Repairing geometric errors in 3D urban models with kinetic data structures

Mulin Yu, Florent Lafarge, Sven Oesau, Bruno Hilaire

Repairing geometric errors in 3D urban models with kinetic data structures

Mulin Yu[a], Florent Lafarge[a], Sven Oesau[b], Bruno Hilaire[b]

[a]*Inria, Université Côté d'Azur, Sophia Antipolis, France*
[b]*CSTB, Sophia Antipolis, France*

Abstract

3D urban models created either interactively by human operators or automatically with reconstruction algorithms often contain geometric and semantic errors. Correcting them in an automated manner is an important scientific challenge. Prior work, which traditionally relies on local analysis and heuristic-based geometric operations on mesh data structures, is typically tailored-made for specific 3D formats and urban objects. We propose a more general method to process different types of urban models without tedious parameter tuning. The key idea lies on the construction of a kinetic data structure that decomposes the 3D space into polyhedra by extending the facets of the imperfect input model. Such a data structure allows us to re-build all the relations between the facets in an efficient and robust manner. Once built, the cells of the polyhedral partition are regrouped by semantic classes to reconstruct the corrected output model. We demonstrate the robustness and efficiency of our algorithm on a variety of real-world defect-laden models and show its competitiveness with respect to traditional mesh repairing techniques from both Building Information Modeling (BIM) and Geographic Information Systems (GIS) data.

Keywords: Mesh repairing, Semantic reconstruction, Kinetic data structure, BIM, CityGML, IFC

Email addresses: Mulin.Yu@inria.fr (Mulin Yu), florent.lafarge@inria.fr (Florent Lafarge), Sven.Oesau@cstb.fr (Sven Oesau), Bruno.Hilaire@cstb.fr (Bruno Hilaire)

1. Introduction

Computerized 3D models that recreate real urban environments play a more and more fundamental role in our everyday life for assisting us during navigation, imagining our urban projects, entertaining us with video games and movies, optimizing our telecommunication networks or the construction of our houses, reducing our energetic consumption, or protecting us by anticipating disaster scenarios. At the building scale, 3D models are usually conceptualized through the Building Information Modeling (BIM) framework, typically with the Industry Foundation Classes (IFC) format that spatially decomposes a building into volumetric objects. The boundary of these objects can be represented under the form of 2-manifold watertight polygon meshes enriched with semantic properties indicating their nature, e.g. wall, floor, door or empty space. At the city scale, Geographic Information Systems (GIS) practitioners rather rely on the CityGML formalism to represent urban objects with different Levels Of Detail (LOD), as described in (Groger and Plumer, 2012). In particular, the popular LOD2 CityGML models are typically single solids whose surface components are enriched with semantic properties. Both IFC and CityGML 3D models can thus be represented under the form of 2-manifold watertight polygon meshes whose volume or surface components are enriched with semantic properties.

During their creation (either interactively with human operators or automatically with reconstruction algorithms), these 3D models are frequently corrupted by geometric and semantic errors such as degenerated facets, self-intersections, gaps or non-manifold components that require to be corrected before use. Figure 1 shows such errors degrading CityGML and IFC models. Repairing these 3D models in an automated manner is an important scientific challenge. Existing methods typically detect and identify errors before correcting them by local geometric operations such as insertion, removal or snapping of facets. Built on heuristics and a fine parameter tuning, these methods do not generalize well to the variety of buildings.

To address this challenge, we adopt a more global strategy based on the construction of a space partitioning data structure. Built from the facets of the defect-laden input model, the latter decomposes the 3D space into a valid embedding of polyhedra from which the corrected model is extracted. Intuitively, such a data structure can be seen as a scaffold in which all the relations between facets are re-built in a natural manner. This strategy is based on the observation that most of errors contained in 3D models originate from a wrong connectivity between adjacent facets. Such a strategy has been inspired by Diakité et al. (2014)

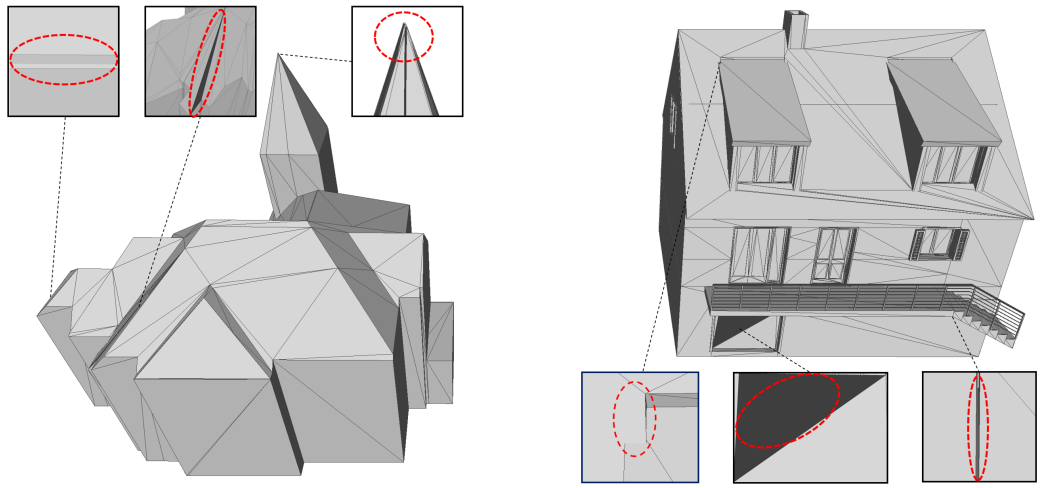


Figure 1: Imperfect CityGML (left) and IFC (right) urban models. The closeups show some typical geometric and semantic errors contained in the models with, from left to right: self-intersection, hole, vertex misalignment, overlapping, wrong facet orientation and gap.

who construct Combinatorial Maps (Damiand and Lienhardt, 2014) from soups of facets. Their construction mechanism, however, suffers from a severe drawback: based on snapping and cutting operations, it cannot guarantee to return valid combinatorial maps. In contrast, our method relies upon a recent kinetic data structure (Bauchet and Lafarge, 2020) whose construction is exact, time-efficient, parameter-free and conceptually natural: input facets extend at constant speed until colliding and forming a polyhedral partition of the 3D space. Once built, the cells of the polyhedral partition are regrouped by semantic classes, the output 3D model being defined as the set of facets at the interface between cells of different semantic labels. This step is formulated as an energy minimization problem.

Contrary to local heuristic-based algorithms, our approach has a low number of parameters and offers a high genericity for repairing geometric and semantic errors contained in different types of urban models. We demonstrate the robustness and efficiency of our algorithm on a variety of real-world defect-laden models and show its competitiveness with respect to traditional mesh repairing techniques on both BIM and GIS data.

Our approach presents several contributions to the field. First, from a conceptual point of view, we propose a repairing system that, in contrast to conventional

detection-then-correction pipelines, undo the connectivity between all facets of the input model before rebuilding them in a natural and efficient manner. Then, we present a general and flexible formulation to the semantic labeling of the polyhedral partition by seeking both consistency with the input semantics and connectivity simplicity within a Markovian energy minimization problem. Finally, we propose two application scenarios for repairing 3D models of buildings, one for surface-based models using the CityGML formalism and one for volume-based models using the IFC standard. Experiments on these two application scenarios were performed on a variety of building types such as residential, industrial, architectural and building blocks, with different input complexity ranging from 5 to $2,146$ facets for CityGML and from 978 to $142,565$ facets for IFC models.

2. Related Work

Our review of previous work covers mesh repairing techniques, algorithms for reconstructing urban scenes in 3D, as well as studies on data validity and methods for constructing GIS and BIM models automatically.

2.1. Mesh repairing

Our problem is a specific instance of the mesh repairing issue whose goal is to correct geometric errors contained in Computer-Aided Design (CAD) models. Deeply studied in Geometry Processing (Campen et al., 2012; Pérez et al., 2016), most of mesh repairing techniques propose geometric operators to correct non-manifold situations and holes in polygon meshes. Such techniques are local and assume that geometric errors are relatively isolated from each other (Guéziec et al., 2001; Barequet and Kumar, 1997). Some methods also propose strategies in which the surface mesh is re-mesh in a global manner. For instance, Marchandise et al. (2012) build a discrete parametrization with an orthogonal gradient method on radial basis functions before remeshing the surface in the parametric space with a computed inverse mapping. Nooruddin and Turk (2003) propose a voxel-based method with morphological operators to handle holes, double walls and intersecting parts. Bischoff and Kobbelt (2005) rely on a snapping process in the vicinity of intersections and cracks to remesh inconsistencies in a manifold way. These methods are relatively efficient in practice, but do not allow semantic errors contained in BIM and GIS data to be corrected.

2.2. Repairing of GIS and BIM data

Some works study the validity of GIS and BIM models and propose specific methods for repairing them. Such tasks are delicate as the errors contained in

CityGML and IFC models are numerous and typically emerge from geometric, topological and semantic approximations (Biljecki et al., 2016; Ohori et al., 2017). Alam et al. (2014) propose a set of geometric rules to test the validity of CityGML models and local operators to repair certain types of errors. Zhao et al. (2014) exploit a similar strategy with an efficient recursive repairing framework. Diakité et al. (2014) and Zhao et al. (2018) are probably the closest works to our method as, similarly to us, they entirely re-build the input models using a space partitioning data structure. The former proposes an automatic method to convert a topology-free model into a cellular decomposition using combinatorial maps whereas the latter uses an heuristic shrink-wrapping algorithm for reconstructing valid solid-based LOD2 buildings through a constrained tetrahedralization. As explained in Alam et al. (2014), these algorithms typically rely on combinations of heuristics that make them efficient for a specific type of input models only. In contrast, we rely upon a recent parameter-free kinetic data structure and an heuristic-free semantic labeling in order to offer a better genericity.

2.3. Reconstruction of urban scenes

Our approach is also closely related to urban reconstruction methods as our output models are entirely re-built in 3D from the input facets. Such methods usually rely upon a preliminary classification of the objects contained in the scenes, even if semantic information can also be extracted after the geometric reconstruction of the objects (Boulch et al., 2013), or even during (Hane et al., 2013). Many urban reconstruction methods operate from airborne data (Musialski et al., 2013; Rottensteiner et al., 2014), the main goal being to capture the outdoor geometry of buildings with mesh-based representations. Some reconstruction methods such as (Verdie et al., 2015; Fang et al., 2018) even produce 3D models with levels of detail that directly conform to the CityGML formalism. The reconstruction of indoor scenes has also been deeply studied in the literature (Khoshelham et al., 2017). Many methods decompose the space into polygonal cells which are then labeled as empty space or solid elements. This strategy allows the reconstruction of the main structural components of indoor scenes such as walls, ceiling and floor (Oesau et al., 2014; Mura et al., 2016). While these methods typically exploit laser scanning or multi-view stereo imagery, some recent approaches tend to reconstruct the indoor scenes directly from floormaps (Chen et al., 2019; Li et al., 2020). In practice, the automatic reconstruction of buildings is however often imperfect and requires interactive corrections to be fully exploitable by practitioners (Arikan et al., 2013).

3. Overview

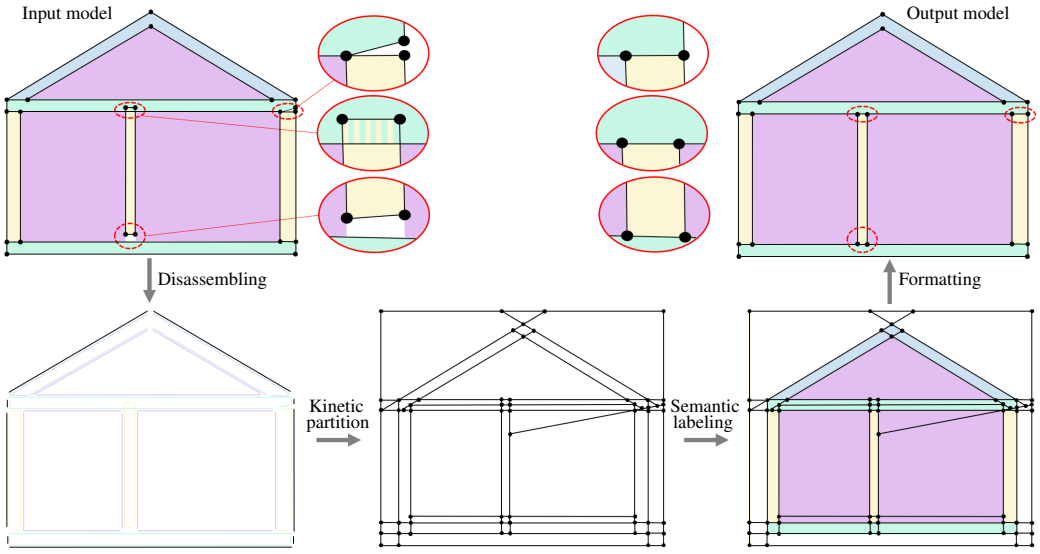


Figure 2: Overview of the proposed approach. Our algorithm departs from an urban 3D model corrupted by geometric and semantic errors, here a BIM model with gap, element overlapping and inaccurate vertex position (top left). We rebuild all the geometric relations between its facets by first collecting and enriching them with semantic information (bottom left) followed by constructing a kinetic data structure that partitions the space into polyhedra (bottom middle). The latter are then regrouped by using a semantic labeling formulation (bottom right) and reformatted into a BIM model in which the initial errors are corrected (top right).

Our algorithm requires as input a polygon mesh describing a building with a CityGML or IFC formalism. While these formalisms are different (the former is based on a surface-based representation whereas the latter relies upon a volume-based representation), one strength of our approach is to first convert these specific input polygon meshes into a general soup of disconnected semantized facets. This step is explained in Section 4.1. We then make the extracted facets grow at constant speed within a kinetic framework to decompose the 3D space into a low number of convex polyhedra (Section 4.2). Finally, these polyhedra are regrouped according to semantic similarities within an energy minimization detailed in Section 4.3. The output 3D model is a polygon mesh whose facets are semantically enriched, similarly to the input model. These three steps of our approach are illustrated in Figure 2.

Our algorithm is designed to correct the most frequent geometric and semantic errors contained in urban 3D models. These errors include vertex misalignment with the duplication of vertices whose coordinates are not exactly identical, inverted facets that have a wrong orientation, semantic defects with volumes associated with the wrong semantic class and general problems of self-intersections, holes or overlapping with typically a wrong connection of adjacent facets of volumes components. Our algorithm does not handle large missing parts that require the use of tailored-made completion algorithms.

4. Our approach

We now describe the different steps of our algorithm and explain how it can be used on both GIS and BIM models.

4.1. Disassembling

The first step consists in extracting a soup of disconnected facets enriched with semantic properties from the input model. Each input facet is oriented: its normal points by construction towards the outside space in case of CityGML models and towards the center of the IfcElement to which the facet belongs in case of IFC models. Our idea is to collect each facet of the input model and to assign it the semantic class pointed by its normal.

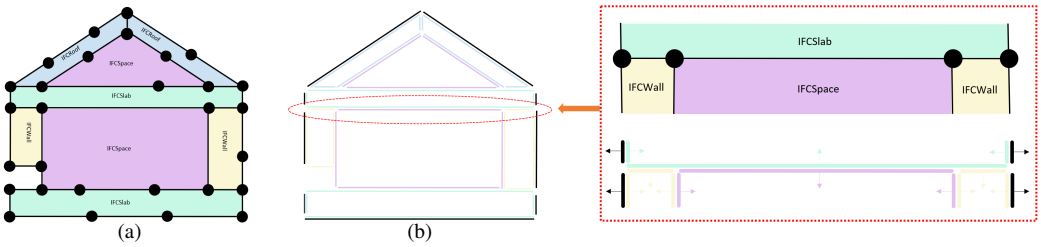


Figure 3: Disassembling of an IFC model. After collecting the boundary facets of each IfcElement (a), we assign them the semantic class of their IfcElement (b). As illustrated in the closeup, each surface component is typically associated with two facets of opposite direction and with different semantic class. Color code: **space**, **outside**, **wall**, **slab**, **roof**).

In case of an IFC model, semantic classes are typically *roof*, *wall*, *slab*, *door*, *window*, *space* or *outside*. These classes correspond to the nature of volume elements that can be found in the IFC formalism where, for instance, the class *wall* refers to an IfcWall element. For each facet of an IfcElement, we thus simply

assign it the corresponding semantic class of this IfcElement. As illustrated in Figure 3, IfcElements usually share common boundaries. The soup of collected facets thus contain coplanar facets with opposite orientations and assigned to different semantic classes.

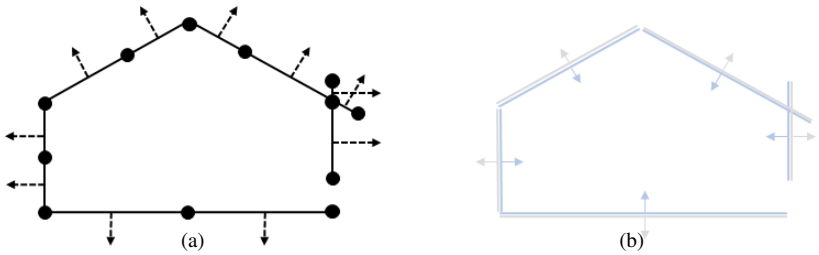


Figure 4: Disassembling of a CityGML model. Each facet of an input CityGML model (a) is duplicated and inverted. The class *outside* is assigned to each input facet while the class *inside* is assigned to the duplicated ones (b). Color code: input facets associated with the class *outside* in grey, duplicated ones associated with the class *inside* in blue.

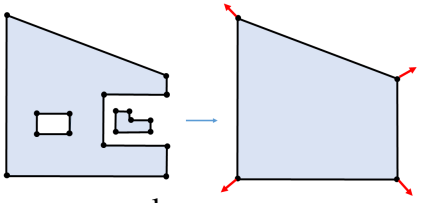
In case of a CityGML model, we consider the input model as a simple boundary-based representation without urban-based semantic classes attached to it, ie without distinction between facade and roof facets for instance. However, by assuming the output model must be watertight, we can distinguish two types of volumes through the inside and the outside of the building. We then consider two semantic classes: *inside* and *outside* the building. As illustrated in Figure 4, we assign the class *outside* to each oriented facet of the input model. We also duplicate each facet, invert its normal, and assign it the class *inside*. This duplication operation is required because, contrary to IFC models that describe volume objects, CityGML models only capture the boundary surface of a building. This assignment is natural and simply relies on the assumption that the normals of CityGML input facets point toward the outside of buildings.

4.2. Kinetic partitioning

Our goal is now to build a kinetic data structure that partitions the space into polyhedra aligned with the collected facets.

We first regroup coplanar facets, *i.e.* facets that share the same supporting plane, and convert them into a single convex polygon. This polygon is defined as the convex hull of the coplanar facets, *i.e.* the smallest convex polygon that contains all the coplanar facets. As illustrated in the inset, input coplanar facets (left) are converted into a simpler convex polygon (right) that will be

propagated in the 3D space more easily during the kinetic simulation (see red arrows). Note that converting input facets into convex polygons does not lead to a loss of information as, for each edge of an original facet, it also exists an adjacent input facet that will be converted to a convex polygon.



We then exploit the kinetic framework proposed by Bauchet and Lafarge (2020) to make the convex polygons grow at constant speed until colliding with each other and decomposing the 3D space into polyhedra. The principle of such a kinetic framework has been first formalized in Computational Geometry by Guibas (2004). A kinetic data structure corresponds to a set of geometric primitives, here convex polygons, whose coordinates are continuous functions of time. A kinetic framework consists in maintaining the validity of a set of geometric properties while the primitives evolve over time. When a property is no longer valid (also called an *event*), some geometric modifications are operated on the set of primitives to make the property valid again. This situation typically appears when two primitives collide. The algorithmic purpose of kinetic frameworks is to dynamically order the times of occurrence of the events within a priority queue in an efficient manner. A kinetic simulation then consists in un-stacking this queue until no primitive can move anymore. More information about this process can be found in Section 4 of Bauchet and Lafarge (2020).

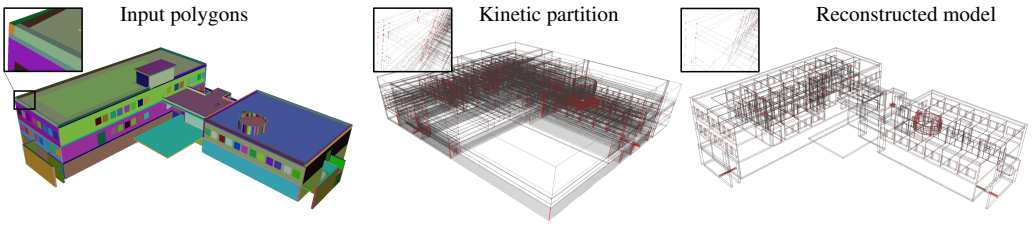


Figure 5: Kinetic partitioning of an IFC model. The $142,565$ facets of the input defect-laden model (left) are regrouped into 668 convex polygons. The kinetic partition built from the convex polygons contains $16,306$ polyhedra (middle) from which the reconstructed model is extracted (right). The IFC model contains 495 IFC elements including 209 IfcWall, 4 IfcRoof, 19 IfcSlab, 120 IfcWindow, 65 IfcDoor, 77 IfcSpace and 1 outside space.

Such a kinetic framework allows us to produce compact partitions with a low number of polyhedra. The output partition comes with strong guarantees including an exact construction with rationals, the convexity of the polyhedra and the validity of the polyhedral embedding.As shown in Figure 5 from an IFC model,

the construction process is also highly scalable and can handle a large number of convex polygons (themselves made from potentially many input coplanar facets) in reasonable time. Figure 6 shows an example of such a construction process from a CityGML model. In particular, we can see the kinetic strategy allows us to properly re-build the erroneous facet connections.

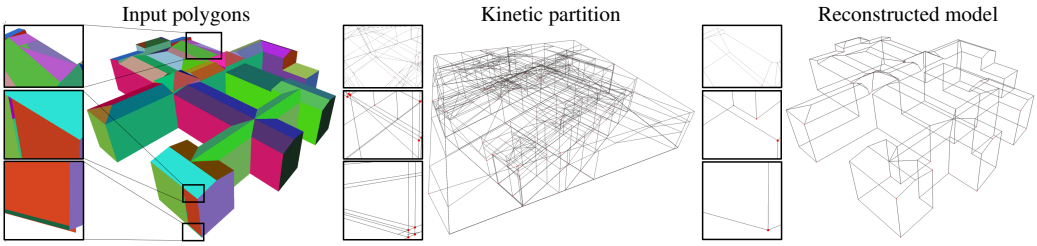


Figure 6: Kinetic partitioning of a CityGML model. The 378 facets (left) of the input model are grouped into 52 convex polygons, leading to the construction of a kinetic partition of 356 polyhedra (middle). The corrected model is extracted by selecting polyhedra located inside the building (right). Closeups show situations where facets are initially not well connected, the kinetic partition allowing us to properly re-build these connections.

4.3. Semantic labeling

We now assign one of the possible semantic classes to each polyhedral cell of the kinetic partition. We formulate this problem as a multi-label energy minimization where the label set is $\mathcal{L} = \{inside, outside\}$ in case of CityGML models and $\mathcal{L} = \{slab, space, wall, roof, door, window, outside\}$ in case of IFC models. These label sets were used for our experiments, but can freely be extended to further labels. In particular, the label set for IFC models can be reduced or augmented depending on the semantic complexity of models. Such a semantic labeling is not an obvious operation as i) some facets of the kinetic partition are not present in the input model and ii) some facets of the kinetic partition can inherit from conflicting semantic information when errors are present in the input model. To solve this problem, we proposed an energy minimization framework with a multi-objective energy function.

We denote by $\mathbf{C}$ the set of polyhedral cells of the kinetic partition, and by $x_i \in \mathcal{L}$, the label that specifies the semantic class associated with polyhedral cell $i \in \mathbf{C}$. We measure the quality of a possible configuration $\mathbf{x} = (x_i)_{i \in \mathbf{C}}$ with a two-term energy of the form

$$U(\mathbf{x}) = \sum_{i \in \mathbf{C}} D_i(x_i) + \lambda \sum_{i \sim j} V_{ij}(x_i, x_j) \tag{1}$$

where D_i is an unary data term, and V_{ij} a potential over pairs of adjacent polyhedral cells. λ is a parameter balancing these two terms. $i \sim j$ denotes the set of pairs of adjacent cells i and j.

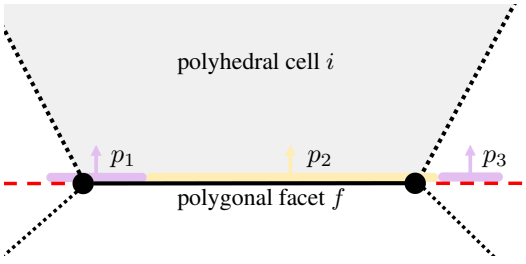
The unary data term $D_i(x_i)$ measures the consistency between the semantic class x_i assigned to the polyhedral cell i and the semantic properties of enriched facets collected during the disassembling step. More specifically, this term is formulated as a sum of local consistency measures over all the polygonal facets forming the polyhedral cell i so that

$$D_i(x_i) = \sum_{f \in i} d_f(x_i) \tag{2}$$

where $d_f(x_i)$ is a function measuring the local consistency on polygonal facet f. This function checks whether the enriched facets collected during the disassembling step which have the same supporting plane than polygonal facet f and oriented toward the center of mass of cell i have the same semantic class than label x_i. The function is defined by

$$d_f(x_i) = \begin{cases} 0 & \text{if } \Omega_f = \varnothing \\ \sum\limits_{p \in \Omega_f} a_f(p) \times 1_{\{x_i \neq l_p\}} & \text{otherwise} \end{cases} \tag{3}$$

where Ω_f is the set of enriched facets collected during the disassembling step which **i)** have the same supporting plane than the polygonal facet f, and **ii)** are oriented toward the center of mass of cell i. $a_f(p)$ is a function that measures the overlapping area between the enriched facet p and the polygonal facet f, $1_{\{.\}}$ is the characteristic function, and l_p is the semantic class associated to the enriched facet p. In the inset for instance, Ω_f is composed of three enriched facets p_1, p_2 and p_3, but only p_1 and p_2 are involved in the computation of d_f as $a_f(p_3) = 0$. The dash red line represents the supporting plane common to the three enriched facets and to the polygonal facet f (black line-segment).



The potential V_{ij} penalizes configurations that are either geometrically overly complex or semantically improbable. This term is formulated as a generalized Potts model so that:

$$V_{ij}(x_i, x_j) = a_{ij} \cdot W(x_i, x_j) \tag{4}$$

where a_{ij} is area of the common facet between adjacent polyhedral cells i and j and W is a matrix of size $|\mathcal{L}| \times |\mathcal{L}|$ where coefficient $W(x_i, x_j)$ indicates whether label x_i and label x_j are likely to be assigned to two adjacent cells. Three cases are distinguished to set the value of its coefficients:

- $W(x_i, x_j) = 0$ if $x_i = x_j$. This condition favors label homogeneity between adjacent cells. It allows us to encourage solutions where reconstructed volume elements are connected in a simple way.
- $W(x_i, x_j) = 1$ else if label x_i and label x_j are unlikely to be spatially next to each other, *e.g.* class *door* and class *roof* as doors are traditionally positioned vertically along walls.
- $W(x_i, x_j) = 0.1$ otherwise, *i.e.* when label x_i and label x_j are likely to be spatially next to each other while being different.

To decide whether two semantic classes are likely to be next to each other, we conducted a statistical analysis over a large range of real-world BIM models in which occurrences below 1% were considered as unlikely[1]. Figure 7 shows the impact of this potential term on an IFC model.

4.4. Formatting

The last step consists in converting the polyhedral partition in which each cell is assigned to a semantic class into a IFC or CityGML model. This operation is realized by simply grouping the adjacent cells with an identical semantic class: each resulting polyhedron gives, in case of a BIM model, an IFC element whose surface boundary is the polyhedron itself and whose type is given by the semantic class of the polyhedron. In case of a CityGML model, we simply regroup the adjacent cells labeled as *inside*. Because the kinetic partition is a valid embedding with an exact construction, the formatting step is simple and guarantees output volume elements do not overlap and share common polygonal facets. It also allows us to easily retrieve and save the adjacency relationship between the volume elements which was not originally available in the input file.

[1]For CityGML models, the notion of spatial proximity between classes is not used as only two classes are considered: we simply have $W(x_i, x_j) = 0$ if $x_i = x_j$, and $W(x_i, x_j) = 1$ otherwise.

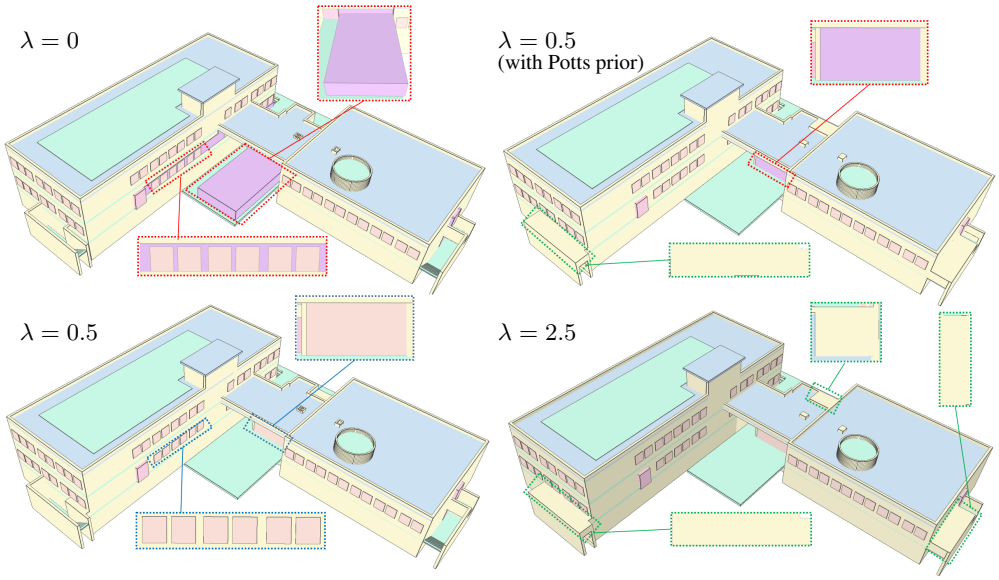


Figure 7: Impact of parameter λ. When only the data term is taken into account ($\lambda = 0$, top left), polyhedral cells are mislabeled at several locations, as shown on the closeups where, for instance, an outside large cell is wrongly assigned to *space*. Our potential typically allows the correction of such mislabeling ($\lambda = 0.5$, bottom left). Giving too much importance to the potential can however oversimplify the labeling and makes small elements such as windows disappear ($\lambda = 2.5$, bottom right). When considering simply a standard Potts model with a binary matrix W (top right), some unrealistic configurations can appear such as horizontal facade components.

5. Implementation details

The proposed approach has been implemented in C++. The $CGAL$ library has been used for the manipulation of the geometric data structures and the geometric operations and the $OpenSceneGraph$ library for the I/O operations on the IFC models. The efficiency of our approach relies upon several technical details that we explain below.

Coplanar facet grouping with tolerance. After the disassembling step, coplanar facets are grouped and converted into a single convex polygon that will be propagated during the kinetic simulation. However, due to floating point imprecision in the coordinates of the facets, an exact coplanarity between facets is often improbable. In practice, we rather rely upon the notion of *near-coplanarity* by introducing two tolerance parameters. The first one defines the maximal angle between the normal vectors of two near-coplanar facets whereas the second specifies the

maximal orthogonal distance between the centroids of two near-coplanar facets. Reasoning with near-coplanar facets allows us to avoid rounding issues which constitute a common source of errors in urban models (Biljecki et al., 2016). In our experiments, we typically set the maximal angle to 1 degree and the maximal distance to 0.001 meter.

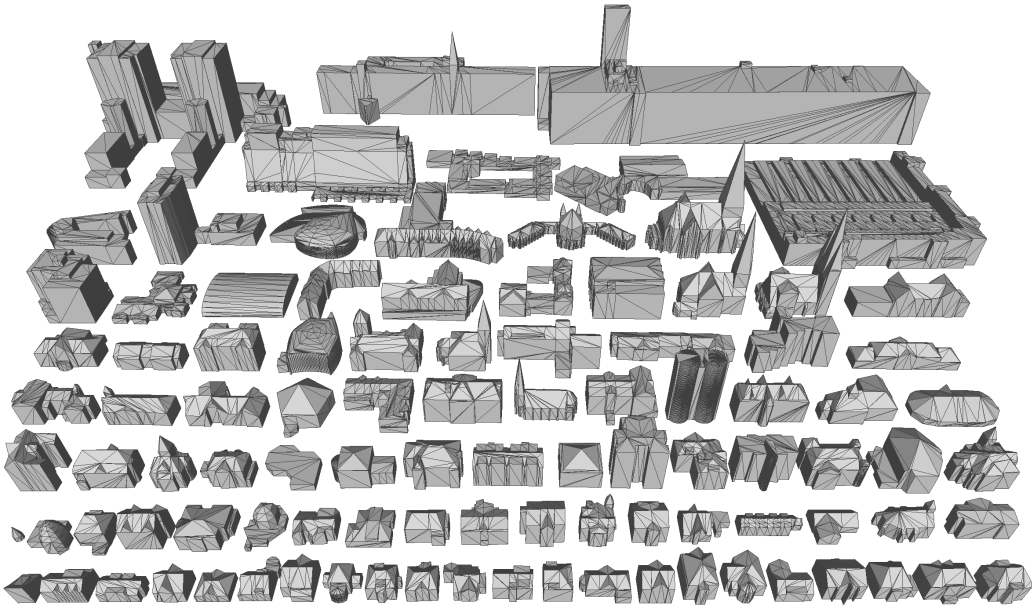


Figure 8: 100 LOD2 models of varying complexity collected from the Hanover 3D digital city model database.

Simplification of some fine geometric details. Input models can contain some fine geometric details, as for instance, handles on doors or windows. Such details are not important in the semantic understanding of the building and have a high geometric complexity that reduces performances of the kinetic partitioning. We thus replace these geometric details by 3D bounding boxes in our pipeline. The user then has the possibility to reintegrate these geometric details in the output models, but we do not offer specific operations to repair their potential errors.

Creation of an outside class for IFC models. We introduce the semantic class *outside* to label the cells that are not part of the building. This class does not exist as an IfcElement in IFC models but is necessary in our labeling formulation in order to assign a semantic label at any location in the 3D space. In practice, we

duplicate all the non-overlapping facets, invert their normals and assign them the class *outside* and *space* during the disasembling step. During the labeling step, we also label the polyhedral cells located on the borders of the bounding-box with the class *outside*.

6. Experiments on CityGML models

We evaluated our algorithm for repairing urban models represented with the CityGML formalism, in particular with a LOD2 (Levels Of Detail) description. We used the Val3Dity tool proposed by (Ledoux, 2018) to test whether a model is error-free or corrupted. This tool allows us to check the validity of models against a large range of geometric and topological errors, from self-intersecting facets to vertex misalignment through non-planarity and holes.

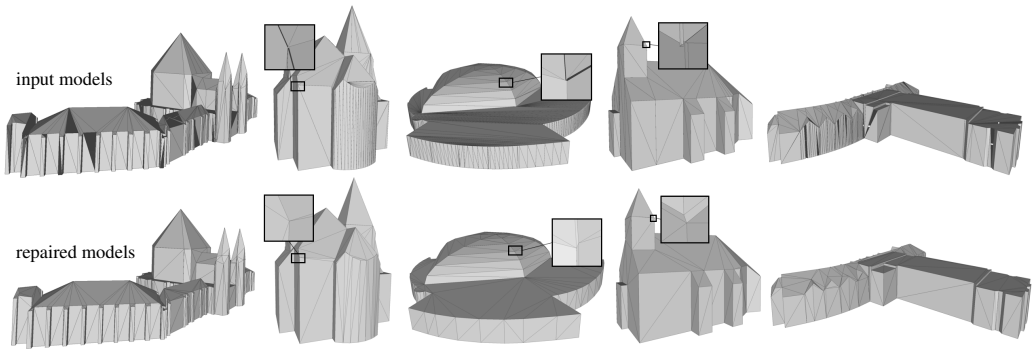


Figure 9: Visual results on five buildings from the Hanover 3D digital city model database. The simulated errors introduced in the input models (top) are correctly fixed by our algorithm (bottom).

Tests with simulated errors. We first tested our algorithm from 100 buildings of varying complexity in the Hanover 3D digital City model dataset (Hanover City and Urban Development, 2021), in which we artificially introduced random errors. We selected a collection of 100 LOD2 buildings of different types, from residential houses to architectural structures through industrial estates, as illustrated in Figure 8. The average number of facets per building is 317. As illustrated in Figure 8, structures such as churches, industrial estates and exhibition centres are more complex and can contain up to $2,146$ facets. We then generated 300 defect-laden models from this collection, all invalid with respect to the Val3Dity tool, by randomly perturbing vertices and removing facets, leading to the generation

of errors such as self-intersections, holes, or vertex misalignment. Our algorithm successfully repaired 292 of the 300 invalid models. The 8 still invalid models returned by our algorithm had typically too many missing facets to properly reconstruct the buildings, as shown in Figure 12. Figure 9 shows some buildings before and after being repaired by our algorithm. The defects such as holes, self-intersections and vertex misalignment are properly fixed, even for the buildings with a complex geometry.

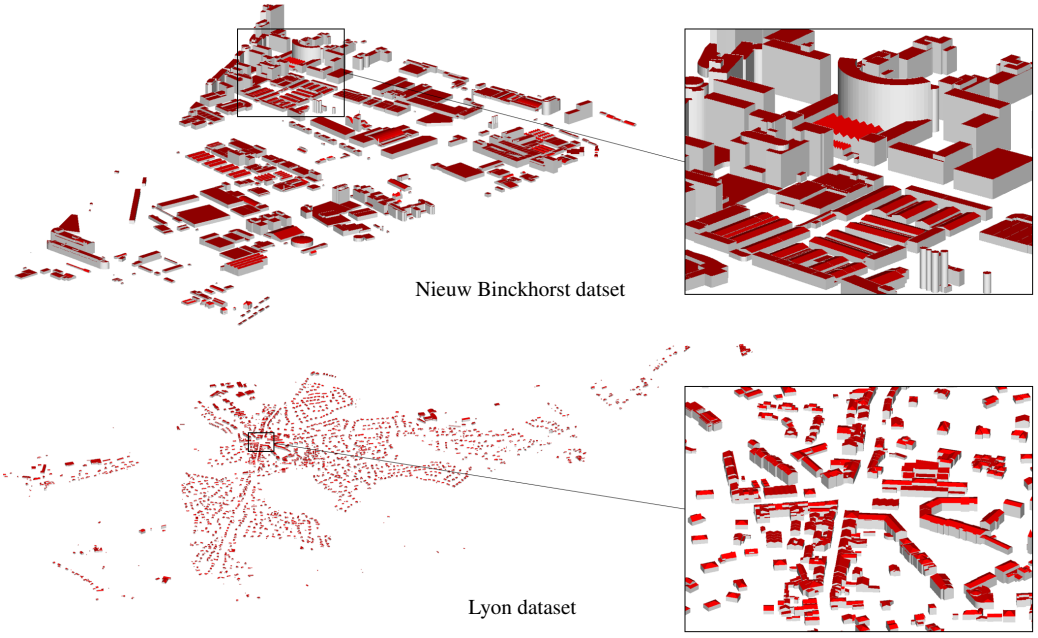


Figure 10: Lyon and Nieuw Binckhorst CityGML datasets. Produced by national organisms in real-world conditions, most of buildings of these two datasets are corrupted by geometric errors. 1158 out of 1421 buildings are invalid according to the Val3Dity tool for Lyon as well as all the 414 buildings for Nieuw Binckhorst. Each building in Lyon CityGML dataset is represented as *Solid* while as *MultiSurface* in Nieuw Binckhorst CityGML dataset.

Tests with real-world errors. We also conducted experiments on two defect-laden CityGML datasets on the cities of Lyon, France (Metropolis, 2015) and Nieuw Binckhorst, The Hague, The Netherlands (Municipality, 2018). They represent a dense city downtown and a complex industrial area respectively. These two datasets, illustrated in Figure 10, have been produced by national organisms in real-world conditions. Residential houses and warehouses are typically described

by a few dozen facets while skyscrapers, churches and industrial estates can reach up to 684 facets. Most of buildings contain errors and do not pass the validity test of Val3Dity, *i.e.* 1158 out of 1421 buildings are invalid for the Lyon dataset as well as all the 414 buildings of the Nieuw Binckhorst dataset. Our algorithm achieves repairing 99.6% (respectively 92.5%) of the invalid buildings for the Lyon (respectively Nieuw Binckhorst) dataset.

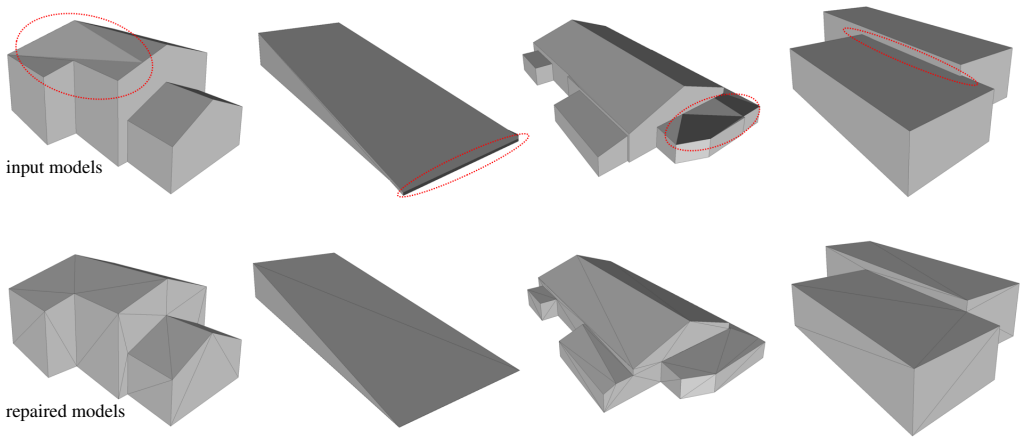


Figure 11: Visual results on four buildings of the Lyon and Nieuw Binckhorst CityGML dataset. Our algorithm achieves repairing geometric errors such as (from left to right) non-planar facets, vertex misalignment, holes and self-intersecting facets.

Figure 11 shows some examples of invalid buildings with non-planar facets, vertex misalignment, holes and self-intersecting facets that have been correctly repaired by our algorithm. Besides correcting these geometric errors, our algorithm produces meshes which are guaranteed to be watertight. As illustrated in Figure 12, our algorithm fails to repair the input mesh when too many facets are missing or when the mesh is highly non-manifold.

Comparisons. We compared our algorithm with the specialized HSW algorithm (Zhao et al., 2018) designed for repairing errors in CityGML LOD2 buildings. Three evaluation criteria were considered: (i) applicability that measures the ratio of buildings processed by the algorithm, (ii) accuracy that measures the ratio of buildings successfully corrected by the algorithm to the number of processed buildings, and (iii) the processing time. As illustrated in Table 1, our algorithm outclasses the HSW algorithm on each of the three criteria. Thanks to our global strategy that disassembles then reassembles facets, our algorithm is very general

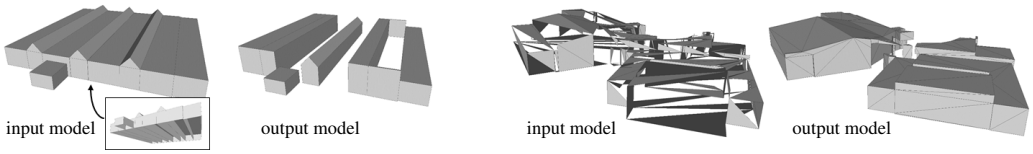


Figure 12: Failures cases. When the input mesh is highly non-manifold (left), some parts of the building are typically omitted. Our algorithm also fails to repair the input mesh when too many facets are missing (right).

		Lyon	Nieuw Binckhorst
#input invalid buildings		1158	414
HSW	Applicability	50.3%	91.1%
	Accuracy	29.9%	35.8%
	Processing time (sec. per building)	2.89	10.51
ours	Applicability	100%	100%
	Accuracy	99.6%	92.5%
	Processing time (sec. per building)	0.62	0.3

Table 1: Quantitative comparison against the HSW algorithm (Zhao et al., 2018) on the Lyon and Nieuw Binckhorst datasets. Applicability refers to the ratio of buildings processed by the algorithm to the number of invalid input buildings. Accuracy refers to the ratio of buildings successfully corrected (i.e. valid according to the Val3Dity tool) to the number of buildings processed by the algorithm.

and can be applied on all the input invalid models of the Lyon and Nieuw Binckhorst datasets. This is not the case for the HSW algorithm that cannot be run on the complex and highly corrupted buildings. Our accuracy is higher by a large margin and our processing times are faster by approximately one order magnitude.

We also compared our algorithm with three generic mesh repairing techniques, *i.e.* PMP (Sieger and Botsch, 2020), MeshFix (Attene, 2010) and PolyMender(Ju, 2004). Figure 13 shows a visual comparison on four different CityGML LOD2 buildings. By exploiting a local repairing strategy, the surface-based methods MeshFix and PMP often generate geometric inconsistencies when input errors cannot be fixed, which often also leads to a loss of geometric details. Similarly to our approach, PolyMender is a volume-based method that offers more robustness to this problem and guarantees a watertight output mesh. However, PolyMender relies upon a volumetric grid as space-partitioning data structure that over-simplifies the output mesh and does not preserve sharp features. In contrast, our kinetic data structure is resolution-independent and not axis-aligned which

allows us to repair meshes without any geometric accuracy loss compared to the input mesh. In Figure 13, only our method can correctly repair the four input models.

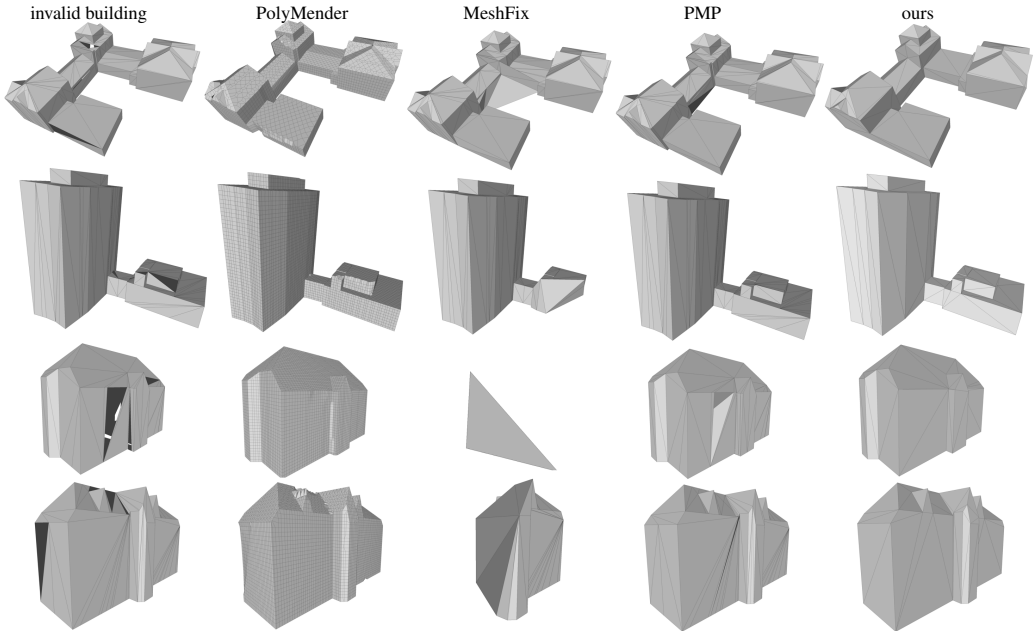


Figure 13: Visual comparisons with the generic mesh repairing methods PolyMender(Ju, 2004), MeshFix (Attene, 2010) and PMP (Sieger and Botsch, 2020). Only our method can correctly repair the four input models without remaining errors or loss of geometric accuracy.

7. Experiments on IFC models

We tested our algorithm on IFC models. Contrary to experiments realized in Section 6 on CityGML models, databases of IFC models are more difficult to find. For our experiments, we rely on a set of nine IFC models of real-world buildings designed by BIM experts. This set includes five houses (three poorly detailed and two highly detailed), two tall residential structures, a single-floor office building and a highly detailed school. They have different complexity, with the number of IfcElements ranging from 10 to 509 and the number of input facets ranging from 978 to $142,565$. Because there is no tool available in the literature to test the validity of IFC models, we propose several criteria to check whether an IFC model contains some obvious geometric or semantic errors:

- Manifoldness test: We check that the boundary of each IFC element is a watertight, 2-manifold and consistently oriented polyhedron.
- Overlapping test: We check for each pair of adjacent IfcElements that their corresponding volumes do not overlap through a non-zero volume.
- Gap test: We check for each pair of adjacent IfcElements that there is no non-empty space between their corresponding volumes.
- Semantic test: We check that (i) the semantic class of each IfcElement is correct and (ii) there is no large volume inside the building not associated with an IfcElement.

		#IfcElement	#ME	#OE	#GE	#SE
case-1	Input	509	236	12	31	0
	Output	498	0	0	0	0
case-2	Input	125	90	8	2	5
	Output	144	0	0	0	4
case-3	Input	577	182	17	1	2
	Output	554	0	0	0	0
case-4	Input	48	25	9	3	2
	Output	46	0	0	0	2
case-5	Input	51	20	3	3	1
	Output	49	0	0	0	0
case-6	Input	10	6	2	0	0
	Output	10	0	0	0	0

Table 2: Quantitative results on the IFC models shown in Figure 14. #ME (respectively #OE, #GE and #SE) refers to the number of times where the manifoldness test (respectively overlapping test, gap test and semantic test) is not valid.

We tested our algorithm on IFC models that differ in terms of complexity and levels of detail. Figure 14 presents visual results while Table 2 provides a quantitative evaluation of these results. Figure15 illustrates the robustness of our algorithm on challenging errors. One can see in Table 2 that the number of IfcElements slightly decreases after the repairing process: some input adjacent IfcElements with the same semantic class are typically merged during the semantic labeling. A strength of our algorithm is to guarantee that the boundary of each output IfcElement is, by construction, a watertight, 2-manifold and consistently oriented polyhedral surface mesh. As a result, the manifoldness test is always

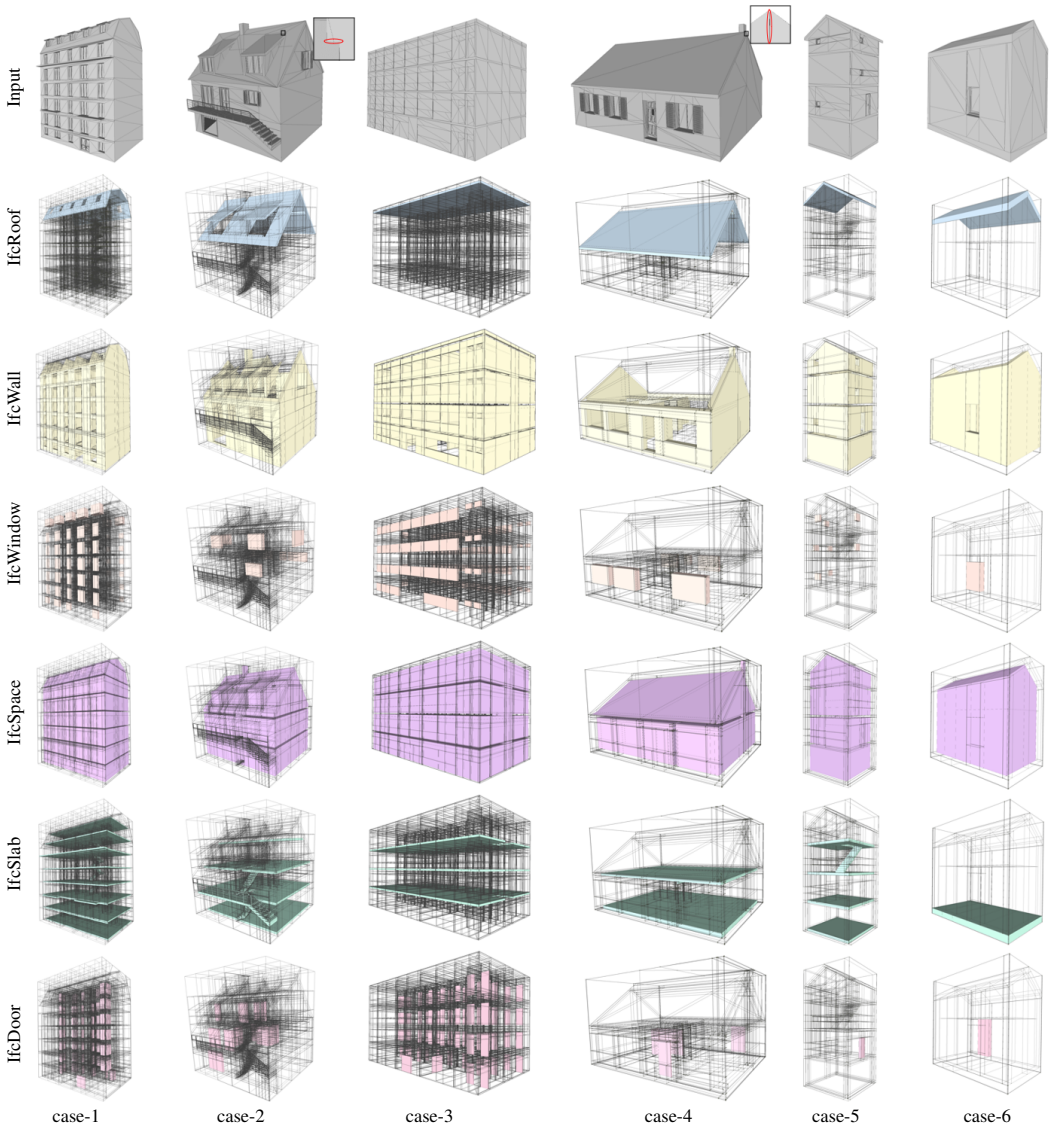


Figure 14: Visual results on different IFC models. Our algorithm rebuilds the various IfcElements of an input IFC model (top) while guaranteeing a valid embedding of the volumes. Input defects such as overlapping IfcElements or gaps between IfcElements (see closups) are fixed even for complex buildings. The reconstructed IfcElements of each semantic class are shown from the second to last rows inside the polyhedral partition (black wireframe). Color code: roof, wall, window, space, slab and door.

valid on the corrected output model. This is not the case in the input models where around half of the IfcElements typically suffers from manifoldness issues (see the #ME column in Table 2). Our algorithm can also fill the potential gap in between adjacent IfcElements or separate them when they overlap, as we can see in the #OE and #GE columns in Table 2. This is made possible thanks to the grouping of near-coplanar input facets that allows us to realign geometric approximations in a global manner. Our algorithm finally reduces the number of semantic errors, but is not able to entirely correct them. Situations where semantic errors cannot be corrected occur when the human-operator wrongly annotated a large IfcElement in the input model.

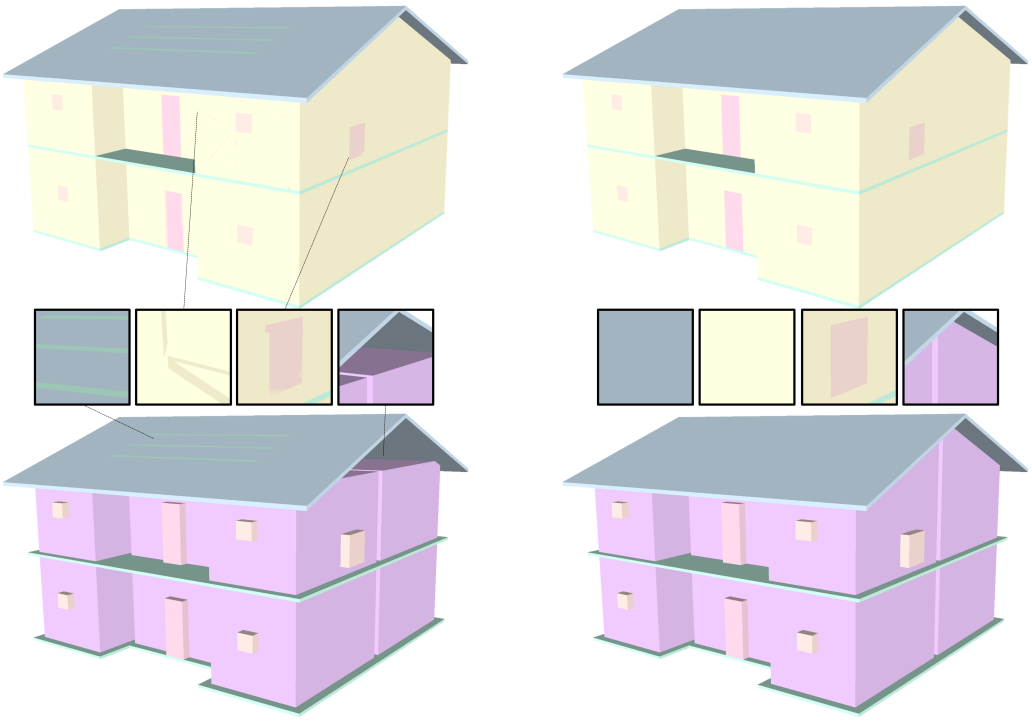


Figure 15: Correction of an IFC model. Our algorithm automatically corrects a defect-laden IFC model (left) containing multiple types of errors, as shown in the closeups with (from left to right) confusions between semantic classes, non-manifold situations, IfcElement overlaps and gaps. Note in particular that our corrected IFC model (right) properly fills in the large gap as IfcSpace under the roof.

Table 3 gives some statistics on the complexity of the IFC models presented in Figure 14 and the performance of our algorithm. Kinetic partitioning is typically

	case-1	case-2	case-3	case-4	case-5	case-6
#Input facets	101967	27532	13508	12394	5012	978
#Primitives	958	598	176	120	109	28
#Polyhedra	19636	9629	13999	1188	1041	124
Kinetic partitioning (sec.)	24557.4	4507.82	108.775	24.1	18.17	0.618
Semantic labeling (sec.)	54.508	14.285	24.0995	1.437	1.3	0.232

Table 3: Performance of our algorithm on the IFC models presented in Figure 14.

the most time-consuming step, especially when the number of input facets is high. In order to reduce the processing time of this step, one possibility is to exploit a subdivision scheme within the kinetic framework as detailed in (Bauchet and Lafarge, 2020). Processing time for the semantic labeling step is shorter, typically a few seconds on a medium-complexity IFC model.

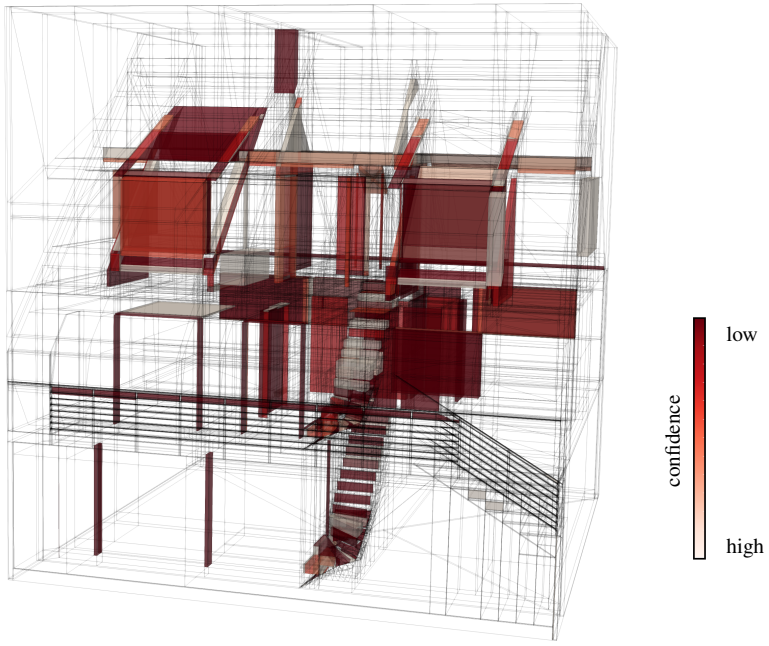


Figure 16: Confidence score. Our algorithm provides to the user a confidence score that highlights the locations where the semantic labeling might be erroneous. Transparent cells mean high confidence on the result whereas dark red means uncertainty.

Because errors in IFC models are sometimes subject to interpretation, it can be interesting to provide to the user a confidence map on the semantic class associ-

ated with each polyhedral cell. This can be done by simply choosing this score as the unary data term of Equation 2, where lower means higher confidence. Figure 16 illustrates the use of such a confidence score as a way to check spatial locations where our algorithm might have taken the wrong choice. A human operator can then easily select a polyhedral cell and modify its semantic label. Such a case of user assistance is shown in Figure 17 with examples of semantic errors that were not corrected by our algorithm but can be easily fixed by a human operator guided by the confidence map. Note that only the results from Figure 17 have not been obtained automatically.

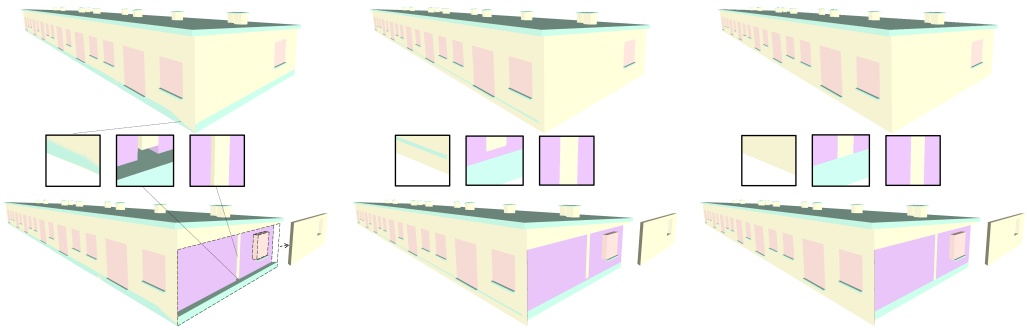


Figure 17: Failure cases corrected by human interactions. The input IFC model (left) contains errors (see closeups) that cannot be repaired by our algorithm (middle). These errors can be located by an human operator checking the low confidence scores and modifying the semantic classes of the concerned cells (right).

8. Conclusion

We proposed an automatic approach for repairing geometric errors contained in urban models. The originality of our approach lies on the construction of a kinetic data structure that decomposes the 3D space into polyhedral cells by extending the original input facets. It allows us to re-build all the relations between the input facets in an efficient and robust manner, and thus to repair a variety of geometric errors including self-intersections, gaps and vertex misalignment. One strength of our approach is its capacity to adapt to different types of urban models, both CityGML and IFC models. In particular, we demonstrated its efficacy on a variety of real-world defect-laden models and showed its competitiveness with respect to specialized CityGML model repairing methods and general mesh repairing techniques.

As future work, we will investigate the design of a new labeling framework able to learn and predict the semantic and geometric errors from large databases of urban models, leading to a correction of a wider range of errors. We also plan to extend our method to include the repair of CAD models that are not represented by a piece-wise planar geometry.

Acknowledgments

This work was partially supported by the French Scientific and Technical Center for Building (CSTB). We thank Hugo Ledoux, Junqiao Zhao and Mingchen Li for providing material for comparisons and validity tests.

References

Alam, N., Wagner, D., Wewetzer, M., von Falkenhausen, J., Coors, V., Pries, M., 2014. Towards automatic validation and healing of citygml models for geometric and semantic consistency, in: Innovations in 3D Geo-Information Sciences.

Arikan, M., Schwarzler, M., Flory, S., Wimmer, M., Maierhofer, S., 2013. O-snap: Optimization-based snapping for modeling architecture. Trans. on Graphics 32.

Attene, M., 2010. A lightweight approach to repairing digitized polygon meshes. The visual computer 26.

Barequet, G., Kumar, S., 1997. Repairing cad models, in: Proceedings. Visualization'97.

Bauchet, J.P., Lafarge, F., 2020. Kinetic Shape Reconstruction. ACM Trans. on Graphics 39.

Biljecki, F., Ledoux, H., Du, X., Stoter, J., Soon, K.H., Khoo, V., 2016. The most common geometric and semantic errors in citygml datasets. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences 4.

Bischoff, S., Kobbelt, L., 2005. Structure preserving cad model repair, in: Computer Graphics Forum.

Boulch, A., Houllier, S., Marlet, R., Tournaire, O., 2013. Semantizing Complex 3D Scenes using Constrained Attribute Grammars. Computer Graphics Forum 32.

Campen, M., Attene, M., Kobbelt, L., 2012. A practical guide to polygon mesh repairing., in: Eurographics (Tutorials).

Chen, J., Liu, C., Wu, J., Furukawa, Y., 2019. Floor-sp: Inverse cad for floorplans by sequential room-wise shortest path, in: Proc. of International Conference on Computer Vision (ICCV).

Damiand, G., Lienhardt, P., 2014. Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing. A K Peters/CRC Press.

Diakité, A.A., Damiand, G., Van Maercke, D., 2014. Topological reconstruction of complex 3d buildings and automatic extraction of levels of detail, in: Eurographics Workshop on Urban Data Modelling and Visualisation.

Fang, H., Lafarge, F., Desbrun, M., 2018. Planar Shape Detection at Structural Scales, in: Proc. of Computer Vision and Pattern Recognition (CVPR).

Groger, G., Plumer, L., 2012. Citygml - interoperable semantic 3d city models. Journal of Photogrammetry and Remote Sensing 71.

Guéziec, A., Taubin, G., Lazarus, F., Hom, B., 2001. Cutting and stitching: Converting sets of polygons to manifold surfaces. IEEE Transactions on Visualization and Computer Graphics 7.

Guibas, L.J., 2004. Kinetic data structures, in: Handbook of Data Structures and Applications.

Hane, C., Zach, C., Cohen, A., Angst, R., Pollefeys, M., 2013. Joint 3d scene reconstruction and class segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Hanover City, D.o.P., Urban Development, G., 2021. City model hannover citygml lod2. `https://opengeodata.hannover-stadt.de/Stadtmodell_Hannover_CityGML_LoD2.zip`.

Ju, T., 2004. Robust repair of polygonal models. ACM Transactions on Graphics (TOG) 23.

Khoshelham, K., Díaz-Vilariño, L., Peter, M., Kang, Z., Acharya, D., 2017. The isprs benchmark on indoor modelling, in: Proceedings ISPRS geospatial week.

Ledoux, H., 2018. val3dity: validation of 3d gis primitives according to the international standards. Open Geospatial Data, Software and Standards 3.

Li, M., Lafarge, F., Marlet, R., 2020. Approximating shapes in images with low-complexity polygons, in: Proc. of the IEEE conference on Computer Vision and Pattern Recognition (CVPR).

Marchandise, E., Piret, C., Remacle, J.F., 2012. Cad and mesh repair with radial basis functions. Journal of Computational Physics 231.

Metropolis, L., 2015. 3d textured model of the town of la tour-de-salvagny. https://download.data.grandlyon.com/files/grandlyon/localisation/bati3d/LA_TOUR_DE_SALVAGNY_2015.zip.

Municipality, T.H., 2018. 3d stadsmodel den haag 2018 citygml. https://ckan.dataplatform.nl/dataset/36049d1a-4a0f-4c5d-8adb-21dbfb7252f9/resource/8fb03a5a-5872-4bd6-afd6-9093c1e2e87a/download/39_binckhorst.zip.

Mura, C., Mattausch, O., Pajarola, R., 2016. Piecewise-planar reconstruction of multiroom interiors with arbitrary wall arrangements. Computer Graphics Forum 35.

Musialski, P., Wonka, P., Aliaga, D.G., Wimmer, M., Van Gool, L., Purgathofer, W., 2013. A survey of urban reconstruction. Computer Graphics Forum 32.

Nooruddin, F.S., Turk, G., 2003. Simplification and repair of polygonal models using volumetric techniques. IEEE Transactions on Visualization and Computer Graphics 9.

Oesau, S., Lafarge, F., Alliez, P., 2014. Indoor Scene Reconstruction using Feature Sensitive Primitive Extraction and Graph-cut. ISPRS Journal of Photogrammetry and Remote Sensing 90.

Ohori, K.A., Biljecki, F., Diakité, A., Krijnen, T., Ledoux, H., Stoter, J., 2017. Towards an integration of gis and bim data: What are the geometric and topological issues. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences 4.

Pérez, E., Salamanca, S., Merchán, P., Adán, A., 2016. A comparison of hole-filling methods in 3d. International Journal of Applied Mathematics and Computer Science 26.

Rottensteiner, F., Sohn, G., Gerke, M., Wegner, J.D., Breitkopf, U., Jung, J., 2014. Results of the isprs benchmark on urban object detection and 3d building reconstruction. ISPRS Journal of Photogrammetry and Remote Sensing 93.

Sieger, D., Botsch, M., 2020. The polygon mesh processing library.

Verdie, Y., Lafarge, F., Alliez, P., 2015. LOD Generation for Urban Scenes. Trans. on Graphics 34.

Zhao, J., Ledoux, H., Stoter, J., Feng, T., 2018. Hsw: Heuristic shrink-wrapping for automatically repairing solid-based citygml lod2 building models. ISPRS Journal of Photogrammetry and Remote Sensing 146.

Zhao, J., Stoter, J., Ledoux, H., 2014. A framework for the automatic geometric repair of citygml models, in: Cartography from pole to pole.