Representations of 3D and 4D city models

Ken Arroyo Ohori Hugo Ledoux Jantien Stoter
Anna Labetski Stelios Vitalis Kavisha Kumar

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

Representations of 3D and 4D city models. Ken Arroyo Ohori, Hugo Ledoux, Jantien Stoter, Anna Labetski, Stelios Vitalis and Kavisha Kumar. In Sultan Kocaman, Devrim Akca, Daniela Poli and Fabio Remondino (eds.), *3D/4D City Modelling - From Sensors to Applications*, Chapter 6, Whittles Publishing, 2024. ISBN: 978-1-8499-5475-4, Electronic ISBN: 978-1-5231-5811-9.

Although level of detail (LoD) is a central concept in 3D city modelling, specifying different LoDs in an unambiguous manner is not straightforward. To resolve this, a set of frameworks have been developed. This paper evaluates the suitability of the LoD framework of Biljecki et al. [2016c] for 3D building models that have been generated directly from BIM models. The output of two BIM shell extractors are tested on how well they can be defined by the framework. It was found that although BIM-derived models can be specified by the framework to a certain degree, the framework is not fully capable to also specify lower quality models and to support all the output that may come from BIM shell extractors. This can be resolved by either addressing issues in the shell extractors' output or in the framework itself. The results of this research can be used to improve the LoD framework and to adjust the shell extractors output to better comply with unambiguous definitions of building models at different LoDs and could be a first step to standardise the conversion of BIM models at different LoDs to be used in urban applications.

1 Introduction

Recent advances in technologies to collect 3D elevation information, eg LiDAR and photogrammetry [Mallet and Bretar, 2009, Haala and Rothermel, 2012, Shahzad and Zhu, 2015], have made it relatively easy for practitioners in different fields to automatically reconstruct 3D city models. These models typically contain buildings [Alexander et al., 2009, Rottensteiner, 2003], as well as other city objects such as roads, overpasses, bridges, and trees [Oude Elberink, 2010]. Their availability and applications are steadily increasing in the fields of city planning and environmental simulations such as urban noise mapping, flood simulations, and disaster management [Biljecki et al., 2015]. Furthermore, since elevation data can be acquired every few months/years at a relatively low cost, so-called 4D city models—ie 3D city models covering the same region at different periods in time—can also be reconstructed. Historical 4D city models for the cities of Rotterdam, Duisburg, Solothurn, Prague, Toul, and Hamburg have been constructed using close range photogrammetry techniques and a variety of data sources such as maps, photographs, paintings, and wooden models [Kersten et al., 2012]. These models not only offer a view of these cities in the past, but they are also useful for understanding the changes happening in a city over time. As further explained in Section 2.5, the fourth dimension can also be, instead of time, the different levels-of-detail (LoDs) of a 3D city model.

It should be noticed that since different 3D/4D city models are in practice generated independently using different reconstruction methods, software and sensor data, the resulting models often significantly differ in their geometry (eg a collection of surfaces versus a volumetric representation), appearance, and semantics (often not present at all). In addition, every application requires its own specific semantic and geometric LoD of the 3D data. Also, as these models are stored using different formats (eg text, XML, or binary formats), their underlying data models often also differ. Substantial differences in models can even happen to models that were originally identical through updates or through conversions between different formats. All these differences have profound influences in practice, such as the applications for which a 3D or 4D model can be used, the processing that is necessary to use it, and the likely errors that will be present in the end result.

It is thus important to be aware of the way in which 3D and 4D city models are actually modelled. In this chapter we focus on a specific aspect of this: the main data models and formats used in practice to store and exchange city models. We focus primarily on open standards in 3D (Section 2) and their possible extension to 4D. Since interoperability between different 3D/4D data models is required to facilitate seamless exchange and use of city models, we also describe some of the interoperability issues between different standards and some possible solutions to convert between them (Section 3).

Next, we explain briefly some of the difficulties around extending reconstruction methods from 3D to 4D, such as how to deal with topology (Section 4). Despite the infancy of 4D modelling, we also discuss the issues that 4D representations will face in this regard. We also discuss in Section 5 how the 3D, and 4D, city models can be visualised in practice.

2 Data models for modelling cities

2.1 Standard 3D visualisation formats

The 3D visualisation formats encode the geometry and appearance of a 3D model, as well as some ancillary information that is useful within a 3D modelling program, such as the model's scene (position of light sources and cameras), and/or a set of animations done with it. These are used in several fields, mostly for visualisation purposes. However, while they can be used for 3D city modelling, they do not model two

very important aspects of a city model: its semantic and its topological details.

The main standard 3D visualisation formats are:

VRML (Virtual Reality Modelling Language)[1] is an open text-based format for modelling dynamic and interactive 3D scenes. It was accepted as the first web based 3D standard by the web 3D consortium in 1995 and is still a widely supported format by popular tools such as FME, and SketchUp. The format offers 3D geometries, several texturing mechanisms, animations, and scripting.

X3D (Extensible 3D)[2] was developed in 2001 as the successor of VRML. It is essentially an XML based encoding of VRML with added functionalities, such as geospatial positioning, shaders, and the capability to store scene information. The rendering and texturing of 3D models are based on the functionalities provided by the low-level graphics engines such as OpenGL and DirectX. It also has support for classic VRML, binary, and JSON encoding. It is well supported by many browser plug-ins, as well as data generation and conversion tools.

OBJ (Wavefront Object)[3] is one of the most popular text-based formats in the 3D graphics community. It has simple 3D geometries such as polygons and triangles for storing 3D models. The OBJ format can also encode colour and texture information which is stored in a separate file with the extension .MTL (Material Template Library). It does not support animation nor the modelling of scenes.

COLLADA (COLLAborative Design Activity)[4] is an open XML-based format by Khronos Group for the representation and exchange of 3D assets. Its focus is primarily on the exchange of geometry data and 3D scenery. COLLADA supports triangular mesh geometry, and has extensive shading and texturing options, animations, physics, and even multiple version representations of the same asset. It is commonly used with KML (Keyhole Markup Language) to render 3D city models in Google Earth.

glTF (GL Transmission Format)[5] is a JSON-based open 3D format by Khronos Group for the exchange of 3D models. It also has a binary encoding for storing mesh geometry and animation data. It provides compact representation of geometries, and small file sizes.

Other 3D visualisation formats include: PLY (Polygon File Format), OFF (Object File Format), STL (STereoLithography), I3S (Indexed 3D Scene Layer), etc.

2.2 CityGML and CityJSON

While the aforementioned standard 3D modelling formats can be used to store cities, it is often more useful to store a 3D city model in a specially structured format with semantic information stored in a standardised way. In this manner, a semantic 3D city model can be readily processed and visualised using existing tools. **CityGML** [OGC, 2012] is the main of such standards. Its aim is to define the basic standard classes that can be used to describe the most common types of objects present in a 3D city model, their components, their attributes and the relationships between different objects. CityGML is based on a number of standards from the ISO191xx family, and it is used both as an information model (eg in the form of UML models of its classes) and a data format, which is an XML-based representation of its classes using some definitions from the Geography Markup Language (GML) [OGC, 2007].

Although most CityGML examples and datasets available focus only on buildings,

[1] https://www.w3.org/MarkUp/VRML/
[2] http://www.web3d.org/x3d/what-x3d
[3] http://paulbourke.net/dataformats/obj/
[4] https://www.khronos.org/collada/
[5] https://www.khronos.org/gltf/

CityGML allows us to represent other feature classes such as: relief, roads and railways, vegetation, bridges, and city furniture. These can be supplemented with textures and/or colours to give a better impression of their appearance. Specific relationships between different objects can also be stored using CityGML, for example that a building is decomposed into three building parts, or that a building has a both a carport and a balcony.

CityGML defines different standard levels of detail (LoDs) for all 3D objects; Figure 1 shows the 5 possible levels for buildings, which is the most known concept of CityGML. These provide the possibility to represent objects for different applications and purposes, and can be useful for visualisation.

While CityGML prescribes a data model (feature classes and their attributes) for a 'generic' city (eg the function or year of construction of a building), it is possible to extend it for specific domains by defining ADEs (Application Domain Extensions). An ADE extends the CityGML schema with some new classes and/or new attributes for the existing features/elements. Examples of such extensions are the ones for: the energy demand of buildings [Agugiaro et al., 2018] and for a country-specific data model for the Netherlands [Van den Brink et al., 2013]. The main issue that ADEs are facing is that few software packages and libraries can read them automatically and thus process the information.

Apart from the most common XML-based CityGML, there are others that implement its data model. For instance, **CityJSON**[6] is a format that encodes a subset of the CityGML data model using JavaScript Object Notation (JSON). It offers an alternative to the GML encoding of CityGML, in which objects can be defined in a very large number of possible ways, and which can therefore be verbose and complex (and thus rather cumbersome to work with). CityJSON aims at being easy-to-use, both for reading datasets, and for creating them. It was designed with programmers in mind, so that tools and APIs supporting it can be quickly built. It is also designed to be compact, with a compression factor of around seven when compared to XML-based CityGML, and it is also friendly for web and mobile development. A CityJSON object, representing a given area, is as 'flat' as possible, ie the hierarchy of CityGML has been flattened out and only the city objects which are 'leaves' of this hierarchy are implemented. This considerably simplifies the storage of a city model compared to CityGML, and all information is kept.

Another implementation is **3DCityDB**[7], which is an open-source database, built upon Oracle Spatial[8] or PostGIS[9], to store in a relational database the CityGML data model. It offers several extra functionalities to import/export city models from/to different formats, and can thus help for interoperability.

2.3 LandInfra/InfraGML

The LandInfra conceptual model was developed by the OGC in cooperation with buildingSMART, with the aim to bridge the gap between two disciplines: Architecture, Engineering and Construction (AEC), and GIS industry [OGC, 2016c]. It defines implementation-independent concepts for representing land and civil engineering infrastructure facilities such as buildings, roads, railways, and other features such as vegetation and terrain [OGC, 2016c].

Currently there is only one implementation of the LandInfra model: InfraGML, which is a GML-based implementation. It has 7 parts covering land features, facilities, alignment, roads, railways, surveys (including equipment, observations, and survey results), and land division [OGC, 2017]. Each part of InfraGML is a separate OGC standard. *Core* is the mandatory part of the standard which is extended by the other parts. It models the information contained in an InfraGML

[6] http://www.cityjson.org

[7] https://www.3dcitydb.org

[8] http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/index.html

[9] https://postgis.net/

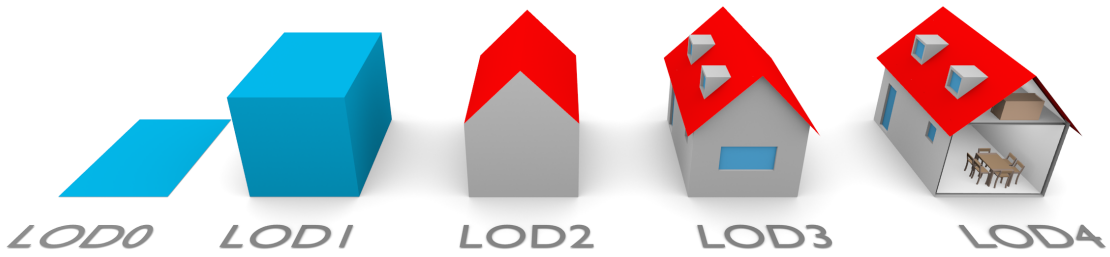


Figure 1: A building represented in LoD0 to LoD4 (Figure from Biljecki et al. [2014a]). Notice there are four (volumetric) LoDs (LoD1 to LoD4); the non-volumetric LoD0 is an horizontal footprint and/or roof surface representation for buildings.

dataset, the definitions of the feature types, and the associations used in other parts.

InfraGML is expected to easily integrate with other OGC standards such as CityGML. There are significant overlaps between CityGML and InfraGML with respect to modelling of spatial objects [OGC, 2016c]. For instance, many CityGML objects are present as feature types in InfraGML such as Building (Building), Road (Road), ReliefFeature (LandSurface), WaterBody (LandFeature), VegetationObject (LandFeature), and LandUse (AdministrativeDivision) [OGC, 2016c]. The alignment between InfraGML and CityGML standards is currently a topic of interest in the GIS community [Kumar, 2017].

2.4 BIM/IFC

The Industry Foundation Classes (IFC)[10] standard is an open data model used in the Building Information Modelling (BIM) domain for managing the information sharing processes throughout the life cycle of the building. While it was originally designed only for 3D buildings (and detailed representation of their interiors), the next version (IFC5) will support various infrastructure domains (eg roads, bridges, viaducts, etc.).

It has been adapted as the ISO 16739 international standard [ISO, 2013]. Its geometric aspects are however mostly defined or derived from a different standard, ISO 10303 [ISO, 2014], which also specifies the STEP encoding that is most commonly used in IFC files.

IFC files can contain many types of classes (representing different aspects of a building, for instance a door, a staircase, a wall, the electric heater type, etc.), and the geometries used can be of different representation paradigms: (1) primitive instancing; (2) constructive solid geometry (CSG); (3) sweep volumes; (4) boundary-representation (b-rep). These four paradigms can be combined more or less freely; however, in practice, most IFC objects are built using sweep volumes and CSG [El-Mekawy and Östman, 2010].

IFC models are highly relevant for 3D and 4D city modelling because, for many cities around the world, there is already a number of such models available, mostly for new buildings, manually made by architects. The automatic conversion between IFC models and other simple ones (eg CityGML) is a challenge in practice [Arroyo Ohori et al., 2017a, OGC, 2016b].

As mentioned before, the IFC standard is currently being extended so that other infrastructures of a city (eg roads and bridges) can be modelled[11]. The alignment between IFC and other relevant standards, eg CityGML, is also being investigated.

[10] http://www.buildingsmart-tech.org/specifications/ifc-releases

[11] http://www.buildingsmart-tech.org/infrastructure

2.5 4D city modelling

2.5.1 4D = 3D+time

The fourth dimension in 4D modelling often refers to the temporal aspect of the 3D data. Several researchers have tried to embed temporal aspects of geographic data in a 2D/3D+time data. The semantics of the time dimension included in these models vary from model to model but are often represented as separate attributes either of an object or an event. An exception is the Space Time Cube (STC) [Kraak, 2008]. However, the aim of the STC is merely to provide visual insight into the temporal aspect rather than to realise a data structure to fully handle changes upon position, attributes and/or extent of the objects in a unified space-time-scale continuum. Arroyo Ohori et al. [2017c] describe spatiotemporal models that have been developed over the years.

The earliest approach to show time-related information of geo-data is as a series of separate snapshots where each snapshot is a representation of the dataset at one moment in time. Every object in the dataset is considered static until the time of the next snapshot. The main problem of this approach is that objects are represented multiple times, ie in all snapshots that they are part of, which causes data redundancy and possible data inconsistencies (due to edits in a snapshot that are not propagated into the others). Another problem is that it is not recorded when a change occurred, ie it could have happened at any time between two snapshots. A solution to the problem of redundancy is to assign timestamps to every object which demarcate the start and end of the period during which they exist. Therefore, objects only need to be represented again when they change. However, this approach also does not contain explicit events. In addition, in practice it appears ambiguous to decide when a change results in eliminating the object and creating a new one or keeping the existing one with changed attributes. Other approaches use events as principal entities, ie points in time where objects change, such as by keeping a list of changes per object. This makes it possible to know when events exactly occurred, and to identify and attach attributes to individual changes and events (eg what an event represents or why it occurred).

All these solutions aim at capturing the change of objects (and their attributes). Chaturvedi and Kolbe [2016] propose a new concept called 'Dynamizer' allowing to also model dynamic properties (ie temperature change over a day or year) of city objects and sensor observations. The 'Dynamizer' concept can be applied to all city objects and is implemented as an Application Domain Extension for the CityGML standard.

2.5.2 4D = 3D+LoD

As stated in the Introduction, 3D city models often differ in their geometry, appearance, LoD, and semantics. This is due to the different reconstruction methods, software, and sensor data, used to reconstruct the 3D city model. In addition, every application requires its own specific semantic and geometric LoD of the 3D data [Biljecki et al., 2015]. Take the urban object `Building` in Figure 1. Block models (LoD1) are sufficient for shadow simulations and the estimation of noise pollution, energy demand and fluid dynamics. Roof structures (LoD2) with information on the roof materials are needed for solar potential estimation, or in energy demand estimation. More detailed building models with information about windows and doors (LoD3) are important for estimating heat losses and for calculating the area available on vertical walls for solar panel installation. Building models that contain indoor spaces (LoD4) are required for indoor navigation and evacuation models.

LoD has a strong relationship to "scale" as traditionally used for 2D maps, but in 3D it has a wider meaning [Biljecki et al., 2014b]. It does not only reflect a ratio between measures in reality and on a map, but also the amount of information (semantics and geometry) to be included to serve a specific application.

One of the strengths of CityGML is the support for five different LoDs. However, there

are two major challenges of the CityGML LoDs. First, the standard allows for many alternatives for one LoD, and therefore in practice a lot of variances of one LoD occur. For example, a LoD2 building with or with or without roof overhangs; with or without explicit modelling of roof, wall and floor; with or without dormers, etc. These variances are not further specified in CityGML, but, as argued in Biljecki et al. [2016b], a more formal subdivision will significantly improve the quality of 3D city models. The authors therefore propose a further subdivision of the four CityGML LoD outdoor models for buildings (LoD0 to LoD3). The subdivision (resulting into 16 LoDs) is based on criteria as the minimum size of an element (such as wall indentations and dormers) and existence of specific elements such as roof overhangs, chimneys and openings (latter two only for LoD3).

The second challenge of the CityGML LoDs is that while the LoD for buildings are frequently studied and well-known, the LoDs of other object types have obtained less attention. CityGML does not contain clear specifications for LoDs for other object types with the exception of tunnels and bridges for which the LoDs are modelled similar to those of buildings. For vegetation, a `SolitaryVegetationObject` or a `PlantCover` may have a different geometry in each LoD, but the standard does not specify this further. Also `LandUse` objects (which can model the surface of the Earth as a subdivision into several polygons) may have different geometries in all LoD0–LoD4. But the CityGML standard does not provide any further guidance or specifications either, nor does it for the LoDs of the `Relief` class which can be used to model the shape of the terrain.

Roads, another prominent object type in 3D city models, is modelled in CityGML and gets a higher complexity at higher LoDs. But these are different from the LoDs of CityGML `Buildings`. At LoD0, they are represented as a linear network and starting from LoD1, all transportation features are geometrically described by 3D surfaces, with a further thematic division of the road surface at higher LoDs. The CityGML standard has no information on the difference between the road surface representation at LoD1, LoD2, LoD3 or LoD4.

2.5.3 Representation of 4D data models

So far there is no established data model for 3D+time and the 3D+LoD city models as described in the previous sections. They are therefore usually stored as separate 3D city models with some added metadata that explains the date of the model and possibly how the model was generated. While this is sufficient in some cases, it makes it difficult to automatically process such a model, requiring additional tasks such as performing a topological reconstruction of the model in 3D or 4D (Section 4).

Another challenge for 4D city models is their size—something that is easy to see if we compare the standard ways in which 2D city models are represented. In the Simple Features Specification [OGC, 2011], which reflects the prototypical way in which geometries are stored in 2D and 3D, independent 2D primitives are defined as sequences of points (with coordinates) that are connected using implicit line segments. When 3D structures constructed from these 2D primitives are described (eg polyhedra), they are simply considered as sets of these 2D primitives.

Despite their apparent limitations, these types of structures can be in fact used to store objects of any dimension thanks to the Jordan-Brouwer theorem [Lebesgue, 1911, Brouwer, 1911]. Just as independent 2D primitives are defined as sequences of points (with coordinates) that are connected using implicit line segments, 3D primitives can be then represented as sets of such 2D elements, which are otherwise unlinked, and nD objects can be represented as aggregations of their $(n-1)$D-face bounding elements. Since every point in such an object can have any number of coordinates, an nD object can be embedded in nD space as well.

However, such representations become exponentially more inefficient as the dimension increases: lower-dimensional primitives need to be encoded multiple times, re-

cursively once for each time they appear in a higher-dimensional primitive. This means that they are difficult to navigate and that even simple geometric and topological queries involve searching many objects [Hazelton, 1998]. As an example, it is possible to consider Simple Features-like [OGC, 2011] representations of simple objects of various dimensions, as shown in Figure 2. A tesseract, or 4-cube, is the four-dimensional analogue of a square (in 2D) or cube (in 3D), it consists of 16 vertices, 32 edges, 24 faces, 8 volumes.

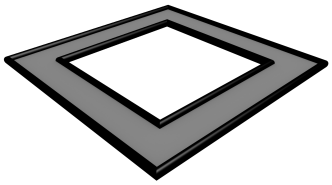
Considering how inefficient such representations can already be in 4D, it therefore makes more sense to use a topological representation for 4D city models [Arroyo Ohori, 2016]. There are a number of ways in which this can be done in theory, such as with Nef polyhedra [Bieri and Nef, 1988], and with ordered topological models such as the cell-tuple [Brisson, 1993] and generalised/combinatorial maps [Lienhardt, 1994]. These are still rather memory-intensive, but they can still be more compact than a non-topological approach. Research into applying these for 4D city models is still ongoing, and more works need to be done in order to make this a practical approach.

3 Interoperability between 3D city models

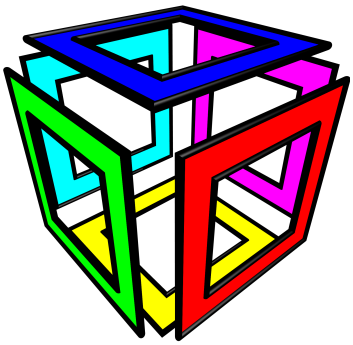
While there exist international standards providing unambiguous definitions of the 3D geometries in a 3D city model, as explained in Ledoux [2013, 2018], the majority of 3D GIS software ignore them and use their own definitions. The differences are fundamental: the abstract specification ISO 19107 [ISO, 2003] and its implementation in XML/GML [OGC, 2007] allow surfaces embedded in 3D space and solids to have inner boundaries (as is the case in 2D where polygons can have "holes"), while many software packages ignore these. Indeed, solids/volumes are often assumed to be bounded only by simple surfaces (thus forming a 2-manifold), while in fact they can be non-manifold objects. Such limitations can prevent practitioners from exchanging and converting datasets (since information is lost), and thus to use these in other software and applications.

Another issue that does not facilitate the exchange of 3D city models is that, in practice, their quality is often poor. As highlighted by Biljecki et al. [2016a], most openly available 3D city models contain geometric and topological errors, eg duplicate vertices, missing surfaces, self-intersecting volumes, etc. Often these errors are not visible at the scale the datasets are visualised [Laurini and Milleret-Raffort, 1994], and as a consequence, practitioners are not aware of the problem. But these errors prevent us from using the datasets in other software and applications, see Nouvel et al. [2017], Steuer et al. [2015], and Bruse et al. [2015] for concrete examples in different application areas. While these geometric errors are many, they could be prevented if modelling software enforced the 3D geometries to be ISO 19107-compliant. Another solution to this problem is to use automatic repair algorithms; Attene et al. [2013] offer a survey of the methods to repair 3D models, as found in several disciplines. However, city models are not addressed, and the focus is on "smooth surfaces", which are seldom in a city context since buildings and bridges tend to have planar surfaces perpendicular to other surfaces.

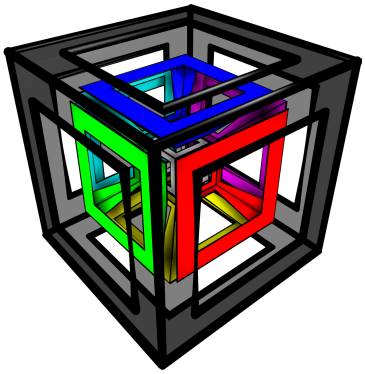
Besides the geometry, the conversion of semantic 3D city models, from one format to another, is problematic because different data models might have incompatible semantics. One example is the differences between the object classes modelled in IFC and in CityGML. Take for instance a building (which both standards model), the mappings between the semantic classes are complex because different semantic information is attached to the geometrical primitives in the two models, and IFC has many more classes, whereas CityGML contains a limited number of classes structured in a hierarchy. A solution that has been proposed, among others by El-Mekawy et al. [2012], is to construct a common data model; this however does not allow us to use the files in



(a) square



(b) cube



(c) tesseract

```
[[0, 0], [0, 1], [1, 1], [1, 0], [0, 0]]
```

(d) square (SFS)

```
[[[0, 0, 0], [0, 1, 0], [1, 1, 0], [1, 0, 0], [0, 0, 0]],
 [[0, 0, 0], [0, 1, 0], [0, 1, 1], [0, 0, 1], [0, 0, 0]],
 [[0, 1, 0], [1, 1, 0], [1, 1, 1], [0, 1, 1], [0, 1, 0]],
 [[1, 1, 0], [1, 0, 0], [1, 0, 1], [1, 1, 1], [1, 1, 0]],
 [[1, 0, 0], [0, 0, 0], [0, 0, 1], [1, 0, 1], [1, 0, 0]],
 [[0, 0, 1], [0, 1, 1], [1, 1, 1], [1, 0, 1], [0, 0, 1]]]
```

(e) cube (SFS)

```
[[[[0, 0, 0, 0], [0, 1, 0, 0], [1, 1, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0]],
  [[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 1, 0], [0, 0, 1, 0], [0, 0, 0, 0]],
  [[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 1, 0], [0, 1, 1, 0], [0, 1, 0, 0]],
  [[1, 1, 0, 0], [1, 0, 0, 0], [1, 0, 1, 0], [1, 1, 1, 0], [1, 1, 0, 0]],
  [[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 1, 0], [1, 0, 1, 0], [1, 0, 0, 0]],
  [[0, 0, 1, 0], [0, 1, 1, 0], [1, 1, 1, 0], [1, 0, 1, 0], [0, 0, 1, 0]]],
 [[[0, 0, 0, 0], [0, 1, 0, 0], [1, 1, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0]],
  [[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 0, 1], [0, 0, 0, 1], [0, 0, 0, 0]],
  [[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 0, 1], [0, 1, 0, 1], [0, 1, 0, 0]],
  [[1, 1, 0, 0], [1, 0, 0, 0], [1, 0, 0, 1], [1, 1, 0, 1], [1, 1, 0, 0]],
  [[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1], [1, 0, 0, 1], [1, 0, 0, 0]],
  [[0, 0, 0, 1], [0, 1, 0, 1], [1, 1, 0, 1], [1, 0, 0, 1], [0, 0, 0, 1]]],
 [[[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 1, 0], [0, 0, 1, 0], [0, 0, 0, 0]],
  [[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 0, 1], [0, 0, 0, 1], [0, 0, 0, 0]],
  [[0, 1, 0, 0], [0, 1, 1, 0], [0, 1, 1, 1], [0, 1, 0, 1], [0, 1, 0, 0]],
  [[0, 1, 1, 0], [0, 0, 1, 0], [0, 0, 1, 1], [0, 1, 1, 1], [0, 1, 1, 0]],
  [[0, 0, 1, 0], [0, 0, 0, 0], [0, 0, 0, 1], [0, 0, 1, 1], [0, 0, 1, 0]],
  [[0, 0, 0, 1], [0, 1, 0, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]]],
 [[[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 1, 0], [0, 1, 1, 0], [0, 1, 0, 0]],
  [[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 0, 1], [0, 1, 0, 1], [0, 1, 0, 0]],
  [[1, 1, 0, 0], [1, 1, 1, 0], [1, 1, 1, 1], [1, 1, 0, 1], [1, 1, 0, 0]],
  [[1, 1, 1, 0], [0, 1, 1, 0], [0, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 0]],
  [[0, 1, 1, 0], [0, 1, 0, 0], [0, 1, 0, 1], [0, 1, 1, 1], [0, 1, 1, 0]],
  [[0, 1, 0, 1], [1, 1, 0, 1], [1, 1, 1, 1], [0, 1, 1, 1], [0, 1, 0, 1]]],
 [[[1, 1, 0, 0], [1, 0, 0, 0], [1, 0, 1, 0], [1, 1, 1, 0], [1, 1, 0, 0]],
  [[1, 1, 0, 0], [1, 0, 0, 0], [1, 0, 0, 1], [1, 1, 0, 1], [1, 1, 0, 0]],
  [[1, 0, 0, 0], [1, 0, 1, 0], [1, 0, 1, 1], [1, 0, 0, 1], [1, 0, 0, 0]],
  [[1, 0, 1, 0], [1, 1, 1, 0], [1, 1, 1, 1], [1, 0, 1, 1], [1, 0, 1, 0]],
  [[1, 1, 1, 0], [1, 1, 0, 0], [1, 1, 0, 1], [1, 1, 1, 1], [1, 1, 1, 0]],
  [[1, 1, 0, 1], [1, 0, 0, 1], [1, 0, 1, 1], [1, 1, 1, 1], [1, 1, 0, 1]]],
 [[[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 1, 0], [1, 0, 1, 0], [1, 0, 0, 0]],
  [[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1], [1, 0, 0, 1], [1, 0, 0, 0]],
  [[0, 0, 0, 0], [0, 0, 1, 0], [0, 0, 1, 1], [0, 0, 0, 1], [0, 0, 0, 0]],
  [[0, 0, 1, 0], [1, 0, 1, 0], [1, 0, 1, 1], [0, 0, 1, 1], [0, 0, 1, 0]],
  [[1, 0, 1, 0], [1, 0, 0, 0], [1, 0, 0, 1], [1, 0, 1, 1], [1, 0, 1, 0]],
  [[1, 0, 0, 1], [0, 0, 0, 1], [0, 0, 1, 1], [1, 0, 1, 1], [1, 0, 0, 1]]],
 [[[0, 0, 1, 0], [0, 1, 1, 0], [1, 1, 1, 0], [1, 0, 1, 0], [0, 0, 1, 0]],
  [[0, 0, 1, 0], [0, 1, 1, 0], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 1, 0]],
  [[0, 1, 1, 0], [1, 1, 1, 0], [1, 1, 1, 1], [0, 1, 1, 1], [0, 1, 1, 0]],
  [[1, 1, 1, 0], [1, 0, 1, 0], [1, 0, 1, 1], [1, 1, 1, 1], [1, 1, 1, 0]],
  [[1, 0, 1, 0], [0, 0, 1, 0], [0, 0, 1, 1], [1, 0, 1, 1], [1, 0, 1, 0]],
  [[0, 0, 1, 1], [0, 1, 1, 1], [1, 1, 1, 1], [1, 0, 1, 1], [0, 0, 1, 1]]],
 [[[0, 0, 0, 1], [0, 1, 0, 1], [1, 1, 0, 1], [1, 0, 0, 1], [0, 0, 0, 1]],
  [[0, 0, 0, 1], [0, 1, 0, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]],
  [[0, 1, 0, 1], [1, 1, 0, 1], [1, 1, 1, 1], [0, 1, 1, 1], [0, 1, 0, 1]],
  [[1, 1, 0, 1], [1, 0, 0, 1], [1, 0, 1, 1], [1, 1, 1, 1], [1, 1, 0, 1]],
  [[1, 0, 0, 1], [0, 0, 0, 1], [0, 0, 1, 1], [1, 0, 1, 1], [1, 0, 0, 1]],
  [[0, 0, 1, 1], [0, 1, 1, 1], [1, 1, 1, 1], [1, 0, 1, 1], [0, 0, 1, 1]]]]
```

(f) tesseract (SFS)

Figure 2: A unit square, cube and tesseract shown (a–c) graphically with hollowed and separated facets for better visibility, and with a (d–f) Simple Features-like representations where one face (per volume and per 4-cell) is shown in each line. Due to the repetition of structures in this type of representation, a 5-cube, which has only 80 faces, requires 480 lines to be represented, which is 10 times the number of lines used to represent the tesseract.

already existing software, which would typically support either of these formats.

A vivid example of the current difficulties related to the interoperability of 3D city models is the automatic conversion between IFC models and CityGML models. While in theory this is attractive since the 3D reconstruction phase could be skipped (since in many countries IFC files of new buildings are available), in practice the differences in semantics, coupled to the fact that different software and geometric modelling paradigms are used, have made the conversion impossible. OGC [2016a] and Arroyo Ohori et al. [2017a], among others, explain what are the issues that prevent us from automating the process, and provide recommendations so that both standards become better aligned.

It should be said that the conversion between the purely geometrical formats (such as VRML, OBJ, glTF, and OFF in Section 2.1) and semantic 3D city models is usually not a major issue since these are often formed solely of triangles without any semantics, and the geometric errors are thus relatively easily solvable.

4 Construction of 3D and 4D models

4.1 Topological representations of 3D city models

One of the key benefits of 3D city models is that they can be processed automatically in a variety of applications [Biljecki et al., 2015]. This often involves the evaluation of topological relations between geometrical objects in 3D space. When these topological relationships are not known, we need to first identify the adjacency and intersection relationships between the objects [Zlatanova, 2000, Guo et al., 2010].

As an alternative to expensive computations to obtain the topological relationships between objects on the fly, these can be precomputed and a topological data structure can be used in order to store those relationships directly in the dataset. By doing so, the calculation efficiency for many applications can be improved, such as doing network analysis of evacuation scenarios [Choi and Lee, 2009] or performing rendering on mobile devices [Ellul and Altenbuchner, 2014]. The most commonly used topological data structures are the Doubly Connected Edge List (DCEL) or half-edge data structure [Muller and Preparata, 1978] and the quad-edge data structure [Guibas and Stolfi, 1985]. Both have extensively been used in 2D GIS applications, but their extension to 3D is not trivial.

A generalised model independent of dimension was originally proposed by Edmonds [1960], which later evolved into the data structure known as combinatorial maps (C-Maps) and defined by Vince [1983]. C-Maps by themselves only store abstract information regarding incidence and adjacency between the cells that it describes, but when they are combined with coordinate information on their vertices, they can represent any subdivision of 3D space with geometry. Those geometrically enhanced C-Maps are called linear cell complexes (LCCs). Feng et al. [2013] studied the concept of storing LCCs by proposing a solution for compact storage of C-Map for mesh data in 3D space. Damiand and Teillaud [2014] implemented a solution for storage and manipulation of dimension-independent data through LCCs.

LCCs have been used in the context of 3D city models and their applications. Horna et al. [2015] explore such an approach in modelling and simulation studies, where he describes the advantages of topological representations for 3D buildings. Diakité et al. [2015] have proposed EBM-LCC, a specific implementation of an LCC where attributes are used in order to describe 3D buildings which derive from BIM objects and LoD2 CityGML datasets. They also develop a method for creating EBM-LCCs from the topological reconstruction of buildings in order to automatically extract different LoDs from one main city object [Diakité et al., 2014].

4.2 Creation of a topological 4D city model

Based on a series of 3D city models representing a city at different moments in time and which are all loaded into a 3D topological data structure, it is in theory possible to create a 4D city model that incorporates them all into one. However, it is worth noting that the methods to do so in practice are still in development.

For instance, it is possible to take a 3D city model that exists at one point in time and then extrude it to 4D using the method described in Arroyo Ohori et al. [2015b]. This is the easiest method to load existing 2D or 3D data into a higher-dimensional structure, representing a set of cells that exist along a given dimension, such as a length of time or a range of scales. Unlike with other methods, it is also easy to guarantee that the output cell complex is valid and can be used as a base for further operations, such as dimension-independent generalisation algorithms. After such an extruded model is created, it is possible to perform some operations in order to represent different situations, such as those expressed by transformations (ie translation, scaling and rotation) and the collapse of objects [Arroyo Ohori et al., 2016].

Another possibility is based on the Jordan-Brouwer separation theorem [Lebesgue, 1911, Brouwer, 1911], since we know that a 4D object can be described based on a set of its bounding 3D objects. Since individual 3D objects are easier to describe than the 4D object, this can be used to subdivide a complex representation problem into a set of simpler, more intuitive ones [Arroyo Ohori et al., 2014]. Finally, one more possibility is to link the series of 3D city models by: identifying corresponding elements in different LoDs, deciding how these should be connected according to a linking scheme, and finally linking relevant 3-cells into 4-cells. Different linking schemes yield 4D models having different properties, such as objects that suddenly appear and disappear, gradually change in size or morph into different objects along the fourth dimension [Arroyo Ohori et al., 2015a].

4.3 Generalisation of 3D city models

3D city models at lower levels of detail can be derived from models of a higher LoD, this is done through a process known as generalisation. The generalisation is motivated by the need to reduce the size and semantic complexity of a model to a level at which it can be utilised within a specific application while avoiding the loss of relevant information [Guercke et al., 2009]. Not all applications require the highest level of detail, but rather data needs are task-specific and data volume dependent [Baig and Rahman, 2012]. Furthermore, there are often errors present in datasets that can occur due to different modelling software, different workflows used to produce the models and different approaches to quality assurance [Biljecki et al., 2016a]. Generalisation can therefore produce geometrically-valid models that are tailored for usage within a specific application. Generalisation considers the geometry and/or semantics while targeting various city objects within a 3D city model. It is an iterative process and there is no one perfect output but rather various possibilities for a data user to experiment with.

Currently, due to the increasing availability of LoD2 models, there has been a higher focus on generalising from LoD2 to LoD1 (see Figure 3). Generalisation of buildings from LoD2 to LoD1 can be accomplished by focusing on elements such as the floor plan and geometric reference. Simplifying the floor plan can be achieved by reducing the number of vertices through the decomposition of space along the major planes [Kada, 2008] or the Douglas-Peucker approach which splits down line segments in a polygon until they are within a specific tolerance [Douglas and Peucker, 1973]. The geometric reference of a building can be defined as the boundaries of the captured feature determined for a specific model [Biljecki et al., 2016c]. The vertical reference of a building at LoD1 can be set as either

Figure 3: An example of generalisation from LoD2 (right) to LoD1 (left) on a subset of buildings in Montréal, Canada. Data courtesy of Portail Données Ouvertes of the city of Montréal, visualised with azul.

the minimum, maximum, mean, median, mode, percentage or percentile based on the input roof values [Labetski et al., 2017]. Further, LoD1 models can either be generated by extruding from a generalised floor plan to its vertical reference, or the roof perimeter can instead be prioritised and "downtrusion" can be employed to generate the building.

Harmonisation between city objects in the generalisation process is an important consideration because topological errors can occur if city objects are generalised without the consideration of surrounding objects. If generalisation is applied without taking the surroundings into account, errors can occur. Examples of these are neighbouring buildings with overlaps, the loss of geographic relationships, or a misalignment with the underlying terrain. In CityGML the *Terrain Intersection Curve* (TIC) denotes the exact position where the terrain touches a 3D object and is meant to ease the integration of city objects within the terrain [OGC, 2012]. If generalisation does not take into account the relations between the 3D buildings and the terrain, the TIC can become inconsistent.

5 Visualisation of 3D and 4D city models

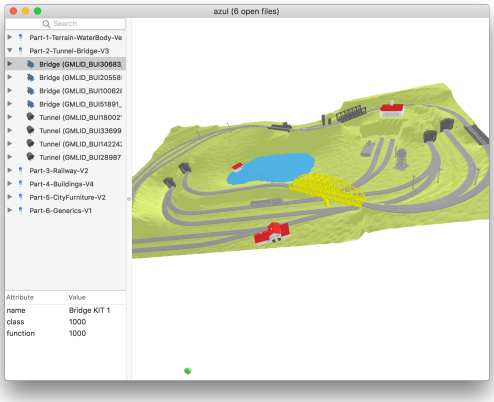
5.1 3D

When 3D city models are stored in standard 3D visualisation formats, they can be visualised in one of the many 3D viewers and 3D modellers that are readily available. While the exact features differ, there is a wide range of software available for this purpose. For instance, good choices include the mesh viewer and editor MeshLab[12] or the 3D modeller and renderer Blender[13], both of which are free, open source and available in multiple platforms. In a web interface, these models can also be viewed with the help of libraries like three.js[14] and Cesium[15].

When 3D city models are instead stored in CityGML or other semantic 3D city model formats, the options are more limited. However, such models can be viewed in Windows with the FZKViewer[16] of the Elyx 3D Viewer[17], in Mac with azul[18] (Figure 4),

[12] http://www.meshlab.net
[13] https://www.blender.org
[14] https://threejs.org
[15] https://cesiumjs.org
[16] https://www.iai.kit.edu/1302.php
[17] https://1spatial.com/products/elyx/elyx-gis-platform/elyx-3d/
[18] https://itunes.apple.com/nl/app/azul/

(a) Ettenheim



(b) CityGML 2.0 test dataset

Figure 4: Two 3D city models visualised in azul. Different colours represent different semantics.

or with cross-platform FME[19]. A list of software with CityGML support is available at `https://www.citygml.org/software/`.

5.2 4D

Since 4D city models are still in their infancy, their visualisation is not a topic that has been studied much so far. Most visualisations are simply animations that show 3D city models evolving in time.

However, there is also a significant amount of work on the visualisation of general 4D objects, which can then equally serve to visualise 4D city models [Arroyo Ohori et al., 2017b]. This includes early work using visual metaphors of 4D space, such as Flatland: A Romance of Many Dimensions [Abbott, 1884] and A New Era of Thought [Hinton, 1888]

More recently, Beshers and Feiner [1988] describe a system that displays animating 4D objects that are rendered in real-time and use colour intensity to provide a visual cue for the 4D depth. Banks [1992] describes a system that manipulates surfaces in 4D space, including interaction techniques and methods to deal with intersections, transparency and the silhouettes of every surface. Hanson and Cross [1993] describes a high-speed method to render surfaces in 4D space with shading using a 4D light and occlusion, and Chu et al. [2009] also describe a system to visualise 2-manifolds and 3-manifolds embedded in 4D space and illuminated by 4D light sources. Notably, it uses a custom rendering pipeline that projects tetrahedra in 4D to volumetric images in 3D—analogous to how triangles in 3D that are usually projected to 2D images. Arroyo Ohori et al. [2017b] shows how a 4D model can be viewed in its entirety by projecting it to 3D using a variety methods (Figure 5).

A different possible approach lies in using meaningful 3D cross-sections of a 4D dataset. For instance, Kageyama [2016] describes how to visualise 4D objects as a set of hyperplane slices. Bhaniramka et al. [2000] describe how to compute isosurfaces in dimensions higher than three using an algorithm similar to marching cubes. D'Zmura et al. [2000] describe a system that displays 3D cross-sections of a 4D virtual world one at a time.

6 Conclusions

In the near future, it is inevitable that we will have more and more 3D city models representing cities at different levels of detail, at different periods of time, and for different applications. It is therefore important to have adequate ways to store such his-

`id1173239678?mt=12`

[19] `http://www.safe.com/`

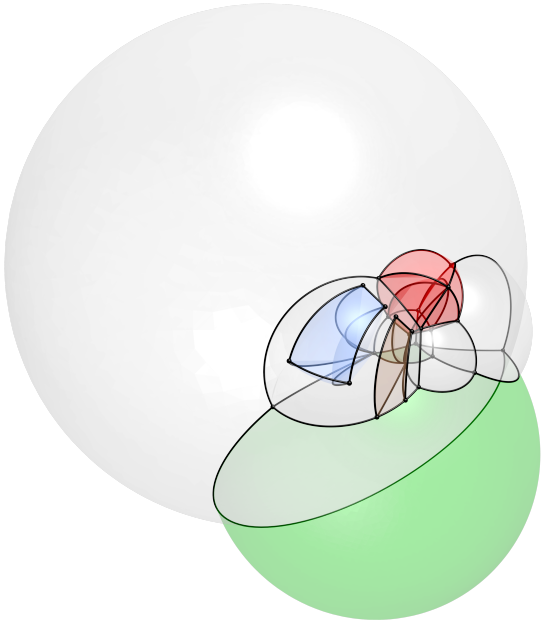


Figure 5: The model of a 4D house projected to 3D by first projecting inwards/outwards to the 3-sphere S^3, then stereographically to 3D.

torical collections of 3D city models in a manner that is both standardised and structured with semantics.

Within this chapter, we have looked at how this relates to visualisation formats (eg OBJ and glTF), 3D city modelling formats (eg CityGML, CityJSON, and InfraGML), and BIM formats (eg IFC). In addition, we have explained how such formats can be extended in the context of *4D city modelling*, where time or the level of detail are modelled as an additional dimension to the three spatial ones.

There are still many challenges to be solved before 4D city modelling becomes widespread. Some issues include: using appropriate 4D representations, properly integrating different 3D city models, the reconstruction of topology in 3D before doing so in 4D, dealing with the many geometric errors usually present in 3D models [Biljecki et al., 2016c], and the automated generalisation of 3D city models. Visualising and querying 4D city models is also undoubtedly a challenge.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677312 UMnD).

References

Edwin A. Abbott. *Flatland: A Romance of Many Dimensions*. Seely & Co., 1884.

Giorgio Agugiaro, Joachim Benner, Piergiorgio Cipriano, and Romain Nouvel. The energy application domain extension for CityGML: Enhancing interoperability for urban energy simulations. *Open Geospatial Data, Software and Standards*, 3(1), 2018.

Cici Alexander, Sarah Smith-Voysey, Claire Jarvis, and Kevin Tansey. Integrating building footprints and LiDAR elevation data to classify roof structures and visualise buildings. *Computers, Environment and Urban Systems*, 33(4):285–292, 2009.

K. Arroyo Ohori, F. Biljecki, A. Diakité, T. Krijnen, H. Ledoux, and J. Stoter. Towards an integration of GIS and BIM data: What are the geometric and topological issues? In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-4/W5, pages 1–8, 2017a. doi: http://dx.doi.org/10.5194/isprs-annals-IV-4-W5-1-2017.

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

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

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

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

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

Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. Visualising higher-dimensional space-time and space-scale objects as projections to R3. *PeerJ Computer Science*, jul 2017b. doi: http://dx.doi.org/10.7717/peerj-cs.123. ISSN: 2376-5992.

Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. Modelling and manipulating spacetime objects in a true 4D model. *Journal of Spatial Information Science*, 14: 61–93, 2017c.

Marco Attene, Marcel Campen, and Leif Kobbelt. Polygon mesh repairing: An application perspectice. *ACM Computing Surveys*, 45(2), 2013. article 15.

Siddique Ullah Baig and Alias Abdul Rahman. Generalization and visualization of 3D building models in CityGML. *Lecture Notes in Geoinformation and Cartography*, pages 63–77, oct 2012. doi: 10.1007/978-3-642-29793-9_4. URL http://dx.doi.org/10.1007/978-3-642-29793-9_4.

David Banks. Interactive manipulation and display of surfaces in four dimensions. In *I3D '92 Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 197–207. ACM, 1992.

Clifford M. Beshers and Steven K. Feiner. Real-time 4D animation on a 3D graphics workstation. In *Graphics Interface '88*, pages 1–7. CHCCS/SCDHM, 1988.

Praveen Bhaniramka, Rephael Wenger, and Roger Crawfis. Isosurfacing in higher dimensions. In *VIS '00 Proceedings of the conference on Visualization '00*. IEEE, 2000.

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.

Filip Biljecki, Hugo Ledoux, and Jantien Stoter. Redefining the level of detail for 3D models. *GIM International*, 28(11):21–23, 2014a. URL http://www.gim-international.com/issues/articles/id2167-Redefining_the_Level_of_Detail_for_D_Models.html.

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, 2014b.

Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu Çöltekin. Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, 2015.

Filip Biljecki, Hugo Ledoux, Xin Du, Jantien Stoter, K. H. Soon, and V. H. S. Koon. The most common geometric and semantic errors in citygml datasets. In *Proceedings 3D Geoinfo 2016*, 2016a. In Press.

Filip Biljecki, Hugo Ledoux, and Jantien Stoter. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37, 2016b.

Filip Biljecki, Hugo Ledoux, Jantien Stoter, and George Vosselman. The variants of an LOD of a 3D building model and their influence on spatial analyses. *ISPRS Journal of Photogrammetry and Remote Sensing*, 116:42–54, June 2016c. doi: http://doi.org/10.1016/j.isprsjprs.2016.03.003.

URL http://dx.doi.org/10.1016/j.isprsjprs.2016.03.003.

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

L.E.J. Brouwer. Beweis des Jordanschen Satzes für den *n*-dimensionalen Raum. *Mathematische Annalen*, 71:314–319, 1911.

Marcel Bruse, Romain Nouvel, Parag Wate, Volker Kraut, and Volker Coors. An Energy-Related CityGML ADE and Its Application for Heating Demand Calculation. *International Journal of 3-D Information Modeling*, 4(3):59–77, 2015.

K. Chaturvedi and T. H. Kolbe. Integrating dynamic data and sensors with semantic 3D city models in the context of smart cities. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1:31–38, 2016.

Jinmu Choi and Jiyeong Lee. *3D GeoNetwork for Agent-based Building Evacuation Simulation*, pages 283–299. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-87395-2. doi: 10.1007/978-3-540-87395-2_18.

Alan Chu, Chi-Wing Fu, Andrew J. Hanson, and Pheng-Ann Heng. GL4D: A GPU-based architecture for interactive 4D visualization. In *IEEE Transactions on Visualization and Computer Graphics*, volume 15, pages 1587–1594. IEEE, 2009.

Guillaume Damiand and Monique Teillaud. A generic implementation of dD combinatorial maps in CGAL. *Procedia Engineering*, 82:46–58, 2014. doi: 10.1016/j.proeng.2014.10.372.

Abdoulaye A. Diakité, Guillaume Damiand, and Gilles Gesquière. Automatic semantic labelling of 3d buildings based on geometric and topological information. 2015.

Abdoulaye Abou Diakité, Guillaume Damiand, and Dirk Van Maercke. Topological Reconstruction of Complex 3D Buildings and Automatic Extraction of Levels of Detail. In Vincent Tourre Gonzalo Besuievsky, editor, *Eurographics Workshop on Urban Data Modelling and Visualisation*, pages 25–30, Strasbourg, France, April 2014. Eurographics Association. doi: 10.2312/udmv.20141074. URL https://hal.archives-ouvertes.fr/hal-01011376.

David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2): 112–122, dec 1973. doi: http://dx.doi.org/10.3138/FM57-6770-U75U-7727. URL https://doi.org/10.3138%2Ffm57-6770-u75u-7727.

Michael D'Zmura, Philippe Colantoni, and Gregory Seyranian. Virtual environments with four or more spatial dimensions. *Presence*, 9(6):616==631, 2000.

J.R. Edmonds. *A Combinatorial Representation for Oriented Polyhedral Surfaces*. 1960. URL https://books.google.nl/books?id=vo2ENwAACAAJ.

Mohamed El-Mekawy and Anders Östman. Semantic mapping: an ontology engineering method for integrating building models in IFC and CityGML. *Proceedings of the 3rd ISDE Digital Earth Summit*, pages 12–14, 2010.

Mohamed El-Mekawy, Anders Östman, and Khurram Shahzad. A unified building model for 3D urban GIS. *ISPRS International Journal of Geo-Information*, 1:120–145, 2012.

Claire Ellul and Julia Altenbuchner. Investigating approaches to improving rendering performance of 3d city models on mobile devices. *Geo-spatial Information Science*, 17(2):73–84, jan 2014. doi: 10.1080/10095020.2013.866620.

Xin Feng, Yuanzhen Wang, Yanlin Weng, and Yiying Tong. Compact combinatorial maps: A volume mesh data structure. *Graphical Models*, 75(3):149–156, may 2013. doi: 10.1016/j.gmod.2012.10.001.

R Guercke, C Brenner, and M Sester. Generalization of 3D City Models as a service. In

ISPRS Workshop on quality, scale and analysis aspects of city models, 2009.

Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Transactions on Graphics*, 4(2):74–123, apr 1985. doi: 10.1145/282918.282923.

Yanjun Guo, Mao Pan, Zhe Wang, Honggang Qu, and Xiangrong Lan. A spatial overlay analysis method for three-dimensional vector polyhedrons. In *2010 18th International Conference on Geoinformatics*. IEEE, jun 2010. doi: 10.1109/geoinformatics.2010.5567674.

Norbert Haala and Mathias Rothermel. Dense multi-stereo matching for high quality digital elevation models. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 2012(4):331–343, 2012.

Andrew J. Hanson and Robert A. Cross. Interactive visualization methods for four dimensions. In *VIS '93 Proceedings of the 4th conference on Visualization '93*, pages 196–203. ACM, 1993.

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.

Charles Howard Hinton. *A New Era of Thought*. Swan Sonnenschein & Co. Ltd., 1888.

Sébastien Horna, Guillaume Damiand, Abdoulaye Diakité, and Daniel Meneveaux. Combining Geometry, Topology and Semantics for Generic Building Description and Simulations. In Filip Biljecki and Vincent Tourre, editors, *Eurographics Workshop on Urban Data Modelling and Visualisation*. The Eurographics Association, 2015. ISBN 978-3-905674-80-4. doi: 10.2312/udmv.20151343.

ISO. ISO 19107:2003: Geographic information—Spatial schema. International Organization for Standardization, 2003.

ISO. *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries*. International Organization for Standardization, March 2013.

ISO. *Industrial automation systems and integration - Product data representation and exchange*. International Organization for Standardization, August 2014.

Martin Kada. Generalization of 3D building models for map-like presentations. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences: XXXVII.[S. l.]: ISPRS*, pages 399–404, 2008.

Akira Kageyama. A visualization method of four dimensional polytopes by oval display of parallel hyperplane slices. Available at `https://arxiv.org/pdf/1607.01102.pdf`, 2016.

Thomas P Kersten, Friedrich Keller, Jerome Saenger, and Jochen Schiewe. Automated generation of an historic 4d city model of hamburg and its visualisation with the ge engine. In *Euro-Mediterranean Conference*, pages 55–65. Springer, 2012.

M. J. Kraak. *Geovisualization and time : new opportunities for the space - time cube*, pages 293–306. Wiley & Sons, 2008.

K. Kumar. Integration of CityGML & InfraGML. OGC Technical and Planning Committee Meeting 2017, Delft, The Netherlands., 2017. `https://3d.bk.tudelft.nl/kavisha/pdf/Kavisha_OGC_LandinfraDWG_March212017.pdf`.

Anna Labetski, Hugo Ledoux, and Jantien Stoter. Generalising 3D buildings from LoD2 to LoD1. In *GISRUK 2017 Conference Proceedings*, 2017. URL `http://huckg.is/gisruk2017/GISRUK_2017_paper_92.pdf`.

Robert Laurini and Françoise Milleret-Raffort. Topological reorganization of inconsistent geographical databases: A step towards their certification. *Computers & Graphics*, 18(6):803–813, 1994.

M. Lebesgue. Sur l'invariance du nombre de dimensions d'un espace et sur le theorème de M. Jordan relatif aux varieté fermées. *Comptes rendus de l'Académie des Sciences*, 152:841–844, 1911.

Hugo Ledoux. On the validation of solids represented with the international standards for geographic information. *Computer-Aided Civil and Infrastructure Engineering*, 28(9):693–706, 2013.

Hugo Ledoux. val3dity: validation of 3D GIS primitives according to the international standards. *Open Geospatial Data, Software and Standards*, 3(1):1, 2018.

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

Clément Mallet and Frédéric Bretar. Full-waveform topographic lidar: State-of-the-art. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(1):1–16, 2009.

D.E. Muller and F.P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7(2):217–236, 1978. doi: 10.1016/0304-3975(78)90051-8.

Romain Nouvel, Maryam Zirak, Volker Coors, and Ursula Eicker. The influence of data quality on urban heating demand modeling using 3D city models. *Computers, Environment and Urban Systems*, 64:68–80, 2017.

OGC. Geography markup language (GML) encoding standard. Open Geospatial Consortium inc., 2007. Document 07-036, version 3.2.1.

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

OGC. OGC city geography markup language (CityGML) encoding standard. Open Geospatial Consortium inc., 2012. Document 12-019, version 2.0.0.

OGC. Future City Pilot-1: Using IFC/CityGML in urban planning engineering r/eport. Open Geospatial Consortium inc., 2016a. Document OGC 16-097.

OGC. Future City Pilot-1: Using IFC/CityGML in urban planning engineering report. Open Geospatial Consortium inc., 2016b. http://docs.opengeospatial.org/per/16-097.html.

OGC. OGC Land and Infrastructure Conceptual Model Standard. OGC Document reference 15-111r1. Open Geospatial Consortium inc., 2016c. http://docs.opengeospatial.org/is/15-111r1/15-111r1.html.

OGC. OGC InfraGML 1.0: Part 0 - LandInfra Core - Encoding Standard. OGC Document reference 16-100r2. Open Geospatial Consortium inc., 2017. http://www.opengis.net/doc/standard/infragml/part0/1.0.

Sander Oude Elberink. *Acquisition of 3D topgraphy: automated 3D road and building reconstruction using airborne laser scanner data and topographic map*. PhD thesis, University of Twente Faculty of Geo-Information and Earth Observation (ITC), 2010.

Franz Rottensteiner. Automatic generation of high-quality building models from lidar data. *IEEE Computer Graphics and Applications*, 23(6):42–50, 2003.

Muhammad Shahzad and Xiao Xiang Zhu. Robust reconstruction of building facades for large areas using spaceborne TomoSAR point clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 53(2):752–769, 2015.

Horst Steuer, Thomas Machl, Maximilian Sindram, Lukas Liebel, and Thomas H. Kolbe. *Voluminator—approximating the volume of 3D buildings to overcome topological errors*, pages 343–362. Lecture Notes in Geoinformation and Cartography. Springer Science, 2015.

Linda van den Brink, Jantien Stoter, and Sisi Zlatanova. UML-based approach to developing a CityGML application domain

extension. *Transactions in GIS*, 17(6):920–942, 2013.

Andrew Vince. Combinatorial maps. *Journal of Combinatorial Theory, Series B*, 34(1):1–21, feb 1983. doi: 10.1016/0095-8956(83)90002-3.

Sisi Zlatanova. On 3d topological relationships. In *Database and Expert Systems Applications, 2000. Proceedings. 11th International Workshop On*, pages 913–919. IEEE, 2000. doi: 10.1109/DEXA.2000.875135.