Automation in Construction 103 (2019) 80–103

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

An algorithm to facet curved walls in IFC BIM for building energy analysis

Huaquan Ying, Sanghoon Lee[*]

The Department of Civil Engineering, The University of Hong Kong, Hong Kong

A R T I C L E I N F O

Keywords:
Building energy model
Building Information Model (BIM)
Curved wall
Building geometric information
Industry Foundation Class (IFC)

A B S T R A C T

The automatic transformation of a Building Information Model (BIM) in Industry Foundation Classes (IFC) to a building energy model (BEM) demonstrates significant benefits on the improvement of efficiency and accuracy in a building energy modeling process. However, so far, building objects with curved surfaces in IFC models are not supported by existing transformation approaches as they only handle polyhedral geometries. This study presents an algorithm to automatically facet curved walls and convert their geometries into polyhedrons, so that they can be further processed by existing transformation approaches. In the process of faceting curved walls, geometric connections between each target wall and adjacent walls are detected and handled to maintain the connections in the results. Furthermore, the algorithm updates the geometries of relevant building objects, including straight walls connected to curved walls, openings (i.e., windows and doors) on curved walls and spaces enclosed by curved walls, based on the faceted geometries. This removes curved surfaces of these objects on the one hand, and maintains their geometric relationships on the other. Moreover, slabs with the curved surfaces are also faceted. This algorithm takes an IFC BIM that pertains to the model view definition Concept Design BIM 2010 as an input. It generates a new IFC BIM, in which all curved walls and relevant building objects are represented as polyhedral geometries with correct geometric relationships. A prototype application that implements the algorithm was developed to evaluate the performance of the algorithm with a simple building model and a complex real-world building model. In the results, all the curved walls were faceted, and all relevant straight walls, openings, spaces and slabs were correctly updated, which indicated that the algorithm worked as intended.

1. Introduction

1.1. Background

The building sector, which consumes around 35% - 40% of energy use in the world, is one of the major energy consumption sectors [1,2]. In Hong Kong, residential and commercial buildings even account for 64% of total energy end-use from 2004 to 2014 [3]. To achieve ambitious reductions in building energy consumptions, building energy performance (BEP) becomes an increasingly common and important factor considered in building design and operations. Building energy modeling, which provides a powerful and computerized method to predict or assess BEP, is thus widely used to quantitatively justify relevant decisions on building design and operations [4,5].

Building geometry is an essential input to a building energy model (BEM). It generally consumes the largest portion (up to 80%) of the effort in generating a BEM [6]. In a traditional building energy modeling process, building geometric information is often manually prepared by energy modelers using design documents (e.g., 2D CAD drawings, specifications, and photos), which is laborious and error prone [5,7]. In addition, building geometric models may not be consistently reproduced as different modelers may have different interpretations of the design documents [7].

A Building Information Model (BIM), which provides a digital representation of building physical and functional characteristics [8], contains detailed building geometry information that can be utilized to create a BEM. Studies have shown that automating geometric information transformation from BIM to BEM provides an effective solution to address those problems found in the traditional building energy modeling process [4,9–12]. Various transformation approaches have been developed, and open data schemas such as Industry Foundation Classes (IFC) and Green Building XML (gbXML) have been commonly used to represent and convey building information [5,13,14].

IFC is an open data schema for openBIM that aims to promote the sharing and exchanging of accurate building information throughout the building life cycle [15]. It provides multiple solid modeling approaches (e.g., Boundary Representation and swept area solid

* Corresponding author.
E-mail addresses: u3004315@connect.hku.hk (H. Ying), sanghoon.lee@hku.hk (S. Lee).

https://doi.org/10.1016/j.autcon.2019.03.004
Received 9 March 2018; Received in revised form 2 February 2019; Accepted 6 March 2019

modeling) to precisely define three-dimensional (3D) building geometry. Furthermore, it offers the concept of space boundary (SB), by which building geometry can be defined as systems of planar surfaces that enclose thermal zones [16]. This planar surface-based building geometry definition is used by most building energy modeling tools (ibid). Hence, the SB plays a critical role in the process of preparing geometric inputs to create a BEM from IFC BIM. Various workflows have been developed and they can be generally distinguished into three types:

- Exporting an IFC model with SBs from a BIM authoring tool and subsequently importing that model into BEM. This type of workflows relies on the use of building energy modeling tools (e.g., Simergy™ [17]) that provide the IFC import function.
- Exporting an IFC model with SBs from a BIM authoring tool, and then extracting and converting the SBs to input formats of building energy modeling tools. This type of workflows is suitable to the tools that cannot directly import IFC models. Currently, several format converters, such as IFC-Modelica [18], IFC-DOE-2 INP [19], and IFC-EnergyPlus IDF [20,21], have been developed.
- Exporting an IFC model with solid geometries from a BIM authoring tool, enriching it with SBs, and then importing the enriched model into building energy modeling tools or converting it into specific input formats of building energy modeling tools [12,22–24]. In this type of workflows, the initially exported IFC model does not carry SBs required for building energy modeling. Thus, an additional enrichment step to generate such geometric data is needed.

The first two types of workflows use BIM authoring tools such as Revit [25] and ArchiCAD [26] to directly export an IFC model with SBs. Nevertheless, outputs from the export functions of these tools often have errors (e.g., incomplete and incorrect exporting), especially when exporting complex-shape building models, and thus the functions are still unreliable [22,27,28]. Fig. 1 shows a test of SB generation functions in two commonly used BIM authoring tools. Although the test building is simple, problems and errors are found from processing curved geometries in both IFC SB geometric models, as stated in the figure.

By including a step of extracting SBs from IFC solid geometries, the third type of workflows are more flexible than the other two as they do not require to use BIM authoring tools that export SBs. Furthermore, they can avoid the quality issues of SBs from the exporting functions. As an effort to facilitate the third type of workflows, the Space Boundary Tool (SBT) [29], developed by Lawrence Berkeley National Laboratory, aims to transform a building geometric model in IFC to a BEM in the EnergyPlus IDF format [4]. This tool implements a graph theory-based algorithm [11] to compute SBs from an input IFC solid model. One of the main limitations is that it cannot process building objects with curved surfaces and its outputs do not often fulfil all the requirements for energy analysis tools [29]. OpenStudio provides an IFC import utility which utilizes the BIMserver as middleware to transform IFC solid geometries into SBs in the OpenStudio Model (OSM) format [30]. Similar to the SBT, the utility is limited to handling building objects in regular shapes (ibid). El-Diraby et al. [22] proposed an online BIM-based system for collaborative design and socio-technical analytics of green buildings. This system also includes a module that translates an IFC building geometry model to an OSM model for OpenStudio. Again, the underlying SB generation algorithm is still limited to non-curved building objects. Other SB generation algorithms can be found in [10,31]. All these algorithms start with an IFC model, in which building objects are geometrically represented as polyhedrons with flat polygonal faces.

In summary, existing efforts in the third type of workflows find it challenging to process building objects with curved surfaces. They require curved building objects to be manually segmented as a collection of polyhedral partitions in BIM authoring tools. This would add a large amount of extra remodeling work, especially when a building model is large and contains many curved objects. More importantly, the manual process may distort the original geometry definition because of inappropriate human decisions on the segmentation methods.

To address this issue, Ladenhauf et al. [9] suggested a pre-processing step that triangulates solid geometries of building objects so as to eliminate curved surfaces before extracting SBs. However, triangulating curved building objects individually may cause incorrect connections (e.g., geometric clashes) between adjacent building objects, which are often not allowed by SB generation algorithms (e.g., [10,11], and [31]).

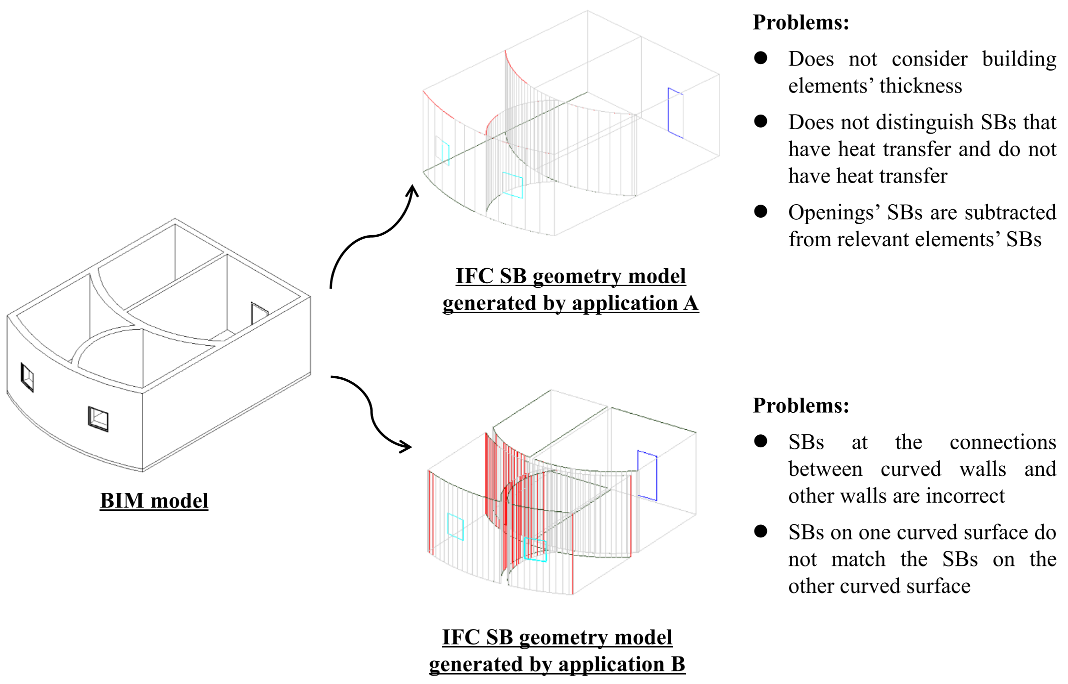

Problems:
Does not consider building elements' thickness
Does not distinguish SBs that have heat transfer and do not have heat transfer
Openings' SBs are subtracted from relevant elements' SBs
IFC SB geometry model generated by application A
BIM model
Problems:
SBs at the connections between curved walls and other walls are incorrect
SBs on one curved surface do not match the SBs on the other curved surface
IFC SB geometry model generated by application B

Fig. 1. A test of IFC SBs exporting functions in two BIM authoring tools.

Furthermore, triangulation may be not an efficient way to approximate some common curved shapes (e.g. thin walled cylinders) as it may produce many tiny triangle patches, which would significantly affect the running efficiency of building energy modeling tools. Kim and Yu [32] developed a process to divide curved walls in IFC models into several straight wall segments for energy analysis. However, this process focuses on single curved walls only and lacks the consideration of their geometrical relationships with other building objects such as other walls, spaces, and openings. Thus this would also result in incorrect connections between the divided walls and relevant building objects.

1.2. Research aim

In order to address the challenge of transforming IFC models with curved building objects into BEMs, this study presents an algorithm that automatically identifies and facets curved geometries in IFC. The resulting polyhedral geometries can be further processed by existing IFC BIM-to-BEM transformation tools or approaches following the third type of workflows. This study particularly focuses on curved walls as they widely exist in typical buildings and have influential impacts on building energy performance. Furthermore, the proposed algorithm takes into account the geometrical relationships between curved walls and relevant building objects. As shown in Fig. 2, the relevant building objects include straight walls connected to curved walls, openings including doors and windows hosted by curved walls, spaces enclosed by curved walls, and slabs bearing curved walls. The geometries of these objects are updated to fit relevant faceted walls. This eliminates their curved surfaces on the one hand and maintains correct geometrical relationships between them and faceted walls on the other.

Fig. 3 shows a proposed workflow which integrates the proposed algorithm with existing IFC BIM - to - BEM transformation methods: first, an IFC model is prepared and exported from a BIM authoring tool; second, the proposed algorithm facets curved walls in the IFC model and update relevant building objects; and finally, the processed IFC model with polyhedral geometries only is transformed by existing tools or approaches into a BEM in different formats for different building energy modeling tools.

2. Research scope

This study focuses on a common shape of curved walls with the following features:

- Having an unchanged thickness along the wall path;

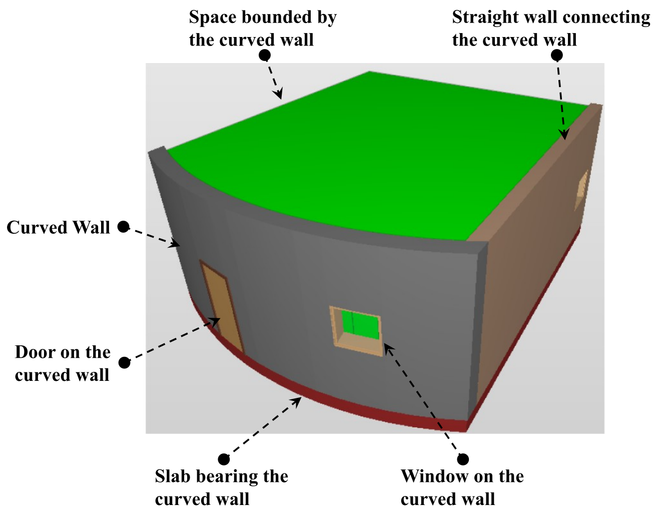

Space bounded by the curved wall
Straight wall connecting the curved wall
Curved Wall
Door on the curved wall
Slab bearing the curved wall
Window on the curved wall

Fig. 2. Curved wall and relevant connected building objects considered by the proposed algorithm.

- Having a planar footprint bounded by a composite curve consisting of circular arcs and/or line segments only (i.e., curved surfaces of curved walls are cylindrical); and
- Being able to be represented by a linear extrusion from a planar footprint or a result of clipping the linear extrusion by planes. This means that the study targets not only vertical walls (when the extrusion axis is equal to the Z axis) with a uniform height (when there is no clipping), but also non-vertical walls with a non-uniform height. In IFC, curved walls with above-mentioned features are defined by IfcWallStandardCase (if vertical) or IfcWall (if non-vertical) [33,34]. Their geometries can be defined as a ‘SweptSolid’ representation by IfcExtrudedAreaSolid (if the height is uniform) or a ‘Clipping’ representation by IfcBooleanClippingResult (if the height is non-uniform) (ibid).

Similar to curved walls, building objects including straight walls, openings, spaces, and slabs targeted in this study need to have the following shape features:

- Having a planar footprint bounded by a composite curve that consists of circular arcs and/or line segments only; and
- Being able to be represented by a linear extrusion from a planar footprint (i.e., IfcExtrudedAreaSolid) or a result of clipping the linear extrusion by planes (i.e., IfcBooleanClippingResult).

3. Algorithm requirements

3.1. Input requirements

The IFC specifications provide rich approaches to define building object geometries. For example, a wall solid geometry can be defined as a swept solid (e.g., IfcExtrudedAreaSolid), a B-rep (e.g., IfcFacetedBrep), or a result of a Boolean operation on two other solids (e.g., IfcBooleanClippingResult) [35]. Even for a specific representation approach, the selection of IFC model structures and IFC entities to organize the geometric information can be diverse. All these make it difficult and troublesome to extract building geometric information from an IFC model and analyze the shape feature of a building object.

To consider the practicality of the proposed algorithm, an input IFC model is required to comply with the model view definition (MVD) called Concept Design BIM 2010 (CDM 2010) [36]. CDM 2010 covers information exchange requirements for building energy modeling and has been implemented in several widely used BIM authoring tools such as Revit [25] and ArchiCAD [26]. In IFC models complying with the MVD, the geometric data of building objects required by the proposed algorithm is defined based on unambiguous IFC data structures. This greatly eases the extraction and processing of the geometric data. Furthermore, the MVD is based on IFC 2 × 3 TC1 schema. Most existing IFC BIM-to-BEM transformation tools (e.g., SBT [29]) and energy simulation tools (e.g., Simergy™ [17]) that can directly import IFC models still work only on IFC2x3 based models.

Several geometry representations (including IfcBoundingBox, IfcFacetedBrep, IfcFacetedBrepWithVoids, IfcExtrudedAreaSolid, IfcBooleanClippingResult, and IfcMappedItem) and their corresponding implementation methods are specified in CDM 2010. Among them, IfcExtrudedAreaSolid and IfcBooleanClippingResult are capable of accurately defining the curved walls targeted in this study and relevant building objects that have geometric relationships with the curved walls. The specific IFC data structure in CDM 2010 with the two IFC entities is explained later in Section 4.2.

It is note-worthy that the MVD concept of SB is mandatorily required in CDM 2010. This means that when exporting an IFC model in the mode of CDM 2010 from a BIM authoring tool, the model would already have SBs. However, as explained earlier, they often carry errors. Therefore, the proposed algorithm needs to detect and remove these original SBs.

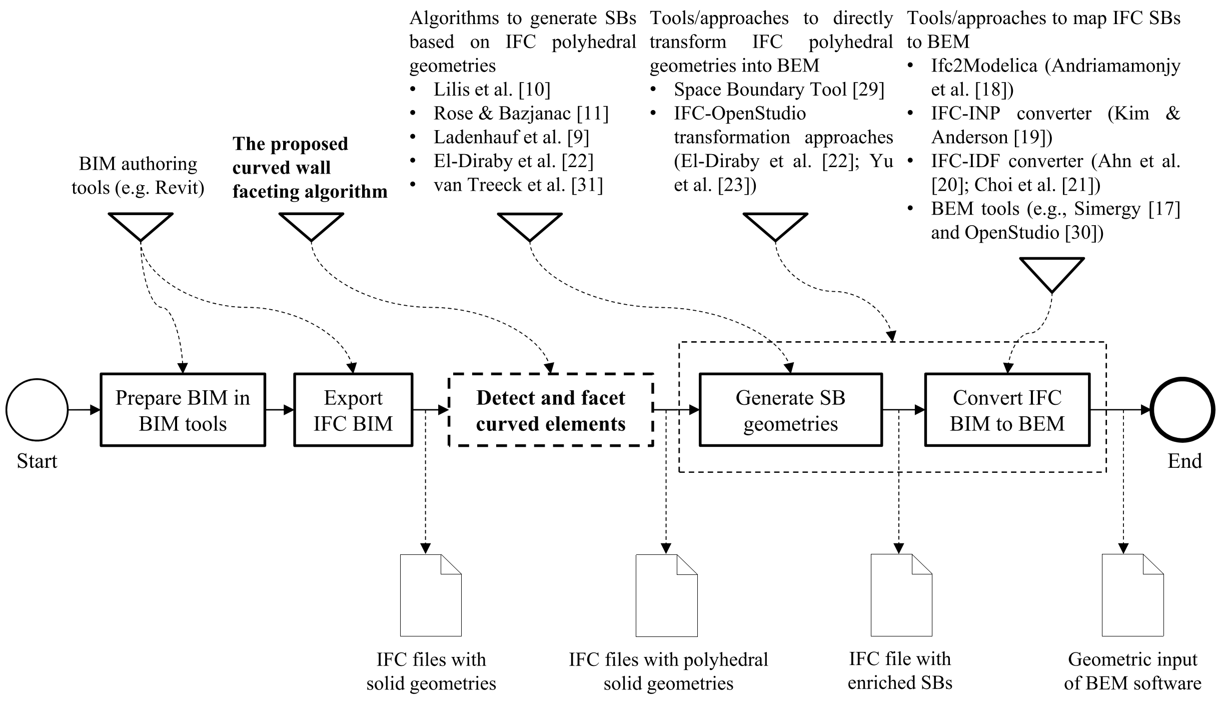

Algorithms to generate SBs based on IFC polyhedral geometries
• Lilis et al. [10]
• Rose & Bazjanac [11]
• Ladenhauf et al. [9]
• El-Diraby et al. [22]
• van Treeck et al. [31]
Tools/approaches to directly transform IFC polyhedral geometries into BEM
• Space Boundary Tool [29]
• IFC-OpenStudio transformation approaches (El-Diraby et al. [22]; Yu et al. [23])
Tools/approaches to map IFC SBs to BEM
• Ifc2Modelica (Andriamamonjy et al. [18])
• IFC-INP converter (Kim & Anderson [19])
• IFC-IDF converter (Ahn et al. [20]; Choi et al. [21])
• BEM tools (e.g., Simergy [17] and OpenStudio [30])
BIM authoring tools (e.g. Revit)
The proposed curved wall faceting algorithm
Start
Prepare BIM in BIM tools
Export IFC BIM
Detect and facet curved elements
Generate SB geometries
Convert IFC BIM to BEM
End
IFC files with solid geometries
IFC files with polyhedral solid geometries
IFC file with enriched SBs
Geometric input of BEM software

Fig. 3. A proposed workflow integrating the curved wall faceting algorithm proposed in this study and existing IFC BIM - to - BEM transformation approaches to transform curved geometries.

3.2. Output requirements

To adapt results of faceting curved walls to existing IFC BIM-to-BEM transformation approaches, the output of the algorithm should meet the following requirements:

First, each curved surface of walls should be faceted as a collection of planar polygonal surfaces. As discussed earlier, curved geometries in IFC shall be converted into polyhedral geometries so that they can be further processed by existing IFC BIM-to-BEM transformation approaches.

Second, faceted surfaces of one original curved surface along the wall path shall have a parallel segment that belongs to the faceted surfaces of another curved surface, as these pairwise surfaces form a source of thermal SBs. Based on the "other side" information, the thermal SBs are classified into two types (see Fig. 4): (1) SBs having heat exchange (Type 2a), the other side of which is a thermal space or outside environment; and (2) SBs having no heat exchange (Type 2b), the other side of which is neither a thermal space nor outside environment [16]. Each internal Type 2a SB has a corresponding Type 2a SB belonging to the common wall, and they are parallel and geometrically symmetric [11], as shown in Fig. 4. Consequently, only faceted surfaces that satisfy this requirement can be further processed to generate correct thermal SBs.

Third, the geometry shape of curved walls shall be maintained as much as possible in the faceting results. This is important to preserve the geometry-related thermal features of original curved walls and the original spaces bounded by the curved walls.

Fourth, each curved wall after faceting shall have correct geometrical relationships with relevant building objects including connected walls, contained openings, enclosed spaces, and the slab that bears the curved wall. This requires the geometries of the relevant building objects to be updated or faceted to fit with the faceted curved walls. On the one hand, similar to curved walls, these objects are also essential to building energy modeling and their geometries need to be transformed. Therefore, their curved surfaces caused by the geometric connections with the curved walls need to be faceted as well. On the other hand, incorrect geometric relationships such as gaps and intersections in

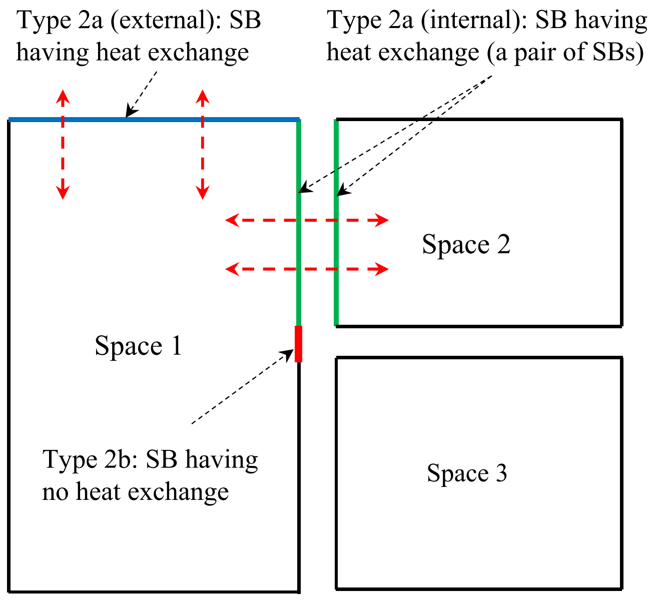

Type 2a (external): SB having heat exchange
Type 2a (internal): SB having heat exchange (a pair of SBs)
Space 1
Space 2
Space 3
Type 2b: SB having no heat exchange

Fig. 4. Two types of thermal SBs.

faceting results would fail downstream IFC BIM-to-BEM transformation processes.

4. The proposed algorithm

4.1. Overview of the algorithm

As specified in Section 2, the geometries of curved walls and relevant building objects (i.e., straight walls, openings, spaces and slabs) targeted in this study are represented as extruded area solids by IfcExtrudedAreaSolid or clipped extruded area solids by IfcBooleanClippingResult. In IfcExtrudedAreaSolid, an extruded area solid is defined by extruding a bounded planar surface (i.e., the footprint) along

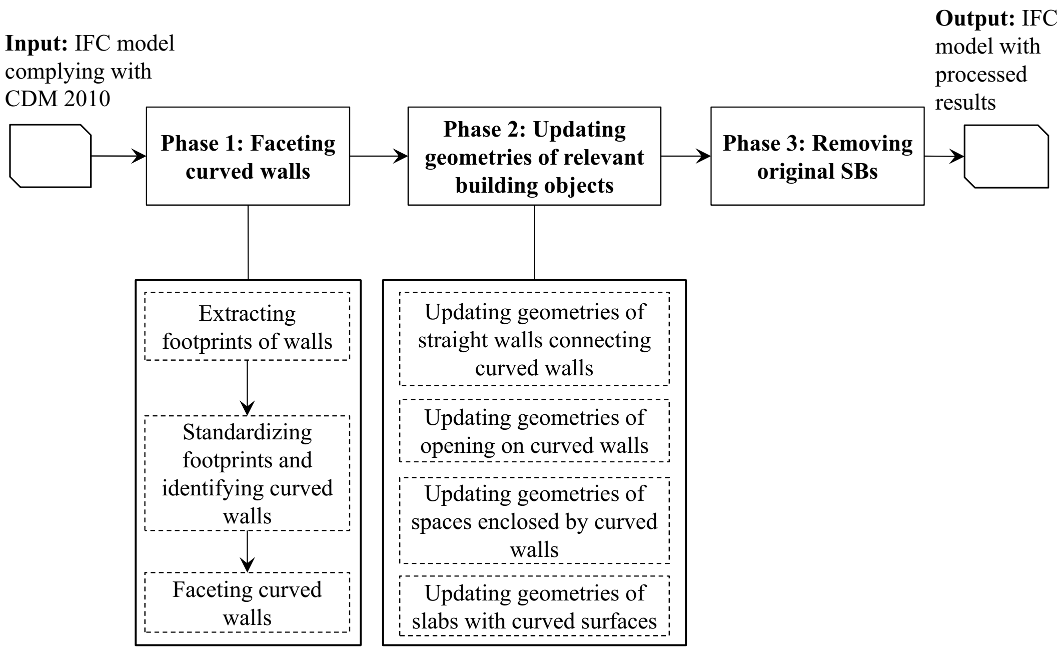

Input: IFC model complying with CDM 2010
Phase 1: Faceting curved walls
Phase 2: Updating geometries of relevant building objects
Phase 3: Removing original SBs
Output: IFC model with processed results
Extracting footprints of walls
Standardizing footprints and identifying curved walls
Faceting curved walls
Updating geometries of straight walls connecting curved walls
Updating geometries of opening on curved walls
Updating geometries of spaces enclosed by curved walls
Updating geometries of slabs with curved surfaces

Fig. 5. The main phases of the proposed algorithm.

with a linear direction and with a given extrusion length. The curved edges of the footprint correspond to the curved surfaces of the solid. Therefore, a curved solid represented by IfcExtrudedAreaSolid or IfcBooleanClippingResult can be faceted by approximating all curved edges in the footprint boundary into sets of line segments. With this concept in mind, this study proposes a footprint-based algorithm to facet curved walls and update all relevant building objects. The algorithm replaces their original footprint boundaries that contain curved edges with a closed polyline that consists of continuous line segments only.

The proposed algorithm has three primary phases (see Fig. 5). In the first phase, curved walls in an IFC model are identified and faceted in three steps: extracting footprints of walls (Section 4.2), standardizing footprints and identifying curved walls (Section 4.3), and faceting curved walls (Section 4.4). In the second phase (Section 4.5), straight walls with curved surfaces, openings on curved walls and spaces enclosed by curved walls are detected and their geometries are updated based on the results from faceting corresponding curved walls. Slabs with curved surfaces are also identified and faceted. In the third phase (Section 4.6), the original SBs are removed so the resulting IFC model can be enriched with new valid SBs by existing IFC BIM-to-BEM transformation approaches.

4.2. Extracting footprints of walls

The footprints of walls are extracted from their geometry representations in IFC models following the schema of CDM 2010. As discussed earlier, walls in this study are defined by IfcWallStandardCase or IfcWall and their geometries are represented by IfcExtrudedAreaSolid or IfcBooleanClippingResult. Fig. 6 shows the IFC structure and entities used to define the solid geometry of a wall in CDM 2010.

As shown in Fig. 6, the footprint of a wall is specified by the SweptArea attribute of IfcExtrudedAreaSolid, which refers to an IfcArbitraryClosedProfileDef instance. There are two methods to define the footprint: (1) using IfcArbitraryClosedProfileDef with a reference of IfcPolyline (denoted as IfcArbitraryClosedProfileDef – IfcPolyline), which defines a footprint as a closed polyline; (2) using IfcArbitraryClosedProfileDef – IfcCompositeCurve, which defines a footprint as a composite curve that consists of a collection of basic curves joined end-to-end. The basic curve is either a line segment defined by IfcPolyline or a circular arc by IfcTrimmedCurve. In summary, according to the types of contained geometric primitives, the footprints are classified into four representation forms:

- Form 1: a closed polyline (IfcArbitraryClosedProfileDef – IfcPolyline);
- Form 2: a collection of continuous line segments (IfcArbitraryClosedProfileDef – IfcCompositeCurve – IfcCompositeCurveSegment – IfcPolyline);
- Form 3: a collection of circular arcs (IfcArbitraryClosedProfileDef – IfcCompositeCurve – IfcCompositeCurveSegment – IfcTrimmedCurve);
- Form 4: a combination of line segments and circular arcs (IfcArbitraryClosedProfileDef – IfcCompositeCurve – IfcCompositeCurveSegment – IfcPolyline/IfcTrimmedCurve).

Therefore, extracting footprints in different representation forms is essentially based on the interpretation of three types of geometric primitives defined in IFC:

- A closed polyline or a line segment defined by IfcPolyline: a list of ordered vertices is extracted from the IfcCartesianPoint instances referenced by the IfcPolyline instance.
- A circular arc defined by IfcTrimmedCurve: four types of information including coordinates of the center point, radius, coordinates of two trimming points (i.e., two endpoints) and the arc direction (i.e., clockwise or count-clockwise from one endpoint to another) are extracted from the IfcTrimmedCurve instance and its referenced instances.

4.3. Standardizing footprints and identifying curved walls

This step aims to identify curved walls based on the extracted footprint representations, which, however, is complicated due to the multifarious representations of footprints (see Section 4.3.1). To address this issue, a footprint standardization process that transforms these various representations into the "standard" form is introduced. Then the shape of a wall is inferred from its standard footprint.

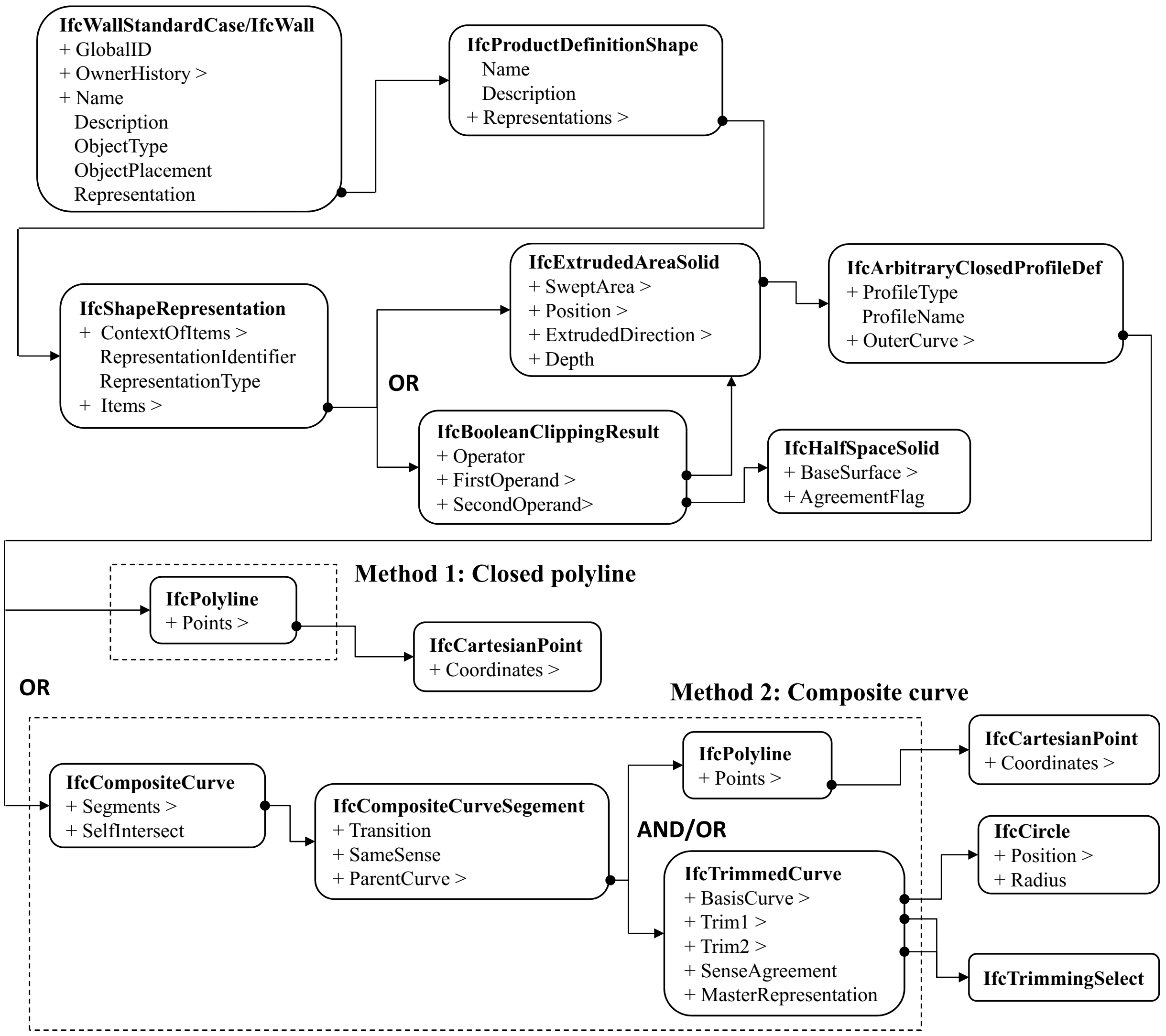


Fig. 6. Instantiation diagram to represent an extruded area solid and a clipped solid of a wall in CDM 2010 [36]. Note: the FirstOperand of IfcBooleanClippingResult can also refer to another IfcBooleanClippingResult when there is more than one clipping. For clarity, it directly refers to IfcExtrudedAreaSolid.

4.3.1. Multifarious representations of footprints

According to Section 4.2, extracted footprints have four representation forms. Naturally, walls with footprints containing circular arc boundaries (i.e., footprints in Form 3 and Form 4) are the candidates of curved walls. In IFC models, however, a circular arc may also be approximately defined as a collection of line segments (see Fig. 7(a)). This means even if a footprint is only bounded by a polyline (i.e., Form 1) or a set of line segments (i.e., Form 2), the wall still could be curved. Furthermore, one straight edge may also be divided into several smaller line segments (see Fig. 7(b)), which adds more complexities to interpret the footprint feature. Fig. 8 illustrates the representative footprints of curved walls and straight walls in different representation forms: walls in Form 1 & Form 2 can be curved or straight; walls in Form 3 must be curved as two long edges are circular arcs; and walls in Form 4 can be curved or straight.

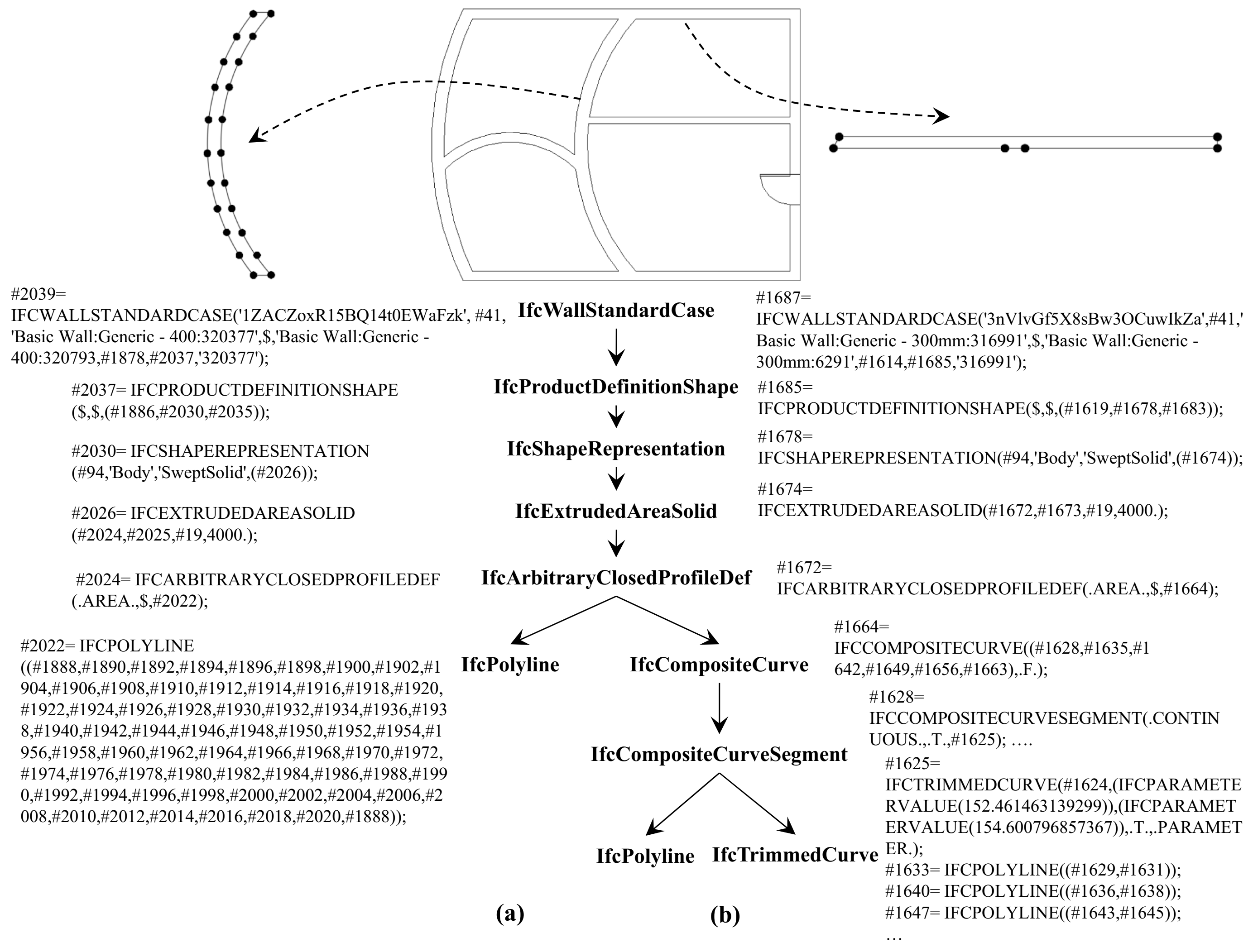
4.3.2. Footprint standardization

The footprint standardization aims to transform the various footprint representations into a “standard” form for the downstream processing. The standard footprint is defined as a closed boundary that consists of four connected edges including two long edges and two short edges. Each edge is defined by a geometric primitive, which is either a circular arc (i.e., arc edge) or a line segment (i.e., straight edge). Fig. 9 shows the standard footprint of a typical curved wall and a typical straight wall. It is apparent that the footprint in Form 3 where all circular arc edges are explicitly defined is the standard from; while, as discussed in Section 4.3.1, footprints in Form 1, Form 2 and Form 4 may be not and need to be standardized. Geometrically, Form 1 and Form 2 are equivalent as footprints in both forms are essentially composed of line segments. Hence, they are discussed together hereafter.

The proposed standardization method consists of four steps (see Fig. 10(a)): temporary footprint construction, corner vertex detection, reconstruction of temporary footprint edges and standard footprint construction. Four illustrative examples are shown in Fig. 10(b)–(e).

(1) Temporary footprint construction

The temporary footprint is a closed polyline constructed by using the endpoints of line segments and/or circular arcs in the extracted footprint. A footprint in Form 1 can be directly used as the temporary footprint. For a footprint in Form 2, the endpoints of all line segments are extracted and stored in order and duplicate points (i.e., endpoints shared by two adjacent line segments) are removed. For a footprint in Form 4, the endpoints of all line segments and all circular arcs are stored to construct a corresponding temporary footprint. Similarly, duplicate endpoints need to be removed. During the construction all the



```
#2039=
IFCWALLSTANDARDCASE('1ZACZoxR15BQ14t0EWaFzk', #41,
'Basic Wall:Generic - 400:320377',$,'Basic Wall:Generic -
400:320793,#1878,#2037,'320377');

        #2037= IFCPRODUCTDEFINITIONSHAPE
        ($,$,(#1886,#2030,#2035));

        #2030= IFCSHAPEREPRESENTATION
        (#94,'Body','SweptSolid',(#2026));

        #2026= IFCEXTRUDEDAREASOLID
        (#2024,#2025,#19,4000.);

        #2024= IFCARBITRARYCLOSEDPROFILEDEF
        (.AREA.,$,#2022);

#2022= IFCPOLYLINE
((#1888,#1890,#1892,#1894,#1896,#1898,#1900,#1902,#1
904,#1906,#1908,#1910,#1912,#1914,#1916,#1918,#1920,
#1922,#1924,#1926,#1928,#1930,#1932,#1934,#1936,#193
8,#1940,#1942,#1944,#1946,#1948,#1950,#1952,#1954,#1
956,#1958,#1960,#1962,#1964,#1966,#1968,#1970,#1972,
#1974,#1976,#1978,#1980,#1982,#1984,#1986,#1988,#199
0,#1992,#1994,#1996,#1998,#2000,#2002,#2004,#2006,#2
008,#2010,#2012,#2014,#2016,#2018,#2020,#1888));
```

```
#1687=
IFCWALLSTANDARDCASE('3nVlvGf5X8sBw3OCuwIkZa',#41,'
Basic Wall:Generic - 300mm:316991',$,'Basic Wall:Generic -
300mm:6291',#1614,#1685,'316991');
#1685=
IFCPRODUCTDEFINITIONSHAPE($,$,(#1619,#1678,#1683));
#1678=
IFCSHAPEREPRESENTATION(#94,'Body','SweptSolid',(#1674));
#1674=
IFCEXTRUDEDAREASOLID(#1672,#1673,#19,4000.);
#1672=
IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,$,#1664);
    #1664=
    IFCCOMPOSITECURVE((#1628,#1635,#1
    642,#1649,#1656,#1663),.F.);
      #1628=
      IFCCOMPOSITECURVESEGMENT(.CONTIN
      UOUS.,.T.,#1625); ….
        #1625=
        IFCTRIMMEDCURVE(#1624,(IFCPARAMETE
        RVALUE(152.461463139299)),(IFCPARAMET
        ERVALUE(154.600796857367)),.T.,.PARAMET
        ER.);
        #1633= IFCPOLYLINE((#1629,#1631));
        #1640= IFCPOLYLINE((#1636,#1638));
        #1647= IFCPOLYLINE((#1643,#1645));
        …
```

Fig. 7. The representations of wall footprints in an IFC model: (a) two circular arc edges of a curved wall are approximated as two set of line segments; (b) one straight edge of a straight wall is defined as three connected line segments.

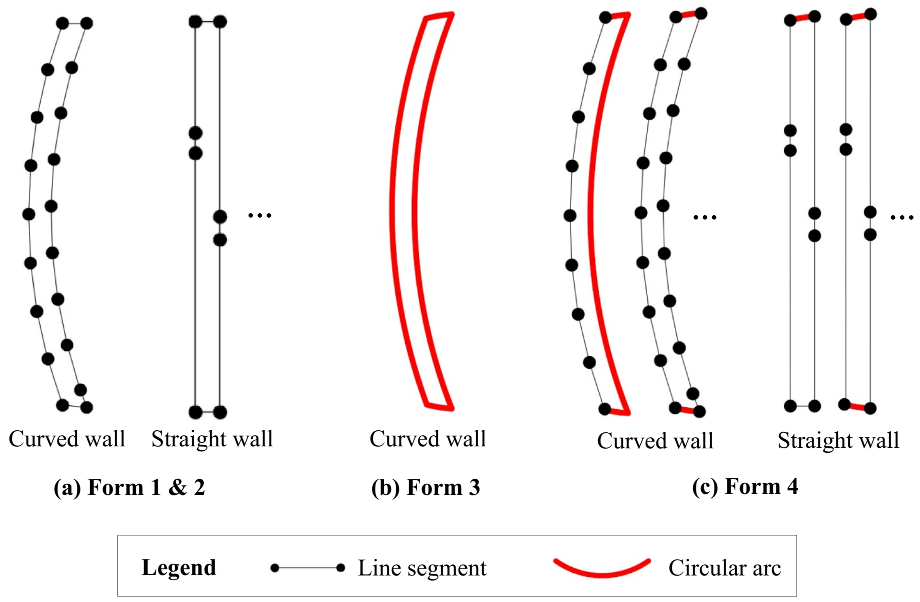


Fig. 8. The representative footprint representations of curved walls and straight walls in different forms.

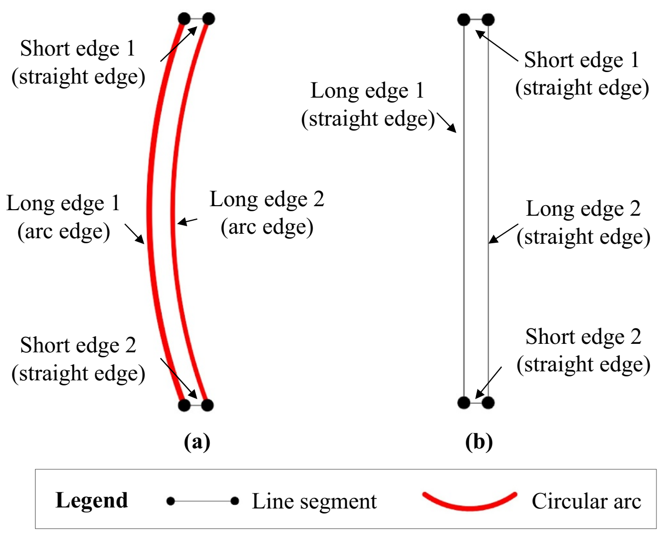


Fig. 9. Standard footprint representation form of: (a) a typical curved wall; (b) a typical straight wall.

original circular arcs are preserved for the subsequent steps.

(2) Corner vertex detection

A corner vertex refers to the common point shared by a long edge and a short edge of the standard footprint. The standard footprint has four corner vertices. Once all corner vertices are detected, all the endpoints in the temporary footprint are clustered into four groups. Each group contains the points belonging to one edge of the standard footprint.

A vector-based method is developed to detect corner vertices. First, for any two adjacent endpoints in the temporary footprint, this method computes a vector that represents the direction from one point to the other. All these direction vectors are stored in order. Second, the angle between two adjacent direction vectors is calculated. Each angle involves a group of three successive endpoints. The four corner vertices are included in four groups of points that correspond to the top-four largest angles. More specifically, the second point in each group is one of the four corner vertices. This is because if the second point is not a corner vertex, then the three successive points must belong to a same edge. In this case, the corresponding angle value would be relatively small (if the edge is curved) or even zero (if the edge is straight).

Fig. 11 illustrates this method with a curved wall in Form 1. Fig. 11(a) shows that the footprint is defined as a closed polyline with 68 ordered endpoints. By using the proposed method, the direction vectors between two adjacent endpoints are constructed (see Fig. 11(b)) and the angles of all the pairs of two adjacent direction vectors are calculated (see Fig. 11(c)). It is clear that four angle values are much larger than the others, and the involved groups of endpoints are: (#1950, #1952, #1954), (#1952, #1954, #1956), (#2018, #2020, #1888), and (#2020, #1888, #1890). The element in parentheses such as #1950 denotes an IfcCartesianPoint instance that defines an endpoint. From the results, four corner points (i.e. the second point in each group) namely #1952, #1954, #2020 and #1888 are recognized. Accordingly, all endpoints of the polyline are classified into four groups (see Fig. 11(b)): (#1888, #1890, …, #1952), (#1952, #1954), (#1954, #1956, …, #2020), and (#2020, #1888). Each group corresponds to an edge of the footprint.

(3) Reconstruction of temporary footprint edges

Based on the four groups of endpoints, four edges of the temporary footprint are reconstructed as four explicit geometric elements. The reconstruction operation obeys the following rules:

- If the number of endpoints in a group (N_G) is two, then this group represents a straight edge, which is defined as a line segment by the two endpoints.
- If N_G is more than two and angles of direction vectors corresponding

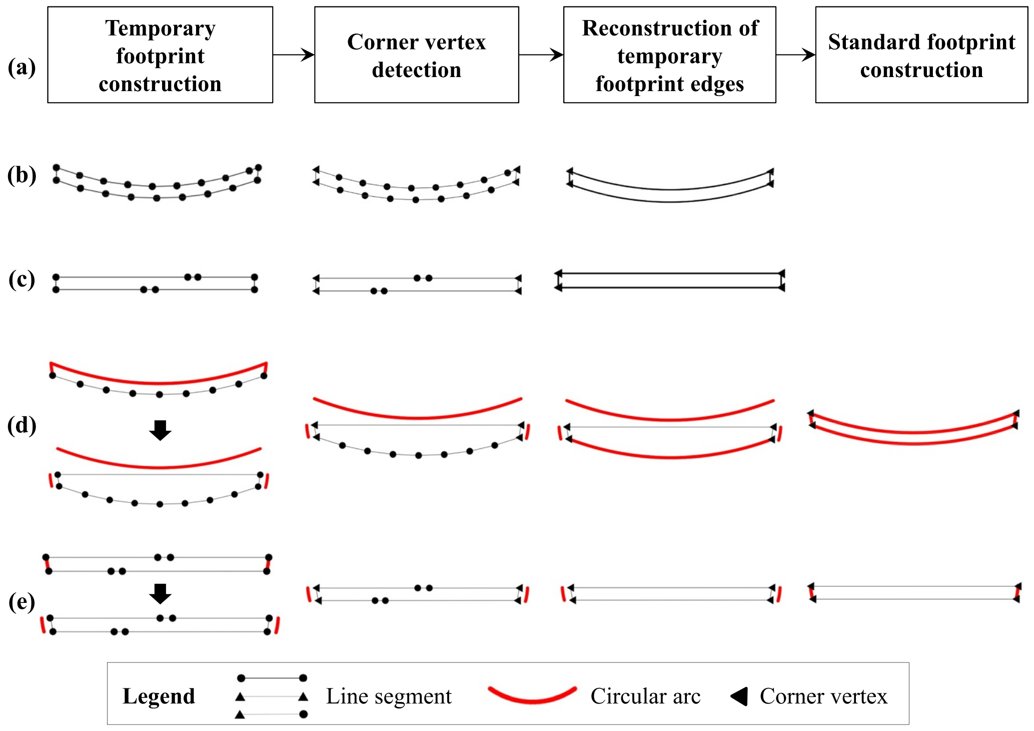
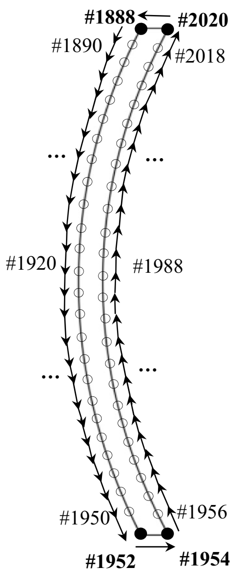
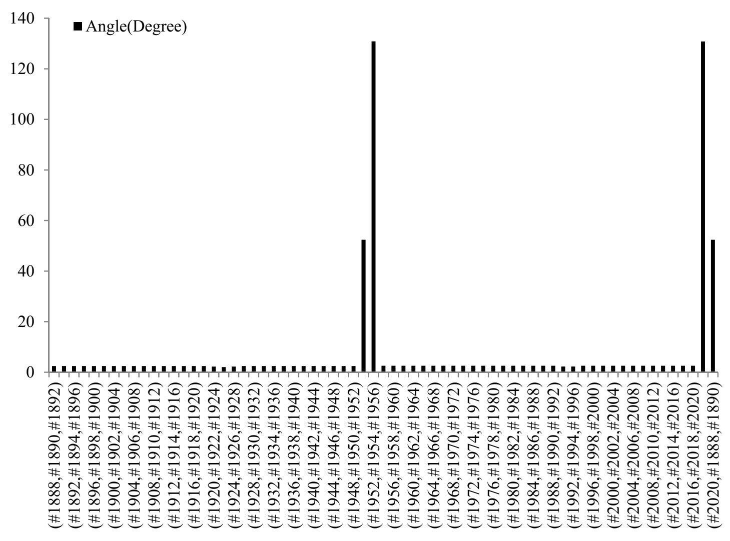

(a)
Temporary footprint construction
Corner vertex detection
Reconstruction of temporary footprint edges
Standard footprint construction
(b)
(c)
(d)
(e)
Legend
Line segment
Circular arc
Corner vertex

Fig. 10. Standardization of footprints in different representation forms: (a) standardization process; (b) example of a curved wall in Forms 1 & 2; (c) example of a straight wall in Forms 1 & 2; (d) example of a curved wall in Form 4; (e) example of a straight wall in Form 4.

```
#2022= IFCPOLYLINE
((#1888,#1890,#1892,#1894,#1896,#1898,#1900,#1902,#1904,#1906,#1908,#1910,#1912,#1914,#1916,#1918,#
1920,#1922,#1924,#1926,#1928,#1930,#1932,#1934,#1936,#1938,#1940,#1942,#1944,#1946,#1948,#1950,#19
52,#1954,#1956,#1958,#1960,#1962,#1964,#1966,#1968,#1970,#1972,#1974,#1976,#1978,#1980,#1982,#1984
,#1986,#1988,#1990,#1992,#1994,#1996,#1998,#2000,#2002,#2004,#2006,#2008,#2010,#2012,#2014,#2016,#
2018,#2020,#1888));
```

(a) Footprint in Form 1 (closed polyline)





(b) Vector construction **(c) Angles between two adjacent vectors**

Fig. 11. Identification of the corner points of a temporary footprint.

to the endpoints in a group (θ_G) are smaller or equal to a threshold of 0.5 degree, then this group represents a straight edge, which is defined as a line segment by the first and the last endpoints;

- If N_G is more than two and θ_G are larger than 0.5 degree, then this group represents an arc edge, which is defined as a circular arc. The circular arc is computed by: taking the first and the last endpoints in the group as two trimming points; and using the two trimming points and one other endpoint in the group to compute parameters including radius, the center point and the arc direction from one trimming point to the other.

(4) Standard footprint construction

The standardization of footprints in Forms 1 & 2 has been completed in Step (3), as their temporary footprints are geometrically equivalent to the standard form (see the examples in Fig. 10(b)–(c)). For footprints in Form 4, their temporary footprints need to be updated with the preserved circular arcs that are explicitly defined in original footprints (see the examples in Fig. 10(d)–(e)). More specifically, the update is to replace straight edges in the temporary footprints with corresponding circular arcs in the original footprints.

4.3.3. Curved wall identification

After the footprint of a wall is standardized, the shape feature is inferred by the following criteria: if two long edges in the standard footprint are circular arcs, the wall is identified as curved; otherwise, the wall is straight. The long and the short edges are distinguished by comparing their lengths. As an edge in the standard footprint is either a line segment or a circular arc, its length can be easily calculated.

4.4. Faceting curved walls

In this step, the identified curved walls are faceted while fulfilling the three output requirements specified in Section 3.2 (i.e., all boundary surfaces are planar polygons; meet the requirement of SB generation; and maintain original curved shape as much as possible). The basic idea of the algorithm is to replace original footprints of curved walls with closed polyline-based footprints, which is achieved by four steps: (1) detecting connections (see Section 4.4.1), (2) dividing arc edges (see Section 4.4.2), (3) segmenting arc edges (see Section 4.4.3), and (4) updating the footprint (see Section 4.4.4). An illustrative example is shown in Fig. 12.

4.4.1. Detecting connections

From the view of 2D footprints, connections between a curved wall and other walls are classified into two types (see Fig. 12(a)): (1) Type 1, which refers to connections between a long edge of the curved wall and short edges of other walls; (2) Type 2, which refers to a connection between a short edge of the curved wall and a long edge of another wall. Only Type 1 connections are detected as Type 2 connections do not affect the curved wall faceting process.

In IFC, the connectivity relationship between two walls can be defined by IfcRelConnectsPathElements. However, CDM 2010 does not mandatorily specify the requirement of connectivity information, which means that this information may be missing in an IFC model that pertains to this MVD. Therefore, the algorithm includes steps to detect Type 1 connections of a curved wall by a 2D geometry operation, as follows. First, walls that need to be checked for computing Type 1 connections are narrowed down by filtering out walls that do not have arc short edges in their standard footprints. Second, for each candidate wall, a circular arc – circular arc overlap test is performed on its short arc edges and the curved wall's long arc edges. The test succeeds if the following three conditions are satisfied: (1) two circular arcs ($C1$, $C2$) share a common center (e.g., Distance($C1_{center}$, $C2_{center}$) $\leq$ 0.01 mm); (2) two circular arcs have the same radius (e.g., $|C1_{radius} - C2_{radius}| \leq 1$ mm); and (3) two circular arcs have an intersection. The first two

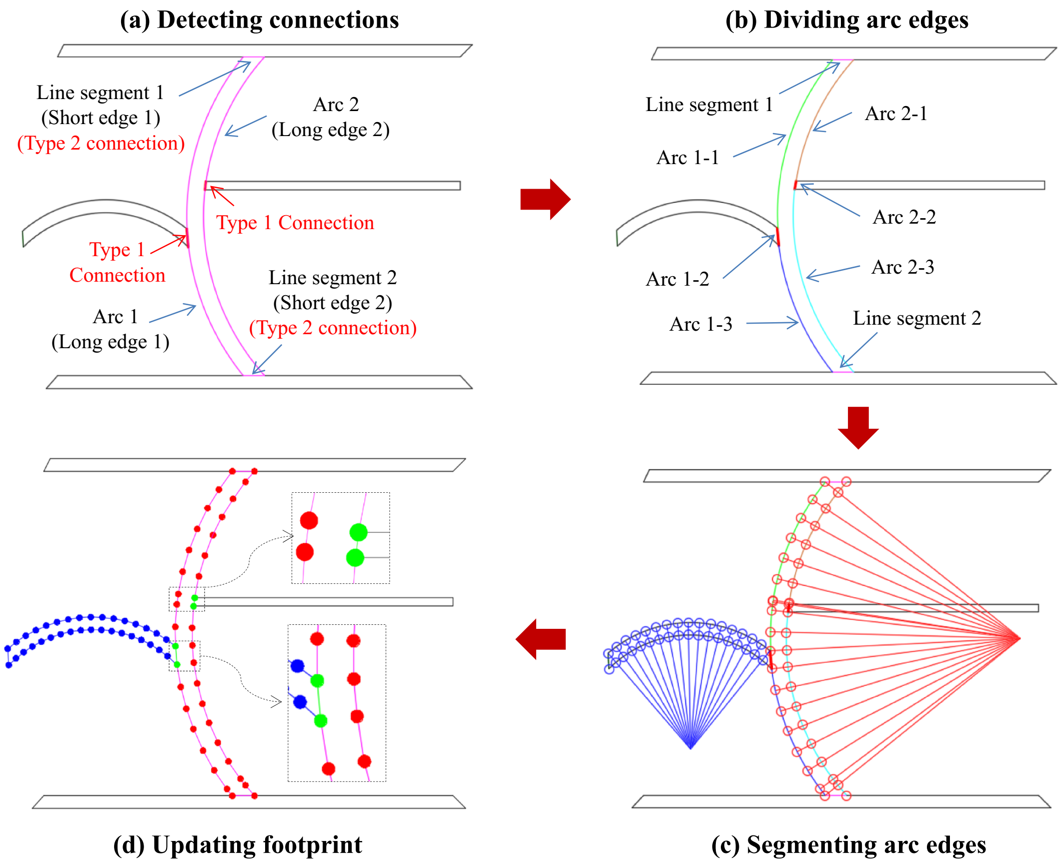


Fig. 12. Curved wall faceting steps.

conditions can be checked straightforwardly. For checking the third condition, the two circular arcs are pre-processed as counter-clockwise from the first trimming point to the second and then transformed into a common polar coordinate following the method detailed in Section 4.4.3. As a result, each circular arc can be simply represented as a linear interval (e.g., $[\theta_{Trim1}, \theta_{Trim2}]$) by its two trimming points in the polar coordinate. Then the intersection between two circular arcs can be easily computed. Once a short edge of a candidate wall passes the test, the resulting intersection would be recognized as a Type 1 connection of the curved wall.

It is worth noting that, while implementing the detection, the standard footprints of candidate walls defined in their own coordinate systems need to be transformed into the coordinate system where the footprint of the curved wall is defined. For each candidate wall, the transformation refers to pre-multiplying its footprint geometry with a corresponding transformation matrix ($T_{Ca\to Cu}$). In IFC, building objects are depicted in hierarchically organized coordinate systems, which generally follow the hierarchy of IFC spatial structure elements (i.e., IfcBuildingStorey, IfcBuilding, and IfcSite). Typically, a wall is defined in a local coordinate system (LCS) relative to the LCS of a building story, which is further referenced to a building. Given that candidate walls and the curved wall belong to a same building, the footprints of candidate walls are transformed into the LCS of the building first and then into the LCS of the curved wall. Accordingly, $T_{Ca\to Cu}$ of a candidate wall can be computed by

$$\begin{aligned} T_{Ca\to Cu} &= T_{Build\to Cu} \times T_{Ca\to Build} \\ &= (T_{Stor\_Cu\to Build} \times T_{Cu\to Stor\_Cu})^{-1} \times (T_{Stor\_Ca\to Build} \times T_{Ca\to Stor\_Ca}) \end{aligned} \quad (1)$$

where $T_{Ca\to Build}$, $T_{Build\to Cu}$, $T_{Ca\to Stor\_Ca}$, $T_{Stor\_Ca\to Build}$, $T_{Cu\to Stor\_Cu}$ and $T_{Stor\_Cu\to Build}$ refer to the transformation matrices from $\text{LCS}_{\text{CandidateWall}}$ to $\text{LCS}_{\text{Building}}$, from $\text{LCS}_{\text{Building}}$ to $\text{LCS}_{\text{CurvedWall}}$, from $\text{LCS}_{\text{CandidateWall}}$ to $\text{LCS}_{\text{BuildingStoreyContainingCandidateWall}}$, from $\text{LCS}_{\text{BuildingStoreyContainingCandidateWall}}$ to $\text{LCS}_{\text{Building}}$, from $\text{LCS}_{\text{CurvedWall}}$ to $\text{LCS}_{\text{BuildingStoreyContainingCurvedWall}}$, and from $\text{LCS}_{\text{BuildingStoreyContainingCurvedWall}}$ to $\text{LCS}_{\text{Building}}$, respectively. The latter four matrices can be directly derived from the IFC model.

4.4.2. *Dividing arc edges*

By taking the endpoints of detected connections as new trimming points, each corresponding long edge in the standard footprint of the curved wall is spilt into several ordered circular arc segments for the following processing step. These segments are distinguished as connection arc segments and non-connection arc segments. As an example in Fig. 12(a), both long edges (Arc 1 and Arc 2) of the curved wall have a connection with another wall. Accordingly, both are spilt into three arc segments based on the endpoints of the connections, i.e., (Arc 1–1, Arc 1–2, Arc 1–3) and (Arc 2–1, Arc 2–2, Arc 2–3), as shown in Fig. 12(b). Among these segments, Arc 1–2 and Arc 2–2 are connection arc segments and the others are non-connection arc segments.

4.4.3. *Segmenting arc edges*

In this step, two groups of circular arc segments that represent two long edges are segmented as two polylines. A ray-based segmentation method is developed to ensure that segmentation results are complied with the output requirements. The basic idea of this method is to construct a set of organized rays emitted from the center of the long edges and then compute the intersection points between these rays and the two groups of circular arc segments. The intersection points constitute the vertices of the two polylines, as shown in Fig. 12(c).

(1) Polar coordinate system (PCS)

A PCS is used for each footprint to help the construction of rays and the implementation of ray-circular arc intersection tests, as shown in Fig. 13. The PCS takes the center ***C*** of the long arc edges as the pole ***O***

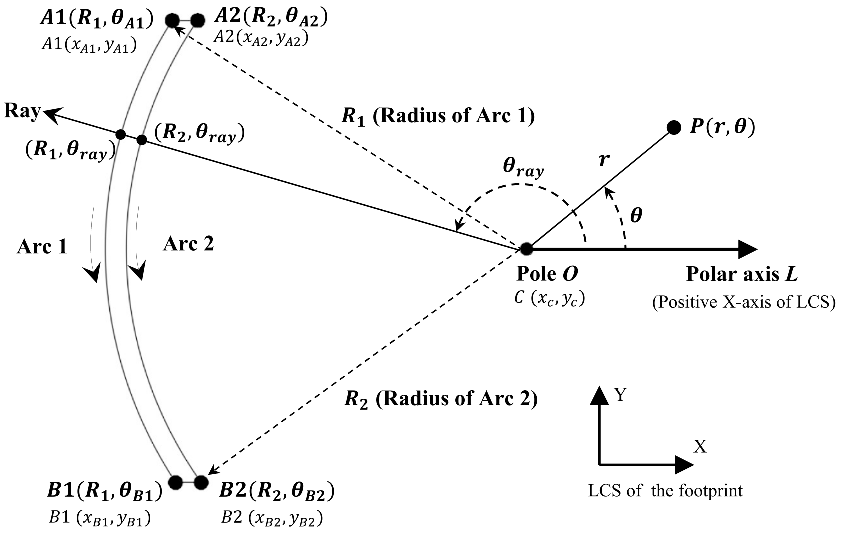


Fig. 13. Polar coordinate system for long arc edge segmentation.

and positive X-axis of the Cartesian LCS of the footprint as the polar axis L. Then a point $\boldsymbol{P}$ in the PCS is defined as (r, θ), where r refers to the Euclidean distance between $\boldsymbol{P}$ and $\boldsymbol{C}$, and θ refers to the angle measured from $\boldsymbol{L}$ in the counter-clockwise direction. A long arc edge of the footprint, e.g. Arc 1 in the figure, is thus described in the PCS by the following four parameters: radius R_1, the first trimming point $\boldsymbol{A1}(R_1, \theta_{A1})$, the second trimming point $\boldsymbol{B1}(R_1, \theta_{B1})$, and the arc direction. Furthermore, two settings are made to ensure that θ_{B1} is always larger than θ_{A1}, which greatly simplify the segmentation process: (1) setting the arc direction to be counter-clockwise. Exchange the role of two trimming points if the original direction is clockwise; and (2) computing θ_{B1} and θ_{A1} by the following formulas:

$$\theta_{A1} = \begin{cases} arcos\left(\dfrac{y_{A1} - y_c}{\sqrt{(y_{A1} - y_c)^2 + (x_{A1} - x_c)^2}}\right) & y_{A1} \geq y_c \\ 360 - arcos\left(\dfrac{y_{A1} - y_c}{\sqrt{(y_{A1} - y_c)^2 + (x_{A1} - x_c)^2}}\right) & y_{A1} < y_c \end{cases} \quad (2)$$

Let $\theta_1 = arcos\left(\frac{y_{B1} - y_c}{\sqrt{(y_{B1} - y_c)^2 + (x_{B1} - x_c)^2}}\right)$ and $\theta_2 = 360 - \theta_1$. Then,

$$\theta_{B1} = \begin{cases} \theta_1 & y_{B1} \geq y_c,\ \theta_1 \geq \theta_{A1} \\ 360 + \theta_1 & y_{B1} \geq y_c,\ \theta_1 < \theta_{A1} \\ \theta_2 & y_{B1} < y_c,\ \theta_2 \geq \theta_{A1} \\ 360 + \theta_2 & y_{B1} < y_c,\ \theta_2 < \theta_{A1} \end{cases} \quad (3)$$

where (x_{A1}, y_{A1}) and (x_{B1}, y_{B1}) refer to the Cartesian coordinates of $\boldsymbol{A1}$ and $\boldsymbol{B1}$ in the LCS of the footprint.

In the PCS, a ray emitted from the Pole $\boldsymbol{O}$ is defined by a polar angle θ_{ray} (see Fig. 13). The intersection between the ray and a circular arc, for example Arc 1, is tested by checking whether θ_{ray} is in the interval $[\theta_{A1}, \theta_{B1}]$. If so, the intersection exists, and the polar coordinate of the intersection point is (R_1, θ_{ray}) (see Fig. 13). Similarly, the intersection between the ray and another long arc edge (i.e., Arc 2) is denoted as (R_2, θ_{ray}). The Cartesian coordinates (x_{IP}, y_{IP}) of the intersection point (R_1, θ_{ray}) can be computed by

$$\begin{cases} x_{IP} = x_c + R_1 \times \cos\theta_{ray} \\ y_{IP} = y_c + R_1 \times \sin\theta_{ray} \end{cases} \quad (4)$$

(2) Segmentation method

The segmentation method, depicted in Fig. 14, takes the following as inputs (see Input section in the figure): two lists of circular arc segments (all segments are pre-processed such that their directions are counter-clockwise) that represent two long edges, a segmentation degree, four trimming points of the two long edges. All the circular arcs and points are defined in the PCS. All connection arcs are explicitly remarked in two lists for downstream processing.

The method approximately divides the two lists of circular arc segments into two polylines with incrementally constructed rays. The first and the last rays are created first (see Lines 2–3). The first ray is set to pass through the first trimming point of an edge. The selection of the edge depends on the shape of the wall end. The principle is that the ray built with one edge must intersect with the other. This makes sure that the shape of the wall end is maintained after the segmentation. For the same reason, the last ray must pass through the second trimming point of one arc edge and intersect with the other. Fig. 15 shows the construction of the first and the last rays in different cases. Other rays between the two rays are determined in a While loop (see Lines 5–29). Starting with the first ray (see Line 4), the intersection points between the ray and two lists of circular arc segments are detected, and the next ray is generated based on a set of rules. The loop repeats the operations until the ray (i.e., the polar angle) is not smaller than the last ray (see Line 5).

The procedure of new ray generation in each iteration is as follows. The first step is to find a circular arc segment (LA_i) that the current ray (r) intersects from the corresponding list of circular arc segments (LA) (see Line 7 in Fig. 14). The second step is to create a temporary new ray (r_{Anext}) for the LA by the following two rules: 1) if LA_i is not a connection arc, r_{Anext} is assigned with the minimal value between (r + $SegementDegreee$) and LA_{i_Trimm2} (see Lines 8–11); and 2) if LA_i is a connection arc, then it should not be segmented and the r_{Anext} is set as LA_{i_Trimm2} (see Lines 12–15). The third step is to repeat the previous two steps for another list of circular arc segments (LB) and get a corresponding temporary ray (r_{Bnext}) (see Line 19). The last step is to generate a new ray (r_{next}) for the While loop based on the two temporary rays: 1) if both temporary rays intersect a connection circular arc segment, then the smaller one is selected as r_{next} (see Lines 20–21); 2) if only one temporary ray intersects a connection circular arc segment, then the temporary ray is selected as r_{next} (see Lines 22–25); and 3) if neither intersects a connection circular arc segment, then the larger one is selected as r_{next} (see Lines 26–27).

While executing the loop, the endpoints of two lists of circular arc segments and the intersection points between constructed rays and these circular arc segments are progressively inserted into two corresponding output lists (i.e., $\boldsymbol{R_A}$ and $\boldsymbol{R_B}$ in Fig. 14; see Lines 9 and 13 in the figure). Furthermore, as the loop ends before the last ray is reached,

```
Input: Arc segments of long edge A: $LA = \{LA_1, LA_2, \dots, LA_m\}$ (Connection arcs are labelled);
       Arc segments of long edge B: $LB = \{LB_1, LB_2, \dots, LB_n\}$ (Connection arcs are labelled);
       Segmentation degree for an arc: $SegmentDegree$;
       Trimming points of A and B: $r_{A\_Trim1}$, $r_{A\_Trim2}$, $r_{B\_Trim1}$, $r_{B\_Trim2}$.
Output: Segmentation points of long edge A: $\boldsymbol{R_A}$;
        Segmentation points of long edge B: $\boldsymbol{R_B}$.
1:  Initialize $\boldsymbol{R_A} = \{\}$ and $\boldsymbol{R_B} = \{\}$;
2:  Construct the first ray $r_{first} = \mathrm{Max}(r_{A\_Trim1}, r_{B\_Trim1})$;
3:  Construct the last ray $r_{last} = \mathrm{Min}(r_{A\_Trim2}, r_{B\_Trim2})$;
4:  $r = r_{first}$;
5:  while $r < r_{last}$ do
6:      for $i = 1 : m$ do
7:          if $r \geq LA_{i\_Trim1}$ && $r < LA_{i\_Trim2}$ then    // Ray $r$ intersects with arc segment $LA_i$
8:              if $LA_i$ is not a connection arc then
9:                  $\boldsymbol{R_A}$.Add ($LA_{i\_Trim1}$), $\boldsymbol{R_A}$.Add ($r$);
10:                 $r_{Anext} = \mathrm{Min}(r + SegmentDegree, LA_{i\_Trim2})$;
11:                 break;
12:             else
13:                 $\boldsymbol{R_A}$.Add ($LA_{i\_Trim1}$), $\boldsymbol{R_A}$.Add ($LA_{i\_Trim2}$);
14:                 $r_{Anext} = LA_{i\_Trim2}$;
15:                 break;
16:             end if
17:         end if
18:     end for
19:     repeat the same process from Line 6 to Line 18 for $LB$ to get $\boldsymbol{R_B}$ and $r_{Bnext}$;
20:     if Intersection($r_{Anext}$, connection arcs in $LA$) && Intersection($r_{Bnext}$, connection arcs in $LB$) then
21:         $r = \mathrm{Min}(r_{Anext}, r_{Bnext})$;
22:     else if Intersection($r_{Anext}$, connection arcs in $LA$) then
23:         $r = r_{Anext}$;
24:     else if Intersection($r_{Bnext}$, connection arcs in $LB$) then
25:         $r = r_{Bnext}$;
26:     else
27:         $r = \mathrm{Max}(r_{Anext}, r_{Bnext})$;
28:     end if
29: end while
30: if $r_{last} == r_{A\_Trim2}$ then
31:     $\boldsymbol{R_A}$.Add ($r_{last}$), $\boldsymbol{R_B}$.Add ($r_{last}$), $\boldsymbol{R_B}$.Add ($r_{B\_Trim2}$);
32: else
33:     $\boldsymbol{R_A}$.Add ($r_{last}$), $\boldsymbol{R_A}$.Add ($r_{A\_Trim2}$), $\boldsymbol{R_B}$.Add ($r_{last}$);
34: end if
35: $\boldsymbol{R_A}$ = RemoveDuplicates($\boldsymbol{R_A}$), $\boldsymbol{R_B}$ = RemoveDuplicates($\boldsymbol{R_B}$);
36: return $\boldsymbol{R_A}$, $\boldsymbol{R_B}$;
```

Fig. 14. Long circular arc edge segmentation method.

the ray-circular arc intersection test for the last ray is specifically supplemented (see Lines 30–34). In addition, any duplicate intersection points in the two output lists need to be removed (see Line 34). Finally, by using Eq. (4), the intersection points in the two output lists are transformed from polar coordinates into Cartesian coordinates. These transformed intersection points constitute the vertices of two polylines, which are the required results for the segmentation of two long edges.

4.4.4. *Updating the footprint*

The vertices of two polylines generated from the arc edge segmentation are stored in the counter-clockwise direction as the segmentation is implemented along that direction (see the example in Fig. 16(a)). To construct a closed polyline to represent the new footprint, the two polylines are merged by inversely inserting the vertices of a polyline into the other (see Fig. 16(b)). Specifically, the first vertex of the merged polyline needs to be repeated as the last vertex to form a closed polyline (see the vertex 39 in Fig. 16(b)). Then the polyline is defined as the new footprint by an IfcPolyline instance, which further references a set of IfcCartesianPoint instances that define the coordinates of the vertices of the polyline. Updating the footprint of the curved wall is completed when relevant IFC instances that represent the original footprint are replaced with the new footprint representation (see Fig. 17). Moreover, according to the IFC specifications, the curved wall after faceting should be defined as an IfcWall instance as the resulting wall has a non-uniform thickness. To maintain the original reference relationships between the wall instance and other instances (e.g., IfcProductDefinitionShape and IfcRelContainedInSpatialStructure), only the entity name of the wall instance is changed to IfcWall if the original is IfcWallStandardCase (see Fig. 17).

The resulting wall meets all three output requirements. The footprint of the resulting wall is a closed polyline, which means that the solid geometry represented by IfcExtrudedAreaSolid or IfcBooleanClippingResult is a polyhedron with planar polygonal faces only. Thus the first out requirement is satisfied. Next, in the segmentation process, two arc edges are segmented simultaneously by the rays. The intersections between any two adjacent rays and the two edges

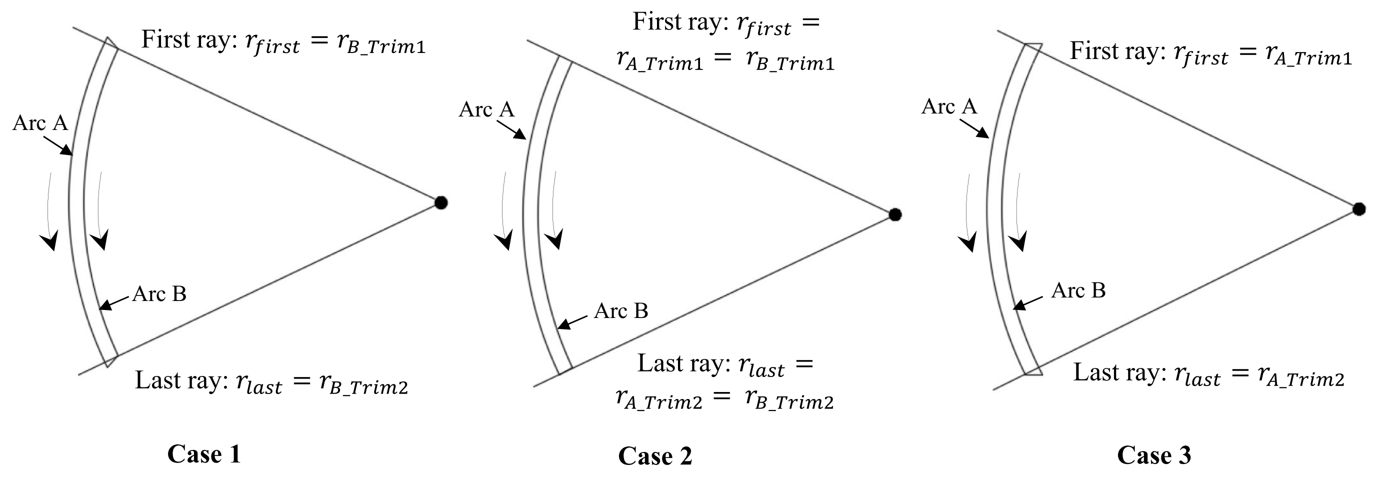

First ray: $r_{first} = r_{B\_Trim1}$
Arc A
Arc B
Last ray: $r_{last} = r_{B\_Trim2}$
Case 1
First ray: $r_{first} = r_{A\_Trim1} = r_{B\_Trim1}$
Arc A
Arc B
Last ray: $r_{last} = r_{A\_Trim2} = r_{B\_Trim2}$
Case 2
First ray: $r_{first} = r_{A\_Trim1}$
Arc A
Arc B
Last ray: $r_{last} = r_{A\_Trim2}$
Case 3

Fig. 15. The edge selections for the first and last ray constructions in three cases.

produce pairwise line segments (e.g., AB and CD in Fig. 16(a)). The pairwise segments are parallel as together with two rays (i.e., Ray 1 and Ray 2) they make two similar triangles (e.g., ΔABO and ΔCDO). This means that two planar surfaces corresponding to the two line segments are parallel to each other, and thus the second output requirement is satisfied. Regarding the third requirement, how much the original shape characteristic is maintained depends on how many line segments the arc edges are divided into. In the proposed faceting algorithm, this is mainly controlled by the input parameter "*SegmentDegree*": the smaller the value is, the closer the faceting result is to the original shape. The recommended value of the circular arc segmentation for each segment is 5 or 10 degrees [37]. This study takes 5 degrees for demonstration.

4.5. Updating geometries of relevant building objects

In this phase, straight walls with curved surfaces, openings on curved walls, spaces enclosed by curved walls, and curved slabs are updated to fit the faceted geometries of the curved walls. This updating, on the one hand, facets the curved surfaces of these building objects for building energy modeling, and on the other hand, eliminates incorrect geometric relationships between these building objects and faceted walls (see examples in Fig. 18). All the geometry updating operations are implemented on the footprints of relevant building objects given that their geometries are represented by IfcExtrudedAreaSolid or IfcBooleanClippingResult.

4.5.1. Updating geometries of straight walls with curved surfaces

Straight walls with curved surfaces can be identified by checking whether their standard footprints contain short arc edges. This type of walls mainly occurs when they are connected to curved walls. In this case, as discussed in Section 4.4.3, the short arc edges of straight walls would be identified as Type 1 connections of the curved walls. In the curved wall faceting process, all the Type 1 connections are not segmented but directly approximated as a line segment. Therefore, the new footprint of a straight wall (no matter connected or not connected to curved walls) can be straightforwardly created by replacing short arc edges with corresponding line segments and then merging all linear

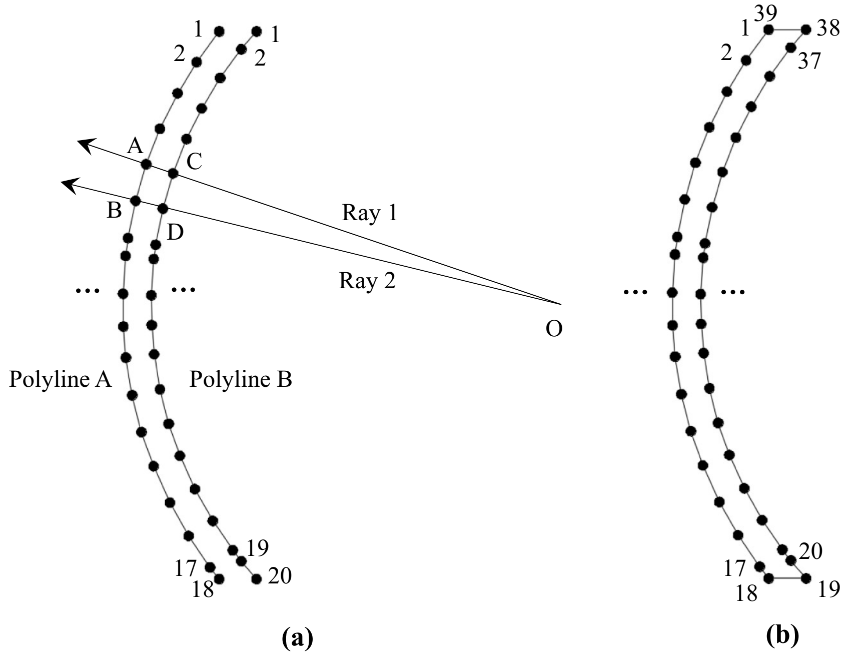
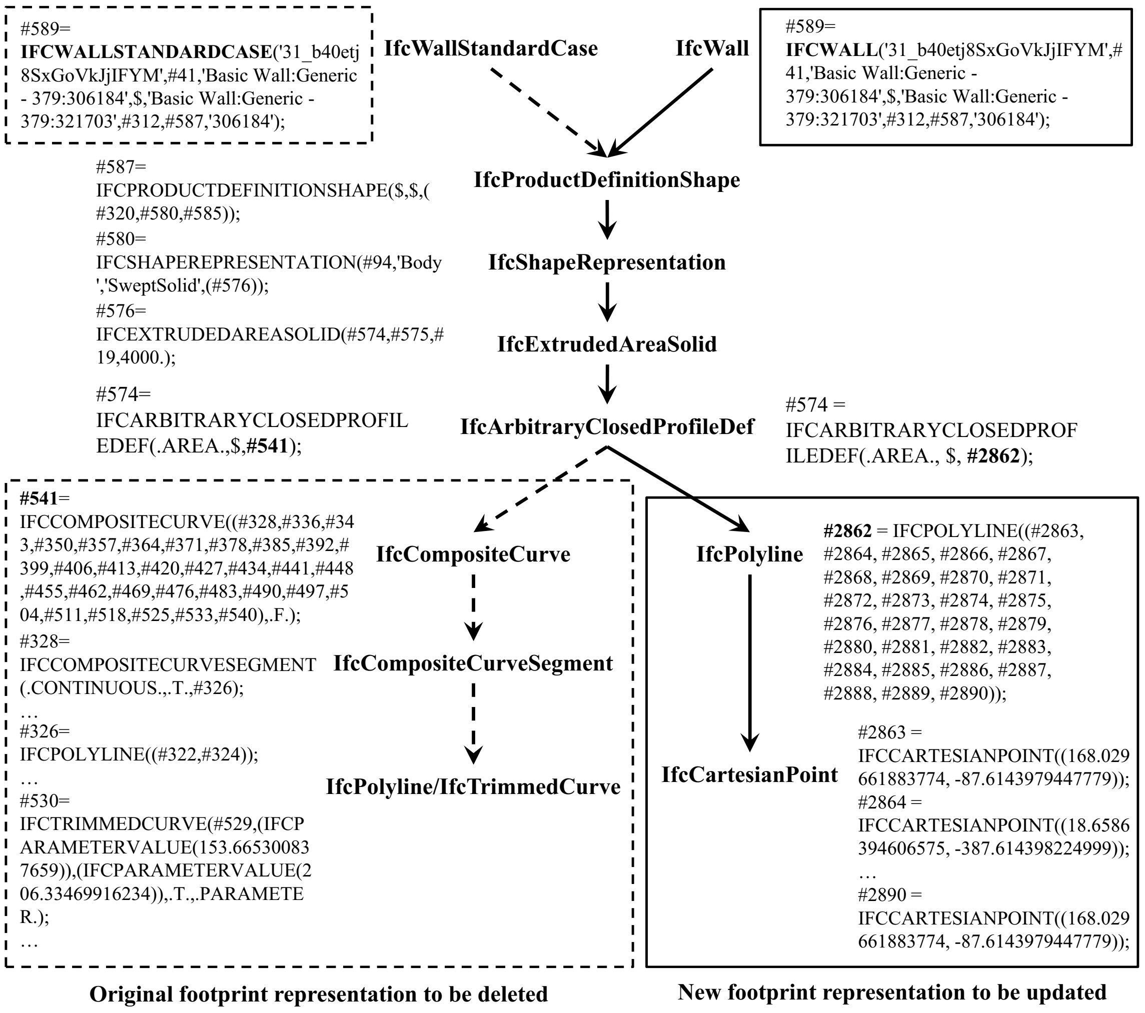


Fig. 16. Construction of closed polyline-based footprint: (a) two polylines resulting from the arc edge segmentation; (b) closed polyline-based footprint.


IfcWallStandardCase
IfcWall
IfcProductDefinitionShape
IfcShapeRepresentation
IfcExtrudedAreaSolid
IfcArbitraryClosedProfileDef
IfcCompositeCurve
IfcCompositeCurveSegment
IfcPolyline/IfcTrimmedCurve
IfcPolyline
IfcCartesianPoint
Original footprint representation to be deleted
New footprint representation to be updated

```
#589=
IFCWALLSTANDARDCASE('31_b40etj8SxGoVkJjIFYM',#41,'Basic Wall:Generic - 379:306184',$,'Basic Wall:Generic - 379:321703',#312,#587,'306184');
```

```
#589=
IFCWALL('31_b40etj8SxGoVkJjIFYM',#41,'Basic Wall:Generic - 379:306184',$,'Basic Wall:Generic - 379:321703',#312,#587,'306184');
```

```
#587=
IFCPRODUCTDEFINITIONSHAPE($,$,(#320,#580,#585));
#580=
IFCSHAPEREPRESENTATION(#94,'Body','SweptSolid',(#576));
#576=
IFCEXTRUDEDAREASOLID(#574,#575,#19,4000.);
#574=
IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,$,#541);
```

```
#574 =
IFCARBITRARYCLOSEDPROFILEDEF(.AREA., $, #2862);
```

```
#541=
IFCCOMPOSITECURVE((#328,#336,#343,#350,#357,#364,#371,#378,#385,#392,#399,#406,#413,#420,#427,#434,#441,#448,#455,#462,#469,#476,#483,#490,#497,#504,#511,#518,#525,#533,#540),.F.);
#328=
IFCCOMPOSITECURVESEGMENT(.CONTINUOUS.,.T.,#326);
…
#326=
IFCPOLYLINE((#322,#324));
…
#530=
IFCTRIMMEDCURVE(#529,(IFCPARAMETERVALUE(153.665300837659)),(IFCPARAMETERVALUE(206.33469916234)),.T.,.PARAMETER.);
…
```

```
#2862 = IFCPOLYLINE((#2863, #2864, #2865, #2866, #2867, #2868, #2869, #2870, #2871, #2872, #2873, #2874, #2875, #2876, #2877, #2878, #2879, #2880, #2881, #2882, #2883, #2884, #2885, #2886, #2887, #2888, #2889, #2890));
#2863 =
IFCCARTESIANPOINT((168.029661883774, -87.6143979447779));
#2864 =
IFCCARTESIANPOINT((18.6586394606575, -387.614398224999));
…
#2890 =
IFCCARTESIANPOINT((168.029661883774, -87.6143979447779));
```

Fig. 17. Updating the footprint representation of a curved wall in IFC STEP 21. Note: instances in dash box refer to the deleted original representation and instances in solid box refer to the updated representation.

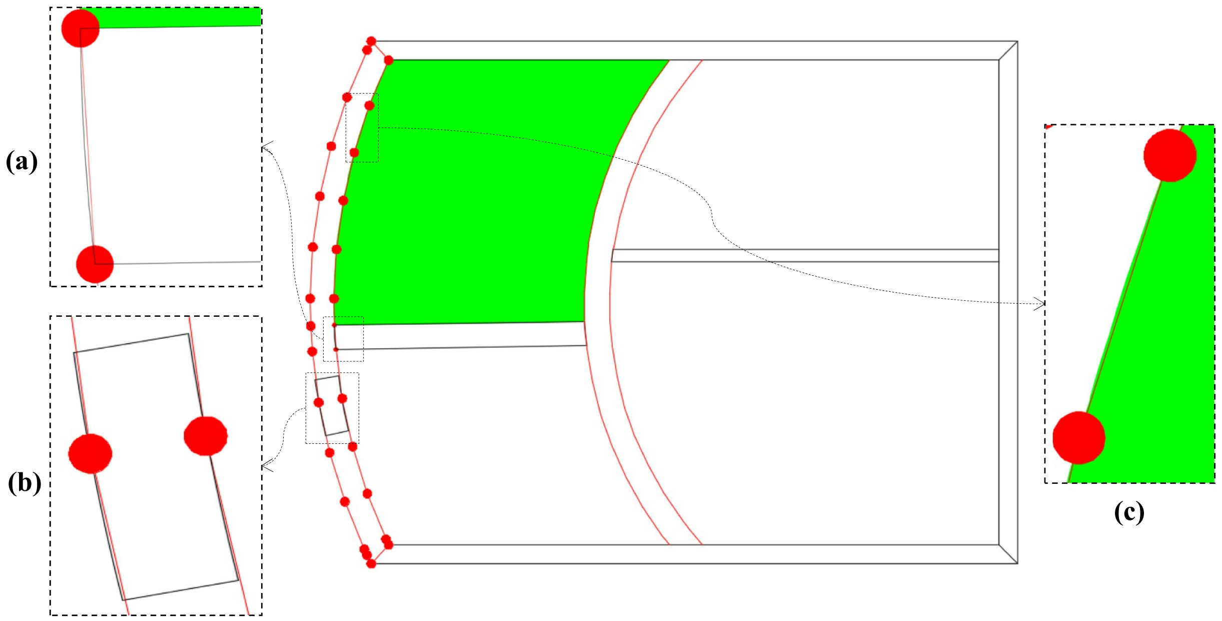


Fig. 18. Incorrect geometric relationships: (a) geometric clash between a straight wall and a faceted wall; (b) geometric inconsistency between a window and a faceted wall; and (c) geometric clash between a space and a faceted wall.

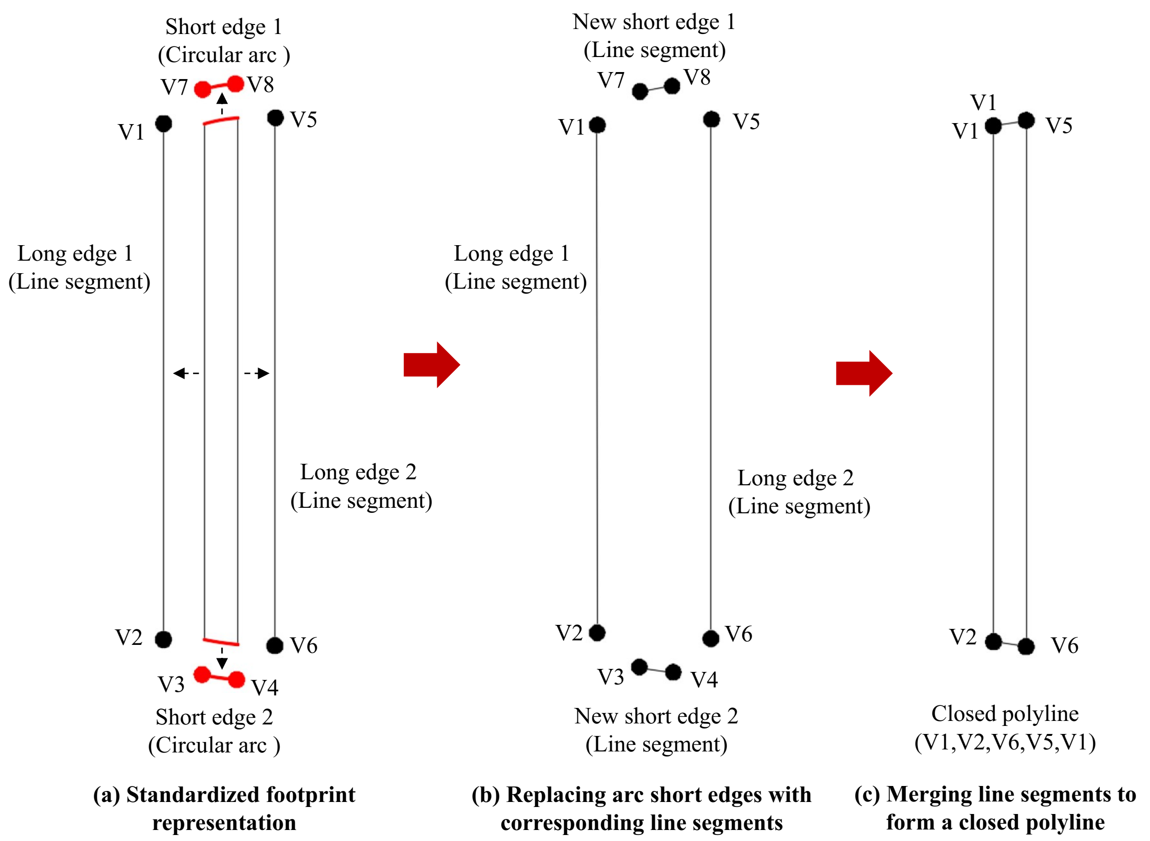


Fig. 19. Updating the footprint of a straight wall with curved surfaces.

edges to form a closed polyline-based footprint, as illustrated in Fig. 19. Finally, the new footprint is written into the IFC model to replace the original one. It is identical to the footprint updating of curved walls specified in Section 4.4.4.

4.5.2. *Updating geometries of openings on curved walls*

In IFC models, window and door geometries are often defined in great detail. For example, the geometry of a window may contain dozens of faces when each individual mullion is described. However, this level of detail is not necessary for a BEM, where openings only need to be represented as planar surfaces on the relevant wall surfaces [11,16]. In most existing IFC BIM - to - BEM transformation methods, not the geometries of doors and windows but the geometries of corresponding opening elements (i.e., IfcOpeningElement) are processed and transformed [10,11]. IfcOpeningElement defines a void in a wall that fills a door or a window and its geometry is much simpler. Therefore, this study updates the geometries of the IfcOpeningElement instances corresponding to doors and windows. The updating process is explained as follows:

(1) Identifying opening elements on curved walls

In CDM 2010, the relationship between a wall and an opening element (i.e., IfcOpeningElement), and the relationship between an opening element and the filled window (i.e., IfcWindow) or door (i.e., IfcDoor) are mandatorily defined by IfcRelVoidsElement and IfcRelFillsElement respectively. The opening elements filling windows and doors on curved walls can thus be identified based on these relationships.

(2) Extracting and standardizing footprints

The methods to extract and standardizing footprints of opening elements are identical to those of curved walls, which have been described in Sections 4.2 and 4.3.2 respectively. It is noteworthy that, for a cuboid opening element, the required footprint refers to either one of two faces that cross the wall thickness. Those two faces are distinguished from others by comparing the angle between the normal of each cuboid face and the normal of the footprint plane of the wall. In addition, as the LCS of an opening element is relative to the wall, the standard footprint of an opening element needs to be transformed in the LCS of the wall for geometry operations in the next step.

(3) Reconstructing footprints

This step is to process the standard footprint of an opening element to generate a new footprint which fully overlaps with the footprint of the relevant faceted wall. As an example, Fig. 20 illustrates the process of reconstructing the footprint of an opening element that fills a window.

First, the standard footprint is projected onto the footprint plane of the wall (see Fig. 20(a)). As the footprint has been transformed in the LCS of the wall, the projection direction refers to the wall's extrusion direction defined in IfcExtrudedAreaSolid or IfcBooleanClippingResult.

Second, four corner points of the new footprint are computed (see Fig. 20(b)). These corner points (i.e., CP1, CP2, CP3 and CP4 in the figure) refer to the intersection points between the two lines (i.e., Line 1 and Line 2) where the opening's two short edges are and the wall's polygonal footprint boundary which essentially consists of continuous line segments. The corner points are thus detected by performing a set of line - line segment intersection tests.

Third, a new footprint of the opening element consisting of two long and two short edges is reconstructed by the following procedure:

- Each long edge is constructed by tailoring a long edge of the wall with two corner points on that edge. In Fig. 20(c), two portions

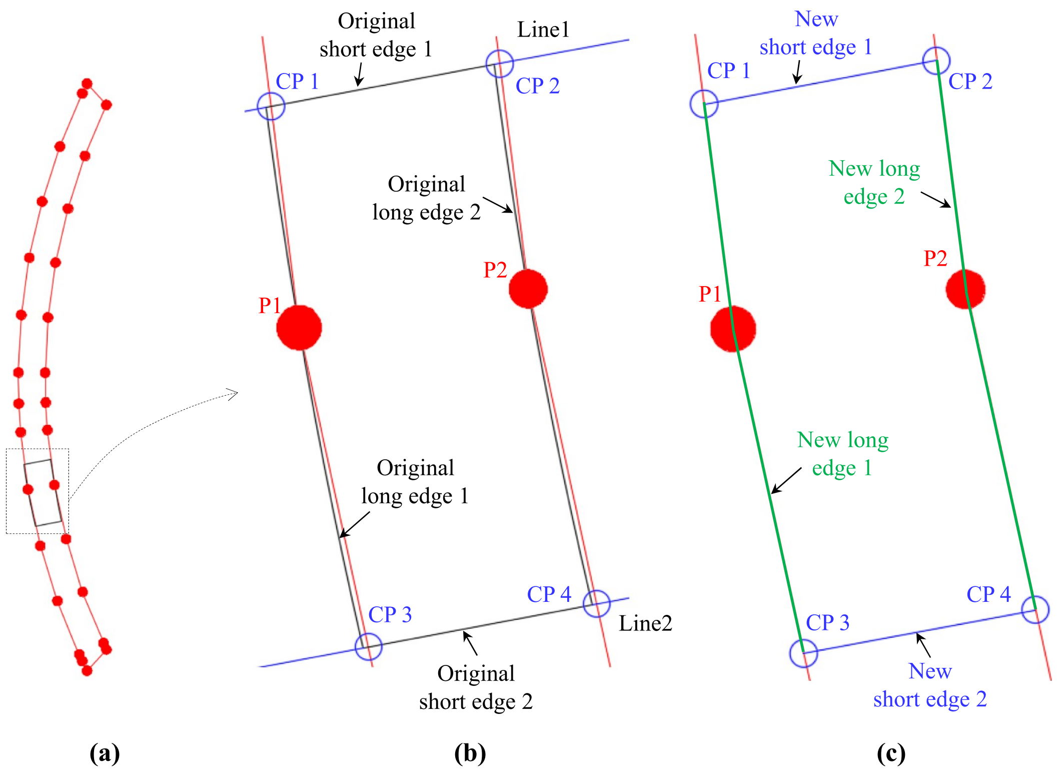


Fig. 20. Reconstruct the footprint of an opening element that fills a window: (a) footprint projection; (b) corner point detection; (c) new footprint construction.

representing two new long edges (i.e., polyline (CP1, P1, CP3) and polyline (CP2, P2, CP4)) are tailored from the two long edges of the wall by two pairs of corner points, ⟨CP1, CP3⟩ and ⟨CP2, CP4⟩.

- Each short edge is constructed by a pair of corner points that result from the intersection between the line where the original short edge is and the wall's footprint boundary. In Fig. 20(c), two new short edges (i.e. line segment (CP1, CP2) and line segment (CP3, CP4)) are produced by two pairs of corner points, ⟨CP1, CP2⟩ and ⟨CP3, CP4⟩.
- A closed polyline representing the new footprint on the wall footprint plane is obtained by merging the four reconstructed edges. Corresponding to the example in Fig. 20(c), the closed polyline (CP1, P1, CP3, CP2, P2, CP4, CP1) refers to the new footprint.
- The new footprint is projected back onto the original footprint plane of the opening element and transformed back into the LCS of the opening element.

(4) Updating footprints in the IFC model

The process of updating the footprint of an opening element is identical to that of a curved wall (see Section 4.4.4), including two steps: 1) wrap the new footprint by an IfcPolyline instance together with its referenced IfcCartesianPoint instances; and 2) replace the IFC instances that represent the original footprint with the new footprint representation.

4.5.3. Updating geometries of spaces enclosed by curved walls

The geometries of spaces in IFC models are updated by the following three steps:

(1) Extracting and standardizing the footprint of a space

The footprint extraction method for a curved wall (Section 4.2) is directly used to extract the footprint of a space from an IfcSpace instance in an IFC model. The extracted footprint need to be standardized for the following steps as arc edges of the footprint may be approximately represented as line segments or polylines. The corner vertex - based method presented in Section 4.3.2 is specifically for standardizing a footprint that has evident corner vertices. However, this is not always the case for space footprints. Common representative case is that two connected arc edges with close curvatures on the connected point are approximately represented as a polyline. It would be difficult for that method to identify the corner vertex (i.e., the connected point) from the vertices of that polyline.

To address this issue, a three-step method is developed for space footprint standardization. First, given a space footprint, a closed polyline-based temporary footprint is constructed by using the endpoints of line segments and/or circular arcs in the space footprint. Second, the temporary footprint is standardized by reconstructing straight and circular arc edges. Third, the final standard footprint is created by updating the standard temporary footprint with original circular arcs. The details of the first and the third steps are omitted here as they are identical to the first and the fourth steps of the wall footprint standardization method (see Section 4.3.2). The pseudo code of the second step is shown in Fig. 21.

The edges of the temporary footprint are mainly reconstructed via a while loop (see Lines 3–23 in the figure). In each iteration, the first three vertices in the temporary footprint are checked first whether they are on a common line. If so, other vertices of the temporary footprint also need to be checked one by one whether they are on that line (Lines 5–8). The checking terminates when a vertex not on that line is found. Then a straight edge is constructed with the first and the last vertices in the set of vertices on that line (Line 9). If the first three vertices in the temporary footprint are not on a common line, they are fitted with a circle (Line 11). Then two central angles corresponding to the three vertices are computed. If both central angles are smaller than a threshold (e.g., 10 degree), the three vertices belong to an arc edge. Checking whether other vertices are on the circle continues until an unqualified vertex is found (Lines 13–16). Then a circular arc edge is constructed by tailoring the circle with the first and the last vertices in the set of vertices on that circle (Line 17). If both central angles are

```
Input: Temporary footprint (a closed polyline) consisting of ordered vertices: $TF$;
Output: New temporary footprint (a mixture of line segments and circular arcs): $\boldsymbol{NTF}$.
1:  Initialize $\boldsymbol{NTF} = \{\}$;
2:  $l$ = NumerOfVertices($TF$);
3:  while  $l > 2$  do
4:      if IsOnALine($TF_0, TF_1, TF_2$) then
5:          $m = 2$;
6:          while $m < l$ && DotProduct$\left(\frac{\overrightarrow{TF_0TF_2}}{|\overrightarrow{TF_0TF_2}|}, \frac{\overrightarrow{TF_0TF_m}}{|\overrightarrow{TF_0TF_m}|}\right) \geq 0.9999$ do // Check whether $TF_m$ is
                                                                    // on the line ($TF_0, TF_2$)
7:              $m = m + 1$;
8:          end while
9:          $\boldsymbol{NTF}$.Add(CreateLineSegment($TF_0, TF_m$));    // Create a line segment edge
10:         $TF$.RemoveRange($\{TF_i | i \geq 0$ and $i < m\}$);   // Update TF by removing checked vertices
11:     else if IsOnACircle($TF_0, TF_1, TF_2$) then
12:         $circle$ = CreateCircle($TF_0, TF_1, TF_2$);
13:         $n = 2$;
14:         while $n < l$ && |Distance($TF_n, circle.centre$) $- circle.radius| \leq$ 1mm do
15:             $n = n + 1$;
16:         end while
17:         $\boldsymbol{NTF}$.Add(CreateCircularArc($circle, TF_0, TF_n$)); // Create a circular arc edge
18:         $TF$.RemoveRange($\{TF_i | i \geq 0$ and $i < n\}$);
19:     else
20:         $\boldsymbol{NTF}$.Add(CreateLineSegment($TF_0, TF_1$));
21:         $TF$.Remove($TF_0$);
22:     end if
23: end while
24: if  $l == 2$  then
25:     $\boldsymbol{NTF}$.Add(CreateLineSegment($TF_0, TF_1$));
26: end if
27: Return $\boldsymbol{NTF}$;
```

Fig. 21. Temporary footprint standardization for spaces.

larger than the threshold, the first two vertices belong to a straight edge (Line 20). In each iteration, the temporary footprint is updated by removing the vertices that have been used for edge reconstruction (Lines 10, 18, 21). The while loop ends when no > 2 vertices remain in the temporary footprint. If there are still two vertices left, they are directly used to define a straight edge (Lines 24–26).

(2) Identifying spaces enclosed by curved walls

After all footprints are standardized, spaces with arc edges are identified. Only their geometries need to be updated.

(3) Reconstructing footprints of identified spaces

The footprint of each identified space is reconstructed by replacing all the arc edges with corresponding sections in the faceted footprints of the curved walls that enclose the space. Fig. 22 shows the pseudo code of the proposed space footprint reconstruction method and Fig. 23 illustrates the main steps with an example.

The proposed method takes the followings as inputs: the standard footprint of a space, both standard and faceted footprints of curved walls, and transformation matrices from LCSs of curved walls to the LCS of space. The calculation of a transformation matrix from a curved wall to a space is similar to the transformation matrix computation between two walls, which has explained in Section 4.4.1. As a pre-processing step, the standard and faceted footprints of curved walls are transformed from their own LCS into the coordinates of the space (see Lines 2 and 3 in Fig. 22).

Second, for each arc edge of the space, all the curved walls connected to the space on that edge are identified and the connections are computed by a circular arc - circular arc overlap test (see details in Section 4.4.1) (Line 8). Specifically, this test is implemented on the arc edge of the space and two long edges of the standard footprint of each curved wall. As the arc edge of the space may connect different curved walls, all curved walls need to be checked to compute the connections until the sum of the connections covers the entire arc edge. In the example of Fig. 23(a), Space A has one arc edge V2V3 and connects only with Curved Wall A. Thus, the connection arc of the arc edge V2V3 is itself.

Third, each arc edge of the space is updated with a polyline. This polyline is produced by merging a set of polylines in the faceted footprints of curved walls, which correspond to the connection arcs obtained in previous step (Lines 9–15). In Fig. 23(b), the connection arc V2V3 corresponds to the polyline (P1, P2, P3, P4, P5, P6, P7, P8) in the faceted footprint of Curved Wall A. Hence, this polyline is used to replace the arc edge V2V3 of Space A.

After all the arc edges of the space are processed, a closed polyline-based footprint for the space is obtained by merging its original straight edges and the new polyline-based edges (Line 18). Corresponding to the example in Fig. 23, the new footprint for Space A is constructed as (V1, P1, P2, P3, P4, P5, P6, P7, P8, V4, V1).

(4) Updating footprints of identified spaces in the IFC model

Updating space geometries is completed by writing all the new reconstructed footprints into the IFC model to replace their original ones. The process is the same as the process of updating the footprint of a curved wall (Section 4.4.4).

4.5.4. Updating geometries of slabs with curved surfaces

Unlike updating geometries of straight walls, openings and spaces, updating slab geometries does not need to consider geometric relationships with curved walls. This is because slabs generally are not connected with curved surfaces of walls. This means that the connections of curved walls with slabs are not affected by the curved wall faceting processes. Updating the geometry of a slab is thus simply to

```
Input: Space standard footprint consisting of circular arcs and line segments: $S\_SF$;
       Standard footprints of all curved walls: $CW\_SFs$;
       Faceted footprints of all curved walls: $CW\_FFs$;
       Transformation matrices from all curved walls' LCSs to the space's LCS: $Ts$.
       // Note: The corresponding relationships between $CW\_SFs$, $CW\_FFs$ and $Ts$ are preserved
Output: Updated space footprint (a closed polyline): $\boldsymbol{USF}$.
1:  Initialize $\boldsymbol{USF} = \{\}$ and $UpdatedEdges = \{\}$;
2:  $CW\_SFs$ = Transform($CW\_SFs, Ts$);
3:  $CW\_FFs$ = Transform($CW\_FFs, Ts$);
4:  foreach curve $C$ in $S\_SF$ do
5:      if $C$ is a line segment then
6:          $UpdatedEdges$.Add($C$);
7:      else if $C$ is a circular arc then
8:          List<circular arc> $Connections$ = ComputeOverlaps($C, CW\_SFs$); // Find curved
                // walls that bound the space on arc C and compute the overlap arc sections
9:          Initialize $Ps = \{\}$;
10:         foreach circular arc $CA$ in $Connections$ do
11:             $P$ = FindPolyline($CA, CW\_FFs$); // Find the polyline section corresponding
                                                // to the overlap arc CA in CW_FFs
12:             $Ps$.Add($P$);
13:         end foreach
14:         $edge$ = Merge($Ps$); // Merge all found polyline sections into a single polyline
15:         $UpdatedEdges$.Add($edge$);
16:     end if
17: end foreach
18: $\boldsymbol{USF}$ = Merge($UpdatedEdges$); // Merge all updated edges to form a closed polyline
19: Return $\boldsymbol{USF}$;
```

Fig. 22. Space footprint reconstruction method.

facet all curved surfaces (i.e., approximate arc edges of a footprint into line segments), which consists of the following three steps:

(1) Identifying slabs that need to be updated

Footprints of slabs are extracted from IfcSlab instances by using the footprint extraction method for curved walls (see Section 4.2). Slabs with a footprint containing circular arc edges are recognized for downstream processing.

(2) Constructing new footprints for the identified slabs

The footprint of each identified slab is constructed by segmenting all the circular arc edges into sets of line segments and then merging them with other linear edges to form a closed polyline. The computational method is shown in Fig. 24.

The new footprint is progressively constructed by processing the continuous curve elements of an extracted footprint one by one. For a linear curve element (i.e., IfcPolyline), its vertices are inserted into the vertex set of the new footprint. The order of these vertices should be consistent with current vertex set and there should be no duplicate vertices (see Lines 3–14 in the figure). A circular curve element (i.e., IfcTrimmedCurve) is segmented as a polyline within the following four steps (see Lines 15–27). First, the circular arc is redefined as counter-clockwise if not and further transformed into a polar coordinate system by the method explained in Section 4.4.3 to ease the segmentation (Lines 16–17). Second, the ray-based segmentation method explained in Section 4.4.3 is fine-tuned to segment the circular arc into a polyline (Lines 17–24). Subsequently, the vertices of the generated polyline are transformed from the built polar coordinates back into the original

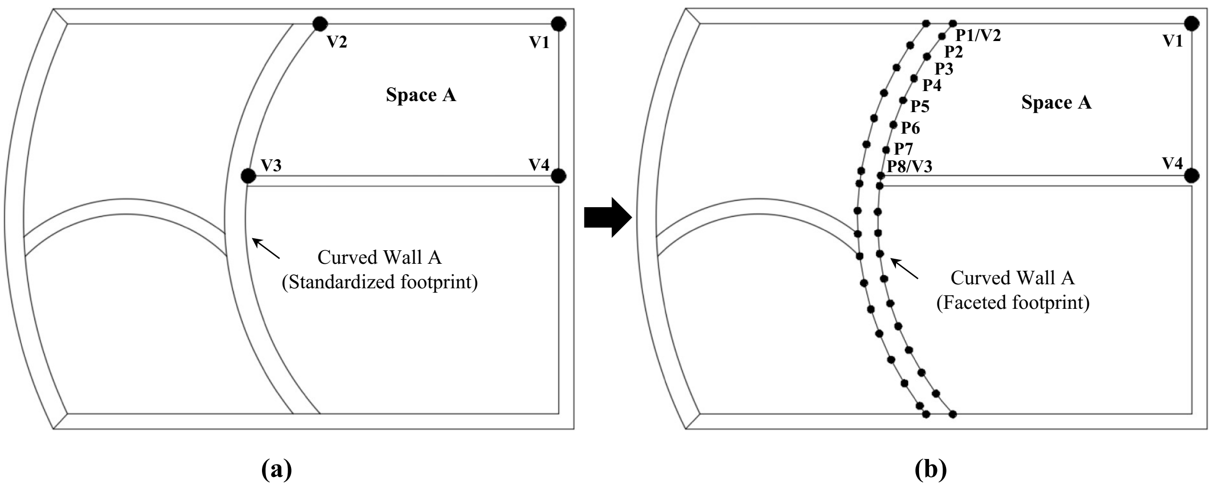


Fig. 23. Footprint reconstruction of a typical space: (a) identifying curved walls that enclose the space; (b) updating arc edges of the space footprint with corresponding faceted boundaries of curved walls.

```
Input: Original slab footprint consisting of continuous curves: $OF$;
       Segmentation degree for circular arc edges: $SD$.
Output: Segmented slab footprint (a closed polyline): $\boldsymbol{SF}$.
1:  Initialize $\boldsymbol{SF} = \{\}$;
2:  foreach curve $C$ in $OF$ do
3:      if $C$ is a polyline (IfcPolyline) then
4:          if Distance(FirstVertex($C$), LastVertex($\boldsymbol{SF}$)) $\leq$ 0.01mm then // Check whether
                                            // $FirstVertex(C)$ and $LastVertex(\boldsymbol{SF})$ are a same point
5:              $C$ = RemoveFirstVertex($C$);
6:              foreach vertex $V$ in $C$ do
7:                  $\boldsymbol{SF}$.Add($V$);
8:              end foreach
9:          else if Distance(LastVertex($C$), LastVertex($\boldsymbol{SF}$)) $\leq$ 0.01mm then
10:             $C$ = ReverseVertexOrder($C$);
11:             Repeat the same processes from Line 5 to Line 8;
12:         else
13:             Repeat the same processes from Line 6 to Line 8;
14:         end if
15:     else if $C$ is a circular arc (IfcTrimmedCurve) then
16:         $C$ = SetArcDirectionAsAnticlockwise($C$); // Redefine $C$ as anti-clockwise if not
17:         $C\_Polar$ = TransformToPolarCoordinates($C$);
18:         Initialize $Points\_Polar = \{\}$;        // Polar angles of segmented points
19:         $\theta = C\_Polar.\theta_{Trim1}$;       // Polar angle of the first trimming point
20:         while $\theta < C\_Polar.\theta_{Trim2}$ do
21:             $Points\_Polar$.Add($\theta$);
22:             $\theta = \theta + SD$;
23:         end while
24:         $Points\_Polar$.Add($C\_Polar.\theta_{Trim2}$);
25:         $Points\_Cartesian$ = TransformToCartesianCoordinates($Points\_Polar$);
26:         Repeat the same processes from Line 4 to Line 14 for $Points\_Cartesian$;
27:     end if
28: end foreach
29: $\boldsymbol{SF}$.AddVertex($\boldsymbol{SF}[0]$);
30: Return $\boldsymbol{SF}$;
```

Fig. 24. New footprint construction for slabs.

Cartesian coordinates using the Eq. (4) in Section 4.4.3 (Line 25). Finally, these vertices are inserted into the vertex set of the new footprint by the linear curve element processing method shown in Lines 4–14. Once all the curve elements are processed, the new footprint construction is completed by repeating the first vertex as the end to form a closed polyline (Line 29).

(3) Updating footprints in the IFC model

The geometry of each slab is updated by writing the new footprint representation into the IFC model to replace the original one. The process is same as the process of updating the footprint of a curved wall (see Section 4.4.4).

4.6. Removing original SBs

After the geometries of curved walls and relevant building objects have been faceted and updated, the original SBs in the input IFC model need to be removed. In IFC, SB is defined as an objectified relationship by a set of linked entities, as shown in Fig. 25. To cleanly delete SBs without destroying syntactic integrity of the IFC model, all the entity instances involved in defining SBs but not referenced by other irrelevant instances are deleted. In this regard, IfcRelSpaceBoundary instances need to be deleted as they only relate to SB definition. Entity instances including IfcConnectionSurfaceGeometry, IfcCurveBoundedPlane, IfcPlane, IfcPolyline, IfcAxis2Placement3D, IfcDirection and IfcCartesianPoint can be deleted only if they do not have reference relationships with other instances that are out of the scope of SB definition. IfcOwnerHistorey instance is reserved as it is referenced by other instances such as IfcBuilding and IfcBuildingStorey. Instances representing spaces (i.e., IfcSpace) and building elements (i.e., IfcBuildingElement) must be kept as they define important information irrelevant to SBs.

5. Algorithm implementation and validation

5.1. Algorithm implementation

To prove the feasibility of the proposed algorithm, a prototype application is developed in C#. The IFC Engine DLL [38] is used to process IFC instance files, including reading and extracting required information from input IFC files as well as writing new IFC files with the processed results. Fig. 26 shows a screenshot of the graphical user interface of the prototype application.

5.2. Algorithm validation

To verify the performance of the algorithm, a simple building model and a real-world building model are used. Both models were created in Revit 2017 and exported as an IFC instance file in the mode of CDM 2010.

The simple building model (see Fig. 27(a)) is specifically designed by including curved walls with all footprint representation forms. This model also demonstrates all basic types of geometrical relationships between curved walls, straight walls, openings, spaces and slabs: curved walls have connections with other walls (curved walls and straight walls) on both curved faces (Type 1 connections) and end faces (Type 2 connections); curved walls host an opening (a door or a window); curved walls enclose spaces; and curved walls are placed on slabs with curved surfaces. Table 1 gives a detailed explanation of the features of

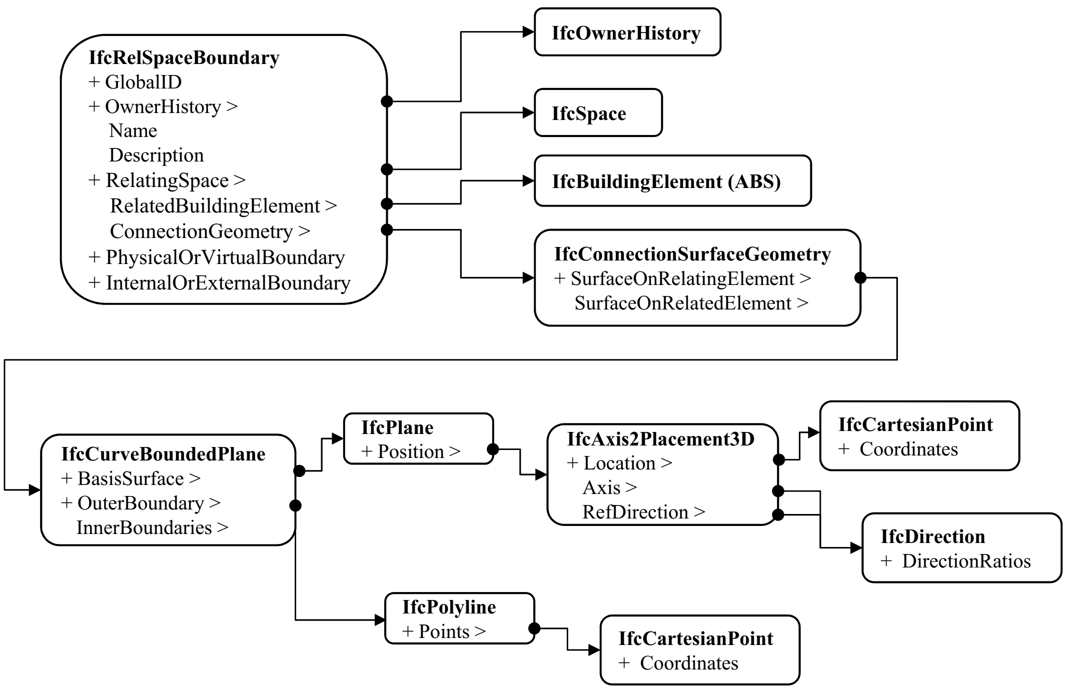
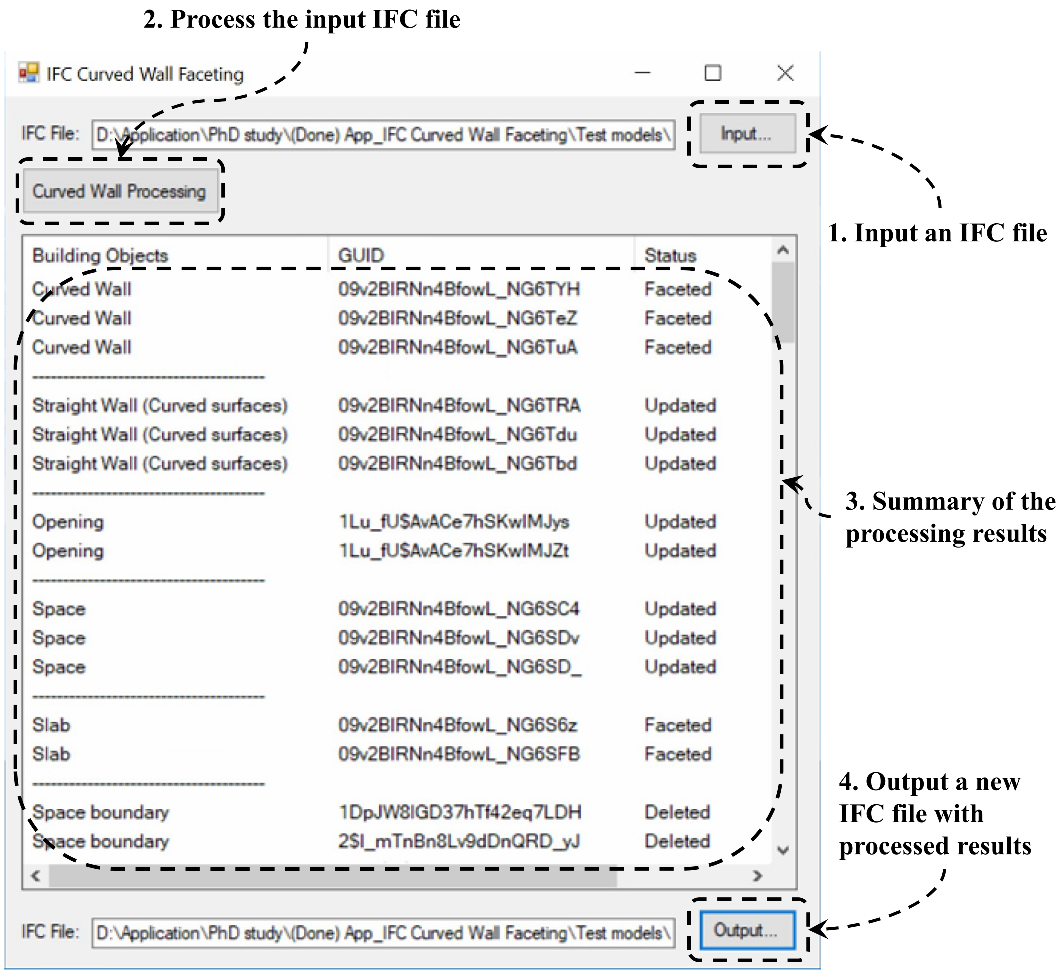


Fig. 25. Instantiation diagram to define SBs in CDM 2010.


2. Process the input IFC file
IFC Curved Wall Faceting
IFC File: D:\Application\PhD study\(Done) App_IFC Curved Wall Faceting\Test models\
Input...
Curved Wall Processing
1. Input an IFC file
3. Summary of the processing results
4. Output a new IFC file with processed results
Output...

Building Objects	GUID	Status
Curved Wall	09v2BIRNn4BfowL_NG6TYH	Faceted
Curved Wall	09v2BIRNn4BfowL_NG6TeZ	Faceted
Curved Wall	09v2BIRNn4BfowL_NG6TuA	Faceted

Straight Wall (Curved surfaces)	09v2BIRNn4BfowL_NG6TRA	Updated
Straight Wall (Curved surfaces)	09v2BIRNn4BfowL_NG6Tdu	Updated
Straight Wall (Curved surfaces)	09v2BIRNn4BfowL_NG6Tbd	Updated
--------------------------------		
Opening	1Lu_fU$AvACe7hSKwIMJys	Updated
Opening	1Lu_fU$AvACe7hSKwIMJZt	Updated

Space	09v2BIRNn4BfowL_NG6SC4	Updated
Space	09v2BIRNn4BfowL_NG6SDv	Updated
Space	09v2BIRNn4BfowL_NG6SD_	Updated

Slab	09v2BIRNn4BfowL_NG6S6z	Faceted
Slab	09v2BIRNn4BfowL_NG6SFB	Faceted

Space boundary	1DpJW8IGD37hTf42eq7LDH	Deleted
Space boundary	2$I_mTnBn8Lv9dDnQRD_yJ	Deleted

Fig. 26. Graphical user interface of the prototype application.

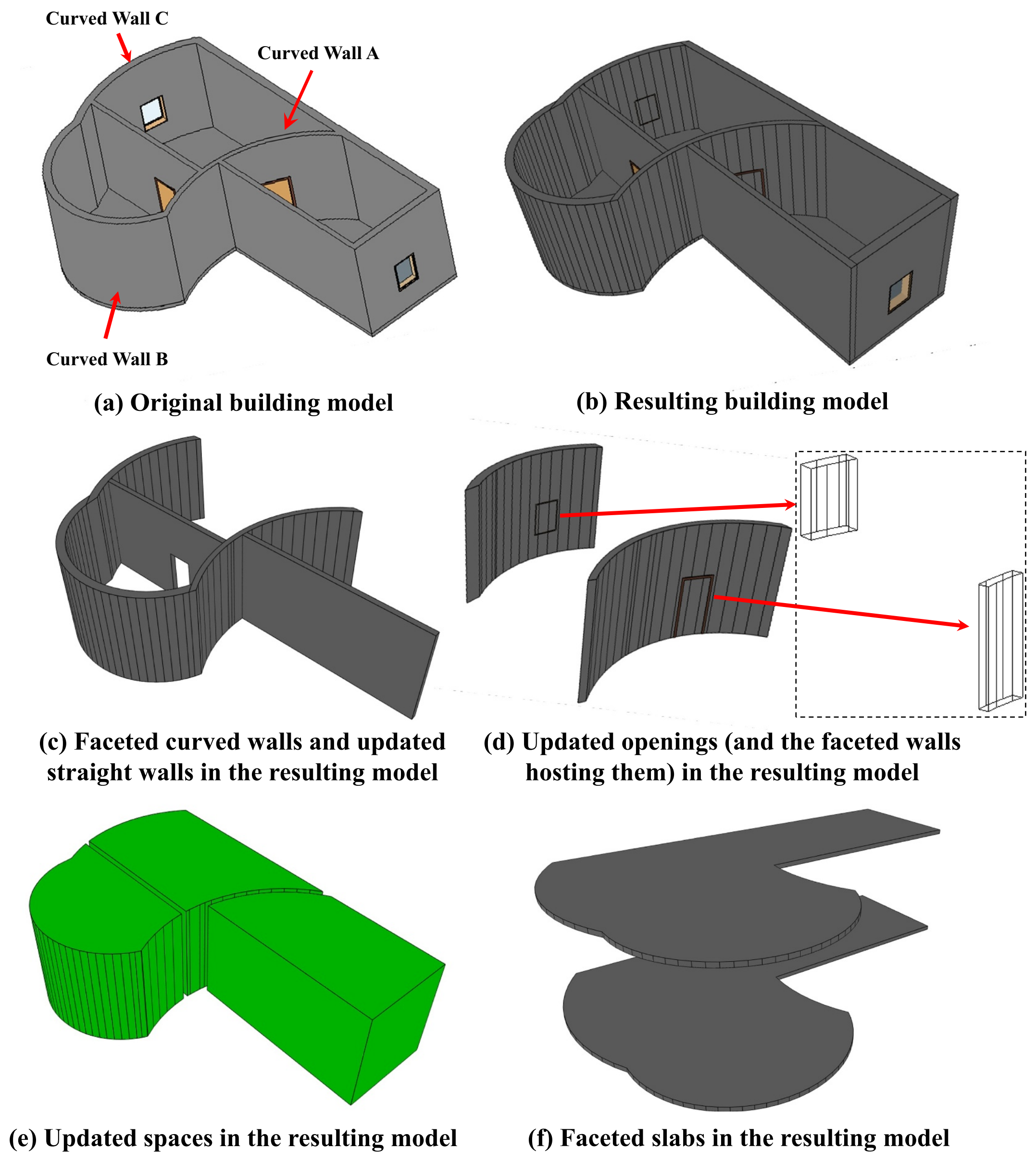

Curved Wall C
Curved Wall A
Curved Wall B
(a) Original building model
(b) Resulting building model
(c) Faceted curved walls and updated straight walls in the resulting model
(d) Updated openings (and the faceted walls hosting them) in the resulting model
(e) Updated spaces in the resulting model
(f) Faceted slabs in the resulting model

Fig. 27. Simple building model for the experiment and its faceting results. Note: The ceiling slab and spaces are hidden in (a) and (b) for the visualization of building interiors.

Table 1
Features of the simple building model.

Curved walls	Footprint representation form	Type 1 connection (Y/N)	Type 2 connection (Y/N)	Hosting openings (Y/N)	Placed on curved slabs (Y/N)
Curved wall A	Forms 1&2	Y	Y	Y (door)	Y
Curved wall B	Form 3	N	Y	N	Y
Curved wall C	Form 4: LE1(CA); LE2(P); SE1(LS); SE2(LS)	Y	Y	Y (window)	Y

Note: LE = Long edge; SE = Short edge; CA = Circular arc; P = Polyline; LS = Line segment.

this model.

Fig. 27(b)–(f) shows the experiment results of the simple building model. These figures clearly demonstrate that the proposed algorithm performs as intended: (1) all curved walls are correctly identified and faceted with correct geometric connections with their connected walls; (2) straight walls with curved surfaces (caused by the connections with curved walls) are fully updated; (3) openings on curved walls and spaces enclosed by curved walls are accordingly updated to be consistent with corresponding faceted walls; and (4) slabs with curved surfaces are faceted.

The real-world model representing one story of a complex university building (see Fig. 28(a)) is further used to test the reliability and feasibility of the algorithm. This model has a total of 99 walls, 29 openings, and 47 spaces (the complex curved corridor space was divided into 4 smaller spaces using “Room Separator” in Revit). Among them, there are 12 curved walls, 13 straight walls with curved faces, 9 openings on curved walls, 18 spaces partially enclosed by curved walls, and 2 curved slabs. Table 2 gives a summary of the features of this model. Compared to the simple building model, this model covers all types of curved walls in terms of the footprint representations in Form

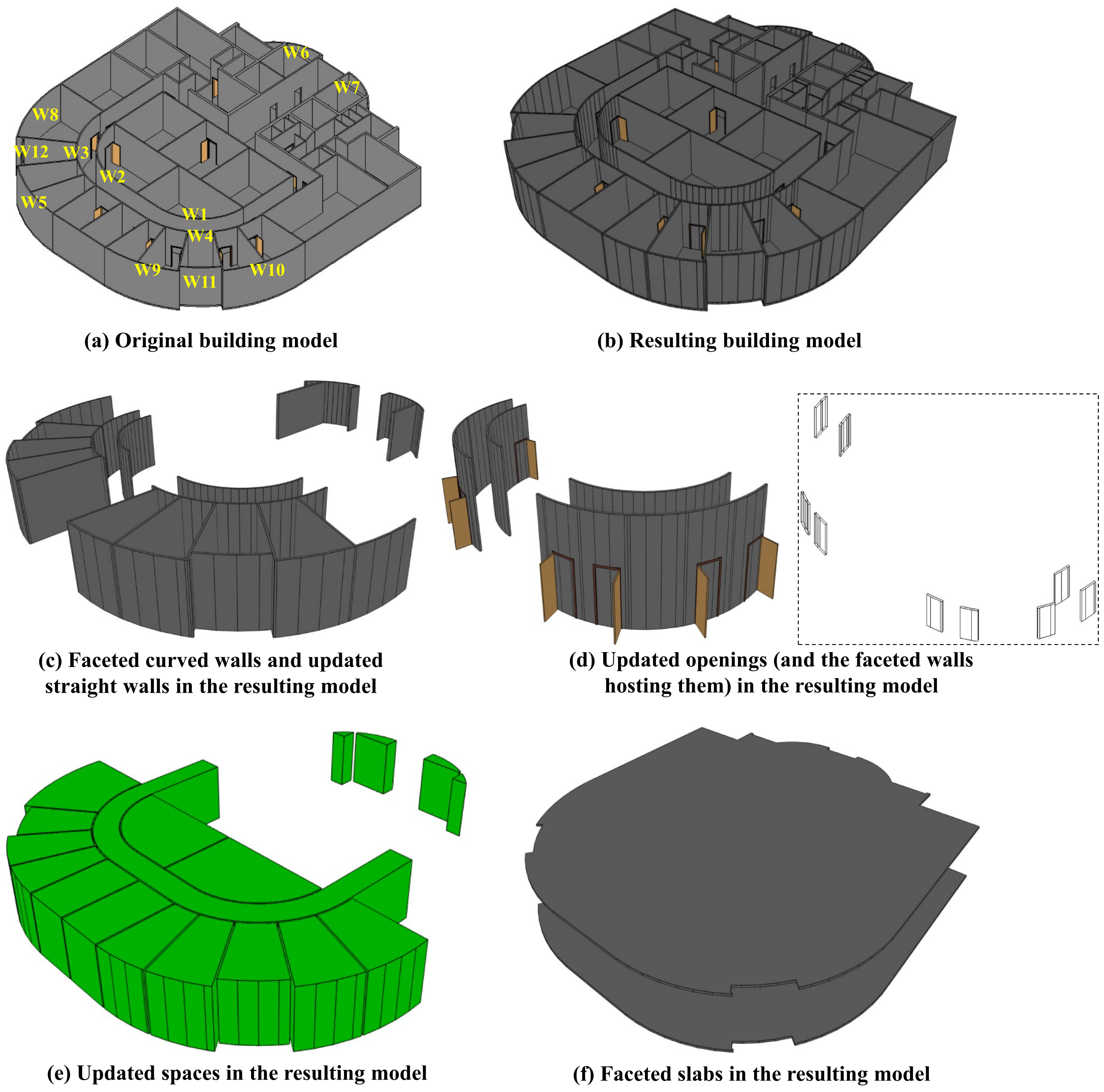

W6
W7
W8
W12
W3
W2
W5
W1
W4
W9
W11
W10
(a) Original building model
(b) Resulting building model
(c) Faceted curved walls and updated straight walls in the resulting model
(d) Updated openings (and the faceted walls hosting them) in the resulting model
(e) Updated spaces in the resulting model
(f) Faceted slabs in the resulting model

Fig. 28. Real-world building model for the experiment and its faceting results. Note: The ceiling slab and spaces are hidden in (a) and (b) for the visualization of building interiors.

4. Furthermore, more geometrical relationships between various building objects are included.

Fig. 28(b) depicts the output IFC model produced by the prototype application. Fig. 28(c)–(f) demonstrates the processing results of curved walls and their connected straight walls, all the openings on curved walls, all the spaces enclosed by curved walls, and all curved slabs respectively. These results show that all relevant building objects in original building model are fully and correctly processed.

6. Limitations and future work

Four main limitations of the proposed algorithm are acknowledged.

First, the proposed algorithm is restricted to curved walls and relevant building objects (i.e., straight walls, openings, spaces and slabs) whose shapes can be represented by IfcExtrudedAreaSolid or IfcBooleanClippingResult. It facets curved surfaces of these objects by approximating or updating corresponding curved edges in their 2D planar footprints with polylines. In other words, the algorithm is 2.5D based and mainly works on planar footprints rather than curved surfaces directly. Therefore, it cannot address curved building objects (e.g., walls with toroidal surfaces, spherical surfaces or B-spline surfaces) that cannot be defined as a footprint-based representation. Curved walls in this case need to be handled on the 3D surface level, and so do the geometric updating of relevant building objects. Currently, many curved surface meshing algorithms (e.g., Delaunay triangulation and Advancing Front Method) have been developed [39]. However, those algorithms may not be able to produce results complying with the four output requirements (see Section 3.2) for the purpose of building energy modeling. Further research on developing 3D surface - based curved wall faceting algorithm for building energy modeling is thus needed.

Second, the proposed algorithm can update the geometries of

Table 2
Features of the university building model.

Curved walls	Footprint representation form	Number of Type 1 connections	Number of hosting openings
W1	Form 4–1	0	1
W2	Form 4–1	0	1
W3	Form 4–2	4	3
W4	Form 4–2	5	4
W5	Form 4–3	2	0
W6	Form 4–3	1	0
W7	Form 4–3	1	0
W8	Form 4–4	0	0
W9	Form 4–3	3	0
W10	Form 4–3	2	0
W11	Form 4–2	0	0
W12	Form 4–2	0	0

Note: (1): LE = Long edge; SE = Short edge; LS = Line segment; CA = Circular arc; P = Polyline.
(2): Form 4–1 = (LE1(CA); LE2(CA); SE1(CA); SE2(P)).
(3): Form 4–2 = (LE1(CA); LE2(CA); SE1(LS); SE2(LS)).
(4): Form 4–3 = (LE1(CA); LE2(P); SE1(LS); SE2(LS)).
(5): Form 4–4 = (LE1(CA); LE2(CA); SE1(LS); SE2(P)).

openings whose footprints run through the thickness of curved walls only (the footprint of an opening can be parallel or non-parallel to the footprint of the curved wall). However, some shapes of openings such as circular and arched windows, the footprints of which are usually along the side surfaces of walls, are also common in practice. The improvement of the algorithm to handle these openings on curved walls is thus also suggested as future work.

Third, the proposed algorithm can process curved walls and relevant building objects with footprints bounded by a closed composite curve that consists of circular arcs and/or line segments only. It cannot directly process footprints with other parametric curves (e.g., ellipse arc and B-spline) and nonparametric arbitrary curves. Segmentation methods to process those parametric curves while faceting curved walls need to be studied in future. A promising approach for arbitrary curve-based footprints is to add a pre-processing step to the algorithm that approximates footprint boundaries with circular arcs and line segments. Several arbitrary planar curve approximating approaches such as genetic algorithms [40], adaptive smoothing [41], and dynamic programing [42] found in pattern recognition and image processing field are considerable for further investigation.

Fourth, at this point, the proposed algorithm processes IFC models complying with CDM 2010. Although this MVD supports the definition of curved walls and relevant building objects targeted in this study, it only provides three types of curve elements (i.e., line segment, polyline and circular arc) to define footprint boundaries. Other curve elements such as ellipse arc and B-spline are not supported, which limits the ability of this MVD to represent relevant curved shapes. Furthermore, this MVD is based on the IFC2x3 TC1 schema, which cannot explicitly define some common curved surfaces such as toroidal surfaces, spherical surfaces and B-spline surfaces. However, IFC4 Add2 not only allows to define those curved surfaces (by IfcToroidalSurface, IfcSphericalSurface and IfcBSplineSurface, respectively), but also provides powerful curved solid representation approaches including IfcTriangulatedFaceSet and IfcAdvancedBrep. In future, the IFC model parsing module of the algorithm needs to be improved to be able to process IFC4 based models.

7. Conclusions

Existing IFC BIM-to-BEM transformation approaches are not fully capable of handling building objects with curved surfaces in IFC models. To address this limitation, this study extends existing transformation approaches by suggesting a pre-processing step to detect and facet curved geometries in IFC models. The pre-processed IFC models with polyhedral geometries only thus can be processed by the existing transformation approaches to create geometries of a building energy model.

Specifically, this study introduces an algorithm to facet curved walls in IFC models into polyhedron-based geometries. The main idea of the algorithm to facet curved walls is to update their footprints by segmenting all arc edges into sets of line segments. By embedding the footprint standardization method, the algorithm is able to process walls with various footprint representation forms that mainly result from linear approximations of arc footprint boundaries. The proposed ray-based footprint segmentation method enables the algorithm to produce faceted walls that satisfy the three output requirements, namely: (1) the result geometries shall be polyhedrons; (2) the result geometries shall be able to be further processed to generate thermal SBs; and (3) geometry-related thermal features of original curved shapes shall be maintained as much as possible. Another important feature of the segmentation method is that the connections between a curved wall and its adjacent walls are also considered. This ensures the correctness of these connections in the faceting results.

In addition to faceting curved walls, this algorithm also provides mechanisms to update the geometries of relevant building objects, including straight walls connected to curved walls, openings on curved walls, and spaces enclosed by curved walls, to fit with the faceted geometries of relevant curved walls. This eliminates curved surfaces of these objects and ensures correct geometric relationships between them and corresponding faceted walls. Furthermore, this algorithm detects and facets slabs with curved surfaces.

A prototype application implementing the proposed algorithm was developed and applied to a simple building model and a complex real-world building model. The evaluation results demonstrate that the algorithm performs as intended. This algorithm is thus expected to expand the efforts on the automatic information exchange from IFC BIM to BEM by equipping the curved geometry processing ability.

Acknowledgments

The work described in this paper was supported by a grant from the Seed Funding Programme for Basic Research of the University of Hong Kong, Hong Kong (Project No. 201502159002). The authors would also like to greatly thank the anonymous reviewers for their invaluable comments on improving this manuscript.

References

[1] IEA, Transition to Sustainable Buildings: Strategies and Opportunities to 2050. Available from: https://www.iea.org/publications/freepublications/publication/Building2013_free.pdf (Last accessed: September 14, 2017).
[2] UNEP, Why Buildings. Available from: http://staging.unep.org/sbci/AboutSBCI/Background.asp (Last accessed: September 14, 2017).
[3] EMSD, Hong Kong Energy End-use Data, Available from: http://www.emsd.gov.hk/filemanager/en/content_762/HKEEUD2016.pdf, (2016) , Accessed date: 11 November 2016.
[4] J. O'Donnell, T. Maile, C. Rose, N. Mrazović, E. Morrissey, C. Regnier, et al., Transforming BIM to BEM: Generation of building geometry for the NASA Ames Sustainability Base BIM, Lawrence Berkeley National Laboratory, Berkeley, CA, 2013. Available from: https://escholarship.org/uc/item/1x09b1xd (Last accessed: June 6, 2018).
[5] U.S. GSA, GSA BIM Guide 05 - Energy Performance. Available from: http://www.gsa.gov/portal/content/102283 (Last accessed: November 12, 2016).
[6] V. Bazjanac, Acquisition of Building Geometry in the Simulation of Energy Performance, Lawrence Berkeley National Laboratory, Berkeley, CA, 2001 Available from: https://escholarship.org/uc/item/2k58j3k8 , Accessed date: 6 August 2018.
[7] V. Bazjanac, IFC BIM-based Methodology for Semi-automated Building Energy Performance Simulation, Lawrence Berkeley National Laboratory, Berkeley, CA, 2008 Available from https://escholarship.org/uc/item/0m8238pj , Accessed date: 6 August 2018.
[8] NBIMS-US™, Frequently asked questions about the national BIM standard-United States: What is a BIM? Available from: https://www.nationalbimstandard.org/faqs#faq1 (Last accessed: July 5, 2016).
[9] D. Ladenhauf, K. Battistia, R. Berndtd, E. Eggeling, D.W. Fellner, M. Gratzl-Michlmair, et al., Computational geometry in the context of building information

modeling, Energy and Buildings 115 (2016) 78–84, https://doi.org/10.1016/j.enbuild.2015.02.056.

[10] G.N. Lilis, G.I. Giannakis, D.V. Rovas, Automatic generation of second-level space boundary topology from IFC geometry inputs, Autom. Constr. 76 (2017) 108–124, https://doi.org/10.1016/j.autcon.2016.08.044.

[11] C.M. Rose, V. Bazjanac, An algorithm to generate space boundaries for building energy simulation, Eng. Comput. 31 (2015) 271–280, https://doi.org/10.1007/s00366-013-0347-5.

[12] A. Farzaneh, J. Carriere, D. Forgues, D. Monfet, Framework for using building information modeling to create a building energy model, J. Archit. Eng. 24 (2) (2018) 05018001, , https://doi.org/10.1061/(ASCE)AE.1943-5568.0000306.

[13] R.J. Hitchcock, J. Wong, Transforming IFC Architectural View BIMs for Energy Simulation: 2011, 12th Conference of International Building Performance Simulation Association, Sydney, Australia, 2011, pp. 1089–1095. Available from: http://ibpsa.org/proceedings/BS2011/P_1391.pdf , Accessed date: 25 January 2019.

[14] G. Gourlis, I. Kovacic, Building information modelling for analysis of energy efficient industrial buildings - a case study, Renew. Sust. Energ. Rev. 68 (2017) 953–963, https://doi.org/10.1016/j.rser.2016.02.009.

[15] buildingSMART, Technical Vision. Available from: https://www.buildingsmart.org/standards/technical-vision/ (Last accessed: November 12, 2016).

[16] V. Bazjanac, Space boundary requirements for modeling of building geometry for energy and other performance simulation, 27th CIB W78 International Conference, Cairo, Egypt, 2010, pp. 16–18. Available from: http://www.e3lab.org/upl/website/publication11111/spaceboundary.pdf (Last accessed: January 25, 2019).

[17] Digital Alchemy, Simergy™. Available from: https://d-alchemy.com/products/simergy (Last accessed: August 1, 2018).

[18] A. Andriamamonjy, D. Saelens, R. Klein, An automated IFC-based workflow for building energy performance simulation with Modelica, Autom. Constr. 91 (2018) 166–181, https://doi.org/10.1016/j.autcon.2018.03.019.

[19] H. Kim, K. Anderson, Energy modeling system using building information modeling open standards, J. Comput. Civ. Eng. 27 (3) (2012) 203–211, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000215.

[20] K.U. Ahn, Y.J. Kim, C.S. Park, I. Kim, K. Lee, BIM interface for full vs. semi-automated building energy simulation, Energy and Buildings 68 (2014) 671–678, https://doi.org/10.1016/j.enbuild.2013.08.063.

[21] J. Choi, J. Shin, M. Kim, I. Kim, Development of openBIM-based energy analysis software to improve the interoperability of energy performance assessment, Autom. Constr. 72 (2016) 52–64, https://doi.org/10.1016/j.autcon.2016.07.004.

[22] T. El-Diraby, T. Krijnen, M. Papagelis, BIM-based collaborative design and socio-technical analytics of green buildings, Autom. Constr. 82 (2017) 59–74, https://doi.org/10.1016/j.autcon.2017.06.004.

[23] N. Yu, Y. Jiang, L. Luo, S. Lee, A. Jallow, D. Wu, et al., Integrating BIMserver and OpenStudio for energy efficient building, 2013 ASCE International Workshop on Computing in Civil Engineering, Los Angeles, California, USA, 2013, pp. 516–523, doi:https://doi.org/10.1061/9780784413029.065.

[24] S. Pinheiro, R. Wimmer, J. O'Donnell, S. Muhic, V. Bazjanac, T. Maile, et al., MVD based information exchange between BIM and building energy performance simulation, Autom. Constr. 90 (2018) 91–103, doi:https://doi.org/10.1016/j.autcon.2018.02.009.

[25] Autodesk, Export a Project to IFC. Available from: https://knowledge.autodesk.com/support/revit-products/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Revit-DocumentPresent/files/GUID-14037C31-EBAD-41A8-9099-E6DD65BB626E-htm.html (Last accessed: November 13, 2016).

[26] Graphisoft, Help Center: Working With IFC. Available from: http://helpcenter.graphisoft.com/guides/archicad-18/archicad-18-int-reference-guide/interoperability/file-handling-and-exchange/working-with-ifc/model-view-definitions/ (Last accessed: October 10, 2016).

[27] H. Ying, S. Lee, A framework for rule-based validation of IFC space boundaries for building energy analysis, 2017 ASCE International Workshop on Computing in Civil Engineering, Seattle, WA, USA, 2017, pp. 110–117, doi:https://doi.org/10.1061/9780784480823.014.

[28] T. Maile, J. O'Donnell, V. Bazjanac, C. Rose, BIM-geometry Modelling Guidelines for Building Energy Performance Simulation, 13th Conference of International Building Performance Simulation Association, Chambéry, France, 2013, pp. 3242–3249. Available from: http://www.ibpsa.org/proceedings/BS2013/p_1510.pdf , Accessed date: 25 January 2019.

[29] SBT-IFC SPACE BOUNDARY TOOL. Available from: https://gaia.lbl.gov/interoperability/SBT/ (Last accessed: April 4, 2017).

[30] OpenStudio, Import IFC. Available from: http://nrel.github.io/OpenStudio-user-documentation/tutorials/tutorial_ifcimport/(Last accessed: August 1, 2018).

[31] C. van Treeck, E. Rank, Dimensional reduction of 3D building models using graph theory and its application in building energy simulation, Eng. Comput. 23 (2) (2007) 109–122, https://doi.org/10.1007/s00366-006-0053-7.

[32] K. Kim, J. Yu, A process to divide curved walls in IFC-BIM into segmented straight walls for building energy analysis, J. Civ. Eng. Manag. 22 (3) (2016) 333–345, https://doi.org/10.3846/13923730.2014.897975.

[33] buildingSMART, IfcWall. Available from: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcsharedbldgelements/lexical/ifcwall.htm (Last accessed: August 1, 2018).

[34] buildingSMART, IfcWallStandardCase. Available from: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcsharedbldgelements/lexical/ifcwallstandardcase.htm (Last accessed: August 1, 2018).

[35] buildingSMART, Start Page of IFC2x3 Final Documentation: IfcShapeRepresentation. Available from: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ (Last accessed: November 12, 2016).

[36] Blis-Project, Concept Design BIM 2010 – IFC 2X3 Binding Concepts Model. Available from: http://www.blis-project.org/IAI-MVD/reporting/browseMVD.php?MVD = GSA-005&BND = IFC2x3&LAYOUT = H (Last accessed: December 1, 2017).

[37] M. Weise, T. Liebich, R. See, V. Bazjanac, T. Laine, Implementation Guide: Space Boundaries for Energy Analysis. Available from: http://www.buildingsmart-tech.org/downloads/accompanying-documents/agreements/ (Last accessed: October 10, 2016).

[38] RDF, IFC Engine DLL. Available from: http://rdf.bg/ifc-engine-dll.php (Last accessed: December 1, 2017).

[39] P.J. Frey, P-L George, Mesh Generation: Application to Finite Elements, ISTE, London, UK, 2010, doi:https://doi.org/10.1002/9780470611166. ISBN 9781848210295.

[40] B. Sarkar, L.K. Singh, D. Sarkar, Approximation of digital curves with line segments and circular arcs using genetic algorithms, Pattern Recogn. Lett. 24 (15) (2003) 2585–2595, https://doi.org/10.1016/S0167-8655(03)00103-X.

[41] J.H. Horng, An adaptive smoothing approach for fitting digital planar curves with line segments and circular arcs, Pattern Recogn. Lett. 24 (1–3) (2003) 565–577, https://doi.org/10.1016/S0167-8655(02)00277-5.

[42] F. Tortorella, R. Patraccone, M. Molinara, A Dynamic Programming Approach for Segmenting Digital Planar Curves Into Line Segments and Circular Arcs, 19th International Conference on Pattern Recognition, Florida, USA, 2008, pp. 1–4, https://doi.org/10.1109/ICPR.2008.4761177.