Topological Reconstruction of 3D City Models with preservation of semantics

Stelios Vitalis
3D geoinformation, TU Delft
Julianalaan 134
Delft, the Netherlands
s.vitalis@tudelft.nl

Ken Arroyo Ohori
3D geoinformation, TU Delft
Julianalaan 134
Delft, the Netherlands
g.a.k.arroyoohori@tudelft.nl

Jantien Stoter
3D geoinformation, TU Delft
Julianalaan 134
Delft, the Netherlands
j.e.stoter@tudelft.nl

Abstract

3D city models are becoming increasingly important for applications such as evacuation scenarios and energy consumption estimation. For these applications embedding semantic information on geometry is a key factor. The most popular implementation of modern 3D city models is based on the CityGML data model which describes spatial 3D data using a geometrical representation according to the GML encoding standard. While CityGML supports some basic storage of topological relationships between geometric objects, it fails to offer a true 3D topological representation of the city model. Alternatively, a true topological data structure can be used as an intermediate data model, to enable enforcing certain restrictions and operations that are more efficient for specific applications. In this article, we discuss a method that we have developed for the automatic conversion of CityGML models to a topological structure, while maintaining semantic information that was initially attached to the city objects. Such an approach raises certain challenges, as the geometries are not one-to-one analogous to the topological objects that are needed to represent them. We also provide a few examples that indicate that such a method is not trivial for retaining all information that was initially stored in a city model.

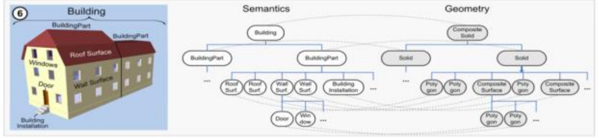
Keywords: 3D city models, CityGML, Combinatorial Maps, Linear Cell Complex, Topological Reconstruction

1 Introduction

3D city models are becoming increasingly popular in modern GIS applications, such as the simulation of evacuation scenarios (Choi & Lee, 2009) and energy consumption estimation (Kaden & Kolbe, 2014 and Zhivov, et al., 2017). Their main advantage is that they describe both geometric and semantic information of city objects, such as their purpose of use or their year of construction.

Currently, the most common data model for city models' representation is CityGML (Open Geospatial Consortium, 2012). It describes geometries through boundary representation (B-Rep) as it is defined by the Simple Feature Specification (SFS) (Open Geospatial Consortium, 2011). It also offers a way of storing semantic information such as the type and hierarchical relationships between objects, as well as additional attributes that can be assigned to them (Figure 1).

Figure 1: Example of semantic information stored on a city model to represent type and hierarchy of objects.



Source: Stander & Kolbe (2012)

While this polygonal representation of spatial data has been proven robust for 2D spatial data, there is still a lack of processing algorithm for 3D data that are described by polygonal modelling (Guo, et al., 2010). Meanwhile, topological data structures have been proven more efficient for describing 3D data for certain GIS applications, such as network analysis (Choi & Lee, 2009). Furthermore, simulations that are based on physical propagation, such as ray casting on architectural models (Maria et al., 2014), can be optimised by topological information to increase performance of spatial calculations through the adjacency and incidence relationships information.

For this reason, we investigate the use of ordered topological structures and, more specifically, combinatorial maps (C-Maps). Ordered topological structures combine the powerful algebra of geometric simplicial complexes, such as triangulated meshes. In addition, they are easy to construct like cell complexes which are based on a polygonal boundary representation (Arroyo Ohori, et al., 2015). There are also good software implementations of them, such as the C-Map package of CGAL (Damiand & Teillaud, 2014), which can be enhanced with geometric information in order to describe a linear cell complex (LCC).

LCCs based on C-Maps have been used before for the representation of city objects. Diakité, et al. (2014) propose a methodology for the topological reconstruction of an existing building from geometric B-Rep. He uses the adjacency and incidence information of the LCC to extract lower levels of detail (LoDs) of the initial model. The topological reconstruction is based on the conversion of all faces to a triangular soup in the C-Map and then merging them according to geometric rules such as their common edges. However, semantics is lost during the conversion, and therefore the soup does not retain semantic information that was stored in the initial model. Diakite, et al. (2015) also study a similar topological reconstruction process for BIM and GIS models to classify the containing features according to heuristic rules. While this reconstruction is applicable to CityGML models, it results in a continuous surface-based representation of the model, where the original city objects' subdivision and semantics are lost.

This paper describes a methodology for the topological reconstruction of a CityGML model to a linear cell complex based on combinatorial maps, while retaining semantic information. We propose two variations of the methodology: (a) a geometric-oriented one, that focuses on the geometric characteristics of the model and (b) a semantic-oriented one, which forces the initial subdivision of city objects to retain in the resulting topological structure.

2 Data Models

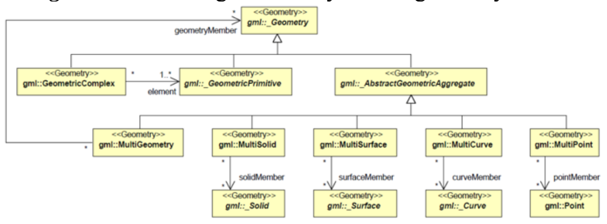
2.1 CityGML

CityGML is a data model that is based on the extension markup language (XML) in order to store and exchange data regarding virtual 3D city models (Open Geospatial Consortium, 2012). It describes feature classes through an object-oriented design approach, where flexibility is achieved through the use of a multi-level abstraction mechanism. That makes CityGML structures complex, which also led to high diversity of its implementation.

In CityGML, a city model contains a number of city objects of different types, all of which inherit from the abstract class *CityObject*. The derived classes can be: (a) composite objects, such as *CityObjectGroup*, (b) specialised abstract classes, such as *AbstractBuilding*, or (c) actual city objects, such as *CityFurniture* and *LandUse*. Given that a CityGML dataset follows a tree structure, the objects can be either listed as immediate child nodes in the model or they can be represented in a deeper layout, by grouping objects using *CityObjectGroup*.

CityGML follows the geometric representation of the geographic markup language (GML) (Open Geospatial Consortium, 2012), which is an implementation of the simple feature specification (SFS) (Open Geospatial Consortium, 2011). This allows for a certain degree of freedom regarding the composition of a 3D object. In a general form, every *CityObject* can contain one or more *Geometry* objects. The later follows the composite design pattern (meaning, that a *Geometry* can be a parent of other *Geometry* objects).

Figure 2: UML Diagram of CityGML's geometry model.



Source: (Open Geospatial Consortium, 2012).

CityGML stores semantic information by utilising two main concepts. First, the type of an object is implied through the class that implements it and its hierarchical position in the city model. Second, every *CityObject* may contain further information as attributes that are either defined by the CityGML specification or that the user of the model introduces to the model through either the *GenericAttribute* class or an Application Domain Extension. In a more simplified way, we can conclude that a *CityObject* contains a list of attributes which can be represented as a list of key-value pairs.

2.2 Combinatorial Maps

A combinatorial map (C-Map) is a data structure that can represent a partition of *n*-dimensional space to cell, such that a 0-cell is a vertex, a 1-cell is an edge, a 2-cell is a face and a 3-cell is a volume (Damiand & Lienhardt, 2014).

The basic element of a C-Map is called a *dart* and can be considered as the part of an edge that belongs to every combination of *i*-cells (for $0 < i \leq n$). Every dart contains links to other darts that are connected to it and that belong to a neighbouring *i*-cell. Those links are denoted as β_i (where $0 \leq i \leq n$) and every dart contains one β_i $\forall i \in \{1, \dots, n\}$. A β_i can be better understood as a link to the dart that is contained in the same combination of all cells except for the *i*-cell. For example, in a 3D C-Map a β_2 of the dart d_1 links to the dart that belongs to the same edge (1- cell) of the same volume (3-cell) as d_1, but is part of the neighbouring face (2-cell). In cases where a dart does not have a neighbouring *i*-cell, its β_i is assigned to the *null* dart (denoted as $\varnothing$) and we say that this dart is *i*-free.

C-Maps define a modification operation called *sewing*, which links pairs of corresponding darts of two *i*-cells. Intuitively, an *i*-sew operation will glue together two *i*-cells along their common $(i-1)$-cell. This is done by linking the β_i of every pair of corresponding darts along the two $(i-1)$-cells.

A C-Map can contain additional information related to the represented *i*-cells by attaching attributes for this dimension to an incident dart of the cell. Therefore, a dart holds a set of attributes as well, one for every dimension (called *i*-attribute).

Hence, we can use a C-Map data structure to represent an actual *n*-dimensional model while also storing topological relationships. These relationships are stored by attaching geometric information to the map as attributes. For representing linear geometries, we normally assign coordinates to every vertex (0-cell) of the C-Map. This is, then, a linear cell complex (LCC).

3 Methodology

3.1 Description of the method

As mentioned in Section 1, the purpose of the conversion is to topologically reconstruct a CityGML model while maintaining its original structure and semantics. The main objective is to construct a LCC where the information of the initial city objects will be stored as *i*-attributes.

In order to achieve this, it is required that darts are created on a C-Map while keeping track of those that are 2-free and 3-free in order to sew them respectively with newly created darts that describe adjacent edges. We have implemented this procedure in incremental steps per city object described by individual algorithms. Hence, every *i*-dimensional object is processed by the respective algorithm.

The conversion starts by iterating through the root city objects of a given CityGML model. Two indexes for 2-free and 3-free darts are maintained during this process. Every root city object is processed by another algorithm that appends the LCC and the indexes respectively.

The algorithm that processes the city object iterates through all of its immediate geometries and calls the *ReadGeometry* algorithm for every geometry. It also iterates all child city objects calling, recursively, itself in order to process them. After a polygon has been converted to the equivalent 2-cell, then the algorithm will attempt to 3-sew any 2-cells that share the same geometry.

ReadGeometry converts a geometry (polygon) to a 2-cell in the destination LCC. It loops through the edges that bound the polygon and calls *GetEdge* in order to construct and sew together the respective 1-cells that are needed in order to describe the 2-cell. After every edge is created, the algorithm will find 2-free darts that can be 2-sewed to the new ones and will apply the 2-sew operation.

The *GetEdge* algorithm creates a 1-cell based on two end points. A 1-cell needs two darts in order to be described, therefore *GetVertex* is used in order to either find a 1-free dart or to create a new one dart with the coordinates that are stored for every point.

```
Algorithm 3: ReadGeometry
  Input: linear cell complex $lcc$ where the new 2-cell will be
         created,
         index $I_2$ of 2-free darts on $lcc$,
         index $I_3$ of 3-free darts on $lcc$,
         geometry $poly$ whose coordinates will be the vertices
         of the new 2-cell
  Result: a new 2-cell is created on $lcc$ and $I_2$ and $I_3$ are updated
          accordingly
  Output: darts $D$ that were used for the creation of the 2-cell
1  $D \leftarrow \emptyset$;
2  foreach $v_{cur} \in poly$ except the last do
3      $v_{next} \leftarrow$ Get next vertex of $poly$;
4      $d_{new} =$ GetEdge($v_{cur}$, $v_{next}$, $I_2$);
5      push($D$, $d_{new}$);
6  foreach $d \in D$ do
7      if $\exists d_{op} \in I_3$ : reverse key of $d$ = key of $d_{op}$ then
8          Sew($d$, $d_{op}$, 3);
9          Remove $d_{op}$ from $I_3$;
10     else
11         Add $d$ to $I_3$;
12 return $D$;
```

```
Algorithm 2: GetEdge
  Input: linear cell complex $lcc$ where the new edge belongs
         index $I_2$ of 2-free darts on the $lcc$
         vertex $v_1$ that will be the starting point of the edge
         vertex $v_2$ that will be the ending point of the edge
  Output: dart $d_{new}$ that describes the edge in $lcc$
1  $d_{new} \leftarrow$ GetVertex($v_1$, 1);
2  $d_{next} \leftarrow$ GetVertex($v_2$, 0);
3  Sew($d_{new}$, $d_{next}$, 1);
4  if $\exists d_{op} \in I_2 : 0\text{-}attr(d_{op}) = v_2$ and $0\text{-}attr(\beta_1(d_{op}) = v_1)$ then
5      Sew($d_{new}$, $d_{op}$, 2);
6      Remove $d_{op}$ from $I_2$;
7  else
8      Add $d_{new}$ to $I_2$;
9  return $d_{new}$;
```

```
Algorithm 1: GetVertex
  Input: linear cell complex $lcc$ where the output dart belongs
         vertex $v$ to set of the output dart
         degree of freedom $i$ that the output dart must have
  Output: dart $d$ that belongs to $lcc$, have the vertex $v$ and is
          $i$-free
1  $D \leftarrow$ Get all darts of $lcc$;
2  foreach $d \in D$ do
3      if 0-attr($d$) $= v \wedge \beta_i(d) = \emptyset$ then
4          return $d$;
5  $d \leftarrow$ Create new dart of $lcc$;
6  0-attr($d$) $\leftarrow v$;
7  return $d$;
```

3.2 Variations of the process

The main algorithm has two variations: one where the index of 2-free darts is cleared after every root city object is finished and one where the 2-free darts remain available through the whole process. The first approach can be described as semantic-oriented, since it forces 2 polygons to be sewed only in case they belong to the same root city object. The second approach as geometric-oriented, as it allows for previously individual volumes to become one 3-cell in case they share a common edge across one of their bounded polygons.

3.3 Implementation

The described methodology has been implemented in a computer program using the C++ programming language[1]. For CityGML reading, we used the libcitygml library[2]. For the basic LCC representation we used the CGAL LCC package[3]. Visualisation was accomplished through a modified version of the LCC demo provided with the CGAL software package, which is based on the Qt5 graphical user interface and the libqglviewer component for viewing 3D graphics[4].

4 Results

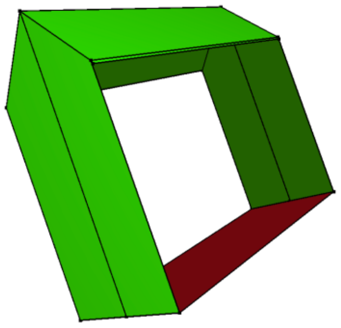
We applied the process described in Section 3 to the city model of *Agniesebuurt*, a neighbourhood of Rotterdam. This dataset is provided by the municipality as open data[5] in the form of CityGML files. In the data set, volumes (buildings) that share surfaces (walls) in the case of terraced houses, are not modelled properly. That is, the wall is only assigned to one of the two volumes. This makes it quite challenging to parse and process the data.

Initially, we applied the reconstruction to an individual building (Figure 3). The resulting LCC indeed shows the

[1] https://github.com/liberostelios/citygml2cmap
[2] https://github.com/jklimke/libcitygml
[3] http://doc.cgal.org/latest/Linear_cell_complex/index.html
[4] http://libqglviewer.com/
[5] https://www.rotterdam.nl/werken-leren/3d/

challenges of missing inner walls for terraced houses. First, the walls between neighbouring buildings are also missing in the LCC representation. Second, the geometry is topologiacally invalid, as the building has resulted in 2 volumes (3-cell).

Figure 3: An individual house from the Agniesebuurt dataset, after it has been topologically reconstructed as a LCC. This view highlights the absence of walls neighbouring with other buildings. It, also, shows how the topologically invalid surfaces that bound the house could not be "sewed" together, ending in different 3-cells. Every individual 3-cell is distinguished by different colours.



Finally, the two variations of the reconstruction were applied to the complete city model. Figure 4 demonstrates the difference between the two resulting LCCs, the geometric-oriented (LCC A) and semantic-oriented (LCC B). Due to the fact that neighbouring walls are missing, the geometrically-oriented reconstruction sewed together all adjacent buildings in one volume (3-cell), except for non-topologically valid objects. That, of course, causes all original semantic information to be lost, as only one original city object's attributes are retained in the final merged 3-cell.

The semantically-oriented reconstruction enforced the separation of individual buildings, which resembles better the original structure of the city model. That means, that no individual houses are sewed in one volume in the resulting LCC, therefore all information of the original model are preserved. Unfortunately, that also concluded in more non-topological surfaces being present in the final model, as some faces cannot to be sewed when geometric elements of the neighbouring building are not included in the 3-cell. This ends in redundant information as every one of those faces concludes to an individual 3-cell, which repeats the semantic information of the original city object.

This is also evident in Table 1, which presents statistical aspects of the resulting LCCs.

5 Discussion and conclusions

In this paper, we presented an approach for the conversion of city models from CityGML to a Linear Cell Complex while retaining the model's initial structure and semantics. For this, we have developed two variations of the topological reconstruction: (a) a geometric-oriented one, which creates volumes strictly according to their topological relation as calculated from geometry; and (b) a semantic-oriented one, which forces the volume creation to follow the initial city object structure as much as possible. We implemented this in a software program and applied it in the Agniesebuurt dataset from municipality of Rotterdam, in order to evaluate the significance of the semantics as part of the topological reconstruction. The two approaches have certain benefits and caveats.

Figure 4: Examples of the two different approaches for the topological reconstruction of the Agniesebuurt dataset. The geometric-oriented approach (top) ended up sewing all neighbouring terraced houses in one volume 3-cell (volume) due to missing in-between walls. The semantic-oriented approach (bottom) retains most of the original structure of the dataset, but also is more prone to topologically invalid surfaces, which caused one original building to be reconstructed as multiple 3-cells. Every individual 3-cell is distinguished by different colours.

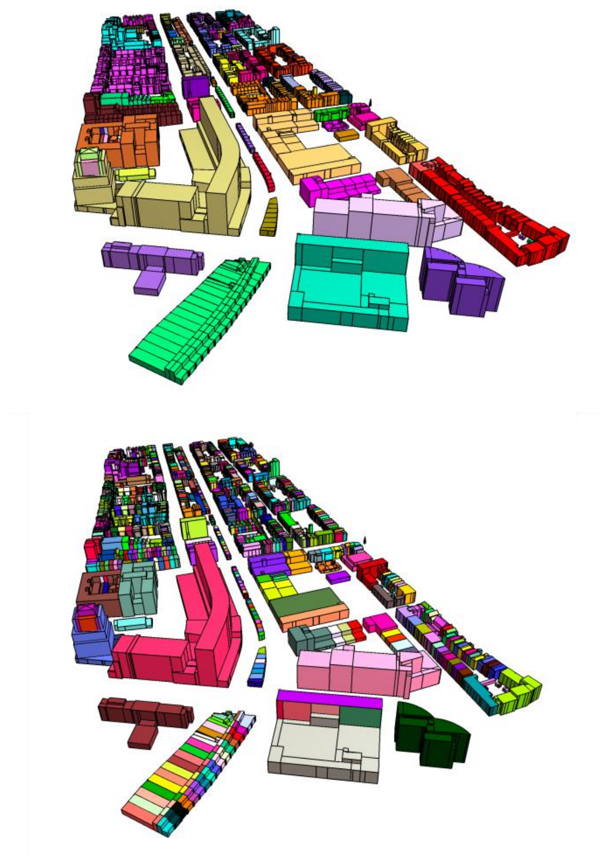


Table 1: The values that represent the size of the resulting linear cell complexes of Agniesebuurt (Rotterdam) dataset

		Number of				
Model	Size (MB)	Darts	0-cells	1-cells	2-cells	3-cells
LCC A	4.4	59188	22280	36710	12876	469
LCC B	4.7	59188	25771	38810	12876	1148

The geometric-oriented method can better highlight the true topological characteristics of the city model's geometry. This can be useful in highlighting errors in the dataset, for example the missing walls in the dataset that we tested. However, the problem is that the final structure might not reflect the original structure of the city model, for instance different buildings can be merged into one volume. This does not only alter the structure of the model, but it makes semantic preservation much more challenging.

The semantic-oriented method keeps the original structure of the city model as much as possible, which helps preserving semantics during the process. Nevertheless, there are issues of redundant semantics given that an original city object might be converted in multiple volumes in the final structure if its faces are not topologically-valid. In this case, semantic information of the original city object is repeated for all individual volumes, as there is no mechanism incorporated in LCC in order to preserve information once.

From our tests on the variations of the topological reconstruction we conclude that the storage of semantics strictly through *i*-attributes on LCC might be inefficient for the storage of geometry and semantics of a city model. Therefore, we are interested in a future exploration of a more traditional, hybrid object-oriented city model data structure that might be more appropriate for this purpose. This hybrid approach could be implemented as a data model similar to CityGML, where instead of the original SFS representation a LCC can be used in order to store geometric information.

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

Arroyo Ohori, K., Ledoux, H. & Stoter, J., 2015. An evaluation and classification of nD topological data structures for the representation of objects in a higher-dimensional GIS. *International Journal of Geographical Information Science,* 2, Volume 29, pp. 825-849.

Choi, J. & Lee, J., 2009. 3D Geo-Network for Agent-based Building Evacuation Simulation. In: J. Lee & S. Zlatanova, eds. *3D Geo-Information Sciences.* Berlin(Heidelberg): Springer Berlin Heidelberg, pp. 283-299.

Damiand, G. & Lienhardt, P., 2014. Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing. 1st ed. Natick(MA): A. K. Peters, Ltd.

Diakite, A. A., Damiand, G. & Gesquie`re, G., 2015. Automatic Semantic Labelling of 3D Buildings Based on Geometric and Topological Information. s.l., s.n.

Diakité, A. A., Damiand, G. & Van Maercke, D., 2014. *Topological Reconstruction of Complex 3D Buildings and Automatic Extraction of Levels of Detail.* Strasbourg, Eurographics Association, pp. 25-30.

Guo, Y., Pan, M., Wang, Z., Qu H. and Lan, X., 2010, *A spatial overlay analysis method for three-dimensional vector polyhedrons*. Beijing, 18th International Conference on Geoinformatics, pp. 1-5.

Kaden, R. & Kolbe, T. H., 2014. Simulation-Based Total Energy Demand Estimation of Buildings using Semantic 3D City Models. *International Journal of 3-D Information Modeling,* 4, Volume 3, pp. 35-53.

Maria, M., Horna, S. and Aveneau, L., 2014. Topological Space Partition for Fast Ray Tracing in Architectural Models, *GRAPP 2014 - 9th International Joint Conference on Computer Graphics Theory and Applications,* 225 - 235

Open Geospatial Consortium, 2011. *OpenGIS Implementation Standard for Geographic information - Simple feature access.* 1.2.1 ed. s.l.:s.n.

Open Geospatial Consortium, 2012. *City Geography Markup Language (CityGML) Encoding Standard, version: 2.0.0.* s.l.:s.n.

Open Geospatial Consortium, 2012. *OpenGIS Geography Markup Language (GML) Encoding Specification, version: 3.1.1.* s.l.:s.n.

Stadler, Alexandra & Kolbe, Thomas. (2012). Spatio-semantic coherence in the integration of 3D city models.

Zhivov, A. M. et al., 2017. Planning Tools to Simulate and Optimize Neighborhood Energy Systems. In: *Green Defense Technology.* s.l.:Springer.

Zlatanova, S., 2000. *On 3D topological relationships.* s.l., s.n., pp. 913-919.