杨萍，赵荣．基于二次度量误差的 BIM 模型轻量化关键技术研究[J]．智能计算机与应用，2025，15(5)：188-193．DOI：10.20169/j.issn.2095-2163.250526

基于二次度量误差的 BIM 模型轻量化关键技术研究

杨 萍，赵 荣
（陕西科技大学 电子信息与人工智能学院，西安 710021）

摘 要：目前城市建筑场景愈发复杂，相应的三维 BIM 模型体量也愈发庞大，本文针对三维模型展示不便、由于体量庞大引起的实时渲染显示慢以及 WebGL 对一般的 Revit 文件无法直接支持等问题，提出了基于二次度量误差的 BIM 模型轻量化展示方法。本文首先通过 Revit 二次开发将 BIM 模型转换成. glTF 格式文件；其次，使用二次度量误差边折叠算法对模型进行轻量化，并使用 Draco 算法进行压缩，压缩完成后再通过 LOD 优化层级算法等方法对其进行 Web 端的渲染展示。实验结果表明，本文提出的方法可以在有效保留模型几何特征的前提下，实现 Web 端三维建筑模型的渲染显示，最终数据优化压缩比例最高可达 96.36%，Web 端实时渲染所需时间最高可提升 4.9 s。

关键词：二次度量误差；BIM 模型轻量化；Revit 二次开发；WebGL；渲染优化
中图分类号：TP391.9 **文献标志码**：A **文章编号**：2095-2163(2025)05-0188-06

Research on key technologies of BIM model lightweight based on quadratic error measure

YANG Ping, ZHAO Rong
(School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an 710021, China)

Abstract: At present, urban building scenes are becoming more and more complex, and the corresponding 3D BIM model volume is becoming larger and larger. This paper proposes a lightweight display method of BIM model based on quadratic error measure in view of the inconvenient display of 3D models, the slow real-time rendering display caused by the large volume, and the inability of WebGL to directly support general Revit files. Firstly, convert the BIM model into a . glTF format file through Revit secondary development. Secondly, use the quadratic metric error edge folding algorithm to lighten the model, and then use the Draco algorithm to compress it. Finally, render and show the model which is operated by above methods on the Web side by LOD optimization hierarchy algorithm. Experimental results show that the proposed method can realize the rendering and display of the Web-side 3D architectural model under the premise of effectively retaining the geometric features of the model, and the final data optimization compression ratio can reach up to 96.36%, and the time required for Web-side real-time rendering can be increased by up to 4.9 s.
Key words: quadratic error measure; BIM model lightweight; Revit secondary development; WebGL; rendering optimization

0 引 言

建筑信息模型(Building Information Modeling, BIM)是应用于工程设计建造管理的数据化工具，具有参数化、可视化、协调性、可优化性等特点[1]。BIM 技术的基本原理是通过创建三维数字化模型，并将创建的信息模型应用到工程项目设计、施工以及运营维护阶段，BIM 模型中包含了建筑中整个生命周期的包含设计、施工、运维在内的各方面数据信息，具有极高的应用价值。也因如此，为满足目前日新月异的需求，建筑体量也越来越大，这就意味着数据信息的庞大，同时 BIM 专业软件也占用着较大的计算机内存，而随着 Web 端以及 3D 渲染引擎的迅速发展，对 BIM 模型的 Web 端渲染提供了支持。

基金项目：深圳市科技计划项目(JSGG20210802154545031)；陕西省重点研发计划项目(2023-YBGY-208)。
作者简介：赵 荣(1999—)，女，硕士研究生，主要研究方向：模型轻量化，信息可视化。
通信作者：杨 萍(1979—)，女，副教授，硕士生导师，主要研究方向：人工智能与机器学习，物联网体系研究，嵌入式系统开发。Email：yangping@sust.edu.cn。
收稿日期：2023-10-27

文献[2]针对三维建筑模型 Revit 模型进行了重建，首先通过 Revit API 将 BI 模型转换成 JSON 格式文件，再通过 WebGL 解析 JSO 文件中的几何、属性等信息，最终在 Web 端实现了三维模型重建。文献[3]提出了 BIM 模型轻量化策略，并成功将其在 Web 端进行了展示，然而在数据优化和渲染过程中其效果并不理想。文献[4]通过剔除视野范围内不可见构件方法，降低了计算机 GPU 计算压力，但进行剔除视野的计算过程十分复杂，也导致了最终的渲染时间难以得到保证[5]。文献[6]以 BIM 模型的 IFC 标准文件作为输入数据源对 BIM 模型进行重构，再将重构后的 IFC 模型文件向 glTF 格式文件进行转换，该方法能够显著减少 BIM 模型数据冗余，但轻量化效果并不理想。文献[7]提出了利用八叉树优化算法进行场景优化渲染，但构造八叉树的过程会消耗极大的存储空间。而在三维模型轻量化的方法中，利用三角网格的简化实现是一种比较有效的方法，其中由 Hoppe 等学者[8]提出的边折叠算法是较为常用的方法，文献[9]也提出了使用边折叠算法简化模型，对三角网格进行了简化，降低了面片复杂度，实现了模型的轻量化，但却影响了模型外观，这是由于在简化率较高时，经典的边折叠算法会出现细节几何特征损坏或丢失问题。

因此，本文首先对 BIM 模型进行了二次开发，对庞大的 BIM 模型进行了初步的轻量化，将其转换为在 Web 端常用的. glTF 格式；其次，提出使用以二次度量误差为代价的边折叠算法进行模型网格数据轻量化处理，对处理后的模型进行 Draco 压缩，以确保 Web 端网络传输通畅；最后，在 Web 端进行模型渲染时，使用 LOD 优化算法进行渲染控制。实验结果表明，本文提出的轻量化方法可以很好地压缩模型体量，在 Web 端显示时也可实现在不同帧率下自适应选择先渲染优先级较高的、再依次渲染优先级较低的图元信息，大大降低了 GPU 压力，从而也有效保证了整体模型渲染效果。

1　Revit 二次开发

在进行 BIM 模型轻量化时，首先通过 Revit 二次开发技术在尽可能减少 BIM 模型文件的大小和数据量的同时，保留关键信息[10]，从而在 Revit 模型导出时实现轻量化，以支持 BIM 模型的有效使用。

本文使用 Visual Studio 和 Revit 软件进行 Revit 二次开发[11]，使用 C#作为二次开发语言。考虑到开发的插件需要兼容 Revit 2020，Visual Studio 中的代码编程模型需要使用. Net Framework4. 8 及以上的版本，在开发工作前需要先引用 RevitAPI. dll 和 RevitAPIUI. dll 两个接口文件。在 Revit 二次开发时，需要使用 Revit 自带的一些工具。Revit SDK[12]是 Revit 提供的开发套件，其中包含 Revit 项目样例文件、应用程序编程接口等。SDK 部分命令及其说明详见表 1。Add In Manager 接口的功能是允许 Revit 进行外部代码开发，开发者可以通过这个插件在 Revit 中测试运行二次开发设计的代码，具体的开发前准备如图 1 所示。

表 1　SDK 部分命令及说明

Table 1　Commands and descriptions of the SDK

命令	说明
AddSpaceAndZone	创造空间和区域
APIAppStartup	启动外部应用
AreaReinParameters	显示和修改钢筋参数
AutoTagRooms	自动创建房间标签
AvoidObstruaion	障碍物检测和解决
BotmdaryCondition	边界条件
CompoundStructure	复合结构
CreateBeamStystem	生成梁系统
CreateSection′views	创建剖面图
DeckProperties	显示楼板属性

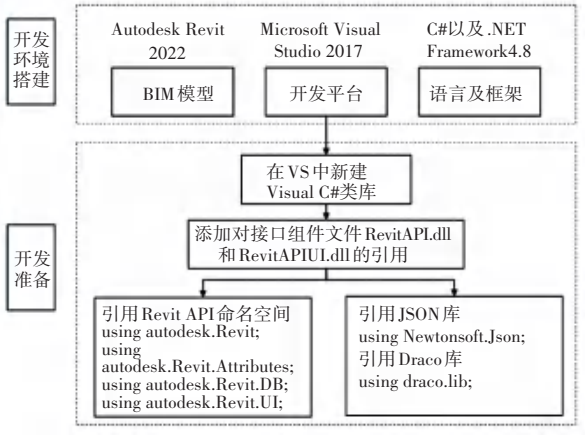

图 1　Revit 二次开发准备

Fig. 1　Preparation for Revit secondary development

本文围绕模型内部数据的轻量化目标，通过自定义 IExportContext 接口导出模型的几何网格、材质、纹理和属性信息。调用该接口中函数实现 BIM 模型数据信息提取以及由. rvt 文件向. glTF 格式文件的转化，其数据导出的过程如图 2 所示。

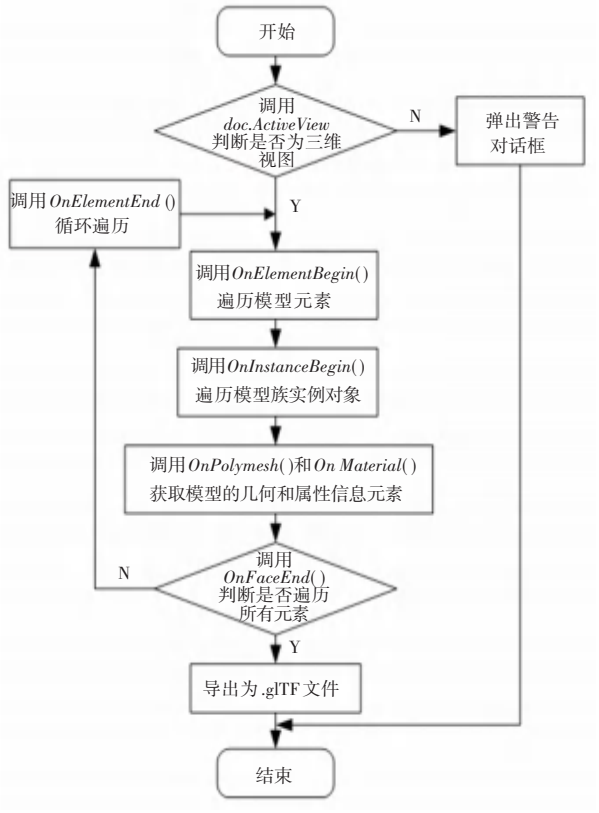

图 2 Revit 模型数据导出流程

Fig. 2 Process of Revit model data exporting

具体流程如下：

(1)首先定义存储模型对象的全局变量和堆栈。然后,通过调用 *doc. ActiveView* 来判断是否为三维视图,若不是则弹出警告框。

(2)在 *OnViewBegin*( ) 中定义操作以最初缩小模型的表面。*OnElementBegin*( ) 和 *OnInstanceBegin*( ) 分别表示元素和族实例导出的开始。Revit 模型由多个网格或不同实例组成,因此在解析时嵌套实例子对象。

(3)在获取几何数据时,通过访问模型的几何属性 Geometry 来获取 Geometry-Element 的实例,其中包含了实体、线条等几何对象。通过导出类中的 *OnMaterial*( ) 方法来获取三维模型的材质信息,并使用 *OnPolymesh*( ) 方法来获取模型的几何信息,此后再调用完成 *OnFaceEnd*( ) 实例加载。

(4)在导出结束时写入. glTF 文件。

2 模型数据优化

2.1 边折叠及二次度量误差

边折叠算法是几何元素删除方法中比较常用的方法之一。该算法的核心思想是首先计算折叠每条边的成本并进行排序,从成本最低的边开始折叠边缘,直到所有边缘都无法折叠。边折叠操作如图 3 所示。一次边折叠操作是指将原三维网格模型中的点 $\boldsymbol{V}_1$、$\boldsymbol{V}_2$ 折叠到一个新的点位置 $\boldsymbol{V}$ 上,删除掉原来与 $\boldsymbol{V}_1$、$\boldsymbol{V}_2$ 相连接的边,并将与 $\boldsymbol{V}_1$、$\boldsymbol{V}_2$ 相连接的所有点都连接至点 $\boldsymbol{V}$,将退化掉的三角形进行删除,从而达到三角简化的效果。

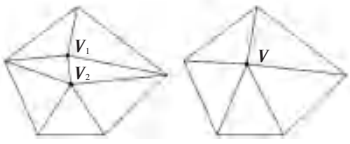

图 3 边折叠操作示意图

Fig. 3 Schematic diagram of edge collapse operation

对于三维网格模型的简化,其最终目的在于减少模型三角面片数,同时需要保证三维模型整体的几何特征基本不变,其关键问题在于如何定义一个误差度量使得压缩模型与原始模型之间的误差最小化。针对这一问题,大量学者对此进行了研究,并衍生出多种方法。其中,Garland 和 Heckbert 提出的二次误差测量(Quadratic Error Measure, QEM)算法较为突出。在进行边折叠的主要目标是寻找一组最优的点 $\boldsymbol{V}_1$、$\boldsymbol{V}_2$, 对其进行折叠形成新的顶点,QEM 算法在进行边折叠的过程中,将这里的最优定义为合并后的顶点应当到原先合并前 2 个顶点周围若干三角形所在平面的距离的平方和最小,也就是以新顶点到原相邻曲面的距离平方和作为边折叠的误差度量。折叠的过程中,通过为每个顶点建立一个二次误差矩阵,并利用该矩阵计算每个点的误差,从而确定折叠后的新顶点的最优位置和折叠所需的代价,然后经过一系列的迭代,最终实现简化的目的,该算法的流程如图 4 所示。

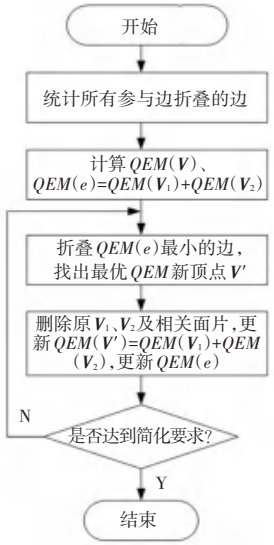

图 4 QEM 算法简化过程

Fig. 4 The simplification process of QEM algorithm

具体过程如下：首先将模型中所有顶点初始矩阵定义为 $\boldsymbol{Q}$，顶点 $\boldsymbol{V}_i=[x \quad y \quad z \quad 1]^{\mathrm{T}}$ 的顶点误差定义为 $\triangle \boldsymbol{V}_i$，$\triangle \boldsymbol{V}_i$ 表示为模型中顶点 $\boldsymbol{V}_i$ 到其所有的一阶领域三角面片距离的平方和，其表达式为：

$$\begin{aligned}\triangle \boldsymbol{V}_i &= \triangle([x \quad y \quad z \quad 1]^{\mathrm{T}}) = \sum_{p\in P(V_i)}(\boldsymbol{p}^{\mathrm{T}}\boldsymbol{V}_i)^2 = \\ &\sum_{p\in P(V_i)}(\boldsymbol{V}_i^{\mathrm{T}}\boldsymbol{p})\cdot(\boldsymbol{p}^{\mathrm{T}}\boldsymbol{V}_i) = \\ &\sum_{p\in P(V_i)}\boldsymbol{V}_i^{\mathrm{T}}(\boldsymbol{p}\boldsymbol{p}^{\mathrm{T}})\boldsymbol{V}_i = \\ &\boldsymbol{V}_i^{\mathrm{T}}\Big(\sum_{p\in P(V_i)}K_p\Big)\boldsymbol{V}_i \end{aligned} \tag{1}$$

其中，$\boldsymbol{p}=[a \quad b \quad c \quad d]^{\mathrm{T}}$ 表示平面方程 $ax+by+cz+d=0$ 系数 $a^2+b^2+c^2=0$。

$\boldsymbol{Q}=\sum_{p\in P(V_i)}\boldsymbol{K}_p$ 为 4 × 4 的基本误差对称矩阵，其中 $\boldsymbol{K}_p=\boldsymbol{p}\boldsymbol{p}^{\mathrm{T}}=\begin{bmatrix}a^2 & ab & ac & ad\\ ab & b^2 & bc & bd\\ ac & bc & c^2 & cd\\ ad & bd & cd & d^2\end{bmatrix}$。

定义进行折叠后形成的新顶点为 $\boldsymbol{V}_N$，原网格模型折叠边为 $\boldsymbol{V}_1\boldsymbol{V}_2$，则新顶点的误差矩阵为：

$$\boldsymbol{Q}_N=\boldsymbol{Q}_1+\boldsymbol{Q}_2 \tag{2}$$

其折叠代价为：

$$\triangle V_N=\boldsymbol{V}_N^{\mathrm{T}}\boldsymbol{Q}_N\boldsymbol{V}_N \tag{3}$$

进一步，推得的表达式为：

$$\begin{aligned}\triangle V_N &= q_{11}x^2+2q_{12}xy+2q_{12}xz+2q_{14}+\\ &\quad q_{22}y^2+2q_{22}z^2+2q_{24}z+q_{44}\end{aligned} \tag{4}$$

折叠代价的大小取决于折叠后生成顶点 $\boldsymbol{V}_N$ 的位置，$\boldsymbol{V}_N$ 与折叠边 $\boldsymbol{V}_1\boldsymbol{V}_2$ 所关联的三角面片的距离越大，其折叠代价也就越大。对式(4) 中 x、y、z 分别求偏导，如令求偏导后的算式为 0，求 $\boldsymbol{V}_N$ 最佳位置以使 $\triangle V_N$ 达到局部最小值解。推得的数学公式为：

$$\frac{\partial\triangle(m)}{\partial x}=\frac{\partial\triangle(m)}{\partial y}=\frac{\partial\triangle(m)}{\partial z}=0 \tag{5}$$

$$\boldsymbol{V}_N=\begin{bmatrix}q_{11} & q_{12} & q_{13} & q_{14}\\ q_{12} & q_{22} & q_{23} & q_{24}\\ q_{13} & q_{23} & q_{33} & q_{34}\\ 0 & 0 & 0 & 1\end{bmatrix}^{-1}\times\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \tag{6}$$

若矩阵可逆，则可通过式(6) 计算新顶点 $\boldsymbol{V}_N$ 位置；若矩阵不可逆，则在折叠边端点 $\boldsymbol{V}_1$、$\boldsymbol{V}_2$ 以及折叠边中点中选择使收缩代价 $\triangle V_N$ 达到最小值的点作为新顶点。

2.2　Draco 压缩算法

为应对网络传输和存储大型专业三维模型的挑战，Google 开发了 Draco 开源库[13] 用于进行模型压缩。Draco 专注于高效解压缩网格数据，能够对模型的点、连接、颜色和与几何属性相关的通用数据进行压缩。通过使用 Draco 进行压缩，可以显著减小模型的大小，同时在 Web 端进行网络传输和三维模型存储的性能也得到了显著提升。

Draco 网格模型解压缩是通过网格压缩算法 Edgebreaker[14] 和熵编码技术来实现的。Edgebreaker 压缩算法是基于一种坍缩操作的三角网格压缩算法。在该算法中，每个顶点与 6 个三角形网格相关联。通过一系列步骤，可以从当前网格中移除 1 个三角形网格。Edgebreaker 算法将三角形网格的坍缩情况分为 5 类，每种情况分别对应了不同的边折叠压缩代码，以适应性地完成不同的边折叠操作，网格压缩分类示意如图 5 所示。

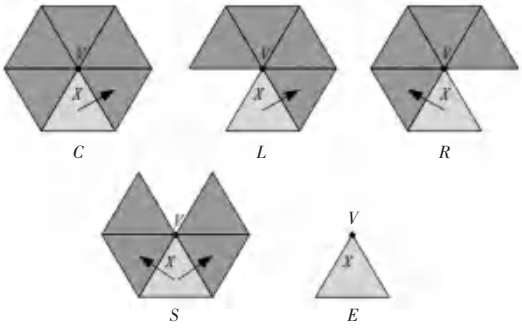

图 5　Draco 网格压缩分类示意图

Fig. 5　Draco grid compression classification diagram

图 5 中，X 即为当前遍历的三角形，V 表示被遍历三角形的第三顶点。网格压缩过程如图 6 所示。图 6 中，箭头表示对该不规则网格进行压缩的路径示例。

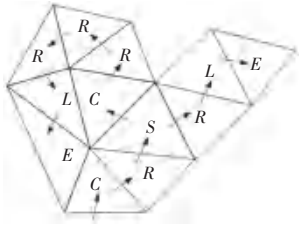

图 6　Edgebreaker 网格分割路径示例

Fig. 6　Example of Edgebreaker mesh segmentation path

2.3　Web 端模型渲染优化

在 Web 端对三维模型进行渲染时，通常使用帧率 (Frames Per Second，*FPS*) 作为其性能指标，帧率越高，表示模型渲染越流畅。但相应地，帧率越高、就代表着计算机面临的 GPU 运算压力越大。

层次细节(Level Of Detail，LOD)[15] 是一种优化技术，用于在不同距离和视角下呈现场景中的物体。

当距离远时,在不影响视觉效果的前提下,场景中细节物体得到相应的简化,以此减少使用计算机渲染资源;当距离近时,场景细节渲染更为精细,以保证模型的精确性。因此,本文通过 LOD 算法进行场景渲染效果的优化。

3 实验验证

3.1 Revit 二次开发实现

首先按照第 1 节图 2 中的 Revit 模型数据导出流程,将开发并编译好的插件保存为 RevitToGltfJson. dll 文件,然后在 Revit 软件中选择“附加模块”中“Add-In Manager”选项来加载该文件并启动程序,如图 7 所示。

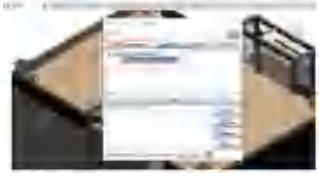

图 7 在 Revit 中选择开发的插件

Fig. 7 Select the developed plug-ins in Revit

选择要提取的图元 ID 的名为“ids”的 JSON 文件来进行数据提取,该 JSON 文件是在实验中所使用的 BIM 模型经过 OutDet 算法提取得到的外轮廓构件的 ID 集合。待数据提取完成后,会自动弹出“提取成功”的提示框如图 8 所示。

图 8 弹出的导出成功窗口

Fig. 8 The exported success dialog box

Revit 模型导出过程中,将模型几何信息、纹理信息存储于. glTF 文件,属性信息存储于 JSON 文件。最终提取成功后的 3 个文件如图 9 所示,包括. bin、. gltf 和. json 文件。可以看出,通过开发的插件轻量化数据提取后,模型文件明显减小,进一步也可以提高数据传输效率。

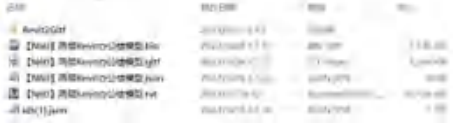

图 9 导出成功后生成的文件

Fig. 9 The generated file after a successful export

为评估本文二次开发方法的效果,对不同复杂程度的 Revit 模型经过轻量化处理后导出的模型构件数和模型文件大小进行比较,得到的结果见表 2。二次开发前后 BIM 模型对比如图 10 所示。

表 2 Revit 二次开发前后体量对比

Table 2 Comparison of volume before and after Revit secondary development

实验模型	原文件大小/KB	处理后文件大小/KB	处理后优化比例/%
建筑模型 1	36 124	7 366	79.60
建筑模型 2	26 124	6 724	74.26
建筑模型 3	36 650	8 034	78.07
建筑模型 4	14 562	4 185	71.26

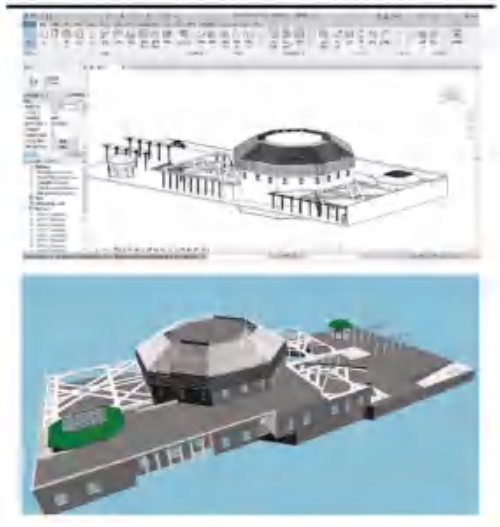

图 10 模型前后对比图

Fig. 10 Comparison before and after the model design

3.2 模型数据轻量化验证

将经过二次开发后的. glTF 格式三维模型进行轻量化压缩,实验环境见表 3。

表 3 实验开发环境

Table 3 Experimental development environment

名称	开发环境
操作系统	Windows10
编程平台	Visual Studio、Visual Studio Code
开发语言	C++、JavaScript、H5
浏览器	Chrome

首先将模型通过 QEM 边折叠算法进行简化并利用 Draco 算法来进行压缩,在 Microsoft Visual Studio 2017 软件实现代码编写。处理前后渲染时间对比、轻量化压缩后模型体量以及 BIM 模型最终轻量化压缩比例见表 4。

表 4　轻量化压缩后模型体量以及渲染时间对比

Table 4　Comparison of rendering time and model volume after lightweight compression

实验模型	处理前后渲染时间/s	处理后模型大小/K	轻量化后模型压缩比例/%
建筑模型 1	8.3/3.4	1 506	95.83
建筑模型 2	6.1/2.5	950	96.36
建筑模型 3	6.5/2.2	1 375	96.24
建筑模型 4	3.3/1.3	1 062	92.70

在 Web 端通过 WebGL 显示三维模型，过程中使用 LOD 为模型渲染显示进行优化。首先，根据三维模型中不同部位的重要程度进行划分，为不同部位设置一个 Lod 属性，不同 Lod 属性值优先级各不相同。在交互的过程中，以三维模型到相机视点的距离来划分渲染层次，当距离远时只渲染 Lod 属性优先级较高的图元信息，距离近时依次渲染更多图元。在 Web 端经过 LOD 优化渲染后不同远近程度的建筑模型示意如图 11 所示。由图 11 可以看出，在视角拉近时，模型明显更加清晰，而在远处时，也能相应地降低计算机模型渲染计算量。

图 11　不同距离 LOD 优化示意图

Fig. 11　Schematic diagram of LOD optimization at different distances

3.3　实验小结

根据实验结果可以看出，经过本文的轻量化方法，在经过 Revit 二次开发和 QEM 算法轻量化压缩以及 Web 端的渲染优化后，模型轻量化比例最高可达 96.36%，Web 端渲染模型时间最高提升了 4.9 s，并且能够在确保模型几何特征的前提下，达到减小计算机 GPU 渲染压力的目的。

4　结束语

本文针对 BIM 模型体量大、在 Web 端模型加载困难的问题，制订了 BIM 一系列轻量化方法。首先，通过 Revit 二次开发的实现，完成了由. rvt 模型文件到 Web 端常见的. glTF 格式文件的转换，转化后体量变化比例最高可达 79.60%；同时，针对模型数据量大的问题，对转换后的三维模型几何数据轻量化和模型压缩进行了研究，在不影响使用效果的情况下，模型的数据量大大减少，压缩比例最大达到了 96.36%。最后，使用 LOD 优化算法来优化渲染，成功减少了 GPU 的负载并进一步提高了加载速率和渲染效率。

参考文献

[1] 潘婷，汪霄. 国内外 BIM 标准研究综述[J]. 工程管理学报，2017，31(1)：1-5.

[2] 陈志杨，罗飞. 基于 WebGL 的 Revit 三维建筑模型重建[J]. 浙江工业大学学报，2016，44(6)：608-613.

[3] HUANG Xinyao，YANG Shufen，LEE K F. Research on 4D visualized dynamic construction of BIM building decoration[J]. IOP Conference Series：Earth and Environmental Science，2020，619 (1)：12-18.

[4] WANG Shouhua，LI Sheng，LAI Shunnan. Realtime rendering of large-scale static scene [J]. Computer Aided Drafting，Design and Manufacturing，2017，27 (2) ：1 -6.

[5] ZHOU Xiaoping，ZHAO Jichao，WANG Jia，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.

[6] 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.

[7] 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.

[8] HOPPE H，DEROSE T，DUCHAMP T，et al. Mesh optimization [C]// Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH ′93). New York：ACM ，1993：19-25.

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

[10] 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

[11] SELIM F，ELKHOLY S M，BENDARY A F. A new trend for indoor lighting design based on a hybrid methodology[J]. Journal of Daylighting，2020，7(2)：137-153.

[12] GAO C，LI S，WANG Z. Research on data fusion method of Revit and non-BIM software in complex shaped buildings[J]. Journal of Physics：Conference Series，2022，2202(1)：012027.

[13] DAI Chengqiu，FENG Junying. Algorithm of 3D geometric data compression[J]. Journal of Simulation，2015，3(3)：62 -64.

[14] ROSSIGNAC J. Edgebreaker：Connectivity compression for triangle meshes [J]. IEEE Transactions on Visualization and Computer Graphics，1999，5(1)：47-61.

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