Automatic repair of CityGML buildings in LOD 1-3 using a voxel-based method

MSc Geomatics Thesis Proposal

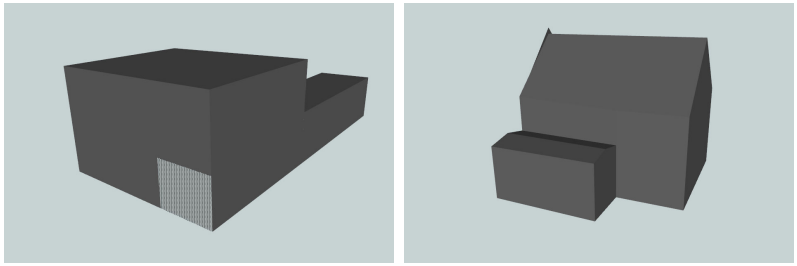
D.T. Mulder

January 20, 2015

1 Introduction

3D city models are gaining in importance as more 3D geometry of urban areas is becoming available. Real world entities such as terrain, buildings, infrastructure, vegetation and city furniture can be represented, making 3D city models usable for many applications like urban planning, spatial analysis, disaster management (Kolbe et al., 2008), augmented reality (Zamyadi et al., 2013) and navigation systems. To enable the widespread use and reuse of 3D city models, a common format for storage and exchange is needed. For this reasons the Open Geospatial Consortium (OGC) has developed CityGML as an open standard with the goal of creating a common definition for the objects, attributes and relations of a 3D city model. Based on a set of criteria such as geometry, texture, semantics and acceptance among others, CityGML has been adopted in 2011 as the main format of the Dutch 3D SDI (Stoter et al., 2011). Some characteristics of CityGML are multi-scale representation, coherent semantic-geometrical modelling and geometric topological modelling (Kolbe et al., 2005). The multi-scale representations of CityGML are developed as five Levels of Detail (LoD) , describing the scale of the object geometry. For each of these LoD representations, geometry can be stored. CityGML is based on GML3 which uses the ISO 19107 geometry model (Kolbe, 2009). Geometric primitives exist for each dimension. A zero-dimensional primitive is a point, a one-dimensional primitive is an edge, a two-dimensional primitive is a surface and a three-dimensional primitive is a solid.

Solids in CityGML are described by using a boundary representation (b-rep), this means the higher dimensional primitives are topologically built up of the lower dimensional primitives. A solid is built up from multiple surfaces, which are built up of a number of edges. The requirements for valid three-dimensional primitives are described by Ledoux (2013). It is stated that solids should be closed or watertight and each shell should be simple, thus a 2-manifold. Since shells should be 2-manifold, no dangling pieces, gaps or overlaps are allowed. Since all surfaces are represented by polygons, their geometry should be planar.



(a) Hole in building model (b) Overshoot in building model

Figure 1: Examples of visible defects in a CityGML dataset

1.1 Scientific relevance

Although CityGML offers a structured way of storing geometry, in practice a significant amount of geometry is not considered valid. This hinders the further analyzing or processing of the model. Therefore the use of valid geometry is essential to be able to make use of the benefits of 3D city models. A validation process for CityGML is available at `http://geovalidation.bk.tudelft.nl/val3dity`, enabling users to check their datasets. In case datasets are not completely valid, repair methods are needed to restore the geometry.

2 Related work

2.1 General CAD Repair

Two approaches for converting defect CAD models into a manifold triangle mesh are given by Bischoff and Kobbelt (2005), namely surface oriented algorithms and volumetric algorithms. Surface orientated algorithms aim to repair the defects by changing the input slightly, making it less effective in special cases. The strong point of volumetric methods is that a manifold mesh is guaranteed in the output, although the input tessellation has been lost. An overview of mesh defects for CAD models in general is given by Attene et al. (2013) along with the repair methods developed over the past.

2.2 CityGML Repair

A classification of the common defects in CityGML is given by Zhao et al. (2014). Geometric defects that may occur are duplicate points, self-intersections, folding, invalid holes, wrong orientation or non-2-manifold solids. It is stated that these defects may arise as a result of the interactive modelling process, data optimizations, conversion of CAD models or the adding of semantics. Basic repair routines are proposed: triangulation, regularization removing dangling pieces and decomposition based on intersections of the model. An alternative set of rules for validation is given by Wagner et al. (2013) addressing the geometric-semantical consistency of CityGML models. Alama et al. (2013) describe a method which iterates through all polygons and solids, and checks each for a number of polygon/solid specific errors. For some of these errors (not for overused edges and points) surface oriented algorithms are proposed to repair the model. Using this approach does not guarantee valid output, it is stated that new defects are encountered for every other dataset.

A method which can be considered volumetric but still preserves the semantics is given by Zhao et al. (2013). This method of repairing LOD2 buildings is called shrink-wrapping. A constrained tetrahedralization is performed on all faces of the model, which is used in a carving process, removing tetrahedra outside of the model. This method is capable of repairing defects of LOD2 CityGML, although in cases with overshoots the output is often not valid.

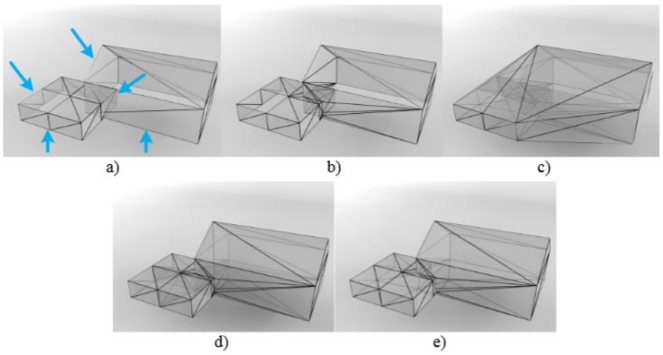


Figure 2: Shrink wrapping method (Zhao et al., 2013)

Another volumetric method of simplification and repair of polygonal models is described by Nooruddin and Turk (2003). By converting a 3D polygon model to a voxelized version and back, holes, double walls and intersecting parts are removed. After this step a surface reconstruction is needed to convert back to a polygon mesh. Using a volumetric representation and surface reconstruction seems to be robust, although semantics will be lost initially. The possibilities of applying a voxel based reconstruction method on invalid CityGML geometry will be explored in this project.

3 Research objectives

3.1 Objectives

The main research question for this thesis is:
To which extent is it possible to automatically repair CityGML buildings LoD 1-3 using a voxel based method?

The goal of this research will be to study the potential and implement a tool for performing an automatic repair for CityGML Buildings in LOD 1-3 using voxel based repair method. To achieve this, the following sub-questions will be relevant:

- What are the most common errors of invalid CityGML models? Which of these errors can be repaired by using a voxel-based repair method?
- Which voxelization algorithm is most suitable for an automatic repair method? How can the semantics be kept during voxelization process?
- Which surface reconstruction method is most suitable for rebuilding sharp geometry characteristics? Is a MarchingCubes algorithm sufficient or are other algorithms needed? Is additional processing such as surface smoothing or edge sharpening required?

3.2 Scope of research

This thesis will focus on the repair of the CityGML data format, limited to LOD 1-3 (no interior models). Furthermore this project will be limited to voxel-based repair methods. Only if necessary, other elementary repair steps will be performed before starting the voxelization process. If a satisfying result regarding geometry is achieved, the possibilities of restoring semantics may be researched.

4 Methodology

As a general guideline the method described in (Nooruddin and Turk, 2003) will be used. Considering the possible limitations of this approach, alternative methods such as 'Pressing' by Chica et al. (2008) will be used for comparison. Figure 3 displays a flowchart of the process of repairing CityGML models with a voxel-based method. These steps are axis alignment, voxelization, surface reconstruction and writing back to the CityGML format. After this process the acquired output geometry may be tested by comparing it to the original data.

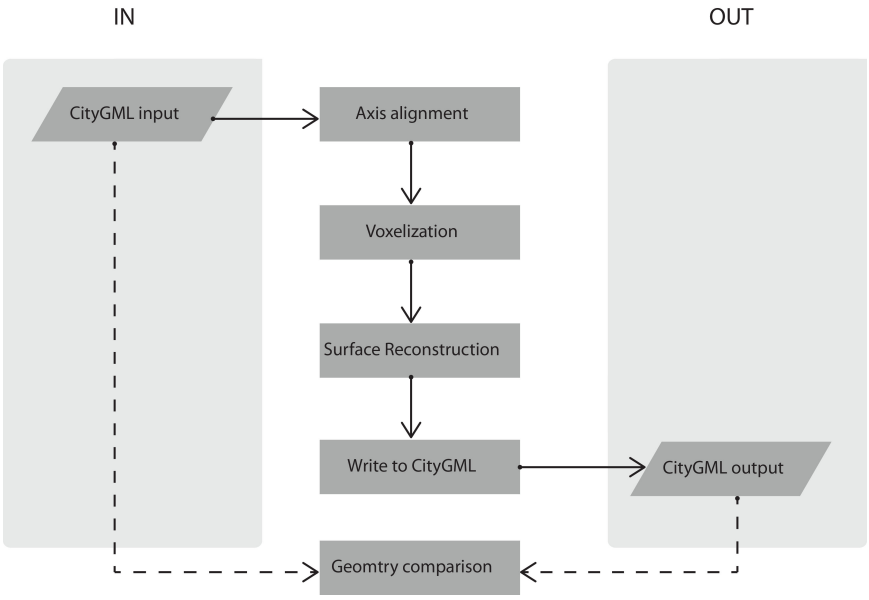
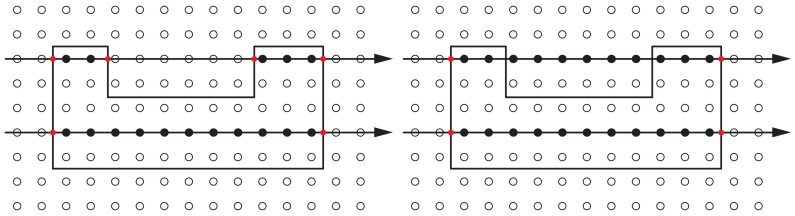


Figure 3: Flowchart

4.1 Voxelization

The goal of the voxelization component is to create a volumetric representation of the input model using appropriate voxel orientation and size, avoiding the creation of unwanted artefacts. There are two algorithms to determine whether a voxel should be considered inside or outside a model: Ray Stabbing or Parity Count. Both methods calculate the number of intersections of the model with a scan line. In Ray Stabbing a voxel is considered to be inside the model when it is positioned in between the first and last intersection along the scan line. Parity Count is more strict, as it only considers the voxel to be inside the model when there is an un-even amount of intersections on both sides along the scan line (see figure 4).

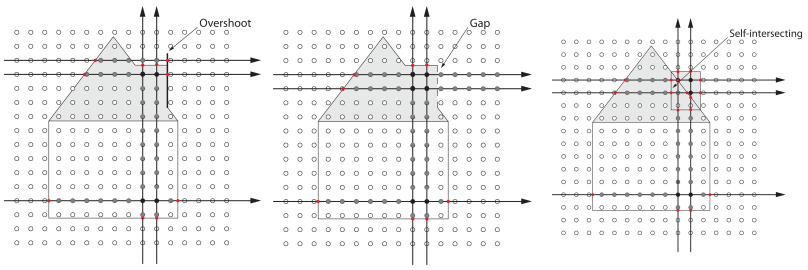


(a) Parity Count: all intersections taken into account

(b) Ray Stabbing: first and last intersection taken into account

Figure 4: Difference in voxelization algorithms

In general the Parity Count method is expected to more often result in correct decisions. However, the Ray Stabbing method may solve cases of self-intersecting models. An option in both of these methods is to use scan-lines in multiple directions and use a majority voting system to determine whether a voxel is inside the model.



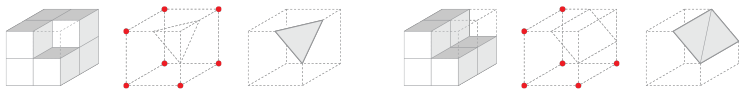
(a) Overshoot (b) Gap (c) Self-intersection

Figure 5: Parity Count voting over two scan directions

Figure 5 shows the potential of a Parity Count voting process for repairing overshoots and gaps. The voxelization method described by Nooruddin and Turk (2003) uses a parity count in multiple directions. A similar method is used in the program Binvox which will be used during the initial stages of the project.

4.2 Surface rebuilding

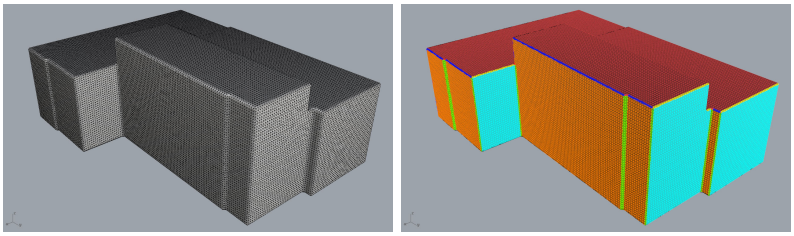
The goal of the surface reconstruction component is to rebuild a polygonal mesh out of the volumetric representation. Figuring out which algorithm is most suitable is a main part of this research. Nooruddin and Turk (2003) use a modified version of the Marching Cubes algorithm (Lorensen and Cline, 1987) since the original Marching Cubes algorithm may result in ambiguity in certain cases. The method of Noorrudin and Turk uses the extended implementation of Wilhelms and Van Gelder (1990). The entire voxel model is looped through, looking at blocks of 8 voxels. This results in 16 possible combinations of empty/filled voxels. Based on the configuration of the voxel-blocks, triangles are added to a mesh. Two examples of cases are visible in figure 6.



(a) Case 1 (b) Case2

Figure 6: Examples of Marching Cubes cases

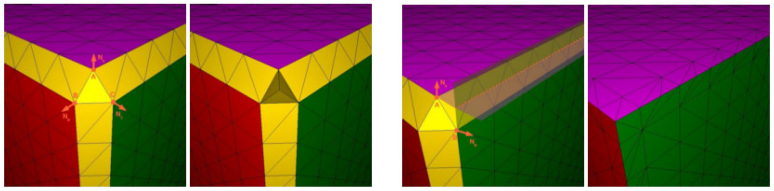
The next step according to Nooruddin and Turk (2003) is simplifying the extracted iso-surfaces using Quadric Error Metrics (Garland and Heckbert, 1997) until the desired number of triangles is achieved. Considering the many different triangle orienations resulting from the Marching Cubes algorithm, the simplification method is likely to leave a higher number of triangles than desirable. Some basic alterations to this method have the potential of improving the results. First of all aligning the input models with the axis will translate into a less 'bumpy' mesh, which will be easier to simplify.



(a) Result after Marching Cubes (b) Result after segmentation

Figure 7: Normal vector segmentation after Marching Cubes

After the axis-alligning of the model, triangles may be segmented based on normal vector orientation. This way neighbouring triangles with similar orientation can easily be grouped together, after which sharp edges may be recovered (see figure 8. However, this method only applies to orthogonal (parts of) buildings.



(a) Insert a vertice in triangles connected to three different faces

(b) Subdivide edges connected to two different faces

Figure 8: Sharp edge recovery (Chica et al., 2008)

An alternative method is Pressing, partially aimed at finding 'large planar tiles' (Chica et al., 2008). Its purpose is to rebuild 3D models from their binary representation. It consists of three steps; (i) finding large planar tiles, (ii) finding curved surfaces and (iii) sharpening the edges. Since curved input models are in general not present, this part can be disregarded. The computing of maximal tiles is described in detail in Andújar et al. (2004).

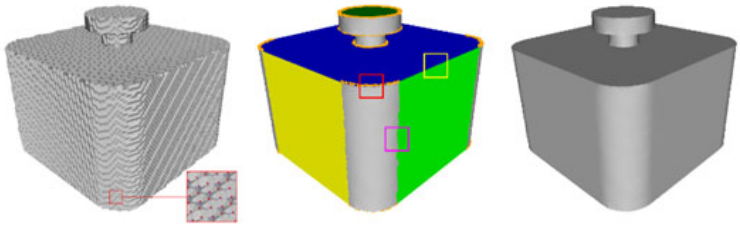


Figure 9: Finding large flats in a 3D binary grid (Chica et al., 2008)

An alternative method combining a volumetric algorithm while preserving the input tessellation described in (Bischoff and Kobbelt, 2005) may also provide a possibility of solving the problem. A decision will have to be made on the best method for implementation. The final implementation will be compared to Shrink Wrapping method (Zhao et al., 2013) to find out which method is most suitable for different defects of the input model.

4.3 Writing to CityGML

Once a 2-manifold polygon mesh with the desired number of triangles is achieved, the next step is writing the model back to CityGML format. During this process there are two challenges. First, any semantics stored with initial surfaces should ideally be preserved. Apart from the semantics per surface, also any aggregations of elements should be kept intact.

4.4 Measuring result

The success of a repair process can be measured in several ways. Three factors for a volumetric repair method for CityGML are proposed here; (i) repair percentage, (ii) number of surfaces and (iii) geometric difference. The *percentage of repairs* simply looks at which percentage of a dataset was considered valid before and after running the repair process. The *number of surfaces* of a repaired CityGML Building should ideally be equal (or close) to the number in original CityGML input. A low number of surfaces in a valid model indicates a successful surface reconstruction. The *geometric difference* between the input and output CityGML model is a measure of accuracy of the model repair. This is relevant since building boundaries may shift during the voxelization process. A method for mesh comparison is proposed by Roy et al. (2002, 2004) with the possibility of measuring both geometric and attribute data. This step requires writing the CityGML data to a 3D triangular mesh in VRML-format.

5 Schedule

5.1 Activities

The following schedule has been set up for the activities which are needed to meet the research objectives .

Start	End	Activity
20 oct	31 oct	Exploring graduation topics
		P1 - Progress review Graduation Plan
01 nov	31 dec	Literature study
6 nov	01 dec	Research existing defects in CityGML data
17 nov	15 jan	Study voxelization methods
17 nov	15 jan	Study surface reconstruction methods
		P2 - Formal assessment Graduation Plan
01 feb	16 mar	Implement existing voxelization methods
01 feb	16 mar	Implement existing surface reconstruction methods
		P3 - Colloquium midterm
24 mar	12 may	Write final implementation
01 may	12 may	Thesis writing
		P4 - Formal process assessment
12 may	15 jun	Finalize thesis
12 jun	22 jun	Prepare final presentation
		P5 Public presentation and final assessment

The graduation calender is shown below. The exact dates of the presentations will be determined during the year.

Date	Event
P1	11 November
P2	22 January
P3	March week 12
P4	May weeks 20-21
P5	June weeks 26-27

5.2 Meetings

Weekly meetings will be held with the daily supervisor dr. H. Ledoux when necessary. Additional guidance and feedback will be provided by the graduation professor Prof. dr. J.E. Stoter. The co-reader is yet to be decided.

6 Tools and Data

6.1 Tools

In order to read, validate and process CityGML data several tools will be required. For the viewing and processing of CityGML FME software will be used. The validity of CityGML datasets before and after repair will be tested with the val3dity tool by Hugo Ledoux, available at `https://github.com/tudelft3d/val3dity`. Python will be used to write the code, along with the package Numpy. Binvox is a fast implementation of the voxelization methods mentioned by Nooruddin and Turk (2003) created by Patrick Min. It is available at: `http://www.cs.princeton.edu/~min/binvox/`. For 3D data visualization/analysis of meshes and binary volumes, Rhino and Paraview will be used respectively. MeshDev is a tool for mesh comparison as described by Roy et al. (2002, 2004) which may be used to measure the repair results. It is available at `http://meshdev.sourceforge.net/`.

6.2 Data

The dataset on which the data will be tested is Rotterdam3D, The municipality of Rotterdam has published their 3D city model as open data. The datasets are in LoD2 and are based on the BAG (Basic Register for Addresses and Buildings) and the height model of Rotterdam, it is available at: `http://www.rotterdam.nl/links_rotterdam_3d`. Another dataset for testing is from Montreal, also in LoD2. This dataset is available at: `http://donnees.ville.montreal.qc.ca/dataset/`. Additionally, manual models will be used to check for specific cases.

References

Alama, N., Wagner, D., Wewetzer, M., von Falkenhausen, J., Coors, V., and Pries, M. (2013). Towards automatic validation and healing of CityGML models for geometric and semantic consistency. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1(1):1–6.

Andújar, C., Brunet, P., Chica, A., Navazo, I., Rossignac, J., and Vinacua, A. (2004). Computing maximal tiles and application to impostor-based simplification. In *Computer Graphics Forum*, volume 23, pages 401–410. Wiley Online Library.

Attene, M., Campen, M., and Kobbelt, L. (2013). Polygon mesh repairing: An application perspective. *ACM Computing Surveys (CSUR)*, 45(2):15.

Bischoff, S. and Kobbelt, L. (2005). Structure preserving CAD model repair. In *Computer Graphics Forum*, volume 24, pages 527–536. Wiley Online Library.

Chica, A., Williams, J., Andujar, C., Brunet, P., Navazo, I., Rossignac, J., and Vinacua, A. (2008). Pressing: Smooth isosurfaces with flats from binary grids. In *Computer Graphics Forum*, volume 27, pages 36–46. Wiley Online Library.

Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co.

Kolbe, T., Gröger, G., and Plümer, L. (2008). CityGML–3D city models and their potential for emergency response. *Geospatial information technology for emergency response*, 257.

Kolbe, T. H. (2009). Representing and exchanging 3D city models with CityGML. In *3D geo-information sciences*, pages 15–31. Springer.

Kolbe, T. H., Gröger, G., and Plümer, L. (2005). CityGML: Interoperable access to 3D city models. In *Geo-information for disaster management*, pages 883–899. Springer.

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

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM.

Nooruddin, F. S. and Turk, G. (2003). Simplification and repair of polygonal models using volumetric techniques. *Visualization and Computer Graphics, IEEE Transactions on*, 9(2):191–205.

Roy, M., Foufou, S., and Truchetet, F. (2002). Generic attribute deviation metric for assessing mesh simplification algorithm quality. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages 817–820. IEEE.

Roy, M., Foufou, S., and Truchetet, F. (2004). Mesh comparison using attribute deviation metric. *International Journal of Image and Graphics*, 4(01):127–140.

Stoter, J., van den Brink, L., Vosselman, G., Goos, J., Zlatanova, S., Verbree, E., Klooster, R., van Berlo, L., Vestjens, G., Reuvers, M., et al. (2011). A generic approach for 3D SDI in the netherlands. In *Proceedings of the Joint ISPRS Workshop on 3D City Modelling&Applications and the 6th 3D GeoInfo Conference Wuhan, China*, pages 26–28.

Wagner, D., Wewetzer, M., Bogdahn, J., Alam, N., Pries, M., and Coors, V. (2013). Geometric-semantical consistency validation of citygml models. In *Progress and New Trends in 3D Geoinformation Sciences*, pages 171–192. Springer.

Wilhelms, J. and Van Gelder, A. (1990). *Topological considerations in isosurface generation extended abstract*, volume 24. ACM.

Zamyadi, A., Pouliot, J., and Bédard, Y. (2013). A three step procedure to enrich augmented reality games with CityGML 3D semantic modeling. In *Progress and New Trends in 3D Geoinformation Sciences*, pages 261–275. Springer.

Zhao, J., Ledoux, H., and Stoter, J. (2013). Automatic repair of CityGML LOD2 buildings using shrink-wrapping. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, II-2 W*, 1:309–317.

Zhao, J., Stoter, J., and Ledoux, H. (2014). A framework for the Automatic Geometric Repair of CityGML Models. In *Cartography from Pole to Pole*, pages 187–202. Springer.