Modelling a 3D city model and its levels of detail as a true 4D model

Ken Arroyo Ohori   Hugo Ledoux   Filip Biljecki

Jantien Stoter

The various levels of detail (LODs) of a 3D city model are often stored independently—without links between the representations of the same object—, causing inconsistencies as well as update and maintenance problems. One solution to this problem is to model the LOD as an extra geometric dimension perpendicular to the three spatial ones, resulting in a true 4D model in which a single 4D object (a polychoron) represents a 3D polyhedral object (e.g. a building) at all its LODs and a multiple-LOD 3D city model is modelled as a 4D cell complex. While such an approach has been discussed before at a conceptual level, our objective in this paper is to describe how it can be realised by appropriately linking existing 3D models of the same object at different LODs. We first present our general methodology to construct such a 4D model, which consists of three steps: (1) finding corresponding 0D-3D cells; (2) creating 1D-4D cells connecting them; and (3) constructing the 4D model. Because of the complex relationships between the objects in different LODs, the creation of the connecting cells can become difficult. We therefore describe four different alternatives to do this, and we discuss the advantages and disadvantages of each in terms of their feasibility in practice and the properties that the resulting 4D model has. We show how the different linking schemes result in objects with different characteristics in several use cases. We also show how our linking method works in practice by implementing the linking of matching cells to construct a 4D model.

1 Introduction

Three-dimensional city models are characterised by their level of detail (LOD), a concept that describes their complexity and fineness, which is also related to their usability [Biljecki et al., 2014b]. An increase in the detail of a model enables more applications, but means that its representations occupy larger sizes and their processing involves higher computational costs. Because of this, it is often desirable to have different LODs for the same 3D objects [Zhao et al., 2012b; Zhu et al., 2009], similarly as in computer graphics [Luebke et al., 2003]. For instance, the international standard for storing and representing 3D city models, CityGML [OGC, 2012], offers the possibility to jointly store data at multiple LODs [Gröger and Plümer, 2012].

However, the different representations of the same set of objects are usually stored separately and are either unconnected or connected only at the object level (e.g. through the use of IDs). This means that complex relationships between objects, such as collapses, aggregations and others that are not one-to-one are difficult to store, which causes, among others, update and maintenance problems as well as inconsistencies [Biljecki et al., 2015]. It also complicates the storage of semantic information about these relationships. Topographic maps, which are also available at multiple LODs, usually suffer from the same problems [Zhang et al., 2014]. As further explained in Section 2, linking the representations on an object-to-object basis is unsatisfactory because complex relationships such as objects that are being displaced or multiple objects being merged into one are only implicitly described.

As described in Section 3, an approach to solve the multiple independent representations problem is to model the LOD as an extra geometric dimension, which is independent and perpendicular to the three spatial ones. This makes it possible to store all correspondences between objects across discrete LODs by modelling them as if they were continuous, and can be used for several applications, such as guaranteeing the

consistency of a model across the LODs or the automatic update of datasets [van Oosterom and Stoter, 2010]. In the case of a model with three spatial dimensions, this implies constructing a true 4D geometric model[1] in which the primitives are *polychora* embedded in 4D space, the 4D analogue of 2D polygons or 3D polyhedra, with every point having 4D coordinates of the form (x, y, z, l), where l defines a point along the LOD axis. A true 4D model can then be stored using a variety of nD data structures [Arroyo Ohori et al., 2015a] and a particular LOD can be obtained from the 4D model by 'slicing' it, i.e. computing its intersection with a certain surface or volume, at a predetermined point along the LOD dimension [Arroyo Ohori et al., 2013].

We present in Section 4 four linking schemes that can be used to construct 4D models from a set of 3D objects, discussing the advantages and disadvantages of each method in terms of their feasibility in practice and of the properties that the 4D model would have. In Section 5 we present several use cases to demonstrate how the different schemes result in objects with different characteristics. The best scheme is therefore application-dependent. For instance, one paradigm might be useful for visualisation, while another one might be more suitable as the input of a noise modelling analysis. In Section 6 we show how our linking method works in practice by implementing the linking of matching cells to construct a 4D model. We conclude in Section 7 with a discussion and our ideas for future work.

2 Related work

2.1 Level of detail in 3D city modelling

The general notion of the level of detail in 3D city modelling has been inves-

---

[1]This is opposed to the more common usage of the term '4D model' in the GIS world, which refers to any system that handles temporal information in any manner, most commonly as a simple timestamp that is attached to every 2D/3D object.

tigated by Biljecki et al. [2014b], who decompose the concept of LOD into multiple (mostly continuous) metrics such as the geometric granularity, dimensionality, appearance, and spatio-semantic coherence. The latter has been investigated in more details by Stadler and Kolbe [2007].

Meng and Forberg [2007] and Fan and Meng [2012] describe the scale space continuum of 3D buildings, and define LOD as a milestone along the scale space. Döllner and Buchholz [2005] also consider the LOD of a model as a conceptual dimension, allowing for the definition of continuous LODs. This is inline with our work since we integrate the scale as an extra dimension, which requires parametrising it.

Managing multiple LODs and switching between them is a topical problem in computer graphics [Clark, 1976], which has also influenced LOD management in 3D GIS [Coors and Flick, 1998]. Coarser LODs are usually generalised from finer LODs to improve visualisation performances [Guercke et al., 2011; Zhao et al., 2012a; Semmo et al., 2012; Glander and Döllner, 2009].

2.2 Identifying and linking corresponding objects

In order to join multiple separate representations, stored as independent datasets, it is first necessary to find the correspondences between (equivalent) objects at different LODs. These independent objects can then be linked using various structures that have been previously described in the scientific literature, such as: hierarchical planar subdivisions [Filho et al., 1995], multi-scale partitions [Rigaux and Scholl, 1995], nested maps [Plümer and Gröger, 1997] and topological generalised area partitioning trees [van Oosterom, 2005].

The simplest linking schemes work within the context of an *automatic generalisation* process [Weibel, 1997], in which less detailed (i.e. generalised) versions of a model can be created algorithmically. In this case the correspondences between objects at different LODs are known and can be linked

during the generalisation process [Hampe et al., 2003].

Other methods attempt to find these correspondences using *map matching* methods, which can take into account the geometry [Veltkamp and Hagedoorn, 2001], topology and semantics of and between the objects [Devogele et al., 1996]. Devogele et al. [1996] does this in three steps: (1) manually finding correspondences between semantic classes; (2) resolving conflicts; and (3) matching objects using geometry, topology and semantics. Zhang et al. [2014] matches features by computing a compatibility coefficient, derived from the similarity in their geometry and that of their neighbours. Biljecki et al. [2015] detect identical geometries across multiple LODs of the same object in a CityGML file, and reuse them to obtain a smaller file size.

2.3 LOD as a dimension

Döllner and Buchholz [2005] discusses the concept of a continuous LOD that can be used to model discrete LODs. Although this is not directly related to the LOD as a dimension, it enables the LOD to be modelled in this manner by parametrising the LOD such that it can be placed in an axis.

Van Oosterom and Meijers [2014] discuss the integration of 2D maps at different LODs in one model that is conceptually 3D (2D+LOD), the space-scale cube (SSC). The polygonal areas in a map are stored as a hierarchy of 2D planar partitions at different LODs which are linked together. However, as this approach does not use actual 3D objects, only the operations that are encoded into the logic of its algorithms can be supported (i.e. line generalisation and merging of areas).

They also discuss a conceptual generalisation of the SSC to 4D, i.e. where volumes are used as input and the level of detail becomes the fourth dimension. However, their description is only theoretical: there is no attempt to construct or store a 4D model. Also, the only case considered is two adjacent rectangular prisms being aggregated

and then simplified to a higher scale. Constructing the 4D model for such a case is the simplest of the four cases we consider in Section 4 because both input objects have the same topology, and finding a correspondence between primitives is thus straightforward.

2.4 Storing a 4D model

There are a few nD data structures that can be used in order to store a 4D model in a computer, Arroyo Ohori et al. [2015a] give a complete overview. The simplest ones involve extensions of the 2D/3D raster [Mason et al., 1994; Bernard et al., 1998] and hierarchical subdivision [Samet and Tamminen, 1985; Varma et al., 1990] representations common in GIS. These can use similar structures and algorithms as their 2D/3D counterparts and are therefore easy to implement, but are only capable of achieving rough approximations of the boundaries of the objects and grow in size exponentially as the dimension increases. Vector data structures with limited topological information, such as Simple Features [OGC, 2011] or incidence graphs [Rossignac and O'Connor, 1989; Masuda, 1993; Sohanpanah, 1989], possibly combined with the computation of topology on the fly [ESRI, 2005] also become intractable in higher dimensions. Implicit models, such as Constructive Solid Geometry and Boolean set operations on half-spaces, are not sufficient by themselves and generally have to be converted into another—more explicit—model in order for them to be analysed and visualised [Mäntylä, 1988].

Because of these shortcomings, the most promising option is to use topological vector nD data structures. Some of these structures involve the subdivision of all objects into geometric simplices (i.e. the generalisation of 2D triangles and 3D tetrahedra to arbitrary dimensions) forming a simplicial complex. There are several such structures [de Floriani and Hui, 2005], which tend to be conceptually simple and easy to implement. However, performing the subdivision of each object into simplices

requires using a constrained or conforming triangulation [Shewchuk, 2000, 2008], which is very hard to realise in practice in more than three dimensions (since there is no known available software).

Ordered topological models, which are able to represent more general cell complexes using the internal structure of a simplicial complex, therefore have clear advantages. Generalised maps (g-maps) [Lienhardt, 1994] and the cell-tuple structure [Brisson, 1993] are capable of storing a large class of cell complexes, including orientable and unorientable manifold objects of arbitrary dimension, including objects with boundaries by Poudret et al. [2007]. They have been implemented in libraries and software, including Moka[2] and Gocad[3]. Combinatorial maps [Lienhardt, 1994] require defining an orientation for the objects, but use half the storage space of generalised maps. They have also been implemented in mature libraries, such as CGAL[4] and CGOGN[5] [Kraemer et al., 2014].

2.5 Construction algorithms for a 4D model

We have previously developed two low-level construction algorithms for nD objects. These can be used as a base to create 4D models in following manner: (1) when the 3D objects remain relatively unchanged along the LOD dimension, the 4D model can be *extruded* from 3D to 4D, creating simple prismatic objects, which are possibly further processed; or (2) when arbitrarily shaped 4D models are needed, the individual faces and volumes on the boundary of the 4D model are *incrementally constructed and linked*.

The first approach uses a dimension-independent extrusion operator [Arroyo Ohori and Ledoux, 2013; Arroyo Ohori et al., 2015b]. It 'lifts' a n-dimensional model into $(n + 1)$-dimensional space by

---

[2]http://moka-modeller.sourceforge.net
[3]http://www.gocad.org/w4/
[4]http://www.cgal.org
[5]http://cgogn.unistra.fr

4

assigning *each n*-dimensional object a range along which it exists, generating a set of prismatic polytopes, analogous to prisms in 3D. Intuitively, this new cell complex consists of 'base' and 'top' faces constructed from the original cell complex, with $r_{\min}$ and $r_{\max}$ respectively added as n-th coordinates in their o-attributes. 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. This method is poorly suited for our case since the two LODs have to have the same topology, which is rare in practice (we show in Section 4 they in almost all cases differ significantly).

The second approach, which is more generic and more suitable to our case, allows us to use the arbitrary cell complexes that have been generated using the linking rules presented in this paper. This entails that all the bounding volumes linking two models have to be generated first, starting from defining all the necessary 4D points of the form (x, y, z, l), where (x, y, z) are the three spatial dimensions and l is a point on the LOD axis. These are then linked in increasing dimension to form faces, volumes and the 4-cell for each 4D object. The schemes for generating these are described in Section 4.2, and these can be appropriately linked together using the incremental construction operator [Arroyo Ohori et al., 2014], which takes a set of $(i-1)$-cells forming a closed i-cell, and joins them along their common $(i-2)$-cells (ridges) to create the aforementioned i-cell by generating all the topological relationships between them.

2.6 Slicing to extract a 3D model at a certain LOD

In order to obtain a 3D model from an integrated 4D model at a certain LOD, it is possible to reduce the dimension of a spatial object by *slicing* [Arroyo Ohori et al., 2013]. In this operation, a higher-dimensional set of objects is geometrically intersected with another lower-dimensional object—often unbounded along some dimensions and orthogonal to the axes of the others. The

end result can then be given in terms of the lower dimensional space induced by this object, which is equivalent to an orthographic projection of the intersection to a coordinate system describing the vector space where it lies.

In order to extract a 3D model at a certain LOD l from a 4D model, the object used is an unbounded 3D volume embedded in 4D. This is similar to how an unbounded plane can be embedded in 3D to generate a cross-section of an object. The unbounded 3D volume is therefore a region of 3D space that is similar to $\mathbb{R}^3$ and orthogonal to the LOD axis l, covering the range $(-\infty, \infty)$ along the three spatial dimensions (x, y, z) and having a single value along the LOD axis l. Note that although in the continuation of this paper we consider only orthogonal slicing along the LOD axis (producing a 3D model of a fixed LOD), mixed-LOD models obtained from non-orthogonal slices planes are possible: van Oosterom and Meijers [2014] demonstrate one case in 3D (which has not been generalised to 4D).

3 Methodology to model LOD as an extra geometric dimension in a true 4D model

Incorporating non-spatial characteristics as additional dimensions in the geometric sense results in a fabric of higher-dimensional space where objects can be embedded. As an extension of the standard spatial data modelling concepts, this space can be modelled as the d-dimensional Euclidean space $\mathbb{R}^d$. In this manner, each dimension is defined by an independent (perpendicular) axis, and a point in this space is defined by an d-tuple of coordinates.

This higher-dimensional space is then populated by a set of objects of dimension from zero (i.e. points) up to d (i.e. d-polytopes). These are non-overlapping and thus, together with the empty space surrounding them, induce a partition of the space, analogously to a set of polygons forming a planar

partition in 2D. As is usual in GIS, we will assume that these objects have linear (i.e. flat) geometries, which significantly simplifies their representation and most operations.

In the specific case of a 4D (3D space+LOD) model, this means that we have an additional LOD axis l and a point in 4D space is defined by a tuple of coordinates (x, y, z, l). It is worth noting that the LOD axis should be properly parametrised, defining quantifiable values for every fixed LOD in a model, or alternatively a function that does so.

The 4D space is filled with a set of non-overlapping polychora, in which a 3D object (e.g. a building) at *all* of its different LODs is represented as a single 4D object. This 4D object is bounded by a set of volumes, two of them being the object at its lowest and highest LOD, and several lateral ones formed by filling the space between corresponding faces across LODs. When sliced, these respectively correspond to the volumes and bounding faces of an extracted 3D model.

Observe that a true 4D model permits us not only to link the different representations of an object together, but it goes further since it enables us to model and store explicitly more complex relationships between the objects. An example is when multiple objects are merged into one; this case is cumbersome to handle with IDs (which ID should the new object get?), however if the topological relationships between the objects are explicitly stored, it suffices to analyse these to detect such a case. If the multiple objects would be simultaneously changing along multiple parameters, such as changing shape (e.g. because of a simplification process) and moving over time, then another dimension (time) could be added to the model as well, yielding a 5D model [van Oosterom and Stoter, 2010] that can be represented as a set of 5D objects modelled as a 5D cell complex[6].

We represent mathematically the aforementioned 4D model with a 4D cell complex, a subdivision of space into cells, such that for all $i, 0 \le i \le 4$, an i-dimensional cell (i-cell) is a topological object homeomorphic to an i-ball (e.g. point, segment, disk, ball, etc.). For all $i > 0$, every i-cell in the complex has a number of $(i-1)$-cells (*faces*) as its boundary which are also part of the complex. The common boundaries of two i-cells is given by their intersection, returning a set of $(i-1)$-cells which are also part of the complex. Every 0-cell (i.e. a vertex) is attached to a tuple of coordinates (x, y, z, l) representing a point in 4D space. This 4D cell complex can then be stored in any of the data structures presented in Section 2.4.

4 Constructing a 4D model from a 3D city model and its LODs

The buildings in a 3D city model are often modelled in multiple LODs. For instance, in the CityGML standard [OGC, 2012] five discrete LODs can be stored (from the 2D footprint of the building up to a representation where the windows, doors and walls and even indoor objects are all modelled in detail). These different representations are in most cases not derived from the most detailed LOD (e.g. with generalisation methods), but are collected with different techniques, often for different purposes, and thus the resulting representations do not necessarily have easy-to-identify correspondences. The same object can be slightly displaced at different LODs, an object can be an aggregate of other objects (think of a terraced house: either each house is represented or one volume for the whole row), or can be modelled in an entirely different way.

The construction of a 4D model from existing LODs of a 3D city model consists of three steps:

1. identifying corresponding 0D–3D cells;

---

[6]Such a model could be represented as a 5D cell complex with an analogous definition as the 4D cell complex presented here. Note that although this is a straightforward change in terms of the model and its storage, 5D operations to create and manipulate such objects would have to be developed in order to make such a model useful.

2. linking them by creating 1D–4D cells connecting them;

3. using an incremental construction algorithm [Arroyo Ohori et al., 2014] to build a 4D cell complex using all 0D–4D cells.

We first present in this section various methods to identify corresponding cells in different LODs of a 3D object (Section 4.1), and then we present four linking schemes to construct a 4D model (Section 4.2). As we demonstrate in Section 5 on use cases, the 4D cells created by the linking schemes are used to construct 4D cell complexes with different properties and shapes, each of which has its own requirements.

4.1 Step 1: Identifying corresponding cells in 3D models

Constructing a 4D model from a sequence of 3D models largely depends on the identification of the corresponding 0-, 1-, 2-, and 3-cells between these 3D models. The aim of this identification is to create a mapping between the 3D models that preserves the topological relationships between the elements in the models so as to create a valid 4D model.

Considering 3D models at different LODs, this identification will often result in matching cells of different dimensions, commonly with some cells in the 3D model at the highest LOD being matched to cells of lower dimension in the 3D model at the lowest LOD. Also, these correspondences will often not result in a one-to-one mapping: groups of adjoining cells in one model, most often in the one at the highest LOD, will commonly be matched to a single cell in the other model.

The identification of matching cells should be done using a combination of the following, arguably in order of preference:

Attributes Using the semantic information stored in the cells, when it is available. For instance, matching two cells that are known to be equivalent through the use of IDs, or if knowledge is kept during the generalisation process, matching a cell with one that is known to be a simplified version of it.

Topology When there is a one-to-one mapping (a bijection) between two cells that preserves all their topological relationships, this mapping is known as an *isomorphism* and the cells are said to be *isomorphic*. Therefore, an isomorphism between two cells already gives a matching between them, although it might be important to check that the isomorphism is compatible with the matching of the other cells in the model and with geometric constraints. Relevantly, Gosselin et al. [2011] describe how to compute isomorphism in a generalised map of any dimension. Another more complex possibility is using subgraph isomorphism on unmatched portions of a generalised map [Eppstein, 1999], even if this problem is known to be NP-complete.

Another way to use topology is to use the topological relationships between cells in order to infer matchings for the remaining cells [Hampe et al., 2003]. This is explained more concretely in the example of Figure 3.

Geometry Using geometric computations, such as those based on computing similarity metrics, simply matching unmatched cells in one model to their nearest neighbour in another model, or attempting to minimise the Earth Mover's Distance (EMD) [Rubner et al., 1998] between them. It is however important to compute these matches using constraints that generally preserve the relative positions and topological relationships between the cells. For instance, a greedy algorithm could match cells iteratively, cascading these matches to adjacent cells (in all models) or rejecting matches that would violate a geometric or topological constraint.

A final possibility to assist when matching cells is to allow splitting an *i*-cell into multiple *i*-cells by adding cells of lower dimensions in a manner that does not alter the

geometry of the cell. For instance, a face can be split into multiple faces by adding a vertex in its interior and creating edges that link it to some of the vertices of the face. This can make two cell complexes isomorphic and directly allow for a one-to-one mapping between two cell complexes.

4.2 Step 2: Linking corresponding cells

Based on the matches that were found between cells, which mathematically define a map between the 3D cell complexes of the 3D models, we can subsequently link them to construct a 4D cell complex. For this, it might be necessary to create or modify 0D-3D cells in the input cell complexes, as well as to create new 1D-4D cells that lie *between* the cell complexes. The resulting 4D cell complex is then embedded in 4D space by assigning new 4D coordinates for every point. We propose four different basic linking schemes, which are shown in Figure 1.

4.2.1 Method #1: Simple linking of corresponding cells

Links are constructed between the corresponding cells of an object at two different LODs, and if a cell has no corresponding cell then it is ignored. While this allows us to easily construct a 4D cell complex in the cases where all cells in the lower LOD model have a corresponding cell in the higher LOD model, when this is not the case, the result will consist of an incomplete 4D cell complex.

To ensure a complete one, cells often need to be split (e.g. those separated by the red dotted line in Figure 1a), which can be performed using geometric intersections. While this is possible in 3D and tools are readily available (see for instance Granados et al. [2003]; Hachenberger [2006]), it should be noticed that the generalisation of this scheme to higher dimensions (i.e. 5D+) is not easy in practice since no robust intersection tools in more than 3D are available. Observe that if a 4D cell complex generated

using this method is sliced at an intermediate LOD, the result is exactly that of the lower LOD.

4.2.2 Method #2: Unmatched cells are collapsed to existing ones

No modifications are made to the 3D models, which is in practice a significant advantage since no complex geometric operations need to be performed and the size of the cell complex will be smaller than that of the one where cells are modified. Instead of geometric operations, unmatched cells in the higher LOD model are linked to nearby matched cells *of a possibly lower dimension* in the lower LOD model while preserving certain geometric and topological constraints (e.g. preserving adjacency and incidence between cells). This implies that some cells will be collapsed (e.g. an edge can be mapped to a vertex), and the cells must be linked with care to ensure that a valid 4D cell complex is created (e.g. no two cells should intersect).

For instance, assume that the left eave of the roof of the house in the middle LOD model in Figure 1b has been (arbitrarily) matched to the roof of the low LOD model, with the right eave remaining unmatched as no unmatched cells remain in the low LOD model. In this case, using the knowledge that the roof and right wall are adjacent in the low LOD model but their corresponding cells (respectively the left eave and right wall) are separated by the right eave in the middle LOD model, the right eave can be collapsed to the common vertex lying between the two (upper right). Using such a mapping, the topological relationships between the cells will be preserved, with the exception of those involving the collapsed cells and those incident or adjacent to them.

Notice that when the mapping has faults and the resulting 4D cell complex thus has geometric problems (e.g. intersecting cells), the slicing operation might not have any geometric meaning, but the main advantage of the integration—consistency—will nevertheless still be guaranteed for all other cells. Ensuring that cells preserve their
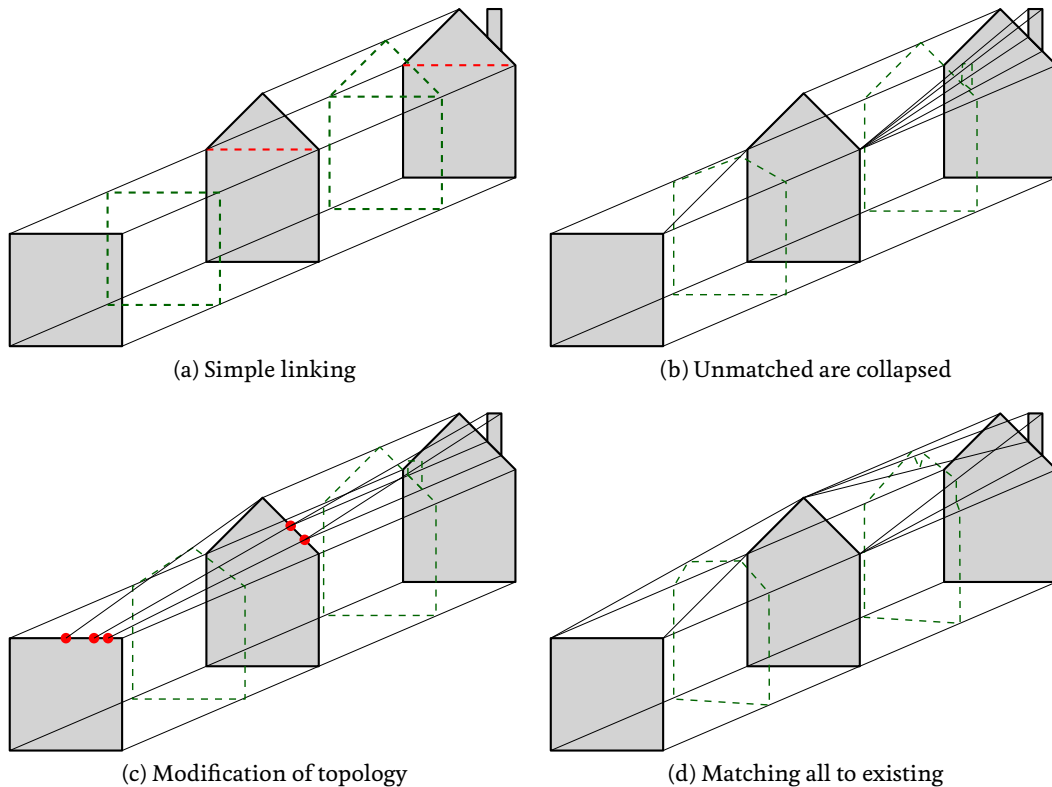
(a) Simple linking

(b) Unmatched are collapsed

(c) Modification of topology

(d) Matching all to existing

Figure 1: The four linking schemes for three LODs of a house, here depicted in 2D. The objects that would obtained by slicing between the LODs can be seen in dashed green contours; the red dashed lines reflect the cells that need to be added and split in order to ensure a valid 3D (2D+LOD) cell complex.

topological relationships and form a partitioning of space in 4D is challenging, and is part of our future work. Finally, observe that even if a combinatorially and geometrically valid 4D cell complex is constructed, the 3D object obtained by slicing might not be consistent with the reality; notice how the chimney in Figure 1b becomes increasingly smaller and closer to the right eave of the roof because of the way the cells have been linked.

4.2.3 Method #3: Modifying the topology

To ensure that there is a mapping between all the cells, we can split or merge cells so that the topology (combinatorial structure) of the objects is identical. For instance, operations like removal and contraction [Damiand and Lienhardt, 2003]

can be used to simplify the more complex object(s) to make them match the simpler one(s) using an iterative process. On the other direction, it is possible to first identify for every cell in the lower LOD model one or more corresponding cells in the higher LOD model, then split cells in the lower LOD model so that their topology is the same as in the higher LOD model as the one to which it must be linked.

In Figure 1c, for the lowest LOD, this implies first finding multiple matches for the roof cells of the lower LOD models, which then need to be split into multiple cells by the insertion of new vertices. For instance, these can be located at the closest location that lies on the matched lower LOD cell for every higher LOD cell. As this example shows, *all* the representations of an object where this approach is used will end up having

9

the same topology. This results in increased storage space and the possibility of degenerate cells e.g. multiple vertices at one location. The geometric operations necessary to split cells can be rather intricate as well. Observe that slicing however results in a different representation of the object, one where it smoothly morphs into the one at lower LOD (e.g. the tip of the roof is slowly lowered as the LOD decreases).

4.2.4 Method #4: Matching all cells to existing ones

As is the case with method #2, this method does not require modifying the topology of the objects. The main difference with it is that cells in the higher LOD model are not necessarily collapsed to a lower dimensional cell in the lower LOD model but are instead matched to one or more cells *of any dimension* while also preserving certain geometric and topological constraints. In Figure 1d, observe that the tip of the roof of the middle LOD model (a point) is matched to the roof of the lowest LOD (an edge since we have a 2D representation) and that the 2 edges representing the middle-LOD roof are matched to the two corners of the lower-LOD roof (points). Slicing thus creates a truncated roof having 3 edges. This can be achieved by matching all cells that have a clear correspondence first, then attempting to match groups of unmatched cells while preserving the topological relationships between cells.

For instance, in Figure 1d it is possible to first match the base and walls of the houses in the lower and middle LOD models, then match the remaining vertex and left/right edges in the middle LOD model respectively to the roof edge and left/right vertices. Observe that in this process, the tip of the roof of the middle LOD model (a point) is matched to the roof of the lowest LOD (an edge) and that the two eaves of the roof (edges) are matched to the two corners of the roof (points) in the lowest LOD. Slicing the resulting 4D cell complex creates a truncated roof having 3 edges. The matches for the chimney to other elements in Figure 1d is achieved by matching the chim-

ney top to the right eave, and the remaining vertices and edges to its left and right sides respectively to the roof tip and right eave/wall vertices. The result is that while the chimney looses resemblance to reality, it slowly converges to the roof in the middle LOD model.

5 Use cases

We present in this section practical examples that describe the matching and the linking of cells for a few simple 3D models representing the same object(s) at different LODs.

5.1 Using the method #1: Simple linking

Figure 2 shows an example where two LODs for a building are linked in such a way that only matched cells are involved. First ob-
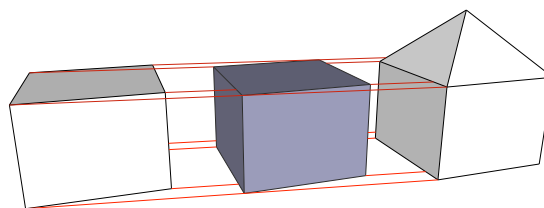

Figure 2: Two LODs of a house simply linked and the intermediate LOD obtained.

serve that since the 2 objects are not isomorphic, some cells are not matched (the ones representing the roof of the higher-LOD model). Observe also that the roof of the lower-LOD model has no match in the higher-LOD model. Thus, to construct a 4D cell complex, the flat roof geometry has to be added to the higher-LOD model. Then, the corresponding cells can be linked. Although it is possible to generate this 4D model by generating the $(i + 1)$-cells that connect a pair of corresponding i-cells and linking all of them together, it is easiest to extrude one cell complex along the range between the two LODs. This already generates the proper combinatorial structure of

the 4D model, and the final cell complex can be obtained by then moving the vertices of the face representing the model at the other LOD so as to match the geometry of the other model at its LOD. Moreover, since we are assuming a linear cell complex and thus only the vertices are storing the geometry of the model, it is only necessary to move the vertices at the lowest LOD without a corresponding vertex at the highest LOD.

5.2 Using the method #2: Collapsing

Figure 3 shows an example with a 3D model at two LODs with differing geometry and topology. The 4D model has been obtained
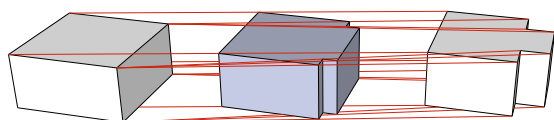

Figure 3: Two LODs of a house with differing geometry and topology are integrated into a 4D model by collapsing cells in the model at the highest LOD.

by first matching the 2-cells with known correspondences (the left, right, front and back large faces) and inferring that the other faces in the model at the highest LOD (right) should be collapsed based on their adjacency relationships with the matched faces. For example, since the front and right faces are adjacent in the lowest LOD but not in the highest LOD, the two faces between them[7] should be collapsed to their common boundary (i.e. their intersection: the edge between them). This example also shows that the topological relationships between the cells are nevertheless preserved with the exception of those that involved collapsed cells. The new topological relationships do however connect cells around the former

[7]Combinatorially, this involves a search for a path between the front and right faces in the graph of the model at the highest LOD, such search being limited to the nodes representing the aforementioned faces and those representing faces that are *not* present in the model at the lowest LOD.

collapsed cells. Note that the 3D model resulting from slicing the 4D model created in this way at an intermediate LOD (middle) will be isomorphic to the model at the highest LOD.

5.3 Using method #3: Modifying the topology

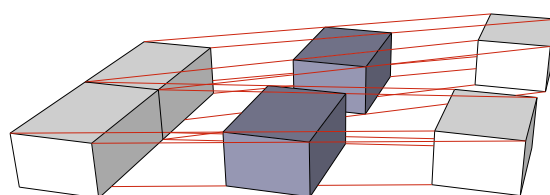Figure 4 shows an example of two 3D models being aggregated. In order to create a

Figure 4: Two LODs of two houses being aggregated are integrated into a 4D model by modifying the topology of the model at the lowest LOD so as to match the topology of the model at the highest LOD.

4D model from this situation, the topology of the simpler of the two models is modified, splitting the single volume into equal two adjacent ones, effectively resulting in a cell complex that also has four more vertices, four more edges, and four of its faces split into two. Note that the two models are however not isomorphic since the common face of the two houses in the lowest LOD becomes two disconnected faces in the model at the highest LOD, but that if we disregard this topological relationship, the two models can be correctly matched independently.

5.4 Combination of methods #2 and #3

Figure 5 shows a more complex example with three LODs which are linked using a combination of schemes: collapsing and modifying the topology of one of the models. Most of the cells in the highest LOD can be directly matched to cells in the middle LOD, with the exception of those that are
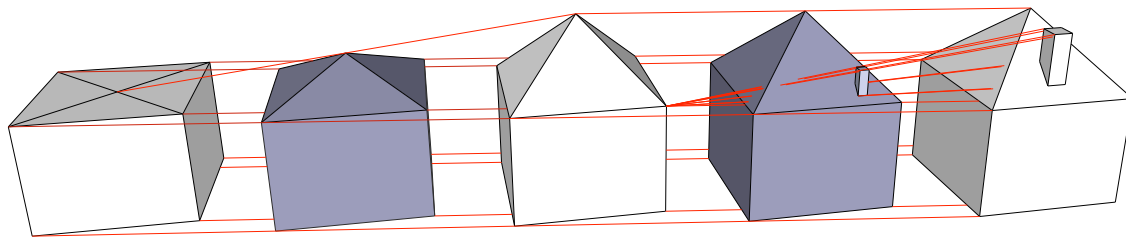
Figure 5: Three LODs of a 3D model of a house are integrated into a 4D model by modifying the topology of the model at the lowest LOD and collapsing a part of the model in the highest LOD.

part of the chimney. As these comprise a small object, these are simply collapsed to a single point in the middle LOD. Matching the roof cells in the lowest and middle LODs is however more complex since collapsing it to a point would ignore its adjacency with the body of the house and therefore not preserve its topology. The best solution is therefore to modify the topology of the lowest LOD in order to split the top face of the cubic house (which we know is a roof based on its attributes) into 4 faces, making the model isomorphic to the middle LOD.

5.5 Using method #4: matching to existing cells

Figure 6 shows two of the LODs of the previous example, but matches the cells of the roof of the house to existing cells rather than modifying their topology. After at-
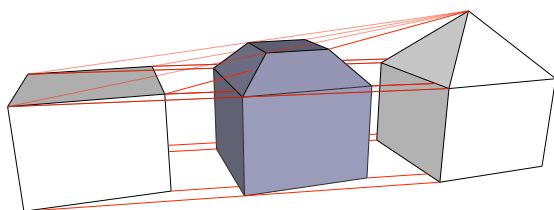

Figure 6: Two LODs of a 3D model of a house (left and right) are linked despite not being isomorphic, with an intermediate LOD that shows the result of slicing the construction at an intermediate LOD (centre).

tempting to match corresponding cells, the top face in the lowest LOD and the top

four faces in the highest LOD remain unmatched. If we collapse each top face in the highest LOD to the closest top *edge* in the lowest LOD (i.e. the edge that forms the bottom of the triangular face in the highest LOD) and the top vertex in the highest LOD (which lies between the faces) is linked to the top face in the lowest LOD, a four-sided pyramid is generated. Slices from it are shown as four trapezoidal faces in the sliced intermediate LOD. Then, if we collapse the top face in the lowest LOD to the top vertex of the highest LOD, another four-sided pyramid is generated. A slice from this one is shown as a square face at the top of the sliced intermediate LOD.

This particular mapping, which correctly preserves all topological relationships between the cells, is interesting since it shows that cells are not necessarily only collapsed from higher LODs to lower LODs. It is worth noting that the result of this mapping is a set of cells that bound the model along the LOD dimension, so that an i-cell and a j-cell that are matched result in a k-cell lying between them, where $k = \max(i, j) + 1$. Concretely, if for instance a 0-cell (the tip of the roof) is matched to the flat roof (2-cell), then the resulting links will create a tetrahedra (a 3-cell). Note also that although the rules needed to generate such a mapping might be more complex, the cell complex generated is identical in size than the equivalent model according to the scheme in Figure 1b.

6 A concrete example: implementing cell matching to construct a 4D model

In order to show how our linking methods work in practice, we have implemented the model shown in Figure 6 using CGAL Linear Cell Complexes and the incremental constructor operator described in Arroyo Ohori et al. [2014]. This model was chosen as it uses most of the linking methods discussed in the Section 4.2: the body of the house in both LODs is directly linked (Method #1), the top face of the house in the lower LOD is collapsed to the tip of the roof in the higher LOD (Method #2), and the roof vertices/edges in the lower LOD are connected to existing roof edges/vertices in the higher LOD (Method #4).

First of all, the 17 vertices of the two 3D models are created as 4D points of the form (x, y, z, l). Afterwards, these are first used to define the 35 faces of the model, the faces are used to define the 12 volumes, and the volumes to define the single 4-cell. Notice that these include faces and volumes within each of the two volumes of the input 3D models, but also include some faces and volumes that lie *between* the two, i.e. having vertices and faces in both input 3D models. Excerpts of the code to generate the 4D cell complex are shown in Figure 7.

The resulting 4D model was then validated by checking the properties of a valid combinatorial map (cf. Lienhardt [1994]). In short, we tested whether the darts (combinatorial simplices) in the map formed correct involutions or permutations and whether any darts remained free after the operations. We also validated individual parts of the model (the triangular or square faces and the parallelepiped- or pyramid-shaped volumes) by verifying that they were isomorphic to similar objects that were known to be valid [Gosselin et al., 2011].
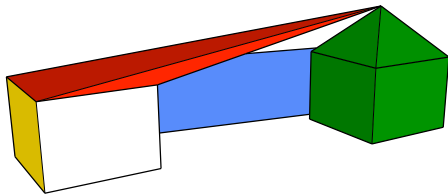
7 Discussion and future work

While integrating different LODs of the same 3D object in a 4D model is technically more complex than maintaining them separately, we have shown that it is possible to create such a 4D model by identifying matching elements in different LODs, linking them to obtain 4D primitives, and finally constructing and storing a 4D cell complex with algorithms and data structures that are readily available. We believe the advantages of this integration to be many.

First, it provides a fundamental solution to the maintenance of these models since any update could be propagated to other LODs; consistency of the model at all LOD can thus be ensured, e.g. using the validity checks in Gröger and Plümer [2011]. Such a 4D model thus supports the *reuse* of collected 3D data.

Second, since every application often requires its own particular LOD specifications [Biljecki et al., 2014b], these could in theory be extracted from the 4D model. We have shown that different linking schemes yield 4D models having different properties and 3D models (obtained from slicing) that are useful for different application. Moreover, if we generalise the slicing operator and allow slices to be non-orthogonal to the LOD axis (orthogonal slices yield a fixed LOD), mixed-LOD 3D models are obtained. That is, we can obtain a 3D model containing objects at different LODs. As an analogy to a view-dependent LOD in computer graphics [Hoppe, 1997; De Berg and Dobrindt, 1998], we envision applications where more details are available in the vicinity of something of interest. Visualisation is an obvious example, but spatial analyses such as noise simulations are also interesting: buildings closer to the sources of emission (e.g. a railway or a factory) would have more details and those further away be a coarser representation.

Third, it allows us to derive *intermediate* LODs, which enable us to refine the five traditional LODs of CityGML, which are deemed as insufficient [Biljecki et al.,

(a)

```
// 2: Left of first house (yellow)
faceLists.push_back(...);
faceLists.back().push_back(vertices[0]);
faceLists.back().push_back(vertices[2]);
faceLists.back().push_back(vertices[6]);
faceLists.back().push_back(vertices[4]);

//21: Back right vertical edge (blue)
faceLists.push_back(...);
faceLists.back().push_back(vertices[3]);
faceLists.back().push_back(vertices[7]);
faceLists.back().push_back(vertices[15]);
faceLists.back().push_back(vertices[11]);

for (int i = 0; i < 35; ++i)
    faces.push_back(
      builder.get_facet_from_vertices(
        faceLists[i].begin(),
        faceLists[i].end(), false).first);

// 1: Right house (green)
volumeLists.push_back(...);
volumeLists.back().push_back(faces[6]);
volumeLists.back().push_back(faces[7]);
volumeLists.back().push_back(faces[8]);
volumeLists.back().push_back(faces[9]);
volumeLists.back().push_back(faces[10]);
volumeLists.back().push_back(faces[11]);
volumeLists.back().push_back(faces[12]);
volumeLists.back().push_back(faces[13]);
volumeLists.back().push_back(faces[14]);

// 11: Roof from left to right tip (red)
volumeLists.push_back(...);
volumeLists.back().push_back(faces[31]);
volumeLists.back().push_back(faces[32]);
volumeLists.back().push_back(faces[33]);
volumeLists.back().push_back(faces[34]);
volumeLists.back().push_back(faces[5]);

for (int i = 0; i < 12; ++i)
    volumes.push_back(builder.get_cell<3>(
      volumeLists[i].begin(),
      volumeLists[i].end()).first);

LCC::Dart_handle house4d = builder.get_cell<4>(
  volumes.begin(), volumes.end()).first;
```

```
float point_coordinates[][4] = {
    // Left house
    {0, 0, 0, 0}, // 0
    {1, 0, 0, 0}, // 1
    {0, 1, 0, 0}, // 2
    {1, 1, 0, 0}, // 3
    {0, 0, 1, 0}, // 4
    {1, 0, 1, 0}, // 5
    {0, 1, 1, 0}, // 6
    {1, 1, 1, 0}, // 7

    // Right house
    {0, 0, 0, 1}, // 8
    {1, 0, 0, 1}, // 9
    {0, 1, 0, 1}, // 10
    {1, 1, 0, 1}, // 11
    {0, 0, 1, 1}, // 12
    {1, 0, 1, 1}, // 13
    {0, 1, 1, 1}, // 14
    {1, 1, 1, 1}, // 15
    {.5, .5, 1.5, 1} // 16
};

for (int i = 0; i < 17; ++i) {
    points.push_back(Point(4,
        point_coordinates[i],
        point_coordinates[i]+4));
    vertices.push_back(
        builder.get_vertex(points[i]));
}
```

(c) Coordinates of the 17 vertices

(b) Constructing other cells

Figure 7: Code excerpts that show how vertices are created based on 4D points, faces as cycles of vertices, volumes as sets of faces, and 4-cells as sets of 3-cells. The colours referred to in (b) correspond to the highlighted faces and volumes in (a).

14

2014a], and to derive continuous LODs, i.e. LODs at an arbitrary level.

In the future, we plan to develop high-level operations to manipulate 4D objects, as well as to implement various slicing methods to extract 3D objects from a 4D space partition. Other characteristics of geographical information can also be modelled as extra geometric dimensions in a similar manner as the LOD, time being an obvious example. While it as been often mentioned that the integration of space and time is desirable [Hornsby and Egenhofer, 2000; Peuquet, 2002; Raper and Livingstone, 2001; Worboys, 1994; de Roo et al., 2013], in practice it is included as separate attribute either of an object or an event. The space-time cube of Huisman et al. [2013] proposes the use of time as an extra geometric dimension, but its aim is merely to provide spatial insight into the temporal aspect rather than on realising a data structure to handle changes upon position, attributes and/or extent of the objects in a unified space-time continuum. We plan to investigate in the future how to realise the integration of space and time. It would not only support changes at discrete moments, as currently supported by most spatio-temporal models via *timestamps* and versioning, but also continuous temporal changes to describe the movement or change of objects independently from their object identification. The main challenge we face is in obtaining datasets that reflect the history of a given building or area (in 3D). Three-dimensional city models are in their infancy, and usually if a model is available it is only for the current situation.

Acknowledgements

References

Ken Arroyo Ohori and Hugo Ledoux. Using extrusion to generate higher-dimensional GIS datasets. In Craig Knoblock, Peer Kröger, John Krumm, Markus Schneider, and Peter Widmayer, editors, *SIGSPATIAL'13: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 398–401. ACM, Orlando, United States, November 2013.

Ken Arroyo Ohori, Filip Biljecki, Jantien Stoter, and Hugo Ledoux. Manipulating higher dimensional spatial information. In *Geographic Information Science at the Heart of Europe. Proceedings of the 16th AGILE International Conference on Geographic Information Science*, May 2013.

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, *ICAA 2014*, volume 8321 of *Lecture Notes in Computer Science*, pages 36–47. Springer International Publishing Switzerland, Kolkata, India, January 2014.

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*, 2015a.

Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. A dimension-independent extrusion algorithm using generalised maps. *International Journal of Geographical Information Science*, 2015b.

Lars Bernard, Benno Schmidt, and Ulrich Streit. AtmoGIS - integration of atmospheric models and GIS. In T.K. Poiker and N. Chrisman, editors, *Proceedings of the 8th International Symposium on Spatial Data Handling*, 1998.

Filip Biljecki, Hugo Ledoux, and Jantien Stoter. Redefining the Level of Detail for 3D models. *GIM International*, 28(11):21–23, November 2014a.

Filip Biljecki, Hugo Ledoux, Jantien Stoter, and Junqiao Zhao. Formalisation of the level of detail in 3D city modelling. *Computers, Environment and Urban Systems*, 48: 1–15, November 2014b.

Filip Biljecki, Hugo Ledoux, and Jantien Stoter. Improving the Consistency of Multi-LOD CityGML Datasets by Removing Redundancy. In Martin Breunig, Al-Doori Mulhim, Edgar Butwilowski, Paul Vincent Kuper, Joachim Benner, and Karl-Heinz Häfele, editors, *Lecture Notes in Geoinformation and Cartography. 3D Geoinformation Science. The Selected Papers of 3D GeoInfo 2014*, pages 1–17. Springer International Publishing, Dubai, UAE, 2015.

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

J H Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, October 1976.

Volker Coors and Sascha Flick. Integrating Levels of Detail in a Web-based 3D-GIS. In Robert Laurini, Kia Makki, and Niki Pissinou, editors, *Proceedings of the 6th ACM international symposium on Advances in Geographic Information Systems*, pages 40–45, Washington, DC, USA, November 1998. ACM.

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.

Mark De Berg and Katrin T G Dobrindt. On levels of detail in terrains. *Graphical Models and Image Processing*, 60(1):1–12, January 1998.

Leila de Floriani and Annie Hui. Data structures for simplicial complexes: an analysis and a comparison. In M. Desbrunn and H. Pottmann, editors, *Eurographics Symposium on Geometry Processing*. The Eurographics Association, 2005.

B. de Roo, N. van de Weghe, J. Bourgeois, and Ph. de Maeyer. The temporal dimension in a 4D archaeological data model: applicability of the geoinformation standard. In *Proceedings of the 8th 3DGeoInfo Conference & WG II/2 Workshop, 27 – 29 November 2013, Istanbul, Turkey*, volume II-2/W1 of *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. ISPRS, November 2013.

T. Devogele, J. Trevisan, and L. Raynal. Building a multi-scale database with scale-transition relationships. In *Proceedings of the 7th International Symposium on Spatial Data Handling*, pages 337–351, Delft, Netherlands, 1996.

Jürgen Döllner and Henrik Buchholz. Continuous level-of-detail modeling of buildings in 3D city models. In Cyrus Shahabi and Omar Boucelma, editors, *GIS '05 Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 173–181, Bremen, Germany, November 2005.

David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3 (3):1–27, 1999.

ESRI. *GIS Topology*. ESRI, July 2005.

Hongchao Fan and Liqiu Meng. A three-step approach of simplifying 3D buildings modeled by CityGML. *International Journal of Geographical Information Science*, 26(6):1091–1107, June 2012.

Waldemar Celes Filho, Luiz Henrique de Figueiredo, Marcelo Gattass, and Paulo Cezar Carvalho. A topological data structure for hierarchical planar subdivisions. Technical Report CS-95-53, Department of Computer Science, University of Waterloo, 1995.

Tassilo Glander and Jürgen Döllner. Abstract representations for interactive visualization of virtual 3D city models. *Computers, Environment and Urban Systems*, 33(5):375–387, September 2009.

Stéphane Gosselin, Guillaume Damiand, and Christine Solnon. Efficient search of combinatorial maps using signatures.

Theoretical Computer Science, 412(15):1392–1405, March 2011.

Miguel Granados, Peter Hachenberger, Susan Hert, Lutz Kettner, Kurt Mehlhorn, and Michael Seel. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, and implementation. In *Proceedings 11th Annual European Symposium on Algorithms (ESA'03)*, number 2832 in Lecture Notes in Computer Science, pages 654–666, Budapest, Hungary, 2003.

Gerhard Gröger and Lutz Plümer. Provably correct and complete transaction rules for updating 3d city models. *Geoinformatica*, 2011.

Gerhard Gröger and Lutz Plümer. CityGML – Interoperable semantic 3D city models. 71:12–33, July 2012.

R Guercke, T Götzelmann, C Brenner, and M Sester. Aggregation of LoD 1 building models as an optimization problem. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(2):209–222, March 2011.

Peter Hachenberger. *Boolean Operations on 3D Selective Nef Complexes Data Structure, Algorithms, Optimized Implementation, Experiments and Applications.* PhD thesis, Saarland University, 2006.

Mark Hampe, Karl-Heinrich Anders, and Monica Sester. MRDB applications for data revision and real-time generalisation. In *Proceedings of the 21 International Cartographic Conference*, pages 192–202, 2003.

Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH '97*, pages 189–198, New York, NY, USA, 1997. ACM Press.

Kathleen Hornsby and Max J. Egenhofer. Identity-based change: A foundation for spatio-temporal knowledge representation. *International Journal of Geographical Information Science*, 14:207–224, 2000.

Otto Huisman, Irvin Feliciano Santiago, Menno-Jan Kraak, and Bas Retsios. Developing a geovisual analytics environment for investigating archaeological events: Extending the space-time cube. *Cartography and Geographic Information Science*, 36 (3):225–236, 2013.

Pierre Kraemer, Lionel Untereiner, Thomas Jund, Sylvain Thery, and David Cazier. CGoGN: n-dimensional meshes with combinatorial maps. In J. Sarrate and M. Staten, editors, *Proceedings of the 22nd International Meshing Roundtable*, pages 485–503. Springer International Publishing Switzerland, 2014.

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

David Luebke, Martin Reddy, Jonathan D Cohen, Amitabh Varshney, Benjamin Watson, and Robert Huebner. *Level of detail for 3D graphics.* Morgan Kaufmann Pub, 2003.

Martti Mäntylä. *An introduction to solid modeling.* Computer Science Press, New York, USA, 1988.

N. C. Mason, M. A. O'Conaill, and S. B. M. Bell. Handling four-dimensional georeferenced data in environmental GIS. *International Journal of Geographical Information Systems*, 8(2):191–215, 1994.

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

Liqiu Meng and A Forberg. 3D building generalisation. In W Mackaness, A Ruas, and T Sarjakoski, editors, *Challenges in the portrayal of geographic information: Issues of Generalisation and Multi Scale Representation*, pages 211–232. Elsevier Science, Amsterdam, the Netherlands, 2007.

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

OGC. OGC city geography markup language (CityGML) encoding standard. Open Geospatial Consortium inc., 2012.

Donna J. Peuquet. *Representations of Space and Time.* Guilford Press, 2002.

Lutz Plümer and Gerhard Gröger. Achieving integrity in geographic information systems—maps and nested maps. *GeoInformatica*, 1(4):345-367, 1997.

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

Jonathan Raper and David Livingstone. Let's get real: spatio-temporal identity and geographic entities. *Transactions of the Institute of British Geographers*, 26:237-242, 2001.

Philippe Rigaux and Michel Scholl. Multiscale partitions: Application to spatial and statistical databases. In Max J. Egenhofer and John R. Herring, editors, *Advances in Spatial Databases*, volume 951 of *Lecture Notes in Computer Science*, pages 170-183. Springer Berlin Heidelberg, 1995.

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.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the 6th International Conference on Computer Vision*, pages 59-66, 1998.

Hanan Samet and Markku Tamminen. Bintrees, CSG trees, and time. In Pat Cole, Robert Heilman, and Brian A. Barsky, editors, *SIGGRAPH '85*, volume 19, pages 121-130. ACM, 1985.

Amir Semmo, Matthias Trapp, Jan Eric Kyprianidis, and Jürgen Döllner. Interactive Visualization of Generalized Virtual 3D City Models using Level-of-Abstraction Transitions. *Computer Graphics Forum*, 31(3pt1):885-894, June 2012.

Jonathan Richard Shewchuk. Sweep algorithms for constructing higher-dimensional constrained Delaunay triangulations. In *Proceedings of the 16th Annual Symposium on Computational Geometry*, pages 350-359, 2000.

Jonathan Richard Shewchuk. General-dimensional constrained Delaunay and constrained regular triangulations, I: Combinatorial properties. *Discrete & Computational Geometry*, 39(1003): 580-637, March 2008.

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

A Stadler and T H Kolbe. Spatio-semantic coherence in the integration of 3D city models. In Alfred Stein, editor, *ISPRS Archives. Proceedings of the WG II/7 5th International Symposium Spatial Data Quality 2007 with the theme: Modelling qualities in space and time*, page 8, Enschede, the Netherlands, June 2007.

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

Peter van Oosterom. Variable-scale topological data structures suitable for progressive data transfer: The GAP-face tree and GAP-edge forest. *Cartography and Geographic Information Science*, 32(4):331-346, 2005.

Peter van Oosterom and Jantien Stoter. 5D data modelling: Full integration of 2D/3D space, time and scale dimensions. In *Proceedings of the 6th International Conference GIScience 2010*, pages 311-324. Springer Berlin / Heidelberg, 2010.

Herman Varma, H. Boudreau, and W. Prime. A data structure for spatio-temporal databases. In *Proceedings of the IHO Review*, pages 1-10, 1990.

Remco C. Veltkamp and Michiel Hagedoorn. State of the art in shape matching. In *Principles of Visual Information Re-*

trieval, Advances in Pattern Recognition, pages 87–119. Springer London, 2001.

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.

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

Xiang Zhang, Tinghua Ai, Jantien Stoter, and Xi Zhao. Data matching of building polygons at multiple map scales improved by contextual information and relaxation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 92:147–163, June 2014.

Junqiao Zhao, Zhu Qingg, Zhiqiang Du, Tiantian Feng, and Yeting Zhang. Mathematical morphology-based generalization of complex 3D building models incorporating semantic relationships. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:95–111, March 2012a.

Junqiao Zhao, Qing Zhu, Zhiqiang Du, Tiantian Feng, and Yeting Zhang. Mathematical morphology-based generalization of complex 3d building models incorporating semantic relationships. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:95–111, mar 2012b.

Qing Zhu, Junqiao Zhao, Xiaochun Liu, and Yeting Zhang. Perceptually guided geometrical primitive location method for 3d complex building simplification. In Thomas H. Kolbe, Hao Zhang, and Sisi Zlatanova, editors, *Proceedings of GeoWeb 2009 Academic Track – Cityscapes*, pages 74–79, 2009.