applied sciences

MDPI

Article

Industry Foundation Class-Based Building Information Modeling Lightweight Visualization Method for Steel Structures

Zhiguo Sun [1], Chen Wang [2] and Jie Wu [2,*]

[1] CCCC Construction Group Co., Ltd., Beijing 100022, China; llhr1236@163.com
[2] College of Civil Engineering, Tongji University, Shanghai 200092, China; w_c@tongji.edu.cn
* Correspondence: wwujie@tongji.edu.cn

Abstract: The efficient extraction, storage, and visualization of geometric and semantic information is a key foundation for the operation of the building information modeling (BIM) platform. This study aims to develop a lightweight BIM system and optimize the system's performance according to the specific characteristics of steel structures. This study proposes several novel techniques for extracting and decoupling the geometric and semantic information of components from industry foundation class (IFC) files. A redundancy removal approach combining the principal content analysis (PCA) algorithm and the Hausdorff-based comparison algorithm is proposed to identify standardized steel components, and a lightweight visualization method on Web3D for redundant instances is also presented. A loading mechanism of the level of detail (LOD) model based on a mesh simplification algorithm is presented to optimize the display efficiency. The developed system is evaluated by three steel structural models. Using the redundancy removal approach, the number of instances is decreased by 96.46% in less than 30 s and over 30 FPS (frame per second) is kept when rendering. Using the LOD loading mechanism, 95.38% of vertices and 98.46% of patches are eliminated under 50 mm precision. The experiment results indicate that users can quickly load large BIM models and fetch sufficient information from the website.

Keywords: BIM; IFC; lightweight; visualization; steel structure

Citation: Sun, Z.; Wang, C.; Wu, J. Industry Foundation Class-Based Building Information Modeling Lightweight Visualization Method for Steel Structures. *Appl. Sci.* **2024**, *14*, 5507. https://doi.org/10.3390/app14135507

Academic Editors: Sangyoon Chin and Seungyeon Choo

Received: 7 May 2024
Revised: 31 May 2024
Accepted: 20 June 2024
Published: 25 June 2024

1. Introduction

Along with economic and social development, traditional construction production faces problems such as labor shortage, large resource consumption, and environmental pollution. Many countries have issued a series of policies to promote building construction industrialization to solve these problems. As one of the industrialized building systems, the steel structural system, whose members are generally processed and produced in factories, has the unique advantages of high industrialization because of its lightweight, high strength, and short construction cycle [1,2]. Since many components are involved in steel structures, high installation quality is required, bringing difficulties in project management. Existing research has indicated that the use of information technology to process data during construction can significantly improve project management and the whole life cycle assessment level [3–5]. As an important information technology, building information modeling (BIM) has been widely used in project design, construction, and management [6–8]. Eastman et al. [9] pointed out that the application of neutral and open BIM standards can achieve the interoperability of BIM data among different types of software. For steel structures, BIM-based steel structural design software has been widely used in studies such as references [10–13], but this software mainly focuses on structural design rather than the management of construction information.

With the rapid development of the Internet, more and more Web-oriented BIM applications are emerging, aiming to solve the problem of information interaction in BIM models. Some Web-oriented open-source BIM systems are available, such as BIMServer [14]

and xBIM [15]. The BIMserver system is developed as an open-source BIM server that supports storage, maintenance, and the query of industry foundation class (IFC). The xBIM plugin is a free software module with a specific set of functions, which can be invoked in application software to support geometric data calculation. However, without specific performance optimization for steel structural models, such systems are mainly used in concrete structural models rather than steel structural ones, and they have difficulty in handling large-scale steel structural models. Although some software providers have developed a Web version of the BIM system, the source code is not available, and secondary development is challenging to achieve. Furthermore, the internal structure of the IFC format is still imperfect, which is mainly caused by the following three drawbacks [16]. (1) IFC contains more than 900 predefined entities and native data types, and its internal logical structure is very complex and coupled. The reference in IFC files needs to follow strict hierarchical rules. In other words, the deeper a reference level is, the more complex its logic structure will be. (2) There is a large amount of information that is unnecessary in IFC files. Because of the intended generality of the IFC format, the parsed model data in real-life projects are usually much larger. Furthermore, according to the reference rules, many intermediate layers are always needed, causing a large quantity of redundant information in IFC files. (3) The IFC format has a loose scene organization.

Therefore, current limitations of the Web-based system include the following: (1) the model file is too large, which requires too much time to transmit; (2) too much unnecessary information in IFC models causes some problems during data storage; and (3) too many components that are similar in terms of geometry overconsume the hardware resource, resulting in a time delay or system crash.

Geometric redundancy removal in BIM presents significant challenges. In the IFC framework, multiple components sharing the same geometry can be recorded, but existing methods for removing redundancy rely heavily on data organization and export logic of the IFC files. If components are not specified as the same type in the modeling software, shared geometry information may not be recorded. Additionally, when an IFC file is imported, manipulated by other software, and then exported again, or when multiple IFC files are merged, shared geometry information may be lost, undermining redundancy removal effectiveness. The method proposed in this study addresses these issues by providing a more robust solution, ensuring that geometric redundancy is effectively removed regardless of the data export and import processes. Another approach involves the iterative closest point (ICP) method based on point cloud matching, as demonstrated by Zhou et al. [17] in their cross-platform online visualization system for open BIM. The ICP method generates point clouds from BIM models to determine matching and identify redundancy. However, traditional ICP methods face limitations such as dependency on initial model positions and efficiency issues due to the multiple iterations required to address measurement errors. The enhanced ICP method offers higher stability and efficiency, making it better suited for geometric redundancy removal in BIM models.

Common mesh simplification algorithms, such as vertex deletion, edge collapse, triangle collapse, region merging, and vertex clustering, often present issues like uncontrollable simplification processes and non-intuitive thresholds. These algorithms typically aim to preserve mesh details in generic meshes, which can be problematic when applied to BIM models for steel structures. In these models, small details like bolts are sharp, and considering curvature, larger flat components are more likely to be removed than smaller parts, distorting the original structural characteristics. The method proposed in this study uses the length of the collapsed edge as an error measure, making the mesh simplification process more understandable for users and more suitable for representing the steel structure model.

To overcome these limitations, this study aims to develop a lightweight BIM system and optimize the system's performance according to the specific characteristics of steel structural models. To achieve this, (i) BIM data needs to be extracted into the Web system and remain intact, and (ii) the visualization of BIM models should be smooth. Although

IFC is a universal BIM data standard, its structure is complex and coupled. In order to achieve a BIM system for the construction management of steel structures, it is necessary to design and optimize the storage of BIM data. Furthermore, the lightweight visualization of BIM models is also an essential part of BIM application. The rendering performance of BIM visualization is closely related to both the project type and information quantity of the BIM model. Therefore, it is an important work to optimize the rendering performance of BIM models with graphics technology. Using BIM technology to solve specific engineering problems is still a hot issue in the field of civil engineering [18,19]. However, few studies have focused on the BIM system for the construction and management of steel structures. Because of the distinct characteristics of steel structures, the performance optimization of the BIM system is urgently required and needs to be studied further.

This study proposed a solution for information extraction and storage for the BIM platform, optimized the system performance through several techniques, including dynamic level of detail (LOD) model loading and redundant instance removal, and finally, realized the key functions of the Web-oriented information management system, which is mainly used in the construction stages for steel structures. During this process, the edge collapse algorithm, principal component analysis (PCA) algorithm, and Hausdorff-based comparison algorithm were employed. Several techniques such as static rendering and cache mechanism were also implemented to improve the speed of data transmission and the efficiency of model display. The developed system was evaluated by three steel structural models, and the results showed that users can quickly load large BIM models and fetch sufficient information from the website.

This paper is organized into six sections. Section 1 presents the introduction of this paper. Section 2 presents the research progress and introduces work related to the Web-oriented BIM system. The scope and methodology of this research are presented in Section 3, and Section 4 presents the theoretical analysis, explaining the key algorithms and techniques utilized in this research. Section 5 introduces the realization of the BIM system, and Section 6 presents the experimental verification and the developed BIM platform. Finally, Section 7 summarizes the research conclusions and proposes future work.

2. Research Background

2.1. Research Progress on the BIM Server

For model storage and visualization, some existing BIM server systems are the IFC model server [20], EDM model server [21,22], BIM server, xBIM, and collaborative design BIM service systems [23–25]. Among them, the BIM server and xBIM toolkit are open-source software, which are available for secondary development. Because of the hindrance of flexibility and extensibility of commercial BIM software, a significant amount of existing research studies are related to developing BIM systems based on the BIM server [26–28], xBIM [29,30], and other open-source BIM systems. As open-source BIM systems provide functions of model visualization, information storage, and rich interfaces, the detailed development of open-source BIM systems can meet the requirements of complex processes and functions. However, this kind of system is mainly used to manage concrete structural models without specific optimization techniques for steel structural models. Because of the slow model analysis and low display efficiency, it is difficult to meet the requirements of information management for large-scale steel structures. Furthermore, owing to the complexity of optimizing existing open-source BIM systems, independent BIM server development has attracted some attention. For instance, Xu et al. [31] proposed a method to create 3D visualization for BIM models in Web browsers by combining IFC and WebGL techniques. However, extracting geometric information from IFC files can barely describe the component attributes, which causes some limitations in display accuracy and efficiency. In order to achieve complete information storage and lightweight visualization for BIM models, some researchers have focused on the areas of IFC information extraction and storage and model lightweight display, which are introduced in the following sections.

2.2. Approaches to IFC Information Extraction and Storage

At present, the architecture, engineering, and construction (AEC) industry mainly employs IFC to support building information exchange [32]. IFC represents a conceptual data schema, and it contains an exchange file format of BIMs according to the ISO 16739-1:2024 standard, inside which the rigid and complex hierarchical structure of the IFC schema prevents simple extraction of building information manually and requires a detailed understanding of the IFC data model [33].

To parse the information in the IFC file efficiently, some researchers have investigated the field of IFC file analysis, processing, and storage [34–37]. Khalili and Chua [38] developed a graph data model (GDM) to employ semantic information, to extract, analyze, and present the topological relationships among 3D objects in 3D space, and to perform topological queries faster. Zhu et al. [39] realized the extraction and transformation of geometry information from IFC to ShapeFile format. In order to better support network transmission, Afsari et al. [40] developed a JavaScript Object Notation (JSON) representation of IFC specification. Some studies were also carried out on BIM database design to improve information processing and retrieval efficiency. For example, Chen et al. [41] developed a cloud-based BIM information storage and visualization system using a Hadoop-distributed file system. Solihin et al. [42] designed a BIM database to store IFC component attributes, spatial relations, and geometric information, and their research mainly focused on the complete storage of IFC files. In addition to these methods, the concept of ifcOWL is also crucial in managing BIM data. ifcOWL converts the IFC schema into Web Ontology Language (OWL), enabling the application of Semantic Web technologies [43]. This transformation promotes the linked data approach, enhancing data interoperability and reducing redundancy. By using ifcOWL, data points within the BIM model can be more effectively connected, allowing for advanced querying and better cross-system data integration. However, IFC files usually contain complex and redundant information, which does not need to be stored entirely during the construction stages of steel structures. Moreover, redundant information burdens the system's performance. Therefore, the extraction and storage of IFC information for steel structural models need to be customized.

2.3. BIM Lightweight Visualization Methods

Nowadays, commercial software companies such as Autodesk, Tekla, and Bentley have already developed mature BIM software to visualize BIM models. However, such software is all desktop and requires hardware with adequate performance. As a neutral file format, IFC has been widely supported by commercial software and open-source software. Therefore, our research focuses on the visualization of the IFC model on the Web side.

In order to improve the display performance, some researchers employed several methods, such as visual culling [44,45], spatial index [46], scene graph [47], LOD [48], cache design, and component redundancy removal. For instance, Leite et al. [49] analyzed the modeling effort required at different LODs and the impact of the LOD on modeling effort and clash detection. Sun et al. [50] presented a comparable content-based compression algorithm to reduce the redundant information in IFC files through an iterative compression procedure based on an IFC model's tree structure. Liu et al. [51] presented a prototype of a lightweight and real-time Web3D visualization system for large-scale BIM scenes, striving to increase the data accessing rate and achieve better data management. Hu et al. [52] performed geometric optimization for BIM models in mechanical, electrical, and plumbing (MEP) projects by improving algorithms and techniques for storage, transmission, and display. Zhou et al. [17] developed an online BIM visualization system, WebBIM, based on IFC and WebGL by rendering decompressed triangular BIM data. In their study, WebBIM first converts the raw IFC geometry data into triangles, which can be directly rendered in WebGL. Then, it removes the redundancy in the geometry information by sharing the geometry data among the object instances generated from the same facility components. Xu et al. [16] provided an implementation plan for data integration and information interaction between BIM and a geographic information system (GIS) and

improved the ability of network transmission and browser rendering. IFC.js specifically excels in the dynamic loading and rendering of IFC models, which is crucial for managing the typically large datasets involved in BIM. It enables more efficient and interactive viewing experiences, thus improving system responsiveness and reducing computational burdens. However, its performance may suffer when dealing with large and complex steel structural models, which contain a large number of members, plates, bolts, welds, and so on. Because of the limitations of JavaScript and WebGL, the rendering of complex steel structural models may cause the browser to respond slowly or even temporarily freeze.

With respect to steel structural models, taking the IFC file exported from Tekla software (2022) as an example, the main components of the file include beams, columns, bolts, and gusset plates. Although the employment of LOD technology can improve the performance of the BIM system, the LOD of different entity categories is not clearly defined in the IFC format. It is important to clarify that the LOD discussed herein pertains specifically to graphical detail management rather than the informational depth traditionally associated with BIM stages. In computer graphics, LOD technology dynamically adjusts the complexity of a 3D model's geometry based on viewing distance or computational constraints, aiming to optimize processing loads and maintain visual fidelity other than the granular information modeling required at different stages of a BIM project. If the components are simply divided by instance categories, their specific sizes will greatly influence the performances of LOD models. For example, the size of gusset plates may be large or small, and removing all of them will reduce the display accuracy of the model. In addition, since no wall, slab, or other component blocks the view in the construction models of steel structures, the spatial index method cannot improve the performance. As for component redundancy removal, since the instance alignment process is not included in the methods proposed by Hu et al. [52] or Xu et al. [16], the specific implementation of the software system and local orientations of the instances will influence the display results. Regarding the sparse ICP method [53], its matching accuracy and efficiency depend on the initial positions of components. Therefore, lightweight display technology needs to be customized for steel structural BIM models. In this paper, dynamic LOD models adjust the level of detail based on context-specific requirements and constraints, which is particularly valuable in Web-based BIM systems. This study focuses on graphical LOD techniques such as dynamic model loading and edge collapse algorithms, specifically designed for the computer graphics domain. These techniques are intended to reduce geometric complexity efficiently, thereby ensuring rapid rendering and high visual quality, which are essential for large-scale visualization projects. These strategies ensure efficient data transmission and visualization, tailored to the unique characteristics of steel structures. Consequently, the proposed LOD technique enhances the overall performance of the BIM system by improving rendering speed, reducing data redundancy, and maintaining high display accuracy, making it highly suitable for managing large-scale steel structural projects.

In this study, the challenges include managing large amounts of complex data, ensuring high-quality steel structure installation, and optimizing the performance of the BIM system for large models. Model view definition (MVD), part of the IFC standard, aims to standardize BIM data representation [54]. Lee et al. [55] developed an XML-based MVD library that supports entity-based concept integration. The conceptual framework defines entities, relationships, and attributes, thereby simplifying the access and reuse of existing concepts and facilitating the development of new model views. Liu et al. [56] proposed MVDLite, a lightweight MVD rule representation method that modifies the XML structure to make it easier to understand. While MVD supports specific data exchange needs and enhances interoperability and data exchange efficiency, it cannot fully address our specific challenges. The high level of detail and specificity required for managing steel structure components, such as detailed records of members, connectors, and welding seam, necessitates more granular data classification and measurement methods than what MVD typically provides. Additionally, the complexity and redundancy in IFC files require advanced data extraction and redundancy removal algorithms beyond MVD's

capabilities. Therefore, a redundancy removal approach by combining the PCA algorithm and the Hausdorff-based comparison algorithm is proposed to identify standardized steel components, and an efficient visualization method on Web3D for redundant instances is also achieved.

3. Scope and Methodology

3.1. Scope

The data in the AEC industry contains almost all information about the life cycle of buildings, resulting in an exponential increase in the amount of data. Meanwhile, a Web-based application is strongly recommended for its compatibility with various operating systems and devices, such that users can access the latest version of the system immediately without any updates and do not need to install third-party software. This results in the challenge of improving the system performance when displaying the 3D models of the buildings on the Web side.

This study aims to improve system performance for displaying 3D models in IFC format on the Web side from two aspects. On the one hand, the developed system can reduce the time of data transmission by caching the model file into the browser's built-in database IndexDB when the model is first displayed. The model can also be loaded automatically from the cache subsequently. On the other hand, the system can reduce hardware resource consumption by using dynamical LOD loading and redundancy removal of component geometry information.

In large steel structures, the display performance of the 3D model may be seriously impaired by a large number of similar components in terms of geometry. Since WebGL can render similar instances only once, these similar components are called "redundant components". In addition, when the viewpoint is too far from the displayed model, there is no need to load the 3D model entirely, which means that the system only needs to load the model without too much detail. For redundancy removal of components, this study presented the following three methodologies: (1) geometry alignment, (2) measurement of the geometric similarity, and (3) redundancy removal. Moreover, as for the dynamic LOD model loading, the edge collapse algorithm was employed.

3.2. Methodology

In order to develop an available and efficient management system for steel structures, it is necessary to optimize the system's functions and performances. The overall technical route of this study can be divided into three parts as follows: information extraction and storage, redundancy removal of component instances, and dynamic LOD model loading. The first part mainly deals with the original text information of the IFC format on the server side. The last two parts perform off-line data processing on the server side, as well as optimization and lightweight display on the browser side. The technical route is shown in Figure 1. The left part of this figure shows the BIM model containing IFC instances, which is the data source of the system, and the right part is the main work of this study. After being extracted from the IFC files, the BIM data falls into two parts as follows: semantic data, which can be stored in a database, and geometric data, which is used for optimized visualization.

Information extraction and storage: Current BIM applications mainly focus on the complete extraction and lossless storage of building information from IFC files. The storage structure is relatively complex, and it is not suitable for network transmission because of the coupled IFC Tree. In this study, the geometric and semantic information of components based on IFC files were extracted, decoupled, and stored in the database. In the next step, the spatial relationship and geometric information of components were stored in the form of files and cached on the client side, reducing the overhead of database storage and network transmission.

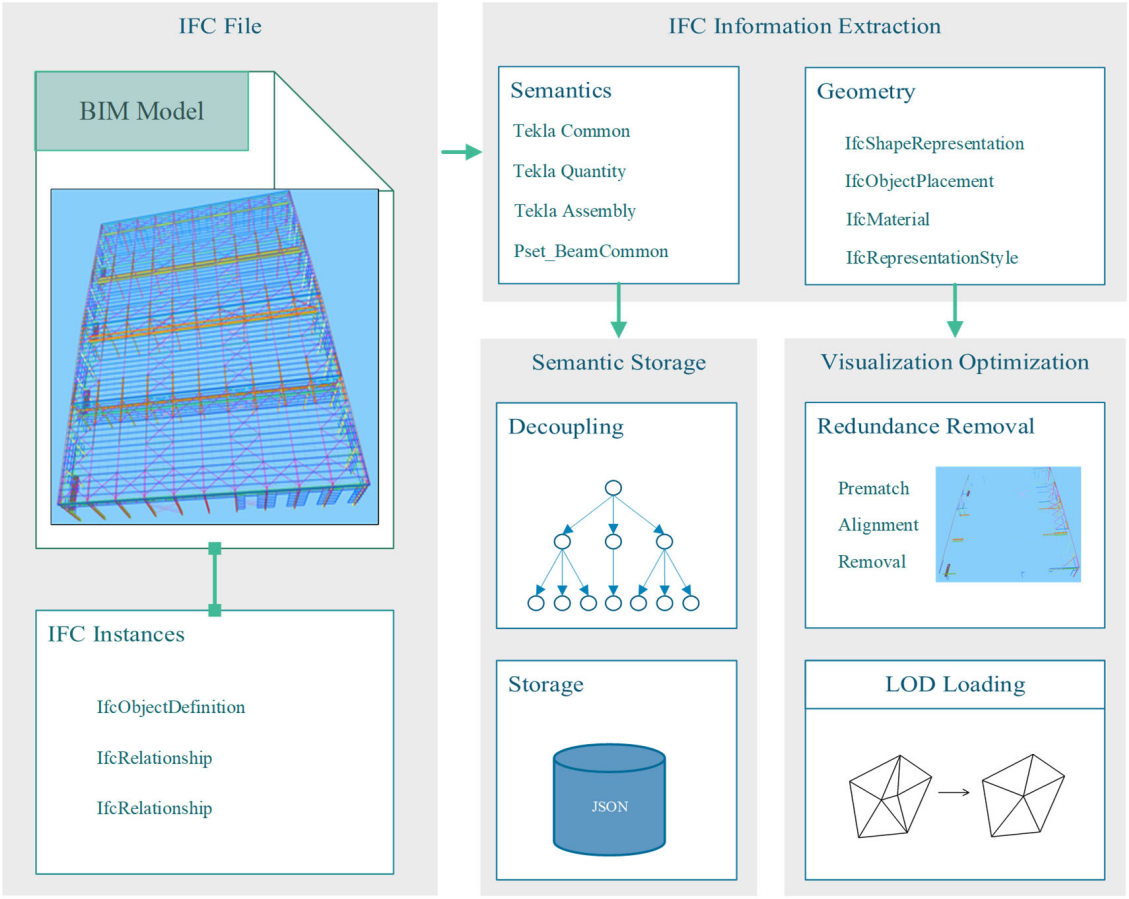


Figure 1. Technology roadmap.

Removal of redundant instances: Since steel structures are highly industrialized, there are many redundant instances in the BIM model. The existing research [57] pointed out that the removal of redundant instances can significantly improve display efficiency and network transmission efficiency. To achieve efficient rendering for standardized components, this study presented an alignment and visualization technology for redundant instances.

Dynamic LOD model loading: In most of the existing studies, BIM models are loaded dynamically according to the IFC instance type by spatial index. However, in the steel construction model, since there is no wall or slab to block the view, the spatial index cannot decide which components to display based on the viewpoint location. Therefore, in this study, LOD models of the whole building were established according to the specific sizes of the components rather than the spatial index, and the display parameters were determined by the camera distances and the sizes of the instances. Based on this mechanism, the dynamic LOD model loading is effectively supported, and the display efficiency is greatly improved.

4. Technical Analysis

4.1. IFC Information Extraction and Storage

In order to better support the cooperation of all participants in the whole life cycle of buildings, buildingSMART [58] developed the IFC standard to create a BIM model, which is becoming the most widely used open standard for BIM data. In the IFC format, data objects are described by predefined entities and data types using the Express language.

In general, the IFC format not only defines instances but also describes models through geometry, spatial relations, and other types of information. Spatial relations describe the association between instances and other types of information. Geometric information

includes instance materials, spatial positions, and geometric expressions, which can be described by solid geometry, extrusions, patches, boundaries, etc.

In IFC files, components are mapped from local coordinates to global ones by multiple transformations. The component coordinate is generally represented by the *IfcLocalPlacement* class, which contains the coordinate origin, X-axis, and Z-axis directions.

The IFC model is a semantic data model, which defines elements contained in the AEC domain as well as their relationship. For instance, *IfcRelDecomposes* describes the composition relationship among IFC entity objects, and *IfcRelDefines* is used to link instances with types or property sets. Figure 2 shows the spatial hierarchy of building elements in the IFC file. According to the IFC format, a project can contain one or more sites, and a site can contain one or more buildings, while a building may have one or more stories. All the building elements are attached to stories. This hierarchy is also referred to as the IFC Tree. Generally, the leaf nodes of the IFC file include columns, beams, plates, and so on. They contain some basic attributes and can be linked with property sets, which involve extended attributes.

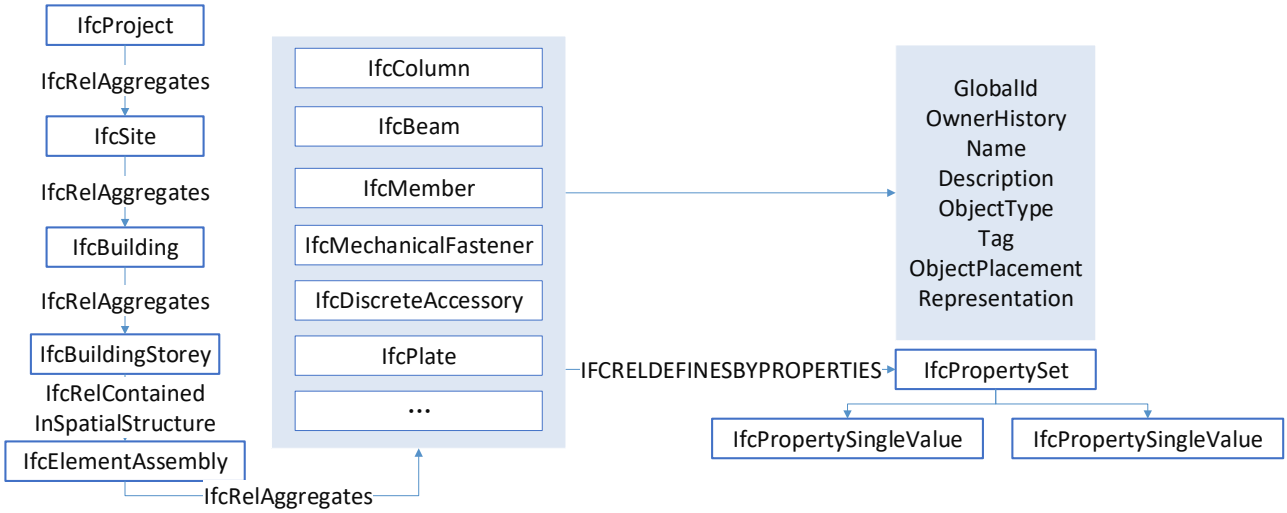


Figure 2. IFC file structure.

Generally, an IFC file has a large amount of data and complex references. Moreover, varieties of geometric descriptions increase the complexity of visualization. Therefore, in order to achieve efficient information storage, retrieval, and rendering, the developed system needs to be optimized according to the characteristics of the original IFC file.

4.2. Redundancy Removal of Component Geometry Information

4.2.1. Geometry Alignment

Although the same components derived from different types of software may have different geometric representations, the geometric shapes remain unchanged in the local coordinate system. Therefore, by aligning the components into a certain coordinate system, the similarity among them can be obtained. An appropriate placement of instance should be determined according to the features of the data themselves. PCA [59] and ICP [60] are two common alignment algorithms that are usually employed for the coarse alignment and fine alignment in point cloud alignment. However, for the errorless model, it is unnecessary to perform ICP analysis to avoid errors in measurement, such that only PCA is available. The coordinate, which can best express the data feature in the reduced dimensionality, can be achieved by PCA. Generally, after aligning the same data in different orientations into this coordinate, the geometry topology remains unchanged. This fact lays the foundation of similarity comparison among different objects.

In the PCA method, the dataset X can be obtained by sampling the centroid of every patch. Suppose there are n points in point set X, and x_i, y_i, and z_i are the coordinates of the i-th point. The coordinate matrix can be expressed as:

$$X = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \ddots & \vdots \\ x_n & y_n & z_n \end{bmatrix} \tag{1}$$

First, the point set needs to be normalized:

$$X_{std} = \begin{bmatrix} \frac{x_1-\mu_x}{\sigma_x} & \frac{y_1-\mu_y}{\sigma_y} & \frac{z_1-\mu_z}{\sigma_z} \\ \frac{x_2-\mu_x}{\sigma_x} & \frac{y_2-\mu_y}{\sigma_y} & \frac{z_2-\mu_z}{\sigma_z} \\ \vdots & \ddots & \vdots \\ \frac{x_n-\mu_x}{\sigma_x} & \frac{y_n-\mu_y}{\sigma_y} & \frac{z_n-\mu_z}{\sigma_z} \end{bmatrix} \tag{2}$$

where X_{std} is the normalized form of point set X; μ_x, μ_y, and μ_z are the average values of the x, y, and z coordinates; and σ_x, σ_y, and σ_z are the standard deviation of the three coordinates. Equation (1) can be written in the following transformation form:

$$X_{qua\ std} = X_{qua} \begin{bmatrix} \frac{1}{\sigma_x} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_y} & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_z} & 0 \\ -\frac{\sigma_x}{\mu_x} & -\frac{\sigma_y}{\mu_y} & -\frac{\sigma_z}{\mu_z} & 1 \end{bmatrix} \tag{3}$$

where X_{qua} is the quaternion form of the point set and $X_{qua\ std}$ is the normalized form of X_{qua}.

Then, the covariance matrix of the point set is calculated as:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} X_{std} X_{std}^T \tag{4}$$

Generally, singular value decomposition can be employed, and eigenvectors of the covariance

$$U = \begin{bmatrix} U_x & U_y & U_z \end{bmatrix} \tag{5}$$

where U_x, U_y, and U_z are the axes directions in the new coordinate placement.

Finally, the point set is projected into the new placement, and a transformation operation can be simply performed:

$$X_{new} = X \begin{bmatrix} U_x & U_y & U_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where X_{new} is the coordinate of the transformed point set. After the above process, the point set is transformed into a coordinate system determined by its data characteristics. Even if the point sets of the same component are in different positions and orientations, they can also be aligned to the coincident state, so the aligned point set data can be used for redundant component identification.

4.2.2. Redundancy Removal

Many components in an IFC file are similar or identical in geometry, which are redundant for the rendering progress. By using Hausdorff distance, the developed system can evaluate the similarity in instances and classify similar ones into one group. Since rendering instances in the same group only needs one drawcall, the time and hardware cost are significantly reduced. Although the BIM software built-in *IfcMappedRepresentation* can

label similar components, this software can only label instances based on the user's specific operation, such as copying. This mechanism may cause failure in recognizing other newly created instances that are same as the previous ones. Therefore, the developed system used PCA and Hausdorff distance to classify instances in terms of their geometry and identified similar components within user-defined precision, such as the default millimeter precision.

Hausdorff distance [61] is used to measure the geometric similarity in components in the local coordinate system, and it is defined as:

$$H(A,B) = max(h(A,B), h(B,A)) \tag{7}$$

$$h(A,B) = \underset{a \in A}{max} \left\{ \underset{b \in B}{max} d(a,b) \right\} \tag{8}$$

where $H(A,B)$ is the Hausdorff distance; A and B are the point sets; a and b are the points in A and B, respectively; and $d(a,\ b)$ is the Euclidean distance between points a and b. For the point a, the distance between point a and each point b in point set B is solved, and the minimum value is set as the distance between point a and point set B. The maximum distance between each point a and point set B is denoted by $h(A,\ B)$. The Hausdorff distance between point sets A and B is defined as the maximum value between $h(a,\ b)$ and $h(b,\ a)$. Figure 3 shows a flowchart of the redundant case recognition algorithm based on Hausdorff distance.

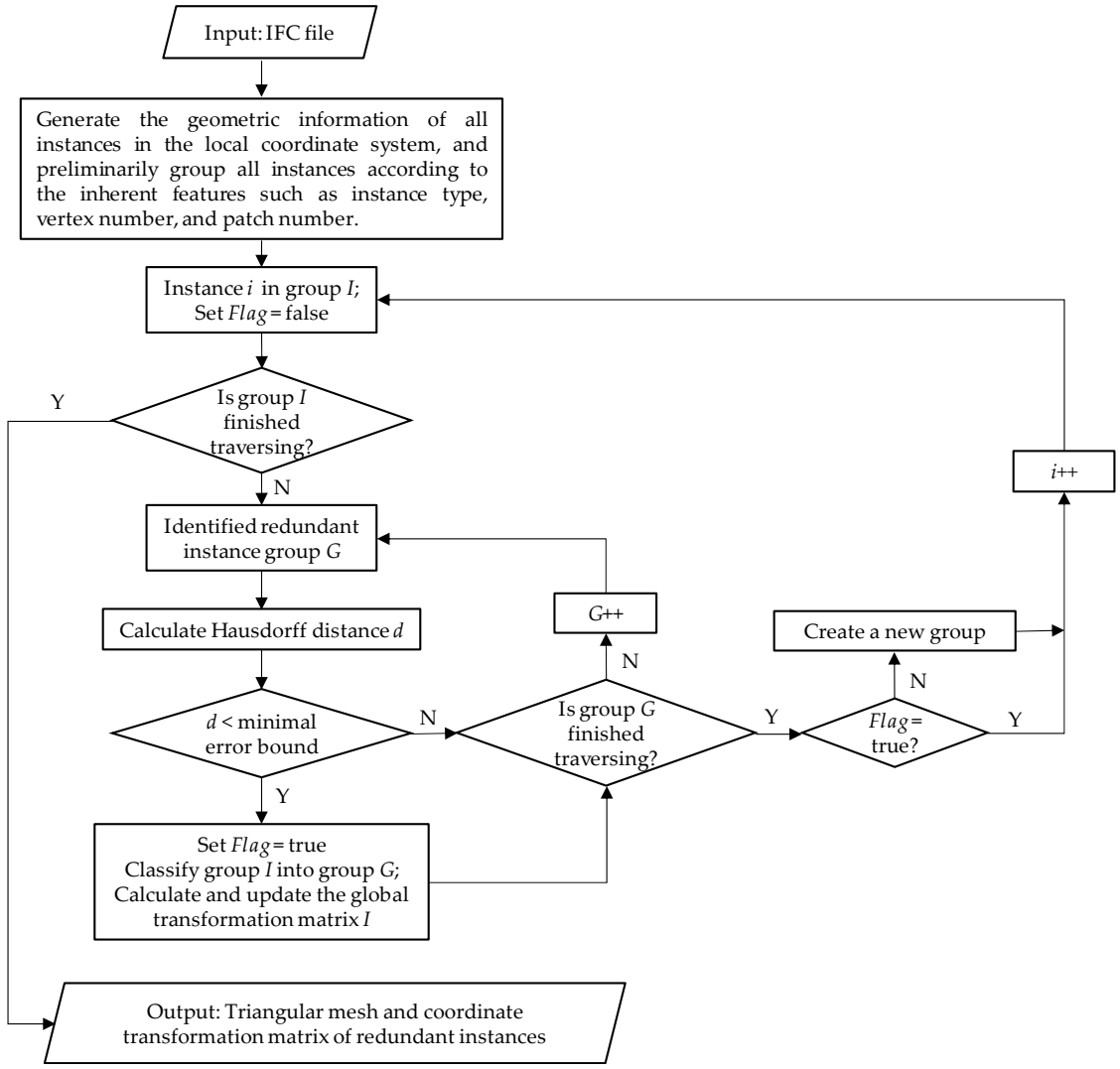


Figure 3. Flowchart of the redundant case recognition algorithm based on Hausdorff distance.

WebGL technology is a common rendering framework in the Web system, and it provides rich interfaces to control the rendering process. The WebGL rendering process is shown in Figure 4. Before rendering an instance, the CPU needs to transfer the vertex, facet, and material data to GPU, which is called a drawcall. The drawcall efficiency is the performance bottleneck of 3D rendering technology in the hardware architecture. Therefore, merging the drawing components with the same shape into a single drawcall can significantly improve the rendering efficiency.

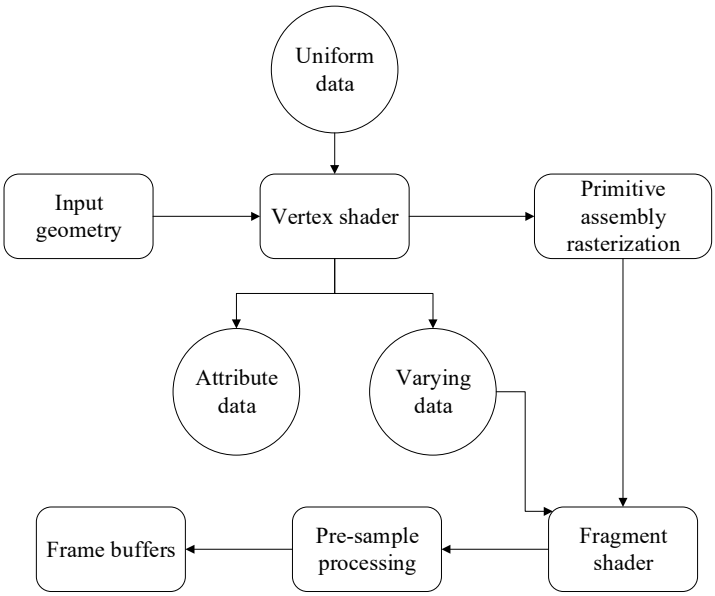


Figure 4. The WebGL rendering process.

In the developed BIM system, the three.js framework was employed to render the model on the browser side. To improve the rendering efficiency, it is necessary to reduce the number of drawcalls as much as possible. Through a single drawcall, a large number of instances can be drawn, such that the rendering efficiency is greatly improved. The three.js framework provides *InstancedMesh* objects to achieve this function. By inputting a shared geometric mesh and multiple coordinate transformation matrices into shaders, multiple objects with the same shape need to be drawn only once.

4.3. LOD Model Loading

It is well known that joints play an important role in steel structures. The layout of joint plates, bolts, and other parts can greatly affect the mechanical performance of structures. Therefore, in the construction models of steel structures, it is necessary to model the joints accurately, which contain a large amount of geometric information. The installation process of joints has little impact on the macroscopic management of steel structures, but it takes up many hardware resources. For general BIM models, some researchers utilized the methods of spatial index and visual field culling to reduce the hardware load during rendering [62,63]. However, since there are no walls, plates, or other components to block the vision in the construction models of steel structures, especially no rooms or stories for some types of structures such as steel structural workshops, the method of vision field culling is not feasible in rendering construction models of steel structures. Therefore, in this study, a multi-level LOD loading mechanism was presented to improve the system performance by combining the characteristics of the steel structural models and perspective projection.

The specific scheme is as follows: (i) simplify the mesh of the components in the geometry files into different degrees; (ii) apply simplified geometry information to save hardware resources when the viewpoint is far away from the instance; and (iii) apply less

simplified geometry information to express the detailed geometry features of the instance when the viewpoint is close to it.

There are two basic principles for the mesh simplification technology [63] as follows:

(1) Minimum vertex principle: minimize the number of model vertices under the given error limit.
(2) Minimum error principle: minimize the error between the generated model and the original one under the given simplification rate.

The commonly used algorithms of mesh simplification include vertex deletion [64], vertex clustering [65], triangle folding [66], region merging [67], edge folding [68], etc. This paper employed the edge folding algorithm to perform mesh simplification, which is widely used in practice. The algorithm process is as follows:

(1) Calculate the error associated with each edge and create the minimum heap.
(2) Take out the edge with the smallest error, and then fold the edge. To decrease the error, place the new vertex, modify the error of the new edge, and adjust the minimum heap.
(3) Repeat the process until the user requirements are met or the minimum heap is empty.

A single step of simplification is illustrated in Figure 5. In this figure, the shortest edge AB is folded, and then the corresponding edges AC, AD, AE, BE, BF, BG, and BC in the left polygon are replaced by edges AC, AD, AE, AF, and AG in the right one. Particularly, the coordinate of the new point A' should follow the principle of minimum error. Figure 6 shows a flowchart of the algorithm. The data structure of the program is shown in Figure 7. The main steps of the implementation are listed as follows.

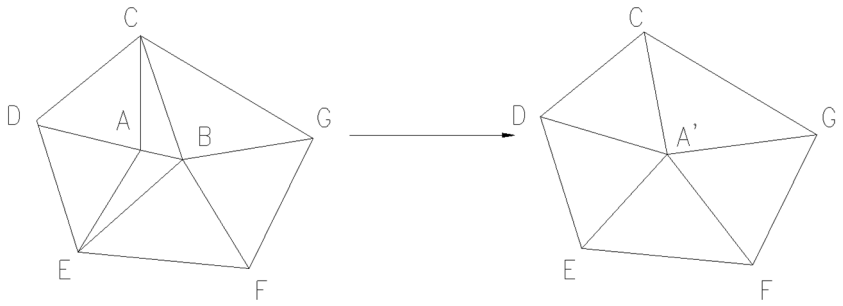


Figure 5. Process of the edge collapse algorithm.

(1) Find the shortest edge denoted as E (V_1, V_2), which needs to be folded.
(2) In the vertex array, delete the two vertices, V_1 and V_2, and add the midpoint V_3 of vertices V_1 and V_2 into the vertex array.
(3) In the edge heap, delete edge E (V_1, V_2) and replace edges E (V_1, ...) and E (V_2, ...) with E (V_3, ...).
(4) In the patch array, delete t patch F (V_1, V_2, ...) and replace patches F (V_1, ..., ...) and F (V_2, ..., ...) with F (V_3, ..., ...). Since the minimum heap can only support the deletion of the top element, the deletion of the second type of edge is implemented by lazy deletion.

In the edge collapsing algorithm, the sum of squares of the distance between the new vertex and the plane associated with the folded edge is set as an error, and it preserves the details of the model as much as possible. During rendering, the triangular mesh models are switched automatically according to the distance from the viewpoint position to the instance position. When the distance is relatively long, the rougher model is loaded such that the simplified mesh model takes up less hardware resources. While the distance becomes short, the more detailed model is loaded, causing most of the instances to be out of view, which will not affect the rendering performance.

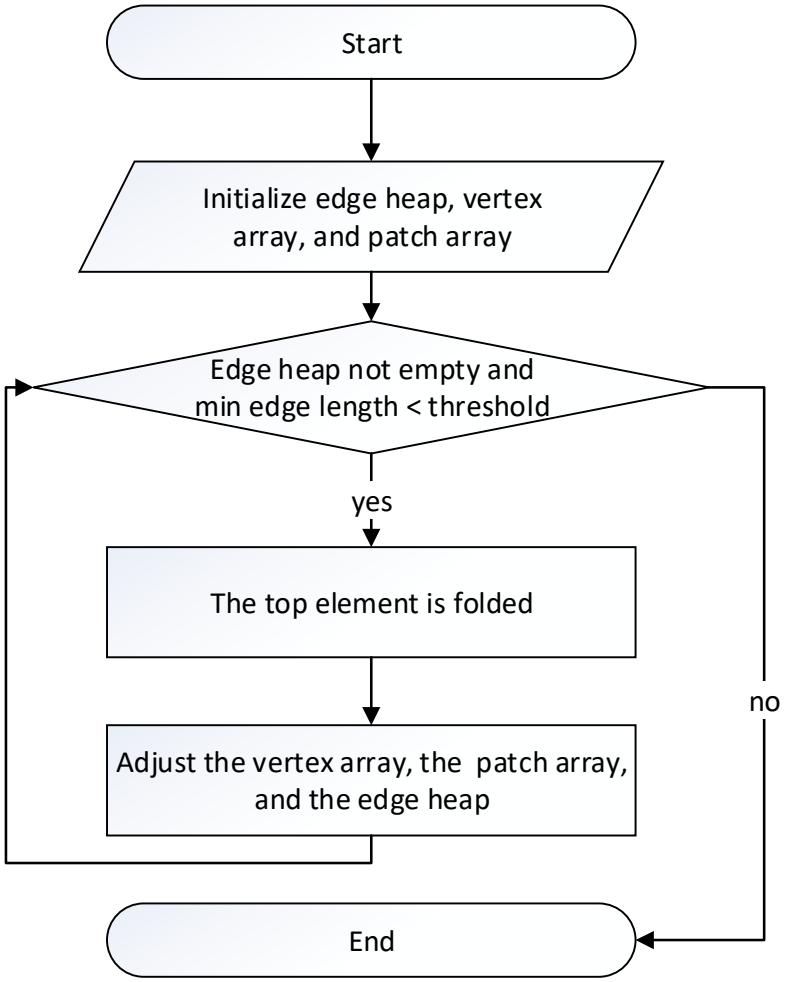


Figure 6. The process of the mesh simplification algorithm.

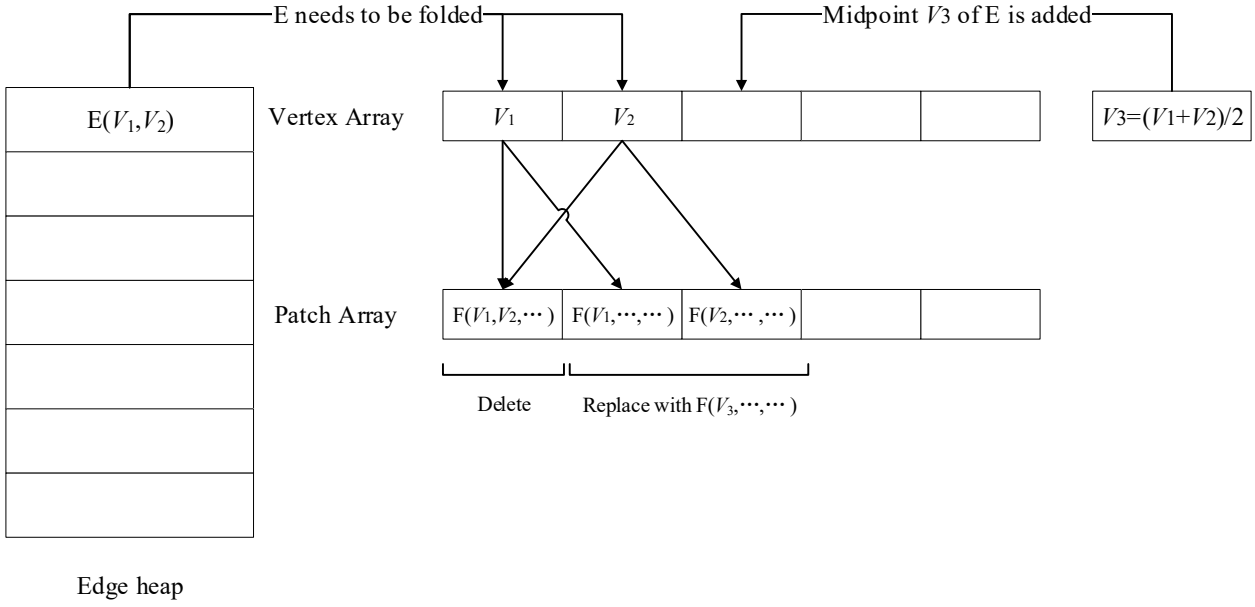


Figure 7. The data structure of the mesh simplification algorithm.

5. System Implementation

5.1. IFC Information Extraction and Storage

IFC information extraction is the basis of BIM system development and model visualization interaction. A reasonable data storage structure can improve system efficiency. Taking the exported IFC file from Tekla software as an example, the *IfcRelAggregates* class defines the inclusion relationship of buildings, floors, components, and other instances. The *IfcRelDefinesByProperties* class defines the inclusion relationship between components and their property set. The *IfcElementAssembly* class defines beams, columns, gussets, and other internal parts. In this paper, the trees of component relationships and attributes were established by using the *IfcOpenShell* [69] plugin first. Then, the tree was decoupled by the depth-first search algorithm. Finally, the reference relationship between the component type and attribute set was recorded.

With respect to data storage, the GUIDs of instances were saved in a MongoDB database as the primary key. Based on the component information saved in the MongoDB database, the integrate node.js and three.js visualization framework was applied to build server and to develop the interface of data interaction. The instance information saved in the database is shown in Figure 8. In this figure, as a child of the *IfcBuildingStorey* node, *IfcElementAssembly* represents the basic unit of a component, and it contains *IfcBeams*, *IfcColumns*, and other elements. The child attribute refers to the inclusion relationship between *IfcBuildingStorey* and *IfcElementAssembly*. In addition, the relationship between the instance and the property set was expressed by *PropertyId*, and the content of the property set was consistent with the predefined data structure. The designed data structure can deal with the semantic information and relationships in IFC files, which can be directly referenced by specific BIM system modules.

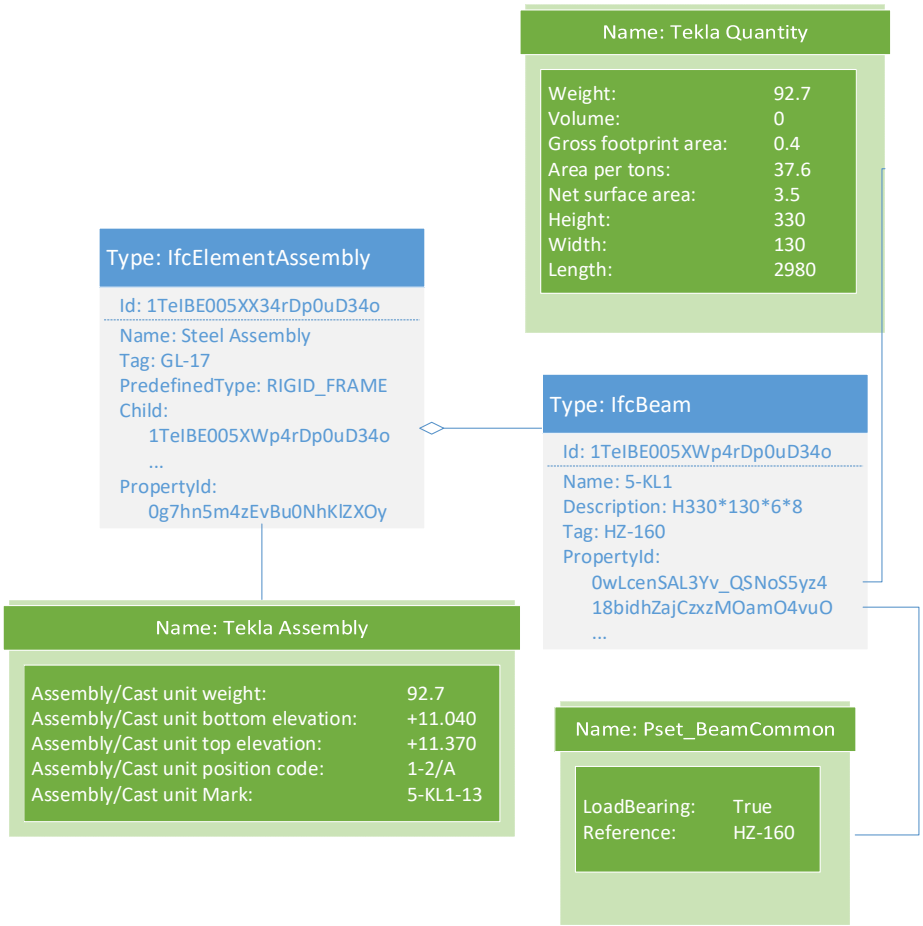


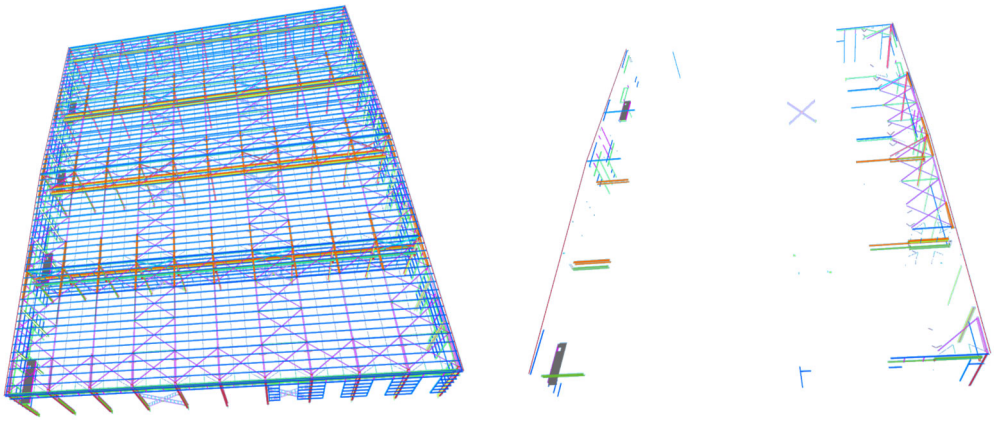
Figure 8. Example of instances and properties stored in the database.

Since the geometry data of instances is usually large, if it is simply stored in the database, the input and output of a vast amount of data will have a significant impact on the performance of the database. To solve this problem, the geometry information of the model was stored in files, which the browser can call. In addition, the GL transmission format [70] is a file format specially designed for 3D model network transmission. The binary code saved in the glTF file can be directly transferred to the graphics card, which provides high display efficiency. Xu et al. [16] used *IfcOpenShell* and the *obj2gltf* plugin to convert geometric information into obj and glTF formats. However, the geometric calculation functions provided by *IfcOpenShell* take plenty of time and space to extract. Moreover, format conversion requires multiple disk inputs and outputs, which may not meet the needs of the quasi-real-time display of the BIM model. Therefore, in order to speed up the extraction of geometric data and reduce hardware overhead, this study employed the IfcEngine [71] plugin to extract the geometric information and utilized the glTF-SDK plugin to output the geometry files.

5.2. Redundancy Removal of Component Geometry Information

For the convenience of design, production, and construction, many components in steel structures have the same geometry. The removal of redundancy components with the same geometry can greatly improve the efficiency of storage, network transmission, and rendering. Given that there are many instances in the steel BIM models, preliminary screening was performed to reduce the scope of comparison according to the type of elements, the number of vertices, the number of patches, and other information. After PCA alignment for all instances, for elements with the same basic attributes, the developed system sampled the geometric vertices and patch centroids and then calculated the Hausdorff distance. If the error of the point set meets the threshold, the point set would be treated as geometric information shared by instances, and then only the GUID and the global coordinate transformation matrix would be marked. In the output glTF files, different geometry nodes shared the same mesh, and the matrix attribute was added to the Node in glTF to record the coordinate positions of different nodes. The glTF files generated according to the above rules are universal and can be used by other software systems.

When using the three.js framework to render the model in the browser, it is necessary to reduce the number of drawcalls as much as possible. Based on the above procedure, many different instances with the same geometry trigger a drawcall only once, significantly reducing the times of input and output. The three.js framework provides the GLTFLoader plugin to load the glTF file. However, in this framework, in order to maintain the names of geometry nodes and other attributes, the shared meshes need to be duplicated multiple times and transformed during the loading process, i.e., performing a drawcall for each shared instance. To solve this problem, the authors improved the plugin by performing coordinate transformation in the vertex shader, as well as creating a mapping relationship between the transformation matrix and the GUID attribute for each instance such that the transformation matrix can be achieved when clicking interaction. Thus, the GUID attribute of the clicked component can be obtained, and the component attribute in the back-end database can be searched and displayed in the front end. Figure 9 shows the display interface before and after the geometric redundancy elimination of a steel workshop model. The testing results are listed in Section 6.

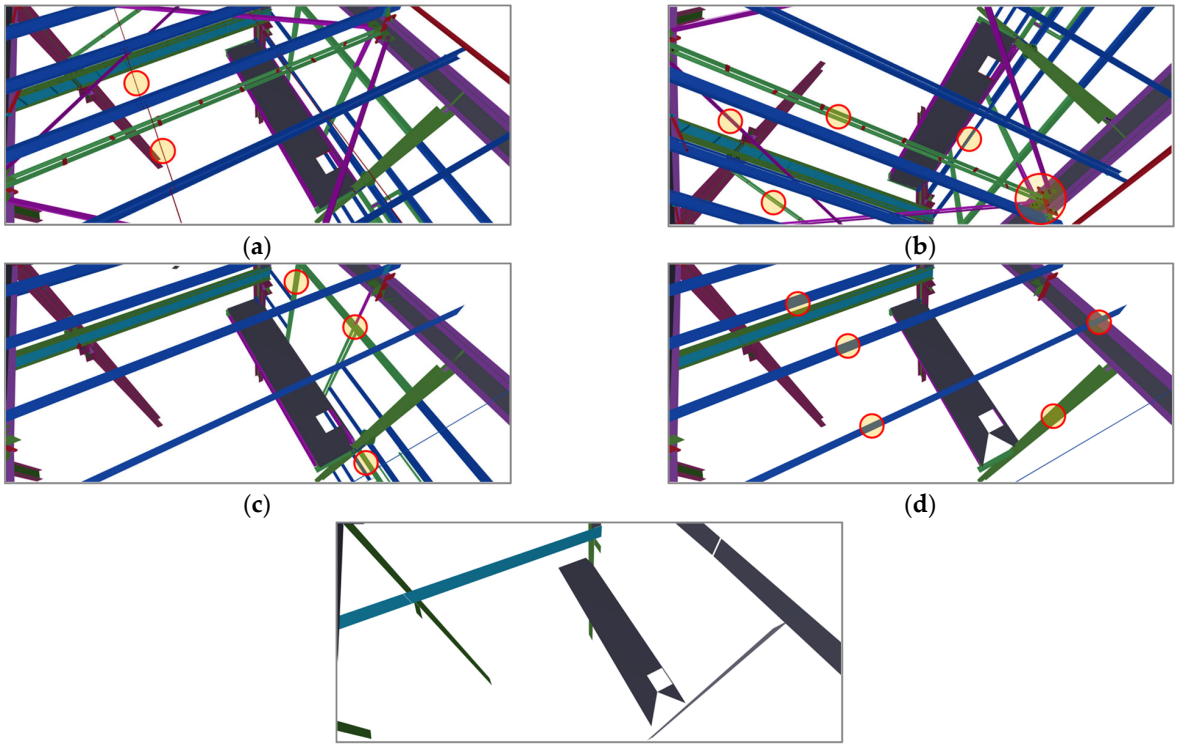


(**a**) (**b**)

Figure 9. BIM models (**a**) before redundancy removal and (**b**) after redundancy removal.

5.3. LOD Model Loading

Given that the size of the model uploaded to the server and the screen size of the user's terminal device are different, the developed system applied user-defined parameters to determine the mesh simplification degree of each LOD model, as well as the threshold distance between viewpoint and instance. According to the edge folding algorithm, since the small components (e.g., bolts) are relatively sharp in the steel structural model, some large plate components might be unexpectedly deleted because of their thin thickness. In addition, it is necessary to ensure that users can understand the meaning of the interactive parameters in the visualized model. Therefore, the length of the folded edge can be defined by users and be used as the error measurement. Figure 10 shows the geometric representation of the same model in different LOD models.



(**a**) (**b**)

(**c**) (**d**)

(**e**)

Figure 10. LOD models under different simplification levels with the shortest edge of (**a**) 0 mm, i.e., original model, (**b**) 10 mm, (**c**) 100 mm, (**d**) 200 mm, and (**e**) 400 mm. In these figures, components marked by circles will disappear in the next LOD model.

In the developed system, when receiving the request to create LOD models, the server-side carries out the program of model transformation and mesh simplification and then stores the transformation results. On the browser side, the loaded LOD models are sorted according to the user-defined distance, and different meshes are switched in real time according to the viewpoint during rendering. Because of the loading of multiple models with different degrees of simplification, the consumption of memory and bandwidth will increase, but the simplified models generally have small volumes, such that little impact is imposed on the overall performance of the system. Moreover, the number of instances that need to be simplified is greatly reduced after redundancy elimination. Therefore, the display speed can meet the requirements of quasi-real-time analysis.

When the user views the model in the roaming mode and changes the camera position, the distances from the camera to the instances will change dynamically. Therefore, different LODs can be switched dynamically according to the changing distances. As shown in Figure 11, the instances with different distances from the viewpoint are displayed by mesh models under different LODs. There are three mesh models in different LODs in Figure 11, whose shortest sides are 0 mm (not simplified model), 10 mm, and 50 mm, respectively. There are three instances, A, B, and C, whose distances from the camera are d_1, d_2, and d_3, respectively. Since the distances between the instances and the camera are different, the mesh models adopted by the three instances can be dynamically adjusted. The current LOD model adopted by the instance is marked with "★" in Figure 11.

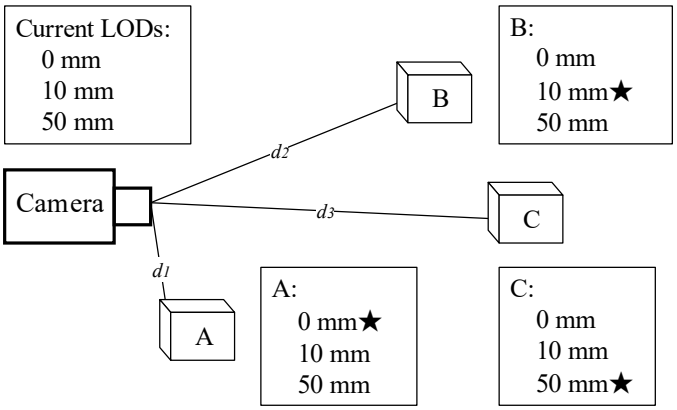


Figure 11. Dynamic loading and switching of LODs.

5.4. Static Rendering Technology

To reduce the network bandwidth of the server and improve off-line processing ability, the geometry information was stored in the IndexDB database of the browser, which can be updated according to the timestamp of the record to ensure the real-time performance of the data interaction. Furthermore, in order to improve the rendering efficiency of the browser, static rendering technology was employed, i.e., the static drawn image was simply maintained when the rendered image was not triggered by users, such that the load of CPU and GPU could be greatly reduced.

6. Experimental Verification

6.1. System Performance

Figure 12 shows the main interface of the developed BIM system. To verify the function and performance of the system, three case models extracted from Tekla software were used. As shown in Figure 13, model 1 is a medium-scale steel structural workshop, model 2 is a multi-story steel structural building, and model 3 is a large-scale steel structural workshop. The model parameters are listed in Table 1.

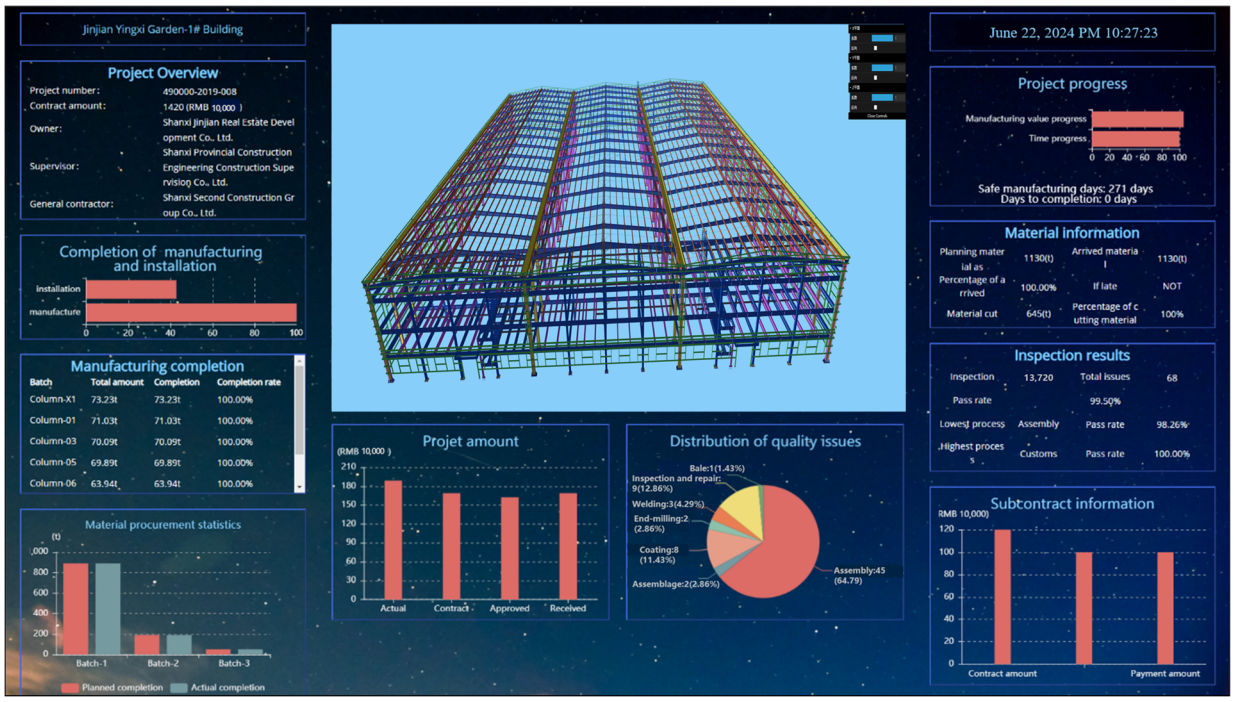
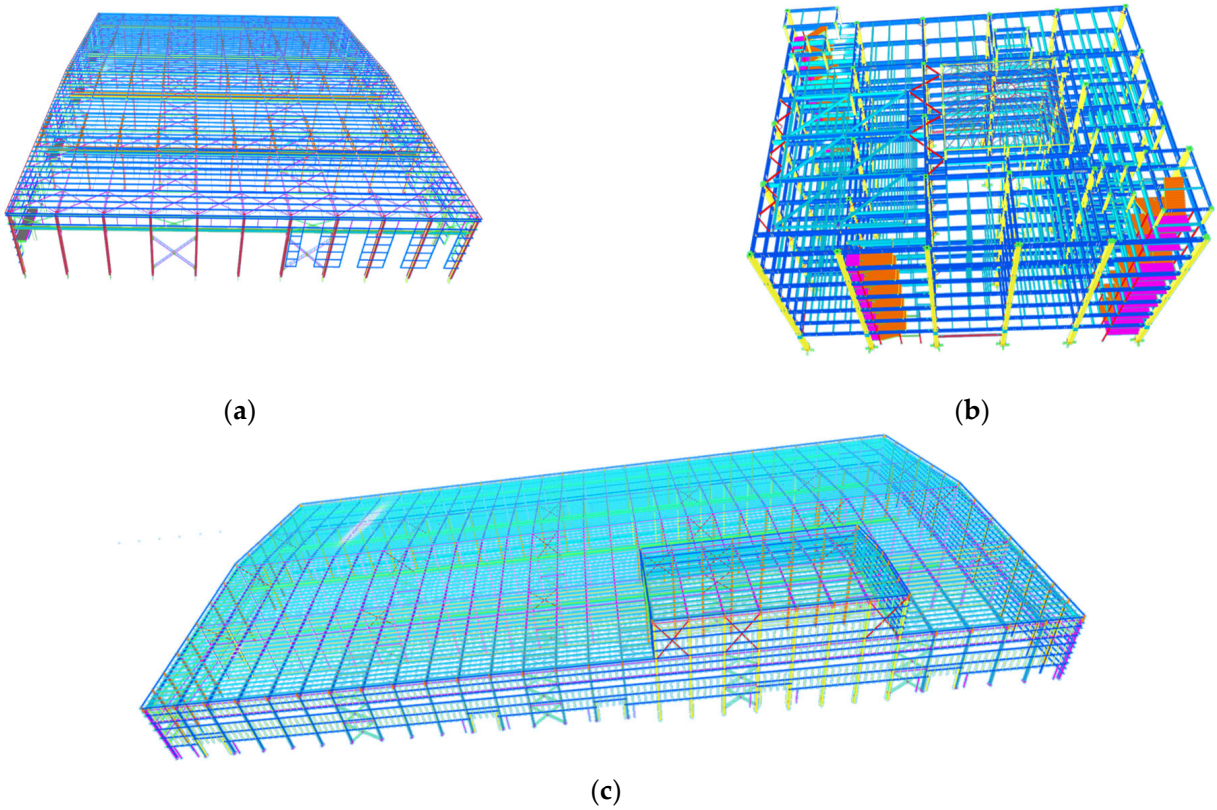


Figure 12. The main interface of the developed BIM system.



(**a**) (**b**)

(**c**)

Figure 13. Three test models: (**a**) model 1 medium-scale steel structural workshop; (**b**) model 2: multi-story steel structural building; and (**c**) model 3: large-scale steel structural workshop.

Table 1. Test model parameters.

Model Category	IFC File Size	Number of Instances in IFC File	Number of Properties	Number of Relationships
Model 1	15.4 MB	29,178	8777	13,483
Model 2	46.4 MB	39,335	9649	13,941
Model 3	61.8 MB	112,897	24,460	44,668

As for the user interface, the developed system provided some interactive functions, such as displaying the component attribute information saved in the IFC file at the user's click position, as shown in Figure 14.

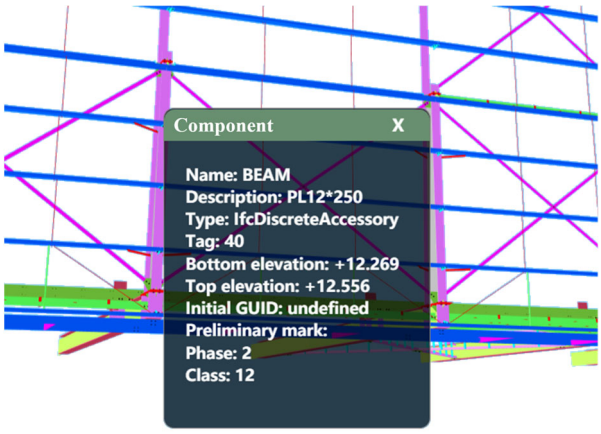


Figure 14. Interactive interface and component information display.

The redundancy eliminate rate r is defined to evaluate the effect of redundancy removal function:

$$r = 1 - \frac{s}{m} \tag{9}$$

where m and s are the numbers of instances with geometric information before and after redundancy elimination, respectively. The redundancy elimination rate denotes the proportion of components identified as redundant in all components of the project, thus verifying the effectiveness of the redundancy removal approach.

Table 2 lists the test results of the system performance by using the proposed approach of component redundancy removal. As indicated in Tables 2 and 3, the size of the created geometry file is almost the same as that of the original IFC file, which guarantees efficient network transmission and small space occupation. In addition, geometric information extraction takes less than 30 s, which can meet the requirement of the quasi-real-time visualization. It can also be seen in Table 3 that the average frame per second (FPS) is over 30 when rendering with low memory consumption. As for the redundancy elimination rate, more than 90% of the model components are identified as redundancy, especially in model 1 and model 3, where the redundancy elimination rates are more than 97%. Therefore, the redundancy elimination approach dramatically improves the performance of the system. For non-redundant instances, each of them is merged into a single drawcall when rendering, significantly reducing the overhead of hardware resources. Above all, the test results show that the proposed algorithms and techniques can greatly improve the system's performance and can support the quasi-real-time geometric information extraction and efficient display for large steel structural models.

Table 2. Test results of the system performance using the approach of component redundancy removal.

Model Type	Geometry File Size	Geometry Conversion Time (s)	FPS	Memory Footprint (MB)	Drawcall Times	Redundancy Elimination Rate
Model 1	18.2	8.22	38	330.6	698	24,310/25,008 = 97.21%
Model 2	57.9	29.57	29	568.4	2249	32,789/35,038 = 93.58%
Model 3	57.8	28.00	33	517.5	1307	91,379/92,686 = 98.59%

Table 3. Test results of system performance using LOD technology.

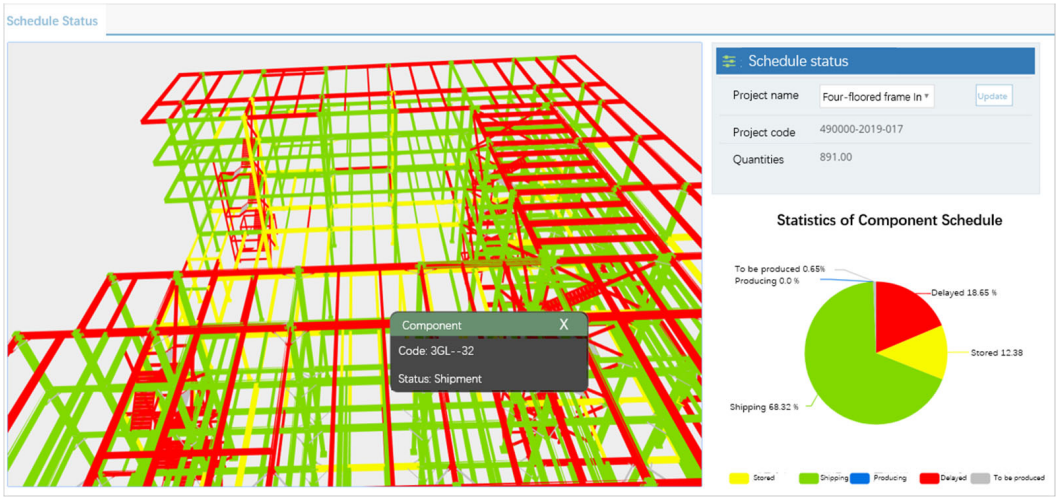
Model Type	Length of Folded Edge (mm)	Instances		Vertices		Patches	
		Number	Simplification Rate	Number	Simplification Rate	Number	Simplification Rate
Model 1	0	25,008	---	7,670,758	---	4,673,856	---
	10	20,247	19.04%	3,017,113	60.67%	890,839	80.94%
	50	14,039	43.86%	519,794	93.22%	74,574	98.40%
Model 2	0	35,038	---	19,967,939	---	11,843,427	---
	10	35038	0.00%	11,648,350	41.66%	3,645,642	69.22%
	50	20,488	41.53%	560,056	97.20%	141,657	98.80%
Model 3	0	92,686	---	19,171,744	---	11,554,938	---
	10	59,889	35.39%	7,787,798	59.38%	2,521,562	78.18%
	50	36,764	60.33%	817,754	95.73%	209,202	98.19%

Taking the above three models as examples, LOD technology was also carried out to simplify their geometric information, and the shortest lengths of the folded edge were 0 mm (without mesh simplification), 10 mm, and 50 mm, respectively. The test results are listed in Table 3, which shows that the number of geometric instances is significantly reduced after simplification, and unnecessary details are omitted when observing from a long distance. It can also be suggested that with the increase in collapsed edge length, the number of instances is still primarily reserved, but the number of vertices and patches to be rendered is greatly reduced. Taking model 1 as an example, when the minimum edge length is 10 mm, 60.67% of vertices and 80.94% of triangular patches are omitted, indicating that some small instances are removed and the rest are simplified. Therefore, the main building components are retained while unnecessary geometric details are removed. The test results show that the triangular meshes to be rendered are greatly reduced, and the GPU load is significantly released.

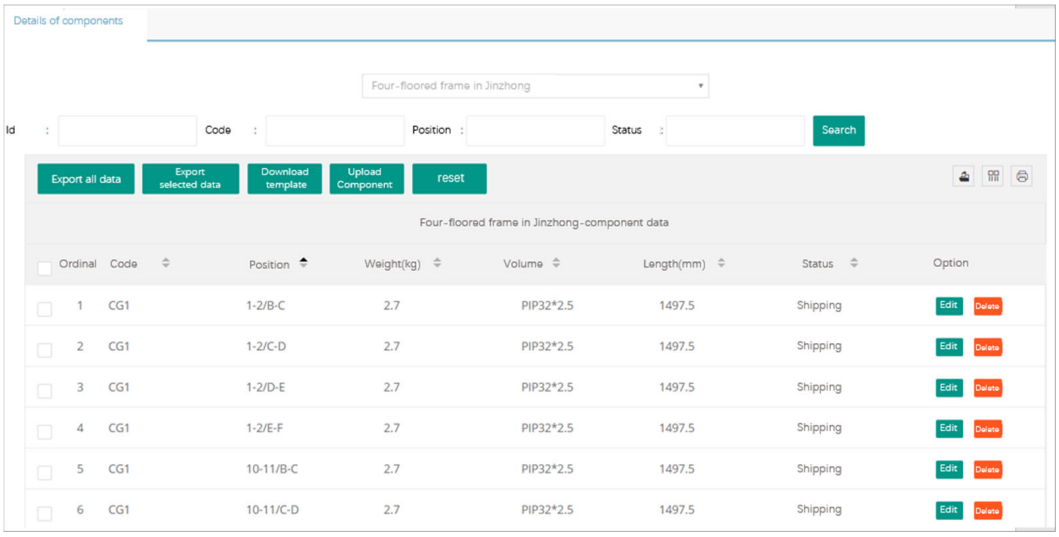
Given that the three models have plenty of internal instances, the test results can reflect the system's performance. In the developed system, the storage and visualization of model information run well, and the components are totally saved. In terms of system performance, the efficiency of information extraction and transformation is high enough to meet the requirement of quasi-real-time analysis. As for rendering, a large number of redundant components are identified, and the multi-level LOD technology has a good simplification effect on the mesh, reaching smooth visualization, fast response, and small hardware overhead.

6.2. System Application

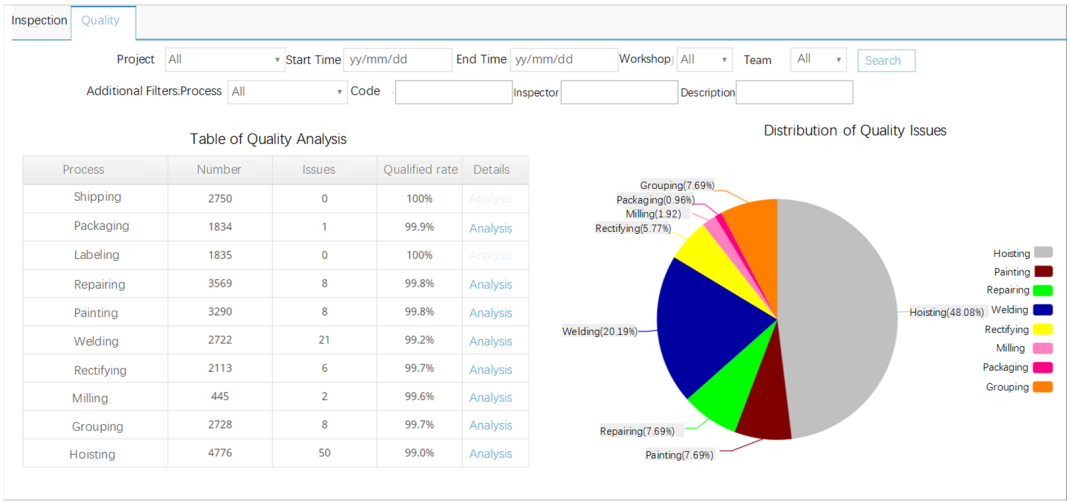
The developed system was applied in the BIM platform for the construction management of steel structures. Figure 15 shows the interfaces of the progress management module, the detailed design module, and the quality management module of the BIM platform.



(**a**)



(**b**)



(**c**)

Figure 15. BIM platform interfaces of (**a**) the progress management module, (**b**) the detailed design module, and (**c**) the quality management module.

In the progress management module, components are displayed in different colors according to their progress status. For instance, green denotes installed components; red denotes delayed components; blue denotes produced components; and yellow denotes components stored in the warehouse. Progress information of components is integrated with the information extracted from the IFC files and displayed through the proposed visualization method. In the detailed design module, key parameters of components, such as weight, length, and location extracted from the IFC files, are shown and can also be edited. In the quality management module, the uploaded data integrated with IFC components are analyzed automatically, and the distribution of quality issues is illustrated by pie charts. The system application results show that the proposed algorithms and techniques can support the implementation of the BIM platform for steel structures.

7. Conclusions

To address the issue of efficient operation of the BIM platform for steel structures, several algorithms and techniques for improving extraction, storage, and visualization of BIM models were presented, and a high-performance BIM platform was developed accordingly. The main work and conclusions of this study are as follows:

(1) Several techniques of information extraction and storage based on IFC files were proposed. Because of the coupled IFC Tree, the complete extraction and lossless storage of building information from the IFC files seriously burden the network transmission. This study proposed several techniques to extract and decouple the geometric and semantic information of components from IFC files. Then, the spatial relationship and geometric information of components were stored in the form of files and cached on the client side, significantly reducing the overhead of database storage and network transmission.

(2) An approach for the redundancy removal of component geometry information was proposed. Because of the large number of similar components in BIM models of steel structures, if the components with the same shape are not specially processed, the processing efficiency of geometry information and the performance of model visualization will be greatly affected. This study proposed an approach combining the PCA algorithm and the Hausdorff-based comparison algorithm to extract the geometric information and remove redundancy in the back end. Additionally, the models are cached in the front end, and the redundant instances are merged when rendering. Thus, the lightweight transmission and highly efficient display of BIM models can be reached.

(3) A loading mechanism of multi-level LOD models was presented. In the BIM models of steel structures, there are beams and columns with larger geometric dimensions, as well as bolts, gusset plates, and welds with small dimensions. Different applied functions and observed positions require different display accuracy. Based on the edge collapsing algorithm, this study presented a loading mechanism of multi-level LOD models, which can be customized by users, significantly improving the display efficiency of models in coarse-grained visualization.

The developed system was tested by using three steel structural models. By adopting the redundancy removal approach, the number of instances is decreased by 96.46% in less than 30 s and over 30 FPS is kept on the test machine when rendering. Adopting the LOD loading mechanism, 95.38% of vertices and 98.46% of patches are eliminated under 50 mm precision. Based on the optimized performance, a BIM platform for the construction management of steel structures was developed and applied to actual projects. The test results showed that the platform has good performance and can support the management of large-scale steel structural projects.

Despite some progress made in developing a high-performance BIM platform for steel structures, this study has several limitations. The volume of semantic data extracted from IFC files remains substantial, posing potential network and database performance issues, especially when multiple models are loaded simultaneously. Additionally, the

methods tailored specifically to steel structures may not be directly applicable to other types such as concrete structures, limiting broader industry applicability. The multi-level LOD model loading mechanism requires further refinement to handle very large and complex models without compromising visual accuracy or system performance. Furthermore, the current system has limited performance in managing multiple large models, necessitating improvements in caching mechanisms and version control. The inefficiencies in caching and version control can lead to slower data retrieval and potential inconsistencies. Future research should focus on optimizing semantic data management, refining LOD algorithms, and enhancing system architecture to ensure scalability, reliability, and broader applicability across various structural types.

Author Contributions: Conceptualization, J.W. and C.W.; methodology, J.W. and Z.S.; software, C.W.; validation, C.W.; investigation, J.W., Z.S. and C.W.; resources, Z.S.; data curation, C.W.; project administration, Z.S.; writing—original draft, C.W. and Z.S.; writing—review and editing, J.W.; visualization, C.W.; supervision, J.W.; funding acquisition, J.W. and Z.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Key R&D Program of China [2023YFC3805700], Shanghai Qi Zhi Institute [SYXF0120020109], and Tongji University Interdisciplinary Research Project [2023-2-YB-06].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The contributions of Anpeng Qiang and Zhongnuo Hong in resources and project administration are gratefully acknowledged.

Conflicts of Interest: Author Zhiguo Sun was employed by CCCC Construction Group Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Lacey, A.W.; Chen, W.; Hao, H.; Bi, K. Structural response of modular buildings—An overview. *J. Build. Eng.* **2018**, *16*, 45–56. [CrossRef]
2. Zhou, T.; Sun, K.; Chen, Z.; Yang, Z.; Liu, H. Automated optimum design of light steel frame structures in Chinese rural areas using building information modeling and simulated annealing algorithm. *Sustainability* **2023**, *15*, 9000. [CrossRef]
3. Laefer, D.F.; Truong-Hong, L. Toward automatic generation of 3D steel structures for building information modelling. *Autom. Constr.* **2017**, *74*, 66–77. [CrossRef]
4. Kineber, A.F.; Othman, I.; Famakin, I.O.; Oke, A.E.; Hamed, M.M.; Olayemi, T.M. Challenges to the implementation of building information modeling (BIM) for sustainable construction projects. *Appl. Sci.* **2023**, *13*, 3426. [CrossRef]
5. Feng, H.; Kassem, M.; Greenwood, D.; Doukari, O. Whole building life cycle assessment at the design stage: A BIM-based framework using environmental product declaration. *Int. J. Build. Pathol. Adapt.* **2022**, *41*, 109–142. [CrossRef]
6. Matarneh, S.T.; Danso-Amoako, M.; Al-Bizri, S.; Gaterell, M.; Matarneh, R. Building information modeling for facilities management: A literature review and future research directions. *J. Build. Eng.* **2019**, *24*, 100755. [CrossRef]
7. Charef, R.; Emmitt, S.; Alaka, H.; Fouchal, F. Building Information Modelling adoption in the European Union: An overview. *J. Build. Eng.* **2019**, *25*, 100777. [CrossRef]
8. Olugboyega, O. Differential relationships in the BIM implementation process in a developing country: The role of essential BIM implementation strategies. *Eng. Constr. Archit. Manag.* **2023**, 3. [CrossRef]
9. Eastman, C.; Teicholz, P.; Sacks, R.; Liston, K. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, 2nd ed.; Wiley: New York, NY, USA, 2011.
10. *Etabs 21.1.0*, Computer Software; Computers and Structures, Inc.: Walnut Creek, CA, USA, 2023.
11. *Midas Gen 2023*, Computer Software; MIDAS Information Technology Co., Ltd.: Gyeongbuk, Republic of Korea, 2023.
12. *Tekla 2022*, Computer Software; Trimble Co.: Espoo, Finland, 2022.
13. Chen, Z.; Chen, L.; Zhou, X.; Huang, L.; Sandanayake, M.; Yap, P.-S. Recent technological advancements in BIM and LCA integration for sustainable construction: A review. *Sustainability* **2024**, *16*, 1340. [CrossRef]
14. BIMserver. Open Building Information Model Server. Available online: https://bimserver.org (accessed on 20 April 2024).
15. xBIM. xBIM Toolkit. Available online: https://docs.xbim.net (accessed on 20 April 2024).

16. Xu, Z.; Zhang, L.; Li, H.; Lin, Y.H.; Yin, S. Combining IFC and 3D tiles to create 3D visualization for building information modeling. *Autom. Constr.* **2020**, *109*, 102995. [CrossRef]
17. Zhou, X.; Wang, J.; Guo, M.; Gao, Z. Cross-platform online visualization system for open BIM based on WebGL. *Multimed. Tools Appl.* **2019**, *78*, 28575–28590. [CrossRef]
18. Wang, Y.; He, X.; He, J. Virtual trial assembly of steel structure based on BIM platform. *Autom. Constr.* **2022**, *141*, 104395. [CrossRef]
19. Zhou, M.; Wang, J.; Yu, B.; Chen, K. A quality management method for prefabricated building desig n based on BIM and VR-integrated technology. *Appl. Sci.* **2024**, *14*, 1635. [CrossRef]
20. Jorgensen, K.A.; Skauge, J.; Christiansson, P.; Svidt, K.; Sørensen, K.B.; Mitchell, J. Use of IFC Model Servers Modeling Collaboration Possibilities in Practice. Available online: https://adk.elsevierpure.com/ws/files/74831/ReportIfcModelServer-Final.pdf (accessed on 20 April 2024).
21. Jotne, I. EDMmodelServer (ifc). Available online: https://jotneconnect.com/products/edmmodelservermanager (accessed on 20 April 2024).
22. Ayman, H.M.; Mahfouz, S.Y.; Alhady, A. Integrated EDM and 4D BIM-based decision support system for construction projects control. *Buildings* **2022**, *12*, 315. [CrossRef]
23. *Revit 2023*, Computer Software; Autodesk, Inc.: San Rafael, CA, USA, 2023.
24. *Bentley STAAD.Pro 2023*, Computer Software; Bentley Systems, Incorporated: Exton, PA, USA, 2023.
25. BIMcloud. BIM Server-Online Collaboration Tool for Architects & Engineers. Available online: https://www.graphisoft.com/bim-server (accessed on 20 April 2024).
26. Zhang, S.; Pan, F.; Wang, C.; Sun, Y.; Wang, H. BIM-based collaboration platform for the management of EPC projects in hydropower engineering. *J. Constr. Eng. Manag.* **2017**, *143*, 04017087. [CrossRef]
27. Ma, Z.; Zhang, D.; Li, J. A dedicated collaboration platform for Integrated Project Delivery. *Autom. Constr.* **2018**, *86*, 199–209. [CrossRef]
28. Rahimian, F.P.; Chavdarova, V.; Oliver, S.; Chamo, F. OpenBIM-Tango integrated virtual showroom for offsite manufactured production of self-build housing. *Autom. Constr.* **2019**, *102*, 1–16. [CrossRef]
29. Park, J.; Cai, H.; Dunston, P.S.; Ghasemkhani, H. Database-supported and web-based visualization for daily 4D BIM. *J. Constr. Eng. Manag.* **2017**, *143*, 04017078. [CrossRef]
30. Ma, Z.; Liu, Z. Ontology- and freeware-based platform for rapid development of BIM applications with reasoning support. *Autom. Constr.* **2018**, *90*, 1–8. [CrossRef]
31. Xu, Z.; Zhang, Y.; Xu, X. 3D visualization for building information models based upon IFC and WebGL integration. *Multimed. Tools Appl.* **2016**, *75*, 17421–17441. [CrossRef]
32. Bourahla, N.; Tafraout, S.; Bourahla, Y.; Sereir-El-Hirtsi, A.; Skoudarli, A. GA based design automation and optimization of earthquake resisting CFS structures in a BIM environment. *Structures* **2022**, *43*, 1334–1341. [CrossRef]
33. Morshed, S.A.; Lv, X.; Tanvir, R.B. Network-based information extraction from IFC files to support intelligent BIM companion (iBcom) technology. In Proceedings of the ASCE Construction Research Congress, College Avenue Commons, Tempe, AZ, USA, 8 March 2020; pp. 427–435. [CrossRef]
34. Yu, Y.; Kim, S.; Jeon, H.; Koo, B. A systematic review of the trends and advances in IFC schema extensions for BIM interoperability. *Appl. Sci.* **2023**, *13*, 12560. [CrossRef]
35. Sibenik, G.; Kovacic, I. Assessment of model-based data exchange between architectural design and structural analysis. *J. Build. Eng.* **2020**, *32*, 101589. [CrossRef]
36. Du, X.; Gu, Y.; Yang, N.; Yang, F. IFC file content compression based on reference relationships. *J. Comput. Civ. Eng.* **2020**, *34*, 04020012. [CrossRef]
37. Artus, M.; Alabassy, M.S.H.; Koch, C. A BIM based framework for damage segmentation, modeling, and visualization using IFC. *Appl. Sci.* **2022**, *12*, 2772. [CrossRef]
38. Khalili, A.; Chua, D.K.H. IFC-based graph data model for topological queries on building elements. *J. Comput. Civ. Eng.* **2015**, *29*, 04014046. [CrossRef]
39. Zhu, J.; Wang, X.; Wang, P.; Wu, Z.; Kim, M.J. Integration of BIM and GIS: Geometry from IFC to shapefile using open-source technology. *Autom. Constr.* **2019**, *102*, 105–119. [CrossRef]
40. Afsari, K.; Eastman, C.M.; Castro-Lacouture, D. JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange. *Autom. Constr.* **2017**, *77*, 24–51. [CrossRef]
41. Chen, H.M.; Chang, K.C.; Lin, T.H. A cloud-based system framework for performing online viewing, storage, and analysis on big data of massive BIMs. *Autom. Constr.* **2016**, *71*, 34–48. [CrossRef]
42. Solihin, W.; Eastman, C.; Lee, Y.C.; Yang, D.H. A simplified relational database schema for transformation of BIM data into a query-efficient and spatially enabled database. *Autom. Constr.* **2017**, *84*, 367–383. [CrossRef]
43. World Wide Web Consortium (W3C). OWL Web Ontology Language Reference. Available online: https://www.w3.org/TR/owl-ref/ (accessed on 28 May 2024).
44. Cohen-Or, D.; Chrysanthou, Y.L.; Silva, C.T.; Durand, F. A survey of visibility for walkthrough applications. *IEEE Trans. Vis. Comput. Graph.* **2003**, *9*, 412–431. [CrossRef]

45. Mattausch, O.; Bittner, J.; Wimmer, M. CHC++: Coherent hierarchical culling revisited. *Comput. Graph. Forum* **2008**, *27*, 221–230. [CrossRef]
46. Terry, J.; Stantic, B. Indexing method for multidimensional vector data. *Comput. Sci. Inf. Syst.* **2013**, *10*, 1077–1104. [CrossRef]
47. Liu, T.; Chaudhuri, S.; Kim, V.G.; Huang, Q.; Mitra, N.J.; Funkhouser, T. Creating consistent scene graphs using a probabilistic grammar. *ACM Trans. Graph.* **2014**, *33*, 1–12. [CrossRef]
48. Xia, J.C.; El-Sana, J.; Varshney, A. Adaptive real-time level-of-detail based rendering for polygonal models. *IEEE Trans. Vis. Comput. Graph.* **1997**, *3*, 171–183. [CrossRef]
49. Leite, F.; Akcamete, A.; Akinci, B.; Atasoy, G.; Kiziltas, S. Analysis of modeling effort and impact of different levels of detail in building information models. *Autom. Constr.* **2011**, *20*, 601–609. [CrossRef]
50. Sun, J.; Liu, Y.S.; Gao, G.; Han, X.G. IFCCompressor: A content-based compression algorithm for optimizing Industry Foundation Classes files. *Autom. Constr.* **2015**, *50*, 1–15. [CrossRef]
51. Liu, X.; Xie, N.; Tang, K.; Jia, J. Lightweighting for Web3D visualization of large-scale BIM scenes in real-time. *Graph. Models* **2016**, *88*, 40–56. [CrossRef]
52. Hu, Z.Z.; Yuan, S.; Benghi, C.; Zhang, J.P.; Kassem, M. Geometric optimization of building information models in MEP projects: Algorithms and techniques for improving storage, transmission and display. *Autom. Constr.* **2019**, *107*, 102941. [CrossRef]
53. Bouaziz, S.; Tagliasacchi, A.; Pauly, M. Sparse iterative closest point. *Comput. Graph. Forum* **2013**, *32*, 113–123. [CrossRef]
54. East, E.W.; Nisbet, N.; Liebich, T. Facility management handover model view. *J. Comput. Civ. Eng.* **2013**, *27*, 61–67. [CrossRef]
55. Lee, Y.C.; Shariatfar, M.; Ghannad, P.; Zhang, J.; Lee, J.K. Generation of entity-based integrated model view definition modules for the development of new BIM data exchange standards. *J. Comput. Civ. Eng.* **2020**, *34*, 04020011. [CrossRef]
56. Liu, H.; Gao, G.; Zhang, H.; Liu, Y.S.; Song, Y.; Gu, M. MVDLite: A fast validation algorithm for Model View Definition rules. *Adv. Eng. Inform.* **2023**, *58*, 102132. [CrossRef]
57. Sellers, G.; Wright, R.S.; Haemel, N. *OpenGL SuperBible: Comprehensive Rutorial and Reference*, 7th ed.; Addison-Wesley Professional: Redding, MA, USA, 2015.
58. buildingSMART. Industry Foundation Classes (IFC)—An Introduction. Available online: https://technical.buildingsmart.org/standards/ifc/ (accessed on 20 April 2024).
59. Jolliffe, I.T. *Principal Component Analysis*, 2nd ed.; Springer: New York, NY, USA, 2002.
60. Besl, P.J.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
61. Huttenlocher, D.P.; Klanderman, G.A.; Rucklidge, W.J. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.* **1993**, *15*, 850–863. [CrossRef]
62. Clark, J.H. Hierarchical geometric models for visible-surface algorithms. *Commun. ACM* **1976**, *19*, 547–554. [CrossRef]
63. Cohen, J.; Varshney, A.; Manocha, D.; Turk, G.; Weber, H.; Agarwal, P.; Brooks, F.; Wright, W. Simplification envelopes. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, Association for Computing Machinery, New York, NY, USA, 1 August 1996; pp. 119–128. [CrossRef]
64. Schroeder, W.J.; Zarge, J.A.; Lorensen, W.E. Decimation of triangle meshes. *SIGGRAPH Comput. Graph.* **1992**, *26*, 65–70. [CrossRef]
65. Rossignac, J.; Borrel, P. *Multi-Resolution 3D Approximations for Rendering Complex Scenes*; Modeling in Computer Graphics; Springer: Berlin, Germany, 1993; pp. 455–465. [CrossRef]
66. Hamann, B. A data reduction scheme for triangulated surfaces. *Comput. Aided Geom. Des.* **1994**, *11*, 197–214. [CrossRef]
67. Kalvin, A.D.; Taylor, R.H. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Comput. Graph. Appl.* **1996**, *16*, 64–77. [CrossRef]
68. Garland, M.; Heckbert, P.S. Surface simplification using quadric error metrics. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques; ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 1997; pp. 209–216. [CrossRef]
69. IfcOpenShell. The Open Source ifc Toolkit and Geometry Engine. Available online: https://ifcopenshell.org/ (accessed on 20 April 2024).
70. glTF. Runtime 3D Asset Delivery. Available online: https://www.khronos.org/gltf/ (accessed on 20 April 2024).
71. IFC Engine. The Fastest and the Most Flexible Solution to Support IFC. Available online: http://rdf.bg/product-list/ifc-engine/ (accessed on 20 April 2024).