An evaluation and classification of nD topological data structures for the representation of objects in a higher-dimensional GIS

Ken Arroyo Ohori   Hugo Ledoux   Jantien Stoter

One solution to the integration of additional characteristics, e.g. time and scale, into GIS datasets is to model them as extra geometric dimensions perpendicular to the spatial ones, creating a higher-dimensional model. Previous work has been mostly limited to higher-dimensional rasters and hierarchies of trees, which grow exponentially with the dimension. As representations with limited topological relationships quickly become intractable in higher dimensions, a topological vector approach seems most suitable for this purpose, requiring the use of higher-dimensional topological data structures. We therefore present in this paper an evaluation and classification of the possible data structures for an nD GIS, including how they can be implemented to support real-world data aspects, such as holes, disconnected components and attributes, as well as practical issues that affect their feasibility, like the availability of algorithms, libraries and software.

1 Introduction

The representation of points, line segments and polygons in 2D Geographic Information Systems (GIS) can be considered a solved problem. There are a myriad of well-known topological data structures to represent them [Worboys and Duckham, 2004], such as the DCEL [Muller and Preparata, 1978] and the quad-edge [Guibas and Stolfi, 1985], and despite substantial differences between them, for practical purposes they are often considered interchangeable as long as some minimum practical requirements can be met (e.g. support for holes and attributes). Furthermore, many GIS applications use data structures with very little or no topology, such as Simple Features [OGC, 2011], and instead compute topological relationships only when they are needed [ESRI, 2005]. The success of this non-topological modelling approach can be attributed to the fact that visualisation does not require explicit topology, to the relatively small size of most 2D datasets, and to the possibility of taking advantage of strong properties that are intrinsic to the 2D case, such as that there can be at most two polygons incident to any edge in a planar partition.

However, many of these properties do not apply to dimensions higher than two—compare how there can be arbitrarily many polygons incident to an edge in a 3D space partition. Despite this fact, the integration into a GIS of additional characteristics such as a third spatial dimension, time, and scale has been so far achieved mainly by ad hoc adaptations to 2D data structures rather than by building new, higher-dimensional ones, effectively limiting the capabilities of GIS software. For instance, as first shown in Raper [1989], 3D GIS often mimic the third dimension by using a so-called 2.5D structure, essentially treating the third dimension as an attribute and limiting the geometries that can be represented; or represent 3D objects individually and only implicitly through their 2D boundary, using a 2D data structure with no 3D topological relationships[1]. This im-

plies that many operations are only possible using expensive searches involving many more volumes than needed. Spatiotemporal GIS keep multiple representations of 2D [Armstrong, 1988] or 3D [Hamre et al., 1997] structures, or a list of changes per object [Worboys, 1992a; Peuquet, 1994], limiting combined spatio-temporal analysis of such objects; and multi-scale datasets generally consist of independent datasets for each scale with some common IDs to link them, potentially causing inconsistencies among them.

An alternative to this use of ad hoc adaptations of 2D structures is the representation of certain parametrisable characteristics as additional geometric dimensions, orthogonal to the spatial ones, such that real-world point to volume (0D–3D) entities are modelled as higher-dimensional objects embedded in higher-dimensional space, which are consequently stored using *higher-dimensional data structures*. While there is no theoretical limit to the number of dimensions that can be used, the increasing space requirements of models with more dimensions limit the practical number of dimensions to 6–8.

The use of higher-dimensional models is well grounded in long-standing mathematical theories and opens the door for practical applications. 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, Arroyo Ohori et al. [2013a] show how 4D topological relationships between 4D objects provide insights that 3D topological relationships cannot, McKenzie et al. [2001] contends that weather and groundwater phe-

---

[1] i.e. topological relationships involving 3D primitives, such as adjacencies between two volumes or incidences between a volume and a vertex/edge/face

nomena cannot be adequately studied in less than four dimensions, 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.

So far, most research related to higher-dimensional models and structures in GIS has been limited to a combination of regular subdivisions along every axis (nD grids) and hierarchical subdivisions using trees [Mason et al., 1994; Varma et al., 1990; Bernard et al., 1998; O'Conaill et al., 1992], something that has only recently been implemented in GIS software [Mielke, 2014]. This takes advantage of the fact that most data structures used for 2D regular and semi-regular tessellations [Boots, 1999] extend trivially to higher-dimensions, even if most research and software are limited to 4D and using time as a fourth dimension. However, these representations suffer from the same problems as 2D/3D grids and hierarchical trees in GIS: they are incapable of representing precise boundaries and are cumbersome for the representation of objects [Fisher, 1997], especially when they are of mixed dimensions, while their space requirements increase exponentially with the dimension.

Vector representations of nD objects avoid these problems: they are generally more compact, can describe boundaries more precisely and can represent objects with their attributes directly. This makes them particularly interesting for higher-dimensional GIS, even if they have been hardly been explored as the data structures commonly used in 2D vector GIS do not extend naturally to higher dimensions. However, there are various higher-dimensional geometric and topological structures that have been developed in other fields and can be applied for this purpose, even if these are significantly different from the 2D ones used in GIS, are complex to implement and use efficiently, and often need special adaptations in order to support some aspects of real-world data (such as holes or non-manifold geometries). They are therefore almost never used in practice.

We believe that this can be overcome with a better understanding of the possibilities and challenges to apply these higher-dimensional structures for the representation of real-world objects. In this paper we therefore summarise the results of our investigation. For this, we provide a fully dimension-independent description of these data structures in Sections 3–5, and accompany each with 2D/3D examples to link it to analogous cases in 2D/3D GIS. While good surveys on higher-dimensional data structures exist, most notably Lienhardt [1991] and Čomić and de Floriani [2012], these cover fewer data structures, focus on theoretical aspects in a geometric modelling context, and do not cover the more practical aspects that also affect the implementation of these structures, such as the availability of algorithms, libraries and software. This paper is an extension of Arroyo Ohori et al. [2013b], in which we discussed how to model higher-dimensional objects using real-world data in one particular data structure, generalised maps [Lienhardt, 1994].

As explained in detail in Sec. 2, we have identified three main families of feasible spatial data structures for the representation of nD objects. Each of these groups discretises space in a similar way, uses similarly shaped primitives and shares most of their properties, including the techniques that can be applied to them in order to support characteristics like holes and attributes. These are therefore analysed separately: simplicial complexes in Sec. 3, cell complexes in Sec. 4, and ordered topological models in Sec. 5. We finish with a discussion on the implications of using these data structures for the representation of nD objects in Sec. 6.

A note on terminology

This paper is meant mainly for a GIS audience who is interested in the representation of real-world objects in more than 3D. We have therefore chosen to use 2D/3D GIS terminology (or terms in their GIS sense)

3

whenever possible (e.g. a surface being a 2D object even when it is non-manifold), and mathematical terms only when GIS terminology is unclear or ambiguous (e.g. a face can be of any dimension).

2 Higher-dimensional data structures and their classification

Spatial data structures consist of two aspects: 1. a combinatorial part, which consists of a set of primitives and some topological relationships between them, and 2. an embedding that links these primitives to their geometry and attributes. This distinction is commonly used in practice to separate the problems that are the domain of geometric modelling from those of computational geometry [Lienhardt, 1994; Mäntylä, 1988], but it is also useful to describe what differentiates higher-dimensional structures from 2D/3D ones.

This is clearer when considering the data structures that contain only 1D topological relationships, which are commonly used in GIS. For instance, in the Simple Features Specification [OGC, 2011], independent 2D primitives are defined as sequences of points (with coordinates) that are connected using implicit line segments. 3D primitives are then represented as sets of such 2D elements, which are otherwise unlinked. Since the implicit line segments in a 2D primitive are known to be adjacent but nothing is known about the relationships between the 2D primitives, such data structures can be considered to store up to 1D topological relationships only (adjacencies between edges and the incidences between edges and vertices). While such data structures can be embedded into any dimension (by assigning coordinates in nD space to every point), they become exponentially more inefficient as the dimension increases: lower dimensional primitives need to be encoded multiple times, recursively once for each time they appear in a higher-dimensional primitive. This means

that even simple geometric and topological queries involve searching many objects [Hazelton, 1998]. For instance, a Simple Features (-like) representation of a square does not repeat most vertices, while one of a cube repeats every vertex at least three times, one of a tesseract (their four-dimensional analogue) repeats every vertex at least 12 times and one of a 5-cube at least 60 times[2].

Accordingly, in a similar vein to the definition of the dimensionality of a GIS in Hazelton et al. [1990], we argue that the defining characteristic of a higher-dimensional spatial data structure is not the ability to be embedded into higher-dimensional space, but being able to explicitly store topological relationships in a dimension-independent manner. For this, we assume that the objects being represented can be made to functionally form a space partition using the standard topological modelling approach used in GIS—all objects are overlaid, their intersecting boundaries are computed and every region in the model is considered to belong to a set of objects [Rossignac and O'Connor, 1989]. Doing so makes it possible to store a traversable structure containing the objects, in which the boundaries of the sets are explicitly represented, and to easily query the topological relationships that exist between the objects [Egenhofer and Franzosa, 1991; Worboys, 1992b; Güting et al., 2000] by checking which objects are in a set.

The most promising structures for higher-dimensional GIS are therefore those that starting from a space subdivision, are capable of storing a well-defined and relatively broad class of objects of arbitrary dimension, embedded into Euclidean space of the same or higher dimension, attributes attached to such objects, and enough topological relationships between them to allow for quick traversal of the structure and the basic operations, such as testing if two given objects are identical, adjacent or overlapping.

In most GIS literature, the terms 'data

---

[2]The number of times a vertex is repeated in Simple Features can be up to doubled since representing a closed polygon involves repeating its first vertex at the end.

model' and 'data structure' are often used interchangeably. However, as Frank [1992] argues, it is useful to separate the data model, i.e. the particular discretisation of the world into abstract primitives, from the data structure used, i.e. their concrete representation in a computer. A data model broadly defines the shape of the primitives and relationships between them that can exist, while a data structure *explicitly* stores some of these relations and omits others (which can be computed with an added computational cost).

This distinction is usually blurred in 2D structures since these usually have definitions based on explicitly stipulated rules for the representation of 0D–2D primitives, which might be different for each dimension, e.g. separate node, arc, and area structures in the winged-edge [Baumgart, 1975] and DCEL [Muller and Preparata, 1978] data structures. This effectively skips the definition of a shared underlying model, with the consequence that these structures do not have a clear generalisation to higher dimensions. By contrast, higher-dimensional models are necessarily defined using dimension-independent primitives (at least for the higher dimensions). They are therefore able to represent a mathematically well-defined class of objects, even if this class of objects is not fully supported by a particular implementation in a higher-dimensional data structure, and can therefore be objectively classified.

A classification of higher-dimensional data structures into a number of models is not only of theoretical interest, but it is very useful for their analysis. Different data structures using a similar model tend to have similar properties in terms of space complexity since they contain a number of primitives of the same order[3]. Operations performed on them also tend to have the same computational complexity since they can have similar topological relationships, and when they do not, their cost is bounded by the conversion cost from one to the other, which is often easy and can

be done primitive by primitive. Since algorithms operate on primitives, which are defined by a data model, this analysis also serves to know which fundamental operations can be applied on a data structure. For instance, Euler operations are defined in terms of cell complexes [Mäntylä, 1988] and stellar operators in terms of simplicial complexes [Velho, 2003]. Therefore, in general terms, any data structure that represents cell complexes can implement Euler operations, and the same for stellar operators and simplicial complexes.

As argued by McKenzie et al. [2001], the ideal data model to use depends on the application. An ideal data model is expected to be both powerful and able to represent a large class of objects, but unfortunately these properties tend to conflict. Powerful structures are built on small and simple primitives of a fixed form with a fixed (or at least bounded) number of links to other related primitives, such as axis-parallel boxes or simplices, enabling them to have a simple but powerful algebra to manipulate them and to navigate between their primitives efficiently, e.g. the quad-edge in 2D [Guibas and Stolfi, 1985] or the facet-edge in 3D [Dobkin and Laszlo, 1987]. However, partitioning complex objects into these simple primitives is non-trivial and can result in a large number of primitives, requiring significant preprocessing, making them space-intensive and limiting the types of objects that might be stored. In this sense, different data models make different choices and necessarily involve a trade-off.

Based on the primitives used and their characteristics, we have classified all data structures that are known to us and are capable of representing vector objects in nD into three categories, which are summarised in Table 1 and explained below:

- **Geometric simplicial complexes (Sec. 3)** They decompose objects directly into simplices (i.e. points, line segments, triangles, tetrahedra, etc.) of dimension up to n, all of which have a direct geometric realisation (i.e. a simplex in the model represents a simplex in real-world space), and store them with a data

---

[3]The number of primitives might not be the same since a discretised element in the model might be implemented as one or multiple separate structures.

Table 1: Summary of the data structures covered in this paper

	Model	Data structure	Description
	Geometric simplicial complexes	Simplex-based	Geometrically split objects into simplices, then store simplices + the adjacency relationships between them
	Cell complexes	Incidence graphs	Store the cells of every dimension as primitives + incidence relationships between them
		Nef polyhedra	Store the local pyramid (projection of the space) around each vertex to reduce the dimension by one
		Manifold data structures	Limit the number of incidence relationships to a fixed number (e.g. half-$(i-1)$-cell data structure)
	Ordered topological models	Generalised maps / cell-tuple	Purely combinatorial decomposition of a cell complex + storage with a simplex-based data structure
		Combinatorial maps	The above but without splitting 1-cells and defining an orientation per connected component
		Chains of maps	Generalised maps + an incidence graph for non-manifold situations

structure that uses simplices as primitives. They are more space intensive and harder to build than other models but provide a stronger algebra to manipulate them.

· **Cell complexes (Sec. 4)** They decompose n-dimensional objects into parts that are homeomorphic to n-balls, and store them with a data structure that uses cells as primitives. They are compact and easy to build, but only have a limited algebra to manipulate them.

· **Ordered topological models (Sec. 5)** First create a cell complex, then decompose the cells in it into abstract simplices *without* a direct geometric realisation (i.e. a simplex in the model does not have a specific meaning in the real-world space), and store them with a data structure with simplices as primitives. These therefore combine the strengths and weaknesses of the previous two, with other surveys correctly classifying them both as structures for cell complexes [Čomić and de Floriani, 2012] and for simplicial complexes [de Floriani and Hui, 2005]. However, they differ in practical terms from the structures described above since they allow a different set of operations and are therefore covered separately.

These are not all the models with which it is possible to represent vector nD data. Models containing only points in nD space [Pasko and Adzhiev, 2001] are widely used in the context of remote sensing and data mining, and are particularly relevant in the representation of fields when used together with an interpolation function [Miller, 1997]. Constructive representations of geometries, such as Constructive Solid Geometry (CSG) trees [Samet and Tamminen, 1985; Paoluzzi et al., 2004], and objects described as sequences of operations [Čomić et al., 2014] are also possible and useful in certain contexts, such as production processes and the computation of homologies [Damiand et al., 2008]. However, none of these are capable of storing objects and

their attributes *directly*, especially those of lower dimensionality, and therefore have to be 'evaluated' into another, more explicit model in order for them to be visualised [Mäntylä, 1988] or to perform the type of computations expected in GIS, such as geometric intersection operations. They are therefore less interesting for a higher-dimensional GIS and not considered within this paper.

3 Geometric simplicial complexes

An i-dimensional simplex (i-simplex) is a combinatorial primitive made from a set of $i + 1$ vertices, and an j-simplex, $j \leq i$, made from a subset of these is an j-dimensional face (j-face) of it. Intuitively, an n-dimensional geometric simplicial complex, also known as a Euclidean simplicial complex, is a subdivision of n-dimensional space into a structure of closed n-simplices directly embedded in space by attaching a tuple of coordinates to each vertex, in a manner that their interiors are non-overlapping geometrically (pairwise disjoint). As shown in Fig. 1, in an n-dimensional simplicial complex, an i-dimensional object, $i \leq n$, is represented as a set of i-simplices.

More formally, a simplicial complex can be defined as a collection of simplices such that:

- Every face of a simplex is also in the simplicial complex
- The intersection of any two simplices is either empty or is a common face of both of them

Representation

An i-simplex is easily described by an $(i + 1)$-tuple of $i + 1$ affinely independent[4] ver-

---

[4]That is, that none of them are in the affine space described by the others. In practical terms: two points with a different coordinate in 1D, three non-collinear points in 2D, four non-coplanar points in 3D, etc.

tices $(v_0, v_1, \dots, v_i)$. Since they are affinely independent, its geometric realisation is the convex hull of its vertices. A vertex in an n-dimensional simplicial complex can be described by an n-tuple of coordinates $(x_0, x_1, \dots, x_{n-1})$, representing its embedding into n-dimensional Euclidean space. Since we are considering a subdivision of n-dimensional space, every simplex in the simplicial complex is a face of at least one n-simplex.

Algebra

Simplicial complexes have a simple but powerful structure and are easy to navigate. Two i-simplices are adjacent if they have a common face, and an i-simplex and a j-simplex, $i \neq j$, are incident if either is a face of the other. In an n-dimensional simplicial complex, each n-simplex is adjacent to at most $n + 1$ other n-simplices and has exactly $n + 1$ $(n - 1)$-faces on its boundary. To each n-simplex it is therefore possible to apply $n + 1$ different boundary and neighbour operators, where the n-th boundary operator returns the $(n - 1)$-face of the n-simplex on the opposite side of the n-th vertex—obtained by removing the n-th element of the tuple of vertices—, and the n-th neighbour operator returns the adjacent n-simplex on the opposite side of the n-th vertex, if any, which is also incident to the face given by the n-th boundary operator.

Construction

The biggest challenge of using a geometric simplicial complex for the representation of nD objects is the difficulty of triangulating the possibly complex nD objects being modelled, i.e. algorithmically dividing them into simplices so that the union of the simplices representing an object corresponds exactly to its original geometry. While triangulations of sets of points in nD have been studied extensively and there is robust software to compute them, e.g. Quickhull [Barber et al., 1996], such triangulations are generally not applicable to nD objects since a triangulation of the vertices of an object usually results in simplices that
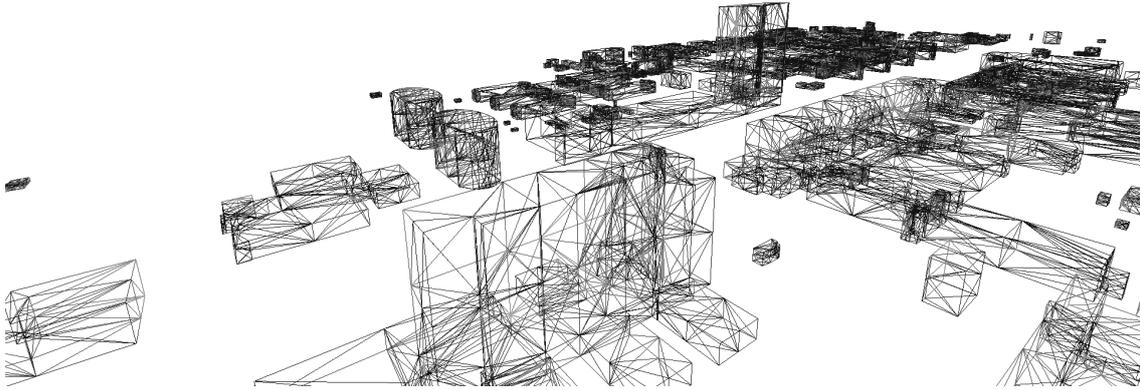
Figure 1: A group of buildings is represented as a geometric simplicial complex. Each 3D building is thus split into a set of non-overlapping 3-simplices (tetrahedra), with their 2D walls and roofs represented as sets of 2-simplices (triangles).

do not conform to the object's boundary, i.e. that are partially inside and partially outside the object. However, it is possible to make sure that the boundaries are part of the triangulation through the use of a constrained or conforming triangulation.

Both of these force the triangulation to include certain simplices as part of the triangulation in the form of *constraints*, which in this case would be the boundaries of the objects, with the difference that the constraints in constrained triangulations are directly part of the structure, while in conforming triangulations they are allowed to be split into smaller simplices. While the properties of nD constrained triangulations have been described [Shewchuk, 2008] and algorithms to construct them in some cases have been developed [Shewchuk, 2000], their construction in the general case is very difficult in practice, having to deal with robustness issues and possibly requiring the addition of a large number of additional vertices (Steiner points) and subsequently additional simplices, as not every object of dimension 3 or higher is triangulable without additional vertices [Ruppert and Seidel, 1992]. Unfortunately, while there are robust and reliable constrained and conforming triangulation libraries existing in 2D and 3D, such as Triangle [Shewchuk, 1996], CGAL [Boissonnat et al., 2002], and Tetgen [Si and Gärtner, 2005], we are not aware of any software or algorithm capable of performing a constrained or conforming triangulation in more than 3D.

Holes, attributes and disconnected components

Assuming that the objects have been triangulated successfully, managing holes and disconnected components in a simplicial complex becomes trivial since these become integrated into the combinatorial structure of the triangulation [Ledoux et al., 2014]. Generally, a triangulation algorithm works initially on the basis of a set of points and ensures that the entire convex hull of all of the points is triangulated, something that can be done using the concept of a 'big triangle' or faraway point [Liu and Snoeyink, 2008], and therefore connects formerly disconnected components or components that were not connected by highest-dimensional simplices. Similarly, as the interiors of polytopes with holes are triangulated, additional simplices connect the void regions representing holes with the rest of the structure. In this manner, 2D holes can be represented as sets of triangles marked as void, 3D holes as sets of tetrahedra, and similarly so for higher (and lower) dimensions. This ensures that it is always possible to navigate between all the simplices in the complex.

Operations

Another advantage of simplicial complexes is that many operations on them are efficient and easy. For instance, a simplicial complex can be traversed efficiently without any external data structure. Starting from any simplex, it is possible to 'walk' the simplicial complex in the desired direction in order to find e.g. in which simplex a point would be located [Devillers, 2002], with each step taking only constant time. Comparing objects composed of sets of simplices is also trivial, assuming that they have been triangulated in a deterministic way[5], since they can be compared one by one (in order) after finding a common simplex, which can be done by walking the respective simplicial complexes. Checking if two objects are adjacent or intersect geometrically are examples of other operations that can be done simplex by simplex.

Simplex-based data structure

Since the number of vertices, $(n-1)$-faces and adjacent simplices of an n-simplex in an n-dimensional simplicial complex is fixed and known, it is easily represented in a computer using simple arrays, rather than linked lists or other variable-length data structures. Most commonly, this is done through the use of an n-simplex based data structure, shown in Fig. 2 for 2D, where n-simplices are the main primitives. It is sometimes called as the adjacency or incidence model, depending on the dimensions of the simplices that are represented in the structure—only the highest-dimensional ones in the adjacency model, but possibly all in the incidence model. An n-dimensional simplicial complex is represented as a collection of primitive n-simplices, each containing $n + 1$ links to its adjacent n-simplices. For the models described above, it also includes information containing:

- **Adjacency model** For every n-simplex, $n + 1$ links to vertex primitives containing the coordinates and attributes of its $n + 1$ vertices, or alternatively this information stored directly in the n-simplex. The simplices of dimensions 1 to $(n - 1)$ and the incidence relationships between them, are therefore not represented explicitly.

- **Incidence model** For every i-simplex, $\forall 0 < i \leq n$, $i + 1$ links to primitives containing the geometry and attributes for its $(i - 1)$-faces, or alternatively this information stored directly in the i-simplex. All simplices of every dimension, together with all their incidence relationships, can therefore be represented explicitly, or as part of a simplex of a higher-dimension.

The adjacency model thus forms an $(n + 1)$-regular graph with the n-simplices as nodes, i.e. a graph where every vertex has degree $n + 1$. The incidence model is a graph with a maximum degree $n + 1$, where the degree of a node representing an i-simplex is $i + 1$. Note that the adjacency model is therefore more compact, but only allows for the explicit representation of n-dimensional objects, while the incidence model is more verbose but supports the representation of objects of any dimension.

Other data structures

Other possibilities involve describing simplices more compactly using the strong algebraic properties of simplicial complexes, although this has been hardly explored in more than 3D. For instance, in 2D and 3D it is possible to: store the geometry of only some of the vertices of every simplex [Blandford et al., 2005] (the rest being stored in other incident simplices), use a collection of adjacent simplices (e.g. the star of a vertex) as a primitive instead of a single one, use progressive representations that approximate the simplicial complex incrementally [Popović and Hoppe, 1997], or compress it as a sequence of simple operations from which the original structure can be deduced [Rossignac and Szymczak, 1999].

---

[5] Note that this is not trivial. Also, if the triangulations are different, such a comparison would necessarily require geometric operations.
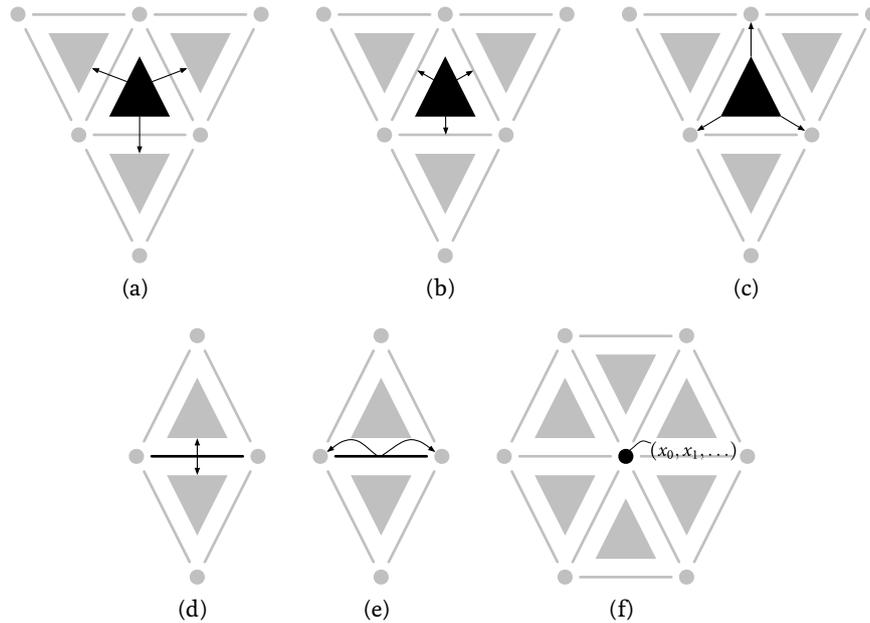
Figure 2: Some of the possible relationships between primitives and storage of geometry in a 2-simplex (triangle) based data structure. (a) Adjacency relationships of a 2-simplex. (b) Boundary relationships of a 2-simplex. (c) Vertices of a 2-simplex. (d) Star (co-boundary) relationships of a 1-simplex. (e) Boundary relationships of a 1-simplex. (f) Embedding of a 0-simplex. In addition to these, it is also possible to store attributes for the 0-, 1-, and 2-simplices in embedding structures. The adjacency model uses at least (a), (c) and (f), while the incidence model uses at least (b), (e) and (f), plus usually (a) to navigate the structure. Many other combinations are possible.

Moreover, since a simplex is also a cell, it is also possible to use any data structure meant for more general cell complexes, such as those described in Secs. 4 and 5. This is commonly used in practice during the process of triangulating a cell complex (since at that point the structure is not a simplicial complex and cannot by definition be stored in a data structure for simplicial complexes only), but it is a very inefficient approach when it has already been triangulated.

Applications

Simplicial complexes in more than 3D are commonly described in theory but rarely put in practice. Within GIS, Paoluzzi et al. [1993] described the winged scheme in order to describe arbitrary polytopes as simplicial complexes, and Karimipour et al. [2010] uses lists of vertices to store simplices of any dimension in order to do spatial analysis using Delaunay triangulations as an example.

4 Cell complexes

An n-dimensional cell complex, related to the concept of a CW complex in topology, is a subdivision of n-dimensional space into non-overlapping (pairwise disjoint) cells so that for all $0 \leq i \leq n$ an i-dimensional cell (i-cell) is an object homeomorphic to an open i-ball (i.e. point, open arc, open disk, etc.). 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. Similarly to simplicial complexes, two i-cells are adjacent if they have a common face, and an i-cell and a j-cell, $i \neq j$, are incident if either is a face of the other.

More formally, a cell complex can be defined inductively as in Hatcher [2002]. An n-dimensional cell complex is built by starting from a set of isolated vertices, and $\forall 0 < i \leq n$ an i-cell is built by attaching itself to the $(i-1)$-faces (facets) on its boundary, these facets having been previously added to the complex. That is, an edge is built by linking the vertices on its boundary, a surface by linking the edges on its boundary, a volume by linking the surfaces on its boundary, and so on. Like in a simplicial complex, a facet of a n-cell in an n-dimensional cell complex lies between it and an adjacent n-cell, unless it is on the boundary of the complex.

Algebra

Cell complexes differ from simplicial complexes in two related aspects: 1. there is a variable number of facets on the boundary of a cell, each of which has also a different number of facets, continuing recursively down to the vertex level; 2. there is not a simple geometric interpretation of a cell that does not require looking recursively at its facets. This leads cell complexes to have a much more complex representation and algebra to manipulate them compared to simplicial complexes. For instance, simplicial complexes can be easily constructed by adding and removing simplices expressed by sets of vertices, or by splitting and merging existing ones, but cell complexes require much more complex operations in order to add or remove cells as these have to be described based on their facets [Arroyo Ohori et al., 2014]. Traversing a cell complex (e.g. to find in which cell a point lies or to see which cells intersect a given geometry) is also much more difficult [de Berg et al., 1997].

Construction and attributes

An i-dimensional object with attributes can generally be represented directly as an i-cell with attached attributes in the cell complex—requiring minimal preprocessing as long as it is homeomorphic to an i-ball or can be modified so that it is functionally so using simple 'tricks' or combinatorial operations only. Also, the lower number of cells in a cell complex compared to the simplices in a simplicial complex representing the same set of objects means that they can store objects more efficiently, as long as an adequate data structure is used.

Representation

Cell complexes are usually modelled based on techniques that represent them based on their $(n-1)$-dimensional boundary (i.e. boundary representation), unlike simplicial complexes in which objects are usually modelled using mainly primitives of the same dimension (i.e. object representation, but better known by its 3D name as solid or volume representation). This is a subtle distinction, but it implies that cells are described (and stored) in a more implicit manner based on a recursive definition, unlike simplices which can be stored explicitly as a tuple of vertices. Since this can slow down many operations, access to cells can be improved by adding indexing structures storing links to all the cells of a given dimension and/or to all the cells in a given spatial extent.

Incidence graph structures

The most common data structures for arbitrary cell complexes are incidence graphs and other related structures [Rossignac and O'Connor, 1989; Masuda, 1993; Sohanpanah, 1989], which store all the cells (of every dimension) in a complex as individual embedding primitives containing geometric information and attributes. Just as with simplicial complexes, if only linear geometries are required, the only geometric information needed is the coordinates of the points for the 0-cells. However, since a set of vertices is not enough to describe a cell, all the embedding structures of every dimension need to exist (albeit they can be as simple as a unique ID or memory address per cell). As shown in Fig. 3, these are connected

through a combinatorial structure containing the incidence relationships between the cells, such as an i-cell having an $(i-1)$-cell on its boundary, vice versa (known as the co-boundary or star), or both. Since the number of these incidences is generally not fixed or bounded, unlike in a simplicial complex, they are more difficult to integrate into the cell primitives. One solution involves using variable length data structures (e.g. a linked list of the facets of a cell). Another would be to store instead combinatorial primitives consisting of pairs of an i-cell and an $(i-1)$-cell—an approach well-suited to a database implementation.

While incidence graphs are efficient in terms of space, they are difficult to navigate and manipulate because of the limited topological information contained in them. This greatly complicates many relatively simple queries that are trivial in simplicial complexes, such as: obtaining the order of a sequence of 0- and 1-cells around a 2-cell; or checking if two n-cells have the same geometry (since their facets, each represented implicitly by its own facets, might be stored in any order).

Half-space structures and Nef polyhedra

Another approach involves representing cells as unions of intersections of half-spaces, something that can be achieved by performing (alternate) decompositions into convex parts [Bulbul and Frank, 2009; Lien and Amato, 2006]. Most data structures that implement this concept therefore store cells implicitly as trees of operations on half-spaces, as shown in Fig. 4, each of which is stored as a tuple containing the coefficients of a hyperplane equation plus an up/down direction [Naylor, 1990]; or as a collection of hyperplanes that represent all the facets of all the cells, and each cell is represented as a set of tuples of Boolean values, where each value states whether the cell is on the up or down side of a specific hyperplane [Tammik, 2007]. While this is rather cumbersome and does not scale well to higher-dimensions and complex geometries, Nef polyhedra [Bieri and

Nef, 1988] follow a similar approach but extend it with the concept of a local pyramid, which stores the neighbourhood information for every vertex as shown in Fig. 5, containing all its incident cells of every dimension, by projecting them on an infinitesimally small hypersphere around the vertex and storing this information in using hyperspherical geometries on an $(n-1)$-dimensional data structure, essentially reducing the dimensionality of the cell complex by one. A set of polygons can thus be reduced to a set of circular intervals, a set of polyhedra to a set of faces on a sphere, and so on to higher dimensions. This means that the global arrangement of hyperplanes is only stored locally at the vertex level, which scales much better to complex geometries and allows the storage of attributes (in the vertices, circular intervals, spherical faces, etc.). This technique is very useful already in 3D, since it allows one to use any of the many 2D data structures available to represent volumes. While in theory this method can be repeated in decreasing dimension in order to have a complete description of the cell complex, the practical benefits of doing do decrease as the dimension increases and the navigation becomes more cumbersome. Moreover, it is very complex to implement this in a fully dimension-independent manner, requiring a hyperspherical projective kernel capable of computing point-set intersections. In fact, all existing implementations that are known to us are limited to 3D, such as the one in CGAL [Hachenberger, 2006; Granados et al., 2003].

Manifold data structures

One more possibility involves limiting the representation to cell complexes with stronger algebraic properties, such as those with a manifold domain. Manifolds are objects that locally resemble the Euclidean space of a certain dimension, even if globally they do not. More precisely, an n-dimensional manifold (or simply n-manifold) M is a topological space for which every point $x \in M$ has a neighbourhood homeomorphic to $\mathbb{R}^n$. Most impor-
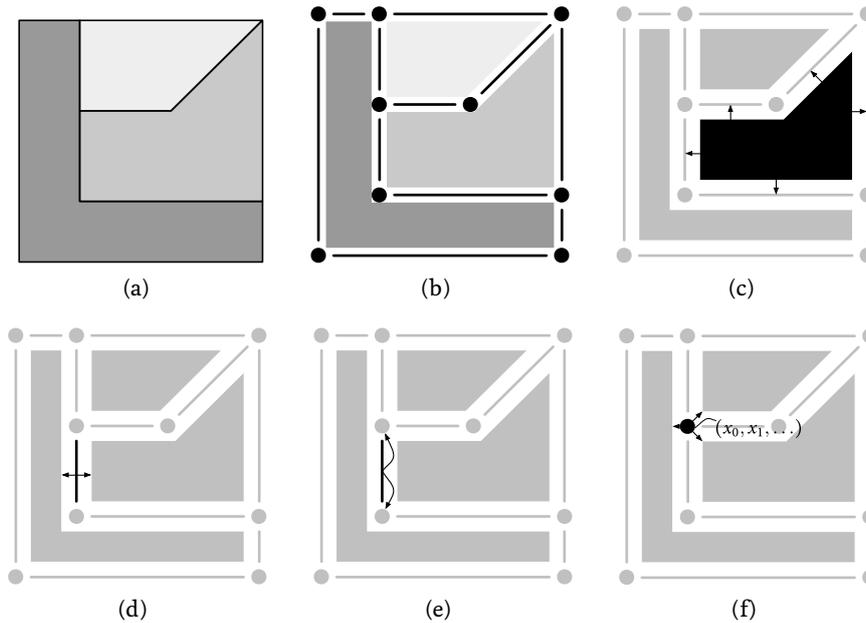
Figure 3: The cells in a cell complex, the most common relationships that are be stored in an incidence graph and similar data structures up to 2D, and the embedding of the 0-cells so as to support a geometry. (a) Three adjacent polygons. (b) The cells in the cell complex. (c) Boundary relationships of a 2-cell. (c) Boundary relationships of a 2-cell. (d) Co-boundary relationships of a 1-cell. (e) Boundary relationships of a 1-cell. (f) Co-boundary relationships and embedding of a 0-cell. Note that (c), (d) and (e) correspond to an i-cell that is linked to the $(i-1)$-cells on their boundary or vice versa.

tantly for a spatial data model, manifolds have two very useful properties:

- An n-manifold with boundary has the property that its boundary is a $(n-1)$-manifold, ensuring that an n-dimensional object can be represented recursively by its $(n-1)$-dimensional boundary using a manifold data structure.

- An $(i-1)$-cell is incident to at most two i-cells within an $(i+1)$-cell[6], limiting the number of links required between primitives in a boundary representation based model of the complex.

Some data structures for cell complexes in 2D and 3D are thus restricted to manifolds, or split objects into manifold sections [Pesco et al., 2004; Lopes and Tavares,

1997], which allows such data structures to be more efficient as they need to handle simpler configurations [Aguila and Ramírez, 2003]. For instance, if we restrict the representation to a n-dimensional cell complex with manifold domains for each cell, an $(n-2)$-cell is incident to at most two $(n-1)$-cells, $(n-2)$-cells can be used as primitives or pointed to while being able to navigate around them. This results in data structures based on half-edges in 2D, such as the DCEL [Muller and Preparata, 1978], and in 3D, such as the CIEL [Lévy et al., 2001], but the concept generalises to any dimension. Various 'half-$(n-2)$-cell' data structures for manifold n-dimensional cell complexes are therefore possible, which would consist of linked pairs of $(n-2)$-half-cell primitives. However, note that this is independent from the problem of the representation of the individual $(n-1)$-cells, which is solved by directed edges in the DCEL and a loop of directed edges in the CIEL.

---

[6]This is the definition of a quasi-manifold, a combinatorial interpretation of the topological concept of a manifold. Quasi-manifolds are always manifolds up to 2D, but not necessarily in higher dimensions.
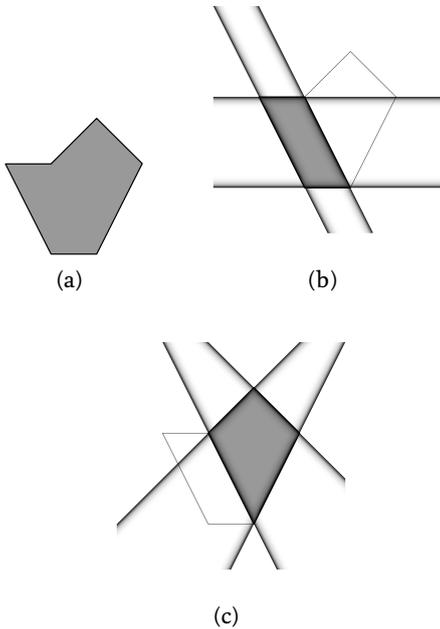
Figure 4: (a) A polygon is stored as a union of convex polygons. (b) & (c) Each convex polygon is represented as a set of intersected half-planes.
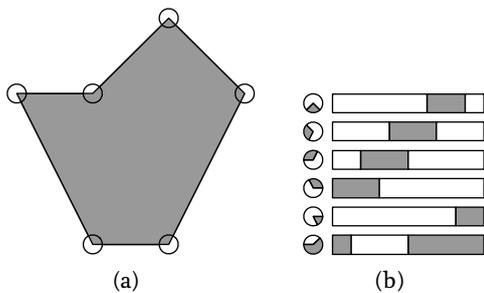
Figure 5: The same polygon is stored based on the local pyramids at the vertices using Nef polyhedra. (a) The local pyramids shown as circles. (b) The views from the local pyramids are projected to the surface of the circles and parametrised based on their angle, reducing the dimensionality of the 2D polygon to 1D. A local pyramid thus becomes a set of adjacent intervals.

While it is generally possible to use manifold data structures to represent non-manifolds using various tricks, like those shown in Fig. 6, such as duplicate elements in a combinatorial structure linked to a single embedding, there is often an ambiguity in the representation (e.g. due to duplicate cells) or a loss of navigability around the 'non-manifold parts' of a model.

Holes and disconnected components

One major difficulty with data structures based on cell complexes is dealing with holes and disconnected objects, both of which would ordinarily result in a disconnected graph using any of the data structures as described above. Since n-cells in a cell complex are assumed to be homeomorphic to open n-balls, holes violate the mathematical definition of a cell complex. However, an n-dimensional hole that is homeomorphic to an n-ball in an n-dimensional cell can be generally handled in an ad hoc manner using any of the following techniques:

· **Splitting cells** The cell containing a hole is subdivided into multiple cells, all of which are adjacent to it. This solution is conceptually simple and is effectively equivalent to what is done with simplicial complexes. This also best respects the definition of a cell, since the subdivided cells can be homeomorphic to balls and the hole is a cell like any other but representing an empty region. However, finding a way to split a cell accordingly requires geometric operations and can be difficult in practice, especially in dimensions 3 and higher.

· **Bridge cells** A hole is connected to the rest of the combinatorial structure using a 'bridge' cell of lower dimensionality than the hole. This bridge cell usually takes the form of an edge connecting the hole with the cell where it is contained. This solution is simpler to implement, usually requiring only the finding of an appropriate bridge (e.g. an edge that lies entirely in the interior of the cell). The cell containing the
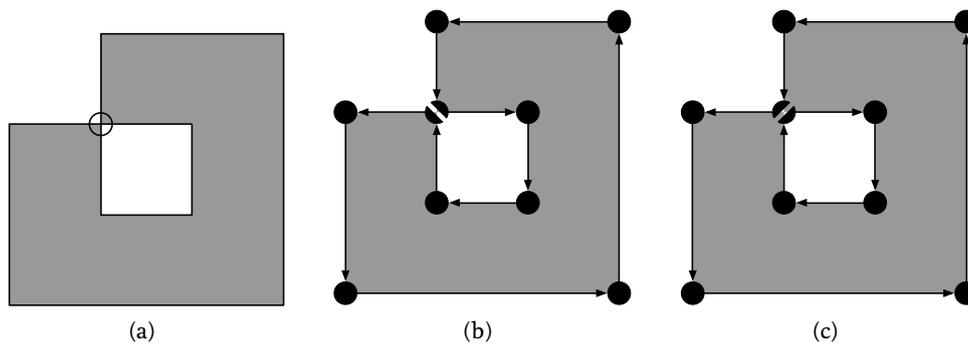
Figure 6: (a) A 2-cell is non 2-manifold as the space around the vertex highlighted in a circle is not homeomorphic to the plane. (b) & (c) It can still be represented using a loop of oriented (half-) edges by having a duplicate vertex at that location (shown as two half balls), but there are two ways in which it can be done. Note that (c) results in a disconnected graph.

hole is no longer homeomorphic to a ball, but most computations work with few or no changes [Bryant and Singerman, 1985]. However, mainly topological queries might need the addition of special cases to handle it (e.g. not displaying such an edge when visualising the cell or accounting for the fact the bridge will have the same cell on all sides).

· **Holes as attributes** Holes can be stored as fully independent cells, linked as special attributes of the objects inside of which they are, or vice versa, something that can be extended so as to have full hierarchies of objects in a tree [Worboys, 2012]. This solution might be the simplest to implement in terms of storage, but it goes against the mathematical definition of a cell complex and will break or require a complex special treatment for topological computations. It is also very simple to create invalid holes in this manner, such as holes lying (partially) outside the cell that is supposed to contain them.

However, it is worth noticing that complex holes that are not homeomorphic to n-balls can exist, e.g. tori, and finding a good subdivision that can be used to deal with such a hole might be very complex. The latter two methods can be applied for disconnected components as well. Two disconnected components can be connected by a

bridge cell, or all components can be considered as holes of the (empty) universe cell.

Applications

Cell complexes in more than 3D have been rarely described or implemented. Hazelton et al. [1990] described how incidence graphs can be put into a relational database schema in order to store 4D polytopes, on which 4D topological relationships can be queried [Hazelton et al., 1992]. However, to the best of our knowledge, this has never been put into practice in a working system.

5 Ordered topological models

Ordered topological models are a representation method that combines characteristics of both simplicial complexes and cell complexes. They work on the basis of a cell complex, but subdivide each cell into *abstract* simplices using a purely combinatorial operation, a concept derived from what is known as an abstract simplicial complex in algebraic topology.

The exact subdivision that is used depends on the data structure, but as shown in Fig. 7, it is similar to a barycentric triangulation

of a convex polytope, consisting of abstract vertices representing cells of different dimensions (i.e. not only the 0-cells), and simplices that connect these vertices in a specific manner. The vertices representing the cells of dimensions higher than 0 do not have a defined location in space (although they are usually depicted lying somewhere in the cells they are supposed to represent), but the simplices nevertheless form a combinatorially correct subdivision of the space. For instance, adjacency and incidence do have meaning and can be defined just as in a geometric simplicial complex. Considering that a j-face of an i-simplex, $j \leq i$, is a j-simplex made from a subset of the vertices of the i-simplex, two i-simplices are adjacent if they have a common face, and an i-simplex and a j-simplex, $i \neq j$, are incident if either is a face of the other.

Representation

In an ordered topological model, objects are represented by sets of simplices, similarly to a simplicial complex. However, unlike in a geometric simplicial or cell complex, where an i-dimensional object is represented as a set of i-simplices or cells, as shown in Fig. 7(c), all objects in an n-dimensional ordered topological model are represented by n-simplices. More concretely, a cell is represented as the set of simplices that have a vertex at that cell. In this manner, even vertices are represented as possibly large sets of simplices.

Attributes

Attributes and geometry can be attached to the simplicial complex in an ordered topological model in a similar way as in a geometric simplicial complex. Since the vertices of each simplex have a known order, and each vertex of each simplex represents a cell, it is possible to link each simplex in an ordered topological model to a tuple of attributes and geometric information for the cells that its vertices represent. Many operations that are possible on a geometric simplicial complex are also possible in an ordered topological model, even considering

that many of its vertices do not have a specific location in space. Topological queries can be handled in an identical way, and some geometric operations require minimal changes. For instance, the length, area, volume, etc. of a cell, can be computed using the inclusion-exclusion principle, which works similarly to the computation of the area of a polygon using the 'shoelace formula'. Comparing objects can also be done simplex by simplex [Gosselin et al., 2011] taking advantage of the ordering properties of ordered topological models.

Construction

Since this simplicial decomposition performed in an ordered topological model is done only combinatorially, it is not necessary to build a constrained triangulation of the input, which as stated in Sec. 3, might be very difficult to accomplish. However, using an ordered topological model still allows one to use the stronger algebraic properties present in a simplicial complex, which are used in order to traverse and manipulate a cell complex. The resulting simplicial complex can have a larger or smaller number of simplices compared to one triangulated geometrically, but it also has the advantage that the number of vertices and simplices in it is known in advance and does not depend on the geometry.

We are aware of two different simplicial decomposition schemes used in ordered topological models, one for the cell-tuple structure [Brisson, 1989] and generalised maps [Lienhardt, 1994], and one for n-dimensional combinatorial maps [Lienhardt, 1994]. Assuming an input n-dimensional cell complex that is in the form of an incidence graph containing all the cells (of every dimension) as nodes, and the incidence relationships from each i-cell to the $(i - 1)$-faces on its boundary as directed edges, the simplicial decompositions are shown in Fig. 7 and explained as follows:

- **Cell-tuple / generalised maps** Each possible path in the graph from an n-cell to a 0-cell directly yields the vertices of one simplex. Each n-simplex
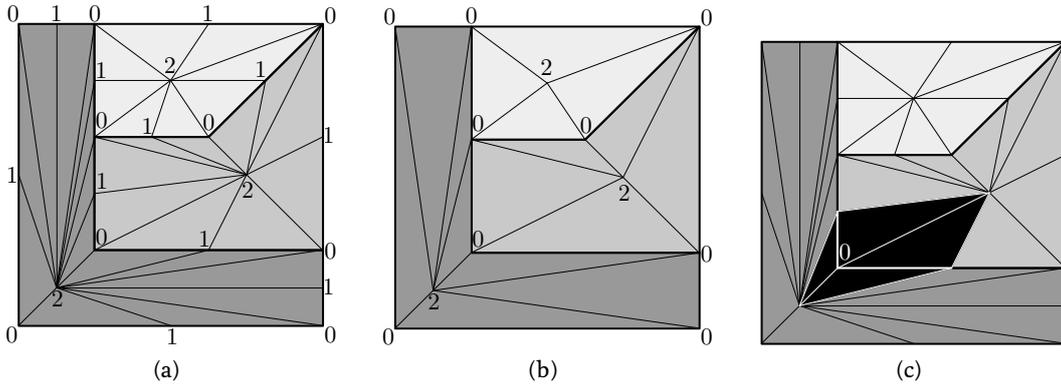
Figure 7: The simplicial decomposition applied in order to obtain the simplices for a 2D: (a) generalised map, (b) combinatorial map. (c) All cells are represented as sets of 2-simplices, such as the vertex shown here, which is formed of a set of 4 triangles. Note that the locations of the vertices for the 1- and 2-cells are arbitrary, those for the 1-cells has been defined as the midpoint of the edge, and those for the 2-cells have been set so that the resulting simplices lie entirely in the interior of their corresponding 2-cell.

can be thus defined by a unique n-tuple $(c_0, c_1, \ldots, c_n)$, where c_i is a vertex representation of an i-dimensional cell in the cell complex, all of which are incident to each other. Such an n-tuple is in fact equivalent to an n-simplex. It can be seen as a generalisation of the lath-based structures [Joy et al., 2003] sometimes used in 2D and 3D GIS.

· **n-dimensional combinatorial maps** Each possible path in the graph from an n-cell to a 1-cell yields one simplex, with the vertices for the cells from dimension n to 2 used directly, prepended with the two vertices corresponding to the two 0-cells incident to the 1-cell (i.e. the vertices at the endpoints of the edge). Each n-simplex can be thus defined by an n-tuple $(c_0, c_0', \ldots, c_n)$, where c_0 and c_0' are the two 0-cells incident to the 1-cell, and for $i > 0$, c_i is also a vertex representation of an i-dimensional cell in the cell complex, all of which are incident to each other and to the two 0-cells. As the two 0-cells could be given in any order, the order is set by specifying an orientation for the cell complex (or for each connected component) by specifying which vertices are respectively c_0 and c_0' in one simplex, and then building the n-tuples for the

other simplices so that adjacent simplices have them in the opposite order. As Fig. 8 shows, this can result in two possible orientations for each cell combinatorial map (or connected component). In practice, this means that if we consider the 1-faces formed by c_0 and
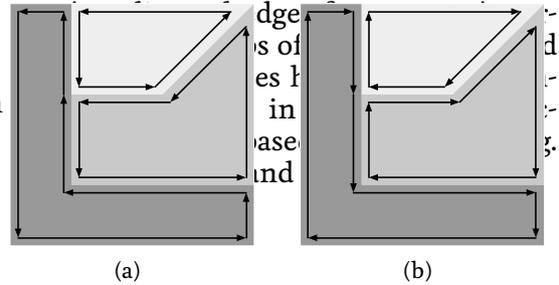
Figure 8: There are two possible consistent orientations that can be set for each connected component in a combinatorial map.

Data structures

The cell-tuple structure [Brisson, 1989], and generalised maps and n-dimensional combinatorial maps [Lienhardt, 1994] support this model directly. Chains of maps [Elter and Lienhardt, 1994] supplement the approach followed by generalised maps with an incidence graph in order to support non-manifolds, but they are rarely used because of their very high space requirements.

Holes, disconnected components and attributes

Holes and disconnected components in an ordered topological model can be dealt with any of the techniques described for cell complexes in Sec. 4, with bridge edges modelled as a pair of simplices with the same vertices but opposite orientations as shown in Fig. 9. Attributes can be handled using

(a) (b)
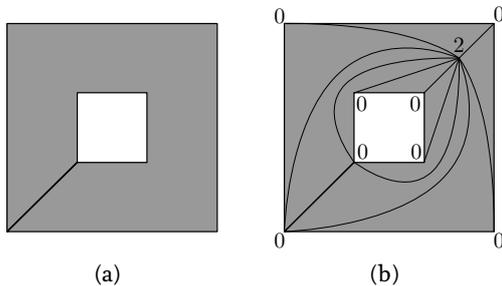
Figure 9: (a) A polygon with a hole is represented using a bridge edge to connect the hole with the exterior boundary. (b) The simplices in the combinatorial maps representation of the situation in (a). There are two simplices with the same vertices representing each side of the bridge edge. As in Fig. 7, the location of the vertex for the 2-cell shown in (b) is arbitrary but chosen so as to lie in the interior of the polygon. The combinatorial triangles are shown with curved lines so as to form a planar embedding to show that they are able to form a partition of the space.

an attributes tuple that links each cell of the vertices tuple to a corresponding structure to store an attribute for it.

Applications

While ordered topological models are relatively difficult to implement, they have nevertheless been implemented in several open source libraries and software, including CGAL [CGAL, 2014], Moka [Damiand, 2014], and CGoGN [Kraemer et al., 2014].

In Arroyo Ohori et al. [2013b], we discussed how generalised maps can be implemented to support various characteristics of real-world data, such as such as markers, locks (for parallel algorithms), simple construction operations and indexing of darts and attributes for the objects of all dimensions. Fradin et al. [2002] used generalised maps in architecture building models such that the level of detail is effectively a geometric dimension. Thomsen et al. [2008] used generalised maps in order to manage topology in 2D/3D GIS in a unified (not dimension dependent) manner. Le [2013] used Gocad [GOCAD Project, 2011] (which internally uses generalised maps) to create 4D geological models by interpolating separate 3D models at various points in time.

6 Conclusions and discussion

Using higher-dimensional representations of objects offers a solution to the integration of parametrisable characteristics, such as time and scale, into GIS in a generic manner. This approach is well-founded on long-standing mathematical theories, and so is able to deal with complex cases and the special requirements of each dimension more robustly than solutions in which parametrisable characteristics are handled in an ad hoc manner.

This approach has been pursued in the past using higher-dimensional rasters and hierarchies of trees [Mason et al., 1994; Varma et al., 1990; Bernard et al., 1998; O'Conaill et al., 1992; Mielke, 2014], but these are only capable of achieving rough approximations of the objects' boundaries and grow

in size exponentially as the dimension increases. Vector data structures with limited topological information, such as Simple Features [OGC, 2011], and the computation of topology on the fly [ESRI, 2005] also become intractable in higher dimensions. We have therefore argued for a topological vector approach that allows for the representation of real-world objects of heterogeneous dimension with holes and attributes, which requires the use of higher-dimensional topological data structures.

Just as in 2D and 3D, there is not a single data structure that works best for all purposes and applications, so choosing an appropriate one is critical. We have classified the dimension-independent data structures that are able to support this approach into three broad data models. Each discretise space in a similar way and use similar combinatorial primitives: geometric simplicial complexes, cell complexes and ordered topological models. The data structures belonging to each of these have like characteristics and support a comparable set of operations.

On one end, geometric simplicial complexes have the most powerful algebra and are the easiest to manipulate, but require the subdivision of each object into simplices using a constrained or conforming triangulation, which is very hard to do in more than 3D and for which there is no known available software. Cell complexes are on the other end, as they are trivial to construct but only support a much weaker set of fundamental operations, which increases the complexity of many computations. Ordered topological models, such as the cell-tuple/generalised maps and combinatorial maps, combine characteristics of the previous two and have both a relatively powerful algebra and are easy to construct, although they are also the most space-intensive and the most complex to implement. However, this last point is easily overcome in practice since there are good implementations of them in several open source libraries and software, including CGAL [CGAL, 2014], Moka [Damiand, 2014], and CGoGN [Kraemer et al., 2014]. All of these models and their corresponding implementations can then be supple-mented with the techniques we described in Sec. 3, Sec. 4 and Sec. 5 in order to support real-world objects with holes, disconnected components and attributes.

Using a higher-dimensional representation of objects enables combined geometric and topological queries across all dimensions. For instance, applied to 4D BIM [Sawyer, 2014] models that include the construction process of a building, it is possible to efficiently do safety checks (e.g. dangerous work is not being performed concurrently and close to other activities, or ensuring that all building components are connected properly at all times) by performing well-defined 4D queries rather than by doing many 3D tests that might not encompass all possible undesirable cases. As more higher-dimensional computational geometry and topology algorithms and libraries become available, such as the recent additions in CGAL [CGAL, 2014] and Gudhi [GUDHI, 2011], this approach also makes it possible to utilise them directly, taking advantage of new developments.

On a final note, we hope that this paper serves to encourage the development of good software libraries to perform key tasks that are currently unavailable in both geometric modelling and GIS, such as geometric computations with simplicial complexes in more than 3D, management of large such complexes, robust Boolean operations on n-polytopes (perhaps on the basis of Nef polyhedra), and general geometric operations on ordered topological models.

Acknowledgements

References

Ricardo Pérez Aguila and Antonio Aguilera Ramírez. Classifying edges and faces as manifold or non-manifold elements in 4d orthogonal pseudo-polytopes. In *WSCG'2003*, 2003.

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

Ken Arroyo Ohori, Pawel Boguslawski, and Hugo Ledoux. Representing the dual of objects in a four-dimensional GIS. In A. Abdul Rahman, P. Boguslawski, C. Gold, and M.N. Said, editors, *Developments in Multidimensional Spatial Data Models*, Lecture Notes in Geoinformation and Cartography, pages 17–31. Springer Berlin Heidelberg, May 2013a.

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

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

C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4): 469–483, December 1996.

Bruce G. Baumgart. A polyhedron representation for computer vision. In *AFIPS '75 Proceedings of the May 19-22, 1975, national computer conference and exposition*, pages 589–596. ACM, 1975.

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.

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

Daniel K. Blandford, Guy E. Blelloch, David E. Cardoze, and Clemens Kadow. Compact representations of simplicial meshes in two and three dimensions. *International Journal of Computational Geometry and Applications*, 15(1):3–24, 2005.

Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. Triangulations in CGAL. *Computational Geometry: Theory & Applications*, 22:5–19, 2002.

B. Boots. Spatial tessellations. In Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind, editors, *Geographical Information Systems*, chapter 36. John Wiley & Sons, 1999.

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

Robin P. Bryant and David Singerman. Foundations of the theory of maps on surfaces with boundary. *Quarterly Journal of Mathematics*, 36(2):17–41, 1985.

Rizwan Bulbul and Andrew U. Frank. AHD: The alternate simplicial decomposition of nonconvex polytopes (generalization of a convex polytope based spatial

data model). In *Proceedings of the 17th International Conference on Geoinformatics*, pages 1-6, 2009.

CGAL. Cgal - computational geometry algorithms library, 2014. Available from: `http://www.cgal.org`.

Guillaume Damiand. Moka modeler, 2014. Available from: `http://moka-modeller.sourceforge.net`.

Guillaume Damiand, Samuel Peltier, and Laurent Fuchs. Computing homology generators for volumes using minimal generalized maps. In Valentin E. Brimkov, Reneta P. Barneva, and Hebert A. Hauptman, editors, *Proceedings of the 12th International Workshop on Combinatorial Image Analysis*, volume 4958 of *Lecture Notes in Computer Science*, pages 63-74. Springer, 2008.

Mark de Berg, Marc van Kreveld, René van Oostrum, and Mark Overmars. Simple traversal of a subdivision without extra storage. *International Journal of Geographical Information Science*, 11(4):359-374, 1997.

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.

René Descartes. *Discours de la méthode*. Jan Maire, Leyde, 1637.

Olivier Devillers. Walking in a triangulation. *International Journal of Foundations of Computer Science*, 13(2):181-199, 2002.

David P. Dobkin and Michael J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. In *Proceedings of the 3rd Annual Symposium on Computational Geometry*, pages 86-99. ACM, 1987.

Max J. Egenhofer and R. D. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161-174, 1991.

Herve Elter and Pascal Lienhardt. Cellular complexes as structured semi-simplicial sets. *International Journal of Shape Modeling*, 1(2), 1994.

ESRI. GIS topology. Technical report, ESRI, July 2005.

P. Fisher. The pixel: a snare and a delusion. *International Journal of Remote Sensing*, 18(3):679-685, 1997.

David Fradin, Daniel Meneveaux, and Pascal Lienhardt. Partition de l'espace et hiérarchie de cartes généralisées : application aux complexes architecturaux. In *Actes des XVèmes journées de l'Association Française d'Informatique Graphique*, volume 28, 2002.

Andrew U. Frank. Spatial concepts, geometric data models, and geometric data structures. *Computers & Geosciences*, 18(4):409-417, 1992.

GOCAD Project. gocad - home, 2011. Available from: `http://www.gocad.org/w4/`.

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 of the 11th Annual European Symposium on Algorithms*, pages 174-186, September 2003.

GUDHI. GUDHI: Geometric Understanding in Higher, 2011. Available from: `https://project.inria.fr/gudhi/team/`.

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

Ralf Hartmut Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider,

and Michalis Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1):1–42, March 2000.

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

Torill Hamre, Khalid Azim Mughal, and Anita Jacob. A 4d marine data model: Design and application in ice monitoring. *Marine Geodesy*, 20(2–3):121–136, 1997.

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

Nicholas W.J. Hazelton, Lisa Marie Bennett, and Joanna Masel. Topological structures for 4-dimensional geographic information systems. *Computers, Environment and Urban Systems*, 16(3):227–237, May–June 1992.

N.W.J. Hazelton. Some operations requirements for a multi-temporal 4-D gis. In M.J. Egenhofer and R.G. Golledge, editors, *Spatial and Temporal Reasoning in Geographic Information Systems*, pages 63–73. Oxford University Press, 1998.

N.W.J. Hazelton, F.J. Leahy, and I.P. Williamson. On the design of temporally-referenced, 3-D geographical information systems: development of four-dimensional GIS. In *Proceedings of GIS/LIS'90*, 1990.

Kenneth I. Joy, Justin Legakis, and Ron MacCracken. Data structures for multiresolution representation of unstructured meshes. In Gerard Farin, Bernd Hamann, and Hans Hagen, editors, *Hierarchical and Geometrical Methods in Scientific Visualization*, Mathematics and Visualization. G. Farin and B. Hamann and H. Hagen, 2003.

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

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.

Hai Ha Le. Spatio-temporal data construction. *ISPRS International Journal of Geo-Information*, 2:837–853, August 2013.

Hugo Ledoux, Ken Arroyo Ohori, and Martijn Meijers. A triangulation-based approach to automatically repair GIS polygons. *Computers & Geosciences*, 66:121–131, May 2014.

Bruno Lévy, Guillaume Caumon, Stéphane Conreaux, and Xavier Cavin. Circular Incident Edge List: A data structure for rendering complex unstructured grids. In *Proceedings of the IEEE Visualization Conference 2001*, San Diego, CA, USA, 2001. IEEE Computer Society.

Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polygons. *Computational Geometry: Theory & Applications*, 35:100–123, 2006.

Pascal Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82, 1991.

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

Yuanxin Liu and Jack Snoeyink. Faraway point: A sentinel point for delaunay computation. *International Journal of Computational Geometry and Applications*, 18(4):343–355, 2008.

Hélio Lopes and Geovan Tavares. Structural operators for modeling 3-manifolds. In Christoph Hoffmann, Wim Bronsvoort, George Allen, Mike Pratt, and David Rosen, editors, *SMA '97, Proceedings of the Fourth Symposium on Solid Modeling and Applications*, pages 10–18. ACM, 1997.

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.

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

Phillip Mielke. New dimensions with arcmap | esri video, 2014. Available from: http://video.esri.com/watch/3653/new-dimensions-with-arcmap.

Eric J. Miller. Towards a 4D GIS: Four-dimensional interpolation utilizing kriging. In Zarine Kemp, editor, *Innovations in GIS*, chapter 13, pages 181-197. Taylor & Francis, 1997.

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

B. Naylor. Binary space partitioning trees as an alternative representation of polytopes. *Computer-Aided Design*, 22(4), 1990.

M. A. O'Conaill, S. B. M. Bell, and N. C. Mason. Developing a prototype 4D GIS on a transputer array. *ITC Journal*, (1):47-54, 1992.

OGC. OpenGIS implementation specification for geographic information - simple feature access - part 1: Common architecture. Technical report, Open Geospatial Consortium, May 2011.

A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56-102, January 1993.

A. Paoluzzi, V. Pascucci, and G. Scorzelli. Progressive dimension-independent boolean operations. In G. Elber, N. Patrikalakis, and P. Brunet, editors, *ACM Symposium on Solid Modeling and Applications*. ACM, 2004.

Alexander Pasko and Valery Adzhiev. Constructive hypervolume modeling. *Graphical Models*, 63:413-442, 2001.

Sinésio Pesco, Geovan Tavares, and Hélio Lopes. A stratification approach for modeling two-dimensional cell complexes. *Computers & Graphics*, 28:235-247, 2004.

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

M.H. Poincaré. Analysis situs. *Journal de l'École polytechnique*, 2(1):1-123, 1895.

Jovan Popović and Hughes Hoppe. Progressive simplicial complexes. In G. Scott Owen, Turner Whitted, and Barbara Mones-Hattal, editors, *SIGGRAPH '97 Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 217-224. ACM/Addison-Wesley Publishing, 1997.

Jonathan Raper, editor. *Three Dimensional Applications in Geographic Information Systems*. Taylor & Francis, 1989.

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

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.

Jarek Rossignac and Andrzej Szymczak. Wrap&zip decompression of the

connectivity of triangle meshes compressed with edgebreaker. *Computational Geometry: Theory & Applications*, 14, 1999.

Jim Ruppert and Raimund Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete & Computational Geometry*, 7(1):227-253, 1992.

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.

Tom Sawyer. Construction managers embrace 4d bim for safety, 2014. Available from: `http://enr.construction. com/technology/bim/2014/ 0602-Dynamic-Models-for-Safer-Sites. asp?page=2`.

Jonathan Richard Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203-222. Springer-Verlag, May 1996.

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.

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

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

Jeremy Tammik. Autocad nef polyhedron implementation. Available at `http://citeseerx.ist.psu.edu/ viewdoc/download?doi=10.1.1.89. 6020&rep=rep1&type=pdf`, September 2007.

Andreas Thomsen, Martin Breunig, and Edgar Butwilowski. Towards a g-map based tool for the modelling and management of topology in multiple representation databases. *Journal of Photogrammetry, Remote Sensing and Geoinformation Processing*, 3:175-186, 2008.

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.

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

Lidija Čomić, Leila de Floriani, Federico Iuricich, and Ulderico Fugacci. Topological modifications and hierarchical representation of cell complexes in arbitrary dimensions. *Computer Vision and Image Understanding*, 121:2-12, 2014.

Luiz Velho. Stellar subdivision grammars. In L. Kobbelt, P. Schröder, and H. Hoppe, editors, *Eurographics Symposium on Geometry Processing*. The Eurographics Association, 2003.

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

Michael Worboys. The maptree: A fine-grained formal representation of space. In Ningchuan Xiao, Mei-Po Kwan, Michael F. Goodchild, and Shashi Shekhar, editors, *Proceedings of the 7th*

International Conference GIScience 2012, volume 7478 of *Lecture Notes in Computer Science*, pages 298–310. Springer, 2012.

Michael Worboys and Matt Duckham. *GIS: A Computational Perspective*. CRC Press, second edition edition, 2004.

Michael F. Worboys. A generic model for planar geographical objects. *International Journal of Geographical Information Systems*, 6(5):353–372, 1992b.