文章编号:1671-4598(2022)09-0177-07 DOI:10.16526/j.cnki.11-4762/tp.2022.09.027 中图分类号:TP391 文献标识码:A

基于 Web 的 BIM 模型轻量化展示及渲染控制研究

王 琪,李智杰,李昌华,张 颉
(西安建筑科技大学 信息与控制工程学院,西安 710055)

摘要:针对当前 BIM 模型数据量庞大且需要专业软件查看,对计算机软硬件要求苛刻以及在 C/S 架构进行模型展示时十分不便的问题,提出了一种 BIM 模型的轻量化展示及自适应渲染控制的方法;该方法以 Revit 模型为研究对象,首先利用二次开发技术编写插件自动提取模型信息并转化成 glTF 格式;然后通过数据优化算法压缩 glTF 格式,并利用 WebGL 技术以及 Three. js 框架解析 glTF 格式实现模型的 Web 端重组;最后通过 LOD-AD 优化算法,实现距离较远且体积较小的三角面片剔除,并且帧率较低时优先渲染高层级构件,以此完成渲染过程中的自适应控制效果;实验结果表明,该方法在确保模型保真度的前提下,极大减少了模型体量以及模型传输时间,且场景的自适应控制能有效剔除低重要度构件,达到显著提升渲染流畅度的效果,为 BIM 模型轻量化展示以及渲染控制提供了一种可行的参考方案。

关键词:BIM 模型轻量化;glTF;WebGL;模型数据;渲染控制

Research on Web-based BIM Model Lightweight Display and Rendering Control

WANG Qi, LI Zhijie, LI Changhua, ZHANG Jie
(School of Information and Control Engineering, Xi'an University of Architectural Science and Technology, Xi'an 710055, China)

Abstract: Aiming at the problems that the current BIM model data is huge and needs professional software to view, it has strict requirements for computer software and hardware, and it is very inconvenient to display the model in the C/S architecture. A lightweight display of Building information model (BIM) model and adaptive rendering control are proposed. This method takes the Revit model as the research object, the secondary development technology is firstly used to write a plug-in to automatically extract the model information and convert it into the glTF format; then the glTF format is compressed through the data optimization algorithm, and the WebGL technology and the Three. js framework are used to parse the glTF format to realize the Web side reorganization of the model. Finally, through the LOD-AD optimization algorithm, the triangular patches with a long distance and a small volume are eliminated, and high-level components are preferentially rendered when the frame rate is low, so as to complete the adaptive control effect in the rendering process. The experimental results show that this method greatly reduces the model volume and model transmission time under the premise of ensuring model fidelity, and the adaptive control of the scene can effectively eliminate the low-importance components and achieve the effect of significantly improving rendering fluency, which provides a feasible reference solution for lightweight display and rendering control of the BIM models.

Keywords: BIM model lightweight; glTF; WebGL; model data; render control

0 引言

建筑信息模型(BIM, building information model)在建筑的设计、施工、运维的全生命周期中具有极高的应用价值[1]。BIM 技术在国内外都有良好的发展,但是对于目前建筑业的需要,建筑体量也是愈来愈庞大,更多的信息数据导致 BIM 软件对于计算机的软硬件要求很高,而且 BIM 的实现大多是基于 C/S 架构的,需要用计算机操作并且安装专业的 BIM 软件,这严重影响了 BIM 的发展。随着 Web 端以及 3D 渲染引擎的迅速崛起,对 BIM 模型的 Web 端渲染提供了一种可行性方案[2]。在信息化社会,移动端已成为主流,所以将 BIM 模型以 B/S 架构可视化渲染,并摆脱专有软件、终端平台的约束,这对于 BIM 的推动以及轻量化研究都有重大意义[3]。因此,对于 BIM 模型数据的轻量化以

收稿日期:2022-04-13; **修回日期**:2022-05-07。
基金项目:国家自然科学基金(61373112, 51878536);陕西省自然科学基金(2020JQ-687);陕西省住房城乡建设科技计划项目(2020-K09)。
作者简介:王 琪(1996-),男,安徽人,硕士研究生,主要从事智能建筑、BIM 模型轻量化等方向的研究。
李智杰(1980-),男,河南人,博士,副教授,硕士生导师,主要从事模式识别、数字建筑等方向的研究。
李昌华(1963-),男,宁夏人,博士,教授,博士生导师,主要从事图形图像处理、模式识别、数字建筑等方向的研究。
引用格式:王 琪,李智杰,李昌华,等. 基于 Web 的 BIM 模型轻量化展示及渲染控制研究[J]. 计算机测量与控制,2022,30(9):177-183.

及将其放置在 Web 端进行渲染优化都是十分必要的。

目前在 Web 端实现 BIM 模型轻量化展示以及场景渲染控制，国内外研发部门和学者已有相关研究。Autodesk 所研发的 Forge 云平台以及广联达推出的协筑平台都包含了 BIM 模型网页端浏览功能，但是不提供用户需要的开发接口且存在敏感数据安全隐患问题。为此，学者们通过建立私有云的方式改善了该问题，文献［4－6］实现了 BIM 模型在 Web 端展示，并提供了模型轻量化的实现方案，但是对于模型数据的优化以及场景渲染效果不佳。文献［7－8］从模型三角面片入手，提出了模型简化边折叠算法，将三角网格简化降低面片复杂度，但是影响了模型外观。文献［9－10］通过剔除视野范围内的不可见构件，降低了场景的 GPU 开销，然而剔除计算过程又极其复杂，严重降低了渲染速度，文献［11］提出了八叉树剔除算法，解决了这一难题，然而构建八叉树的过程会极大的消耗存储空间。

鉴于此，本文提出了一种基于 Web 的 BIM 模型轻量化展示及渲染控制方法，对于模型数据优化，通过冗余顶点删除法和 Draco 算法来压缩模型数据，在确保保真度的前提下，将模型数据体量以及网页端加载时间降低；对于渲染优化，提出了一种 LOD 与自适应渲染相结合的 LOD-AD 优化算法来对场景内构件进行渲染控制，有效剔除低重要度构件。实验结果表明，本文方法极大压缩了模型体量，并且在帧率较低的情况下自适应选择次要图元的剔除以及重要图元的优先渲染，在不影响模型外观情况下渲染流畅度大大提升，从而实现三维模型在 Web 端的轻量化显示及渲染优化。

1 Revit 模型数据优化

1.1 Revit 模式数据转换

Revit 是 Autodesk 公司推出的一款 BIM 软件，在建筑业内盛行应用它来作为 BIM 建模和信息化管理工具，具有较好的数据完整性和兼容性，能够很好的达到工作需求，并且 Revit 提供了完善的开发接口，供用户自行开发需要的功能。Revit 模型中含有大量的数据结构，其中包含了模型的名称、位置、几何、属性信息等，当然也包含了很多冗余信息，因此提取重要信息减少模型数据以实现模型的轻量化具有一定的工程意义。本文主要实现模型中的几何信息以及属性信息的提取，并将它们转化为轻量化数据结构。

几何信息中含有大量的组成建筑图元的面、点、边等信息，属性信息中含有模型的材质、纹理、标高等信息[12]。对于几何信息，将这些点、线、面转化为渲染所需的最基本单位——三角面片的形式存储，包括三角面片的顶点坐标、顶点索引、法线向量、法线向量索引、几何体中心、唯一标识符 ID[13]；对于属性信息，可以在单独导出材质贴图后，直接提取出模型的 name、userData、标识符 ID，以上的几何信息和属性信息，都以数组的形式存储以此来建立轻量化数据结构。

在轻量化实现过程中，将 Revit 模型展现至 Web 端，需要一个中间格式作为桥梁。glTF（Graphics Language Transmission Format，图形语言传输格式）是跨平台的 3D 对象通用标准[14]，该格式在 Web 端的解析与加载更加便捷迅速，因此本文采用 glTF 格式作为 Revit 和 Web 端的传输格式。

对于 Revit 数据的导出，可利用其强大的开发接口，采用 Revit 二次开发技术实现模型信息的提取，其中 IExpotContext 接口中含有所有模型数据的导出函数，因此本文通过调用该接口中含有的函数来完成模型信息的提取并转换为 glTF 格式，具体的 Revit 模型的数据导出流程如图 1 所示。

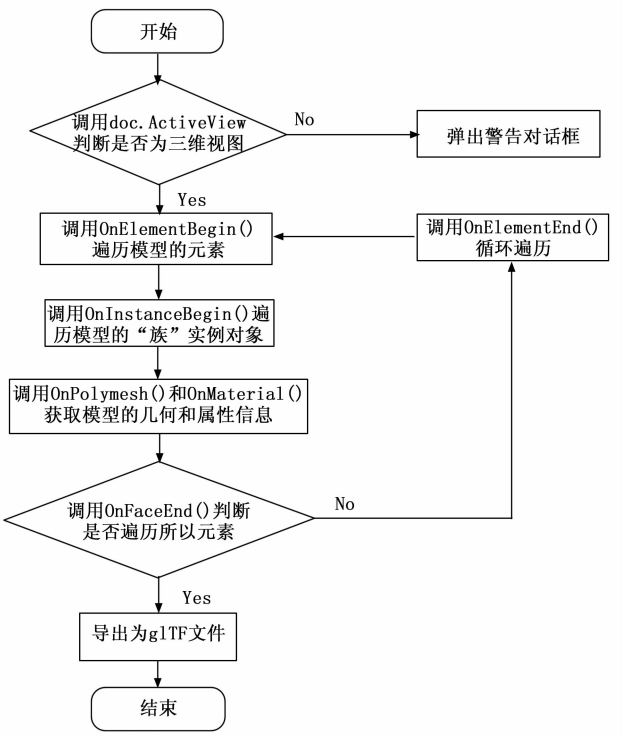


图 1 Revit 模型的数据导出

如图 1 所示，数据在导出之前会首先判断当前的 Revit 是否在三维视图中，若不是会报错并终止模型插件的继续运行，若在三维视图中，模型会通过 OnElementBegin（）以及 OnIstanceBegin（）函数遍历界面中所有的元素（element）和族实例（Instance），然后利用 OnPolymesh（）和 OnMaterial（）函数来提取所有元素的几何、属性信息，最后调用 OnFaceEnd（）函数完成所有几何体的循环，将模型信息导出为 glTF 格式。

1.2 模型数据轻量化

虽然建立了轻量化数据结构，但模型文件中依旧存在着冗余数据，降低了模型解析率和传输速度，导致加载缓慢甚至加载失败，严重影响用户体验，因此模型数据轻量化尤为重要，需要对 glTF 模型文件作进一步优化处理。在 glTF 文件中包含了大量的三角面片，而三角面片是由无数个顶点信息组成，顶点之间也是依靠连接关系相连，因此模型的压缩可以从顶点删除、顶点连接关系压缩方面来研

究。本文主要通过冗余顶点删除法和 Draco 网格压缩算法实现模型数据的优化。

1）冗余顶点删除法。

Revit 构件的几何信息在保存至 glTF 文件中均以三角面片形式存储，三角面片都是由顶点信息组成，以减少运行内存的使用空间，但是文件内部依旧含有大量的冗余顶点，例如模型边界某顶点位于其相邻两个顶点之间且有可能三点共线或者该点与另外两定点之间的夹角在一定范围内并不影响模型外观的展示，则该顶点可称为冗余顶点。如图 2 所示，点 E 在边 BC 上，点 D 在边 BC 下且角 D 接近 180°，此时点 E 与 D 都是冗余顶点，删除后模型外观也基本不变。

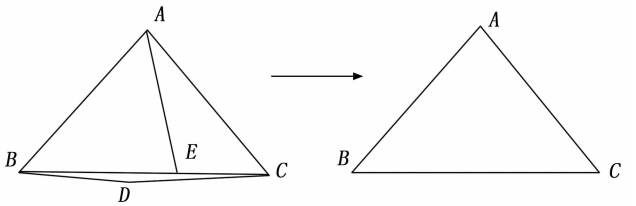


图 2　冗余顶点删除过程图

对于冗余顶点的判断，可以通过顶点 V 与相邻两个顶点连线所组成的边 e_1 和 e_2，其向量为 $\vec{e}_1$ 和 $\vec{e}_2$，并规定向量方向统一指向顶点 v，可依据式（1）计算向量与向量之间的夹角 α。

$$\alpha = \arccos\left(\frac{\vec{e}_1 \cdot \vec{e}_2}{|\vec{e}_1||\vec{e}_2|}\right) \tag{1}$$

设置阈值 ω，若 $\alpha \geqslant \omega$，则表明该顶点影响因子很小，可将改点视为冗余顶点并剔除该顶点；反之，则保留该顶点。三维模型中的剔除原理类似，将顶点相邻三条边向量化并判断夹角与阈值的大小关系进行剔除即可。

2）Draco 压缩算法。

Draco 是由谷歌提出的一种用于压缩和解压三维几何网格数据的库[15]。其算法主要原理是将三维模型在传输或存储的过程中，压缩模型中的数据信息。Draco 中主要使用了经典三维网格压缩算法 Edgebreaker[16]。该算法的网格访问是基于区域增长原理的，在访问网格的过程中，需始终维持一个由边组成的有向边界，该边界将网格划分为两类，即已访问和未访问部分，并且每访问一个多边形，则会输出一个该多边形与边界的拓扑关系操作符，并将该多边形纳为已编码部分，至此三角面的压缩完成[17]。如图 3 表示了压缩过程中的五种基本操作符。

在图中，X 表示当前遍历的三角形，v 表示被遍历三角形的第三顶点。图 3（a）中的操作符 L 表示左侧三角形已经被遍历过，右侧三角形没有被遍历过；图 3（b）中的操作符 C 表示左右两侧三角形都没有被遍历过；图 3（c）中的操作符 R 表示右侧的三角形已被遍历过，而左侧的三角形没有被遍历过；图 3（d）中的操作符 S 表示左右两侧三角形都没有被遍历过；图 3（e）中的操作符 E 表示左右两侧三角形都被遍历过。

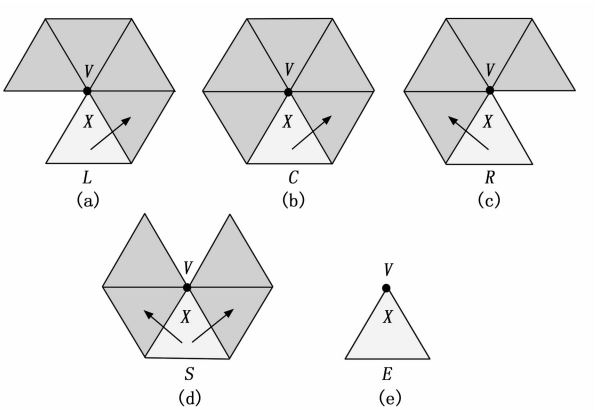


图 3　Edgebreaker 的五种基本操作符

通过这五种基本操作符描述了利用有向环形将网格分割成数条路径的方法，图 4 是对一个不规则图形的三角网格模型进行 Edgebreaker 压缩的示意图，图中的箭头表示压缩时所经过的路径。

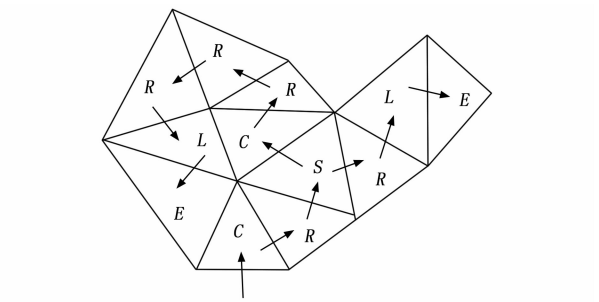


图 4　Edgebreaker 网格分割路径示例

本文利用冗余顶点删除法和 Draco 压缩算法将提取出来的 Revit 模型数据的几何信息进行压缩，降低了模型的数据体量，提高了模型文件在 Web 端的传输速度，并在此基础上保证模型的高保真度。

2　Revit 模型的 Web 端渲染

2.1　模型映射

glTF 数据中包含的几何数据中含有各个三角面片的局部坐标系，而 WebGL 中的所有物体在全局坐标系中都有精确的位置坐标，因此导出的 glTF 模型在 Web 上显示时必须进行坐标系变换，坐标变换如图 5 所示。

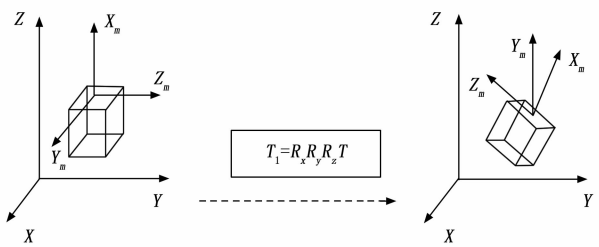


图 5　坐标变换

从全局坐标系到世界坐标系的转变，实际是空间几何图形的平移、缩放与旋转的单独或组合变换而成[18]。两坐标系间的过渡矩阵计算如下：

$$\boldsymbol{T}_1 = \boldsymbol{R}_x\boldsymbol{R}_y\boldsymbol{R}_z\boldsymbol{T} \tag{2}$$

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\boldsymbol{R}_y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\boldsymbol{R}_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_z & \sin\theta_z & 0 \\ 0 & -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_1 & y_1 & z_1 & 1 \end{bmatrix} \tag{6}$$

其中：假设 (x_1,y_1,z_1) 为 glTF 格式中的局部坐标系原点在 Web 端全局坐标系中的坐标；局部坐标系的 Z 轴垂直于地面；Y 轴指向正北；R_x,R_y,R_z 为绕 X 轴、Y 轴、Z 轴旋转得到的旋转矩阵；$\theta_x,\theta_y,\theta_z$ 为旋转后与 X 轴、Y 轴、Z 轴的交角。综上，glTF 文件中的几何属性可依据式（2）来完成空间坐标系的转换，使构件的几何信息能在 Web 端完成关系映射。

2.2 模型的 Web 端重建

完成了 Revit 导出数据与 WebGL 的空间关系映射后需要将文件中的模型信息在 Web 端重组并展示出来。Three.js 是 WebGL 技术的一种表现方式，其代码体量小、加载迅速、操作简单[19]。因此本文选择 Three.js 作为渲染框架来展示模型，基本渲染流程如图 6 所示。

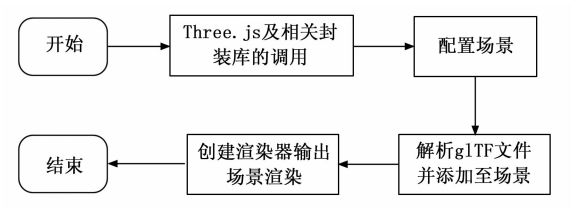


图 6 Three.js 渲染流程图

在三维模型显示的过程中，首先引入 Three.js 及相关功能的类库，以便于开发人员通过库内函数的直接调用完成场景中对象的创建；之后完成场景的建立并对场景内的元素进行加载，包括了设置相机属性、灯光属性、渲染器基本参数等；接着通过 Three.js 框架自带的 glTFLoader 函数解析 glTF 文件，并将所有组成模型的三角面片转化为对象添加至场景中；最后通过渲染器渲染场景内所有组件元素，完成几何模型在 WebGL 上的加载。

3 场景渲染控制算法

3.1 LOD 场景控制算法

LOD（Level Of Detail，层次细节）[20]是为简化场景中可见几何体的多边形细节而提出的一类加速算法。视觉原则中，距离视点越远的模型在计算机上所占面积越小，对几何体的影响也最小，因此提出的模型绘制方法是对近距离几何体绘制详细的细节，而对远距离的几何体进行简化，这便是 LOD 技术的原理[21]。

本文提出了一种最小包围球的 LOD 优化算法对场景进行优化，即大面片以及距离视点近的面片显示渲染，而距离视点远以及小面片隐藏剔除。由于模型三角面片规则形状不一，很难判断其大小，因此可以使用最小包围球的概念，用圆形来包围住三角面片，通过圆形的半径表示面片的大小。用场景中的三角面片最小包围球的半径 r 来表示该三角面片的大小；包围球圆心到相机的距离 d 表示该三角面片距离视点的距离，d 可通过式（7）来确定，其中 (x_0,y_0,z_0) 表示相机坐标，(x,y,z) 表示三角面片包围球圆心坐标；TH 表示设置的初始阈值。通过式（8）来判断是否渲染此三角面片，当圆形半径与圆心到相机的距离之比小于等于阈值时隐藏且不渲染该三角面片，当圆形半径与圆心到相机的距离大于阈值时显示且渲染该三角面片。

$$d = \sqrt{(x_0-x)^2+(y_0-y)^2+(z_0-z)^2} \tag{7}$$

$$\frac{r}{d} \leqslant TH \tag{8}$$

3.2 LOD-AD 场景控制算法

在 Web 端渲染 BIM 模型时，会占用计算机的 GPU 资源，其性能指标可利用 FPS（Frames per Second，帧率）来表达，即 GPU 每秒所刷新的次数，通常情况下 FPS 的上限值设定为 60 FPS，更高的帧率可以使模型渲染更为流畅。对于 Web 端的模型渲染，30 FPS 已能足够流畅展示三维场景，更高的帧率可提高用户使用场景交互功能时的体验感。

在自定义导出的 glTF 格式中导出 Revit 的模型 Category.name 来确定模型构件类型，通过模型构件类型的重要程度划分模型层级，引入帧率来自适应剔除层级类型，本文设置的层级如表 1 所示。

表 1 层级划分

模型层级	渲染对象
第一层级	所有族类型
第二层级	墙、楼板、柱、屋顶、楼梯、门、窗
第三层级	墙、柱、楼板

当 FPS＞30，显示所有族类型；当 15≤FPS≤30，加载第二层级的族类型；当 FPS＜15 严重影响渲染流畅度，仅加载第三层级的族类型。

LOD 场景控制算法虽然能渲染剔除远距离小构件三角面片，但是对于大场景构件的优化却很有限，帧率并不能

达到理想状态。因此本文结合自适应层级划分和 LOD 优化算法，提出 LOD-AD 场景控制算法，先进行 LOD 算法优化后再通过帧率监测来自适应层级划分剔除，从而达到最优场景渲染效果。具体算法原理如图 7 所示。

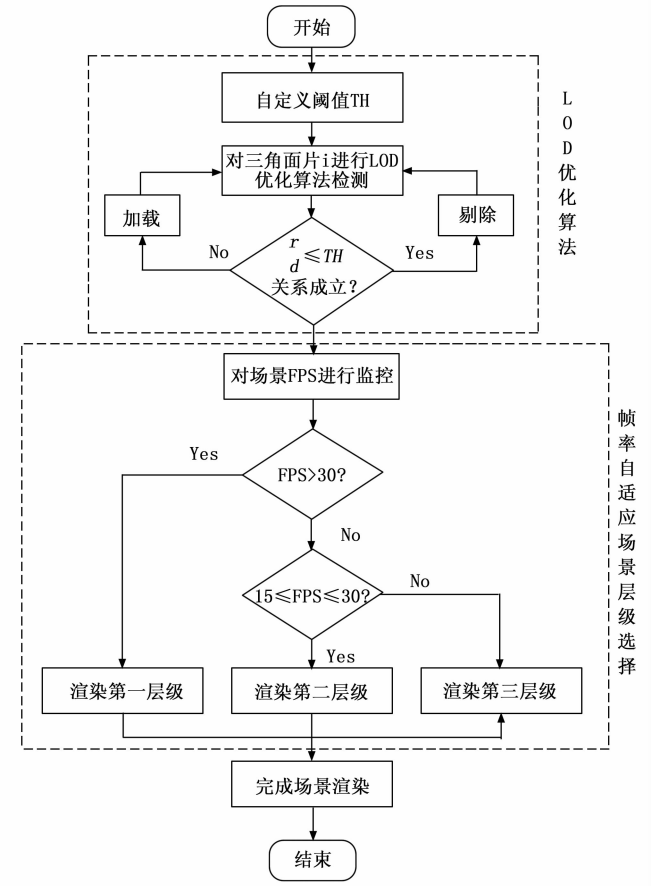


图 7　算法原理图

4　实验与结果分析

为了验证模型数据优化和场景渲染控制方法的可行性，本文选择 5 个不同体量大小的 BIM 模型组成实验数据进行验证，模型基本信息如表 2 所示。实验采用 Intel（R）Core（TM）i5-5257U@ 2.70 GHz、8 GB 内存以及 64 位 Windows10 操作系统的笔记本电脑，基于 Chromr 浏览器实现。

表 2　模型基本信息

模型序号	模型名称	模型三角面片数
A	公寓楼	508
B	科技馆	1 987
C	商务楼	6 009
D	办公楼	9 524
E	医院门诊部	26 035

4.1　模型数据优化有效性分析

针对原模型体量过大，在 Web 端重组加载时间过长的问题，提出以 glTF 为中间格式，并采用冗余顶点删除法和 Draco 算法来达到模型数据优化目的。为分析模型数据优化的有效性，通过模型数据优化前后的体量、传输时间和保真度来进行实验验证。

实验依据表 2 中的 5 个 BIM 模型为实验对象，使用了模型数据压缩算法与未使用算法的模型体量对比如表 3 所示，模型传输时间对比如图 8 所示。

表 3　模型体量对比

模型序号	原模型体量/kB	优化后模型体量/kB	模型体量压缩率/%
A	67 343	4 552	93.24
B	94 124	5 205	94.47
C	143 624	6 954	95.15
D	221 747	8 249	96.28
E	567 401	14 412	97.46

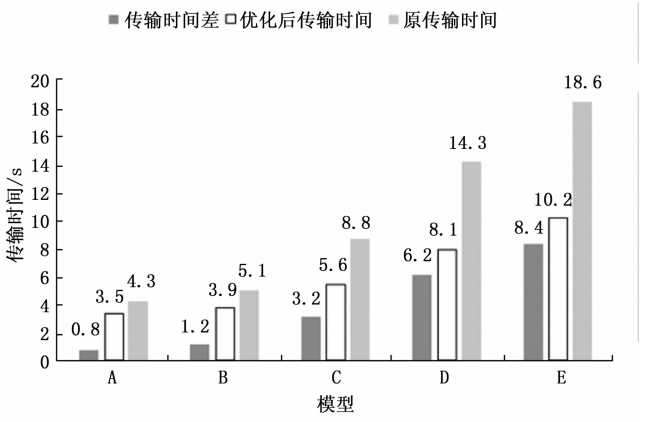


图 8　模型传输时间对比

从表 3 可以看出，随着模型的体量增大，本文的数据优化算法效果更佳，模型体量压缩率提高更明显。这是由于更大体量的模型具有更多的三角面片，而本文的压缩算法是针对三维几何网格的，而网格正是由无数个三角面片组成。尽管不同模型压缩率有所差别，但是该算法相较于原模型整体压缩率均在 93%以上。

从图 8 的模型传输时间可见，模型 A～E 在数据优化后传输时间相较于优化前明显缩短。模型在网页端利用 GPU 重组导出的模型数据会消耗一定时间，且体量越大的模型重组所需的时间更多，而利用数据优化算法后的模型在 Web 端重组速度明显加快，这能极大的降低计算机 GPU 的消耗，提升用户的操作体验。

对于模型的保真度，采用数据优化算法后的 glTF 文件，实现其在网页端的重组渲染，通过对比模型数据保真度与模型渲染保真度两方面来验证。

1）模型数据保真度。

在导出的 glTF 文件中，包含了模型的统一标识符 ID，各个构件拥有自己的标识符 ID，在模型的几何信息与属性信息中都含有标识符 ID 使得它们互相关联。因此通过点击构件后，获取鼠标所在位置来与模型几何信息进行匹配，得到该构件元素的标识符 ID，通过该 ID 调取出模型的属性信息。

对于模型数据的保真度以商务楼的屋顶为例，对比 Revit 原模型与网页端模型的属性信息，如图 9 所示。

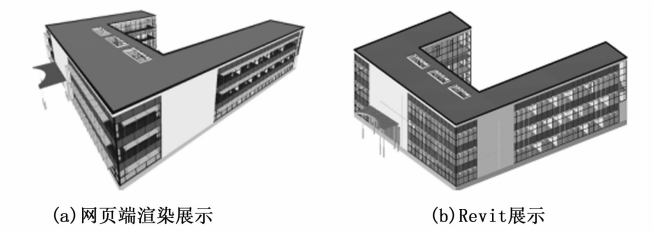
(a)Revit屋顶信息 (b)网页端屋顶信息

图 9 属性信息保真度对比

图 9（a）中的重要信息在图 9（b）中均能一一对应（框图标记），由图中的属性信息对比可见，数据优化后即使信息体量减少明显但是模型主要属性信息并未丢失，也未出现信息错误的情况，确保了模型信息的高保真度。

2）模型渲染保真度。

为了更好呈现模型渲染外观的保真度，在导出 glTF 格式的同时还导出了模型的材质贴图，配上材质贴图后该模型的网页端渲染和该模型的 Revit 原图的对比如图 10 所示。



(a)网页端渲染展示 (b)Revit展示

图 10 模型外观保真度对比图

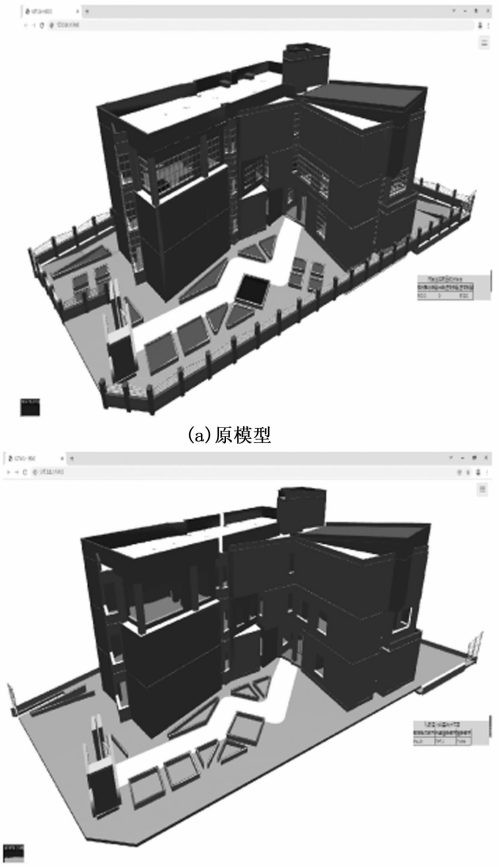
由图 10 的模型渲染保真度对比可见模型外观无明显差异，在网页端的渲染能够很好的代替 Revit 建模软件的模型展示，起到方便用户实时查看模型的效果。

通过上述模型数据优化可行性分析实验可见，优化后模型的压缩及传输过程都有明显改进，且保真度完好。因此，使用本文的模型数据优化算法更有利于模型的 Web 端传输。

4.2 场景渲染控制可行性分析

本文针对模型渲染后场景帧率过低问题提出了 LOD-AD 优化算法来对模型构件进行有效剔除，达到提高帧率的目的。对此，本文以构件类型较多的办公楼为例具体展现 LOD-AD 优化算法的显示效果，并以表 2 中的 5 个 BIM 模型作为实验对象，设置相同阈值和相同视距下这些对象的 LOD-AD 优化后与优化前的 FPS 对比，以此验证本文渲染控制的可行性。

选择族类型较多的办公楼模型为例来展现其显示效果，原模型在 Web 端显示以及 LOD-AD 场景控制算法优化后显示效果图如图 11 所示。



(a)原模型

(b)LOD-AD优化后模型

图 11 办公楼模型展示

可以看出，原模型 FPS 在 15～30 之间，模型自适应选择第二层级渲染，剔除第二层级以外的所有族类型，从图中可见图 11（a）中室内的桌椅、室外绿化等构件在图 11（b）中被剔除，但是模型的重要图元如墙、屋顶、楼板等图元显示渲染，并且在渲染过程中剔除了远距离小构件三角面片，依据此方法完成帧率监测的场景自适应以及远距离小面片构件自动剔除的控制效果，从而完成低重要度构件剔除、高重要度构件实时渲染，达到模型外观保留的效果。

图 12 列举了表 2 中五种不同体量的建筑初始模型、使用 LOD 渲染控制算法模型以及 LOD-AD 渲染控制算法模型（阈值参数 0.05，视点距离模型中心距离 3m）的帧率比较结果。

可以看出，未使用任何优化算法的初始模型，在模型面片数量较少的模型 A、模型 B 中帧率均在 30 以上，能达到流畅渲染的目的，但是当模型体量更大拥有更多的三角面片时，模型帧率极低，网页端渲染卡顿明显；当使用 LOD 场景控制算法时模型帧率相对有所改善，但是在达到模型 D、模型 E 那样较多三角面片组成的大体量模型时，网页端卡顿依旧存在；本文所提出的 LOD-AD 场景渲染控制算法相较于原模型与 LOD 优化后的模型，帧率提高明

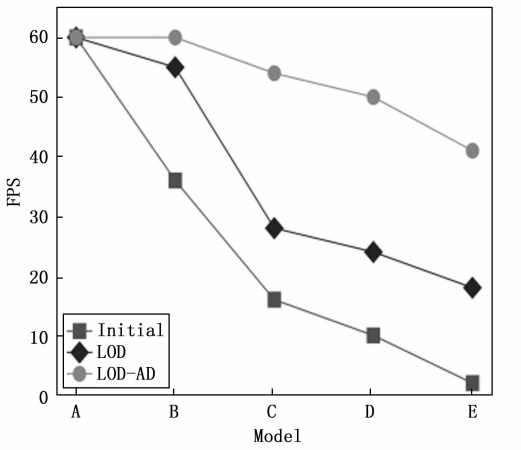


图 12　帧率对比

显，即使是对于模型 E 这样的超大模型也依旧能保持 40 以上的 FPS。

通过场景渲染控制可行性分析可知，LOD-AD 控制算法不仅能确保模型外观基本保留，而且能大幅提高帧率，对于大模型依旧能完成流畅渲染。因此，本文提出的 LOD-AD 渲染控制算法能有效达到场景渲染优化的效果。

5　结束语

本文针对 BIM 模型体量大，在 Web 端模型加载缓慢及渲染卡顿等问题，提出将 glTF 格式作为 Revit 与 Web 端的数据传输格式，进而通过冗余顶点删除法和 Draco 算法将 glTF 格式中的三维网格压缩，并使用提出的 LOD-AD 渲染控制算法将模型场景中的构件选择性剔除及渲染。实验结果表明，本文提出的 BIM 模型轻量化展示以及渲染控制方法极大的减少了模型体量以及模型传输时间，保证了模型的保真度，且极大程度的提高了模型渲染效果，解决了 Web 端重组模型后流畅度较低的问题。但是，本文对相机视野外的模型还可以进一步剔除且 Web 端页面仍可进一步完善，下一步工作可通过可视域分析将不可见模型虚化渲染或者直接渲染剔除，借助大屏技术添加建筑施工信息监测等功能，以期实现更好的轻量化显示效果。

参考文献：

[1] 李光威. WebBIM 的分布式软件架构设计与实现 [J]. 电脑知识与技术，2018，14 (31)：73－74.

[2] XIAO Y Q，HU Z Z，WANG W，et al. A mobile application framework of the BIM-based facility management system under the cross- platform structure [J]. Computer Aided Drafting，Design and Manufacturing，2016，26 (1)：58－65.

[3] 叶　娜，严昱欣，张　翔，等. 基于 BIM＋Cesium 三维可视化校园系统的设计与实现 [J]. 计算机测量与控制，2021，29 (1)：140－145.

[4] 刘北胜. 基于云渲染的三维 BIM 模型可视化技术研究 [J]. 北京交通大学学报，2017，41 (6)：107－113.

[5] 徐　照，张　路，索　华，等. 基于工业基础类的建筑物 3D Tiles 数据可视化 [J]. 浙江大学学报 (工学版)，2019，53 (6)：1047－1056.

[6] XIN YAO HUANG，SHU FEN YANG，KUN FA LEE. Research on 4D Visualized Dynamic Construction of BIM Building Decoration [J]. IOP Conference Series：Earth and Environmental Science，2020，619 (1)：12－18.

[7] LI MINGLEI，NAN LIANGLIANG. Feature- preserving 3D mesh simplification for urban buildings [J]. ISPRS Journal of Photogrammetry and Remote Sensing，2021，173：135－150.

[8] 李少卿，霍　亮，沈　涛，等. 顾及角度误差的三维建筑模型边折叠简化算法 [J]. 武汉大学学报 (信息科学版)，2021，46 (8)：1209－1215.

[9] WANG SHAOHUA，LI SHENG，LAI SHUNNAN. Real-time rendering of large-scale static scene [J]. Computer Aided Drafting，Design and Manufacturing，2017，27 (2)：1－6.

[10] ZHOU X P，ZHAO J C，WANG J，et al. OutDet：an algorithm for extracting the outer surfaces of building information models for integration with geographic information systems [J]. International Journal of Geographical Information Science，2019，33 (7)：1444－1470.

[11] 冯雨晴，奚雪峰，崔志明. 基于 WebGL 的 BIM 模型轻量化研究 [J]. 苏州科技大学学报 (自然科学版)，2021，38 (4)：72－78.

[12] VEDERNIKOVA A A，SHISHMAREV R A. Truss model data analysis in Autodesk Revit [J]. IOP Conference Series：Materials Science and Engineering，2020，944 (1)：012－037.

[13] 宋廷强，刘亚林，张　敏. 基于 STL 文件的柱状支撑结构自动生成算法 [J]. 计算机测量与控制，2018，26 (9)：277－282.

[14] NAM D，LEE D，LEE S，et al. A Comparative Study on 3D Data Performance in Mobile Web Browsers in 4G and 5G Environments [J]. International Journal of Internet，Broadcasting and Communication，2019，11 (3)：8－19.

[15] DAI CHENGQIU，FENG JUNYING. Algorithm of 3D Geometric Data Compression [C] //Journal of Simulation (VOL. 3，NO. 3)，2015：62－64.

[16] ROSSIGNAC J. Edgebreaker：Connectivity compression for triangle meshes [J]. IEEE transactions on visualization and computer graphics，1999，5 (1)：47－61.

[17] 刘　迎，刘学慧，吴恩华. 基于模版的三角网格拓扑压缩 [J]. 计算机辅助设计与图形学学报，2007 (6)：703－707.

[18] 边根庆，陈蔚韬. 面向 Web 的建筑三维模型可视化方法研究 [J]. 图学学报，2021，42 (5)：823－832.

[19] PETTIT J B，MARIONI J C. bioWeb3D：an online webGL 3D data visualisation tool [J]. BMC bioinformatics，2013，14 (1)：1－7.

[20] ABUALDENIEN J，BORRMANN A. A meta- model approach for formal specification and consistent management of multi-LOD building models [J]. Advanced Engineering Informatics，2019，40：135－153.

[21] 甘宸伊，陈向宁. 基于 World Wind 的三维显示算法研究与改进 [J]. 四川兵工学报，2010，31 (11)：88－90，98.