along a range, which can be done using the algorithm described in Arroyo Ohori et al. (2015a). Figure 6 shows the result of extruding the polyhedron in Figure 3, which is equivalent to a polyhedron remaining unchanged along the LOD axis.

In a similar manner as the 3D cases, a prismatic polychoron can also be easily modified to reflect a transformation or a collapse that occurs along the LOD axis. Transforming a polyhedron means applying the transformation to the unextruded vertices of the polychoron. Collapsing an edge, face or volume means moving all of its vertices to the same location. Degenerate edges, faces or volumes can be identified by checking whether all their vertices are in the same location, and can therefore be easily removed.

[5]If the bottom face of the extruded polygon represents it before the transformation, this would mean transforming the vertices in its top face.

[6]In fact, they work in the same manner in any dimension.

3. ND COMBINATORIAL MAPS

Combinatorial maps is a data structure originally proposed by Edmonds (1960) to describe the 2D surfaces of 3D objects. Their extension to arbitrary dimensions is described by Lienhardt (1994) for objects without boundaries (e.g. $\mathbb{R}^n$ or the 'wrap-around' surfaces around objects) and extended to objects with boundaries by Poudret et al. (2007). They are able to describe subdivisions of orientable quasi-manifolds—a specific combinatorial interpretation of the topological concept of a manifold. However, it is worth noting that non-manifold objects can still be stored in a combinatorial map by the use of non-manifold domains (Arroyo Ohori et al., 2015b).

In order to give a more precise description of how we model 4D objects, it is useful to start from the concept of a cell complex. Intuitively, a cell complex is a structure made of connected *cells*, where an i-dimensional cell (i-cell) is a topological object homeomorphic to an i-ball (i.e. point, arc, disk, ball, etc.)[7]. Vertices are thus 0-cells, edges are 1-cells, faces are 2-cells, volumes are 3-cells, and so on. An i-cell can be used to model an i-dimensional object, so considering only linear geometries, 1-cells are representations of line segments, 2-cells of polygons, 3-cells of polyhedra, and 4-cells of polychora. A j-dimensional face (j-face) of an i-cell is a j-cell, $j \leq i$, that lies on the boundary of the i-cell. Two i-cells are said to be adjacent if they have a common $(i-1)$-face, and an i-cell and a j-cell, $i \neq j$, are said to be incident if either is a face of the other.

Combinatorial maps are thus data structures that can be used to represent cell complexes of any dimension and are composed of combinatorial primitives called *darts*, which are equivalent to the *simplices* in a *simplicial decomposition*[8] of the input cell complex. If we consider all the cells of the complex of dimension

[7]See Hatcher (2002) for a more rigorous definition.

[8]i.e. the vertices, edges, triangles, tetrahedra, etc. in an n-dimensional combinatorial triangulation

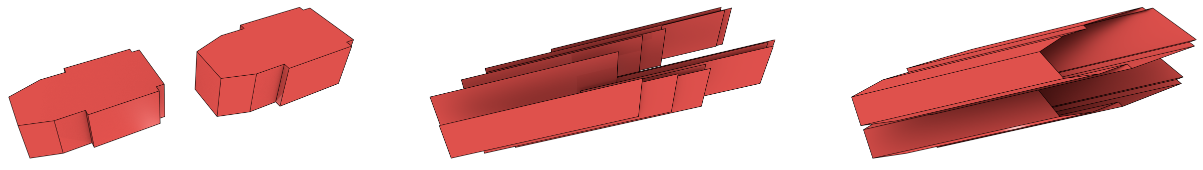

Figure 6: The polyhedron in Figure 3 can be extruded to 4D using the algorithm in Arroyo Ohori et al. (2015a). The result is a single polychoron, whose faces are shown here in parts for clarity: (a) the faces in the two end volumes, (b) the lateral faces connecting corresponding vertical edges, and (c) the top and bottom faces connecting corresponding horizontal edges. Not shown here are the 16 polyhedra that are formed from these faces.

two or higher as symbolic vertices, the simplicial decomposition is computed by creating simplices formed by joining combinations of cells of every dimension higher than zero, all of which are incident to each other. Since every dart thus joins two of the original vertices—in addition to vertices representing cells of every dimension from two upwards—, an orientation is given to a dart by specifying an order among the two original vertices.

As shown in Figure 7, darts in a 2D combinatorial map are thus equivalent to *combinatorial triangles* defined by an incident edge-face pair, which are then given an orientation. In 2D, darts are also equivalent to the oriented half-edges in a typical half-edge data structure, e.g. the DCEL (de Berg et al., 2008).

Higher-dimensional combinatorial maps are easiest to picture from the point of view of the simplicial decompositions that are similar to the 2D one in Figure 7a. As shown in Figure 8, in a 3D combinatorial map, darts are equivalent to tetrahedra defined by an incident edge-face-volume triplet. Although 4D objects are hard to picture, by analogy it is easy to see that a dart in a 4D combinatorial map is a 4-simplex, which is defined by an incident edge-face-volume-4-cell 4-tuple.

The darts in a combinatorial map are connected by ordered relations between them, which in an n-dimensional combinatorial map are denoted by β_1 to β_n. These relations represent the adjacency relations between the simplices in the simplicial complex, such that the β_i relation of a dart d connects it to the adjacent simplex that represents all the same cells except for the i-dimensional one—equivalent to switching in a simplex the vertex representing the i-cell for the i-cell of a neighbouring simplex. Since 1-cells are only implicitly represented through two edge-connected vertices, β_1 connects a dart to the *next* dart within the face according to the predefined order between the vertices. The other relations of a dart (i.e. β_2 and higher), which by definition share both of the original vertices, β_i always connects a dart to an *oppositely-oriented dart*. For example, β_2 connects a dart that represents the same vertices and edge, but that represents the adjacent face (which in the DCEL is commonly known as the *twin*).

4. STORING A 3D+LOD MODEL AS A 4D COMBINATORIAL MAP

As a dart in an n-dimensional combinatorial map is connected to n other darts using the ordered relations ranging from β_1 to β_n, it is possible to navigate through all these links by storing them as a n-tuple per dart. However, it is very inefficient to rotate through all the darts of a face in order to get the *previous* dart of the face. Because of this, storing the inverse of β_1, i.e. β_1^{-1}, is desirable as well. In this manner, both β_1 and β_1^{-1} can be used to cycle through a face in the clockwise and counterclockwise directions. In our particular case of a 4D combinatorial map, given a dart d, we therefore store a tuple of relations:

$$\left(\beta_1^{-1}(d), \beta_1(d), \beta_2(d), \beta_3(d), \beta_4(d)\right).$$

A dart in an n-dimensional combinatorial map is a representation of two 0-cells and one cell of every dimension higher than zero. Because of this, a dart also can be used to store the attributes of all of its cells in the form of another ordered tuple, or to keep links to external structures storing them if they are better stored separately—which depends on the space needed to store said attributes compared to the space needed to store the links. In order to reduce the amount of storage that is required, it is possible to omit the attributes for one of the two 0-cells of the dart in an ordered manner, omitting either always the first or always the second 0-cell according to the predefined orientation of the combinatorial map. These can be obtained from any of their β-linked neighbours as the first 0-cell of a dart is always the second 0-cell of a β-linked neighbour due to the consistent orientation that is set in a combinatorial map. Considering for a given dart d, $a_i(d)$ links it to the attribute(s) of its i-cell, we can therefore similarly store a tuple of attributes:

$$(a_0(d), a_1(d), a_2(d), a_3(d), a_4(d)).$$

Among all the attributes of the cells of all dimensions, those of the 0-cells are particularly important. By embedding every vertex at a location in space defined by a tuple of coordinates, it is possible to embed an abstract cell complex in space. In the case of a 4D cell complex, every vertex should have a location defined by a point in 4D space, which is represented by tuple of coordinates (x, y, z, l), where x, y and z are the coordinates of the point in 3D space, and l is a point on the LOD axis. However, it is important that the vertices of the cell complex are embedded in a geometrically correct manner. The faces of the complex should be coplanar and the volumes of the complex should lie on a flat region of 3D space. That is, the points where their vertices are embedded should lie on the subsets of space defined by a linear combination of respectively two and three vectors.

In CGAL Combinatorial Maps, darts are already implemented as individual primitives that store their relationships to other darts, while the embeddings of the vertices as points in 4D space can be handled through the Linear Cell Complex package. In order to store other relevant attributes for the darts, as well as for cells of every dimension, it is possible to do so by defining custom `Dart` and `Linear_cell_complex` classes. Figure 9 shows a fully dimension-independent example using simple integer IDs.

Based on such a custom `Linear_cell_complex` class, a 3D city model can be loaded into a 3D cell complex incrementally (Arroyo Ohori et al., 2014). In our case, we use the OGR Simple

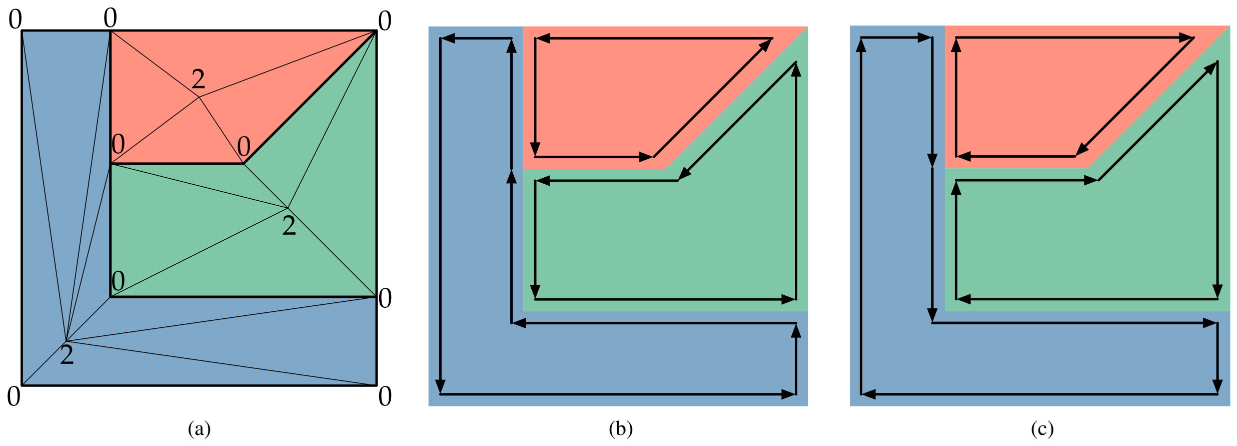

Figure 7: A 2D combinatorial map representing three polygons. (a) The underlying simplicial complex, where every triangle consists of two vertices (marked as 0) at either end of an edge and a symbolic vertex for every face (marked as 2). (b) The combinatorial map that is generated by choosing a counterclockwise orientation for the polygons. (c) Same for the clockwise orientation. Darts are represented by triangles in (a) and arrows in (b) and (c).

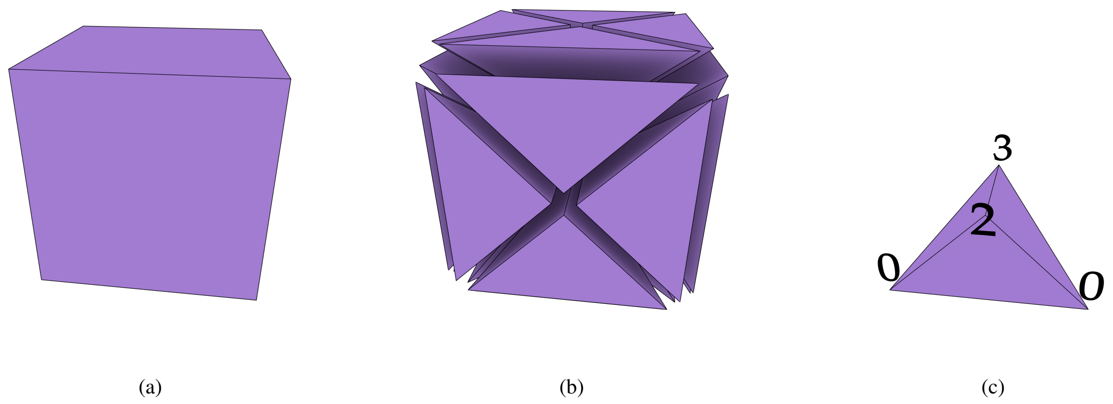

Figure 8: A 3D combinatorial map representation of a (a) cube consists of (b) 24 darts. (c) Each of these darts is defined by an incident edge-face-volume triplet. In this case, they are the lower front edge (between the 0s), front face (2) and the only volume of the cube (3). Note that there are also two possible orientations for such a map, which are not shown here.

```
template <int d, class Refs>
struct Dart_with_id : public CGAL::Dart<d, Refs> {
public:
  typedef CGAL::Dart<d, Refs> Dart;
  typedef typename Refs::size_type size_type;
  static const size_type NB_MARKS = Refs::NB_MARKS;
  int id;

  Dart_with_id() : Dart() {
    id = -1;
  }

  Dart_with_id(int id) : Dart() {
    this->id = id;
  }

  Dart_with_id(const Dart& adart) : Dart(adart) {
    id = -1;
  }
};

template <unsigned int d>
struct Linear_cell_complex_items_with_id {
  template <class LCC>
  struct Dart_wrapper {
    typedef CGAL::Cell_attribute_with_point<LCC, int> Point_attribute_with_id;
    typedef CGAL::Cell_attribute<LCC, int> Attribute_with_id;

    template <unsigned int attributes_to_add, class Result = CGAL::cpp11::tuple<> >
    struct Linear_cell_complex_items_with_id_attributes;

    template <class ... Result>
    struct Linear_cell_complex_items_with_id_attributes<0, CGAL::cpp11::tuple<Result ...> > {
      typedef CGAL::cpp11::tuple<Point_attribute_with_id, Result ...> tuple;
    };

    template <unsigned int attributes_to_add, class ... Result>
    struct Linear_cell_complex_items_with_id_attributes<attributes_to_add, CGAL::cpp11::tuple<Result ...> > {
      typedef typename Linear_cell_complex_items_with_id_attributes<attributes_to_add-1,
        CGAL::cpp11::tuple<Attribute_with_id, Result ...> >::tuple tuple;
    };

    typedef Dart_with_id<d, LCC> Dart;
    typedef typename Linear_cell_complex_items_with_id_attributes<d>::tuple Attributes;
  };
};

template <unsigned int d>
struct Linear_cell_complex_with_ids {
public:
  typedef CGAL::Linear_cell_complex<d, d, CGAL::Linear_cell_complex_traits<d>,
    Linear_cell_complex_items_with_id<d> > type;
};
```

Figure 9: Custom Dart_with_id and Linear_cell_complex_with_id classes. Dart_with_id stores an integer id per dart, while Linear_cell_complex_with_id stores an integer ID for every cell of every dimension. The latter is templated with the dimension d of the cell complex, whereafter it is used to add integer attributes for every cell of dimension higher than zero. This example uses variadic templates to show how it is possible to do this in a fully dimension-independent manner.

Feature Library[9] to read standard GIS data formats. Based on the Simple Features specification (OGC, 2011), we read a dataset face by face, creating a new dart per vertex of the face, embedding it into its 3D coordinates using CGAL's `Point_d` nD data type and assigning it a sequential ID. Every dart within the face is then linked to and from the previously created one respectively by their β_1^{-1} and β_1 relationships, then the last dart of the face is connected to the first one in the same manner. The face is then assigned an ID based on its feature ID obtained through OGR.

The separate faces representing a single volume are then linked together using the incremental construction method described in Arroyo Ohori et al. (2014). Once the faces have been linked, it is possible to assign sequential IDs to every edge (as doing so earlier creates gaps in the numbering due to disconnected edges). Finally, it is possible to assign an ID to the volume.

By associating every volume in the 3D cell complex with a *scale range* along which it is a valid representation, the entire 3D cell complex can be extruded to 4D. Note that the attributes of the cells are preserved—an i-cell is always extruded into two i-cells and an $(i+1)$-cell that lies between them, all of which can inherit the attributes of the unextruded cell. The 4-cells at this stage represent the prismatic polychora discussed in Section 2 and can be used for further operations.

5. CONCLUSIONS

Modelling non-spatial characteristics as additional dimensions in the geometric sense is a powerful technique. Applied to the level of detail of a 3D city model, it enables the storage of arbitrarily complex relationships between objects by keeping track of all possible topological relationships. As this approach is generic, the overarching methods presented in the paper is not only applicable to the LOD of a model. It can be applied to to any characteristic that is *parametrisable*, such as time (van Oosterom and Stoter, 2010).

Combinatorial maps are likely the best option for an integrated 4D GIS model at the present time due to their compactness and the availability of efficient libraries implementing them (Arroyo Ohori et al., 2015b) In terms of space, a given cell complex stored as a combinatorial map generally requires only half the combinatorial primitives compared to a generalised map (Lienhardt, 1994) or cell-tuple structure (Brisson, 1993). The libraries implementing combinatorial maps significantly decrease the effort that is needed to create and manipulate general 4D cell complexes efficiently—something that only becomes more important due to the so-called 'curse of dimensionality' (Bellman, 1957), where the number of combinatorial elements on a higher-dimensional representation can increase in size exponentially on the dimension.

References

Arroyo Ohori, K., Boguslawski, P. and Ledoux, H., 2013. Representing the dual of objects in a four-dimensional GIS. In: A. Abdul Rahman, P. Boguslawski, C. Gold and M. Said (eds), Developments in Multidimensional Spatial Data Models, Lecture Notes in Geoinformation and Cartography, Springer Berlin Heidelberg, Johor Bahru, Malaysia, pp. 17–31.

Arroyo Ohori, K., Damiand, G. and Ledoux, H., 2014. Constructing an n-dimensional cell complex from a soup of (n-1)-dimensional faces. In: P. Gupta and C. Zaroliagis (eds), Applied Algorithms. First International Conference, ICAA 2014, Kolkata, India, January 13-15, 2014. Proceedings, Lecture Notes in Computer Science, Vol. 8321, Springer International Publishing Switzerland, Kolkata, India, pp. 37–48.

Arroyo Ohori, K., Ledoux, H. and Stoter, J., 2015a. A dimension-independent extrusion algorithm using generalised maps. International Journal of Geographical Information Science.

Arroyo Ohori, K., Ledoux, H. and Stoter, J., 2015b. 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), pp. 825–849.

Arroyo Ohori, K., Ledoux, H., Biljecki, F. and Stoter, J., 2015c. Modelling a 3D city model and its levels of detail as a true 4D model. ISPRS International Journal of Geo-Information (3), pp. 1055–1075.

Bellman, R. E., 1957. Dynamic programming. Princeton University Press.

Biljecki, F., Ledoux, H., Stoter, J. and Zhao, J., 2014. Formalisation of the level of detail in 3d city modelling. Computers, Environment and Urban Systems 48, pp. 1–15.

Brisson, E., 1993. Representing geometric structures in d dimensions: topology and order. Discrete & Computational Geometry 9, pp. 387–426.

Damiand, G. and Lienhardt, P., 2014. Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing. CRC Press.

de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O., 2008. Computational Geometry: Algorithms and Applications. 3 edn, Springer-Verlag.

Descartes, R., 1637. Discours de la méthode. Jan Maire, Leyde.

Döllner, J. and Buchholz, H., 2005. Continuous level-of-detail modeling of buildings in 3d city models. In: GIS'05, ACM, pp. 173–181.

Döllner, J., Kolbe, T. H., Liecke, F., Sgouros, T. and Teichmann, K., 2006. The virtual 3d city model of berlin — managing, integrating and communicating complex urban information. In: UDMS 2006.

Edmonds, J., 1960. A combinatorial representation of polyhedral surfaces. Notices of the American Mathematical Society.

Gröger, G. and Plümer, L., 2011. How to achieve consistency for 3d city models. Geoinformatica 15, pp. 137–165.

Hatcher, A., 2002. Algebraic Topology. Cambridge University Press.

Kolbe, T. H., Gröger, G. and Plümer, L., 2005. Citygml: Interoperable access to 3d city models. In: P. van Oosterom, S. Zlatanova and E. M. Fendel (eds), Geo-information for Disaster Management, Springer Berlin Heidelberg, pp. 883–899.

Kraemer, P., Untereiner, L., Jund, T., Thery, S. and Cazier, D., 2014. CGoGN: n-dimensional meshes with combinatorial maps. In: J. Sarrate and M. Staten (eds), Proceedings of the 22nd International Meshing Roundtable, Springer International Publishing Switzerland, pp. 485–503.

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

[9]`http://www.gdal.org`

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

McKenzie, J. W., Williamson, I. P. and Hazelton, N., 2001. 4-D adaptive GIS: Justification and methodologies. Technical report, Department of Geomatics, The University of Melbourne.

OGC, 2011. OpenGIS Implementation Specification for Geographic Information - Simple Feature Access - Part 1: Common Architecture. 1.2.1 edn, Open Geospatial Consortium.

Paul, N., Bradley, P. E. and Breunig, M., 2011. Integrating space, time, version and scale using alexandrov topologies. Available at `http://arxiv.org/abs/1303.2595`.

Poincaré, M., 1895. Analysis situs. Journal de l'École polytechnique 2(1), pp. 1–123.

Poudret, M., Arnould, A., Bertrand, Y. and Lienhardt, P., 2007. Cartes combinatoires ouvertes. Technical Report 2007-01, Laboratoire SIC, UFR SFA, Université de Poitiers.

Raper, J., 2000. Multidimensional geographic information science. Taylor & Francis.

Riemann, B., 1868. Ueber die Hypothesen, welche der Geometrie zu Grunde liegen. PhD thesis, Abhandlungen der Königlichen Gesellschaft der Wissenschaften zu Göttingen.

Stoter, J., Ledoux, H., Meijers, M. and Arroyo Ohori, K., 2012. Integrating scale and space in 3D city models. In: J. Pouliot, S. Daniel, F. Hubert and A. Zamyadi (eds), Proceedings of the 7th International 3D GeoInfo Conference, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII-4/C26, ISPRS, Québec City, Canada, pp. 7–10.

van Oosterom, P. and Meijers, M., 2014. Vario-scale data structures supporting smooth zoom and progressive transfer of 2D and 3D data. International Journal of Geographical Information Science 28, pp. 455–478.

van Oosterom, P. and Stoter, J., 2010. 5D data modelling: Full integration of 2D/3D space, time and scale dimensions. In: Proceedings of the 6th International Conference GIScience 2010, Springer Berlin / Heidelberg, pp. 311–324.

Zlatanova, S. and Holweg, D., 2004. 3d geo-information in emergency response: A framework. In: Proceedings of the 4th International Symposium on Mobile Mapping Technology.