ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-4/W1, 2016
1st International Conference on Smart Data and Smart Cities, 30th UDMS, 7–9 September 2016, Split, Croatia

GENERATION OF MULTI-LOD 3D CITY MODELS IN CITYGML WITH THE PROCEDURAL MODELLING ENGINE RANDOM3DCITY

F. Biljecki *, H. Ledoux, J. Stoter

3D Geoinformation, Delft University of Technology, The Netherlands — (f.biljecki, h.ledoux, j.e.stoter)@tudelft.nl

KEY WORDS: Procedural modelling, CityGML, Level of detail (LOD), Multi-Scale

ABSTRACT:

The production and dissemination of semantic 3D city models is rapidly increasing benefiting a growing number of use cases. However, their availability in multiple LODs and in the CityGML format is still problematic in practice. This hinders applications and experiments where multi-LOD datasets are required as input, for instance, to determine the performance of different LODs in a spatial analysis. An alternative approach to obtain 3D city models is to generate them with procedural modelling, which is—as we discuss in this paper—well suited as a method to source multi-LOD datasets useful for a number of applications. However, procedural modelling has not yet been employed for this purpose. Therefore, we have developed RANDOM3DCITY, an experimental procedural modelling engine for generating synthetic datasets of buildings and other urban features. The engine is designed to produce models in CityGML and does so in multiple LODs. Besides the generation of multiple geometric LODs, we implement the realisation of multiple levels of spatio-semantic coherence, geometric reference variants, and indoor representations. As a result of their permutations, each building can be generated in 392 different CityGML representations, an unprecedented number of modelling variants of the same feature. The datasets produced by RANDOM3DCITY are suited for several applications, as we show in this paper with documented uses. The developed engine is available under an open-source licence at Github at `http://github.com/tudelft3d/Random3Dcity`.

1. INTRODUCTION

3D city models are three-dimensional representations of the urban environment. They can be generated with a multitude of approaches: photogrammetry (Suveg and Vosselman, 2004), laser scanning (Vosselman and Dijkman, 2001, Demir and Baltsavias, 2012), handheld devices (Rosser et al., 2015), conversion from architectural models (Donkers et al., 2015), radar (Stilla et al., 2003), and procedural modelling (Müller et al., 2006). Each acquisition approach is tied to the level of detail (LOD), a measure that indicates the spatio-semantic adherence of a model to its real-world equivalent, and the LOD has implications on its usability (Biljecki et al., 2014b). While in practice the LOD mostly refers to the richness of the geometry, the concept encompasses also the granularity of semantics and the amount of attributes (Stadler and Kolbe, 2007) (see Fig. 1).

Techniques for producing 3D data are capable of deriving data in multiple LODs (e.g. an airborne laser scanning survey can result in both block models, and models with detailed rooftops). However, multi-LOD data of the same real world object are still seldom available, and this will probably not improve in the near future. This deficiency hinders applications that require them as input, such as visualisation. There are a few possible reasons why multi-LOD datasets are rare, among others: (1) GIS software packages are generally not programmed to utilise multi-scale representations of 3D models; (2) 3D city models are usually acquired for one purpose in mind; hence they are acquired in the optimal (single) LOD; and (3) there are limitations in the acquisition and storage process, e.g. a 3D modelling software is not capable of simultaneously producing two or more representations, and to store them consistently and efficiently.

The aim of this paper is to present our method that we developed and implemented to generate 3D city models in multiple LODs. In Section 2 we present related work and we justify procedural modelling as a potentially suitable acquisition technique to derive such data, which despite their synthetic nature may still be suited for various applications and experiments. In Section 3, we introduce RANDOM3DCITY, an experimental procedural modelling engine which we have developed to generate buildings in multiple LODs in the CityGML format. It is the first engine of this kind, and we have released the code open-source for free public use. The engine is composed of two modules and it has been built entirely from scratch with a custom shape grammar. The datasets generated by this engine have already been used in several research projects (Section 4).

Figure 1: A building modelled in multiple LODs. The LOD labels are according to the categorisation of CityGML 2.0, and they denote both the granularity of the geometry and semantics. Besides small and illustrative models such as this one, building datasets containing multiple LODs are virtually non-existent.

2. BACKGROUND AND RELATED WORK

2.1 Motivation for multi-LOD models

In practice, the vast majority of 3D city models is stored as one representation, which is sufficient for many single use case scenarios. However, there are situations and use cases in which having multi-LOD data may bring benefit. For instance, (1) visualisation applications and spatial analyses in which, similarly to

*Corresponding author

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
doi:10.5194/isprs-annals-IV-4-W1-51-2016

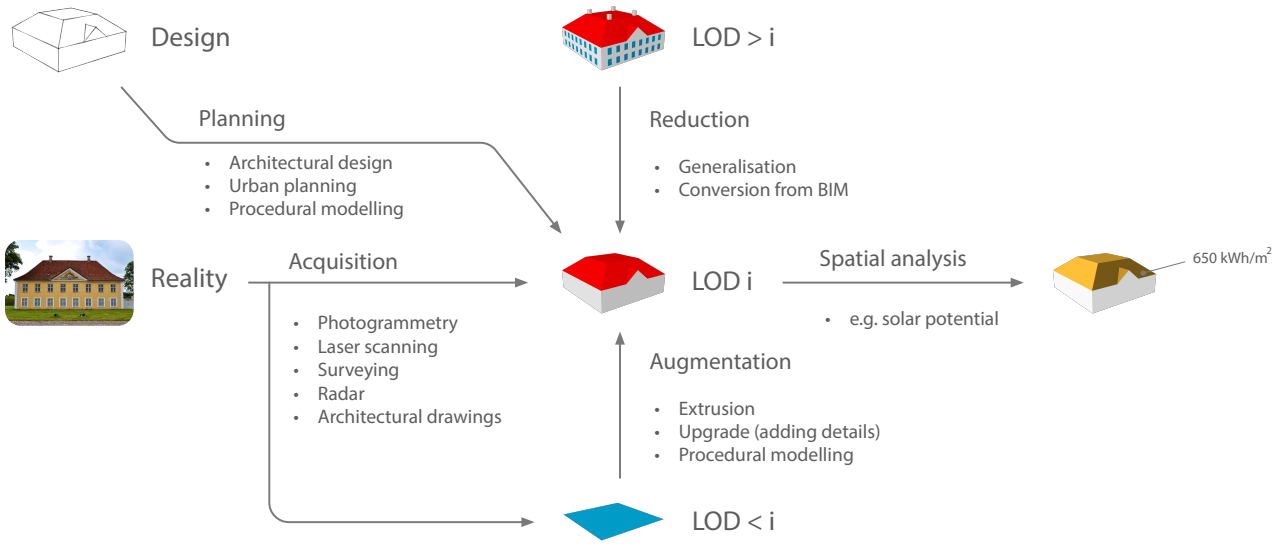

Figure 2: Life cycle of a 3D city model: different production workflows of 3D city models from the perspective of the level of detail.

computer graphics, multiple LODs are switched for efficient visualisation and to increase cognition, and to decrease computational complexity (Çöltekin and Reichenbacher, 2011); (2) as a source of data for testing software implementations that are focused on structuring multi-scale data, e.g. compression methods; (3) to enable data producers to differentiate products by stripping down a high-quality model to attract different segments of the market (cf. quality discrimination and product versioning); and (4) for assessing the suitability of a specific LOD prior to tendering and data acquisition ("LOD benchmarking") and for serving different applications (i.e. noise models only need block models while solar potential analysis needs finer detail). For instance, when planning the procurement of 3D city models for a specific application, it would be beneficial to have a sample dataset in multiple LODs, run a spatial analysis for each, and compare the accuracy of the results to their cost of acquisition (for an example see the paper (Biljecki et al., 2017)). Such performance analysis can then serve as a decision factor for determining the optimal LOD to be acquired, prior to acquisition and to avoid procuring data of an unsuitable LOD (either too fine or too coarse). In this paper we address the problem of the absence of such experimental datasets.

2.2 CityGML

In our work we focus on the CityGML format, as it is the most prominent way to store semantic 3D city models. In this way we also contribute to the growth of publicly available CityGML data.

The OGC CityGML standard represents an information model and format for storing 3D city models (Gröger and Plümer, 2012, Open Geospatial Consortium, 2012a, Kolbe, 2009). Among other strengths, CityGML datasets are semantically structured, enabling various 3D spatial analyses which require semantic information.

The standard defines five LODs and supports storing data in multiple LODs. For buildings, LOD0 is a 2.5D representation of their footprints, LOD1 is a coarse prismatic (block) model. LOD2 represents a model with a roof that is modelled as a standardised roof shape. LOD3 is an architecturally detailed model with openings such as windows and doors. LOD4 completes an LOD3 by including indoor features (Kolbe, 2009) (see Fig. 1).

CityGML is a relatively new format and despite many advantages its software support and adoption are still inferior in comparison to seasoned computer graphics formats such as COLLADA. This limitation, coupled with the general lack of multi-LOD data, renders multi-LOD CityGML data virtually non-existing in practice (Biljecki et al., 2015b).

2.3 Overview of acquisition workflows

We have analysed 3D production workflows and accompanying software support in order to develop an approach to facilitate the production of multi-LOD data. From the LOD perspective, we recognise the following groups of approaches to acquire 3D models (Fig. 2):

Direct acquisition The usual approach of deriving 3D city models: a subset of the real-world is abstracted and modelled according to a predefined LOD, e.g. a photogrammetric survey is carried out to produce LOD2 building models.

Reduction A dataset is obtained by generalisation from an existing 3D city model of a finer LOD. This approach is usually employed when the 3D model is too complex for the intended application (Guercke et al., 2011). Generalisation essentially results in multiple representations (the original one and its simplified counterpart).

Augmentation Existing spatial data is used as a base to generate data of a finer LOD. It can be optionally aided by additional data sources. For instance, if a footprint of a building is available, its height from a cadastral source can be used to generate a block model (extrusion). Another example is adding roof structures to LOD1 block models to obtain an LOD2 model (Sugihara and Shen, 2016).

Planning It should also be noted that a fraction of 3D models are design models which do not represent a real-world setting, e.g. simulated 3D models used in movies, architectural models of planned buildings, and models designed by urban planners to disseminate urban planning concepts (Ben-Joseph et al., 2001). Such models have not been used much in GIS, but our work shows that their usability has been underestimated as they can be used in various GIS experiments where having real-world data is not essential.

When generating data in multiple LODs, the first approach can be laborious and expensive, and it is hindered by software limitations. Furthermore, the data is burdened with acquisition errors and it may include other inconsistencies such as invalid geometries, an unwanted but common outcome of 3D acquisition (Ledoux, 2013, Brasebin et al., 2012).

Second, obtaining multi-LOD models with generalisation is viable in theory, however, there is a lack of implemented solutions, especially those that support CityGML. Furthermore, in this approach a dataset of fine LOD is required, which are usually available for only a limited set of buildings.

2.4 Procedural modelling

In this paper we focus on procedural modelling, as a common augmentation approach and source of design models, but not previously considered as a source of multi-LOD data. Procedural modelling involves creating 3D city models from scratch or based on an existing 2D dataset by using a set of rules (Goetz, 2013, Martinović, 2015). It is an important topic in computer graphics and GIS, and there have been several initiatives to develop procedural engines for modelling urban features. For instance, buildings (Wonka et al., 2003, Müller et al., 2006, Kelly and Wonka, 2011, Besuievsky and Patow, 2013a), landmarks (Rodrigues et al., 2010), roads (Beneš et al., 2014), plants (Lintermann and Deussen, 1999), monuments (Koehl and Roussel, 2015), and land parcels (Vanegas et al., 2012). For a comprehensive list see the overview in (Smelik et al., 2014).

Procedurally generated models have proven to be an efficient technique for generating 3D models, mostly by enhancing existing data—e.g. adding fenestration to an LOD2 model (Kim and Wilson, 2014, Tsiliakou et al., 2014, Müller Arisona et al., 2013). They find their use in GIS in an increasing range of applications, for instance, urban planning and simulation (Besuievsky and Patow, 2014, Rautenbach et al., 2015). An example workflow is to take block models of buildings, and to *synthetically* augment their detail and appearance (e.g. adding a roof shape and textured façade) according to a predefined architecture typical for that spatial extent, resulting in a 3D city model with a much finer level of detail without a significant additional cost. Another example is to take building footprints as input, e.g. detected from a point cloud (Hermosilla et al., 2011), and to generate buildings on top of it.

An advantage of this technique is that it is a quick and simple method to generate 3D city models, usually in large quantities. Furthermore, the models derived with procedural modelling are fine in detail (Rodrigues et al., 2010), and because of their nature they rarely contain topological inconsistencies (e.g. models obtained with automatic reconstruction from LiDAR point clouds are more susceptible to topological errors).

As a disadvantage, due to their generative nature, procedurally modelled datasets are not accurate from the GIS point of view: in fact, they may considerably deviate from the reality they purport to represent (Musialski et al., 2013). This is due to the primary goals of procedural modelling: to quickly generate 3D data, and to increase the LOD of existing models to improve their visual impression, achieved by *artificially* adding features. Nevertheless, this inconsistency does not interfere with many applications, such as flight simulation, and gaming, where the focus is on visualisation, rather than on spatial analyses (Besuievsky and Patow, 2013b). As a result, many synthetic datasets have been generated completely from scratch, representing fictitious settings, for instance, for movies.

At the moment, the most prominent procedural modelling engine is ESRI's CityEngine, widely used by urban planners and other practitioners. However, so far procedural modelling efforts do not appear to have been focused much on multi-scale representations and CityGML. Considering that a procedural engine can be programmed to produce data with a specific granularity, we take advantage of this idea in our work by defining different procedures to generate buildings in a series of different representations.

3. PROCEDURAL MODELLING ENGINE RANDOM3DCITY

In this section we present a CityGML compliant procedural modelling engine that we have developed specifically to produce models in multiple LODs. We have formulated and developed a custom methodology, shape grammar, and rules that can be modified to suit the requirements of a user.

Besides the motivation of tackling the absence of multi-LOD datasets and shortage of diverse sample CityGML data, we have created RANDOM3DCITY for other reasons, for instance, to address the lack of CityGML procedural modelling software to easily create 3D city models in the respective format. The engine has two functions: generating an unlimited number of synthetic models that mimic a real-world setting, completely from scratch, and to augment existing datasets.

The software prototype is composed of two independent and extensible modules (see Figure 3 for the workflow). The first module consists of a customisable set of rules that derives the configuration of the architecture in a parametric description encoded in an XML format (Section 3.1). This means that the architecture and rules can be adapted to a specific setting. For instance, different rules can be encoded, such as to imitate a residential area with buildings that are between 3 and 6 storeys high and have predominantly flat roofs.

The second module reads the generated parametric data, and realises the parametric description of buildings as 3D city models in CityGML in multiple representations (Section 3.2).

The module is designed to generate 3D models according to a refined specification of the LODs of CityGML (Biljecki et al., 2016a), and generates data in 16 geometric LODs, rather than only the 5 standard ones. For instance, it creates a variant of LOD2 with dormers and other roof structures, and another one without. Besides generating each building in multiple representations distinguished by geometric complexity, the engine generates models in multiple geometric references (e.g. LOD2 with walls at their actual location and in the other variant with walls as projections from roof edges; see (Biljecki et al., 2016b) for an overview); in multiple levels of semantic structuring (e.g. LOD3 with and without the thematically enriched surfaces); with the difference in geometric type (boundary representations and solids); and their corresponding indoor geometry further in multiple LODs. Permuting all these combinations results in 392 representations of the same building. To the extent of our knowledge, the models obtained with RANDOM3DCITY present the most complete CityGML (and probably in general 3D building) datasets available to date, contributing to a multitude of application domains (Section 4).

The engine also supports thematic features other than buildings, and the generation of a basic interior of buildings (Section 3.3).

Each module of the engine is independent, facilitating the extensibility and integration with other data or other engines. For instance, since the modelled data is first stored in a parametric form, it can be generated in formats other than CityGML. On the

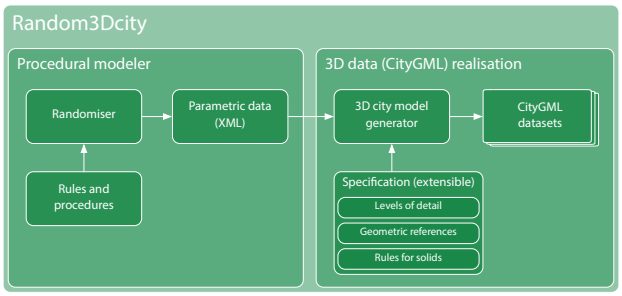

Figure 3: Modular workflow of the engine RANDOM3DCITY.

other hand, an open parametric building format brings two benefits: first, it facilitates the integration with existing data. This is shown in Section 3.4 where we generate a 3D city model based on a real-world 2D cadastral dataset, similarly as in present-day commercial software. And second, the 3D data generator may be independently used to generate buildings derived from other sources with the same rules if stored in this form.

Figure 4 illustrates an example of the output of the engine: a setting with 100 synthetic buildings in four LODs which are randomly placed and rotated. All datasets (incl. the 388 others not shown here) have been generated in less than one minute.

The software has been implemented in Python, and it is available as open-source at Github. A set of generated test datasets is also available for public use on the website of the project.

3.1 Parametric description and rules

In the engine, each building and other real-world feature F_i consists of a set of n parameters p_i that define its architecture:

$$F_i = \{p_i^1, p_i^2, \dots p_i^n\} \tag{1}$$

These parameters are in line with the ones used in photogrammetry (e.g. *cf.* (Zhang et al., 2014, Haala and Kada, 2010, Sinning-Meister et al., 1996)), such as the width of a building, length of ridges and eaves, and roof height. The first part of the engine, described in this section, creates features described by these parameters, using an encoded grammar and a set of rules. The parameters are then stored in an XML schema that we have defined.

Our methodology of procedurally modelling the urban features consists of first defining a top-down hierarchy of parameters p_i. For instance, before determining the number of windows on a wall, first the width and length of the footprint of the building are derived. Thanks to the hierarchical approach, the parameters are context-aware, i.e. the engine contains several procedural rules and constraints. For instance, flat roofs cannot contain dormers, doors have to be located on the ground floor, buildings have to have at least one floor, and windows cannot be taller than the height of the floor. Second, a range for each parametre p_i is defined, e.g. the width of the window is between 0.5 and 1.5 m, from where the engine randomly samples a value according to a defined probability distribution function (also customisable). Third, the rules take care to generate a realistic setting, for instance, that multiple dormers are properly aligned on the roof. All these rules are stored in the code in a series of IF-THEN statements, and they are customisable to conform to a specific setting a user aims to (re)construct.

RANDOM3DCITY supports five types of roofs, which are frequently described as the most common types of roofs (Kada, 2007, Hammoudi and Dornaika, 2011, Haala and Brenner, 1999, Henn et al., 2013). They are shown in Figure 5. Furthermore, the figure demonstrates that a building part such as a garage can also be generated.

In addition to the geometry and semantic representation, the engine is capable of generating a number of attributes, such as building age, number of floors, use of building, which can be useful for some use cases. All together, these parameters are stored in an XML schema (Fig. 6). This example shows the underlying parametric description of the second building from the left in Fig. 5.

3.2 CityGML realisation of the parametric building

The second part of our engine reads the generated parametric data p_i of each feature F_i and constructs CityGML 2.0 datasets in multiple LODs. The process of the construction of the geometric representation from the parametric representation is described in this section by highlighting a few important aspects.

3.2.1 Construction of the geometry and the CityGML file

Each of the representations is generated separately. For each, the engine reads only the set P_i of parameters p_i that it requires for constructing it. For instance, for the LOD2, a set P_i^{LOD2} is defined and the engine fetches the size of the body of the building, roof type, and height of the roof, ignoring other parameters such as windows and dormers. These instructions are stored in the engine, and can be customised to create additional custom LODs, if required. For instance, it is possible to generate an LOD3-like model that contains only roof openings, by simply disabling the constructor for other features such as dormers.

In the construction of the geometry, the engine first creates a local Cartesian coordinate system X_i for the building F_i, and then a coordinate system X_{p_i} for each of its elements defined by p_i (e.g. wall). The vertices of the features are generated in this system (e.g. origin of the window on a wall), and the resulting surfaces are generated. For each different element class (e.g. chimney), an algorithm is designed for their geometric realisation from their parametric description. These "sub-systems" X_{p_i} are then converted to the coordinate system X_i of the building, which is later converted in the global system defined by the user, depending on the location and orientation of the building in space.

The process of the generation of the vertices of the building elements is not equal for all buildings, it mostly depends on the type of the roof. For instance, the vertices of the walls are not equal for buildings with a flat and gabled roof (visible in Figure 5), hence, separate algorithms are designed for each roof type.

In the process of the generation, geometries are given a universally unique identifier (UUID) according to (ISO, 2008), the recommended approach from CityGML and GML (Open Geospatial Consortium, 2012b), and are then structured according to the semantic level of the representation. Furthermore, the attributes have been translated and stored according to the CityGML 2.0 standard. An excerpt of the CityGML of the building exemplified through its parametric description in the previous section is given in Figure 8, which also shows the realised attributes.

3.2.2 Generation of corresponding solids

Each model, with the exception of LOD0 models, is also stored as a `gml:Solid`. Solids are used to facilitate uses in application domains which require the usable volume of a building, such as for the estimation of property taxes (Boeters et al., 2015), and the estimation of the energy demand of households (Strzalka et al., 2011).

Figure 4: Composite rendering of a few exemplary datasets generated by RANDOM3DCITY: randomly generated buildings realised in CityGML in four LODs. The two representations on the left have their walls modelled as projections from roof edges as opposed to the third panel where the walls are modelled at their actual location and where roof overhangs are explicitly modelled, showing varying geometric references that are supported by the engine, besides multiple geometric and semantic LODs.

Figure 5: Different types of roofs (flat, gabled, hipped, pyramidal, and shed) fitted on the same building, with the automatically adjusted walls. This image shows the LOD2 (orthogonal view), and it also features an example of a building part.

In the construction of solids, features that do not contribute towards the usable volume of buildings are disregarded. This applies for instance to roof overhangs, and chimneys. In the `gml:MultiSurface` representation this is also regarded by the generation of a `ClosureSurface` to seal the open sides and to provide a representation as geometrically closed volume object, following the recommendation of (Gröger and Plümer, 2012).

Figure 9 shows an example of the relation between the semantic boundary representation models and its solid counterpart. The solids generated by the engine have been geometrically validated according to the standard ISO 19107 (ISO, 2003) with the implementation of (Ledoux, 2013).

3.3 Generation of indoor and non-building features

We have implemented multiple versions of the interior according to the refinement developed by (Boeters et al., 2015). Further, thematic features other than buildings have been generated, such as vegetation and roads. Figure 7 shows an example with the interior of buildings (LOD2+ model as per (Boeters et al., 2015)).

3.4 Augmenting existing data (2D footprints)

Besides generating all parameters of features from scratch, an experimental feature of the engine is to utilise existing GIS data

Figure 6: Example of the building parameters stored in an XML.

```
<building ID="c209f43f-f137-4dd0-8816-cbc4dc4a407d">
    <footprint>Rectangular</footprint>
    <origin>173469.34 526427.95 0.0</origin>
    <rotation>34.3</rotation>
    <xSize>7.06</xSize>
    <ySize>8.91</ySize>
    <zSize>6.8</zSize>
    <floors>2</floors>
    <floorHeight>3.4</floorHeight>
    <embrasure>0.08</embrasure>
    <wallThickness>0.2</wallThickness>
    ...
    <buildingPart>
      <partType>Garage</partType>
      ...
    </buildingPart>
    <roof>
      <roofType>Gabled</roofType>
      <h>2.48</h>
      <overhangs>
        <xlength>0.5</xlength>
        <ylength>0.5</ylength>
        ...
```

such as footprints, and procedurally model the remaining features producing a 3D model. For instance, Figure 10 shows a setting in which 2D footprints from an existing dataset have been used, and the 3D buildings have been procedurally modelled by specifying the architecture where for instance only hipped roofs are present and where buildings must have large windows (as it is the case for that setting).

4. DOCUMENTED APPLICATIONS OF OUR SOFTWARE

Availability of freely available datasets with a large number of dissimilar buildings represented in multiple LODs opens a door

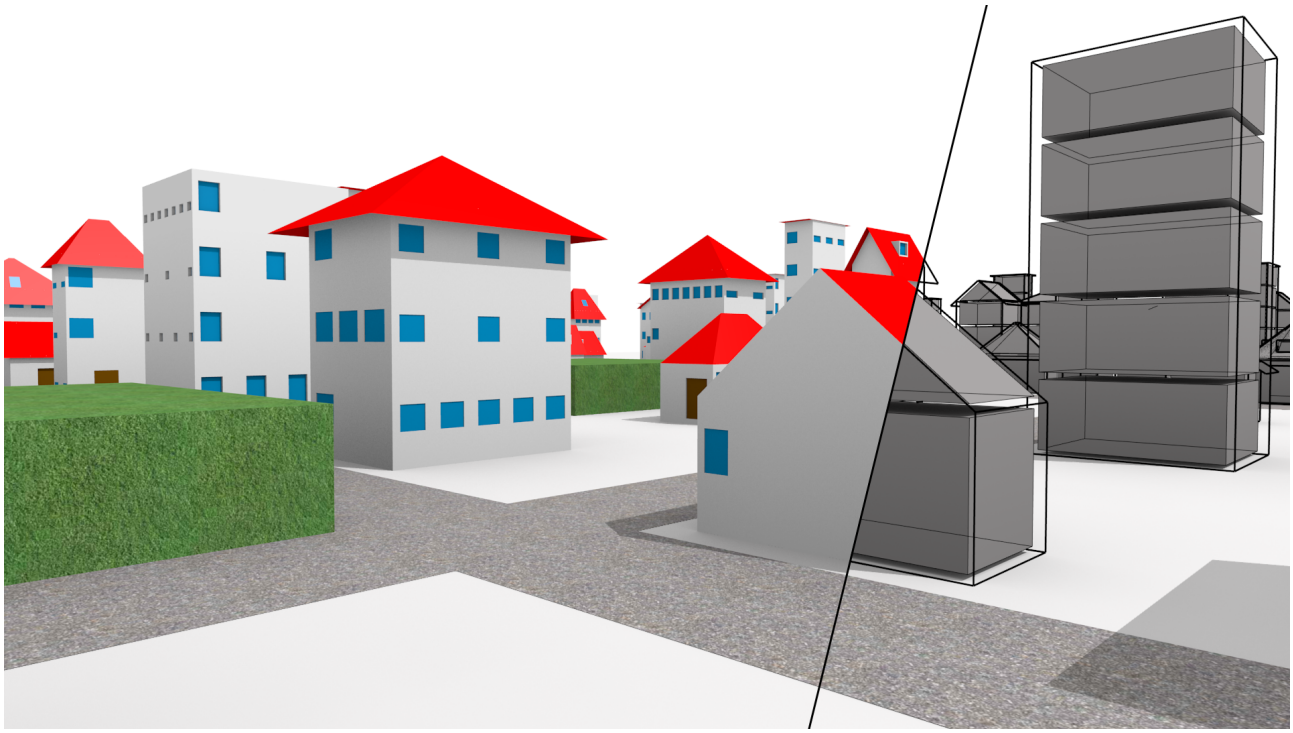

Figure 7: A composite *street view* of a dataset generated by the engine, showing the vegetation (park), street network, and the basic indoor representation of buildings (a solid representing each storey).

```
<cityObjectMember>
 <bldg:Building gml:id="c209f43f-f137-
  4dd0-8816-cbc4dc4a407d">
  <bldg:roofType>Gabled
   </bldg:roofType>
  <bldg:yearOfConstruction>2008
   </bldg:yearOfConstruction>
  <bldg:storeysAboveGround>2
   </bldg:storeysAboveGround>
  <bldg:boundedBy>
   <bldg:GroundSurface>
     ...
```

Figure 8: Excerpt of the generated CityGML 2.0 dataset.

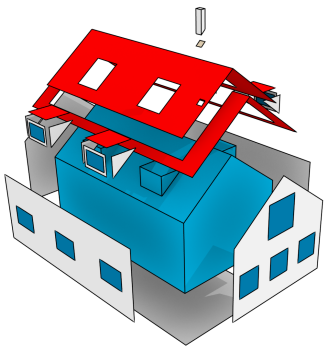

Figure 9: Two representations of LOD3 models: a `gml:Solid` and a thematically structured `gml:MultiSurface`. Note the `ClosureSurface` of the chimney (light brown) in order to separate the usable volume of a building.

for a multitude of research purposes. Instead of an experiment section, we showcase the documented uses of preliminary versions of RANDOM3DCITY from fellow researchers in the 3D GIS community. The generated datasets have already been tested and used in several application domains:

1. Testing and improving CityGML validation software (Ledoux, 2013, Zhao et al., 2013, Coors and Wagner, 2015), primarily in the scope of the OGC CityGML Quality Interoperability Experiment (QIE) (OGC, 2016). Considering that the engine creates topologically consistent datasets, they have been used as exemplary models of valid datasets. Furthermore, for this project the code of the engine was modified to intentionally produce data with topological errors, such as overlapping buildings and broken solids, which were used as input to test validation and repair software packages benchmarked in the QIE.

2. Analyse the propagation of uncertainty from the input data to the result of a GIS operation with a Monte Carlo simulation (Biljecki et al., 2014a, Biljecki et al., 2015a). A stochastic engine on top of the first module of RANDOM3DCITY was implemented to intentionally degrade the building parameters by sampling values from a normal probability distribution function with standard deviation simulating acquisition errors. This approach results in ground truth and erroneous versions of parameters, which the second part of the engine used as an input to create reference and erroneous 3D city models, suited for uncertainty propagation experiments. This is the first instance in which procedurally generated 3D data has been used for uncertainty propagation research.

3. Optimising the coverage of geosensor networks (Doodman et al., 2014, Afghantoloee et al., 2014). Researchers have used our synthetic datasets to test the implementation of their use case, which focuses on line of sight analysis.

4. For testing software which identifies and links topological relationships between similar features across multiple LODs in CityGML (Biljecki et al., 2015b). The project focused on improving the consistency of multi-LOD datasets and their compression. RANDOM3DCITY was found valuable in this

Figure 10: Example of a CityGML model generated with RANDOM3DCITY in conjunction with the existing real-world dataset of 2D footprints of buildings. This setting shows the *Beestenmarkt* in Delft, the Netherlands. The grammar has been adjusted to match the configuration of the buildings in that setting in order to resemble the reality as close as possible.

project as it presented the only source of such data.

5. As a data source in experiments with voxelisation of CityGML models to facilitate volume computation (Steuer et al., 2015).

6. To test the specification of LODs and their differences when used in a spatial analysis (Biljecki et al., 2016a, Biljecki et al., 2017). In these projects, several LODs have been benchmarked in different spatial analyses, such as estimating the shadow cast by a building, to assess if a finer level of detail brings an improvement in a particular spatial analysis.

5. CONCLUSIONS AND FUTURE PROSPECTS

In this paper we have tackled the absence of multi-LOD 3D city models by exploring the possibility of generating them with procedural modelling, as opposed to orthodox techniques such as laser scanning and generalisation. We have developed a procedural modelling engine that is designed with the primary objective to provide models in multiple LODs stored in the OGC CityGML format. In this way we also address the lack of procedural modelling engines that support CityGML, and the shortage of publicly available CityGML data.

The experimental engine RANDOM3DCITY, which we have introduced and built from scratch, is novel: it natively supports CityGML, and it is designed towards producing multi-LOD data. It yields an unprecedented number of variants of models, and does so according to an underlying set of customisable rules. The engine is open-source, and a set of example datasets is available for free public use. The reason why we have made this project open is that other researchers can benefit from multi-LOD datasets in their application domains, and that they could adjust its grammar to suit specific requirements. As a result, the generated data have already been proven useful by being featured in several research projects in different countries and application domains. Such interest suggests the need for open procedural modelling engines. We invite other researchers to take advantage of the availability of these datasets to test their implementations and as a source for experiments.

While the engine generates fictitious settings, it is still suited for applications where having real-world data is not important, and where different scenarios can be evaluated (e.g. to determine whether it is more beneficial to acquire an LOD2 instead of an LOD1 for a specific spatial analysis, by testing both representations before the actual acquisition).

Obviously, this experimental software cannot compete with advanced commercial solutions, such as ESRI's CityEngine, which are capable of creating complex architecture. Nevertheless, it bridges the gap with respect to CityGML data and multiple representations to quickly obtain models suited for experiments and testing, and it is open-source.

For future work we plan to work in two directions. First, we plan to advance the shape grammar for generating more complex buildings, such as structures with less usual roof types and landmarks. This is a natural flow of the work, which is hampered by designing advanced rules that more complex features imply, such as complicated roof shapes and balconies. Second, we plan to increase the LOD of the interior. We have introduced the modelling of a basic indoor (storeys), and we intend to include the generation of rooms and openings (windows/doors), following recent efforts in procedural modelling of interior according to an indoor grammar (Becker et al., 2013, Peter et al., 2013, Gröger and Plümer, 2010, Ilčík and Wimmer, 2013).

ACKNOWLEDGEMENTS

We gratefully acknowledge the comments of the anonymous reviewers, and of Mickaël Brasebin (IGN France) during the development phase. 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

Afghantoloee, A., Doodman, S., Karimipour, F. and Mostafavi, M. A., 2014. Coverage estimation of geosensor in 3D vector environments. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XL-2/W3, pp. 1–6.

Becker, S., Peter, M., Fritsch, D., Philipp, D., Baier, P. and Dibak, C., 2013. Combined Grammar for the Modeling of Building Interiors. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* II-4/W1, pp. 1–6.

Ben-Joseph, E., Ishii, H., Underkoffler, J., Piper, B. and Yeung, L., 2001. Urban simulation and the luminous planning table: bridging the gap between the digital and the tangible. *Journal of Planning Education and Research* 21(2), pp. 196–203.

Beneš, J., Wilkie, A. and Křivánek, J., 2014. Procedural Modelling of Urban Road Networks. *Computer Graphics Forum* 33(6), pp. 132–142.

Besuievsky, G. and Patow, G., 2013a. Customizable LoD for Procedural Architecture. *Computer Graphics Forum* 32(8), pp. 26–34.

Besuievsky, G. and Patow, G., 2013b. Procedural modeling historical buildings for serious games. *Virtual Archeology Review* 4(9), pp. 160–166.

Besuievsky, G. and Patow, G., 2014. Recent Advances on LoD for Procedural Urban Models. *In: Proceedings of the Workshop on Processing Large Geospatial Data*, Cardiff, United Kingdom.

Biljecki, F., Heuvelink, G. B. M., Ledoux, H. and Stoter, J., 2015a. Propagation of positional error in 3D GIS: estimation of the solar irradiation of building roofs. *International Journal of Geographical Information Science* 29(12), pp. 2269–2294.

Biljecki, F., Ledoux, H. and Stoter, J., 2014a. Error propagation in the computation of volumes in 3D city models with the Monte Carlo method. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* II-2, pp. 31–39.

Biljecki, F., Ledoux, H. and Stoter, J., 2015b. Improving the consistency of multi-LOD CityGML datasets by removing redundancy. *In: 3D Geoinformation Science*, Springer International Publishing, pp. 1–17.

Biljecki, F., Ledoux, H. and Stoter, J., 2016a. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems* 59, pp. 25–37.

Biljecki, F., Ledoux, H. and Stoter, J., 2017. Does a finer level of detail of a 3D city model bring an improvement for estimating shadows? *In: Advances in 3D Geoinformation*, Springer International Publishing.

Biljecki, F., Ledoux, H., Stoter, J. and Vosselman, G., 2016b. The variants of an LOD of a 3D building model and their influence on spatial analyses. *ISPRS Journal of Photogrammetry and Remote Sensing* 116, pp. 42–54.

Biljecki, F., Ledoux, H., Stoter, J. and Zhao, J., 2014b. Formalisation of the level of detail in 3D city modelling. *Computers, Environment and Urban Systems* 48, pp. 1–15.

Boeters, R., Arroyo Ohori, K., Biljecki, F. and Zlatanova, S., 2015. Automatically enhancing CityGML LOD2 models with a corresponding indoor geometry. *International Journal of Geographical Information Science* 29(12), pp. 2248–2268.

Brasebin, M., Perret, J., Mustière, S. and Weber, C., 2012. Measuring the impact of 3D data geometric modeling on spatial analysis: Illustration with Skyview factor. *In: Usage, Usability, and Utility of 3D City Models – European COST Action TU0801*, EDP Sciences, Nantes, France, pp. (02001)1–16.

Çöltekin, A. and Reichenbacher, T., 2011. High Quality Geographic Services and Bandwidth Limitations. *Future Internet* 3(4), pp. 379–396.

Coors, V. and Wagner, D., 2015. CityGML Quality Interoperability Experiment des OGC. *DGPF Tagungsband. Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V.* 24, pp. 288–295.

Demir, N. and Baltsavias, E. P., 2012. Automated modeling of 3D building roofs using image and LiDAR data. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* I-4, pp. 35–40.

Donkers, S., Ledoux, H., Zhao, J. and Stoter, J., 2015. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*.

Doodman, S., Afghantoloee, A., Mostafavi, M. A. and Karimipour, F., 2014. 3D extension of the vor algorithm to determine and optimize the coverage of geosensor networks. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XL-2/W3, pp. 103–108.

Goetz, M., 2013. Towards generating highly detailed 3D CityGML models from OpenStreetMap. *International Journal of Geographical Information Science* 27(5), pp. 845–865.

Gröger, G. and Plümer, L., 2010. Derivation of 3D Indoor Models by Grammars for Route Planning. *Photogrammetrie - Fernerkundung - Geoinformation* 2010(3), pp. 191–206.

Gröger, G. and Plümer, L., 2012. CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing* 71, pp. 12–33.

Guercke, R., Götzelmann, T., Brenner, C. and Sester, M., 2011. Aggregation of LoD 1 building models as an optimization problem. *ISPRS Journal of Photogrammetry and Remote Sensing* 66(2), pp. 209–222.

Haala, N. and Brenner, C., 1999. Virtual city models from laser altimeter and 2D map data. *Photogrammetric Engineering and Remote Sensing* 65(7), pp. 787–795.

Haala, N. and Kada, M., 2010. An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing* 65(6), pp. 570–580.

Hammoudi, K. and Dornaika, F., 2011. A Featureless Approach to 3D Polyhedral Building Modeling from Aerial Images. *Sensors* 11(1), pp. 228–259.

Henn, A., Gröger, G., Stroh, V. and Plümer, L., 2013. Model driven reconstruction of roofs from sparse LIDAR point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing* 76, pp. 17–29.

Hermosilla, T., Ruiz, L. A., Recio, J. A. and Estornell, J., 2011. Evaluation of Automatic Building Detection Approaches Combining High Resolution Images and LiDAR Data. *Remote Sensing* 3(6), pp. 1188–1210.

Ilčík, M. and Wimmer, M., 2013. Challenges and ideas in procedural modeling of interiors. *In: Eurographics Workshop on Urban Data Modelling and Visualisation*, Girona, pp. 29–30.

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

ISO, 2008. ISO/IEC 9834-8:2008(E) – Information technology — Open Systems Interconnection — Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components.

Kada, M., 2007. Scale-Dependent Simplification of 3D Building Models Based on Cell Decomposition and Primitive Instancing. *In: Spatial Information Theory*, Springer Berlin Heidelberg, pp. 222–237.

Kelly, T. and Wonka, P., 2011. Interactive architectural modeling with procedural extrusions. *ACM Transactions on Graphics* 30(2), pp. 1–15.

Kim, K. and Wilson, J. P., 2014. Planning and visualising 3D routes for indoor and outdoor spaces using CityEngine. *Journal of Spatial Science* 60(1), pp. 179–193.

Koehl, M. and Roussel, F., 2015. Procedural modelling for reconstruction of historic monuments. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* II-5-W3, pp. 137–144.

Kolbe, T. H., 2009. Representing and exchanging 3D city models with CityGML. *In: 3D Geo-Information Sciences*, Springer Berlin Heidelberg, pp. 15–31.

Ledoux, H., 2013. On the Validation of Solids Represented with the International Standards for Geographic Information. *Computer-Aided Civil and Infrastructure Engineering* 28(9), pp. 693–706.

Lintermann, B. and Deussen, O., 1999. Interactive modeling of plants. *IEEE Computer Graphics and Applications* 19(1), pp. 56–65.

Martinović, A., 2015. Inverse Procedural Modeling of Buildings. PhD thesis, KU Leuven.

Müller Arisona, S., Zhong, C., Huang, X. and Qin, H., 2013. Increasing detail of 3D models through combined photogrammetric and procedural modelling. *Geo-spatial Information Science* 16(1), pp. 45–53.

Müller, P., Wonka, P., Haegler, S., Ulmer, A. and van Gool, L., 2006. Procedural modeling of buildings. *ACM Transactions on Graphics* 25(3), pp. 614–623.

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., van Gool, L. and Purgathofer, W., 2013. A Survey of Urban Reconstruction. *Computer Graphics Forum* 32(6), pp. 146–177.

OGC, 2016. OGC CityGML Quality Interoperability Experiment. Technical Report OGC 16-064.

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

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

Peter, M., Becker, S. and Fritsch, D., 2013. Grammar supported indoor mapping. *In: Proceedings of the 26th International Cartographic Conference*, Dresden, Germany, pp. 1–18.

Rautenbach, V., Bevis, Y., Coetzee, S. and Combrinck, C., 2015. Evaluating procedural modelling for 3D models of informal settlements in urban design activities. *South African Journal of Science* 111(11/12), pp. 1–10.

Rodrigues, R., Coelho, A. and Reis, L. P., 2010. Data model for procedural modelling from textual descriptions. *In: IEEE Congress on Evolutionary Computation (CEC)*, IEEE, Barcelona, Spain, pp. 1–8.

Rosser, J., Morley, J. and Smith, G., 2015. Modelling of Building Interiors with Mobile Phone Sensor Data. *ISPRS International Journal of Geo-Information* 4(2), pp. 989–1012.

Sinning-Meister, M., Gruen, A. and Dan, H., 1996. 3D city models for CAAD-supported analysis and design of urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing* 51(4), pp. 196–208.

Smelik, R. M., Tutenel, T., Bidarra, R. and Benes, B., 2014. A Survey on Procedural Modelling for Virtual Worlds. *Computer Graphics Forum* 33(6), pp. 31–50.

Stadler, A. and Kolbe, T. H., 2007. Spatio-semantic coherence in the integration of 3D city models. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XXXVI-2/C43, pp. 8.

Steuer, H., Machl, T., Sindram, M., Liebel, L. and Kolbe, T. H., 2015. Voluminator—Approximating the Volume of 3D Buildings to Overcome Topological Errors. *In: AGILE 2015*, Springer International Publishing, pp. 343–362.

Stilla, U., Soergel, U. and Thoennessen, U., 2003. Potential and limits of InSAR data for building reconstruction in built-up areas. *ISPRS Journal of Photogrammetry and Remote Sensing* 58(1-2), pp. 113–123.

Strzalka, A., Bogdahn, J., Coors, V. and Eicker, U., 2011. 3D City modeling for urban scale heating energy demand forecasting. *HVAC&R Research* 17(4), pp. 526–539.

Sugihara, K. and Shen, Z., 2016. Automatic generation of 3D house models with solar photovoltaic generation for smart city. *In: 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, Muscat, Oman, pp. 1–7.

Suveg, I. and Vosselman, G., 2004. Reconstruction of 3D building models from aerial images and maps. *ISPRS Journal of Photogrammetry and Remote Sensing* 58(3-4), pp. 202–224.

Tsiliakou, E., Labropoulos, T. and Dimopoulou, E., 2014. Procedural Modeling in 3D GIS Environment. *International Journal of 3-D Information Modeling* 3(3), pp. 17–34.

Vanegas, C. A., Kelly, T., Weber, B., Halatsch, J., Aliaga, D. G. and Müller, P., 2012. Procedural Generation of Parcels in Urban Modeling. *Computer Graphics Forum* 31(2), pp. 681–690.

Vosselman, G. and Dijkman, S., 2001. 3D building model reconstruction from point clouds and ground plans. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XXXIV-3/W4, pp. 37–44.

Wonka, P., Wimmer, M., Ribarsky, W. and Sillion, F., 2003. Instant architecture. *ACM Transactions on Graphics* 22(3), pp. 669–677.

Zhang, W., Wang, H., Chen, Y., Yan, K. and Chen, M., 2014. 3D Building Roof Modeling by Optimizing Primitive's Parameters Using Constraints from LiDAR Data and Aerial Imagery. *Remote Sensing* 6(9), pp. 8107–8133.

Zhao, J., Ledoux, H. and Stoter, J., 2013. Automatic repair of CityGML LoD2 buildings using shrink-wrapping. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* II-2/W1, pp. 309–317.