Automatic semantic-preserving conversion between OBJ and CityGML

Filip Biljecki Ken Arroyo Ohori

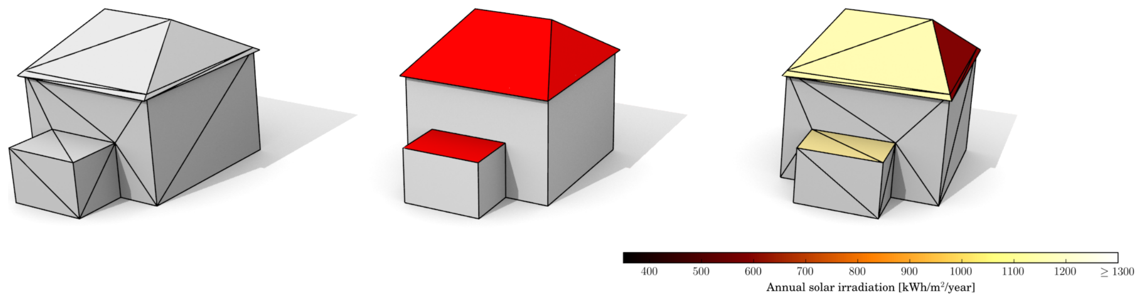

Figure 1: An OBJ dataset (triangles without attributes; left) is converted to CityGML (semantically structured polygons; middle), allowing it to be used to compute the insolation of the roof surfaces. The right model shows the conversion back to OBJ with the insolation value preserved as a material, and an automatically generated colourbar.

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

Automatic semantic-preserving conversion between OBJ and CityGML. Filip Biljecki and Ken Arroyo Ohori. In F. Biljecki and V. Tourre (eds.), *Eurographics Workshop on Urban Data Modelling and Visualisation*, Eurographics Association, Delft, The Netherlands, November 2015, pp. 25–30. ISBN: 978-3-905674-80-4 ISSN: 2307-8251.
DOI: 10.2312/udmv.20151345

We investigate the automatic conversion between two substantially different formats used in 3D city models: the ubiquitous but semantically poor Wavefront OBJ and the semantically rich but less used OGC standard CityGML. We elaborate on their differences and on the challenges involved in their conversion, such as the inference of semantics in an OBJ file for their use in CityGML, and the storage of these semantics back in OBJ. We implement two software prototypes: a conversion of 3D building models from CityGML to OBJ (CityGML2OBJs), and one from OBJ to CityGML (OBJ2CityGML). By presenting both methods and implementations, we aim at increasing the availability of CityGML datasets and the possibility to create them in powerful 3D modelling software.

1 Introduction

3D city models may be stored in a multitude of different formats, which limits their interoperability. Their conversion is thus a topical subject in 3D geoinformation science Stadler et al. [2009]; Donkers et al. [2015]; Zlatanova et al. [2012].

In this paper we show how to convert between two prominent formats used for storing 3D city models: OBJ, which is characteristic of 3D modelling and computer graphics, and CityGML, characteristic for 3D GIS, which has more powerful methods to store attributes and georeferenced datasets. By connecting the two, we also give our contribution in bridging the two disciplines.

OBJ is an ubiquitous 3D format that has wide software support in 3D modelling and 3D visualisation software. However, many GIS software packages do not support OBJ, or require certain semantics that are provided in CityGML, so it would be beneficial to convert them to be able to perform spatial analyses. On the other hand, the conversion from CityGML to OBJ is beneficial to take advantage of the high degree of support of OBJ in 3D modelling software.

The conversion from CityGML to OBJ is not difficult, but it generally entails a loss of data. We introduce a few mechanisms to mitigate it. The conversion of OBJ to CityGML requires inferring different concepts to match the structural level of CityGML.

We develop a method and implement it in two software prototypes to enhance the exchange and storage of these two formats. In this work we focus on 3D building models, however, the concepts can be applied to other types of models with minor adaptations.

2 Background and related work

Kavisha et al. [2015] investigate the conversion from COLLADA to CityGML. They focus on the conversion of texture and geometry. Diakité et al. [2014]; Boeters et al. [2015] develop methods to extract semantics from 3D buildings stored without them (as is done in OBJ), based on geometric and topological information. Donkers et al. [2015]; El-Mekawy et al. [2012] demonstrate the semantic mapping between IFC and CityGML. Scianna [2013] investigates the use of computer graphics software to handle 3D GIS data for GIS purposes, and concludes that there is potentially a high degree of synergy. Zhang et al. [2014] raises the importance of semantics in computer graphics visualisation.

3 Overview of data models

3.1 OBJ

OBJ (Object file) is a geometry definition file format developed in 1980s by Wavefront Technologies for one of their products. Since then it has gained prominence and has achieved wide support, becoming the most widespread format in 3D modelling and visualisation software. Because in practice many 3D city models are stored in OBJ, and the format has been used in the GIS community (e.g. see Arroyo Ohori et al. [2015]; Dimopoulou et al. [2014]; Rumor and Roccatello [2009]; Frommholz et al. [2015]), OBJ can also be considered as a 3D GIS format.

The OBJ standard is powerful in theory but its support is limited in practice. For instance, OBJ files support complex geometries including Bézier, B-spline, Cardinal and Taylor surfaces. However, most software that reads OBJ files only supports triangles or polygons.

3.2 CityGML

The Open Geospatial Consortium standard CityGML Gröger and Plümer [2012]; Open Geospatial Consortium [2012b], is an XML-based format for storing 3D city models that are structured into semantically meaningful parts. It is a format that is intended for spatial analyses, such as estimating the energy demand needed for heating Strzalka et al. [2011]. On the other hand, CityGML is also gaining adoption for visualisation purposes Gesquière and Manin [2012]; Evans et al. [2014].

CityGML provides thematic classes to store different classes of objects, e.g. buildings, tunnels, and bodies of water. Further, the data model supports assigning a semantic theme for each surface. In case of the exterior of buildings, there are classes such as `GroundSurface`—essentially the footprint of a building, and `RoofSurface`, which denotes the surfaces representing the roof. This standardised semantic information is advantageous for many use cases, for instance, in estimating the solar irradiation of rooftops, where only geometries stored as `RoofSurface` are considered in the analysis Biljecki et al. [2015a].

CityGML provides five standard LODs: LOD0 is a 2D footprint, LOD1 is a block model obtained with extrusion, LOD2 is an upgrade of the former with simple roof structures and semantically enriched boundary surfaces, LOD3 are architecturally detailed models with fenestration, and LOD4 contains interior Kolbe [2009]. The LOD concept in CityGML is different from the one in computer graphics since it denotes the model's spatio-semantic adherence to its real-world counterpart Biljecki et al. [2014].

A significant disadvantage of CityGML is that currently it is not as widely adopted as OBJ, and that its storage footprint is considerably larger.

3.3 Comparison and challenges

A comparison of the two formats is given in Table 1. We elaborate here the main differences that are relevant when it comes to their conversion:

1. OBJ datasets usually come as sets of polygons or triangles, while CityGML geometry is based on the types defined in the Geometry Markup Language (GML) Open Geospatial Consortium [2012a], supporting polygons with holes, solids, and other more advanced concepts. Consequently, one CityGML surface is usually represented by multiple faces (triangles) in OBJ.
2. CityGML supports thematically differentiated objects and surfaces, in contrast to OBJ where there is no standard way to differentiate them based on attributes.
3. Attribute support is limited in OBJ, while in CityGML several can be assigned to objects or to their parts.

Table 1: Comparison of OBJ and CityGML.

	Formats	
Concept	OBJ	CityGML
Audience	3D modelling	GIS
Format	Plain text	XML
Geometry	Polygons	GML
Semantics	Weak	Strong
Georeferenced	Partial	Yes
Adoption	High	Low
Storage footprint	Medium	Large

4 Conversion from CityGML to OBJ (CityGML2OBJs)

The conversion of CityGML to OBJ is not difficult, but it entails loss of information since it is not possible to preserve several concepts presented in Section 3.3. However, we mitigate this and partially preserve information in several solutions which we introduce below, and implement in a software prototype CityGML2OBJs.

- OBJ cannot differentiate between different buildings as CityGML, so in our approach geometries are segregated in objects (`o` notion in the OBJ) with the same ID of the original CityGML `<gml:id>` of an object.
- The software separates surfaces according to their CityGML semantic class. However, OBJ has limited support for semantics. We hence store groups of

equivalent semantic surfaces in separate OBJ files. This is reflected in the plural 'OBJs' in the name of the package, as a CityGML dataset is converted to multiple OBJs. E.g. the conversion of Delft.gml results in Delft-WallSurface.obj, Delft-RoofSurface.obj, etc.

- OBJ has limited support for attributes. As a solution to preserve attributes we have decided to take advantage of OBJ materials. Therefore, quantitative attributes of buildings (e.g. year of construction) are converted to a material of a specific colour assigned to all faces of the building. The colour is assigned according to a colourmap that is generated based on the range of values found in the dataset. This process is aided by matplotlib Hunter [2007], and a reference colourmap is rendered and stored separately.
- OBJ is not a geospatial format, so a geodetic datum is generally not stored together with the data. In addition, 3D modelling, rendering and CAD software generally uses local coordinate systems, using small numbers around the origin. For this reason, during the conversion we find the centroid of the geometry, and translate the dataset so that the centroid corresponds to the origin of a local coordinate system centred at $(0, 0)$.

The process of the conversion is briefly described in the continuation. The objects in CityGML are parsed and considered separately. A validator of geometry has been built to check the polygons for validity according to relevant applicable criteria defined in the standard ISO 19107:2003(E) ISO [2003] and Ledoux [2013], such as the planarity of polygons. The polygons of each building are triangulated with Triangle Shewchuk [1996], and the resulting triangles are first stored in separate sets for each semantic class, and then as a group for each building. After the dataset was processed, the algorithm finds recurring vertices, re-indexes the faces to point to a unique point, and stores them only once to save space. Finally, the OBJ files are written.

Fig. 1 shows an example of the conversion. A CityGML dataset (middle), which was enriched with solar irradiation values (from the software Solar3Dcity Biljecki et al. [2015a]), is converted to OBJ (right). The colourmap, which is automatically generated in the process, is also shown.

5 Conversion from OBJ to CityGML (OBJ2CityGML)

Technically OBJ can be directly converted to CityGML (as a soup of non-classified and non-structured triangles), but that does not take advantage of the capabilities of the format, and in CityGML triangles are practically used only to store terrain. Here we attempt to infer the missing concepts from OBJ to match the structural level and conventional usage in CityGML, and have implemented a software prototype OBJ2CityGML. Our approach consists of the following steps: detecting the LOD of the OBJ dataset, detecting objects (connected components) and polygons (set of adjacent coplanar triangles), merging triangles into polygons, and inferring the semantic class of each polygon based on their normals.

5.1 Detection of the LOD

Research has been done to automatically detect the LOD of 2D (cartographic) datasets Touya and Reimer [2015], however, no equivalent work exists in 3D.

We have detected the LOD in the following way: if all surfaces in the dataset are found to be horizontal or vertical the dataset is considered as LOD1. If there are sloped roofs the dataset is either LOD2 or LOD3. However, since we are not aware of many LOD3-grade models stored in OBJ, we classify them as LOD2.

It is important to perform this step in the beginning, so in case the dataset is of an

LOD1 (which does not require semantically differentiated surfaces) the semantic classification can be skipped.

5.2 Two-step segmentation of triangles

5.2.1 Detection of objects

The first step is to group the triangles into those forming connected components, which represent individual buildings. This is a prerequisite for CityGML, in which structure the topmost node is `cityObjectMember`.

In order to do this, all edges of the triangles in the dataset are first indexed. Then, for each triangle, the algorithm detects the adjacent triangles by detecting the shared edges from the index. In a second pass, the algorithm, finds distinct groups of faces that do not share any edge. The outcome of this process is illustrated in Fig. 2.

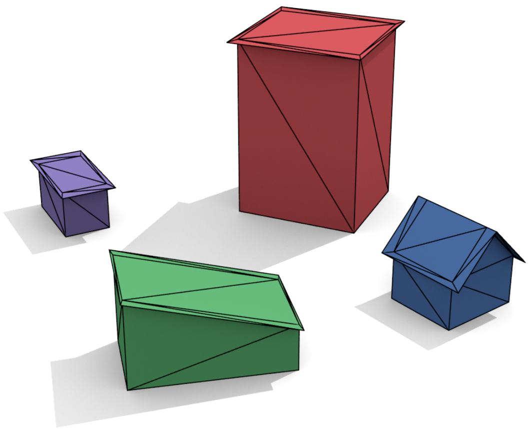
Figure 2: First step of the segmentation of triangles to find standalone city objects (here coloured differently).

5.2.2 Detection of polygons

In the second step, for the triangles of each building, a similar algorithm has been employed to find triangles that form a planar surface that can be merged into a polygon and written in GML. The main difference is that besides the shared edges, the algorithm tests the orientation of the normal of two adjacent triangles. If they are adjacent and have the same normal, it means that they form a planar surface (Fig. 3).

5.3 Construction of polygons from triangles

At this phase, the process has a list of triangles that form a planar surface, which has to be converted to a polygon (see Fig. 3). All edges that form the outer boundary of the surface are part of only one triangle, while those that are not are in more than one triangle. Hence, we find all such edges, and connect them in a ring that forms the boundary of the polygon. The sequence of the edges has been found by matching the start/end points of the set of edges, and the orientation of the resulting ring has been corrected according to the normal of the triangles.

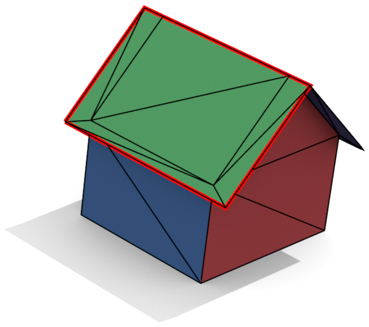
Figure 3: After adjacent planar triangles are found (differently coloured surfaces), they have to be merged into polygons. In this process we find the outer boundary (outlined in red for one of the polygons), and construct the polygon from the sequential edges.

5.4 Normal-based classification of semantics

The semantic segmentation and classification of urban features is a topical subject in computer science Martinović et al. [2015];

Xiong et al. [2013]. We follow the approach of Diakité et al. [2014] and Boeters et al. [2015], who have shown that the semantic class of a surface can be deduced from the orientation of its normal. For instance, a surface whose normal is horizontal is most likely a wall.

5.5 Storage as CityGML

In the last step, our structured geometry is written to CityGML. In the process, each building and polygon are assigned a unique identifier (UUID). The following example shows an excerpt of an OBJ representing a rectangular ground surface (two triangles forming a rectangle):

```
v 7.3257 5.6967 0.0
v 0.0 0.0 0.0
v -2.7931 3.5918 0.0
v 4.5326 9.2885 0.0
...
f 1 2 3
f 3 4 1
...
```

which is detected as a planar surface within a building of an LOD2 dataset, labelled as a ground surface, and finally converted to the geometry in GML and the structure of CityGML:

```
<cityObjectMember>
 <bldg:Building
 gml:id="...">
  ...
   <bldg:boundedBy>
    <bldg:GroundSurface>
     <bldg:lod2MultiSurface>
      <gml:MultiSurface>
       <gml:surfaceMember>
        <gml:Polygon
        gml:id="...">
         <gml:exterior>
          <gml:LinearRing>
           <gml:posList>
            0.0 0.0 0.0
            -2.7931 3.5918 0.0
            4.5326 9.2885 0.0
            7.3257 5.6967 0.0
            0.0 0.0 0.0
           </gml:posList>
          ...
```

Notice the increased storage footprint of CityGML in comparison to OBJ (cf. Biljecki et al. [2015b]).

6 Discussion

For testing the implementation we have used procedurally generated CityGML models from *Random3Dcity* Biljecki et al. [2015c]. An impression of the dataset is also shown in Fig. 2 and Fig. 5. The advantage is that this is one of the rare instances of multi-LOD datasets available, so the performance of data in multiple LODs can be tested. We have also used real-world data, a 3D model (LOD1) of the campus of the Delft University of Technology Ledoux and Meijers [2011], see Fig. 4.

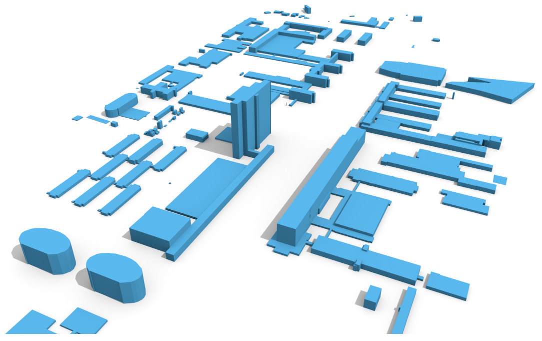

Figure 4: In this LOD1 example the level of detail was correctly inferred from the OBJ dataset, hence, in the conversion to CityGML the deduction of the thematic surfaces was skipped. OBJ dataset courtesy of Ledoux and Meijers [2011].

The results of the prototypes are valuable considering the simplicity of the approach. However, there are some limitations that present opportunities for future work. For instance, the normal-based approach for the classification of surfaces can only distinguish between a low number of classes—for example, it cannot differentiate between a roof, and a terrace or garage top (see Fig. 1). Advancing the method would require substantially more sophisticated approaches, probably reaching into the pattern recognition discipline. Another limitation is that the method does not detect polygons with interior geometry, or closed geometries such as solids.

7 Conclusions and future work

This paper presents an approach to convert a 3D city model stored in OBJ to CityGML, and vice versa. Allowing for the automatic conversion between these two quite different formats benefits practitioners from the best of both worlds—the ubiquitous support of OBJ in 3D modelling software, and the semantics and support of CityGML in GIS software. For instance, as a use case of the developed software, a CityGML dataset was prepared for 3D printing by converting it to OBJ (Fig. 5), which cannot be achieved directly. On the other hand, an OBJ can be converted to CityGML to run a spatial analysis such as the estimation of the insolation (see Fig. 1).

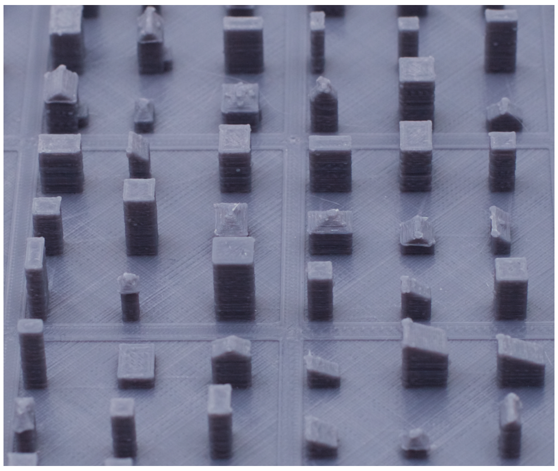

Figure 5: As a use case of CityGML2OBJs, we have converted a procedurally generated dataset in CityGML to OBJ for 3D printing purposes. Due to the limited software adoption, without such a tool CityGML datasets cannot reach their full potential. CityGML data source: Biljecki et al. [2015c].

There are many scientific and software opportunities in this area, and in this paper we have tackled only a part of the semantics that can be applied to a CityGML file. For future work we aim to detect more semantic classes within a city object and their labelling (e.g. as buildings, roads and trees). We also intend to preserve additional CityGML features that are not so easily handled (e.g. coordinate systems).

We also aim to handle the geometric errors that are commonly found in OBJ files Ledoux et al. [2014], as well as the automatic conversion of the more advanced geometry types supported by OBJ (e.g. Bézier surfaces) into CityGML.

Acknowledgements

We thank Marko Čubranić for 3D printing the OBJ dataset. We gratefully acknowledge the comments of the anonymous reviewers. This research is supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs (project code: 11300).

References

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

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

Filip Biljecki, Gerard B M Heuvelink, Hugo Ledoux, and Jantien Stoter. Propagation of positional error in 3D GIS: estimation of the solar irradiation of building roofs. *International Journal of Geographical Information Science*, pages 1–26, 2015a.

Filip Biljecki, Hugo Ledoux, and Jantien Stoter. Improving the consistency of multi-LOD CityGML datasets by removing redundancy. In *3D Geoinformation Science*, pages 1–17. 2015b.

Filip Biljecki, Hugo Ledoux, and Jantien Stoter. An improved LOD specification for 3D building models and its CityGML realisation with the Random3Dcity procedural modelling engine. *Computers, Environment and Urban Systems*, Under review, 2015c.

Roeland Boeters, Ken Arroyo Ohori, Filip Biljecki, and Sisi Zlatanova. Automatically enhancing CityGML LOD2 models with a corresponding indoor geometry. *International Journal of Geographical Information Science*, pages 1–21, August 2015.

Abdoulaye A Diakité, Guillaume Damiand, and Gilles Gesquière. Automatic Semantic Labelling of 3D Buildings Based on Geometric and Topological Information. In *Proceedings of the 9th 3DGeoInfo Conference 2014*, pages 49–63, November 2014.

Efi Dimopoulou, Eva Tsiliakou, Vasso Kosti, George Floros, and Tassos Labropoulos. Investigating integration possibilities between 3D modeling techniques. In *Proceedings of the 9th 3DGeoInfo Conference 2014*, pages 1–16, Dubai, UAE, November 2014.

Sjors Donkers, Hugo Ledoux, Junqiao Zhao, and Jantien Stoter. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 2015.

Mohamed El-Mekawy, Anders Östman, and Ihab Hijazi. An Evaluation of IFC-CityGML Unidirectional Conversion. *International Journal of Advanced Computer Science and Applications*, 3(5):159–171, 2012.

Alun Evans, Marco Romeo, Arash Bahrehmand, Javi Agenjo, and Josep Blat. 3D graphics on the web: A survey. *Computers and Graphics*, 41:43–61, June 2014.

D Frommholz, M Linkiewicz, H Meissner, D Dahlke, and A Poznanska. Extracting semantically annotated 3D building models with textures from oblique aerial imagery. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XL-3/W2:53–58, 2015.

Gilles Gesquière and Alexis Manin. 3D Visualization of Urban Data Based on CityGML with WebGL. *International Journal of 3-D Information Modeling*, 1(3):1–15, July 2012.

Gerhard Gröger and Lutz Plümer. CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33, July 2012.

John D Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

ISO. 19107:2003(E) – Geographic information - Spatial schema, May 2003.

Kavisha, S Saran, and A Senthil Kumar. CityGML based Interoperability for the transformation of 3D Data Models. *Transactions in GIS*, March 2015.

T H Kolbe. Representing and exchanging 3D city models with CityGML. In Sisi Zlatanova and Jiyeong Lee, editors, *3D Geo-Information Sciences*, pages 15–31. Springer Berlin Heidelberg, 2009.

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, October 2013.

Hugo Ledoux and Martijn Meijers. Topologically consistent 3D city models obtained by extrusion. *International Journal of Geographical Information Science*, 25(4):557–574, April 2011.

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

Anđelo Martinović, Jan Knopp, Hayko Riemenschneider, and Luc van Gool. 3D All The Way: Semantic Segmentation of Urban Scenes from Start to End in 3D. In *CVPR 2015: The 28th IEEE Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, Boston, United States, June 2015.

Open Geospatial Consortium. OGC Geography Markup Language (GML) — Extended schemas and encoding rules 3.3.0, 2012a.

Open Geospatial Consortium. OGC City Geography Markup Language (CityGML) Encoding Standard 2.0.0. Technical report, April 2012b.

M Rumor and E Roccatello. Design and development of a visualization tool for 3D geospatial data in CityGML format. In *Urban and Regional Data Management*, pages 31–37. CRC Press, 2009.

Andrea Scianna. Building 3D GIS data models using open source software. *Applied Geomatics*, 5(2):119–132, 2013.

Jonathan Richard Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Lecture Notes in Computer Science*, pages 203–222. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

Alexandra Stadler, Claus Nagel, Gerhard König, and Thomas H Kolbe. Making Interoperability Persistent: A 3D Geo Database Based on CityGML. In Jiyeong Lee and Sisi Zlatanova, editors, *3D Geo-Information Sciences. Lecture Notes in Geoinformation and Cartography*, pages 175–192. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

Aneta Strzalka, Jürgen Bogdahn, Volker Coors, and Ursula Eicker. 3D City modeling for urban scale heating energy demand forecasting. *HVAC&R Research*, 17 (4):526–539, 2011.

Guillaume Touya and Andreas Reimer. Inferring the Scale of OpenStreetMap Features. In Jamal Jokar Arsanjani, Alexander Zipf, Peter Mooney, and Marco Helbich, editors, *OpenStreetMap in GIScience. Lecture Notes in Geoinformation and Cartography*, pages 81-99. Springer International Publishing, March 2015.

Xuehan Xiong, Antonio Adan, Burcu Akinci, and Daniel Huber. Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, 31:325–337, May 2013.

Fan Zhang, Vincent Tourre, and Guillaume Moreau. Applying Level-of-detail and Perceptive Effects to 3D Urban Semantics Visualization. In *Proceedings of the Eurographics Workshop on Urban Data Modelling and Visualisation*, pages 31–36, Strasbourg, France, April 2014.

S Zlatanova, J Stoter, and U Isikdag. Standards for Exchange and Storage of 3D Information: Challenges and Opportunities for Emergency Response. In T Bandrova, M Konecny, and G Zhelezov, editors, *Proceedings of the th International Conference on Cartography GIS*, pages 17–28, Albena, Bulgaria, June 2012.