Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings

Sjors Donkers[1,2], Hugo Ledoux[2], Junqiao Zhao[3], and Jantien Stoter[2]

[1]Grontmij Nederland B.V., the Netherlands
[2]Delft University of Technology, the Netherlands
[3]Tongji University, Shanghai, China

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

The code of the project is available at
`https://github.com/tudelft3d/ifc2citygml`

Although the international standard CityGML has five levels of detail (LODs), the vast majority of available models are the coarse ones (up to LOD2, ie block-shaped buildings with roofs). LOD3 and LOD4 models, which contain architectural details such as balconies, windows and rooms, nearly exist because, unlike coarser LODs, their construction requires several datasets that must be acquired with different technologies, and often extensive manual work is needed. We investigate in this paper an alternative to obtaining CityGML LOD3 models: the automatic conversion from already existing architectural models (stored in the IFC format). Existing conversion algorithms mostly focus on the semantic mappings and convert *all* the geometries, which yields CityGML models having poor usability in practice (spatial analysis is for instance not possible). We present a conversion algorithm that accurately applies the correct semantics from IFC models and that constructs valid CityGML LOD3 buildings by performing a series of geometric operations in 3D. We have implemented our algorithm and we demonstrate its effectiveness with several real-world datasets. We also propose specific improvements to both standards to foster their integration in the future.

1 Introduction

CityGML is a well established international standard for representing and storing three-dimensional (3D) city models [OGC, 2012]. It models the shape of buildings with five different levels of detail (LODs), Figure 1 illustrates the four (volumetric) LODs (LOD1 to LOD4); the non-volumetric

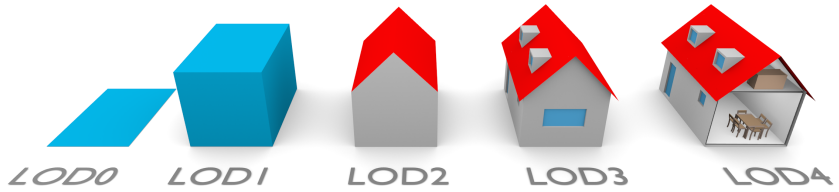


Figure 1: A building represented in LOD0 to LOD4 (Figure from Biljecki et al. [2014a]).

LOD0 is an horizontal footprint and/or roof surface representation for buildings. LOD1 is a block-shaped model of a building (with an horizontal roof); LOD2 adds a generalised roof and installations such as balconies; LOD3 adds, among others, windows, doors, and full architecture exterior; and LOD4 models the interior of the building, potentially with pieces of furniture. Although we do not have precise numbers, we can estimate that the majority of existing 3D city models are LOD1 and LOD2[1]. This is because in practice the automatic construction of LOD3 (and LOD4) models is more complex than that of LOD1 and LOD2 models (or even impossible). Indeed, LOD1 models can be readily constructed by extruding building footprints by their elevation [Ledoux and Meijers, 2011]—buildings' footprints are in most countries readily available, and the elevation can be obtained with airborne laserscanners or photogrammetric techniques. LOD2 models require more detailed roofs: Kada and McKinley [2009] achieve this by fitting templates of common roofs to the footprints and elevation points, and Elberink and Vosselman [2009] reconstruct a model of the roof by using segmentation of the LiDAR points and a graph-based approach. The construction of LOD3 models require however additional data to model architectural details such as windows and doors. These data are difficult—and expensive—to acquire with airborne sensors and often require terrestrial laserscanning surveys. As a consequence, most LOD3 models currently available have been constructed semi-manually, which is tedious and time-consuming.

The openings (windows and doors) and characteristics of the roof of LOD3 buildings are beneficial for several applications. The following are a few examples: (i) more realistic visualisation is possible, which can help improve city planning [Köninger and Bartel, 1998]; (ii) the energy demand can be better estimated if windows are considered [Krüger and Kolbe, 2012]; (iii) the prediction of house prices can be improved since these are based on the amount of sunlight and the view on the surroundings [Helbich et al., 2013]; (iv) the solar potential of rooftops is more realistic if the usable area of the roof is used, thus without dormers.

An alternative to constructing LOD3 models from raw data is to *convert* existing architectural models, which usually contain the necessary information. For many cities, there is already a limited number of such models available (mostly for new buildings, manually made by architects) in the international standard IFC (Industry Foundation Classes). As further explained in Section 2.1, it is an open data model in which all the details of a building are specified. We argue in this paper that the automatic conversion from IFC to CityGML LOD3 models is complex because of three main issues. First, both data models differ fundamentally: the mappings between the semantics are complex because different semantic information is attached to the geometrical primitives in the two models. Second, different paradigms and data structures are used to represent the geometries: boundary-representation (*b-rep*) for CityGML, versus constructive solid geometry (CSG) and sweep volumes in IFC. Third, CityGML LOD3 buildings should be modelled only with the surfaces visible from the outside, and these should form a closed 2-manifold. From a practical point of view, this means that for instance an exterior wall of a building should not have a 'thickness'. However, the surfaces forming the exterior of a building are not explicitly marked as such in an IFC file and cannot be deducted directly from the semantics of the objects [Benner et al., 2005]. As we show in Section 3, current conversion algorithms are mostly concerned with the first two issues. Although these result in *visually* acceptable CityGML LOD3 models containing valid geometries, *all* the IFC geometries are converted to an unorganised set of surfaces (`MultiSurfaces`) with limited semantics, and many of these are often located inside the buildings (eg the ones representing walls and beams, see Figure 2). Therefore, these models do not follow the LOD3 rules of CityGML and have a poor spatio-semantic coherence [Stadler and Kolbe, 2007]. As a consequence, their usability for many applications is limited (except for

[1]The website `www.virtualcitymodels.co.uk` has a overview of the models around the world.

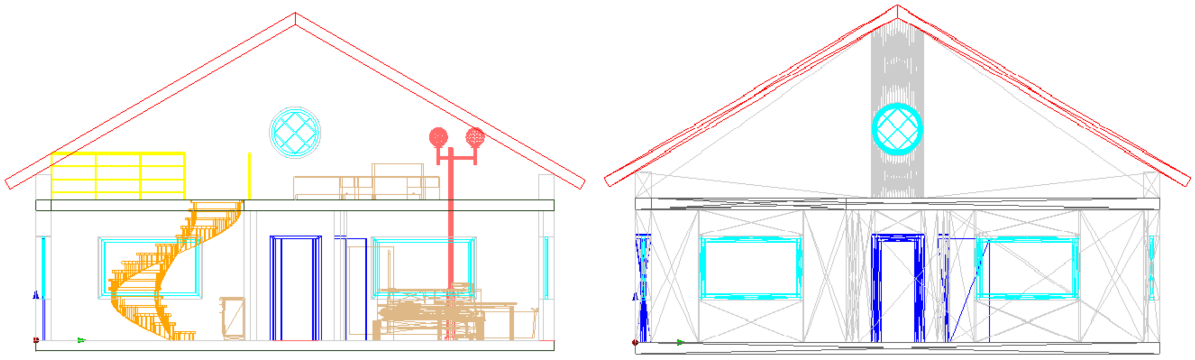


Figure 2: Cross-section of an IFC model (left) for which all the relevant geometries are converted to CityGML geometries (right model). Observe that the result is not a single envelope because several volumes are constructed (eg each slab of the roof is a volume).

visualisation).

In this paper we propose a novel algorithm for the automatic conversion of IFC buildings models into semantically rich and geometrically valid CityGML LOD3 buildings. Unlike the previous algorithms attempting to convert input information in the output model without additional reasoning, we developed an *output-driven* solution where the geometries are modified to construct valid LOD3 models. Our algorithm, described in Section 4 and illustrated in Figure 5, contains three main steps: (1) the filtering and the mappings of the semantics; (2) the geometric transformations needed to extract the exterior envelope of a building; (3) further geometrical refinements to ensure validity of the output model. For the first step we partly reuse existing work and extend it; the second and third steps are the main contributions of this paper. We have implemented our algorithm in C++ and the code is freely available under an open-source licence[2]. Section 5 presents the results we obtain with different IFC models, and shows that our approach does produce geometrically valid and semantically rich LOD3 models. Finally, we discuss in Section 6 the shortcomings of our approach, and we also propose some simple changes to the IFC standard that would yield a simpler and better conversion to CityGML

2 Modelling buildings with the IFC and CityGML standards

In this section, we briefly describe how buildings are modelled with the IFC standard and what a CityGML semantically rich and geometric valid LOD3 model is.

2.1 Industry Foundation Classes (IFC)

IFC is a standardised open data model used in multidisciplinary building projects for managing complex communication and information sharing processes throughout the life cycle of the building [Sebastian and van Berlo, 2010; ISO, 2013]. The most relevant IFC classes for building object types, and their relationships, are shown in Figure 3. These are based on Nagel and Kolbe [2007] and El-Mekawy et al. [2012]. It is important to be aware that an `IfcObject` and its subclasses can be recursively decomposed into other `IfcObjects`. While there are many other relationships possible, only two are relevant for CityGML: (1) `IfcRelContainedInSpatialStructure` to determine whether an object is part of a given building or the surrounding site; (2) `IfcRelDefinesByType` to check whether there is an `IfcTypeObject` which contains more information on the object. This relationship exposes the `PredefinedType`, an attribute—not mandatory but commonly used—available for several classes to further specify the type of object. The `PredefinedType` can also directly be part of the object itself, thus without the use of an `IfcTypeObject`.

IFC allows three representation paradigms for volumetric objects:

- **CSG:** an object is represented as a series of Boolean operations (union, intersection and difference) of objects, which are typically simple shapes such as cylinders, cones, spheres or pyramids (see Requicha [1982] for more details).

[2] `github.com/tudelft-gist/ifc2citygml`

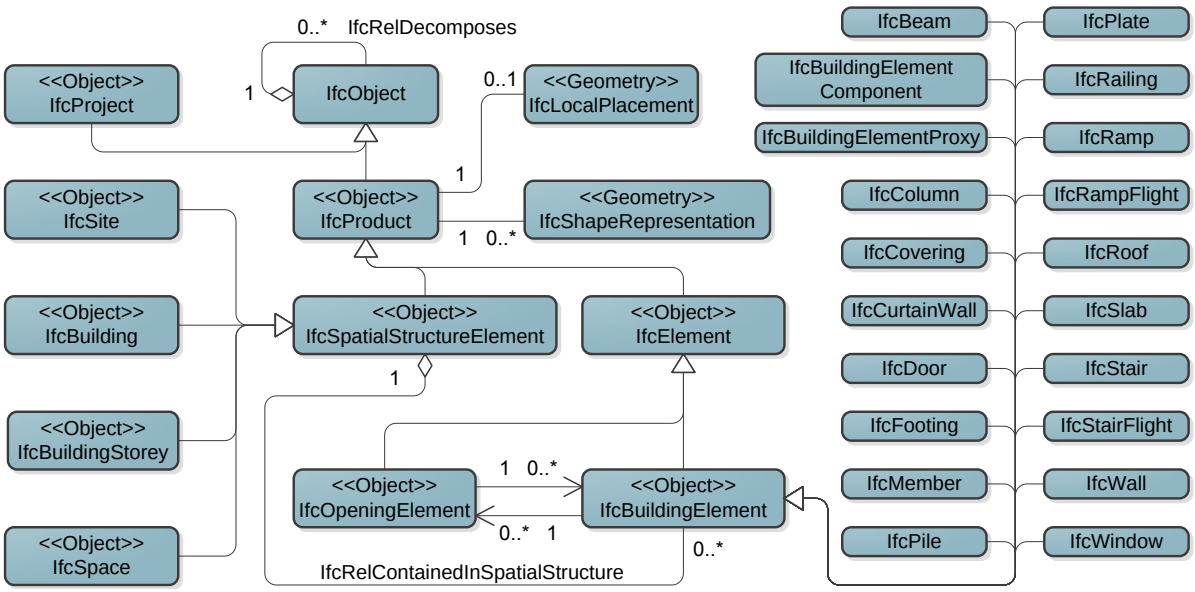


Figure 3: UML diagram for the IFC classes most relevant for CityGML LOD3 models.

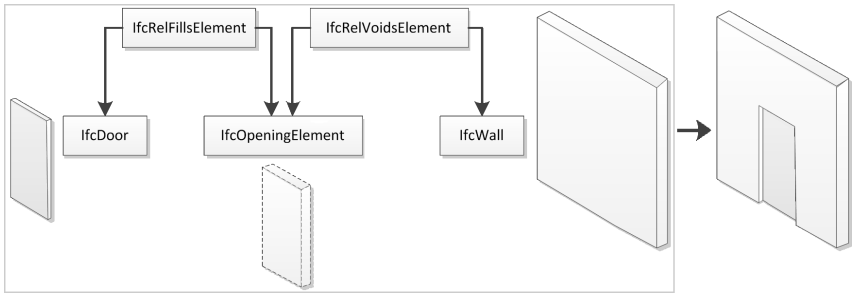


Figure 4: CSG stores geometries implicitly: the wall is subtracted by the opening element using the Boolean difference. The door is then placed within the gap in the wall (Figure adapted from Beetz [2012])

- **Sweep volumes:** a solid can also be defined by a 2D surface (a circle, a rectangle or an arbitrary polygon) and a path [Wang and Wang, 1986], along which the surface is extruded.
- **b-rep:** an object is represented by its bounding surfaces. While relatively rare in IFC, it is used for complex objects only such as `IfcDoor` or `IfcWindow`.

It should be noticed that the first two representations are *implicit*: only the parameters to construct the geometry are stored. To be able to simply visualise a CSG object, geometric transformations are necessary. In practice, most IFC models are built using sweep volumes and CSG [El-Mekawy and Östman, 2010]. Figure 4 illustrates a concrete example for a CSG; the (implicit) geometries of a door and a wall are defined by the relations between them. Constructing explicit geometries to be used in a CityGML model does not always yield a unique solution since basic shapes need to be *discretised*. A sphere should for instance be converted to a set of polygons; however, the number needed is not specified in the standard.

2.2 CityGML

CityGML is based on a number of standards from the ISO191*xx* family and is used both as an information model and a data format. CityGML as a data format is implemented as an application schema for the Geography Markup Language (GML)[3] [OGC, 2004]. Its aim is to define the basic feature classes for the most common geographical features and the relationships between them. The most important CityGML feature for this paper, a building, can be modelled

[3] CityGML uses version 3.1.1 of GML

with three classes (`Building`, `BuildingParts` and `BuildingInstallations`), and a `Building` can recursively consist of several `BuildingParts` and can have multiple `BuildingInstallations`.

Unlike IFC in which three representation paradigms are possible, volumetric geometries in CityGML are solely represented with a *b-rep*. While GML allows the use of non-linear geometries, CityGML uses linear/planar ones only. An LOD3 `Building`, or a `BuildingPart`, must not have features located inside its exterior envelope, ie it must be represented only with surfaces visible from the outside. For representing a building, the natural choice is a `gml:Solid` (without interior shells) because it is a volumetric object that must be watertight. Using a `gml:Solid` however implies that the exterior envelope is a 2-manifold, and while the vast majority of buildings can be modelled this way, there are buildings whose exterior envelope self-intersects (see for instance Figure 12). For these, no volume can be constructed (buildings cannot be stored as `gml:MultiSolid` in CityGML) and the exterior boundary must be stored with a `gml:MultiSurface`, ie a set of unorganised surfaces.

The set of planar surfaces representing a `gml:Solid` must respect several rules: no two surfaces must overlap or touch at any other locations than an edge or a vertex; the orientation of the surfaces must be consistent, etc. (Ledoux [2013] gives a complete list of properties).

From a semantics perspective, the boundary surfaces of an LOD3 object should be one of these types: `WallSurface`, `RoofSurface`, `ClosureSurface`, `GroundSurface`, `OuterCeilingSurface` or `OuterFloorSurface`. While these are not mandatory in CityGML, we argue in this paper that they are crucial for an LOD3 model to be useful in practice for different applications. These boundary surfaces can also have `Openings`, which are either `Doors` or `Windows`. If an `Opening` is located inside a bounding surface, then the surface must be modelled with an interior ring (filled with surfaces representing the `Openings`). It should be noticed that the surfaces representing the `Openings` must be part of the bounding envelope of the building. Balconies, chimneys and verandas should be modelled as `BuildingInstallations`, and not as a part of the boundary surfaces for the exterior envelope of a building.

3 Related work

Most related work is concerned with the conversion of geometries and with the mapping of the semantic attributes in the two models (often by converting all the geometries of one object), and little attention has been paid to the meaningful conversion of IFC data into LOD3 CityGML data. As shown in Section 4, this requires complex *geometrical* processing of the input IFC data, rather than simply a mapping of classes.

A few programs offer the possibility to convert IFC models into CityGML models. Three examples are: the Building Information Modelserver[4], IfcExplorer[5] and Safe FME[6]. All of them allow the users to convert IFC models to CityGML at different LODs. The users can in some cases choose which IFC objects should be used, however, the geometrical transformations to extract the exterior shell are not implemented and the results are neither semantically rich nor geometrically valid (as defined in the previous section).

Hijazi et al. [2010] built an open-source web-GIS in which IFC objects can be imported after having been converted to b-rep models. Unfortunately, only sweep-volume objects can be converted and all the geometries are kept (thus resulting in non-manifold buildings). De Laat and van Berlo [2011] developed an CityGML ADE (application domain extension), called GeoBIM[7], so that new semantic classes defined in IFC are added in a CityGML model. However, no geometric manipulation is performed. El-Mekawy et al. [2011] and El-Mekawy et al. [2012] propose a unified model in which all the semantic properties of IFC and CityGML are present, and they propose using bidirectional mappings for all the semantic classes relevant to IFC and CityGML. However, they extract the geometries separately from the two models and need manual editing to enrich the result. To obtain semantically rich and geometrically valid LOD3 building models, more than a selection of the input objects/geometries must be performed, because walls, beams and windows are represented by volumes in IFC (see Figure 2). A direct conversion results in a set

[4] `http://bimserver.org`

[5] `http://iai-typo3.iai.fzk.de/www-extern/index.php?id=1566&L=1`

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

[7] `http://www.citygmlwiki.org/index.php/CityGML_GeoBIM_ADE`

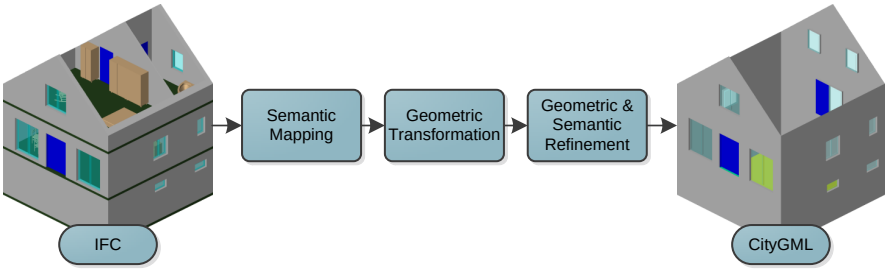


Figure 5: General workflow diagram of our algorithm. Notice that the roofs of both models have been removed so that the interior of the model is visible.

of volumes and surfaces having hardly any meaning, except for visualisation purposes. Further geometrical manipulations are required to obtain geometries appropriate for 3D city modelling.

Benner et al. [2005] describe the general steps needed to convert IFC models to an alternative data model closely related to CityGML (the QUASY model). They first map the semantics from IFC to QUASY and select the relevant boundary objects, and then the outer visible surfaces are extracted by selecting a subset of the input objects. For the (equivalent of) the LOD3 model, they discard geometries inside the building by projecting each floor of a building to horizontal and vertical 'footprints' and keeping only those touching the envelope. This technique does not close possible gaps (which are common in practice, see Section 5) and may yield unclosed shells. Moreover, while the output models appear to be LOD3 models, the walls and the roof are volumes. Nagel [2006] and Nagel and Kolbe [2007] provide a method to extract valid LOD1 from IFC models. Each floor of a building is projected to the ground to obtain its spatial extent (with a series of 2D Boolean operations), and floors are extruded to construct one geometrically valid block-shape model. Although we use different algorithms and methods, we describe in Section 4 a conceptual extension of this approach to construct LOD3 models. We do not project each floor to 2D, instead we work directly in 3D to extract the exterior envelope of a building. We also recover from small errors (eg gaps) that are often present in IFC models.

4 Our IFC to CityGML LOD3 conversion algorithm

As shown in Figure 5, our algorithm contains three main steps: (1) the filtering and mappings of the semantics; (2) the 3D geometric transformations to extract the exterior envelope of a building and store it as a `gml:Solid` or a `gml:MultiSurface`; (3) the refinements to ensure that the output is valid.

4.1 Semantic filtering and mappings

An LOD3 building in CityGML can have semantic properties for both the solid and the surfaces of this solid. As described in Section 2, there are six possibilities for a boundary surface. IFC has a different structure for storing the semantics and objects are connected via a network of relations. For the extraction of CityGML semantics from an IFC object, the IFC class and the type of the object are in most cases sufficient. However, there are cases for which the network of relations needs to be traversed in search of the optimal semantics. In brief, to determine what the semantics are in CityGML for one particular surface, we need a combination of multiple semantic values from IFC and certain geometric properties are required.

For our conversion, we partly reuse the filtering and the mapping methods from El-Mekawy and Östman [2010] and de Laat and van Berlo [2011]. However, with these, an IFC object (a solid) is mapped to a set of surfaces and all the of them get the semantics of the solid. This is problematic because a wall (modelled as a solid in IFC) forming the exterior of a building will have faces that are outside and some that are inside. We have therefore modified several mappings and extended them with new ones such that the surfaces from a solid can get different semantics. We have also defined how and when the network needs to be traversed in search of

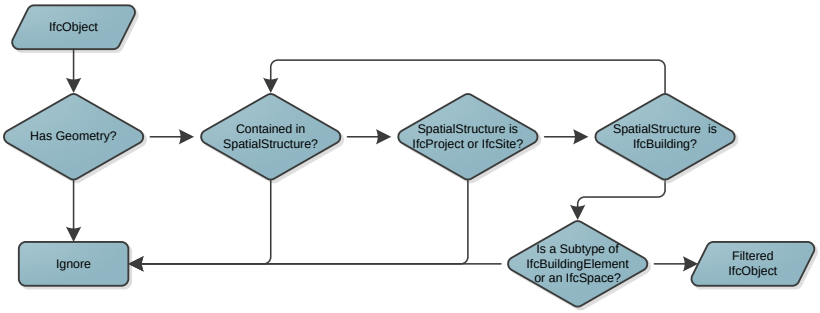


Figure 6: Workflow diagram for the filtering of input IfcObjects.

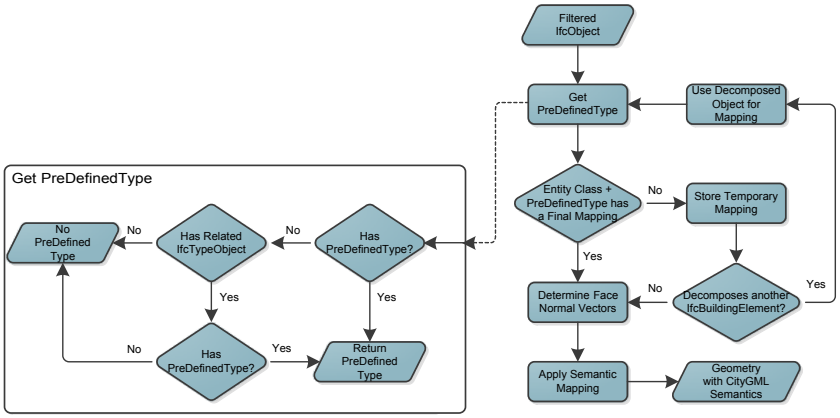


Figure 7: Workflow diagram for the mapping of semantics

better semantics and how to determine whether an IFC object should be part of the conversion or not. To determine the proper semantic for a surface, we use the following criteria:

1. whether it is semantically contained in a building, or not;
2. whether the entity class belongs to a building type;
3. the `PreDefinedType` attribute;
4. whether it decomposes another object, or not;
5. the normal vector of a surface (its orientation).

Figures 6 and 7 give the overview of the filtering and the mapping workflow.

Filtering. There are around 900 classes defined in the IFC schema. However, the most relevant classes for CityGML are only a subset of these: `IfcSpace` and all the subtypes of `IfcBuildingElement`. All other classes either represent movable objects or are abstract classes without geometry. For each `IfcObject` in an IFC file, we verify whether it has a geometry and whether it is contained inside a building; for the latter the `IfcRelContainedInSpatialStructure` relation is used recursively. Filtering these objects leaves us to deal with only objects having meaningful mappings in CityGML.

Mappings. For every surface of the filtered `IfcObjects`, the mappings determine the CityGML boundary surface type. We perform these based on three criteria: (1) the normal direction of the surface; (2) the type of object; and (3) its relations to others (`PredefinedType`). The mappings are summarised in Table 1. For some combinations the mapping is certain and thus final, for example when the IFC type is `IfcRoof`. In other cases, it is possible that more information is needed, we name these cases 'temporary'. For example, an `IfcPlate` on its own could potentially

Table 1: Semantic mappings from IFC to CityGML LOD3.

IFC entity type	**IFC PredefinedType**	**Temporary/ Final**	**CityGML class (based on surface normals)** up	horizontal	down
IfcBeam		Temporary	- BuildingInstallation -		
IfcBuildingElementComponent		Temporary	- BuildingInstallation -		
IfcBuildingElementProxy		Temporary	- BuildingInstallation -		
IfcColumn		Temporary	- BuildingInstallation -		
IfcCovering	CEILING	Final	OuterFloorSurface	WallSurface	OuterCeilingSurface
	FLOORING	Final	OuterFloorSurface	WallSurface	OuterCeilingSurface
	ROOFING	Final	RoofSurface	RoofSurface	OuterCeilingSurface
	Others	Temporary	WallSurface	WallSurface	WallSurface
IfcCurtainWall		Final	WallSurface	WallSurface	WallSurface
IfcDoor		Final	Door	Door	Door
IfcFooting		Final	OuterFloorSurface	WallSurface	GroundSurface
IfcMember		Temporary	WallSurface	WallSurface	WallSurface
IfcPile		Temporary	- BuildingInstallation -		
IfcPlate		Temporary	WallSurface	WallSurface	WallSurface
IfcRailing		Final	- BuildingInstallation -		
IfcRamp		Final	- BuildingInstallation -		
IfcRampFlight		Final	- BuildingInstallation -		
IfcRoof		Final	RoofSurface	RoofSurface	RoofSurface
IfcSlab	FLOOR	Final	OuterFloorSurface	WallSurface	OuterCeilingSurface
	ROOF	Final	RoofSurface	RoofSurface	OuterCeilingSurface
	LANDING	Temporary	- BuildingInstallation -		
	BASESLAB	Final	OuterFloorSurface	WallSurface	GroundSurface
	USERDEFINED	Temporary	OuterFloorSurface	WallSurface	OuterCeilingSurface
IfcStair		Final	- BuildingInstallation -		
IfcStairFlight		Final	- BuildingInstallation -		
IfcWallSurface		Final	WallSurface	WallSurface	WallSurface
IfcWallSurfaceStandardCase		Final	WallSurface	WallSurface	WallSurface
IfcWindow		Final	Window	Window	Window
IfcSpace		Final	ClosureSurface	ClosureSurface	ClosureSurface

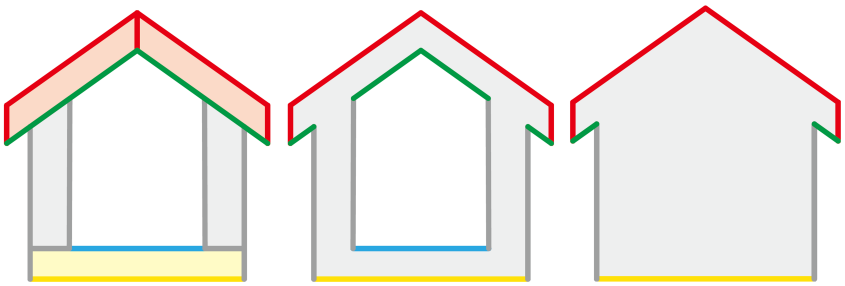


Figure 8: Extraction of the exterior envelope from a set of IFC solids. The colours represent the semantics: lines for boundary surfaces, and area for volume semantics.

be anything and is thus temporarily mapped to a `WallSurface`. However, the `IfcPlate` might decompose an `IfcWindow`, in which case the `IfcPlate` should be mapped to `Window` in CityGML. As such, when a combination is marked as temporary, the mapping of the parent object which the active object decomposes is used. This search is done recursively until a final mapping is found, or when the object does not decompose another parent object, or when the parent object is no longer a subtype of `IfcBuildingElement`. Notice for example that a distinction can be made between columns and beams which are detached from the building, and columns and beams which are part of a wall and thus potentially part of the exterior shell of the building. In case no final mapping could be determined, the last temporary mapping is used.

While CityGML allows the use of boundary surface properties, we do not assign them for `BuildingInstallations`. The main reason is that the only possibilities (the six previously mentioned) are not meaningful for the objects such as stairs, columns, ramps and beams. Only for dormers and chimneys they would make sense, but at this moment it is impossible to detect these in the IFC input file because there are no specific objects defined in IFC for these. If the semantics for balconies and dormers were added to the IFC standard, then for those `BuildingInstallations` the boundary surface properties could be assigned using exactly the same algorithm as for `Buildings`.

4.2 Extraction of the exterior envelope

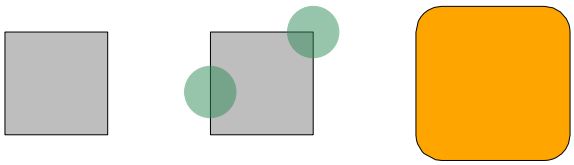
The input of this second step is the output of the previous step: a set of solids for which each boundary surface has been assigned a CityGML semantics. If the solids were represented in IFC with CSG, these have been converted to a b-rep using standard methods [Tawfik, 1991].

As shown in Figure 8, the main idea is to first construct the space partitioning[8] of the geometries, then perform a Boolean union operation on all pair of solid that are face adjacent, and finally to remove all the geometries inside the outer boundary. The last step can be performed by keeping only the 2-manifold having the largest volume, or by using the topological relationships, eg see Diakité et al. [2014] for details. Notice that it is important that the semantics assigned to the faces are maintained.

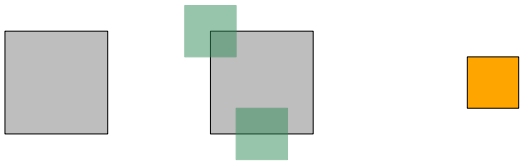
In practice, this approach often fails: we have indeed noticed during our experiments (see Section 5) that buildings represented with the IFC standard are rarely geometrically and topologically perfect. For instance, vents, chimneys and utilities often penetrate the exterior of the buildings, and small gaps between geometries are often present (eg between a roof slab and a wall). These errors have most likely been created (unintentionally) by an operator during the modelling phase, or can be the results of floating-point arithmetic or limited precision of computers [Schirra, 1997]. Furthermore, it should be observed that these errors are often not visible at the scale the data is usually visualised, and are often not known to the practitioners [Laurini and Milleret-Raffort, 1994].

Morphological operators. To recover from these (unavoidable) imperfections, we use the mathematical morphology theory in 3D [Serra, 1982]. Similarly to Zhao et al. [2012] where 3D buildings and buildings' parts are merged together to generate different levels of detail, we use the

[8] A space partitioning is the generalisation to 3D of a planar partition, ie a subdivision of the 3D Euclidean space into non-overlapping volumes.

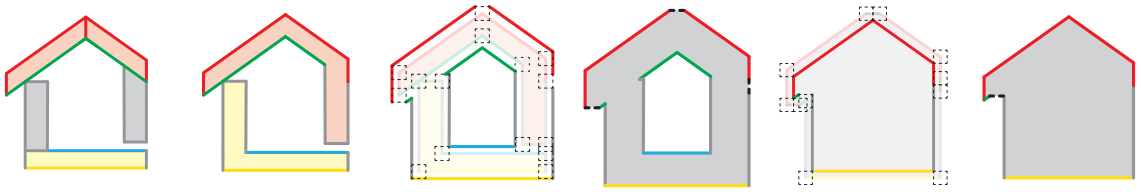


(a) dilation



(b) erosion

Figure 9: Two morphological operators in 2D applied on A (dark grey) with a structuring element B (green), the result is the orange subset of $\mathbb{R}^2$.



(a) input (b) union (c) dilation (d) result (e) erosion (f) final result

Figure 10: Closing operator applied to imperfect geometries.

geometric operations dilation, erosion and closing to close the small gaps and union together different geometries. Let A and B be two solids, which are subset of $\mathbb{R}^3$, the Euclidean space in three dimensions. As shown in Figure 9, the dilation of A by B (B is called the *structuring element* in this case) is defined as follows and is equivalent to the Minknowski sum of A and B:

$$A \oplus B = \{a + b \,|\, a \in A, b \in B\}$$

It is a subset of $\mathbb{R}^3$ formed by adding each point in A to each point in B. If the structuring element B is a sphere centered at the origin, the dilation is equivalent to the buffer operation in GIS (by a value equal to the radius of B). The erosion of A by the structuring element B is the dual of the dilation, and is defined as $A \ominus B$. If B is centered at the origin, it can be understood as shrinking A by moving B inside A, thus $A \ominus B = (A^- \oplus B)^-$, where A^- is the complement of A.

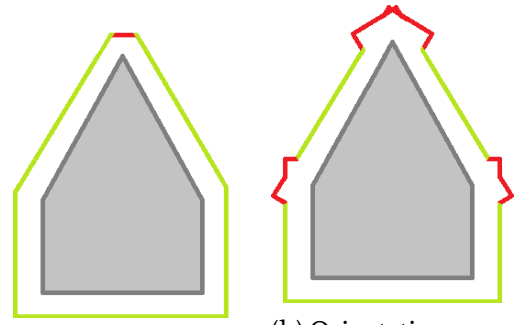
The closing of A by B is the dilation of A by B followed by the erosion of the result by B:

$$A \bullet B = (A \oplus B) \ominus B$$

It can be used to merge objects that lie close to each others (let us say at a distance of ϵ) since the dilation will make them "touch", or overlap, if a structuring element sphere has a radius greater than $\frac{\epsilon}{2}$. We use a slightly modified version of morphological closing to remove small gaps in the input geometries: the interior geometries are removed after the dilation operator. We do this because otherwise the gaps could 'reappear' during the erosion operation. Figure 10 shows the extraction process for a set of geometries representing a building; notice that there is a gap in the exterior envelope of the building, see Figure 10a. It should be observed that the closing operation did modify slightly the input geometries. For instance, the geometry of the roof overhang has been modified (part of it is not an horizontal surface anymore).

We discuss further in Section 5 the artefacts that our approach creates.

Shape, size and orientation of the structuring element. Because man-made buildings usually have surfaces perpendicular to each others, we use a cubical structuring element. Besides yielding



(a) Constant orientation.

(b) Orientation aligned with surfaces.

Figure 11: Dilation of a simple building (grey) with a cubical structuring element.

geometries that are not perpendicular from each other, a spherical structuring element would have a negative influence on the computation time because a higher number of linear geometries would be needed to approximate the sphere.

The size and the orientation of a cubical structuring element will have an influence on the closing operator. From our practical experience with several IFC datasets, a size between 100mm and 300mm (this is the length of one edge of the cube) was found to give the most satisfactory results. A larger width could collapse narrow windows to lines, or merge several windows close to each others into one large window. If a model requires a different value, the size of the cube can be defined by the user in our prototype implementation.

As described in Boeters [2013], there are two strategies to assign the orientation to a cubical structuring element: (1) constant for the whole model (the predominant normals of all the surfaces); (2) aligned with each surface. The result for a simple building is shown in Figure 11. Observe in Figure 11b that although the dilation distance is constant for all surfaces, artefacts are created whenever there is a convex corner with an angle $\theta \neq 90°$. Since these artefacts do not disappear during the erosion, a surface aligned strategy should not be used. In the prototype implementation, we use a constant orientation, and we assume that buildings are vertical, ie the cube is rotated only along its z-axis to match the building's main orientation.

Maintaining the semantics of the surfaces. During dilation of the geometries, the semantics of a surface are transferred only to the parallel surface, which results in surfaces having no semantics. See for instance the horizontal surface created at the apex of the roof or under the roof overhang in Figure 10d (black surfaces). Such surfaces disappear during the erosion (if an appropriate size for the structuring element is used). If two or more adjacent input surfaces with different semantics are in the input, their dilated surfaces overlap. We resolve this by assigning no semantics to the overlapping section since during erosion the overlap will disappear.

4.3 Refinements to produce a valid CityGML LOD3 building

While the previous two steps permit us to extract a surface representation of a building stored in IFC, it is possible that the resulting surfaces do not form a 2-manifold and that some surfaces do not have semantics. We describe in the following two refinements methods to ensure that models are geometrically valid and that every face has the proper semantics.

4.3.1 Geometric and topological refinements

At this point in the algorithm, a non-manifold envelope, eg one where an edge is incident to more than two surfaces and/or where the surface self-intersects at a vertex (see Figure 12), could occur due to the input geometries not forming a 2-manifold.

Mäntylä [1988] proposes to duplicate the non-manifold vertices and edges, and to move them by an infinitesimal value so that they do not touch anymore. This implies that the geometry of an object is slightly changed, and is therefore not optimal. This is especially true for our approach since we assign semantics based on the surfaces' normals. An alternative solution is to

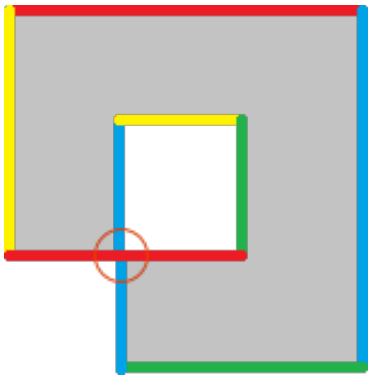


Figure 12: Top view of a non-manifold object, the colours represent the semantics of each surface, and the circle shows an edge having >2 incident faces.

decompose the object into a set of 2-manifold, eg into convex objects [Chazelle, 1984]. However, a CityGML building represented with a `gml:Solid` or `gml:CompositeSolid` must have an exterior envelope that is 2-manifold. The solution is that when a non-manifold envelop is detected, a `gml:MultiSurface` is used for all the surfaces.

4.3.2 Refinements of the semantics

As shown in Figure 13, we assign semantics to a surface based on two criteria: (1) the direction of

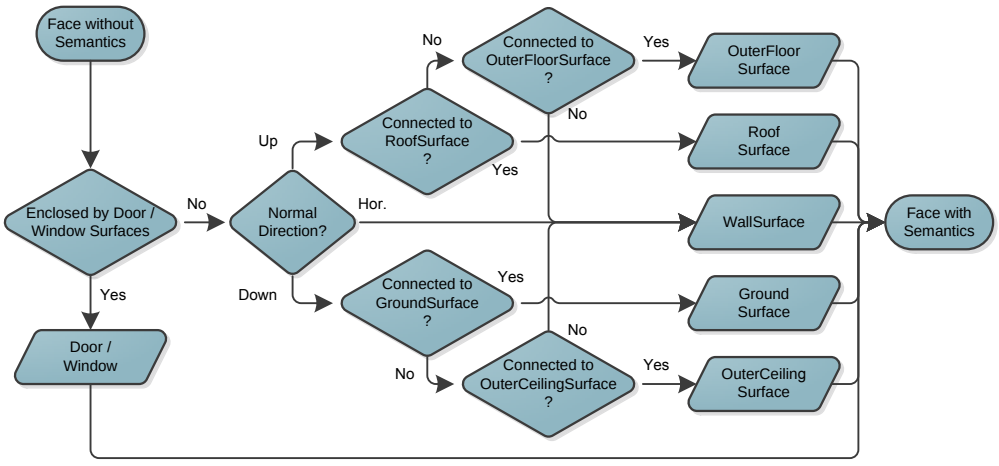


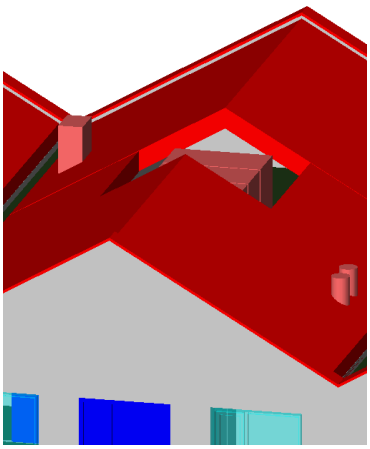
Figure 13: Workflow diagram for determining the semantics of faces without any.

its normal; and (2) whether it is contained inside surfaces having a specific semantics. A normal can either be horizontal (for walls), pointing upwards (ceilings and roof), or downwards (floor)—the rules for certain semantics are summarised in Table 1. We test the containment by grouping surfaces based on their normals, and verifying if the border of this group is incident to specific surfaces. Window and door `Openings` in CityGML are required to be part of a boundary surface (eg `WallSurface`).

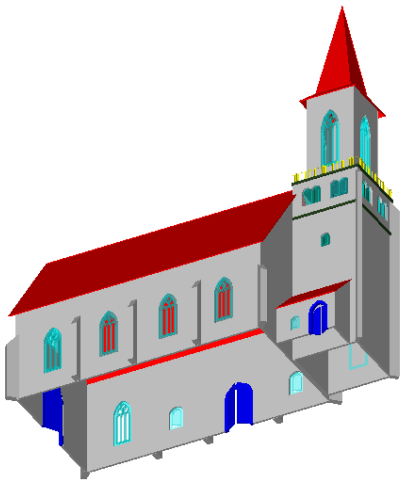
5 Implementation and experiments

We have implemented our approach in C++ using the free and open-source libraries *IfcOpenShell*[9] and *CGAL*[10], and we have released our code as open-source[11]. *IfcOpenShell* is used to read the

[9] `http://ifcopenshell.org`
[10] The computational geometry algorithms library: `http://www.cgal.org`
[11] Available at `https://github.com/tudelft-gist/ifc2citygml`



(a) Building where part of the roof is missing.



(b) A church missing a base slab

Figure 14: Missing geometries in the input IFC are problematic for the conversion to LOD3 buildings.

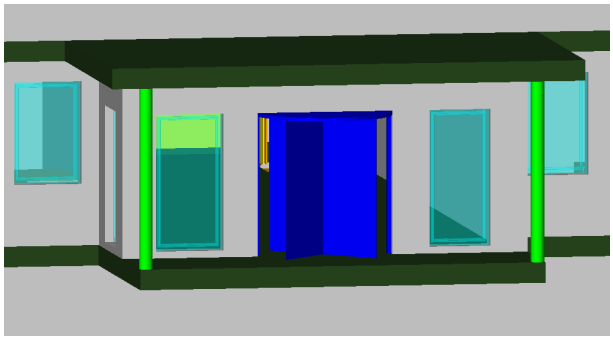


Figure 15: The revolving entrance door of the building is not closed, which causes the exterior surface of the building to contain a hole.

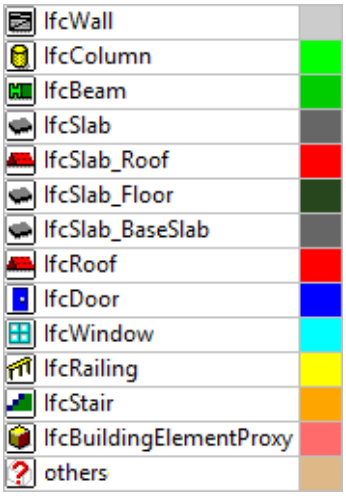
IFC files and to perform the geometric operations necessary to extract planar surfaces and thus create a b-rep representation. These are then processed with the package `Nef_polyhedron_3` of CGAL, which permits us to perform the geometric operations described in Section 4. We have developed ourselves the module to write CityGML files.

The prototype reads IFC files complying to the IFC2x3 specifications. The geometries in IFC should represent physical objects such that the objects can be reconstructed individually, and the solids must not overlap unless a Boolean difference operation is specified between them in the file. If there are overlapping solids, the semantic mapping might be wrong (there can be two different mappings for the same surface). The geometries need not be valid according to ISO19107, as the prototype processes them and even recovers from small gaps. Furthermore, the geometries must represent the complete exterior of the building, including the base slab. If there are major parts of the building missing, the closing operations cannot recover without introducing artefacts (see two examples in Figure 14). Observe that this implies that the geometries should represent doors and windows in their 'closed state'; this also holds for revolving and sliding doors (see Figure 15). The main requirements for the semantics in the IFC file are that (1) the input objects should be subtypes of `IfcBuildingElement`, and (2) they should be (indirectly) contained in a `IfcBuilding` spatial structure.

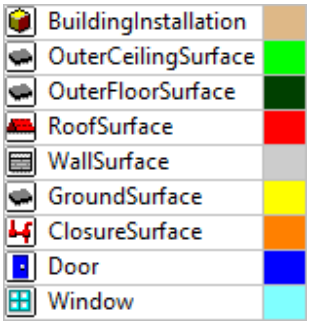
We have tested our prototype with several publicly available IFC datasets. Table 2 shows a selection of them. These were generated with different software packages, and range from a simple house to an office building. With our prototype implementation, all these models were successfully converted to geometrically and semantically valid CityGML LOD3 models. The models were first processed without the closing operation. If the result was not a closed envelope then a cubical structuring element of $300mm$ was used. All the errors in the input models were fixed in

Table 2: Datasets used for the experiments. `BInst` the number of `BuildingInstallation` created.

	input IFC			output CityGML			
	Fig.	# entities	size	`size`	`BInst`	closing	time
FZK-House	17	111	1.5MB	0.7MB	6	no	272s
FJK-House	18	274	13.9MB	1.1MB	12	yes	217s
Haus-G-H	19	301	4.3MB	5.3MB	4	yes	1672s
Model-4351	20	442	7.0MB	2.8MB	7	no	1041s
Office Building	21	1201	2.7MB	2.3MB	2	no	612s



(a) Legend for the most common objects in IFC

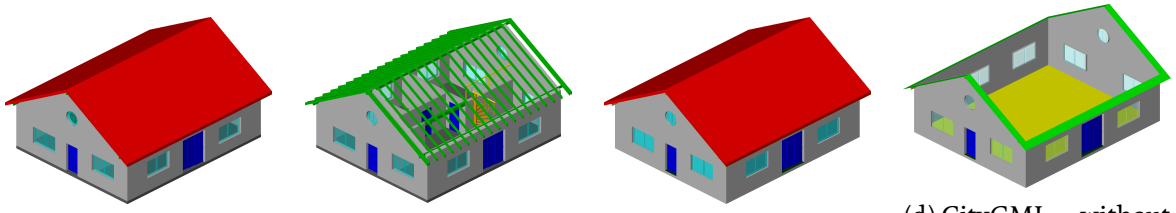


(b) Legend for the CityGML models

Figure 16: Legends for both the IFC and the CityGML models

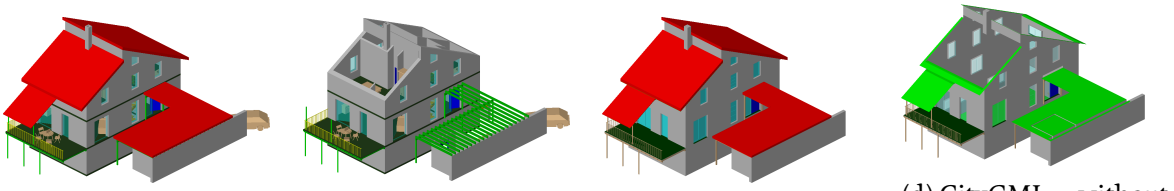
this way. However some artefacts in the output models were introduced, we demonstrate a few of these in Figure 22.

We have validated the output CityGML both geometrically and semantically. To ensure that the geometries are valid according to the international standard ISO19107, we used the algorithm and the implementation described in Ledoux [2013]. For the semantics, we first ensured that all faces had some semantics, and then we verified them visually. Figures 17- 21 show the input IFC models and the generated CityGML models. These were made using the software *FZK Viewer 4.0* [Benner et al., 2013b], and for all the figures the semantics of the surfaces is depicted using colours (see Figure 16).



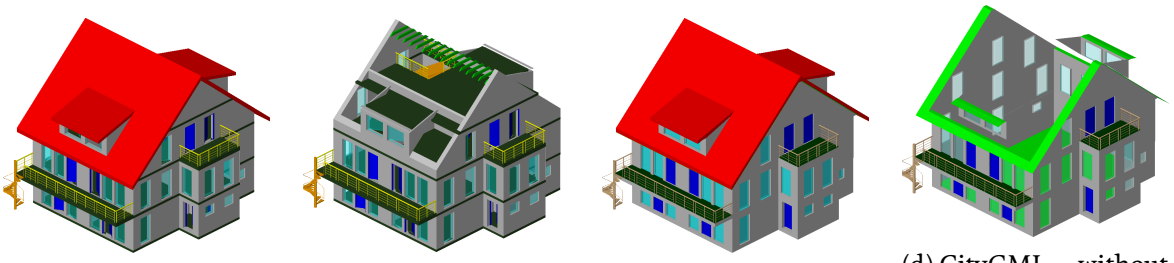
(a) input IFC (b) IFC without roof (c) output CityGML (d) CityGML without roof

Figure 17: **FZK-House** dataset, available at `www.iai.fzk.de/www-extern/index.php?id=1174&L=0`



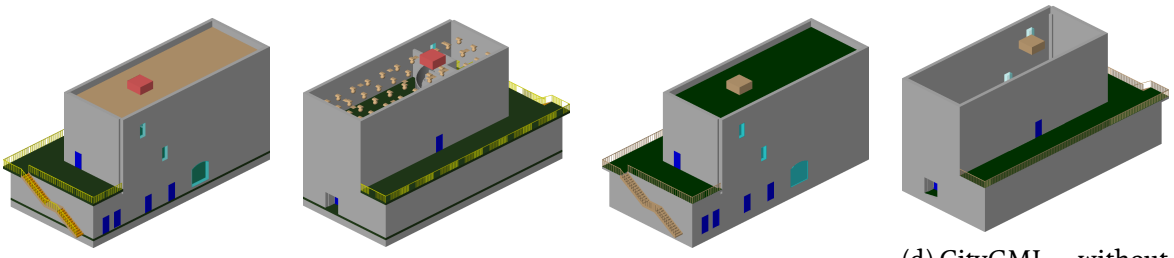
(a) input IFC (b) IFC without roof (c) output CityGML (d) CityGML without roof

Figure 18: **FJK-House** dataset, available at `www.iai.fzk.de/www-extern/index.php?id=1167&L=0`



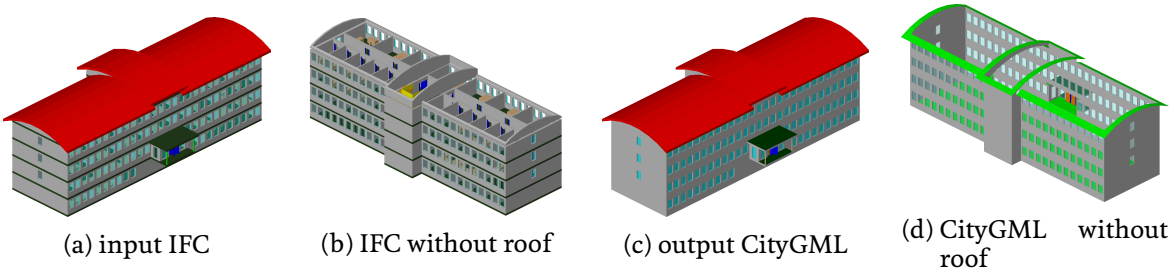
(a) input IFC (b) IFC without roof (c) output CityGML (d) CityGML without roof

Figure 19: **Haus-G-H** dataset, available at `code.google.com/p/bimserver/source/browse/trunk/TestData/data/AC9R1-Haus-G-H-Ver2-2x3.ifc`



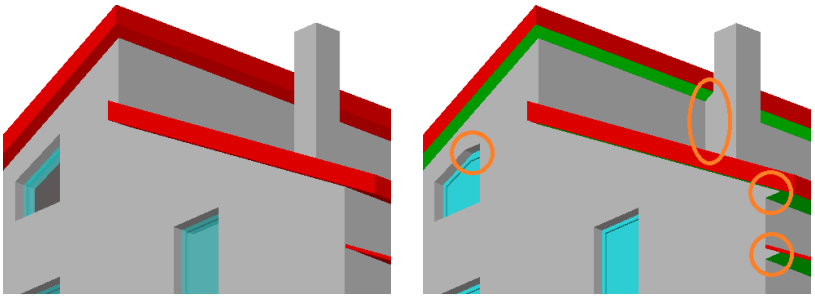
(a) input IFC (b) IFC without roof (c) output CityGML (d) CityGML without roof

Figure 20: **Model-4351** dataset, available at `code.google.com/p/bimserver/source/browse/trunk/TestData/data/4351.ifc`



(a) input IFC (b) IFC without roof (c) output CityGML (d) CityGML without roof

Figure 21: **Office-Building** dataset, available at `www.iai.fzk.de/www-extern/index.php?id=1184&L=0`



(a) input IFC model (b) Output CityGML model

Figure 22: Artefacts occur at concave parts of the geometry that are not aligned with the structuring element

We analyse briefly the results of our IFC to CityGML LOD3 conversion for each model:

FZK-House The IFC model has a roof support structure that overlaps with the roof geometry itself (which is not realistic). However, it does not cause any problems as it is composed of `IfcBeams` which are mapped as `BuildingInstallation` during the conversion (six are created). The input model contains no errors, and thus the closing operation was not necessary.

FJK-House Closing with a structuring element of 300mm is required, otherwise several rooms in the buildings would not be removed. The carport, the balcony and the chimney are part of the exterior shell in the CityGML model, while these should be `BuildingInstallations`. The input semantics of IFC are not expressive enough to permit us to detect these. The 12 `BuildingInstallations` are the beams supporting the carport.

Haus-G-H Closing is also required otherwise not all rooms would be removed. However, this causes artefacts under the roof overhangs, as shown in Figure 22. Aside from the missing 'GroundSurface' the semantics could be better if also the balcony and dormer were extracted as 'BuildingInstallations'.

Model-4351 No closing was necessary. The input model does not have any objects related to a roof. Since the highest slab is surrounded by a balustrade we assume that it is meant as a walkable surface. We believe the resulting `OuterFloorSurface` to be appropriate. The `BuildingInstallations` are the stairs and fences around the building.

Office Building While the input geometry contains a big hole caused by a revolving door not being closed (see Figure 15), this does not cause problem as the entrance hall is modelled with an `IfcSpace` (which effectively closes the gap). Although this is not the preferred way to model the entrance, it is allowed by IFC. The only issue is that the slab above the entrance lacks semantics in the input, and is converted to `FloorSurface` because of the orientation of the normal (while it should be a `RoofSurface`).

6 Conclusions and discussion

This paper reports on our research to develop a method to automatically convert IFC building models to geometrically valid and semantically rich LOD3 CityGML models, while also adhering to the ISO19107 standard for volumetric geometries. The validity requirements for CityGML LOD3 have been defined as well as the semantic information and geometric operations needed for the conversion. To demonstrate the validity of our approach, a prototype has been implemented and the output models created by the prototype have been validated. In contrast to manually reconstructed LOD3 CityGML models, the automatic conversion for exiting IFC data to CityGML makes it feasible to create highly detailed city models that are optimised for analyses. Such city models can be kept up-to-date by merging new IFC building models.

6.1 Recommendations for both standards

From our experiments, we can conclude that limitations in our conversion algorithm occur mostly due to semantic information missing in the IFC file. By making relatively small adjustments to the IFC standard, IFC and CityGML could be better aligned.

One of these would be to define in IFC which parts of a building should be part of the exterior shell, and how it can be stored as a special `IfcSpace`. It is already possible to define such a space, but there is no standardized way of storing this information. Also, for `IfcSpaces`, in general it is useful to know not only whether a space touches the exterior, but also whether it is contained within the building. For example the space for a balcony can then be ignored, whereas its geometry would otherwise be used as `ClosureSurface`.

Another recommended enrichment to IFC would be for balconies and dormers since it is currently impossible to extract them as `BuildingInstallations` without using complex feature recognition techniques. The reason is that IFC does not provide semantics for these features. In IFC a new type of `IfcSpatialStructureElement` should thus be added. At the same time this would also indicate that `IfcSpace`, which is part of the balcony spatial structure, should be ignored during the extraction of the exterior shell.

Although it is already possible in IFC to model columns and beams as part of a wall, it is not yet common practice. Depending on whether a column is part of a wall or not, it is modelled in CityGML by a `WallSurface` or a `BuildingInstallation` respectively after the conversion. By defining in the IFC standard that columns and beams have to be part of a wall if that is indeed the case, the conversion would no longer require user input during the conversion.

Also, for CityGML, there are some recommendations that would make the conversion more straightforward. Since CityGML is a generic standard (the modelling decisions are left to the implementers), by providing more detailed specifications it would become easier for modellers to know how to model 3D information (to facilitate its use in different applications for instance), and for users to know what to expect. Several efforts, eg the Special Interest Group 3D[12] and Stoter et al. [2013], have already defined their own implementation specifications for CityGML to make the general standard more strict. We argue that the CityGML standard should include more detailed specifications before fragmentation occurs among practitioners. From our research, we recommend the standard to clearly define when and how a building should be subdivided into `BuildingParts`, and clearer rules for `BuildingInstallations`. Additionally, for the boundary surfaces, it should be made clearer when a feature belongs to a specific type. For example, it should be explicitly specified whether only the glass should be modelled as a window, or also its frame.

The LOD concept of CityGML is currently under revision (for version 3). The main proposals, eg Biljecki et al. [2014b] and Benner et al. [2013a], are inline with the algorithm we have presented in this paper since the rules for the modelling of the geometries and the semantics are stricter, and thus clearer. The biggest difference between the current definition of LODs and the proposed ones is how the interior of buildings would be modelled, but this has no influence for our work.

6.2 Future Work

The developed conversion has some limitations and further research is needed to improve the conversion. First, research is required to assure that IFC data meets certain criteria since the conversion works best if certain conditions are met. One of them is that the input geometries are valid and do not intersect; while such cases are rare in practice, we have encountered a few cases. One way to solve this is to pre-process all geometries by intersecting and splitting them to construct a topologically consistent dataset [Zhao et al., 2013]. In addition, our assumptions for handling the semantics of surfaces (buildings contain vertical walls, horizontal ceilings and floors) may work in most of the cases, but not for all of them. Our algorithm can be improved by extending the semantic mapping, which will also result in better geometric conversions. This requires additional semantics in the IFC standard. Our algorithm can be further developed for extracting the terrain intersection curve, which only provides the exact location where the terrain surface touches the building, but also enables the `GroundSurface` to be generated properly.

[12] `http://www.sig3d.org`

Finally, not only better a alignment of the standards is required, but also methods for the conversion to other city objects such as, tunnels and bridges.

Acknowledgements

This research is supported by: (1) 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); (2) the National Natural Science Foundation of China (No. 41201379).

References

Beetz J (2012). Gebruik van 3D gebouw-modellen in 3D GIS: Building information modelling (BIM/IFC) voor beginners (*in Dutch*). Geonovum 3D Pilot Fase II Slotdag.

Benner J, Geiger A, Gröger G, Häfele KH, and Löwner MO (2013a). Enhanced LOD concepts for virtual 3D city models. In Isikdag U, editor, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Proceedings of the ISPRS 8th 3DGeoInfo Conference & WG II/2 Workshop*, pages 51–61. Istanbul, Turkey.

Benner J, Geiger A, Häfele KH, and Knüppel H (2013b). IFCExplorer—Ein Werkzeug für die Integration Unterschiedlicher Raumbezogener Semantischer Daten (*in German*). In *Proceedings Geoinformatik 2013*. Heidelberg, Germany.

Benner J, Geiger A, and Leinemann K (2005). Flexible generation of semantic 3D building models. In Gröger G and Kolbe TH, editors, *Proceedings 1st International Workshop on Next Generation 3D City Models*, pages 17–22. Bonn, Germany.

Biljecki F, Ledoux H, and Stoter J (2014a). Improvement of the LOD concept for 3D city models. GIM International.

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*, 48:1–15.

Boeters R (2013). *Automatic enhancement of CityGML LoD2 models with interiors and its usability for net internal area determination*. Master's thesis, Geomatics, Delft University of Technology.

Chazelle B (1984). Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing*, 13:488–507.

de Laat R and van Berlo L (2011). Integration of BIM and GIS: The development of the CityGML GeoBIM extension. In Kolbe TH, Köning G, and Nagel C, editors, *Advances in 3D Geo-Information Sciences*, Lecture Notes in Geoinformation and Cartography, pages 211–225. Springer-Verlag, Berlin Heidelberg.

Diakité AA, Damiand G, and Van Maercke D (2014). Topological reconstruction of complex 3D buildings and automatic extraction of levels of detail. In *Proceedings 2nd Eurographics Workshop on Urban Data Modelling and Visualisation*, pages 25–30. Strasbourg, France.

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

El-Mekawy M, Östman A, and Hijazi I (2012). An evaluation of IFC-CityGML unidirectional conversion. *International Journal of Advanced Computer Science and Applications*, 3(5):159–171.

El-Mekawy M, Östman A, and Shahzad K (2011). Towards interoperating CityGML and IFC building models: A unified model based approach. *Advances in 3D Geo-Information Sciences*, pages 73–93.

Elberink SO and Vosselman G (2009). Building reconstruction by target based graph matching on incomplete laser data: Analysis and limitations. *Sensors*, 9(8):6101–6118.

Helbich M, Jochem A, Mücke W, and Höfle B (2013). Boosting the predictive accuracy of urban hedonic house price models through airborne laser scanning. *Computers, Environment and Urban Systems*, 39:81–92.

Hijazi I, Ehlers M, and Zlatanova S (2010). BIM for geo-analysis (BIM4GEOA): set up of 3D information system with open source software and open specifications (OS). In *Proceedings 5th International 3D Geoinfo conference*, pages 45–49. Berlin, Germany.

ISO (2013). Iso 16739 industry foundation classes (ifc) for data sharing in the construction and facility management industries.

Kada M and McKinley L (2009). 3D building reconstruction from LiDAR based on a cell decomposition approach. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38:47–52.

Köninger A and Bartel S (1998). 3d-GIS for urban purposes. *Geoinformatica*, 2(1):79–103. ISSN 1384-6175.

Krüger A and Kolbe TH (2012). Building analysis for urban energy planning using key indicators on virtual 3D city models—the energy atlas of Berlin. In *Proceedings XXII ISPRS Congress*, volume XXXIX-B2, pages 145–150. Melbourne, Australia.

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

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.

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

Mäntylä M (1988). *An introduction to solid modeling*. Computer Science Press, New York, USA.

Nagel C (2006). *Ableitung verschiedener Detallierungsstufen von IFC Gebäudemodellen (*in German*)*. Master's thesis, Karlsruhe University of Applied Sciences.

Nagel C and Kolbe TH (2007). Conversion of IFC to CityGML. Presentation OGC 3DIM WG.

OGC (2004). Geography markup language (GML) encoding specification. Open Geospatial Consortium inc. Document 03-105r1, version 3.1.1.

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

Requicha AAG (1982). Representation of rigid solids—theory, methods and systems. *ACM Computing Surveys*, 12(4):437–464.

Schirra S (1997). Precision and robustness in geometric computations. In van Kreveld M, Nievergelt J, Roos T, and Widmayer P, editors, *Algorithmic Foundations of Geographic Information Systems*, volume 1340 of *Lecture Notes in Computer Science*, pages 255–287. Springer-Verlag, Berlin.

Sebastian R and van Berlo L (2010). Tool for benchmarking BIM performance of design, engineering and construction firms in the Netherlands. *Architectural Engineering and Design Management*, 6(4):254–263.

Serra JP (1982). *Image Analysis and Mathematical Morphology*. Academic Press.

Stadler A and Kolbe TH (2007). Spatio-semantic coherence in the integration of 3D city models. In Stein A, editor, *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Proceedings of the WG II/7 5th International Symposium Spatial Data Quality 2007 with the theme: Modelling qualities in space and time*, page 8. Enschede, the Netherlands.

Stoter JE, Ledoux H, Reuvers M, van den Brink L, Klooster R, Janssen P, Beetz J, Penninga F, and Vosselman G (2013). Establishing and implementing a national 3D standard in the Netherlands. *Photogrammetrie, Fernerkundung, Geoinformation*, (4):381–392. doi:http://dx.doi.org/10.1127/1432-8364/2013/0184.

Tawfik MS (1991). An efficient algorithm for CSG to B-rep conversion. In *Proceedings 1st ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications*, SMA '91, pages 99–108. ACM.

Wang W and Wang K (1986). Geometric modeling for swept volume of moving solids. *Computer Graphics and Applications, IEEE*, 6(12):8–17.

Zhao J, Ledoux H, and Stoter J (2013). Automatic repair of CityGML LOD2 buildings using shrink-wrapping. In Isikdag U, editor, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Proceedings of the ISPRS 8th 3DGeoInfo Conference & WG II/2 Workshop*, pages 309–317. Istanbul, Turkey.

Zhao J, Zhu Q, Du Z, Feng T, and Zhang Y (2012). Mathematical morphology-based generalization of complex 3D building models incorporating semantic relationships. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:95–111.