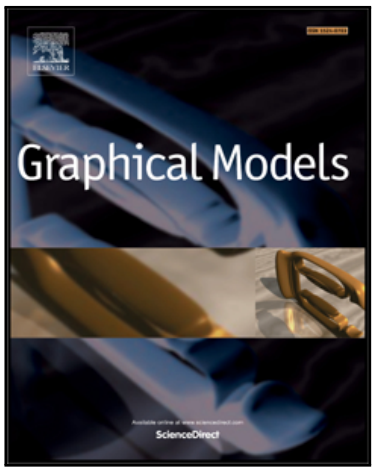


Lightweighting for Web3D Visualization of Large-scale BIM Scenes in Real-time

Xiaojun Liu, Ning Xie, Kai Tang, Jinyuan Jia

PII: S1524-0703(16)30017-0
DOI: 10.1016/j.gmod.2016.06.001
Reference: YGMOD 955

To appear in: *Graphical Models*

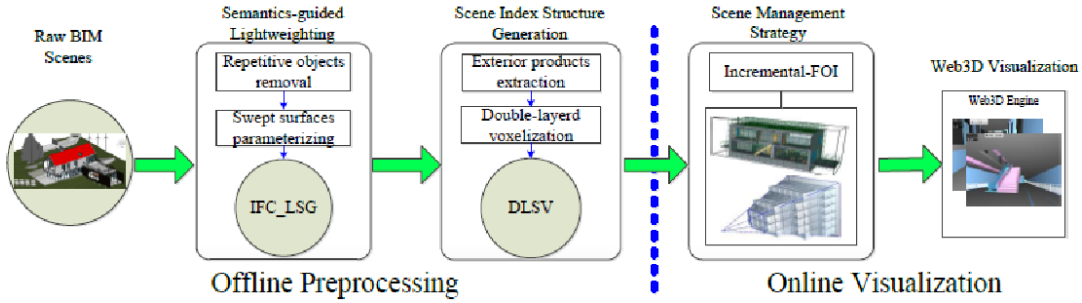
Received date: 14 October 2015
Revised date: 24 April 2016
Accepted date: 14 June 2016

Please cite this article as: Xiaojun Liu, Ning Xie, Kai Tang, Jinyuan Jia, Lightweighting for Web3D Visualization of Large-scale BIM Scenes in Real-time, *Graphical Models* (2016), doi: 10.1016/j.gmod.2016.06.001

Highlights

- Semantics-guided lightweighting is proposed to reduce the amount of data processing in the front end by removing the redundant data and creating an IFC Lightweight Scene Graph (IFC_LSG).
- Double-Layered Sparse Voxel (DLSV) is proposed for data indexing to improve the access efficiency of real-time Web3D building visualization.
- Incremental Frustum of Interest (I-FOI) is proposed to manage the scene by combining the rendering pipeline and the current scene index.

Graphical Abstract



Lightweighting for Web3D Visualization of Large-scale BIM Scenes in Real-time

Xiaojun Liu[a], Ning Xie[a], Kai Tang[b], Jinyuan Jia[a,*]

[a]*School of Software Engineering, Tongji University, Shanghai 201804*
[b]*Hong Kong University of Science and Technology, Hong Kong*

Abstract

As a result of informatization in construction, Building Information Modeling (BIM) has now become a core technology for smart construction. We present a Web3D-based lightweighting solution for real-time visualization of large-scale BIM scenes, considering the redundancy, semantics, and the parameterization of BIM data under the limited resources of network bandwidth and web browsers. Taking the Industry Foundation Classes (IFC) as the input data format, we firstly conduct a semantics-guided lightweighting operation on the raw BIM scenes by removing the repetitive objects and parameterizing the swept surfaces. Secondly, we extract the exterior products from the raw BIM buildings for visibility culling and construct a Double-Layered Sparse Voxel (DLSV) index based on sparse voxelization. Thirdly, we integrate the above two together into a new data structure named Incremental Frustum of Interest (I-FOI) to manage the scene data in real-time. Our experiments demonstrate that: (1) with the semantics information, our method is able to significantly reduce the redundancy of raw large-scale BIM scenes; (2) the DLSV structure supports progressive data loading and facilitates the indoor/outdoor visibility culling efficiently; and (3) the I-FOI introduces a frustum incremental-driven mechanism into progressive data loading or unloading to improve the efficiency of resource consumption.

Keywords: `Web3D`, Real-time, BIM/IFC, Semantics, Lightweight

*Corresponding author
Email address: `jyjia@tongji.edu.cn` (Jinyuan Jia)

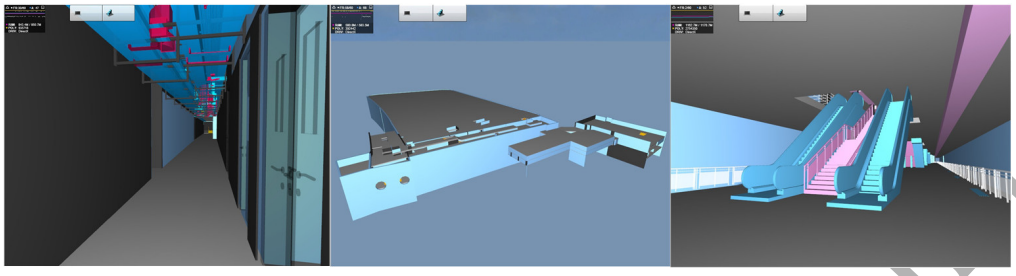


Figure 1: A subway image: this building consists of more than 3,000,000 entities and about 30,000 products. It can be rendered in real-time with over 30fps in our prototype system.

1. Introduction

With the development of mobile internet, 3D urban visualization or walkthrough applications, especially Web3D-based ones, have become a hot research topic [1]. As an effective means of comprehensive inspection/examination of complex buildings, online walkthrough of large-scale Building Information Modeling (BIM) scenes is imperative. However, Web3D-based applications are always restricted by limited resources of network transmission, browser equipment, and webpages, etc. Meanwhile, BIM data always maintains the information employed throughout a building's entire lifecycle [2], resulting in large amount of BIM data and its consumption. For instance, for the subway station scene shown in Figure 1, it includes more than 3,000,000 entities, and in Autodesk Revit 2014 more than 3.7 GB of disk space is needed to store them. However, even this much powerful commercial stand-alone software is still incapable of supporting real-time operations. Furthermore, for web applications, many other limitations have to be considered, such as the capacity of internet bandwidth, and the performance of internet browsers. Therefore, it is of great importance to substantially improve the performance of Web3D applications, both in running time and computer storage space required.

In this work, we make effort to deal with large-scale BIM scenes. The source data format adopted is the mainstream BIM exchange format - Industry Foundation Classes (IFC). More specifically, we focus on Web3D based real-time visualization of large-scale BIM scenes, including semantics analysis, data lightweighting, indexing, and scene management.

1.1. Preliminary on IFC

IFC is a standard for open data format specifically for the construction industry. IFC aims at sharing and exchanging of building information throughout a building's entire lifecycle [3, 4].

IFC employs EXPRESS, an object-oriented language, to predefine a set of entity classes and their native data types which are used to define corresponding entity instances. For example, in the following definition of IfcApplication instance:

#6=IFCAPPLICATION(#5, '15', 'Pro2015', '2015');

"#6" is the instance name and its internal member is initialized by the brackets of four parameters, "#5, '15', 'Pro2015', and '2015' ", separated by commas. The first parameter "#5" here is another instance referenced by #6 and it is a primary way to describe relations between entities.

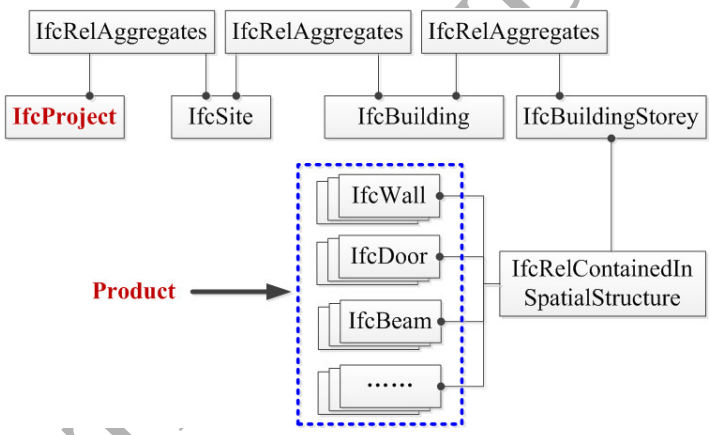


Figure 2: IFC file's structure

In addition, IFC standard also contains hundreds of entity classes or data types. E.g., in the IFC2X3 version, more than 600 entity classes and 300 native data types are predefined. For convenience, these classes and types are classified into 4 layers which, in descending order, are the domain layer, the interoperability layer, the core layer, and the resource layer. By the rule of reference, no layer can reference the layers above it.

When using IFC files, not only the regular entities, but also the relations between them are considered. E.g., an IFC file is usually structured from an IfcProject root instance which is then decomposed into many products, such as IfcWall, IfcBeam, etc., by several reference layers (see Figure 2). The

geometry data of an IFC file is actually organized under these products (see Figure 3).

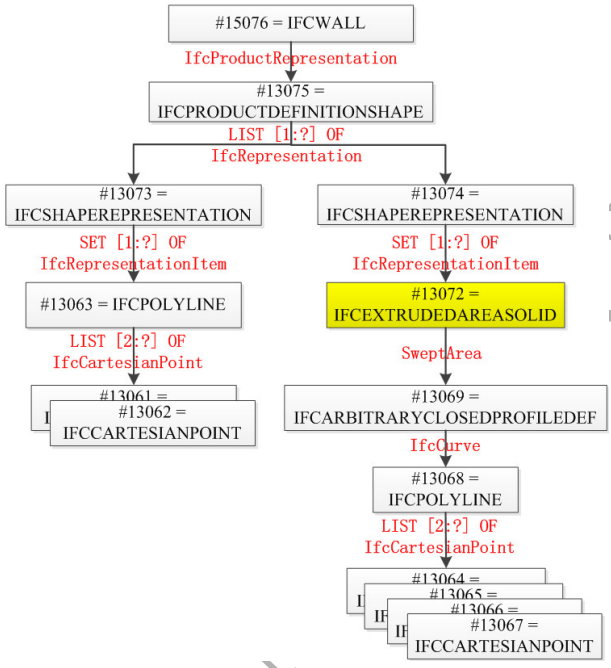


Figure 3: An IfcWall instance

1.2. Challenges brought by IFC

We observe that the IFC standard covers a wide range of building information and provides a rich set of independent schemes for data exchange and sharing. However, it is also these characteristics that bring on the following challenges, among others, for online visualization:

(1) Complex logical structure. IFC contains more than 900 predefined entities and native data types. The reference needs to follow the strict hierarchical rules. In other words, the deeper a reference level is, more complex its logic structure will be. As illustrated in Figure 3, the product IfcWall needs to travel through a tree in 7 levels in order to reach the geometry data.

(2) Large amount of data with high redundancy rate. Because of the intended generality of IFC, the parsed model data in real-life projects are usually much larger than the example of the subway station in Figure 1. Furthermore, according to the reference rules, a large number of entities of

intermediate layers are always needed, which makes the model data often exceedingly huge with an extremely high redundancy rate. Besides, as depicted by the colored node in Figure 3, IFC models have a good parametric feature which can produce more redundancies after parsing.

(3) Loose scene organization. In an IFC scene, the geometry data is represented by product entities, which may have relations with each other such as adjacency and inclusion. However, these relations almost exclusively focus only on building construction processes, making them incapable of meeting the requirement of scene organization and management for real-time Web3D scene visualization.

1.3. Our work and contributions

To meet the above three challenges, we propose a framework for Web3D-based online visualization of large-scale BIM scenes from the perspective of practical feasibility. Our contributions are as follows.

(1) Effective lightweighting for large-scale BIM scenes. BIM technology is important, but hardware requirement for managing BIM data is far beyond the configuration of a personal computer, which brings a formidable obstacle for sharing the data. To our best knowledge, there is no reported work in this area because of the short history of BIM technology. Our work in this paper on lightweighting BIM data is a conscious effort to fill this gap.

(2) Automated lightweighting of BIM scenes. We first locate the entities by semantics analysis according to the semantic content of the BIM data. Then, a geometry confirmation is conducted with these entities. Compared with pure geometric approaches, our approach makes it possible to automatically and quickly lightweight large-scale BIM scenes.

(3) Introduction of occlusion into scene index generation. Occlusion is an important feature in buildings. It has yet not been integrated with data indexing in building data management. In this paper, we first separate buildings into outdoor and indoor parts. Then a Double-Layered Sparse Voxel (DLSV) is generated with this separation for efficient data indexing.

2. Related works

Although the IFC standard has been revised several times, further improvement keeps going on. Until now, the latest version is IFC4 [4], which was issued in December 2013. With the template file and profile material in

the current version, it becomes more convenient for construction of intelligent buildings.

2.1. Visualization tools

3D visualization has become a fundamental task of building web applications. There have been a number of outstanding research works in both application and theory. Application commercial software tools such as Autodesk Revit, Bentley Architecture, Graphisoft ArchiCAD, and Nemetschek ALLPLAN have existed on market for several years. With these tools, operations such as BIM creation, editing, and visualization can be effectively performed. However, resource occupancy is also very large during these operations.

There are many kinds of BIM data formats, and each of them needs a parsing step when being used. This is not convenient for data sharing. IFC standard, as a mainstream exchange format, is supported by almost all BIM tools. Meanwhile, its parsing library BIMServer [5] is also integrated by many tools, in which the raw BIM data is parsed into many IFC products (see Figure 2) and then stored into a database.

There are also some simple commercial tools which are dedicated to viewing IFC files, such as the stand-alone tools FZK Viewer [6], BIM Vision [7], etc., and the WebGL-based tools such as BIM Surfer [8], BIMviews [9], etc.. Meanwhile, mainly from theoretical point of view, Chi et al. [10], Pasewaldt et al. [11] and Karsch et al. [12] respectively studied the problem of augmented reality application integration, the multi-view display, and combining photographs and building models in construction visualization.

2.2. BIM data lightweighting

For the optimization of 3D building models, Solibri Solution Center has provided a related tool Solibri IFC Optimizer (SIO) by which the IFC file could be optimized with certain degree of redundancy removal and data compression [13]. Sun et al. [14] also presented a comparable content-based compression algorithm, named IFCCompressor.

The following methods nevertheless mainly focus on IFC file similarities and discuss this topic from multiple aspects, such as metrics for quantifying the similarities and differences between IFC files, IFC file retrieval, and semantic query language [15, 16]. In particular, many scholars introduced the semantic information into the model retrieval process. Some basis experiences are also reported on semantics-based IFC retrieval and indirect

methods on reused/similarity data detection [17, 18]. However, their solutions mainly rely on semantics information. Actually, geometry processing methods can also be used in this area. E.g., Laga et al. [19] and Zheng et al. [20] combined the geometry model and semantics information via introducing some semantics rules in geometry recognition.

2.3. Scene management

In scene organization and management, the existing works mainly focus on the visibility culling [21, 22], the area of interest [23], the scene graph [24, 25], and the index structure [26, 27]. By taking account of the characteristics of BIM scenes, Varduhn et al. [28] employed the Octree and Level of Detail (LOD) methodology to process BIM data in real-time. Kang et al. [29] presented a database, the Object-relational Database (ORDB), to substitute the traditional relational database to manage the complex IFC scenes.

To summarize, although there are many works related to BIM/IFC, to the best of our knowledge, few are on real-time Web3D visualization of complex building scenes. The existing works are mostly for stand-alone or interactive applications. There is still a large void in the efficient visualization of large BIM data via a web browser without any plugins.

3. Overview of our work

In this work, according to the features of BIM/IFC data, we propose a pipeline for real-time visualization of large-scale BIM scenes, which mainly includes three parts as illustrated in Figure 4: the semantics-guided lightweighting, the scene index structure generation, and the scene management. The first part is a lightweighting operation of the raw BIM scenes; the second part generates a data index structure for efficient data accessing, while the third is a new scene management strategy that integrates the first two parts. In our implementation, the first two parts are of offline preprocessing type, while the third is real-time online operation. In more details, the following are the main components of our work/system.

(1) **Semantics-guided lightweighting.** In the existing IFC applications, the IFC parsing and rendering are usually mixed together. While utilizing them, the problem of resource limitation in web applications is always overlooked. Nevertheless, for Web3D applications, reducing the data amount is imperative because of the resource limitation. Our semantics-guided lightweighting enables us to reduce the amount of data processing

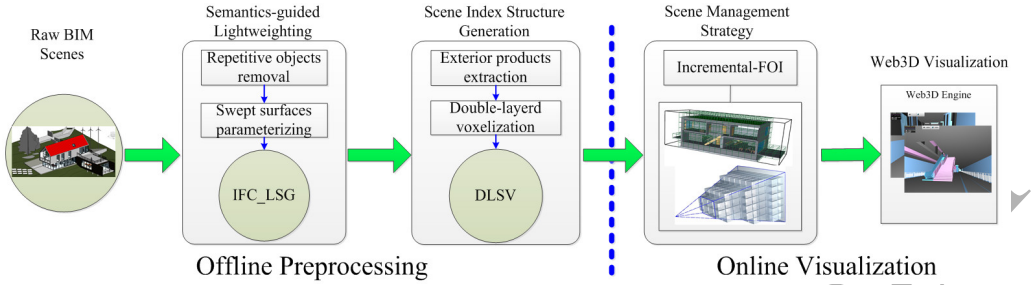


Figure 4: Pipeline for real-time Web3D visualization of large-scale BIM scenes

in the front end by removing the redundant data and creating an IFC Lightweight Scene Graph (IFC_LSG). (Section 4)

(2) **Scene index structure generation.** IFC data covers the building information in many aspects. It focuses on the entire lifecycle of a building's construction. However, it does not take into account the requirement by the applications such as the building visualization in real-time. In order to improve the access efficiency of real-time Web3D building visualization, we propose a Double-Layered Sparse Voxel (DLSV) structure for data indexing. (Section 5)

(3) **Scene management strategy.** A desirable scene management strategy should incorporate with the above two processes to maximize their utilization. We propose a strategy named Incremental Frustum of Interest (I-FOI) by combining the rendering pipeline with the current scene index. (Section 6)

4. Semantics-guided lightweighting

In IFC, the model contains both geometric and syntax data. The former is represented as a parametric representation of points and curves, while the latter encompasses the semantic content such as the text representation of the products and their relations. In practice, the geometry data is more often used under the semantics' guidance. In order to obtain a higher efficiency, we take both semantics and geometry into account and present two operations: repetitive objects removal and swept surfaces parameterizing guided by semantics information.

4.1. Repetitive objects removal

With the semantics information, repetitive objects removal is carried out in three steps: (1) finding out the unique representation items set to cover those multiple IFC products; (2) verifying the results of (1) through geometrical verification and computation of the transformation matrix; and (3) removing the redundant data and constructing the IFC Lightweight Scene Graph (IFC_LSG).

For geometric verification, we extend the ICP-based alignment procedure as in [19] and use the Hausdorff Distance (HD) to define the similarity as given in Eq. (1). We consider two representation items a and b to be repetitive of each other if their similarity is within a threshold (0.1 in our setting) and then the transform matrix between them is recorded.

$$Similarity(a,b) = \frac{HD(a,b)^2}{max(\|a-b\|)^2}, \tag{1}$$

where $HD(a,b)$ can be formulated as

$$HD(a,b) = \max\left\{ \sup_{\alpha\in P(a)} \inf_{\beta\in P(b)} \|\alpha-\beta\|, \sup_{\beta\in P(b)} \inf_{\alpha\in P(a)} \|\alpha-\beta\| \right\}, \tag{2}$$

and $P(a)$, $P(b)$ are two sets of 3D points from representation items a and b respectively.

For checking the similarity between products, e.g., A and B, we first employ their representation items to construct a weighted average similarity as given in Eq. (3), and then consider the two to be repetitive if $Similarity(A,B) > S_{avg}(A,B)$. Meanwhile, the transform matrix between A and B is recorded by the ICP method,

$$S_{avg}(A,B) = \frac{\sum_{i=1}^{N} Area(a_i+b_i) Similarity(a_i,b_i)}{\sum_{i=1}^{N} Area(a_i+b_i)} \tag{3}$$

where a_i and b_i are respectively the representation items of products A and B and N is the number of items obtained from parsing the internal member "Representation" of the corresponding IfcProduct object.

In detail, the procedure of repetitive objects removal is outlined in Algorithm 1 and Figure 5 with the following structure definition.

struct ProductInstance

{

```
    string    strInstanceName;
    string    strEntityName;
    set < RepresentationItem >   sItem;
    set < string >     sMappingSource;
    ......
}
```

Algorithm 1: Repetitive objects removal

```
Input  : raw IFC data files
Output: an IFC_LSG file and unique instances
Parse IFC scene files into products set S;
Classify elements in S by entity names into a cluster set CS;
for each cluster c in CS do
    for each product p in c do
        //Semantics-based product segmentation
        for each element i in p.sItem do
            if (EntityName( i ) == "IfcMappedItem") then
                Record i and i's mapping source of p;
            end
        end
        //Semantics-based items removal
        Sort p.sItem by p.sMappingSource's value;
        Remove and record the mapped items with the same mapping
        source to collect the unique items of p in a list;
        //Unique item list sorting
        Sort the unique item list of p by their instance ID;
    end
    //Semantics-based product removal
    Compare the products in c by their unique items list;
    Remove and record the products with the same unique items list
    to get a unique product list of c;
    Confirm the correspondence with a geometric method;
    Update IFC_LSG file and the unique instances;
end
```

As prescribed in Algorithm 1, after an entity name based classification, we primarily query and analyze the entity referenced information on

IfcRepresentationItem and IfcRepresentationMap, and the mapping source of IfcMappedItem etc. in terms of the semantic processing as illustrated in Figure 5. And finally the unique instances and the IFC_LSG (see Figure 6) are returned.

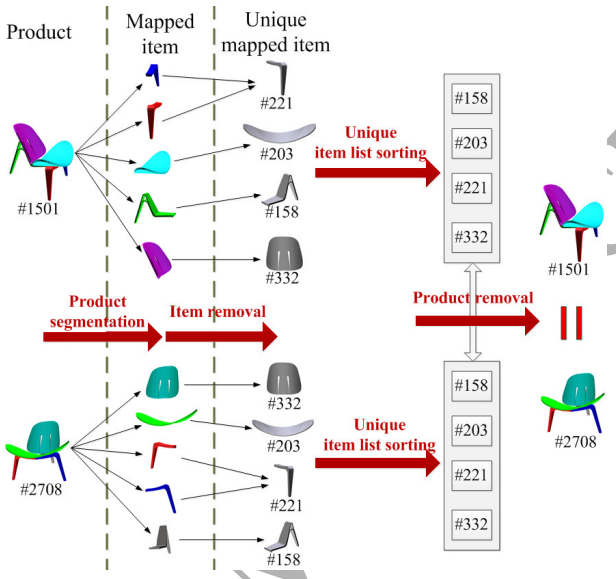


Figure 5: Semantics-based repetitive objects removal

4.2. Swept surfaces parameterizing

Many products in IFC are generated from extrusion/sweeping operations which are primary expressed by entities IfcExtrudedAreaSolid, IfcSweptAreaSolid, IfcSweptDiskSolid and IfcSweptSurface. The traditional parsing methods, although they can reduce the number of these entities through converting them to geometry products, they also simultaneously introduce more geometry data by the extrusion/sweeping operation. The swept surface parameterizing mainly focuses on the lightweighting for these objects.

The semantic information is analyzed firstly to figure out the sweeping type. Then, the corresponding parameter is computed according to the sweeping type by a geometry fitting. For instance, the product IfcBeam is parameterized as illustrated in Figure 7.

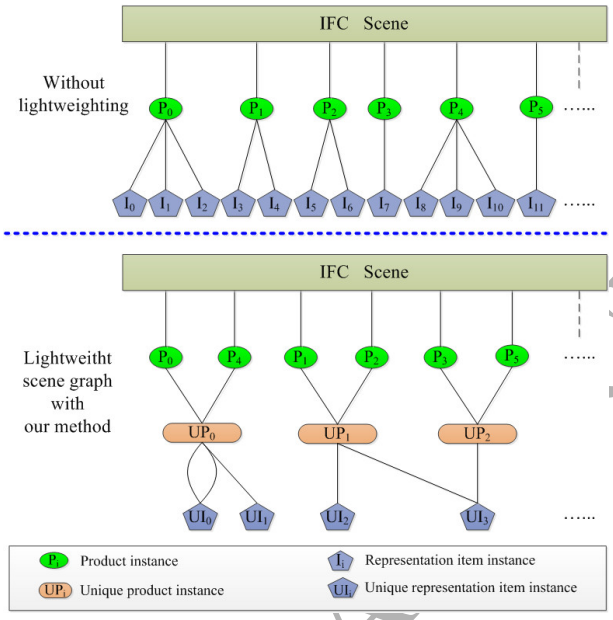


Figure 6: IFC LSG structure

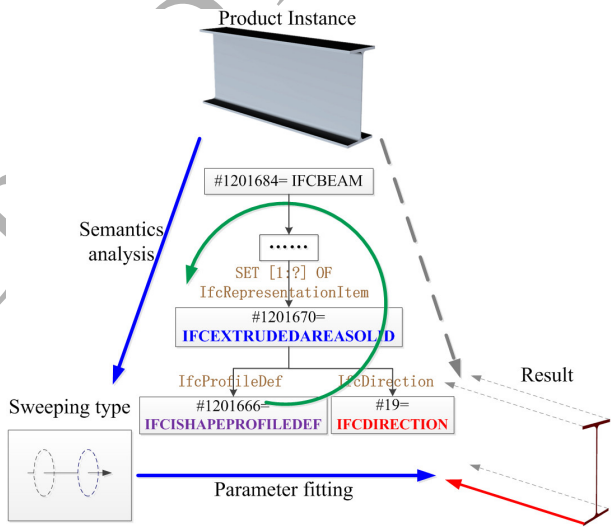


Figure 7: Semantics-guided swept surface parameterizing example

5. Scene index structure generation

Visibility culling is a vital issue in the visualization of complex building scenes. However, because of their rich content, it is hard to cull the indoor or outdoor scenes automatically from a raw IFC data. To achieve an efficient solution, two phases - the exterior products extraction and the scene index generation - are proposed, as described next.

5.1. Exterior products extraction

When roaming in an outdoor scene, loading the interior data of the indoor scene not only reduces the roaming speed, but also drags down the performance of data transmission, rendering, and collision detection. Scene understanding helps to separate the interior and exterior data efficiently.

Recently, the function of IFC exterior data extraction has been added to CityGML format [3]. We propose an exterior products extraction method to separate the exterior products from the building based on visual perception, through a multi-view based projection analysis [30].

We make an assumption that the exterior parts of any IFC building are all those products that can be seen from outdoor. We sample points on the bounding ellipsoid of the building as the viewpoints. Standing on these viewpoints, we observe towards the centroid of the bounding ellipsoid and collect the visible products as the exterior products of the building. In our experiments, we observe that the number of the products, which increases with the number of the viewpoints, will be converged at about 14 viewpoints. Therefore, we setup the number of the viewpoints to 14 in our following experiments as illustrated in Figure 8. In more details, all the viewpoints firstly participate in this exterior extraction. For each viewpoint, we find out its panel which tangents with the ellipsoid and tag its visible products as exterior products. Then, we sum up all thus tagged exterior products from all the viewpoints as the building's exterior products set.

In our implementation, we apply the voxelization result of the building instead of the raw building data in order to improve the computation efficiency, as prescribed in Algorithm 2 and Figure 9. Figure 10 is a 2D diagram of this projection.

5.2. DLSV-based structure generation

Scene indexing is also one of the most important factors that determine the data access performance [26]. Terry J. et al. [27] optimized the space

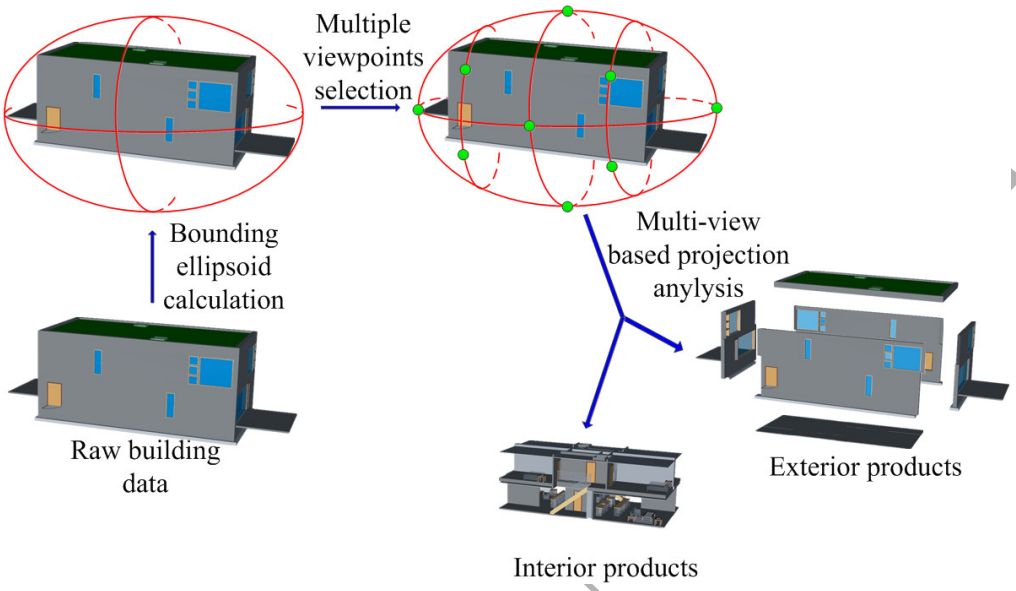


Figure 8: Workflow of the exterior products extraction

Algorithm 2: Exterior products extraction

Input : raw IFC building data
Output: the building's exterior products set
Parse all products from the raw IFC building data;
Get the minimum value f from all the widths of walls and the heights of floors;
Statistics the bounding ellipsoid of the building, set $f/2$ as the voxel size and voxelize the building's products into a voxel set S;
Relate the products to voxels by their positions;
Calculate viewpoints based on the bounding ellipsoid;
for *every viewpoint* v_i **do**
    Project the elements of S with v_i and record the nearest voxel;
end
Collect all the unique voxels from projection result into set S' ;
Get all the products related with voxels of S' as exterior products;
Sort and remove all duplicate exterior products;
Update the raw building products;

filling curves for efficient generation of a unique index. In order to reduce the size of the description file, we propose the fast Double-Layered Sparse Voxel (DLSV) index method based on sparse voxelization [31].

Different from the pure geometry model voxelization in the existing methods, our DLSV strives to efficiently build the scene index without any loss of scene details. Given the interior and exterior separation of the building (from subsection 5.1), we propose a hierarchical product-based indexing strategy, which facilitates the indoor and outdoor dynamic switching. The process includes the following four steps:

(1) Compute the bounding box T of the entire scene and the average bounding boxes of all the buildings in the scene, B_{avg};
(2) Estimate the resolution R_i in the voxelization procedure according to a user input N which indicates the average number of buildings to be indexed in each voxel:

$$R_i = \begin{cases} 1, & L_i = 0 \\ \left\lfloor L_i \sqrt[3]{\frac{\lambda M}{N \times B_{avg}}} + 0{,}5 \right\rfloor, & L_i \neq 0 \end{cases} \tag{4}$$

where L_i is the length of T on the coordinate axes (X, Y, and Z), λ the adjustment parameter (5 in our experiments), and M the number of buildings in T.
(3) Apply the sparse voxelization method [31] to generate the first depth level voxels for indexing each buildings' exterior products.
(4) Set each building as the scene and its interior products as the buildings. Then repeat (1) to (3) steps to generate the second level voxels for indexing the interior products.

As described in the above steps, the DLSV structure can be treated as a hierarchical tree structure with height 3 (see Figure 11). With the entire scene as the root node, the second level (which indexes the buildings' exterior products), is the outdoor index and the third level corresponds to the indoor scenes. With this structure, the switch between indoor and outdoor can be easily implemented by conversion between the parent and child node on the two levels. When the viewpoint is out of the building, all the interior products in the third level can be removed. When the viewpoint moves inside the building, only the current building data should be processed and the others (at the second level) can be culled away directly so as to reduce the

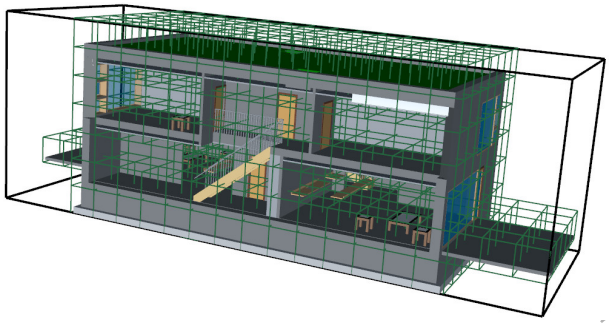


Figure 9: 3D Sparse voxelization of building

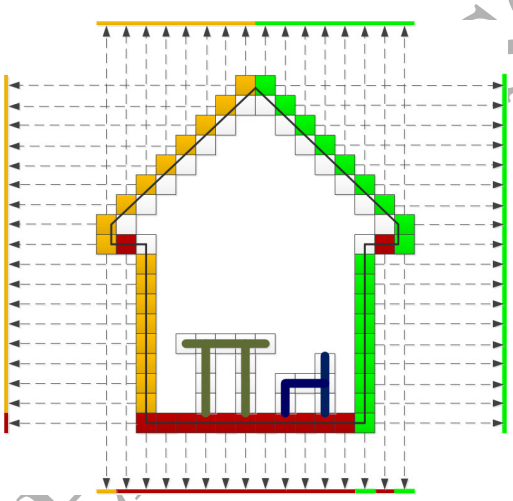


Figure 10: Voxelization and projection (2D profile in XOZ coordinates) based exterior extraction algorithm

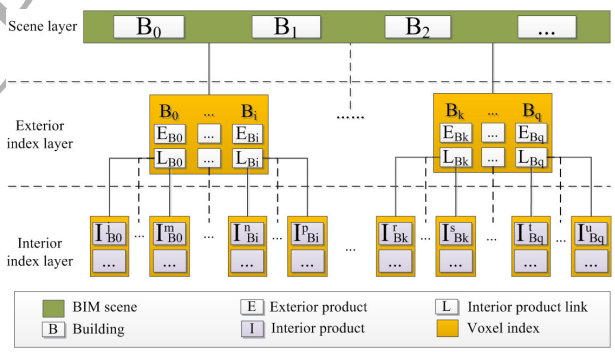


Figure 11: DLSV structure

workload on the network and system resources. In concrete implementation, we can even mark some products, e.g., IfcDoor, as "Portal" in advance.

In order to get the product index efficiently, a map structure is generated directly. The key value of this map is set with a Morton code [31] which is calculated by the three coordinates of the index voxel for simplifying the procedure of [27]. In addition, the index schemes of the second and third level are almost the same (see Figure 11), which makes the maintenance more convenient.

6. Scene management strategy

Considering frustum culling and scene transmission together with our scene index, we now present a new scene management strategy called Incremental Frustum of Interest (I-FOI) to efficiently control the quantity of the loaded data so as to speed up the roaming in real-time.

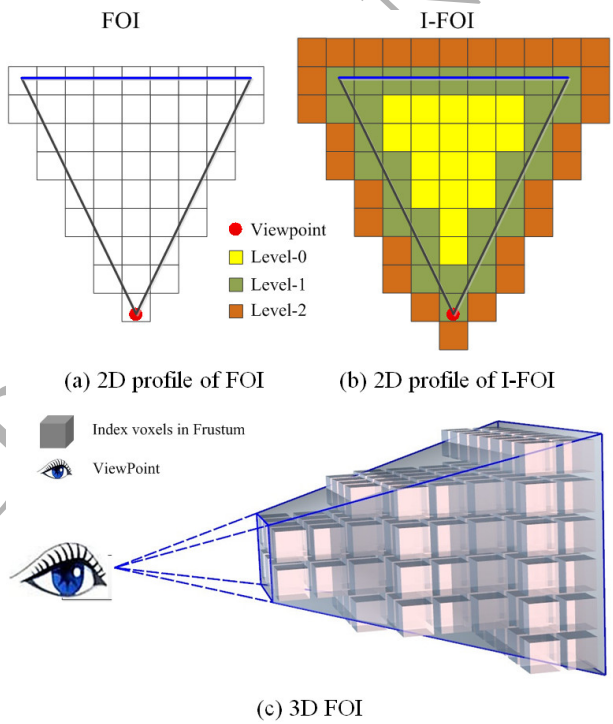
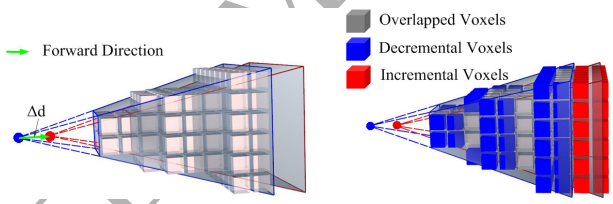


Figure 12: FOI and I-FOI

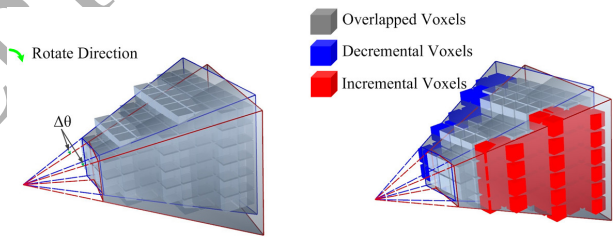
To elaborate, we first define Frustum of Interest (FOI, see Figure 12(a) and (c)) as:

$$FOI = \{\ index\ voxel\ v \mid v \in frustum\ of\ viewpoint\ \} \tag{5}$$

I-FOI is an extension to FOI with an additional voxel bounding shell of FOI for the sake of progressive updating of FOI. As shown in Figure 12(b), I-FOI contains three levels, two of which are in FOI and the third is an extension level. At the very beginning, I-FOI is initialized by quick selection and then indexed on the voxels at each level. Afterward, only the incremented voxels are loaded, as is illustrated in Figure 13. During the roaming, the regions at level-0 and level-1 need to be rendered. In detail, as illustrated in Figure 12, Level-0 represents the overlapped voxels, where the rendering is processed no matter whether it is a forward or backward movement. Level-1 maintains the updates between the visible and invisible parts. Level-2 comprises those potential visible parts at the next time slot. It is the core structure for supporting incremental feature in order to guarantee parallel processing on both the current scene rendering and the potential scene data loading.



(a) Incremental FOI of translation



(b) Incremental FOI of rotation

Figure 13: Incremental FOI

Figure 14 depicts the workflow of I-FOI progressive scheduling.

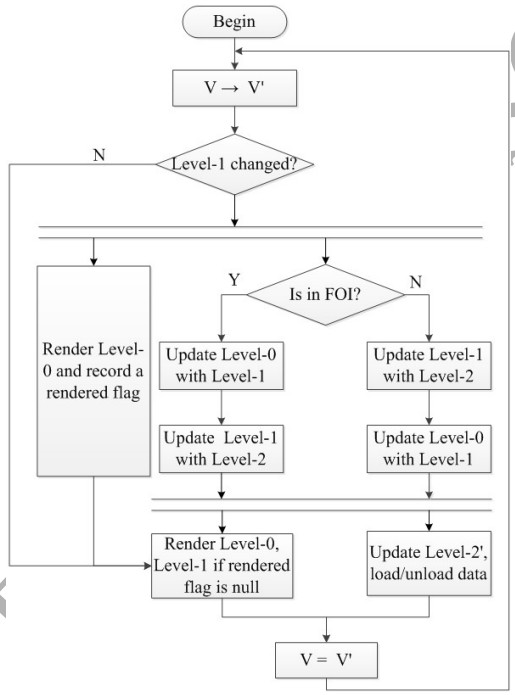


Figure 14: Workflow of I-FOI progressive scheduling

Meanwhile, observing that the change of I-FOI will not be significant if the viewpoint moves only slightly, we also mark the bounding voxels in I-FOI, which are further utilized to estimate the potential scene data and drive the next data transmission and rendering. Then we update I-FOI progressively with a few incremental and decremental voxels, as shown in Figure 13.

In order to avoid frequent change on data loading and releasing which causes hovering nearby, we introduce a buffer distance D:

$$D = v * (t + \Delta t) \tag{6}$$

where v is the speed of navigating, t the elapsed time within the data loading, and Δt the time delay.

To avoid heavy computation due to collision detection on fine models in the scene, we instead apply collision detection on the voxel index structure so as to further speed up the process. Usually, in practice, the nearest object is first to be observed. We implement this by evaluating a priority of approximation between the index voxels and the loaded data with the priority w formulated as:

$$w = \mid v.x - d.x \mid + \mid v.y - d.y \mid + \mid v.z - d.z \mid \tag{7}$$

where v and d are respectively the voxel position of the viewpoint and the data's index, and x, y, z are the coordinates of the position.

7. Experimental results and discussion

To evaluate our method, we ran experiments on a simulation system. We collected in total 119 raw models in IFC2X3 format as the test dataset, of which 85 models (14 buildings and 71 others) are chosen from the public dataset Open IFC Model Repository [32] and the rest 34 building are from our own dataset. Their file sizes range from 0.006MB to 429.33MB. Some of them are shown in Figure 15.

We implemented the proposed semantics-guided lightweighting and the scene index structure generation algorithms in C++ language, and the scene management algorithm on a Flash3D engine (named Away3D) in the action script 3.0 language. The configuration of the computer used for the test is: Intel i5-3210M Processor, 8.0 GB RAM, NVIDIA GeForce 610M and Windows 8 (64 bit). The experiment analysis is divided into the following three parts.

Table 1: The 13 tested models

Model ID	Name	Size (M)
1	171210frame_physical.ifc	0.006
2	091210ISO9705revit9-c.ifc	0.041
3	171210etabs_physical.ifc	0.393
4	Duplex_A_20110505.ifc	2.311
5	301110FZK-Haus-EliteCAD.ifc	7.191
6	301110FJK-Project-Final.ifc	14.275
7	0912102010-03-01 Project.ifc	50.726
8	161210Med_Dent_Clinic_Combined.ifc	109.996
9	DN_Shnz_Project.ifc	226.186
10	Zj-Si-CZ-Prj-1.ifc	270.093
11	Zg_mobileRoom-1.ifc	369.151
12	Ch-Subway-130215.ifc	384.438
13	Cgm-Project-12.ifc	429.433

I. Preprocessing performance on single IFC file

We sampled 13 models from our test dataset across the entire size range, which are listed in Table 1 in ascending order of their sizes. Then we separately conducted our preprocessing operation and the existing methods on these 13 models.

Figure 16(a) plots the results of the model lightweighting. We observe that a much higher lightweighting rate is achieved while using our method in comparison with the existing methods (SIO [13], IFCCompressor [14]). Note that, in the case of very large scenes (IDs from 11 to 13 in Table 1), the SIO tool and IFCCompressor cannot handle them anymore, because the data size is out of their upper bound, but our method successfully processed them with a reduction rate around 60%.

For exterior extraction, the mainstream methods mainly focus on converting the exterior into B-rep models in CityGML format. But in our method, the target is visualization of a real scene in real-time and our extracted exterior data are purely assembled from the original models without data loss. Therefore, we evaluate the performance between the manual extraction and our proposed algorithm by measuring their exterior and volume difference as shown in Figure 16(b). The measurement of volume difference is:

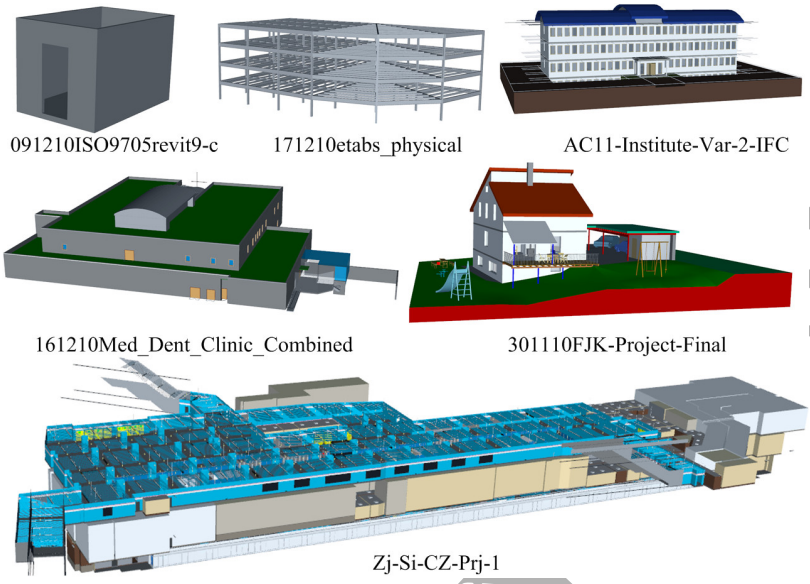


Figure 15: Images of models from our test dataset

$$m = \frac{\sum Volume\,(Difference(EP_A, EP_M))}{\sum Volume(EP_M)}, \tag{8}$$

where EP_M is the manually extracted set and EP_A is the result by our proposed algorithm.

The average running time in each step is shown in Table 2.

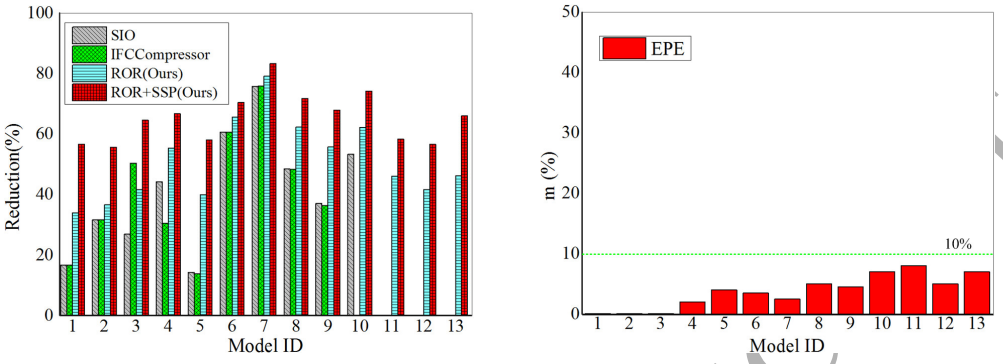
II. Visualization performance on single IFC file

We tested the models from Table 1 in our system and the existing webGL tool BIMviews [9]. During the test we found that BIMviews issued a "Java Heap Space" exception and broke down when the input data's size exceeded 140MB, while our program still ran stably. Moreover, our program was able to reduce the raw data size to around 45% and hold the FPS close to 30.

III. Performance analysis on overall buildings

To evaluate our whole method, we ran it on a total of 48 buildings (14 from [32] and 34 from our own collection) whose total size amounts to 2904.767MB. The average building number N is 5 for the index generation, 6 voxels' size for the FOI's radius, and the maximum FPS is 30. The test results are plotted in Figure 17.

Figure 17 shows our analysis of test data with the following findings:



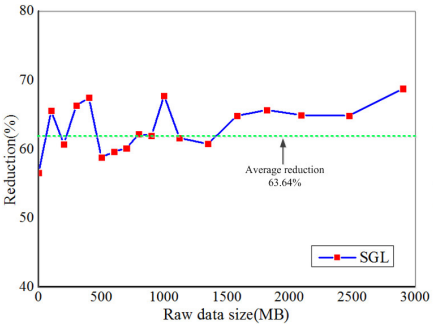
(a) (b)

Figure 16: Results comparison: (a) Comparison between the SIO method, the IFCCompressor method, our Repetitive Objects Removal (ROR) method, and our Swept Surfaces Parameterizing (SSP) method. (b) Comparison between manual extraction method and our Exterior Products Extraction (EPE) method.

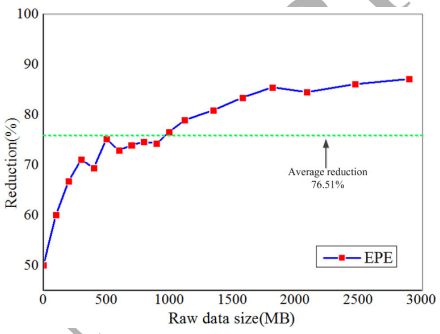
Table 2: Results of average processing time

Model ID	Lightweighting (s)		Scene index generation (s)	
	ROR	SSP	EPE	DSG
1	0.2	0.19	0.16	0.12
2	0.23	1.91	0.50	0.15
3	0.52	3.16	0.95	0.34
4	2.36	5.84	1.53	0.78
5	1.013	5.46	2.03	1.17
6	1.353	3.28	1.63	0.83
7	3.852	7.86	1.14	0.56
8	114.39	76.45	18.6	3.05
9	506.39	331.74	10.58	6.57
10	190.45	127.17	17.93	14.26
11	29.98	26.95	15.86	8.431
12	18.51	18.36	6.5	5.49
13	519.21	342.01	32.60	26.39

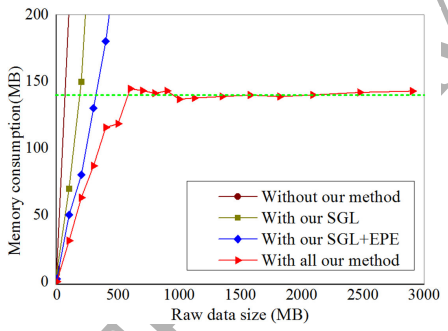
ROR:repetitive objects removal **SSP:**swept surfaces parameterizing
EPE:exterior products extraction **DSG:**DLSV-based structure generation



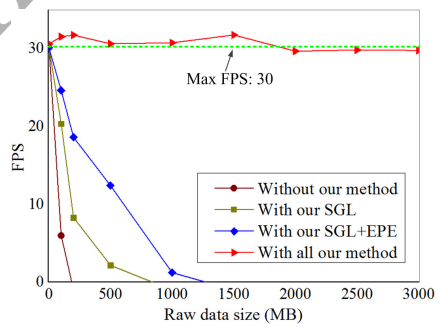
(a) Semantics-guided Lightweighting (SGL)



(b) Exterior Products Extraction (EPE)



(c) Memory consumption (for outdoor)



(d) FPS

Figure 17: Performance analysis results

(1) Although the performance of lightweighting in Figure 17(a) is a little wobbly for small size scenes, it achieves a steady reduction rate of 64% when the size of the scene reaches a certain level. This actually manifests a desired property of our lightweighting algorithm: scenes of large sizes are usually in more need to be lightweighted and our algorithm meets this demand.

(2) The reduction rate in exterior products extraction increases with the scene scale (Figure 17(b)). This indicates that the more complex a building (scene), the lower percentage of its exterior occupied, which agrees with our common observation in life.

(3) In the test of scene management (see Figure 17(c)), with the steps of our methods, the memory consumption rises more slowly, but eventually it exceeds the maximum memory without the entire pipeline. With the implementation of our entire pipeline, the consumption becomes stabilized and holds at a constant rate when the scene scale is beyond the frustum of viewpoint. This implies that, although each step of our method enables to reduce the cost of memory, large scale online roaming can be achieved by optimizing the entire pipeline. Certainly, this memory consumption is more dependent on the frustum size in large-scale scenes. The FPS in Figure 17(d) also confirms our conclusion.

Figure 18 shows some snapshots while roaming in a large-scale scene from the experimental dataset with our method.

8. Conclusion and future work

We have presented a prototype of lightweighting and real-time Web3D visualization system for large-scale BIM scenes, striving to increase the data accessing rate and achieve better data management. First, we propose a semantics-guided lightweighting process that combines the semantics with the geometric information in order to reduce the amount of raw BIM data. Second, we extract the exterior products of a building scene by using multiple viewpoints projection and removing the interior products data. Finally, we generate the DLSV index structure based on the sparse voxelization idea towards the frustum of interests. We have tested the proposed Web3D BIM system with simulations of virtual 3D large-scale scene roaming. The experimental results show that the proposed system makes it possible to significantly reduce the data redundancy and cull the indoor or outdoor data.

Nevertheless, we consider this work to be only a preliminary investigation and there are certain limitations to the proposed method. First, the method

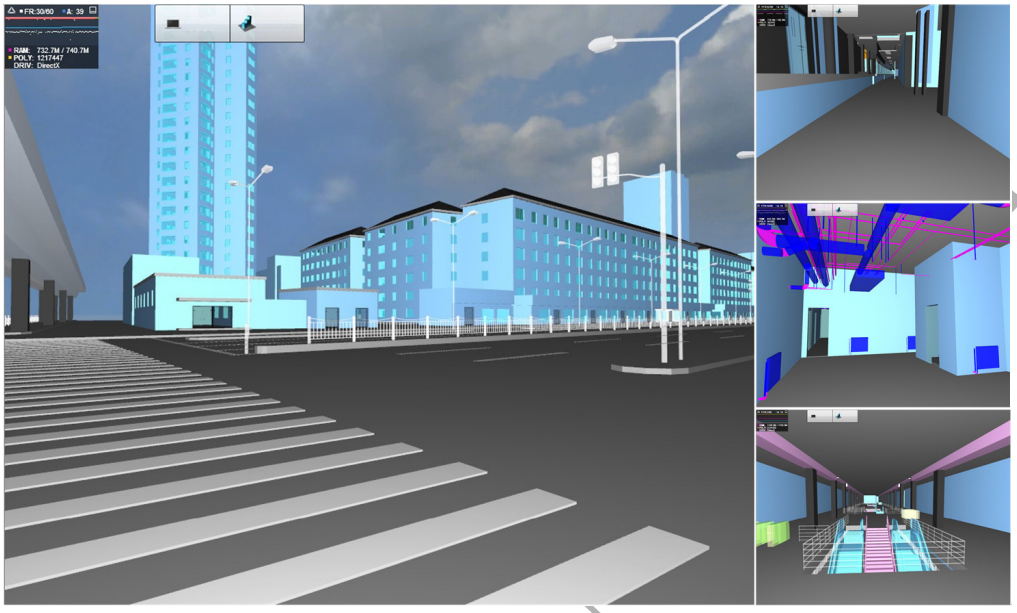


Figure 18: Snapshots from the experiments of our prototype system

for extracting exterior products is suitable only for scenes with occlusion structures like buildings, and the identification accuracy still needs to be improved. Secondly, since we mainly focus on reducing the redundancies of the raw data, the data transmission has not been integrated in our system, and the rendering effects like shadows etc. have not been considered. In the future, we plan to enhance our method tending to these aspects and integrate with some new ideas and technologies such as WebRTC [33].

Finally, we hope to make some contributions in collecting buildings models in the current dataset to make them more comprehensive.

Acknowledgments

The authors appreciate the comments and suggestions of all anonymous reviewers, whose comments significantly improved this paper.

This work is supported by the Fundamental Research Funds for the Central Universities, National Technological Support Program for the 12th-Five-years Plan of China (No.2012BAC11B00-04-03), The Research Fund for the Doctoral Program of Higher Education of China (No.2013007211-0035), Chinese Changbai Valley Talent Plan of Changchun National Hi-Tech Industrial

Development Zone (No.3-2013006), and Key project in scientific and technological of Jilin province in China (No.20140204088GX), Tongji University Young Scholar Plan of China (No.2014KJ074).

References

[1] A. V. M. Nina Valkanova, Sergi Jorda, Public visualization displays of citizen data: design, impact and implications, International Journal of Human-Computer Studies 81 (2015) 4–16. doi:10.1016/j.ijhcs.2015.02.005.

[2] R. d. L. P. B. Jakob Beetz, Leon van Berlo, Advances in the development and application of an open source model server for building information, in: Proceedings of the 28th International Conference of CIB W78, Sophia Antipolis, France, 2011, pp. 1–7.

[3] S. Donkers, Automatic generation of citygml lod3 building models from ifc models, Ph.D. thesis, TU Delft, Delft University of Technology (2013).

[4] T. Liebich, Ifc office release 4, model Support Group (MSG) of buildingSMART (2013).

[5] BIM Server, http://www.bimserver.org, [Online; accessed 24-July-2015] (2015).

[6] FZK Viewer, http://www.iai.fzk.de/www-extern/index.php, [Online; accessed 24-July-2015] (2015).

[7] BIM Vision, http://www.bimvision.eu/home/, [Online; accessed 24-July-2015] (2015).

[8] BIM Surfer, http://bimsurfer.org/, [Online; accessed 24-July-2015] (2015).

[9] BIM views, http://www.bimview.fr/, [Online; accessed 24-July-2015] (2015).

[10] X. W. Hung-Lin Chi, Shih-Chung Kang, Research trends and opportunities of augmented reality applications in architecture, engineering, and construction, Automation in construction 33 (2013) 116–122. doi:10.1016/j.autcon.2012.12.017.

[11] J. D. Sebastian Pasewaldt, Matthias Trapp, Multi-perspective detail+overview visualization for 3d building exploration, in: Proceedings of 11th Theory and Practice of Computer Graphics, 2013, pp. 57–64. doi:10.2312/LocalChapterEvents.TPCG.TPCG13.057-064.

[12] D. F. Kevin Karsch, Mani Golparvar-Fard, Constructaide: analyzing and visualizing construction sites through photographs and building models, ACM Transactions on Graphics (TOG) 33 (6) (2014) 176. doi:10.1145/2661229.2661256.

[13] Solibri IFC Optimizer, http://www.solibri.com/, [Online; accessed 24-July-2015] (2015).

[14] G. G. X.-G. H. Jing Sun, Yu-Shen Liu, IFCCompressor: A content-based compression algorithm for optimizing Industry Foundation Classes files, Automation in Construction 50 (2015) 1–15. doi:10.1016/j.autcon.2014.10.015.

[15] J. L. G. Arthaud, Automatic semantic comparison of step product models, in: Innovations in Design & Decision Support Systems in Architecture and Urban Planning, Springer Netherlands, 2006, pp. 447–463. doi:10.1007/978-1-4020-5060-2_29.

[16] S. H. Y. S. Ghang Lee, Jongsung Won, Metrics for quantifying the similarities and differences between ifc files, Journal of Computing in Civil Engineering 25 (2) (2011) 172–181. doi:10.1061/(ASCE)CP.1943-5487.0000077.

[17] R. R. A. I. Le Zhang, Development of ifc-based construction industry ontology for information retrieval from ifc models, in: Proceedings of the 2011 EG-ICE Workshop, University of Twente, The Netherlands, 2011, pp. 6–8.

[18] M. W. M. G. J.-H. Y. Ge Gao, Yu-Shen Liu, A query expansion method for retrieving online bim resources based on industry foundation classes, Automation in Construction 56 (2015) 14–25. doi:10.1016/j.autcon.2015.04.006.

[19] M. S. Hamid Laga, Michela Mortara, Geometry and context for semantic correspondences and functionality recognition in man-made 3d

shapes, ACM Transactions on Graphics (TOG) 32 (5) (2013) 150. doi:10.1145/2516971.2516975.

[20] M. A. N. J. M. Youyi Zheng, Daniel Cohen-Or, Recurring part arrangements in shape collections, Computer Graphics Forum 33 (2) (2014) 115–124. doi:10.1111/cgf.12309.

[21] C. T. S. F. D. Daniel Cohen-Or, Yiorgos Chrysanthou, A survey of visibility for walkthrough applications, IEEE Transactions on Visualization and Computer Graphics 9 (3) (2003) 412–431. doi:10.1109/TVCG.2003.1207447.

[22] M. W. Oliver Mattausch, Jirí Bittner, CHC++: coherent hierarchical culling revisited, Computer Graphics Forum 27 (2) (2008) 221–230. doi:10.1111/j.1467-8659.2008.01119.x.

[23] M.-F. W. C. F.-C.-X. Z. Y. Y. Jin-Yuan JIA, Wei WANG, Multilayered incremental & scalable sector of interest (missoi) based efficient progressive transmission of large-scale dve scenes, Chinese Journal of Computers 37 (6) (2014) 1324–1334, (In Chinese). doi:10.3724/SP.J.1016.2014.01324.

[24] V. G. K. Q. H. N. J. M. T. F. Tianqiang Liu, Siddhartha Chaudhuri, Creating consistent scene graphs using a probabilistic grammar, ACM Transactions on Graphics (TOG) 33 (6) (2014) 211:1–211:12. doi:10.1145/2661229.2661243.

[25] S. L. Laixiang Wen, Jinyuan Jia, Lpm: lightweight progressive meshes towards smooth transmission of web3d media over internet, Computer Animation and Virtual World(In press).

[26] H. S. Y. L. Weijie GU, Jishui WANG, Research on a hybrid spatial index structure, Journal of Computational Information Systems 7:11 (2011) 3972–3978.

[27] B. S. Justin Terry, Indexing method for multidimensional vector data, Computer Science and Information Systems 10 (3) (2013) 1077–1104. doi:10.2298/CSIS120702022T.

[28] E. R. Vasco VARDUHN, Ralf-Peter MUNDANI, Real time processing of large data sets from built infrastructure, Journal of Systematic, Cybernetics and Informatics 9 (2011) 63–67.

[29] G. L. Hoon-sig Kang, Development of an object-relational ifc server, in: ICCEM/ICCPM, 2009.

[30] T. W. D. C. H. Z. B. C. Yunhai Wang, Minglun Gong, Projective analysis for 3d shape segmentation, ACM Transactions on Graphics (TOG) 32 (6) (2013) 192:1–192:12. doi:10.1145/2508363.2508393.

[31] P. D. Jeroen Baert, Ares Lagae, Out-of-core construction of sparse voxel octrees, Computer Graphics Forum 33 (6) (2014) 220–227. doi:10.1111/cgf.12345.

[32] J. D. Robert Amor, An open repository of ifc data models and analyses to support interoperability deployment, in: Proceedings of the 27th International Conference of CIB W78, Cairo, Egypt, 2010, pp. pp.1–11.

[33] D. C. B. Alan B. Johnston, WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web, Digital Codex LLC, 2012.