Automation in Construction 131 (2021) 103897

Contents lists available at ScienceDirect

Automation in Construction

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

Heterogeneous network modeling and segmentation of building information modeling data for parallel triangulation and visualization

Xiaoping Zhou [a,c], Mengmeng Wang [a], Yu-Shen Liu [b,*], Qian Wang [c,*], Maozu Guo [a,*], Jichao Zhao [d]

[a] Beijing Key Laboratory of Intelligent Processing for Building Big Data, Beijing University of Civil Engineering and Architecture, Beijing 100044, China
[b] School of Software, Tsinghua University, Beijing, China
[c] Department of Building, National University of Singapore, Singapore
[d] CNPC Managers Training Institute, Beijing 100096, China

ARTICLE INFO

Keywords:
Data segmentation
Building information modeling (BIM)
Industry foundation classes (IFC)
Parallel computing
Triangulation
Visualization

ABSTRACT

Building Information Modeling (BIM) is a promising technology for the construction industry, and BIM models have become widely accepted as 3D construction models. Currently, BIM data are often organized by product level to support cross-platform parallel and real-time visualization. However, cross-platform visualization of BIM products with large volumes of geometric data still poses a challenge to both triangulation and rendering processes. Existing efforts have mainly used geometric simplification and geometric data streaming technologies. This study addresses this issue from a different perspective by partitioning original large-scale BIM products into small BIM sub-products. First, a novel heterogeneous geometric relationship model (HeGeo) is proposed to categorize BIM relationships into reference, decomposition, and association relationships according to Industry Foundation Classes. On top of the HeGeo, a strategy for partitioning the geometric data of the original BIM products is proposed. Empirical studies were conducted on nine BIM models with a BIM product embedding all the geometric data. The experimental results showed that the proposed scheme improved triangulation efficiency 3.05 ± 0.57 times with the same hardware resources and the same triangulation tool through a parallel computing framework and raised the triangular mesh loading efficiency 1.53 ± 0.29 times by processing requests concurrently. In addition, the proposed scheme improved the user experience of online BIM visualization tools through incremental rendering. The segmentation scheme was generalized to product-level and floor-level schemes by constraining the segmentation nodes to BIM products and floors respectively. Thus, the segmentation scheme could be applied to any BIM model and facilitate BIM adoption by all stakeholders during a building's life cycle.

1. Introduction

Building Information Modeling (BIM) is a process that involves the generation and management of digital representations of both physical and functional characteristics of facilities [1]. By providing a shared knowledge resource for information about a facility, BIM enables data interoperability and collaboration among stakeholders during a building's lifecycle [2] and has been widely adopted in the construction industry.

As a modern equivalent of visual communication among stakeholders during the lifecycle of a construction project, visualizing BIM data is an essential requirement for an ever-increasing number of interactive applications to support various kinds of decision-making [3]. Cross-platform real-time visualization [4], which supports rendering data on various kinds of devices, is a natural step towards an even larger number of applications and users.

Currently, some studies have explored cross-platform visualization schemes for BIM data using the Web Graphic Library (WebGL) [4–7]. These studies had similar processes. The Industry Foundation Classes (IFC) is an open, international standard (ISO 16739-1:2018) and is supported by most BIM tools. The BIM model is first exported to IFC format. Then the BIM geometric data are converted into triangular mesh

* Corresponding authors.
E-mail addresses: lukefchou@gmail.com, zhouxiaoping@bucea.edu.cn (X. Zhou), liuyushen@tsinghua.edu.cn (Y.-S. Liu), bdgwang@nus.edg.sg (Q. Wang), guomaozu@bucea.edu.cn (M. Guo).

https://doi.org/10.1016/j.autcon.2021.103897
Received 24 June 2020; Received in revised form 16 July 2021; Accepted 14 August 2021

data and stored in the cloud. The triangular meshes are usually computed and organized at the product level to enable parallel real-time rendering [8]. In this manner, a client can request triangular meshes of multiple products on the cloud through network protocols like HTTP, thus enabling parallel rendering of BIM data. However, BIM models often contain products with extremely large and/or complex 3D shapes. One example is the site of a BIM model, which is defined by *IfcSite* in IFC format. Another example is the IFC files exported from tools like SketchUp and Rhino, where all the 3D shapes are aggregated into one or a few *IfcBuildingElementProxy* objects. These large-scale BIM products pose challenges in both triangulation and rendering for current cross-platform visualization schemes, which trigger a new demand for sub-product-level segmentation of BIM files.

1.1. Triangulation

Large and complex BIM products often require massive computation resources and much time to triangulate [8]. On one hand, industries need to equip themselves with high-performance computers to enable triangulation of any BIM file in the era of construction Big Data [9]. This would significantly raise the financial barriers for BIM applications. On the other hand, users must wait a long time to visualize BIM files online.

1.2. Rendering

Large and complex products usually contain massive amounts of triangular mesh data. As a result, clients must wait much longer to render these products because a product can be rendered only when all its geometric data have been obtained [4]. This may result in an unfriendly user experience.

Recent studies have explored floor-level [10] and product-level [8,11] parallel computation of BIM data to improve triangulation efficiency. However, these systems cannot be used to address the cross-platform visualization of large and complex BIM products. Two technologies are available to mitigate the rendering of large-scale complex BIM products. The first uses geometric simplification technologies [12–14] to translate geometric BIM data into a lower level of detail (LoD) [15]. However, geometric simplification may also affect the resolution of the geometries [16]. The second is data streaming [17–19]. To adopt data streaming technology, both the data format and the visualization tools must be carefully designed and upgraded on top of streaming data standards.

This study addresses this issue through a novel perspective of segmenting large complex BIM products from original BIM files. After the authors noticed that the shape of any complex BIM product is composed of several shapes, the original geometric data were split into several small geometric data slices, and each small data segment was treated as an independent new sub-product. This approach can use a parallel computing framework to accelerate BIM triangulation, enable the sub-product-level organization of triangulated mesh data, and render large-scale complex BIM products more efficiently. Unlike streaming technologies, this study mitigates the parallel real-time rendering problem without changing the data format or rendering tools. In addition, the proposed scheme is a generalized version of existing product-level [8,11] and floor-level BIM data segmentation procedures [10] and can be applied to any BIM file.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 provides the necessary definitions and preliminaries. Section 4 presents the heterogeneous geometric relationship model of a BIM, and Section 5 develops the geometric data segmentation algorithm. Section 6 describes the empirical studies, and the last section presents conclusions.

2. Related work

To enable real-time rendering of large-scale geometric data, current efforts have mainly focused on mesh simplification, mesh streaming, and data decomposition. Additionally, compression technologies [20,21] may also be applied to these solutions to shorten mesh data transmission time from the geometric data server to the visualization tools. In this section, these compression schemes are not discussed.

2.1. Mesh simplification

Mesh simplification simplifies the original mesh data with constraints and has been studied extensively in many areas. Huettenberger et al. [22] simplified multivariate data using Pareto sets. Yi et al. [23] developed a novel differential-evolution-based method to compute a low-cost path that leads to a high-quality Delaunay mesh. Choi et al. [24] proposed an octree-based mesh simplification approach to simplify the presentation of 3D objects for augmented-reality head-mounted displays.

Mesh simplification has also been used in computer-aided design (CAD). Kwon et al. [25] investigated a scheme to minimize changes in outer shape and reduce data size in 3D CAD models when applying LoD. Fan et al. [12] presented an approach for deriving LoD2 buildings from LoD3 models in the City Geography Markup Language format. Forberg [13] used scale spaces to generalize 3D building data. Kim et al. [14] studied the customized simplification problem, considering LoDs in an indoor space.

Mesh simplification can reduce geometric data volume and thus shorten mesh data transmission time from the data server to visualization tools, as well as rendering time. However, the resolution of the geometries may be reduced [16]. Moreover, the original meanings of attributes in a BIM may not be retained [26].

2.2. Mesh streaming

Mesh streaming is another technology for efficiently rendering large geometric data sets. As more geometric data are streamed from the data server to visualization tools, the resolution of the 3D objects increases gradually.

Doumanoglou et al. [27] provided a systematic understanding of the requirements of live 3D mesh coding, targeting (tele-) immersive media streaming applications. Englert et al. [17] proposed an optimized streaming approach for large Web 3D applications. Zampoglou et al. [18] developed an adaptive streaming algorithm for complex Web 3D scenes based on the MPEG-DASH standard. Noguera et al. [28] surveyed applications of mesh streaming in mobile volume rendering.

Some studies have also investigated the possibility of applying mesh streaming to BIM Big Data [9] rendering. Limper et al. [19] proposed a streamable format for generalized Web-based 3D data transmission, called the Shape Resource Container (SRC), to facilitate the volumetric rendering of geometric data. However, SRC is not supported in mainstream 3D engines, meaning that a specific rendering engine must be designed.

2.3. Data decomposition

Data decomposition, also called data segmentation, splits the original data into many data segmentations. Visualization tools can correctly render a data segmentation once obtained. In this manner, data decomposition mitigates the real-time rendering of large-scale geometric data using parallel rendering. Gao et al. [29] introduced a volumetric partitioning strategy based on a generalized sweeping framework to seamlessly partition the volume of an input triangular mesh into a collection of deformed cuboids. As Gao's method only supports triangular mesh partitioning, it cannot be directly applied to BIM geometric data. Katz et al. [30] proposed a novel hierarchical mesh decomposition algorithm that computationally decomposes a given mesh into its meaningful components. Theologou et al. [31] presented a fully automatic mesh segmentation scheme using heterogeneous graphs.

These schemes, when applied to BIM data, may affect the latent links in a BIM product. Inspired by the visualization of 3D GIS data, Xu et al. [34] used 3D tiles to visualize large volumes of BIM geometric data. Because data segmentation schemes for 3D GIS data, such as technologies generating 3D tiles, are operated over triangular meshes, they cannot be used to split the BIM model and speed up BIM triangulation.

Currently, some studies have explored data segmentation of BIM models to accelerate the computational efficiency of BIM data. Early in 2014, Jiao et al. [10] split BIM data by floors and developed a floor-level MapReduce framework for BIM. Recently, Zhou et al. [11] proposed a product-level parallel computing scheme by decomposing a BIM file into BIM products. Both floor-level and product-level parallel computing schemes were equipped with a BIM triangulation tool, e.g., *IfcOpenShell* [32], which was then used to triangulate floor-level and product-level BIM data slices in parallel. Accordingly, a BIM visualization tool can render BIM geometric data in parallel using floor-level and product-level triangular meshes. For example, Zhou et al. [8] explored the use of product-level BIM data segmentation to enable product-level BIM parallel triangulation and to facilitate the product-level parallel rendering of BIM geometric data in WebBIM[1] [4]. The product-level and floor-level segmentation schemes modeled the relationships among IFC instances as homogeneous. Take IFC instance "#72=*IfcStyledItem*(#71, (#73, #74), $);" as an example. In the product-level BIM data segmentation scheme [11], IFC instance #72 has the same relationship, namely a reference relationship, with IFC instances #71, #73, and #74. As a result, IFC #72 links directly to #71, #73, and #74. This homogeneous modeling approach works when segmenting a BIM model at the product or floor level because the reference relationship is enough to aggregate all the IFC instances relating to a floor or a product. However, homogeneous modeling is inadequate to enable sub-product-level segmentation of a BIM model. A concrete example is that the homogeneous modeling cannot segment "#72 = *IfcStyledItem*(#71, (#73, #74), $);" into two IFC instances "#721=*IfcStyledItem*(#71, (#74), $);" and "#722=*IfcStyledItem*(#71, (#74), $);". Subsequently, extra effort is required to upgrade the homogeneous modeling of a BIM model to enable the segmentation of BIM files with large complex products. To the best of the authors' knowledge, this study is the first to explore a data decomposition solution for any BIM geometric data.

3. Preliminaries and definitions

The IFC specification is the internationally accepted specification for BIM. Without loss of generality, it is assumed here that the BIM data are expressed by an IFC file. This section presents a brief introduction to the use of IFC involving geometric data expression for BIM products and defines six related terminologies and the data segmentation problem for BIM products. Table 1 lists the notations used in this study.

3.1. Geometric data expression in IFC

A BIM product is an object that relates to a geometric or spatial context. In the IFC specification, a BIM product is defined by the *IfcProduct* class. An *IfcProduct* object will occur at a specific location in space if it is assigned a geometric representation. The *ObjectPlacement* attribute establishes the coordinate system in which all points and directions used by the geometric representation items under *Representation* are found. The *Representation* is provided by an *IfcProductDefinitionShape*, which can be either a geometric or topological representation.

Usually, the 3D geometric data for BIM products are represented by at least one of the following models: boundary representation (Brep), non-uniform rational B-splines (NURBS), constructive solid geometry (CSG), or swept solid model (SweptSolid) [16]. These technologies provide powerful abilities to describe all kinds of shapes in a compressed format. However, the use of hybrid 3D geometric representations also brings disadvantages. One of the crucial problems is that these 3D representation models cannot be directly rendered in many scenarios. For example, due to the limitations of OpenGL ES and WebGL, Web systems and mobile phone applications cannot support the rendering of complex solid geometric descriptions. Hence, the geometric data must be triangulated to support cross-platform visualization. Large and complex BIM products often yield a large volume of triangular mesh data, which poses new challenges for the efficient rendering of BIM products. This study aims to address this issue by proposing a BIM geometric data segmentation algorithm. Unlike traditional solutions that segment the triangular mesh data, the algorithm proposed here splits the original BIM data, which can speed up both triangulation and rendering in a parallel manner.

Table 1
Notations.

Symbol	Description
i	A BIM instance.
p	A BIM product.
g_p	Geometric data of p.
a_p	Semantic attribute data of p.
s_i	Geometric data size of i.
m	A BIM model.
n	Number of instances in BIM model m.
M	Heterogeneous geometric relationship model.
C_i	Child instances of BIM instance i.
D_i	Descendant instances of BIM instance i.
λ	Threshold for geometric data size.
$Q(p)$	Collection of sub-products partitioned from p.
S	Dictionary structure storing geometric data sizes of BIM instances.
F	Dictionary structure storing segmentation flags of BIM instances.
κ	Number of processes in parallel computing

3.2. Terminology definition

3.2.1. Definition 1 (IFC instance/BIM instance)

An IFC instance i is a concrete instance of an IFC entity.

3.2.2. Definition 2 (BIM product/product)

A BIM product p is a manufactured, supplied, or created object (referred to as an element) for incorporation into a construction project. A product often relates to a geometric or spatial context. Therefore, a product in a BIM model is usually composed of two parts: a 3D geometric shape, g, and its semantic attribute, a. Thus, $p = (g_p, a_p)$. In an IFC specification, products are represented by *IfcProduct* objects.

In an IFC specification, p, g_p, and a_p are described by a collection of IFC instances. The semantic attribute a_p is defined by *IfcPropertyDefinition* and its sub-classes. The computation of a_p has only to parse the *IfcPropertyDefinition* instances. This process is much simpler than the triangulation of g_p. The computational results of a_p can be linked to p simply through *IfcRelDefinesByProperties* instances defining relationships between a_p and p. Hence, the computation of a_p is not discussed in this study, and $p = g_p$. The size of g_p is also denoted as s_p. Because an IFC instance $i \in g_p$ describes some 3D shape for p, the size of geometric data contained in i is denoted as l_i. More generally, s_i is taken to represent the size of geometric data defined in i and all its dependent IFC instances. This study aims to partition the geometric data of p into pieces of geometric data with a size less than a threshold size λ, which does not remove p.

3.2.3. Definition 3 (building information model, BIM model)

A building information model, m, is the digital representation of a collection of IFC instances and the relationships among these IFC instances.

[1] http://www.bos.xyz.

3.2.4. Definition 4 (heterogeneous geometric relationship model)

The heterogeneous geometric relationship model, called HeGeo, describes the reference, decomposition, and association relationships among IFC instances defining 3D shapes. The HeGeo of a BIM is denoted as M, and the HeGeo of a BIM product p is denoted as M_p.

3.2.5. Definition 5 (child instance collection)

The child instance collection of an instance i, denoted as C_i, is the collection of instances referring to, decomposed from, or associated with instance i in the HeGeo.

3.2.6. Definition 6 (descendant instance collection)

The descendant instance collection of instance i, denoted as D_i, is the collection of all lower-level instances of i in the HeGeo.

Fig. 1 presents an example of a HeGeo, child instances, and descendant instances. The whole tree is a HeGeo. Take i_1 as an instance. C_1 is composed of i_2 and i_3, and $D_1 = \{i_2, i_3, i_4, i_5, i_6, i_7, i_8\}$. Obviously, C_i is a subset of D_i.

A BIM model may also include other objects like *IfcActor*, *IfcControl*, *IfcGroup*, *IfcProcess*, and *IfcResource*. Because only BIM products contain geometric data, other objects are ignored.

3.3. Problem definition

Traditionally, a BIM model is organized at the product level [11]. However, products with extremely large complex shapes often have a large volume of geometric data that require a long time to triangulation and transfer for cross-platform visualization, resulting in an unfriendly user experience. This study addresses this issue by partitioning the original large geometric data set into small shapes.

3.3.1. BIM product geometric data partition problem

Given a geometric data size threshold λ and a product p with $s_p > \lambda$, the product geometric data partition problem is to split p into a collection of sub-products $Q(p)$ satisfying $\forall q \in Q(p)$, $s_q \le \lambda$, and $g_p = \sum_{q \in Q(p)} g_q$. In addition, it is also expected that the total geometric data size will increase as little as possible during partitioning. Formally, the BIM product geometric data partition problem is defined as:

$$\begin{aligned} Q(p) &= \underset{Q(p)}{\operatorname{argmin}} \sum_{q \in Q(p)} s_q \\ s.t.\ \ g_p &= \cup_{q \in Q(p)} g_q, \forall q \in Q(p), s_q \le \lambda \end{aligned} \tag{1}$$

4. Heterogeneous geometric relationship model

A BIM product is composed of a collection of IFC instances. In the IFC specification, an IFC instance usually depends on other IFC instances. Current studies in BIM [8,10,11] often have modeled the dependencies among IFC instances as homogeneous reference relationships. These homogeneous models are not suitable for sub-product-level geometric data partition because some relationships cannot be divided into BIM data. Using a different approach, HeGeo categorizes relationships among IFC instances into three families: reference, decomposition, and association relationships. It is noticing that relationships in the HeGeo are totally different from the *IfcRelationship* entities. Because the HeGeo mainly focus on the geometry descriptions (or 3D shapes) of BIM products, which are defined in the Resource layer in the IFC specification. Contrarily, the *IfcRelationship* is defined in the Core layer in the IFC specification. Since the *IfcRelationsihp* stays in a higher layer than the Resource layer, it cannot define the relationships in the HeGeo. Each of the three categories is modeled separately. When all the relationships among IFC instances describing the geometry of a BIM product p have been described, M_p is formulated. This section presents the formation of a heterogeneous geometric relationship model from a BIM model.

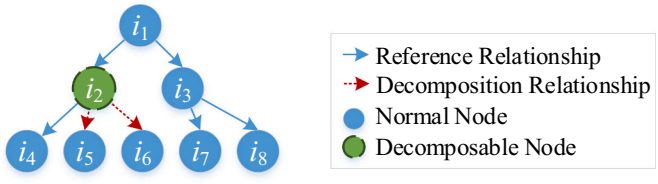


Fig. 1. Example of a heterogeneous geometric relationship model (HeGeo), child instances and descendant instances.

4.1. Reference relationship model

Reference relationships exist widely in BIM data. For example, an *IfcProduct* instance usually refers to an *IfcObjectPlacement* instance and an *IfcProductRepresentation* instance. In this scenario, the *IfcProduct* instance has reference relationships with both the *IfcObjectPlacement* and *IfcProductRepresentation* instances. When modeling the reference relationship, a reference edge from the instance to the referred instance is constructed

Fig. 2 presents the reference relationship model, and Fig. 2(a) is the abstract reference relationship model. When the IFC instance i_l refers to i_m, i_l has a reference edge to i_m. Evidently, an IFC instance may have zero, one, or more than one reference relationships in its definition. Fig. 2(b) presents a concrete example. The instance i_{45} is an instance of *IfcWallStandardCase*, inheriting from *IfcProduct*. Obviously, i_{45} refers to an *IfcOwnerHistory* instance i_2, an *IfcObjectPlacement* instance i_{46}, and an *IfcProductRepresentation* instance i_{51}. Hence, i_{45} has three reference edges to i_2, i_{46}, and i_{51}.

4.2. Decomposition relationship model

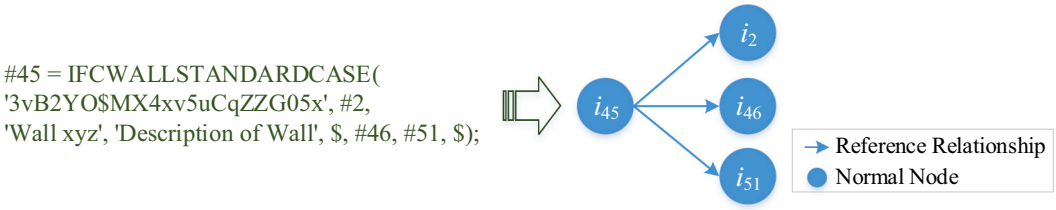
Decomposition relationships exist widely in set-like definitions like *IfcGeometricSet*, *IfcConnectedFaceSet*, and *IfcRepresentation*. Take *IfcRepresentation* as an example. An *IfcRepresentation* instance is composed of at least one *IfcRepresentationItem* instance. In this scenario, the *IfcRepresentation* instance has a decomposition relationship with all the *IfcRepresentationItem* instances. When modeling the decomposition relationship, decomposition edges are constructed from the IFC instance to its decomposed IFC instances.

Fig. 3 shows the decomposition relationship model, and Fig. 3(a) presents the abstract decomposition relationship model. When IFC instance i_l is composed of three IFC instances i_a, i_b, and i_c, three decomposition edges from i_l to i_a, i_b, and i_c are constructed. Fig. 3(b) is a concrete example of the decomposition relationship model. An *IfcShapeRepresentation* instance i_{54} is composed of six *IfcRepresentationItem* instances i_{77}, i_{78}, i_{79}, i_{80}, i_{81}, and i_{82}. In the decomposition model, six decomposition edges from i_{54} to i_{77}, i_{78}, i_{79}, i_{80}, i_{81}, and i_{82} are constructed. Note in this example that i_{54} and i_{20} form a reference relationship.

4.3. Association relationship model

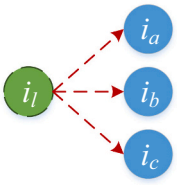
An association relationship model usually occurs when an IFC instance has a relationship with other IFC instances through another IFC instance. A concrete example is *IfcStyledItem*. An *IfcStyledItem* instance has three attributes: an *IfcRepresentationItem* instance, a collection of

(a) IFC instance i_l refers to i_m, and a directed edge from i_l to i_m is constructed.

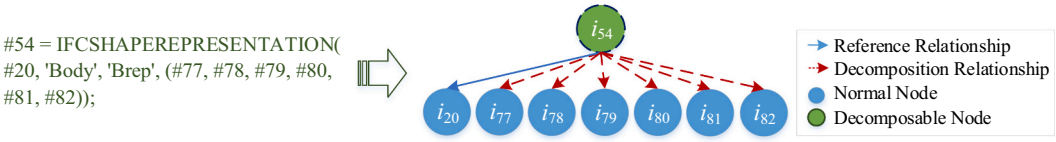


(b) Concrete example of a reference relationship.

Fig. 2. Reference relationship model.

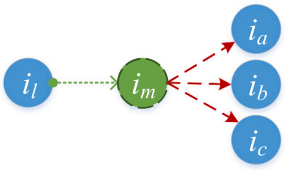


(a) IFC instance i_l is composed of i_a, i_b, and i_c.

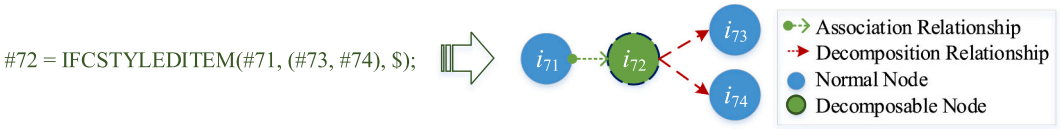


(b) Concrete example of a decomposition relationship model.

Fig. 3. Decomposition Relationship models.



(a) IFC instance i_l has association relationships with i_a, i_b, and i_c through i_m.



(b) Concrete example of an association relationship model.

Fig. 4. Association Relationship models.

styles defined by *IfcStyleAssignmentSelect*, and its name. On top of the *IfcStyledItem*, an *IfcRepresentationItem* instance can construct relationships with several *IfcStyleAssignmentSelect* instances. In this scenario, an *IfcRepresentationItem* instance is said to have association relationships with *IfcStyleAssignmentSelect* instances through an *IfcStyledItem* instance.

Fig. 4 presents the association relationship model, and Fig. 4(a) shows the abstract association relationship model. When i_l has association relationships with i_a, i_b, and i_c through i_m, four edges are formed in the association relationship model, which consist of one association edge from i_l to i_m and three decomposition edges from i_m to i_a, i_b, and i_c. Fig. 4(b) shows a concrete example of an association relationship model. An *IfcRepresentationItem* instance i_{71} is associated with two *IfcStyleAssignmentSelect* instances i_{73} and i_{74} through an *IfcStyledItem* instance i_{72}. In this scenario, i_{71} has an association edge with i_{72}, and i_{72} has decomposition edges with i_{73} and i_{74}.

Each IFC instance that defines the geometry of a given BIM product p can be modeled using the three relationship models described above. Finally, M_p for p can be constructed. Note that the result M_p is a tree-like directed graph without loops for any BIM model. An IFC file, e.g., geometric triangulation, will fail to be computed if a loop reference is present.

Algorithm 1: HierarchicalModel – Build the HeGeo of an IFC file

Input: IFC file f
Output: Hierarchical Models for all BIM products in f
1: **function** HierarchicalModel(f)
2: R = dict(), D = dict(), M = (R, D).
3: **for each** IFC instance i in f:
4: I_R = IFC instances with reference relationships with i.
5: Add I_R to $R[i]$.
6: **if** i defines a decomposition relationship:
7: I_D = IFC instances with decomposition relationships with i.
8: Add I_D to $D[i]$.
9: **end if**
10: **if** i defines an association relationship:
11: Find i_D and I_D with i_D associating with I_D through i.
12: Add I_D to $D[i]$.
13: Add i_D to $R[i]$.
14: **end if**
15: end for
16: return M

Fig. 5 presents a concrete example of a HeGeo. The instance i_{45} is an *IfcProduct* instance, and i_2 is an *IfcOwnerHistory* instance. The IFC instances on the top left define the placement of i_{45}, and the IFC instances on the right describe the representation of i_{45}, or the 3D shape. Apparently, no loops exist in the model. The HeGeo lays the foundation for geometric data segmentation.

Algorithm 1 summarizes the whole process of constructing the HeGeo. Line 2 initializes the variables. M is expressed by two dictionaries R and D, where R and D record the child IFC instances that cannot and can be reorganized respectively. Lines 3 to 15 iterate to construct M for each IFC instance. Lines 4 and 5 add the reference relationships to R. Lines 6 to 9 add the decomposition relationships to D. Lines 10 to 14 process the association relationship. Line 16 returns M. Obviously, all the HeGeo of BIM products in the IFC file are included in M.

Supposing that an IFC file has n IFC instances, it is easy to deduce that Algorithm 1 costs $O(n)$ in time complexity. This means that it is efficient to construct the HeGeo. Because the time complexity of the HeGeo is the same as that of a DrGraph [11], the heterogeneous network modeling of a BIM has a minimal effect on performance compared with homogeneous modeling.

5. Geometric data segmentation

Computation of an IFC instance requires its dependent IFC instances. Take IFC instance "#45 = *IfcWallStandardCase* ('3vB2YO $MX4xv5uCqZZG05x', #2, 'Wall xyz', 'Description of Wall', $, #46, #51, $);" as an example. The computation of IFC instance i_{45} relies on IFC instances i_2, i_{45}, and i_{51}. Without i_2, i_{45}, and i_{51}, little information about i_{45} can be identified. In many more cases, like "#80 = *IfcPolyline* ((#81, #82));", nothing about i_{80} can be computed without i_{81} and i_{82}. To this extent, IFC instances are highly dependent. To correctly identify the full meaning of an IFC instance i, all its children must be considered. This high dependency among IFC instances poses an enormous challenge for data segmentation, as well as for parallel computing, for large and complex BIM products. This section presents a sub-product-level BIM geometric data segmentation algorithm on top of HeGeo, called GeoSeg.

5.1. Segmentation basis

In a HeGeo, the relationships among IFC instances can be modeled as reference, decomposition, and association relationships. According to the IFC specification, the reference relationship cannot be partitioned, but both the decomposition and association relationships can be split. The segmentability of the decomposition and association relationships offers possibilities for the geometric partition of large BIM products.

Fig. 6 presents concrete examples of data segmentation for both decomposition and association relationships. As illustrated in Fig. 6(a), an *IfcShapeRepresentation* instance i_{54} has decomposition relationships with six *IfcRepresentationItem* instances i_{77}, i_{78}, i_{79}, i_{80}, i_{81}, and i_{82}. The instance i_{54} can be divided into at most six segmentations. Here, i_{54} is partitioned into two parts, defined by i_{94} and i_{95}. In this example, i_{94} and i_{95} contain an equal number of IFC instances. In practice, i_{54} can be divided into several segmentations with different numbers of IFC instances. Because i_{54} and i_{20} form a reference relationship, i_{20} is inherited and kept in both i_{94} and i_{95}. Fig. 6(b) presents an example of data segmentation of the association relationship. Originally, an *IfcStyledItem* i_{72} bridges association relationships from i_{71} to both i_{73} and i_{74}. In the example, i_{72} is split into two *IfcStyledItem* instances i_{82} and i_{83}. The instance i_{82} associates i_{71} with i_{73}, whereas i_{83} associates i_{71} with i_{74}.

Although it is impossible to split the reference relationships in an IFC instance, each IFC instance has a limited number of reference relationships with other IFC instances. On the contrary, an IFC instance can contain many IFC instances through decomposition or association relationships. Hence, more effort has been spent on splitting geometric data in terms of decomposition and association relationships.

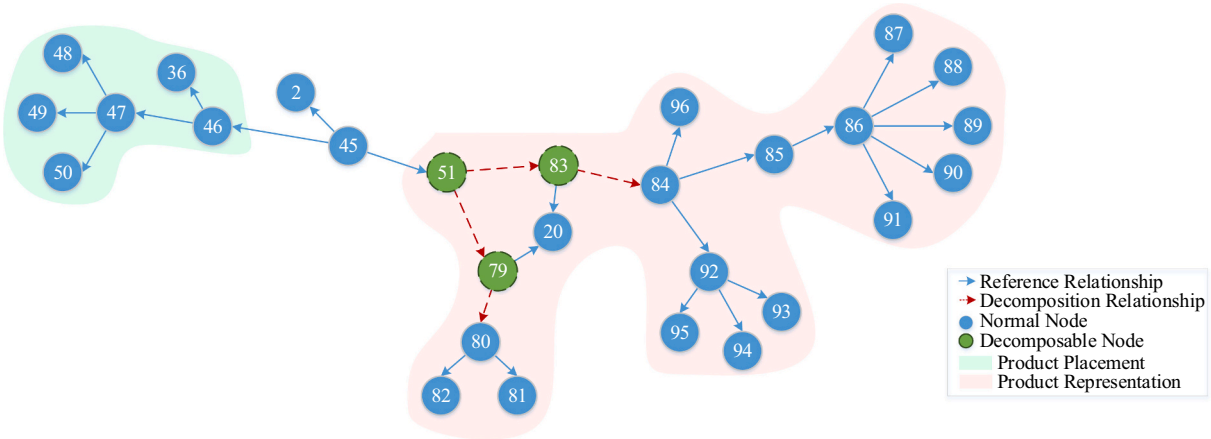


Fig. 5. Example of a HeGeo.

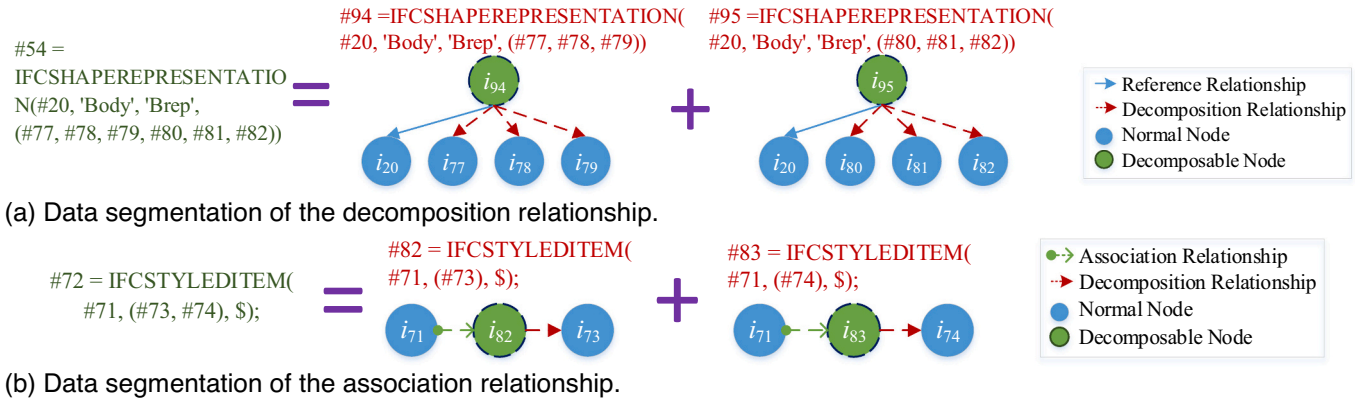


Fig. 6. Examples of data segmentation.

5.2. Segmentation strategy

On top of the HeGeo, the BIM geometric data partition problem is converted into a network partition problem, which is solved using a bottom-up strategy. Undoubtedly, g_p is the IFC instance defining the 3D shape of p. Hence,

$$s_p = \sum_{i \in g_p} l_i \tag{2}$$

This subsection presents four Properties to guide the formation of GeoSeg.

Property 1. If a decomposable node has decomposition relationships with no fewer than two other nodes, then the node is segmentable.

Property 2. *If an IFC instance i is segmentable, the parent of i* is also segmentable.

Property 3. If no fewer than one IFC instance of a product in HeGeo is segmentable, then the product is segmentable.

Property 4. The total size of all sub-products increases less if a BIM product is segmented according to the decomposable node with a larger geometric data size.

Property 1 is evident. As shown in Fig. 7, i_{79} has a decomposition relationship only with i_{80}, and i_{51} has decomposition relationships with both i_{79} and i_{83}. Subsequently, i_{79} is not segmentable, whereas i_{51} is segmentable. Fig. 8(a) illustrates Properties 2 and 3. The instance i_{11} has reference relationships with i_{21}, i_{22}, and i_{23}. The instance i_{31} has decomposition relationships with i_{41}, i_{42}, i_{43}, and i_{44}, and i_{23} has decomposition relationships with i_{32} and i_{33}. Thus, only i_{31} can be segmented. According to Property 2, i_{21} is segmentable because its child i_{31} is segmentable. Similarly, i_{11} is also segmentable. Obviously, when an IFC instance is segmentable, the root IFC instance is also segmentable. A product can be segmented if at least one IFC instance can be segmented in the HeGeo. If all the IFC instances in Fig. 8(a) constitute a BIM product, then the BIM product is segmentable because the four IFC instances (i_{31}, i_{23}, i_{21}, and i_{11}) are all segmentable. Property 3 describes this phenomenon. If an IFC instance in a BIM product is segmentable, then the root IFC instance of the BIM product is also segmentable. Property 4 is easy to deduce. Because both i_{23} and i_{31} are decomposable nodes, the BIM product in Fig. 8(a) can be segmented according to either i_{23} or i_{31}. Fig. 8(b) shows two examples of segmentation according to i_{23} and i_{31}. As shown in the top part of Fig. 8(b), the total size of the two obtained sub-products is 18 if the BIM product is segmented by i_{31}. However, the total size of the two sub-products is 20 after splitting by

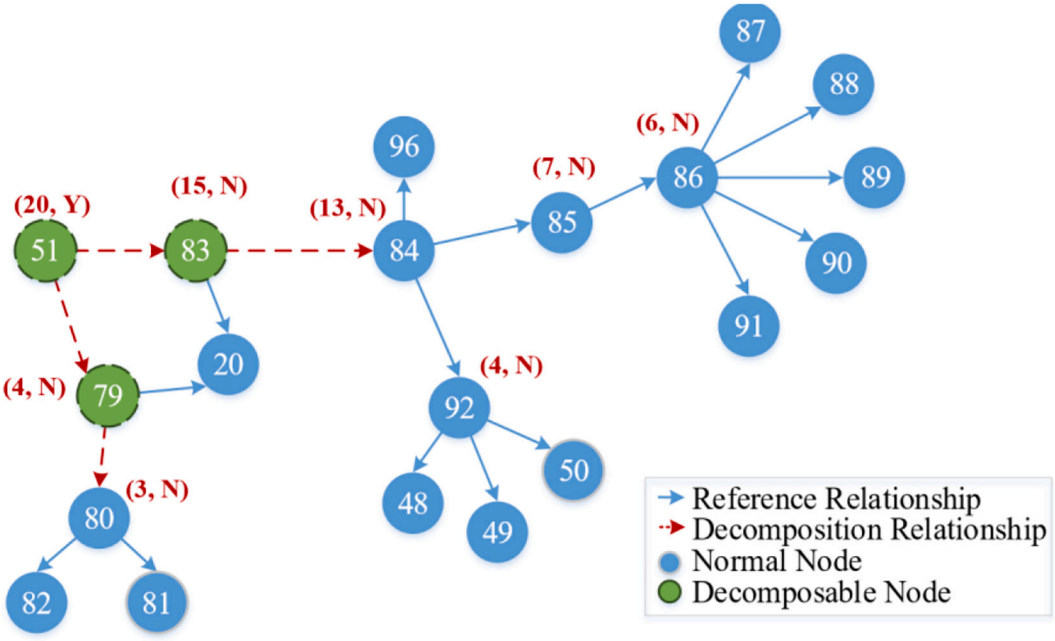


Fig. 7. Bottom-up computation of geometric data size.

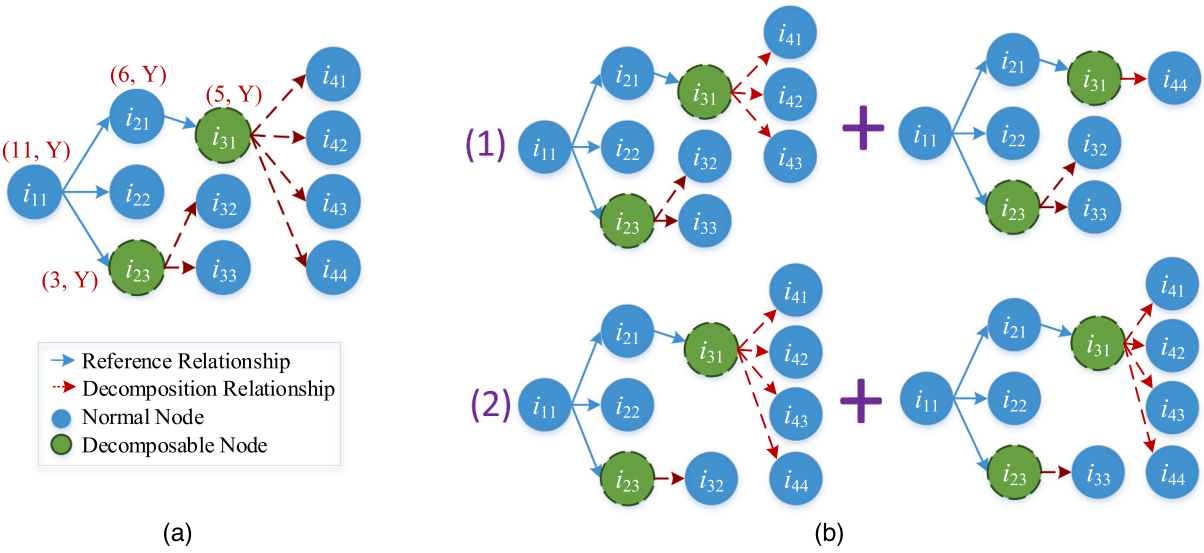


Fig. 8. Illustration of Properties: (a) Original heterogeneous network modeling; (b) Two different segmentation approaches.

i_{23}. In other words, the total size of all sub-products increases less if the BIM product in Fig. 8(a) is split by i_{31} rather than by i_{23}, because i_{31} has a larger geometric data size than i_{23}.

GeoSeg first computes the size and segmentability of each IFC instance using a bottom-up strategy. Specifically, the total size of the geometric data set of an IFC instance is calculated using eq. (2), while the segmentability of an IFC instance is computed using Properties 1 and 2. Fig. 7 presents the computational result for the BIM product p_{45} in Fig. 5. The example assumes that each IFC instance contains an equal amount of geometric data, or in other words, $\forall i_m, i_n \in g_p$, $l_m = l_n = 1$. The computation starts from the leaf nodes. The size of the geometric data contained in each IFC instance is summed from its child IFC instances. The geometric data size and the segmentability status of a non-leaf node are placed around each node. For example, the geometric data size of i_{80} is 3, and i_{80} is unsegmentable. Hence, i_{80} is marked as (3,N) in Fig. 7, where "N" is a symbol denoting unsegmentability. Similarly, i_{51} is marked as (20, Y), where "Y" is a symbol for segmentability, because i_{51} is segmentable and has 30 geometric data attributes. Because i_{51} is an *IfcRepresentation* instance and has decomposition relationships with i_{79} and i_{83}, i_{51} can be segmented.

If a BIM product p turns out to be segmentable based on Property 3, then a top-down strategy was used to split the geometric data according to Property 4. Concretely, the geometric data size threshold λ was checked from the root IFC instance to its leaf IFC instances. When the geometric data size of the IFC instance was greater than λ, λ was transferred to the segmentable child, which included the decomposable node with the largest decomposable node. Accordingly, λ was equal to the difference in total geometric data size from other child instances. Once λ encountered a segmentable IFC instance, its children were split to satisfy λ.Fig. 9 presents an example of the top-down segmentation process. Initially, $\lambda = 10$. Because i_{21} is segmentable, λ is subtracted from the geometric data size of other children and itself, which is 5. Subsequently, i_{21} receives $\lambda = 5$. Similarly, $\lambda = 4$ in i_{31}. Because i_{31} is a segmentable IFC instance, its children are reorganized to satisfy λ. As a result, the four children of i_{31} are split into two segmentations. This study used a greedy strategy to reorganize the IFC instances. Subsequently, i_{41}, i_{42}, and i_{43} were grouped into one data segment, whereas i_{44} was in the other data segment.

After the IFC instances belonging to g_p were segmented into different groups, it was simple to restore the data segmentations according to the IFC instances in each group.

Algorithm 2: GeoSeg – Segment Geometric Data of an IFC File

Input: IFC file f, geometric data size threshold λ
Output: Geometric data segmentations
1: **function** GeoSeg(f, λ)
2: Compute $M = (R, D)$ using Algorithm 1.
3: P = BIM products in f.
4: **for each** BIM product p in P:
5: S = dict(), F = dict().
6: Compute S and F from R using the bottom-up strategy.
7: $s_p = S[p]$.
8: **if** $s_p \leq \lambda$:
9: Output p. **continue**.
10: Segment p using the top-down strategy.
11: Output data segments.
12: **end for**
13:**return**

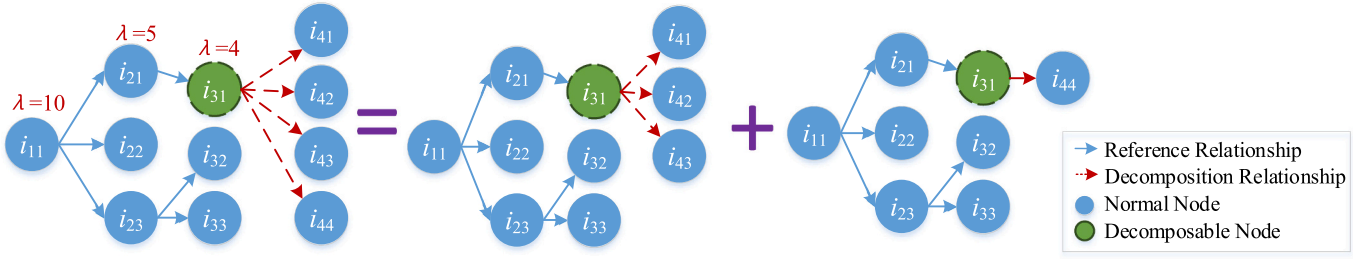


Fig. 9. Illustration of data segmentation.

Algorithm 2 summarizes the process of the GeoSeg. Line 2 uses Algorithm 1 to construct HeGeo. Line 3 finds all the BIM products in f. Two solutions can be chosen. The first one checks whether the class of an IFC instance inherits from *IfcProduct*, whereas the second checks whether an IFC instance refers to an *IfcRepresentation* instance. Lines 4 to 15 iterate each BIM product to segment the geometric data. Lines 5 and 6 initiate the variables, where S and F record the geometric data size and segmentability of each IFC instance respectively. Line 7 uses a bottom-up strategy to compute S, while Line 8 computes F using Properties 1 and 2. Line 9 directly returns p if p is not segmentable according to Property 3. Lines 10 to 12 process the BIM products with geometric data size less than λ, and Lines 13 and 14 segments p with $s_p > \lambda$ and outputs the data segmentations according to Property 4.

Since both Lines 6 and 10 cost $O(s_p)$, Lines 5–11 cost $O(s_p)$ in time complexity. Hence, the time complexity of Lines 4–12 is $\sum_{p\in P} O(s_p) = O(\sum_{p\in P} s_p) = O(n)$. As discussed above, Algorithm 1 (line 2) also costs $O(n)$. In summary, the time complexity of Algorithm 2 is $O(n)$.

The product-level BIM data segmentation scheme [11] splits an IFC file according to the *IfcProduct* instance, whereas the floor-level BIM data segmentation scheme [10] partitions an IFC file according to the *IfcBuldingStorey* instance. Hence, GeoSeg turns out to be a product-level and floor-level segmentation algorithm when it segments IFC instances directly according to *IfcProduct* instances and *IfcBuildingStorey* instances respectively. GeoSeg can be embedded in any parallel computing framework to enable the parallel triangulation of BIM data. When GeoSeg cooperates with MapReduce and splits IFC instances according to *IfcBuildingStorey* instances, it turns out to be a floor-level parallel computing scheme.

6. Experiments

This section systemically evaluates the performance of the proposed scheme in the triangulation and rendering processes.

6.1. Baseline

This study was the first to study the sub-product level partition of BIM data. The proposed scheme is compatible with any triangulation tools and any visualization tools. Without loss of generality, *IfcOpenShell* [32] and WebBIM [4] were taken as baselines for the triangulation and rendering processes respectively. Similar results could be obtained using any other triangulation and visualization tools. BIMTriSer [8], the product-level triangulation scheme, was another baseline for the triangulation process. Because floor-level segmentation and triangulation schemes [9] have similar performance to product-level ones in BIM models with only one or a few BIM products, the performance of the floor-level scheme was not examined in the experiments.

6.2. Datasets

Experiments were conducted on extensive BIM models to evaluate both the triangulation and rendering processes. Because GeoSeg generalizes to the product-level segmentation scheme when constraining segmentation nodes to *IfcProduct* instances, GeoSeg has similar performance to the product-level scheme in most detailed design BIM models produced by design tools like Autodesk Revit. Hence, the performance comparisons between GeoSeg and product-level segmentation schemes were not listed in this study. Nine representative BIM models were selected in the experiments, with sizes from 2.38 M to 205.66 M. All nine BIM models were designed and exported from SketchUp, where all the 3D shapes in a BIM model were organized into a single BIM product. Contrary to BIM models exported from Autodesk Revit that contain many BIM products, those exported from SketchUp usually contained only one BIM product. Occasionally, a BIM model may involve two or more large BIM products. In this scenario, the performance of product-level segmentation schemes used in a parallel computing framework depends on the number of computing processes and the number of large BIM products. This is the case because the running time in parallel computing is equal to the running time of the most time-consuming process. For example, if a BIM model contains three large BIM products and is executed on a parallel computing cluster with no fewer than three processes, then the total running time is equal to the computing time of the largest BIM product. When the size of any BIM product in a BIM model is no greater than λ, GeoSeg and product-level segmentation schemes have equivalent performance. To compare performance more clearly, only BIM models with one BIM product were chosen in the experiments. Fig. 10 shows the 3D shapes, and Table 2 presents the descriptions.

6.3. Environment setting

All timings were obtained on a PC with Intel i7 5600U 2.6 GHz CPU and 16 Gbytes RAM. The algorithms were implemented in Java. Fig. 11 presents a typical use of the proposed GeoSeg. First, an original BIM model was split into several IFC sub-files that could be computed separately. Then the small IFC slices were assigned to different computing units in a parallel computing cluster. In the experiments, each computing unit was equipped with a triangulation tool like *IfcOpenShell* [32]. The parallel computing cluster generated the triangulation meshes, which could be requested by online visualization tools such

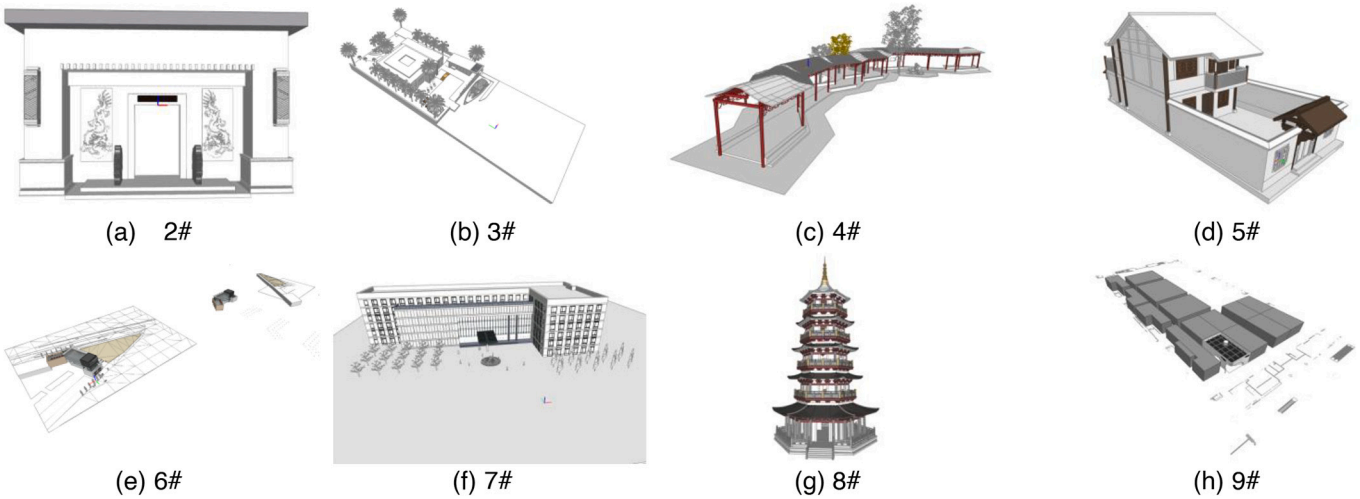


Fig. 10. BIM Models for experiments.

Table 2
Descriptions of BIM models for experiments.

#	Size (Mbyte)	Description
1	2.38	Courtyard home
2	6.89	Main entrance
3	20.47	Seascape architecture
4	23.00	Cloister
5	33.40	Villa
6	60.87	Commercial building
7	111.32	Office building
8	133.94	Pagoda
9	205.66	Factory building

as WebBIM [4] in a multi-thread manner.

6.4. Triangulation evaluation

The first step was to evaluate the triangulation speed-up using the proposed GeoSeg scheme. Java Parallelism [33] was used to simulate parallel computing. The use of any other computing framework will generate similar results.

Some parameters can affect the triangulation process using the proposed GeoSeg, such as the number of triangulation processes κ and the threshold for geometric data size λ. This study first presents the triangulation efficiency with $\kappa = 6$ and $\lambda = 5.0$ Mbytes. Table 3 shows the experimental results. The triangulation of the nine BIM models cost 5.35, 8.05, 30.50, 32.90, 41.81, 93.34, 290.71, 345.42, and 504.23 s respectively using *IfcOpenShell*. When the model was equipped with the proposed GeoSeg algorithm, triangulation times were reduced to 2.53, 3.23, 8.48, 10.07, 13.82, 55.21, 79.15, 91.60, and 173.90 s respectively. Obviously, the larger the BIM model, the more time is required to complete the triangulation. The triangulation process achieved 2.11, 2.49, 3.60, 3.27, 3.02, 2.41, 3.67, 3.78 and 3.07 speed-up factors respectively. The proposed GeoSeg scheme improved the triangulation efficiency of BIM models with large complex products by 3.05 ± 0.57 times with the same hardware resources and the same triangulation tool. These results were obtained because GeoSeg enabled the triangulation of large BIM products in multiple processes using parallel computing. In contrast, state-of-art BIM triangulation tools such as *IfcOpenShell* and floor-level and product schemes can triangulate a large BIM product in only one process. Note also that BIMTriSer performed the worst among all nine BIM models because the nine BIM models were organized into only one BIM product, which disabled the BIM segmentation in BIMTriSer. In addition, BIMTriSer required extra time when trying to segment the original BIM file into BIM products.

The next step was to evaluate the effects of κ and λ on triangulation using GeoSeg. The triangulation times may fluctuate slightly due to resource scheduling from the operating system. To mitigate the noise of resource scheduling, the larger BIM models, i.e., 6#, 7#, 8#, and 9#, were taken as representative in the experiments.

To observe triangulation times, λ was set to 5.0 Mbytes and κ was increased from 1 to 10. Fig. 12(a) shows the empirical results. Obviously, the larger κ is, the longer the triangulation time. This phenomenon indicates that using parallel computing can improve the triangulation efficiency for BIM geometric data. This result is the same as that obtained by applying a product-level parallel computing scheme to BIM models constructed with several BIM products [8]. Note that the triangulation times can be reduced only slightly when $\kappa \geq 4$ because the CPU is fully occupied by triangulation processing. In other words, with a single computer, the CPU becomes the bottleneck. In this scenario, the suggestion is to dispatch the segmented data to multiple computers in a cluster to improve computing efficiency. This works because more computers can provide more computing resources and more triangulation processes can be launched to process the split sub-files simultaneously.

Experiments were also conducted to check the effects of λ and κ in the proposed scheme. As discussed earlier, there was no noticeable benefit in BIM triangulation efficiency with $\kappa \geq 4$. Without loss of generality, $\kappa = 6$. The thresholds for geometric data size λ were set to 3.0, 5.0, 10.0, 20.0, and 30.0 Mbytes. Fig. 12(b) shows the experimental results. Intuitively, the larger λ is, the lower the triangulation efficiency. This is the case mainly because a larger λ generates a smaller number of sub-files. An insufficient number of sub-files may cause incomplete use of computer resources like CPU and internal memory. Take the 6# BIM model as an example. When λ was set to 3.0, 5.0, 10.0, 20.0, and 30.0 Mbytes, GeoSeg split the 6# BIM model into 25, 12, 10, 4, and 3 IFC slices respectively. Although the computers used in the experiments could execute $\kappa = 4$ triangulation processes simultaneously, only three processes were launched when $\lambda = 30.0$ Mbytes because only three sub-files were generated by GeoSeg with $\lambda = 30.0$ Mbytes in the 6# BIM model. In these scenarios, BIM triangulation might not make full use of the computer resource, resulting in lower triangulation efficiency. The 6# BIM model was observed to be most efficient with $\lambda = 10.0$ Mbytes. Similarly, as λ becomes larger, the smaller number of sub-files leads to lower triangulation efficiency. It was also found that triangulation took somewhat more time when $\lambda = 2.0$ Mbytes than when $\lambda = 10.0$ Mbytes. The reason for this was that more time was required to split the original BIM file when $\lambda = 2.0$ Mbytes than when $\lambda = 10.0$ Mbytes.

The number of triangulation processes κ theoretically defines the maximum number of triangulation processes. Due to its limited

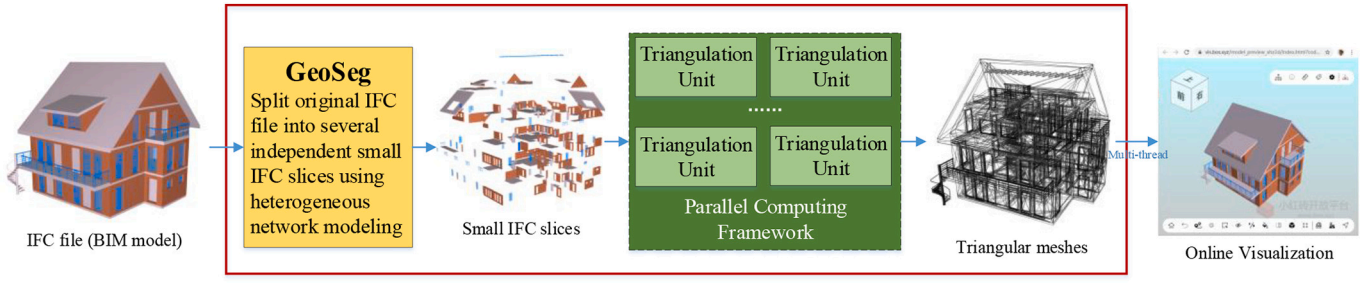


Fig. 11. Use of GeoSeg in a parallel computing framework to improve BIM triangulation and visualization.

Table 3
Comparison of Triangulation Efficiency. $\kappa = 6$ and $\lambda = 5.0$ Mbytes.

#	BIMTriSer(s)	IfcOpenShell(s)	IfcOpenShell + GeoSeg(s)	Speed-up
1	5.85	5.35	2.53	2.11
2	8.65	8.05	3.23	2.49
3	31.2	30.50	8.48	3.60
4	34.7	32.90	10.07	3.27
5	46.51	41.81	13.82	3.02
6	163.44	133.34	55.21	2.41
7	315.91	290.71	79.15	3.67
8	373.42	346.42	91.60	3.78
9	630.93	534.23	173.90	3.07

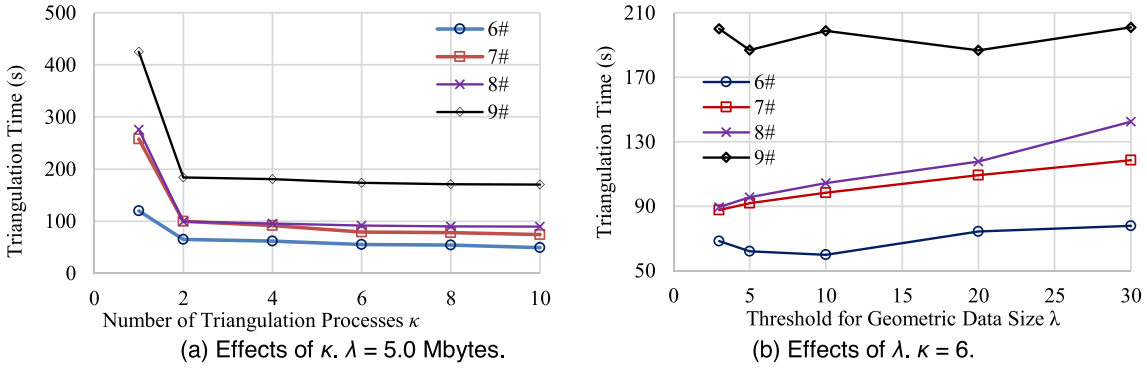


Fig. 12. Effects of the number of triangulation processes κ and the geometric data size λ.

computing resources, a computer can launch a limited number of triangulation processes. If Java Parallelism has not set the number of processes, then the computer will launch an optimal number of triangulation processes according to its computing resources. In other words, κ can be automatically set by the Java platform using Java Parallelism. The best value of λ differs according to the computing resources available. In practice, the best value of λ can be obtained by testing typical sizes of BIM models.

6.5. Online rendering evaluation

The online rendering evaluation experiments were conducted on a computer connected to the Internet through a 100 MBps bandwidth wireline. In the experiments, the geometric data were triangulated and re-organized with $\lambda = 3.0$ Mbytes. In addition, four threads were sent simultaneously to obtain the triangular meshes using the proposed segmentation scheme. Table 4 presents the experimental results. WebBIM required 0.44, 1.44, 3.23, 5.82, 4.23, 7.56, 16.28, 20.30, and 31.71 s respectively to load the triangular meshes for rendering through WebG. With the proposed GeoSeg scheme, the triangular mesh loading times were reduced to 0.29, 0.73, 2.13, 3.10, 2.87, 4.18, 9.19, 14.59, and 17.51 respectively. The data size of the triangular mesh directly affects the loading time. A larger BIM model usually has a larger triangular data set, resulting in a longer loading time. Use of the proposed scheme was observed to improve loading efficiency by 1.52, 1.56, 1.52, 1.88, 1.47, 1.81, 1.77, 1.39, and 1.81 times respectively. In other words, the proposed segmentation scheme increased the triangular mesh loading efficiency by 1.53 ± 0.29 times. The triangular meshes of IFC slices generated by GeoSeg can be stored separately, which enables WebBIM to request the triangular meshes of a large BIM product using multiple threads. Without GeoSeg, WebBIM can request the triangular meshes of a large BIM product in only one thread. Loading efficiencies do not improve as much as with BIM triangulation because bandwidth becomes the crucial bottleneck when obtaining triangular meshes from the cloud.

Table 4
Comparison of Triangular Mesh Loading Efficiency. $\kappa = 4$ and $\lambda = 3.0$ Mbytes.

#	WebBIM (s)	WebBIM + GeoSeg (s)	Speed-up
1	0.44	0.42	1.05
2	1.14	1.03	1.11
3	3.23	2.13	1.52
4	5.82	3.10	1.88
5	4.23	2.87	1.47
6	7.56	4.18	1.81
7	16.28	9.19	1.77
8	20.30	14.59	1.39
9	31.71	17.51	1.81

Another advantage of the proposed scheme is the incremental rendering of triangular mesh data. Once the triangular meshes of a sub-product have been obtained, they are rendered using WebGL. In this manner, users can see the 3D shapes of the BIM model incrementally. Fig. 13 presents the 3D shapes of the 8# BIM model after 6.25%, 25%, 50%, and 75% of the sub-products have been loaded and rendered. Traditionally, WebBIM can render a BIM model only after the whole BIM product has been fetched. Apparently, this incremental rendering improves the user experience of Web BIM visualization tools to some extent.

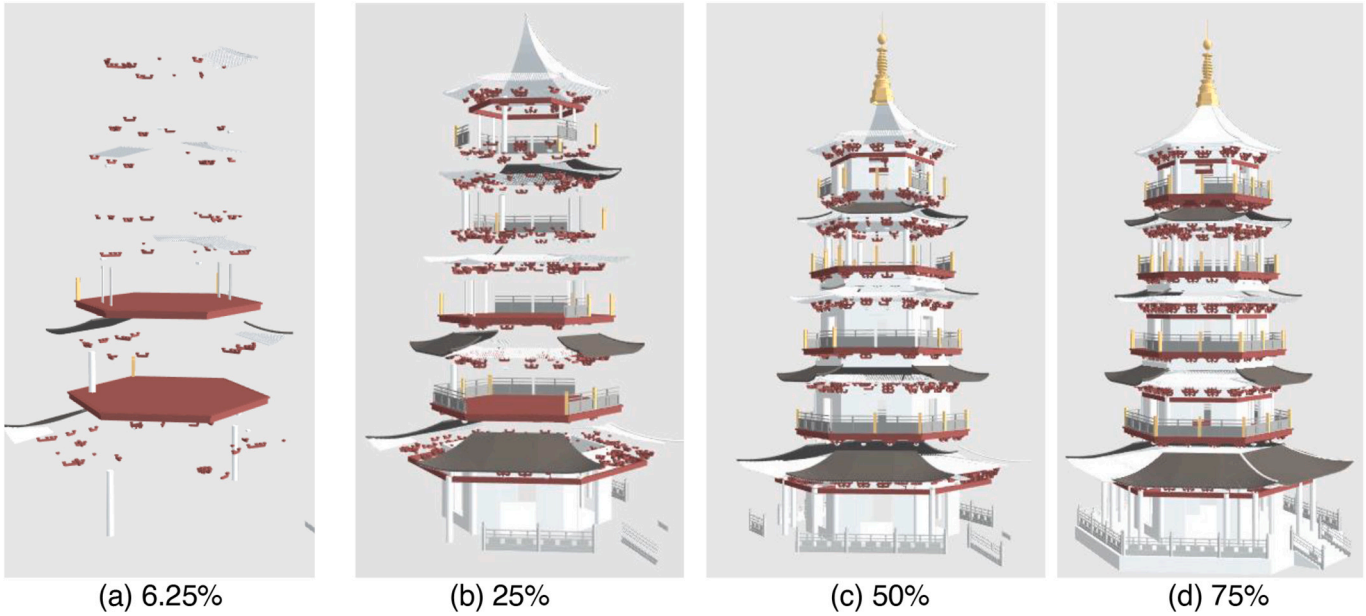


Fig. 13. Incremental rendering of the 8# BIM model.

7. Conclusions

Large and complex BIM products pose challenges in both triangulation and rendering processes for current cross-platform BIM visualization tools. To address this issue, this study proposed a BIM geometric data segmentation scheme that can be applied to any BIM file. The first step was to present a heterogeneous geometric relationship model, called HeGeo for short. On top of HeGeo, a geometric data segmentation algorithm, called GeoSeg, was then developed. GeoSeg divides the original BIM model into several independent sub-files with similar sizes, which can be computed separately in a parallel computing unit. The experimental results showed that GeoSeg improved the triangulation efficiency by 3.05 ± 0.57 times with the same hardware resources and the same triangulation tool through a parallel computing framework, and increased the triangular mesh loading efficiency by 1.53 ± 0.29 times by processing requests concurrently. In addition, GeoSeg improved the user experience of online BIM visualization tools through incremental rendering.

The innovation of this study involved a novel heterogeneous modeling and segmentation algorithm for parallel triangulation and visualization that works on any BIM data. A heterogeneous network modeling scheme for BIM was proposed, which categorizes the geometric relationships into reference, decomposition, and association relationships according to IFC. Then, a segmentation algorithm was developed to split the original BIM model at any level. If the segmentation nodes are constrained to BIM products and floors, the proposed algorithm generalizes to product-level and floor-level schemes respectively. Hence, the proposed scheme can be applied to any BIM model and can facilitate BIM adoption by all stakeholders during a building's life cycle.

Real-time rendering of large-scale complex BIM products is still a challenging task. Admittedly, although the proposed scheme cannot solve this issue completely, it provides a substantial supplement to many solutions in this area. In other words, the solution proposed here can be used jointly with current solutions such as mesh simplification and mesh streaming to develop a more powerful real-time BIM visualization tool. This possibility will be studied in future work.

Declaration of Competing Interest

None.

Acknowledgements

This work was supported by the Beijing Natural Science Foundation under grant no. 4202017, the National Natural Science Foundation of China under grant nos. 61871020, 61671188 and 71601013, the Youth Talent Support Program of Beijing Municipal Education Commission under grant no. CIT&TCD201904050, the Youth Talent Project of Beijing University of Civil Engineering & Architecture (BUCEA), the Fundamental Research Funds for BUCEA under grant no X20039, the Key Science and Technology Plan Project of Beijing Municipal Education Commission of China under grand no. KZ201810016019, and the High Level Innovation Team Construction Project of Beijing Municipal Universities under grant no. IDHT20190506.

References

[1] C.Y. Lee, H.Y. Chong, X. Wang, Streamlining digital modeling and building information modelling (BIM) uses for the oil and gas projects, Archiv. Comput. Meth. Eng. 25 (2) (2018) 349–396, https://doi.org/10.1007/s11831-016-9201-4.
[2] R. Vanlande, C. Nicolle, C. Cruz, IFC and building lifecycle management, Autom. Constr. 18 (1) (2008) 70–78, https://doi.org/10.1016/j.autcon.2008.05.001.
[3] Q. Lu, J. Won, J.C. Cheng, A financial decision-making framework for construction projects based on 5D Building Information Modeling (BIM), Int. J. Proj. Manag. 34 (1) (2016) 3–21, https://doi.org/10.1016/j.ijproman.2015.09.004.
[4] X. Zhou, J. Wang, M. Guo, Z. Gao, Cross-platform online visualization system for open BIM based on WebGL, Multimed. Tools Appl. 78 (20) (2019) 28575–28590, https://doi.org/10.1007/s11042-018-5820-0.
[5] X. Liu, N. Xie, K. Tang, J. Jia, Lightweighting for Web3D visualization of large-scale BIM scenes in real-time, Graph. Model. (2016) 40–56, https://doi.org/10.1016/j.gmod.2016.06.001.
[6] J. Beetz, L. van Berlo, R. de Laat, P. van den Helm, BIMserver.org – an open source IFC model server, in: Proc. of the CIB W78, 2010, p. 8. https://www.researchgate.net/publication/254899282.
[7] Z. Xu, L. Zhang, H. Li, Y.H. Lin, S. Yin, Combining IFC and 3D tiles to create 3D visualization for building information modeling, Autom. Constr. 109 (2020) 102995, https://doi.org/10.1016/j.autcon.2019.102995.
[8] X. Zhou, J. Zhao, J. Wang, X. Huang, X. Li, M. Guo, P. Xie, Parallel computing-based online geometry triangulation for building information modeling utilizing big data, Autom. Constr. 107 (2019) 102942, https://doi.org/10.1016/j.autcon.2019.102942.
[9] M. Bilal, L.O. Oyedele, J. Qadir, K. Munir, S.O. Ajayi, O.O. Akinade, H.A. Owolabia, H.A. Alaka, M. Pasha, Big Data in the construction industry: a review of present status, opportunities, and future trends, Adv. Eng. Inform. 30 (3) (2016) 500–521, https://doi.org/10.1016/j.aei.2016.07.001.
[10] Y. Jiao, S. Zhang, Y. Li, Y. Wang, B. Yang, L. Wang, An augmented MapReduce framework for building information modeling applications, in: Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative

Work in Design (CSCWD), 2014, pp. 283–288, https://doi.org/10.1109/CSCWD.2014.6846856.
[11] X. Zhou, J. Zhao, J. Wang, M. Guo, J. Liu, H. Shi, Towards product-level parallel computing of large-scale building information modeling data using graph theory, Build. Environ. 169 (2020) 106558, https://doi.org/10.1016/j.buildenv.2019.106558.
[12] H. Fan, L. Meng, A three-step approach of simplifying 3D buildings modeled by CityGML, Int. J. Geogr. Inf. Sci. 26 (6) (2012) 1091–1107, https://doi.org/10.1080/13658816.2011.625947.
[13] A. Forberg, Generalization of 3D building data based on a scale-space approach, ISPRS J. Photogramm. Remote Sens. 62 (2) (2007) 104–111, https://doi.org/10.1016/j.isprsjprs.2007.01.002.
[14] J.S. Kim, K.J. Li, Simplification of geometric objects in an indoor space, ISPRS J. Photogramm. Remote Sens. (2019) 146–162, https://doi.org/10.1016/j.isprsjprs.2018.11.017.
[15] C.E. Tolmer, C. Castaing, Y. Diab, D. Morand, CityGML and IFC: going further than LoD, Digit. Herit. Int. Congr. (2013) 645–648, https://doi.org/10.1109/DigitalHeritage.2013.6743808.
[16] X. Zhou, J. Zhao, J. Wang, D. Su, H. Zhang, M. Guo, M. Guo, Z. Li, OutDet: an algorithm for extracting the outer surfaces of building information models for integration with geographic information systems, Int. J. Geogr. Inf. Sci. 33 (7) (2019) 1444–1470, https://doi.org/10.1080/13658816.2019.1572894.
[17] M. Englert, M. Klomann, Y. Jung, Optimized streaming of large web 3D applications, Virtual Syst. Multimedia (2017) 1–8, https://doi.org/10.1109/vsmm.2017.8346265.
[18] M. Zampoglou, K. Kapetanakis, A. Stamoulias, A.G. Malamos, S. Panagiotakis, Adaptive streaming of complex Web 3D scenes based on the MPEG-DASH standard, Multimed. Tools Appl. 77 (1) (2018) 125–148, https://doi.org/10.1007/s11042-016-4255-8.
[19] M. Limper, M. Thöner, J. Behr, D.W. Fellner, SRC - a streamable format for generalized web-based 3D data transmission, in: International ACM Conference on 3D Web Technologies, 2014, pp. 35–43, https://doi.org/10.1145/2628588.2628589.
[20] C. Portaneri, P. Alliez, M. Hemmer, L. Birklein, E. Schoemer, Cost-driven framework for progressive compression of textured meshes, in: 10th ACM Multimedia Systems Conference, 2019, pp. 175–188, https://doi.org/10.1145/3304109.3306225.
[21] A. Maglo, G. Lavoue, F. Dupont, C. Hudelot, 3D mesh compression: survey, comparisons, and emerging trends, ACM Comput. Surv. 47 (3) (2015) 1–41, https://doi.org/10.1145/2693443.
[22] L. Huettenberger, C. Heine, C. Garth, Decomposition and simplification of multivariate data using Pareto sets, IEEE Trans. Vis. Comput. Graph. 20 (12) (2014) 2684–2693, https://doi.org/10.1109/TVCG.2014.2346447.
[23] R. Yi, Y. Liu, Y. He, Delaunay mesh simplification with differential evolution, ACM Trans. Graph. 37 (6) (2019) 263, https://doi.org/10.1145/3272127.3275068.
[24] J. Choi, H. Park, J. Paek, J.G. Ko, Poster: reactive mesh simplification for augmented reality head mounted displays, in: MobiSys 2018, 2018, p. 527. http://nsl.cau.ac.kr/~jpaek/docs/lpgl-poster-mobisys2018.pdf.
[25] S. Kwon, D. Mun, B.C. Kim, S. Han, Feature shape complexity: a new criterion for the simplification of feature-based 3D CAD models, Int. J. Adv. Manuf. Technol. 88 (5) (2017) 1831–1843, https://doi.org/10.1007/s00170-016-8937-1.
[26] T.W. Kang, C.H. Hong, A study on software architecture for effective BIM/GIS-based facility management data integration, Autom. Constr. 54 (54) (2015) 25–38, https://doi.org/10.1016/j.autcon.2015.03.019.
[27] A. Doumanoglou, P. Drakoulis, N. Zioulis, D. Zarpalas, P. Daras, Benchmarking open-source static 3D mesh codecs for immersive media interactive live streaming, IEEE J. Emerg. Select. Topics Circuits Syst. 9 (1) (2019) 190–203, https://doi.org/10.1109/jetcas.2019.2898768.
[28] J.M. Noguera, J.R. Jimenez, Mobile volume rendering: past, present, and future, IEEE Trans. Vis. Comput. Graph. 22 (2) (2016) 1164–1178, https://doi.org/10.1109/TVCG.2015.2430343.
[29] X. Gao, T. Martin, S. Deng, E. Cohen, Z. Deng, G. Chen, Structured volume decomposition via generalized sweeping, IEEE Trans. Vis. Comput. Graph. 22 (7) (2016) 1899–1911, https://doi.org/10.1109/TVCG.2015.2473835.
[30] S. Katz, A. Tal, Hierarchical mesh decomposition using fuzzy clustering and cuts, Int. Conf. Comput. Graph. Interact. Techniques 22 (3) (2003) 954–961, https://doi.org/10.1145/1201775.882369.
[31] P. Theologou, I. Pratikakis, T. Theoharis, Unsupervised spectral mesh segmentation driven by heterogeneous graphs, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2) (2017) 397–410, https://doi.org/10.1109/TPAMI.2016.2544311.
[32] T. Krijnen, A. Kämäräinen, IfcOpenShell [Online]. http://ifcopenshell.org, 2020. Accessed date: 1 May 2020.
[33] G. Pinto, A. Canino, F. Castor, G. Xu, Y.D. Liu, Understanding and overcoming parallelism bottlenecks in ForkJoin applications, Autom. Softw. Eng. (2017) 765–775, https://doi.org/10.1109/ASE.2017.8115687.
[34] Z. Xu, L. Zhang, H. Li, Y.H. Lin, S. Yin, Combining IFC and 3D tiles to create 3D visualization for building information modeling, Autom. Constr. 109 (2020) 102995, https://doi.org/10.1016/j.autcon.2019.102995.