Computer Science and Application 计算机科学与应用, 2023, 13(4), 902-914
Published Online April 2023 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2023.134088

Hans汉斯

城市三维模型通用格式转换与可视化研究

饶加旺，周　松，邢策梅，尹跃强，于　琳，袁　星，陶金梅

江苏省测绘工程院江苏时空大数据建设中心，江苏　南京

收稿日期：2023年3月22日；录用日期：2023年4月21日；发布日期：2023年4月28日

摘　要

目前，城市三维模型有多种数据格式，格式间相互转换除了带来较高成本以外，还会导致模型部件信息与属性信息发生变化，以及模型精度损失等不足。3D tiles格式以其WebGL渲染机制为基础，能够实现海量三维数据的快速加载，因此被作为城市三维模型数据转换的目标格式和通用格式，但在实践过程中，其三维格式数据无法直接被获取，给模型的通用性带来一定阻碍。为此，本文以某县级市商品住宅小区为例，基于开源语言与工具，探索一套城市三维模型(IFC格式)向通用格式3D tiles转换解决方案与可视化优化研究，完成了坐标转换、空间索引构建、视域剔除，以解决城市三维模型在大规模应用中存在的使用成本较高、平台与数据不兼容等问题。

关键词

三维模型，可视化，智慧城市，格式转换，WebGL，BIMServer

Research on Universal Format Transformation and Visualization of Urban 3D Model

Jiawang Rao, Song Zhou, Ceimei Xing, Yueqiang Yin, Lin Yu, Xing Yuan, Jinmei Tao

Jiangsu Spatiotemporal Big Data Construction Center, Jiangsu Province Surveying & Mapping Engineering Institute, Nanjing Jiangsu

Received: Mar. 22nd, 2023; accepted: Apr. 21st, 2023; published: Apr. 28th, 2023

Abstract

At present, many data formats were presented in three-dimensional urban models. In addition to high cost, conversion between formats will also result in changes in the information of model parts and attributes, and loss of model accuracy too. The 3D tiles format was based on its WebGL render-

文章引用: 饶加旺, 周松, 邢策梅, 尹跃强, 于琳, 袁星, 陶金梅. 城市三维模型通用格式转换与可视化研究[J]. 计算机科学与应用, 2023, 13(4): 902-914. DOI: 10.12677/csa.2023.134088

ing mechanism and can realize fast loading of massive three-dimensional data. Therefore, it was the target format and common format for data conversion of the city three-dimensional model. However, its three-dimensional format data cannot be obtained directly in practice, which hinders the universality of the model. For this reason, based on the open source language and tools, this paper takes a commercial residential district in a county-level city as an example, and explored a set of technical process and solution for lossless transformation from the three-dimensional urban model (IFC format) to a universal format 3D tiles, coordinate conversion, spatial index and viewing frustum culling were completed during the conversion process, in order to solve the problems of high cost, incompatible platform and data in large-scale application of the three-dimensional urban model.

Keywords

3D Model, Visualization, Smart City, Format Conversion, WebGL, BIMServer

1. 引言

随着科技的快速发展，城市三维模型逐步向语义化和拓扑化发展[1]，不仅日益成为智慧城市建设的一项重要工作，也是构建自然资源三维立体时空数据库的重要组成部分，在三维不动产、二三维一体化、实景三维等应用领域发挥重要作用[2]。城市三维模型数据包含两类信息：几何信息(存储了建筑物的组成部分)、语义属性信息。IFC (Industry Foundation Classes)标准能够较好地支持模型中建筑物对象层中实体语义描述、空间定位、几何形态、属性特征、要素相互关系及演化的表达[3]。Cesium 所使用的 3D tiles，以 WebGL 渲染机制为基础，能够实现海量三维数据的快速加载，因此被作为城市三维模型数据转换的目标格式，也是 OGC (Open Geospatial Consortium)组织的标准化格式与通用格式之一[4]。

在三维模型建模过程中，3D tiles 格式无法直接被获取[5]，因此国内外多家单位与研究人员在三维模型通用格式转换及优化方面进行了探索与实践，主要有以下三个特点：1) 以 Bentley 公司研发的 Cesium Sandcastle 功能包为例，在实现三维格式的转换后，将 3D tiles 格式文件以 URL 地址封装，转换过程完全“黑盒”，价格较为昂贵[6]；2) AutoDesk 公司研发出商用的格式工厂软件 FormItConverter，在转换过程中存在三维模型的渲染效果降低和改变原模型的几何纹理等问题[7]；3) Chen 等人开发了一套开源转换方案[1]，在转换过程中需要预先获知模型的结构信息[8]。OBJ、IFC、glTF 等格式的模型数据在几何表达、属性信息、展示范围、和坐标系统等方面均与 3D tiles 格式存在明显差异[1]，使得当前城市三维模型在大规模应用中具有较大局限。为解决城市三维模型在大规模应用中存在的使用成本较高、平台与数据不兼容、模型渲染加载慢等问题，本研究以国内某县级城市商品住宅小区为例，基于开源语言与工具，探索一套城市三维模型格式向 3D tiles 转换技术流程与可视化优化方法。

2. 研究方法

以某县级市的商品住宅小区为例，该小区包含 17 幢楼，如图 1 所示，其中 1~8 幢为花园洋房、12~13 幢为高层建筑(层高 33 层)、14~15 幢为小高层、16 幢为售楼处、9 幢和 17 幢为商铺，本文以该小区 5 幢、13 幢、15 幢三栋为例，建模软件为 Revit，模型格式为 IFC 2 × 3 TC，将 IFC 格式的模型按照实际建筑物构件进行拆分，分解成包含模型部件属性的 JSON 格式文件和包含部件几何形状信息的 IFC 文件，转化

为 OBJ 格式，再转换为 glTF 格式，由于 3D tiles 数据内嵌了二进制的 glTF (glb)，并完成坐标转换和空间索引构建，最终将 glTF 文件和包含模型部件属性信息的 JSON 文件封装得到瓦片格式的 b3dm 文件，获得 3D tiles 格式的成果。具体转换流程如图 2 所示。

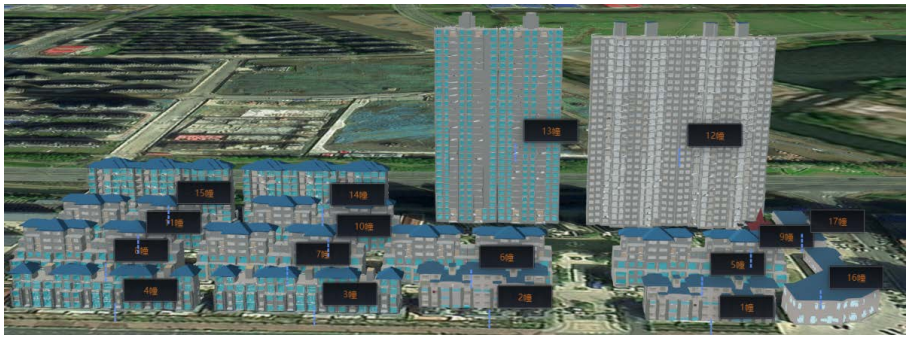


Figure 1. Three dimensional overview of the community
图 1. 小区三维概况

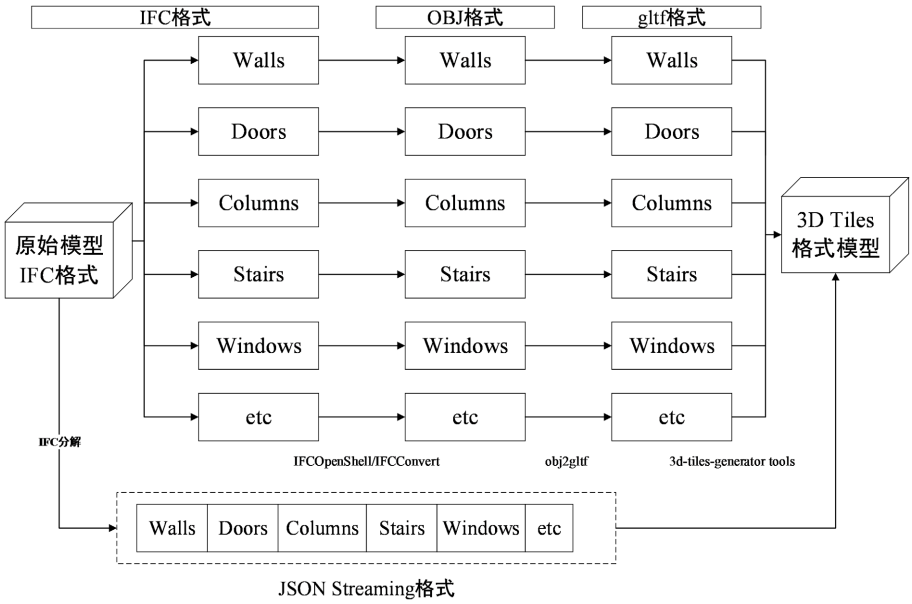


Figure 2. The conversion flowchart of IFC to 3D tiles format
图 2. IFC 向 3D tiles 格式转换流程图

2.1. IFC 文件分解

将较大的 IFC 模型直接转换成 OBJ 格式会导致属性信息的缺失[1]，因此需要将 IFC 模型按照部件进行分解。实践中首先要解决 IFC 部件查询的问题，其次是分解过程中信息无丢失，第三是保留各个部件属性的完整性与关联性。已有转换工具可直接将较大的 IFC 文件按部件转换成单个、较小的 OBJ 格式的文件，如 Open 3D Model Viewer 软件和 BIMServer，前者转换过程中存在部件信息缺失、无法关联等问题；后者作为一个开源开发项目，用于存储与管理 BIM 模型，具有模型检查、版本控制、合并与过滤 BIM 模型并动态输出 IFC 文件等功能[9]，通常将 IFC 文件上传至 BIMServer，其次按照建筑物部件进行分解，对 IFC 文件内所有对象进行遍历，将包含几何信息的对象按照模型分解成较小的 IFC 文件导出，同时以 JSON 文件保留对象的属性信息，是较为常用的方法之一，但该方法需要对 IFC 对象模型模式以

及结构等有先验知识，在查询过程中容易存在部件丢失的情况。综上，本文结合图论方法[10]对 IFC 模型实施查询与分解，将获取的 IFC 部件信息与 BIMServer 内的部件信息进行匹配，从而获得与其对应的 JSON 文件。

首先将 IFC 模型按照各部件的关系转换成有向图 G(V, E)，如图 3 所示，将墙体(IFCWall)关联的所有材质、附属设备关联图 3(a)映射成了有向图，如图 3(b)所示；其次利用最短路径算法确定各对象的关联关系，从而实现对整个 IFC 模型的遍历，实现对 IFC 的分解。

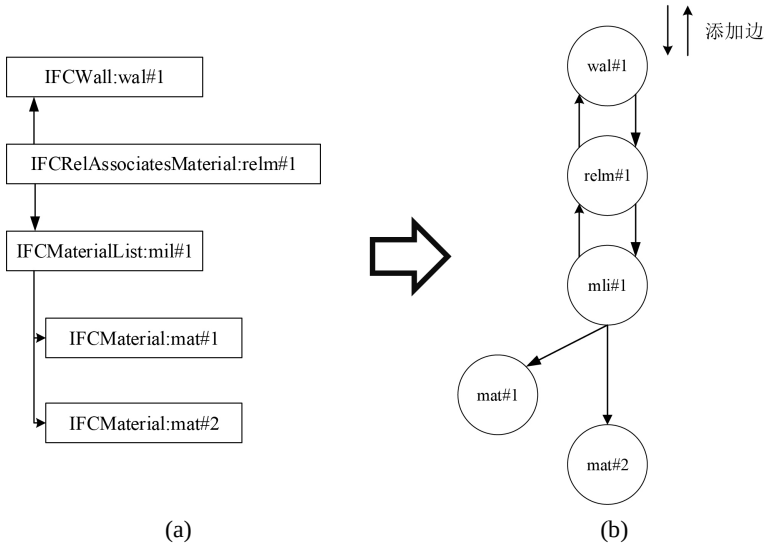


Figure 3. Transformation of IFC model into directed graph: (a) IFC model graph; (b) Directed graph
图 3. 将 IFC 模型转换成有向图：(a) IFC 模型图；(b) 有向图

以 5 幢首层为例，IFC 模型格式为 IFC 2 × 3，从 IFCRelationship、IFCRelAssigns、IFCRelDecomposes、IFCRelAssociates、IFCRelDefines、IFCRelConnects 类型中获取的 56 个目标对象，例如需要获取标记为 wal#1 的 IFCWall 的材质信息，计算 wal#1 与 IFCMaterial 的最短距离(wal#1, relc#1, wal#2, relm#2, mat#3)值为 5，但其语义上正确路径是(wal#1, relm#1, mlsu#1, matls#1, matl#1, mat#2)值为 6，两者的最短距离不一致，如图 4 所示。

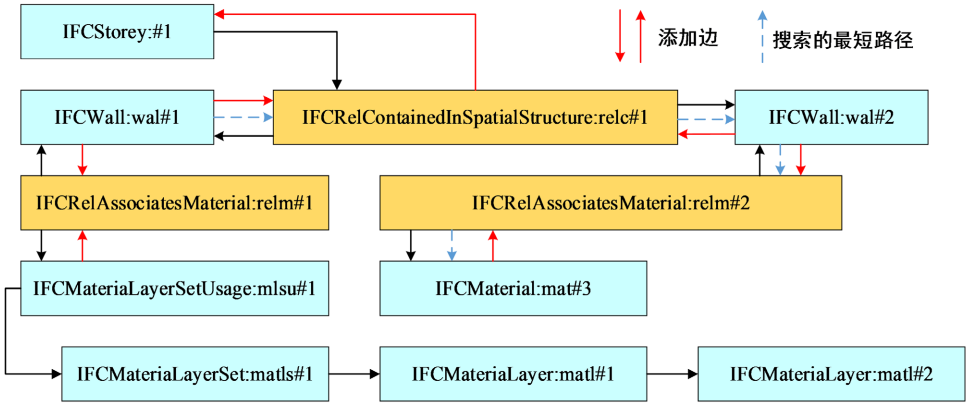


Figure 4. Paths of directed graph of 5#'s first floor
图 4. 5 幢首层的有向路径图

为避免错误的语义结果输出，引入了关系主图 G_M、关系子图 G_R 和模型查询图 G_S 三个部分，其中 V 为顶点集合、E 为结点集合，各目标对象对应的主图、关系子图和模型查询图如图 5 所示。

关系主图 G_M 作为 IFC 模型查询的基础，由模型所有的顶点 V_M 及其关系组成，无需获知 IFC 模型的格式、也无需预先定义。模型查询图 G_S 用于实际查询，是主图与所有关系子图的并集，定义如式(1)所示：

$$G_S\left(V_S,E_S\right)=G_M\cup G_S\mid G_S\subseteq G_R \tag{1}$$

关系子图 G_R 由所有的顶点 $V_{sub,i}$ 及其继承顶点 $V_{sub,j}$ 和两者的关系组成。

每个关系子图 G_{Sub} 由代表 IFC 模型对象化的顶点及对应各个关系属性和边组成，关系子图的定义为 $G_R\left\{G_{Sub,1}\left(V_{Sub,1},E_{Sub,1}\right),\cdots,G_{Sub,n}\left(V_{Sub,n},E_{Sub,n}\right)\right\}$。图 5 中的关系子图 $G_{Sub(IFCRelAssMat)}$ 定义如式(2)。

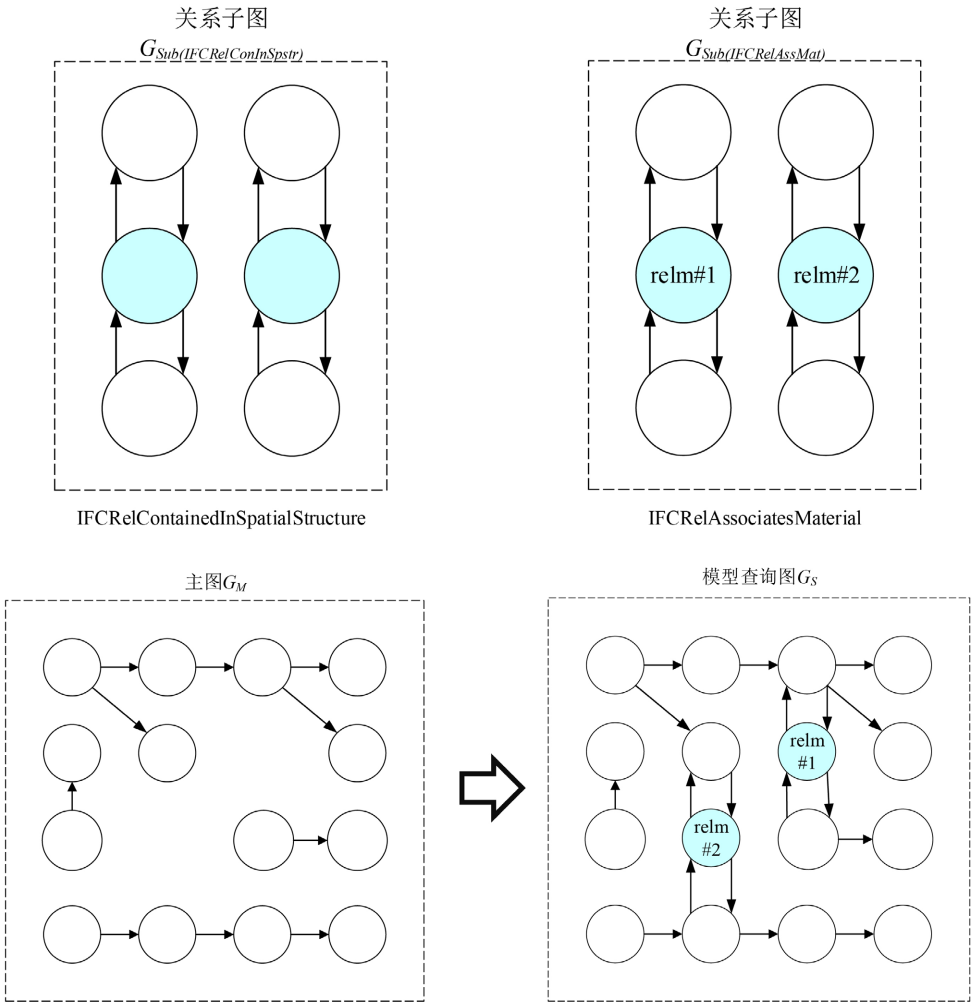


Figure 5. Model search graph using relation subgraphs and main graphs
图 5. 使用关系子图与主图生成模型搜索图

$$G_{Sub(IFCRelAssMar)}=\left(V_S,E_S\right) \tag{2}$$

其中：

$$V_S=\left(\text{relm}\#1,\text{wal}\#1,\text{mlsu}\#1,\text{relm}\#2,\text{wal}\#2,\text{mat}\#3\right) \tag{3}$$

$$E_S = \big((\text{relm}\#1, \text{wal}\#1), (\text{wal}\#1, \text{relm}\#1), (\text{relm}\#1, \text{mlsu}\#1), (\text{mlsu}\#1, \text{relm}\#1), (\text{relm}\#2, \text{wal}\#2), (\text{wal}\#2, \text{relm}\#2), (\text{relm}\#2, \text{mat}\#3), (\text{mat}\#3, \text{relm}\#2)\big) \tag{4}$$

最短路径算法参考 Dijkstra 提出的(具体请参考文献[11])，用于 IFC 模型的查询，在 5 幢首层的结构上，IFCWall(wal#1)与 IFCMaterial(mat#3)对象的最短路径已确定，IFCMaterial 的属性已确定，例如为“石膏板”，因此可知所有的 IFCWall 对象由石膏板组成。利用模型查询图依次遍历，最终获取了 IFC 模型的各个部件的名称及对应的 ID。

2.2. IFC 文件向 OBJ 格式转换

IFC 和 OBJ 文件，因为它们都是常见的数据格式。如前所述，IFC 包含三维模型和大量有关建筑的语义信息。OBJ 文件仅包含几何图形信息和材料，因此可以作为用于转换 3D 建筑模型的介质。

2.3. IBJ 格式转换成 glTF

glTF 代表 GL 传输格式，这种格式是开源的，结构如图 6 所示，模型的属性信息包括材质、节点的级别等存放在 JSON 格式文件中，bin 文件包含了模型的元数据信息、坐标等；glsl 包含了着色信息。

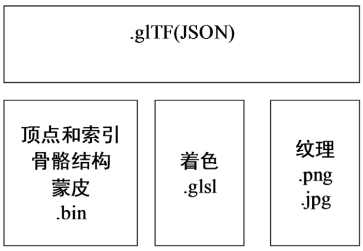


Figure 6. Format structure diagram of glTF
图 6. glTF 格式结构图

2.4. glTF 转换成 3D Tiles 格式

3D tiles 数据中内嵌了二进制的 glTF，3D tiles 生成器首先为提供的每个 glTF 文件创建一个实例，当生成器开始创建实例时，它会解析每个 glTF 文件并找到其对应的 JSON 文件，由于该实例的类名与不带扩展名的文件名相同，并且其属性是从 JSON 文件中提取的，因此每个组件的文件中的第一个 JSON 对象包含诸如“对象名称”、“IFC 类型”、“GUID”和“BATID”之类的信息，通过在每个文件中查询名称为“Dimensions”和“Constraints”的 IFCPropertySets 来提取剩余的属性；然后将这些实例包含在批处理表层次结构中的 b3dm 头文件中，最终由 glTF 格式转换后的 b3dm 数据与瓦片数据集文件(Tileset.json)文件共同组成 3D tiles 数据。

2.5. 坐标转换

由于数据范围不同，使得 IFC 模型与 3D tiles 展示数据的坐标差异问题，因此需要将 IFC 的几何信息向三维数字地球对应的世界坐标系进行转换。通常假设 IFC 模型中坐标原点在 3D tiles 格式的世界坐标系的坐标为(x_1, y_1, z_1)，Z 轴垂直地面向上，Y 轴指北，各坐标轴对应的转换矩阵是 R_x、R_y、R_z；θ_x、θ_y、θ_z 是转换矢量与 X 轴、Y 轴、Z 轴的夹角。两坐标系之间的转换矩阵 T_t 的计算方法如式(5)~式(9)所示：

$$T_t = R_x R_y R_z T \tag{5}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \tag{6}$$

$$R_y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \tag{7}$$

$$R_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_z & \sin\theta_z \\ 0 & -\sin\theta_z & \cos\theta_z \end{bmatrix} \tag{8}$$

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ x_1 & y_1 & z_1 \end{bmatrix} \tag{9}$$

2.6. 空间索引构建

通过空间索引的构建，使得瓦片数据集存在合理的 LOD 层级划分，确保三维模型数据在加载效率的同时，提高模型渲染速率，由于 3D tiles 瓦片空间索引结构存在相同层级之间可以交互存在，且可以不规则分布，因此本文选用八叉树结构构建空间索引，使用三个正交拆分平面将 tile 细分为八个子级来扩展四叉树。

2.7. 视域剔除

三维模型在加载与渲染时往往只有小部分数据被用于展示，在空间索引建立的基础上，需要对其他数据进行视域剔除，以减少渲染时的干扰，提高数据的加载速率，包括视锥体剔除、背面剔除两个部分。

1) 剔除视锥体

剔除包围盒与视锥体不相交的数据，如图 7 所示，(x, y)为视点的坐标，h 为视域范围半径，α 为夹角，r 为几何误差，h 由以下几何关系构建，如式(10)所示。

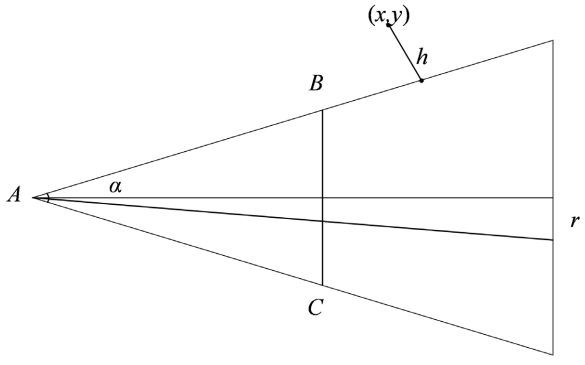


Figure 7. Schematic diagram of culling frustum
图 7. 剔除视锥体示意图

$$h = \left(y - x \times \tan\frac{\alpha}{2} \right) \times \cot\frac{\alpha}{2} \tag{10}$$

若 $h < r$，则不加载该节点；若 $h > r$，加载该节点。

2) 剔除背面

在模型加载与渲染时，物体 *A* 的实线面是可见的，如图 8 所示，虚线面不可见。利用 WebGL 提供的接口函数，可剔除虚线面。

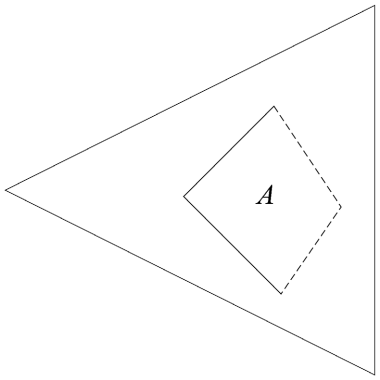


Figure 8. Schematic diagram of culling backface
图 8. 背面剔除示意图

3. 实施流程

3.1. IFC 分解

选择 5 幢、13 幢、15 幢三幢 IFC 格式的 BIM 为测试数据，测试电脑的性能：处理器 Intel(R) Core(TM) I7-11800H @ 2.30 GHz，内存：16 GB，操作系统：Windows 10 64 位专业版。

利用 JGraphT [12]软件执行 IFC 模型的查询与分解，该步骤主要完成三类部件的拆分，首先是 IFC 部件类别，包含了 IfcWindow、IfcDoor、IfcColumn、IfcBuildingElementProxy、IfcBeam、IfcCovering、IfcRailing、IfcFlowTerminal、IfcFurnishingElement；其次是部件的墙体信息，包含了 IfcSlab、IfcWall；第三是部件的组合部件，包含 IfcStair、IfcSite 等，部分执行代码如下：

```
#1= IFCWALL('2kdjgs2dDDfGGFD7dfjg3D',#41,'drywall',$,'STD 16.0:3918',#457,#90);
#2= ICRELASSOCIATESMATERIAL('3DKFdgdkfdg34DFldfdg',#4,$,$,(#43303),#91);
#3= IFCMATERIALLAYER(#930,0.02,$);
#4= IFCMATERIALLAYER(#,0.5,$);
...
#20= IFCMATERIALLAYERSET((#925,#921,#956),$);
#21= IFCMATERIAL('plasterboard');
#22= IFCMATERIAL('plaster');
```

获取了 IFC 模型的各个部件的名称及对应的 ID，并按照 ID 对各 IFC 部件文件进行命名，保存至本地，待下一步转换使用。

将其与 BIMServer [13]中部件信息进行匹配，得到各部件对应的 JSON 格式的属性信息。经测试通过使用图论的搜索方法，查询与分解速率比使用 BIMServer 均有不同程度的提高，如图 9 所示。

经分解后，各类 IFC 部件名称以及各自的数量如表 1 所示。

3.2. IFC 向 3D Tiles 格式转换

IfcOpenShell 是一个开源(LGPL)软件库，用于处理 IFC 文件格式到 OBJ 格式的转换框架，使用

IfcConvert [14]工具将分解后的 IFC 文件批量转换成每个 OBJ 格式的文件，同时生成 MLF 格式的每个部件的材质信息。

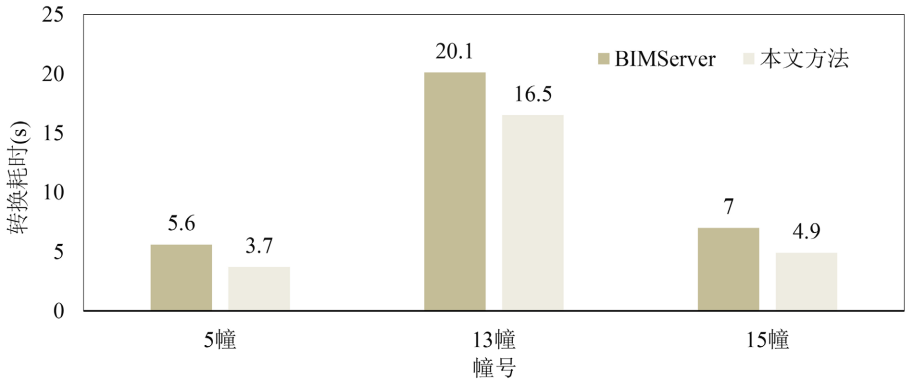


Figure 9. Comparison time of IFC decomposition
图 9. IFC 分解时间对比

Table 1. IFC component name and quantity statistics
表 1. IFC 部件名称以及数量统计

部件名称	5 幢	13 幢	15 幢
IfcBuilding	1	1	1
IfcBuildingStorey	1	1	1
IfcColumn	391	1555	735
IfcDoor	18	1714	626
IfcProject	326	1	1
IfcRoof	40	4	2
IfcSite	1	1	1
IfcSlab	43	59	67
IfcStair	152	248	84
IfcWall	1488	6883	2482
IfcWindow	224	1691	291
IFC 格式大小	8.27 MB	40 MB	13.8 MB
JSON 格式大小	6.24 MB	11.5 MB	4.88 MB

如表 1 所示，由于拆分后每个部件数据较多，为确保转换的高效运行，本文基于 R 语言，编写 shell 语句，执行 IfcConvert 工具的 IfcConvet.exe 程序，批量化实现 IFC 格式向 OBJ 格式转换处理。

obj2glTF [15]工具是由 Cesium 团队开发，用于将 OBJ 格式转换成 glTF 格式的工具，本文使用 obj2glTF 工具将上一步生成的 OBJ 格式文件转换成 glTF 格式。

使用 Cesium 发布的“3D-tiles-tool Sample Generator”[16] (3D tiles 生成器)开源工具，将 glTF 和 JSON (Streaming)文件分组为具有批处理表层次结构的 b3dm 文件，由于各部件的几何形状与属性信息具有一一对应关系，由此在转换过程中不存在信息丢失的情况。IFC 文件分解与格式转换、5 幢 IFC 各部件分解及转换结果如表 2、表 3 所示，在由 IFC 向 OBJ，以及 OBJ 向 glTF、glTF 向 3D tiles 转换的过程中，仍然存在耗时比较长的情况。

Table 2. Time and file size required for IFC file decomposition and format conversion
表 2. IFC 文件分解以及格式转换所需时间、文件大小

	5 幢		13 幢		15 幢	
	耗时	文件大小	耗时	文件大小	耗时	文件大小
IFC 分解(IFC 与 JSON 文件)	5.6 s	14.5 MB	20.1 s	51.5 MB	7.0 s	18.7 MB
IFC 分解(仅生成 IFC 格式)	3.1 s	8.47 MB	16.2 s	41.02 MB	2.8 s	5.0 MB
IFC 向 OBJ 转换	70 s	5.75 MB	366 s	30.1 MB	112 s	9.66 MB
OBJ 向 glTF 转换	371 s	8.89 MB	1795.6 s	42.7 MB	593 s	14.2 MB
glTF 向 3D tiles 转换	3.8 Min	18.9 MB	37.9 Min	89 MB	15.6 Min	33 MB

Table 3. Decomposition and conversion of IFC documents
表 3. 5 幢 IFC 文件分解及转换情况

	IFC 文件数量	OBJ 文件数量	OBJ 文件大小	MTL 文件大小	GlTF 文件大小	b3dm 文件大小	转换成 b3dm 时间
IfcWall	1488	1466	2.9 MB	125 KB	4.6 MB	9.79 MB	2078 Ms
IfcDoor	18	18	36 KB	1.54 KB	56.5 KB	120 KB	87 Ms
IfcColumn	391	391	793 KB	33.4 KB	1.28 MB	2.61 MB	1820 Ms
IfcWindow	224	224	455 KB	19 KB	703 KB	1.5 MB	508 Ms
IfcStair	152	152	308 KB	13 KB	477 KB	1.01 MB	158 Ms
IfcSlab	43	43	87 KB	3.7 KB	135 KB	287 KB	258 Ms
IfcCurtainWall	538	538	1.09 MB	46 KB	1.69 MB	3.59 MB	3553 Ms

3.3. 可视化优化

利用 WebGL 可视化软件对转换后的结果经剔除视锥体和剔除背面可视化的模型进行加载渲染测试，如图 10 所示。以 5 幢为例优化前后加载的帧数均在 55 fps 左右，经可视化优化后，加载帧数稳定且加载时间更短，当模型较大时，帧数不稳定与加载时长较长将会导致用户体验不佳，且优化前后不改变模型的大小，本方法与传统使用几何算法以减少三维模型大小的轻量化方法[17]相比，更适用于较大模型的 Web 端加载与渲染。

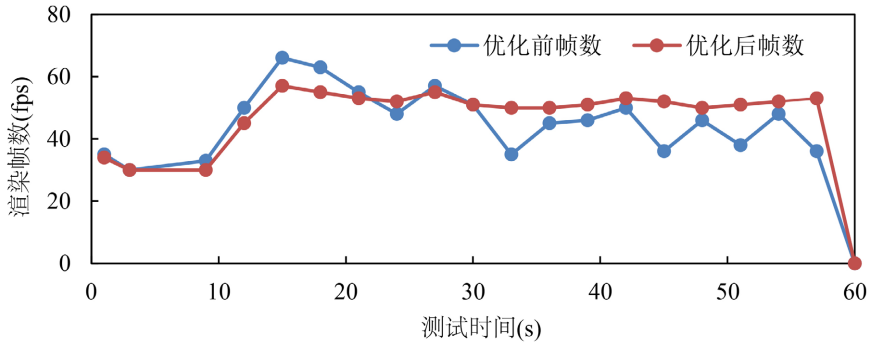


Figure 10. Compare graph of frame rate before and after visual optimization procession
图 10. 优化前后帧数对比图

在 Cesium 环境下，为转换的 b3dm 文件添加 Tileset.json 瓦片集文件，在 Chrome 浏览器下直接使用 URL 地址访问，5 幢的模型效果图(3D tiles 格式)如图 11 所示。

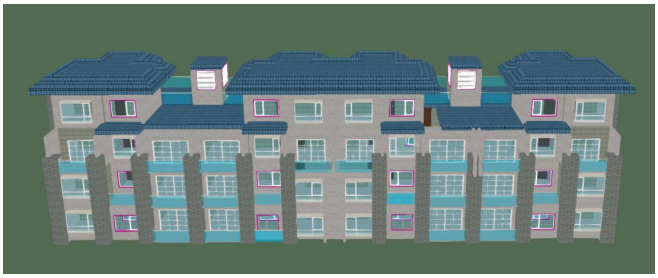
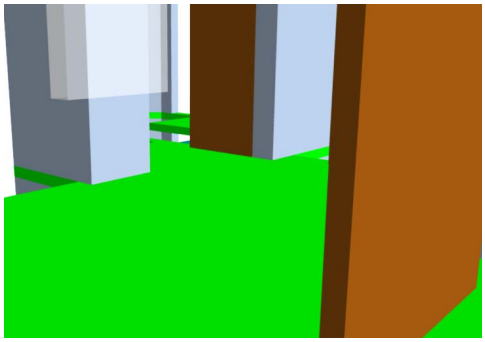


Figure 11. Effect of converting to 3D tiles format
图 11. 转换成 3D tiles 格式效果

以 5 幢的 Slab (厚板)信息为例，IFC 格式的原始模型几何形状与属性信息如图 12 所示，经上述 IFC 的分解和转换，最终生成 3D tiles 格式，在 Cesium 环境下查询该厚板信息，返回的属性信息与原始模型的相同，如图 13 所示，表明本文方法在较好实现格式转换的同时，保持了各部件属性信息。



Property	Value
Type	IfcSlad
Oid	66052
GlobalId	3y$cHbdnb1j8T9JWJEVICe
OwnerHistory	IfcOwnerHistory (65804)
Name	2a2d7a58-dfb7-43be-99b9-f2a686f1b839:2a2d7a58-dfb7-43be-99b9-f2a686f1b839:213539

Figure 12. Information of slab in origin IFC model (Green highlight)
图 12. IFC 原始模型厚板信息(绿色高亮部分)

Figure 13. Information of slab in 3D tiles format (Gray part)
图 13. 3D Tiles 格式下厚板信息(灰色部分)

4. 结论与讨论

本文基于开源语言 R、开源软件 JGraphT、BIMServer、IfcConvert、Obj2glTF、3D-tiles-tool Sample Generator 设计了三维模型 IFC 格式按照部件进行分解，通过格式转换最终得到通用的 3D tiles 格式文件，并保留了模型的属性信息，且在转换过程中无需预知模型的结构信息。在模型可视化优化方面，通过格式转换将模型转换成以瓦片加载的方式，提高了模型加载与渲染效率，实现了对模型的可视化优化。由表 2、表 3 可知，IFC 到 3D tiles 的转换是一个耗时且资源密集型的过程，在最终的 glTF 到 b3dm 转换步骤中，从单个组件构造 b3dm 文件的方式在很大程度上取决于模型的复杂程度，为了优化转换结果，本文下一步考虑使用层次细节层次(HLOD)，这对于提高渲染性能至关重要。本文方法主要对 IFC 2 × 3 TC 格式进行了尝试，对 IFC4 格式支持较少，对 3D 模型的渲染材料考虑较少，这些将作为后续改进的动力。

致　谢

感谢江苏省测绘工程院刘许清高级工程师为本文提供部分素材。

基金项目

江苏省自然资源厅指导性科研项目。

参考文献

[1] Chen, Y., Shooraj, E., Rajabifard, A. and Sabri, S. (2018) From IFC to 3D Tiles: An Integrated Open-Source Solution for Visualising BIMs on Cesium. *International Journal of Geo-Information*, **7**, Article 393. https://doi.org/10.3390/ijgi7100393

[2] 自然资源部. 自然资源三维立体时空数据库主数据库设计方案(2021 版) [EB/OL]. http://search.mnr.gov.cn/axis2/download/P020211015565071755006.pdf, 2021-10-15.

[3] Industry Foundation Classes (IFC)—An Introduction. International Building Smart. http://technical.buildingsmart.org/standards/ifc/?sfw=pass1651025936

[4] 3D Tiles. Open Geospatial Consortium. https://www.ogc.org/standards/3DTiles

[5] 徐照, 张路, 索华, 迟英姿. 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] Buyukdemircioglu, M. and Kocaman, S. (2021) Geovisualizaiton of Aerial Photogrammetric Flights for Data Auqlity Assessment. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **43**, 333-338.

[7] Huang, X.-Y., Yang, S.-F. and Lee, K.-F. (2020) Research on 4D Visualized Dynamic Construction of BIM Building Decoration. *IOP Conference Series*: *Earth and Environmental Science*, **619**, Article ID: 012081. https://doi.org/10.1088/1755-1315/619/1/012081

[8] Tauscher, E. and Smarsly, K. (2016) Generic BIM Queries Based on the IFC Object Model Using Graph Theory. *The* 16*th International Conference on Computing in Civil and Building Engineering*, Osaka, 6-8 July 2016, 1-8.

[9] Moyano, J., León, J., Nieto-Julián, J.-E. and Bruno, S. (2021) Semantic Interpretation of Architectural and Archaeological Geometries: Point Cloud Segmentation for HBIM Parameterisation. *Automation in Construction*, **130**, Article ID: 103856. https://doi.org/10.1016/j.autcon.2021.103856

[10] Gradišar, L. and Dolenc, M. (2021) IFC and Monitoring Database System Based on Graph Data Models. *Advances in Civil Engineering*, **2021**, Article ID: 4913394. https://doi.org/10.1155/2021/4913394

[11] Bulut, O., Shin, J. and Cormier, D.C. (2022) Learning Analytics and Computerized Formative Assessments: An Application of Dijkstra's Shortest Path Algorithm for Personalized Test Scheduling. *Mathematics*, **10**, Article 2230. https://doi.org/10.3390/math10132230

[12] JGraphT: A Java Library of Graph Theory Data Structures and Algorithms. JGraphT. https://jgrapht.org/

[13] BIMServer API. BIM Server. https://github.com/opensourceBIM/BIMServer

[14] IfcOpenShell. IfcOpenShell. https://github.com/IfcOpenShell/IfcOpenShell

[15] Obj2gltf. AnalyticalGraphics Inc. https://github.com/AnalyticalGraphicsInc/obj2gltf

[16] 3D-Tiles-Tools Sample Generator. Cesium Inc. https://github.com/AnalyticalGraphicsInc/3d-tiles-tools

[17] 高喆, 王佳, 周小平, 等. 跨平台的建筑信息模型展示技术研究[J]. 建筑技术, 2017, 48(4): 405-407.