面向手机网页的大规模 WebBIM 场景轻量级实时漫游算法

刘小军^{1,2}, 贾金原^{1*}

1. 同济大学软件学院, 上海 201804

2. 嘉兴学院南湖学院数理与信息工程系, 嘉兴 314001

* 通信作者. E-mail: jyjia@tongji.edu.cn

收稿日期: 2017-11-14; 接受日期: 2017-12-27; 网络出版日期: 2018-02-13

高等学校博士学科点专项科研基金项目 (批准号: 20130072110035)、同济大学中央高校基本科研业务费专项资金——培育类项目 (批准号: 2100219066)、中央高校基本科研业务费——学科交叉类项目重点类项目 A 类 (批准号: 0200219153)、吉林省重点科技攻关课题 (批准号: 20140204088GX) 和长春高新区“长白慧谷”英才计划 (批准号: 3-2013006) 资助

摘要 当今, 使用手机网页来访问大规模建筑环境逐渐被人们所追求. BIM (building information modeling) 是新一代智能建造的核心技术. 本文以其主流格式 IFC (industry foundation classes) 作为输入数据格式, 提出一套面向手机网页的大规模 WebBIM 场景实时漫游算法. 与图形学领域的可视化程序相比, 该方法不只注重可视化渲染时的面片级处理, 更是在遵从建筑语义的前提下, 以建筑构件为单位, 对大规模 WebBIM 场景进行轻量化处理. 方法主要从构件实例冗余、构件及空间混合索引、多粒度数据动态装/卸载策略 3 方面对场景数据量、组织结构、动态数据管理进行优化, 并集成到原型系统. 实验表明本文方法的去冗比例达 70% 左右, 原型系统的 FPS 能保持在 30 左右, 内存损耗约 400 MB, 能较稳定、流畅地在手机网页上进行大规模 WebBIM 场景的实时漫游.

关键词 手机 Web3D, BIM/IFC, 轻量化处理, 建筑空间分析, 索引结构

1 引言

当前, 各行业的数据都在以前所未有的速度增长, 建筑业更是如此. 随着 AEC (architecture, engineering and construction) 产业的发展, 建筑数据几乎包含了建筑生命周期中的所有信息, 数据量急剧增大. 而建筑又与人类生活息息相关, 是众多其他数据产生的基础性平台. 因此, 对大规模建筑数据的高效共享和管理在众多行业领域均有着较深远的影响, 已成为当今数据共享管理的重中之重.

同时, 在 Pocket PC、智能手机普及和轻量级网页应用涌现的今天, 通过手机网页来访问共享建筑信息更将成为新潮流. 本文以智能建造的核心技术 BIM (building information modeling) 为基础, 以

引用格式: 刘小军, 贾金原. 面向手机网页的大规模 WebBIM 场景轻量级实时漫游算法. 中国科学: 信息科学, 2018, 48: 274–292, doi: 10.1360/N112017-00240
Liu X J, Jia J Y. Mobile web-based lightweight and real-time roaming algorithm for large-scale WebBIM scenes (in Chinese). Sci Sin Inform, 2018, 48: 274–292, doi: 10.1360/N112017-00240

BIM 技术的主流数据转换格式 IFC (industry foundation classes) 为源数据格式, 以大规模 BIM 建筑场景在手机平台下的可视化共享为目标, 通过建筑构件去冗、索引构建, 以及多粒度动态管理等过程, 实现面向手机网页的大规模 BIM 场景轻量级实时漫游。

然而, 大规模 WebBIM 场景的手机网页漫游存在着多方面的问题, 其中资源问题最为突出。一方面基于 Web3D 的在线应用总是受到网络传输、浏览设备及网页浏览器等的资源限制; 另一方面, BIM 场景漫游的资源损耗非常高, 远远超出手机应用的系统负载, 如使用单机版商用软件 Autodesk Revit 2016 浏览 300 MB 的 IFC 文件通常需要消耗 4 GB 以上的内存空间。因此, 要在资源有限的手机浏览器中进行在线漫游, 必将是极具挑战性的问题。

当前, 出现的可视化渲染公共平台或程序已有不少, 技术也较成熟, 但在处理渲染任务时, 公用平台通常只能从一般性渲染管线等层面上进行优化, 数据处理单位主要为几何片元。而对 BIM 场景来说, 场景本身就涵盖了多方面的建筑信息, 直接使用公用渲染平台技术, 还难以对这些建筑信息的实时访问以及关系分析等提供支持。也正是由于这些原因, 本文基于公用平台技术, 将 BIM 场景按建筑构件为单位进行数据优化, 使其更符合轻量级实时漫游的要求, 主要面临如下挑战:

(1) BIM 场景对应的模型数据量大, 相对冗余度高。IFC 标准涉及的范围较广, 导致其本身的信息量过多, 模型的数据量过大。另在数据表示时多个构件可能同时引用同一份几何数据, 还可能部分模型为参数化模型, 导致构件间存在大量的冗余数据。

(2) 逻辑结构复杂, 场景组织松散。IFC 标准包含了近 1000 个预定义的实体类型, 而且实体间的引用还必须遵循严格的层次性规则, 使得 IFC 场景逻辑结构复杂。如图 1 所示的 “#13376= IfcBeam” 构件中, 要访问到真正的几何数据, 至少需经过一棵高度为 7 的树, 增加了数据访问复杂度。同时, 构件间的相邻、包含等关系, 更体现了建筑方面的需求, 未考虑轻量级实时漫游时的场景组织和管理。

(3) 资源损耗大, 动态装载信息缺乏。系统资源有限是 Web3D 应用的一大特点, 需要设计高效的动态数据管理策略。避免在一次性装载整个 BIM 场景时, 由于损耗的资源超出系统负载而出现异常情况。然而, IFC 数据中并未包含数据装载优先级等相关信息, 远无法支持数据动态装载和更新。

针对以上问题, 本文以建筑构件为组织单位, 从实践的角度提出了一套面向手机网页的大规模 WebBIM 场景轻量级实时漫游算法。主要的创新点如下:

(1) 基于内容匹配的 BIM 构件快速去冗。当前的 BIM 可视化应用通常需要几何数据的在线解析, 较少考虑展示前的解析等待时间、展示时的数据规模及冗余等。本文充分利用 BIM 构件的语义内容, 提出先通过构件内容分析缩小可能存在冗余的 BIM 构件范围, 再引入少量的几何计算进行冗余确认的方式, 能快速高效地完成 BIM 构件的去冗工作。

(2) 基于空间分析的 BIM 建筑混合索引。在承载量有限的 Web3D 应用中, 不仅需要快速的数据访问, 更需要高效的数据剔除。基于空间分析的 BIM 建筑混合索引, 在分离 BIM 建筑室内外数据的同时, 进一步对复杂室内空间进行 “楼层 (storey) – 空间 (space) – 构件 (product)” 等多粒度分解, 使得场景漫游时能根据视点位置、漫游方式等实时调度和剔除合适粒度的数据。

(3) 多粒度数据渐进式装载。数据的实时更新是在线漫游系统的基本要求之一。本文在结合传统视域等管理策略的同时, 通过空间触发区管理、构件优先级预估等策略, 对粒度间区域过渡、粒度内构件加载顺序等进行实践性优化, 尽力将处理单位数据的资源损耗降到最低。

图 1 (网络版彩图) IFC 文件结构
Figure 1 (Color online) IFC file structure

2 IFC 标准

IFC 是一套专为建筑行业制订的开放数据格式标准, 致力于构筑物整个生命周期内的信息共享和交换^[1,2]. 整个标准采用面向对象语言 EXPRESS 描述, 通过预定义一系列的实体 (entity) 类和内部数据类型, 进一步定义相应的数据对象. 如下即为一个 IFCAPPLICATION 类实例对象的定义示例:

#6 = IFCAPPLICATION(#5, '2015', 'SketchUp Pro 2015', '2015'),

其中, 实例名称为 “#6”, 其内部成员则通过括号中逗号隔开的 4 个参数 #5, '2015', 'Pro 2015', '2015' 来进行初始化. 而第 1 个参数 #5 则表示对象 #6 引用实体对象 #5 来进行自身的初始化, 这也是表示对象间关系的常见方式.

总体上, IFC 标准中分层组织了上千个预定义的实体类型, 且为了规范, 以层的方式定义了类或类型间的引用规则. 新建数据时, 不仅要遵守这些引用规则, 还需考虑实际应用中对象间的关系. 如图 1 所示, 一个 IFC 文件通常从一个项目 (IfcProject) 实例出发, 逐级分解, 才到所有包含的墙 (IfcWall)、门 (IfcDoor) 和梁 (IfcBeam) 等建筑构件 (product). 而真正的几何数据, 则是由类似图中的梁等构件分别进行引用和组织的. 可见, 原始的 BIM 场景数据沉重且复杂, 要实现轻量级在线漫游, 需要从数据和结构等多方面进行优化.

3 相关工作

在图形学领域, 网络化渲染技术已较成熟, 出现的成果也较多. 但要将其移植到 BIM 建筑领域, 仍有较大的发展空间. 因为 BIM 标准本身也仍在完善, 这无疑会影响到对其可视化分析的方法和手段, 最新版 IFC4^[2] 已于 2013 年 12 月发布, 该版本通过引入模板文件、剖面素材等方式更加丰富了建筑物元素. 同时, 也出现了较多与本文相关的其他工作.

3.1 BIM 可视化

基于 BIM 技术的软件已有很多, 著名的有 Autodesk 公司的 Revit 系列, 以及 Bentley 公司的 Bentley Architecture 等. 它们功能强大, 在对 BIM 数据进行可视化的同时还专注于 BIM 数据的编辑. 也正因为它们的功能强大, 导致在操作 BIM 数据时需要的系统资源远远高于其他功能相对简单的工具. 而且随着这些工具的发展, 出现的 BIM 文件格式也逐渐增多. 在 BIM 主流的标准格式 IFC 发布以后, 大部分的 BIM 工具都迅速将其作为数据格式之一, 以方便格式之间的转换. 如解析程序库 BIMServer¹⁾ 将 IFC 解析后以实体的方式存入数据库中, 为众多软件提供数据支持. FZK Viewer²⁾, BIMVision³⁾ 等即属于该类工具中的单机版, 支持手机平台或网页版的则以基于 WebGL 技术的 BIMviews⁴⁾ 和 BIMSurfer⁵⁾ 为代表. 同时, 随着 BIM 技术的发展, 更多的网页轻量级 BIM 应用正在出现, 如仍在完善的 A360, 以及 BimViz 等. BIM 可视化方面的理论研究也在进行, 如文献 [3] 从理论的角度对现有的 4 种 BIM 可视化工具进行了对比性分析. 文献 [4, 5] 分别从建筑模型可视化、增强现实应用集成、多视角展示研究、图片和模型相结合等方面进行 BIM 场景的可视化探讨.

3.2 BIM 数据去冗

关于 BIM 数据去冗, 与本文最为相关的有 Solibri Solution Center 提供的 Solibri IFC Optimizer (以下简称 SIO)⁶⁾, 以及清华大学 Liu 等^[6] 提供的 IFCCompressor. 两项成果中均采用内容分析的方式对 IFC 场景文件进行去冗和压缩等优化, 其中后者更在语义分析方面有相应的改进. 此外, 文献 [7~10] 也从 IFC 场景数据的相似性展开讨论, 将语义信息引入模型搜索过程, 为使用语义信息来搜索 IFC 大规模场景的元素相似性提供了依据; 文献 [11~13] 更是对 IFC 文件内、文件间的语义分析比较做了深入探讨. 这些方法侧重于讨论实例数据在某些语义属性或字符取值相等层面上的相似程度或相关性, 如 owner history 属性、GUID 值、相邻关系、包含关系等. 然而, 对于面向 BIM 数据可视化的优化, 还必须将语义数据与可视化的几何数据相关联, 降低可视化过程中的数据量. 如文献 [14] 提出的 3D 模型几何分割方法; 文献 [15, 16] 则通过一定的语义性规则来过滤数据, 或建立数据之间的对应关系, 来减少后续复杂几何计算的规模, 也给本文将语义内容分析和几何构件查重相结合提供了借鉴.

3.3 BIM 场景管理

场景的组织和管理在大规模场景数据的调度优化等方面尤为重要. 当前针对一般场景管理的代表性工作有可见性剔除^[17, 18]、兴趣区域研究^[19], 以及场景索引结构^[20, 21] 等, 分别从场景数据的可视

1) BIMServer. <http://www.bimserver.org>.

2) FZK Viewer. <http://www.iai.fzk.de/www-extern/index.php>.

3) BIMVision. <http://www.bimvision.eu/home/>.

4) BIMviews. <http://www.bimview.fr/>.

5) BIMSurfer. <http://bimsurfer.org/>.

6) Solibri IFC Optimizer. <http://www.solibri.com/>.

图 2 (网络版彩图) 本文技术路线
Figure 2 (Color online) Pipeline for our method

域判断、不可见区域剔除, 以及数据的高效访问等进行探讨. 面向 BIM 数据相结合的相关算法也陆续出现, 如文献 [22] 中使用八叉树和 LOD (level of detail) 相结合的方式实现了 BIM 大场景的实时处理; 文献 [23] 则提出使用 ORDB (object-relational database) 代替传统的关系数据库来管理复杂的 BIM 大规模场景, 简化了从 IFC 结构到关系数据库的转化.

我们前期也有部分工作 [24] 与基于 Web3D 的大规模 BIM 场景可视化相关, 其中对场景轻量化、建筑外体提取等有相应介绍. 与本文方法相比, 虽然这些前期工作集中在建筑构件的优化处理上, 数据加载时注重对视域范围进行有效管理 (称为 frustum of interest, FOI 管理) 和进行构件数据调度, 未深入到本文方法中遵从建筑结构, 以建筑空间为粒度的数据装载等程度, 但这些工作为本文方法的开展做了多方面的准备, 本文方法在一定程度上可以看作是对前期工作的进一步推进.

与 BIM/IFC 相关的工作还有很多 [25], 但真正关于手机网页的大规模 BIM 场景漫游工作仍较少. 官方仅有的少量网页可视化应用也主要以插件形式, 做为现有服务器程序, 如 BIMServer 等的某个子模块存在, 通过对原始数据进行在线直接解析的方式进行 BIM 数据可视化. 不仅对这些现有服务器程序的依赖性较大, 还不支持对原始数据的去冗、结构重组等预处理操作及碰撞检测等漫游效果. 要达到基于手机平台下大规模 BIM 场景的、真正免插件网页漫游还需要多方面的努力和更深入的研究.

4 技术路线

整体的技术路线主要分为 3 部分 (见图 2): BIM 场景去冗处理、建筑空间混合索引构建和多粒度动态管理. 其中前两部分主要是对原始数据的优化处理, 在服务器端离线完成; 第 3 部分则面向手机网页漫游, 是对前面部分的合理调用和集成, 最终发布为真正的示范性应用. 具体如下:

- **BIM 场景去冗处理.** 当前的 BIM 应用往往将场景解析和渲染集成到一起, 较少考虑基于手机平台下网页应用的系统资源限制, 更未涉及去冗等操作. 本部分通过快速匹配方式来定位 BIM 场景中的冗余数据, 并对其进行去冗操作, 降低网页端在线处理的负担 (详见第 5 章).
- **空间混合索引构建.** BIM 标准虽覆盖了多方面的建筑信息, 但这些数据主要关注的是建筑物的整个生命周期, 并未考虑场景的在线展示等应用需求. 多粒度混合索引在进行建筑空间分析的基础上, 建立更加合理的场景索引结构, 提高场景数据的访问和更新效率 (详见第 6 章).

● **多粒度数据动态管理.** 服务器端预处理后, 整个建筑被细分为由楼层、空间等表示的多粒度数据, 将被网页端程序进一步调用. 如何在不影响用户体验的同时, 设计合适的策略能及时、准确地管理并调度最少的数据来完成在线展示便是该部分需要考虑的问题 (详见第7章).

5 BIM 场景去冗处理

BIM 场景去冗处理就是通过构件匹配算法找出重复构件并去除这些重复构件的过程. 其中模型的相似性匹配在计算机视觉领域已存在很多^[26~28]. 然而, 这些方法主要依据特征数据来完成形状匹配, 通常需要在高维空间下构建图元对象规模的特征向量集来描述整个形状的特征, 再通过聚类、统计等较复杂的方法来得出匹配结果.

注意到 BIM 构件包括几何和语义两部分, 其中几何构件主要由 IfcProduct (以下简称构件) 类型派生的实例组成, 如 IfcDoor, IfcWindow 等的实例; 语义数据则体现为实体定义时的文本内容和实例间的相互引用关系. 在此先通过语义内容分析出可能重复的构件对, 再在这些可能重复的构件对间进行几何相似性验证以实现快速去冗.

5.1 构件内容匹配

分析 BIM 数据中丰富的语义信息, 可知构件在内容上包含了一个或多个 IfcRepresentationItem (以下简称 ITEM) 实例, 且经过 ITEM 内容分析可得出更多的构件信息. 在此以构件内容分析为主要手段, 先得到构件的 ITEM 映射矩阵, 再通过简单的矩阵运算来估计构件几何匹配关系, 避免了直接使用几何形状匹配中的面片级复杂计算过程. 具体步骤如下:

(1) 对 BIM 场景中包含的全部 n 个构件 $P_1, P_2, \dots, P_n$, 分析构件内容, 根据构件类型和包含的 ITEM 数量初步形成多个构件对. 依次对每个构件对执行步骤 (2), (3).

(2) 对于构件对 $\langle P_a, P_b \rangle$, 分别解析 P_a, P_b 中的 m 个 ITEM 实例, 形成集合 $S_a = \{R_{a,1}, R_{a,2}, \dots, R_{a,m}\}$ 和 $S_b = \{R_{b,1}, R_{b,2}, \dots, R_{b,m}\}$. 将 S_a 和 S_b 看作 P_a 和 P_b 在构件内容空间上的两个“特征向量”, 通过分别查找和比较每个 ITEM 的 IfcMappedItem 类型以及语义内容 (如实例名称、MappingSource 信息等) 可初步得到构件 P_a 和 P_b 间的内容匹配关系, 大致过程如图3所示.

(3) 通过第 (2) 步的过程, 整理构件对 $\langle P_a, P_b \rangle$ 间的内容匹配关系并记录为一个由 0 和 1 组成的 $m \times m$ 的内容匹配矩阵 C , 其中元素 $C(i, j)$ 的取值如下:

$$C(i, j) = \begin{cases} 1, & R_{a,i} \text{与} R_{b,j} \text{内容一致,} \\ 0, & R_{a,i} \text{与} R_{b,j} \text{内容不一致,} \end{cases} \quad 1 \leq i, j \leq m, \quad (1)$$

仅当 C 中每行或每列均存在 1, 即对构件 P_a 中的每个 ITEM 总能在 P_b 中找到一个 ITEM 与其映射关系相同时, 则认为构件内容匹配.

在构件匹配过程中, 还可通过“迁移”前期匹配结果来进一步提高匹配效率. 如在匹配 P_5 和 P_8 时, 查询到它们在前期匹配中已都与构件 P_t 相匹配, 则可直接借助前期的匹配矩阵推算出它们之间的匹配矩阵, 完成匹配“迁移”.

5.2 匹配构件去重

构件内容匹配借助构件的 ITEM 信息完成了语义上的构件预匹配, 从很大程度上避免了对几何差异较大的构件进行相似性计算等复杂过程. 但若继续对 ITEM 实例内容进行深层次分析, 将涉及

图 3 (网络版彩图) 构件内容匹配示意图

Figure 3 (Color online) Diagram of product content matching

到 Boolean 操作、参数化等几何计算过程. 匹配构件去重就是在前期匹配的基础上, 通过规模较小的 ITEM 级别的几何计算来度量构件间的相似度, 进一步确认需要去重的构件.

在计算构件几何相似度时, 需要先将两个构件 (P_a 和 P_b) 进行对齐, 本文采用较成熟的 ICP (iterative closest point) 方法来完成. 但考虑到 ICP 对输入模型的初始位置较敏感、计算规模大等原因, 在此结合式 (1) 中的内容匹配矩阵 C , 通过两次小规模 ICP 优化来进行, 其中第 1 次从全局层面上只通过一次迭代估计两个构件的大致变换关系, 为第 2 次精确匹配时构件的初始位置做调整, 克服 ICP 对模型初始位置的敏感性, 具体如下:

首先, 根据式 (1) 中的矩阵 C 选择所有内容上一一匹配的 ITEM 实例对, 且要求选择的实例对数量不少于 3, 否则从未被选择的一对多匹配中随机选择两个匹配 ITEM 作为一一匹配来进行补充. 计算这些匹配 ITEM 实例的重心, 形成两个有匹配关系的重心序列, 将这两个重心序列作为第 1 次 ICP 匹配的输入, 并通过一次迭代得出两个构件之间的变换关系 M .

其次, 将变换关系 M 应用到 P_a , P_b , 使得两个构件的相对位置得以调整, 并将调整后的两个构件作为第 2 次 ICP 匹配的输入, 结合两个构件中 ITEM 的几何数据进行第 2 次 ICP 迭代.

同时, 为了提高稀疏点集的对齐精度, 两次 ICP 均借鉴文献 [29] 中的稀疏 ICP 方法, 最终采用 Hausdorff 距离 (HD) 来判断对齐后的实例是否重复. 具体计算时通过式 (2) 所示的方式, 对所包含的

图 4 (网络版彩图) 主流解析器和本文去重处理后的构筑物数据对比

Figure 4 (Color online) Example of building data between mainstream parsing method and our duplication removal method

ITEM 相似度进行面积加权来得到两个构件的相似度

$$\text{Similarity}_{\text{product}}(P_a, P_b) = \frac{\sum_{i=1}^m [(\text{Area}(P_a^i) + \text{Area}(P_b^i)) \text{Similarity}_{\text{item}}(P_a^i, P_b^i)]}{\sum_{i=1}^m (\text{Area}(P_a^i) + \text{Area}(P_b^i))}, \quad (2)$$

其中 P_a, P_b 表示两个构件, P_a^i 和 P_b^i 表示这两个构件中相匹配的 ITEM 实例, 其几何相似度即使用式 (3) 中 $\text{Similarity}_{\text{item}}$ 进行计算, m 为 ITEM 个数.

$$\text{Similarity}_{\text{item}}(P_a^i, P_b^i) = 1 - \frac{\text{HD}(P_a^i, P_b^i)^2}{\max(\|P_a^i - P_b^i\|^2)}. \quad (3)$$

当相似度大于某一阈值 (在此取 0.9) 时, 则认为对象重复, 并使用 ICP 返回的变换矩阵进行去重处理. 图 4 即为一个 BIM 构筑物在去重前后的逻辑结构对比示例.

此外, 对于冗余数据, BIM 建筑还存在部分参数化实体数据, 如 `IfcExtrudedAreaSolid` 及 `IfcSweptAreaSolid` 等. 在进行去重处理的同时, 还对部分拉伸型构件, 使用几何计算方法对其拉伸模型进行验证性拟合, 使用参数化模型表示构件几何数据, 减少参数化计算带来的冗余数据.

6 空间混合索引构建

场景索引结构极大地影响着场景数据的访问效率, 文献 [20] 在总结当前场景索引结构的同时, 更强调了索引结构的设计要充分考虑到数据本身的特点. IFC 文件中虽包含了 `IfcBuildingStorey`, `IfcSpace` 等与构筑物空间相关和 `IfcRelContainedInSpatialStructure`, `IfcRelDecomposes`, `IfcRelCoversSpaces` 等表示空间关系的实体信息. 但由于应用领域等的差异, 还远不能用来识别空间数据. 本文将整个场景的

图 5 (网络版彩图) 外体提取流程
Figure 5 (Color online) Workflow of exterior extraction

室内外构件进行分层索引, 室外层建立整个场景的宏观性结构, 再对室外层节点对应的复杂构筑物进行室内结构细化, 形成包含场景层、室外索引层、室内索引层和构件层的 4 层多粒度混合索引 FLHS (four-layered hybrid structure). 如此, 索引后的多粒度数据既可从可视性出发支持最小粒度数据的拾取、剔除及更新等, 提高资源利用率, 更为后续的室内路径规划^[30]等其他扩展做数据准备.

6.1 室内外空间分析

考虑到构筑物室内外特点, 通常将构筑物分解为一个表示构筑物外壳的外体空间和多个位于构筑物内部且呈封闭状态的内体空间, 并进一步分析这些空间之间的连通性.

6.1.1 外体空间抽取

外体抽取过程在我们前期的工作^[24]中已有较完整描述, 整个抽取过程从直观认识“外体即室外可观察到的所有构件”出发. 通过在构筑物的包围椭球上采样形成“投影点”, 将整个构筑物由椭球中心向“投影点”投影并进行投影分析, 收集分析结果中的构件作为构筑物外体 (如图 5 所示).

6.1.2 内体空间分析

室内空间 (如图 6(a) 所示) 分析时, 首先将整个构筑物分解成以 IfcBuildingStorey 为单位的多个楼层; 其次再将楼层进一步分解为多个 IfcSpace, 用于表示房间等相对封闭的空间; 最后, 通过分析空间的关系性实体及其参数, 如 IfcRelSpaceBoundary 实体的 RelatedBuildingElement, ConnectionGeometry 和 InternalOrExternalBoundary 参数, 以及 IfcRelContainedInSpatialStructure 实体的 RelatedElements 和 RelatingStructure 等得到属于当前空间的构件数据并完成整个分解过程 (见图 7). 当然, 整个分解

图 6 (网络版彩图) 构筑物空间分析示例

Figure 6 (Color online) Example of building space analysis. (a) Horizon section of the building; (b) result of space extraction; (c) result of space connected graph construction

图 7 (网络版彩图) 建筑空间语义信息

Figure 7 (Color online) Semantics information for building space

图 8 (网络版彩图) 室内空间分析流程

Figure 8 (Color online) Workflow of interior space analysis

过程中可能会出现语义信息的 Space 与视觉尺度下的构筑物空间不一致等情况, 如实际中的同一空间对应多个 IfcSpace 实体等. 几何计算则用于通过计算的方式来辅助和补充这些信息.

内体分析的大致流程见图 8, 几何计算处理作为图中 BIM 空间分析过程的一部分. 本文主要通过体素追踪的方式来完成几何计算, 具体见算法 1. 如此, 整个构筑物在室内外被分解为一个外体空间和多个内体空间, 且每个空间有统一的数据结构. 如图 6(b) 即为图 6(a) 的室内空间分析结果.

6.2 混合索引构建

如图 6(b) 所示, 构筑物结构分解后, 室内主要以空间形式管理. 但只分解成独立的室内空间还无法进行空间之间的切换和空间数据的动态装载. 需要在分解室内空间结构的同时, 考虑记录室内空间之间的连通关系、连通属性及连通数据等.

通过语义分析得到空间之间共用的构件, 如邻近房间的共用墙, 上下层房间共用的地板等; 再用构件名称和几何计算, 找出用来连通这些空间的相应构件; 最后记录并生成空间连通图. 主要体现在 IfcRelSpaceBoundary 实体的 RelatingSpace, RelatedBuildingElement, ConnectionGeometry 和 Inter-

算法 1 基于体素追踪的 BIM 构筑物内体分析算法

Input: 构筑物所有构件数据.
Output: 构筑物内体数据.

- 1: 体素化输入的构件并构建体素索引, 记索引体素集为 A ;
- 2: 在索引体素外的空体素集中进行采样;
- 3: **for** 采集到的每个空体素 v **do**
- 4: 将 v 作为种子点向 X, Y 两个轴向发出射线;
- 5: 在 A 中找出与射线正负相交且距离 v 最近的体素集 B ;
- 6: 在 A 中对 B 包含体素进行邻接性追踪, 形成体素集 C ;
- 7: 向 Z 方向平移 C 中体素, 直到与空间顶底体素相交;
- 8: 收集平移时经过的体素, 将其作为新空间的索引;
- 9: **if** 新空间已存在 **then**
- 10: 直接进入下次循环;
- 11: **else**
- 12: 查询新空间索引对应的构件作为新空间的构件;
- 13: **end if**
- 14: **end for**

图 9 (网络版彩图) 4 层混合索引结构

Figure 9 (Color online) Four-layered hybrid structure (FLHS)

nalOrExternalBoundary 参数. 对于起连通作用的构件, 可先查找共用构件中的 IfcDoor, IfcStair 等与 IfcOpeningElement 实体相关联的“出/入口”构件, 再计算连通空间之间的最短距离用作辅助信息来确定连通分析的正确性, 如图 6(c) 即为图 6(a) 所对应的室内空间连通图.

经过以上过程后, 混合索引 FLHS 构建完成. 整个 FLHS 相当于一棵高度为 4 的树 (如图 9), 其中, 根节点为场景分块形成的表示位置的虚拟结点, 第 2, 3 层才是真正的索引信息, 方便准确访问第 4 层中的构件库数据. 索引信息分别用于索引外体和室内构件数据, 且同一构筑物的室外和室内通过第 2,

3层的父子关系来体现,方便室内外场景的动态切换和相应粒度数据的操控.

7 多粒度数据动态管理

Web3D应用的缓存有限,致使较多应用在初始化阶段就因一次性装载整个场景而导致“超载”并被强制中止,即使这些初始化数据有较多将在绘制阶段被剪裁掉.在此通过建立混合索引,实现索引粒度动态管理,支持从可视角度只装载部分必要数据进行初始化.

7.1 室内外触发式更新

在使用视锥剔除等常规数据裁剪技术的同时,还充分利用混合索引进行空间遮挡,分别以室外-室内、室内-室内、室内-室外3种空间切换模式进行数据剔除和更新.逻辑上,可通过图9中位于第2,3层的父子节点来完成.但在实现时,空间之间的平滑过渡尤为关键.在此通过空间触发区的统一管理来进行底层支持.为了方便,首先使用如下代码来统一描述构筑物的空间数据.

```
struct Space
{
    string          spaceName;    //名称
    Data            spaceData;    //数据
    array<string>   relatingProduct; //出入口
    array<string>   relatedSpace;  //所有的连通空间
    array<float>    preloadRange;  //触发区范围参数
};
```

可见,空间结构中不只保留当前空间信息,还保留了与该空间直接相连通的邻接空间的部分信息,以及加载这些邻接空间时的预加载范围,即在空间内漫游时,通过视点与触发区的相对位置(即代码中的preloadRange)来提前或延后相应邻接空间的更新过程.使得在穿过建筑出入口时即已预加载并能及时观察到相应的空间数据,离开原始空间足够距离(即延后)时才卸载原始空间.如图10即表示从图6(a)中的空间 S_1 穿过入口 D_1 进入连通空间 S_2 漫游的流程图,相应的预加载距离计算如下:

$$R(S_1, S_2) = \left(\frac{\text{Size}(S_2)}{V_{\text{load}}} + \Delta t \right) \times V_{\text{roam}}, \quad (4)$$

其中, $\text{Size}(S_2)$ 表示空间 S_2 的数据量, V_{load} 为数据装载速度, V_{roam} 则表示漫游速度, Δt 为时间延迟.

图11即为图6(a)中所展示路径的室内空间渐进式装载情况.可见,在多粒度数据管理下,尽管构筑物中共包含8个室内空间,但整个漫游过程只装载当前可视的、不超过两个的空间数据,大大减小了需要处理的数据量.另外,预加载操作与当前的漫游操作并行执行,保证漫游的流畅性,且预加载触发区的管理起到一定的缓冲作用,防止了用户在同一入口附近徘徊导致的同一空间高频装/卸载问题.

7.2 构件优先级装载

在线漫游时,首次装载的用户等待时间极大地影响着用户体验质量.基于“体积越大越易被观察到,数据量越小越快被传输和渲染”等直观认识,统一对建筑的所有构件进行统计性分析并通过视锥填充度(frustum fill ratio, FFR)来对其量化

$$\text{FFR}(A) = \frac{\text{Volume}(A)}{\text{Size}(A)}, \quad (5)$$

图 10 (网络版彩图) 室内空间渐进式装载案例

Figure 10 (Color online) Progressive data loading case of interior space

图 11 (网络版彩图) 渐进式装载时的内存变化情况

Figure 11 (Color online) Memory update status with progressive data loading

其中 A 为构件实例, Volume 和 Size 分别表示 A 的几何体积和数据量. 如图 12 即为某 BIM 构筑物示例中各类构件的 FFR 分布情况, 其中 Size 以 MB 为单位, 可见与实际情形一致, 体积大但外形较简单的顶和底 (covering 和 slab) 以及立面 (wall) 构件有明显较高的 FFR 值, 应该优先加载.

此外, 通过实践测试发现, 当 FFR 相差不大时, 距离视点近的构件应优先加载, 故将构件加载优先级 Priority 做简单定义如下:

$$\text{Priority}(A|\text{View}) = \frac{\omega \times \text{FFR}(A)}{\text{Distance}(A, \text{View})}, \quad (6)$$

其中 A 为构件实例本身, View 为视点位置, ω 为调整参数 (默认取 0.5), Distance(A , View) 则表示构件与视点之间的距离.

由于优先级计算的频率比较高, 为了降低网页端的压力, 本文尽量使用简单运算来对其定义; 同时 FFR 为常量, 可通过服务器预先计算; 最后, 在进行优先级计算时可任取构件所在的索引体素与视点位置进行优先级的粗略快速估算.

8 算法实验与性能分析

在进行算法完善的同时, 从公开数据集 Open IFC Model Repository^[31], buildingSMART, International Alliance for Interoperability (IAI) 官方示例, 以及当前实际项目中收集不同规模的 12 个原始构筑物场景组成实验数据进行验证性实验, 同时与 SIO, IFCCompressor 进行结果对比. 实验中采用的笔记本电脑配置为 Intel i5-3210M 处理器, 8.0 GB 内存, NVIDIA GeForce 610 M 显卡及 64 位 Windows 8 操作系统. 测试数据的 IFC 版本统一为 IFC2X3, 具体见表 1.

图 12 (网络版彩图) 某构筑物中的 FFR 分布情况
Figure 12 (Color online) FFR distribution from a building example

表 1 实验场景列表
Table 1 Test models in our experiment

Model ID	Model name	Data size (MB)
1	Duplex_A_20110505	2.311
2	301110FZK-Haus-EliteCAD	7.191
3	Schultz_Residence	21.94
4	0912102010-03-01 Project	50.726
5	161210Med_Dent_Clinic_Combined	109.996
6	SG-CsSj_Project	154.27
7	SG-M13-CZ-WNL-ARC	270.093
8	Zg_mobileRoom-1	369.151
9	Ch-Subway-130215	384.438
10	Cgm-Project-12	429.433
11	Zhe-YY-Scene-1	1854.62
12	Luyanon-Project	2904.77

8.1 构件去重及空间分析

构件去重和外体抽取的部分实验结果如图 13, 14 所示. 可见, 本文方法去重比例明显高于 SIO 和 IFCCompressor 方法, 且在场景规模增加时, 去重效果明显且稳定, 平均去重比例 60% 左右. 虽然后两种方法的去重效果也较明显, 但当数据达到一定规模时 (如图 13(a) 中的场景 8~12), 这些方法已不够稳定, 甚至在运行时崩溃并退出. 在构件去重支持下, 外体抽取效果同样明显, 去除的内体数据量约达总数据量的 70%. 另外, 图 13(b) 描述了对各测试数据中包含的构件数据量排序后, 前 13 类构件与人工标注结果相比下的查重结果 (若无结果表示当前数据中不包含该类构件), 可知, 随着 BIM 建筑的复杂化, 构件数量的增加, 整体的重复检测率在缓慢下降, 并逐步稳定在 90%~95% 之间, 且在各类构件中, 几何信息丰富且特征较明显的 Door, BuildlingElementProxy 等构件重复更易被算法所识别.

图 13 (网络版彩图) 构件去重及外体抽取结果分析

Figure 13 (Color online) Result analysis of duplication removal and exterior extraction. (a) Data reduction analysis and (b) duplicate product detection analysis (duplicate product detection ratio = duplicate number detected from our method / duplicate number from manual annotation $\times 100\%$)

图 14 (网络版彩图) 构件去重效果

Figure 14 (Color online) Visible comparison before and after duplication removal

8.2 场景漫游性能分析

在线漫游方面, 首先将本文与我们前期的 FOI (frustum of interest) 数据管理和调度方法^[24]进行比较, 其中 FOI 参数采用文献中的默认值. 通过在同一场景、同一视点时使用两种方法进行构件数据调度和结果观察 (见图 15), 可见在 FOI 方法中虽能较好地剔除不可视数据, 但在整个构件加载过程中顺序较随机, 离视点较近或 FFR 较高的构件 (如图中的立面墙对象) 往往未优先加载, 特别是遮挡性较高的外墙构件, 很大程度上影响了用户体验质量. 同时, 比较装载完成后的结果可以看出, FOI 方法中最终加载的构件数据为 3882 个, 仍有部分外墙未被包含在内, 说明只通过统一设置视域范围还无法对渲染过程的遮挡剔除、建筑的空间特点等做较全面考虑, 而本文方法的构件 FFR 优先级调度、建筑空间分析和可加载到 4317 个构件, 更能较准确地装载可视数据.

为了进一步分析, 将本文的原型系统与另外两种同样基于 WebGL 且手机平台下运行较稳定的官方工具: BIMviews 和 BIMSURFER 进行性能对比, 其中这两种方法中依赖的 BIMServer 服务器程序均使用当前最新版 BIMServer1.5.76.

为了便于获取和分析 BIMviews 和 BIMSURFER 的内存损耗、FPS 等参数, 同时降低不同设备在硬

图 15 (网络版彩图) 前期方法^[24]与本文方法的比较

Figure 15 (Color online) Visible comparison between Liu's method [24] and ours

图 16 (网络版彩图) 场景漫游中的性能分析

Figure 16 (Color online) Performance analysis while roaming in scenes. (a) Memory consumption analysis and (b) FPS analysis

件配置、网络环境等方面的影响, 场景漫游测试均在笔记本电脑的网页浏览器上进行, 并通过多次测试取平均值作为测试结果. 具体的服务器配置为 Intel Xeon E3-1220 v3 处理器, 16.0 GB 内存, NVIDIA K620 显卡及 64 位 Windows Server 2008 标准版操作系统. 如图 16(a) 为 3 种方法中的平均内存损耗分析结果, 可以看出, 随着场景规模的增加, 整体的内存损耗都在增加, 然而, 其他两种方法增加的速度更快, 且数据量达到场景 8 的规模时, 整个系统已不够稳定, 出现浏览器崩溃现象. 本文方法则在内存增加的过程中逐渐趋于稳定, 更适于处理大规模 BIM 场景.

在进行 FPS 对比时, 借助 Chrome 浏览器的 Rendering 监听器, 将 Paint Flashing 区域固定到 3D 场景绘制区域以进行 FPS 监听, 具体的 Chrome 版本为 64 位的 59.0.3071.115 版本.

从图 16(b) 可以看出, 随着场景规模的增加, 整体的 FPS 值逐渐降低, 且从场景 8 开始其他方法已开始不稳定, 但本文方法仍能在降低过程中保持稳定, 最终保持在每秒 30 帧左右. 同时, 在场景规模较小时, 如小于场景 5 的规模时, 虽然所有方法都能流畅运行 (即 FPS 大于 24), 但其他两种方法的

图 17 (网络版彩图) 场景漫游效果

Figure 17 (Color online) Snapshots from our prototype system

FPS 明显高于本文方法, 再次验证了本文方法的装载策略比其他两种方法更适用于处理大规模 BIM 场景. 此外, 其他两种方法中只是对场景进行显示, 不需要考虑漫游时频繁的碰撞检测等计算过程, 这也是导致本文方法的 FPS 在小规模场景中低于其他两种方法的原因之一. 且经实验性验证, 使用本文方法后所有的实验场景均能在手机网页下较流畅漫游, 部分场景的漫游效果见图 17.

9 结论及展望

本文从大规模 BIM 数据包含多方面的建筑信息、冗余度高等特点出发, 提出一套面向手机网页的大规模 WebBIM 场景轻量级实时漫游算法. 首先在解析 BIM 场景的基础上, 结合语义内容分析对重复性构件和拉伸实体进行冗余去除处理, 然后再将整个虚拟场景分解成构筑物, 外体、内体空间等多粒度数据, 降低了场景数据量并统一了数据结构; 其次, 在现有方法的启发下, 室内外分别构建不同的索引结构, 且通过树型结构的父子关系进行层级关联, 建立整个场景的混合索引结构 FLHS, 方便在后续进行室内外场景数据的快速可视性剔除. 同时, 从实验的角度, 设计一系列简单有效的渐进式装载策略, 将前期处理的结果进行集成性耦合, 最后通过仿真实验进行验证, 整个实验表明这些方法能大幅度地降低 BIM 大规模场景解析后的冗余数据, 能进行高效的室内外场景剔除, 也能及时地进行数据访问, 整个方案的提出为大规模 WebBIM 场景的手机网页轻量级漫游起到了较好的推动作用.

当然, 本文对手机平台下大规模 WebBIM 场景的轻量化还处于初步阶段, 仍有很多方面要去探索和完善. 如外体空间抽取算法只适用于遮挡关系较强的构筑物, 否则抽取的结果会不准确; 内体分析时与建筑空间的复杂程度有一定关系; 系统对网络环境也存在依赖性, 但鉴于篇幅, 在此暂未讨论该部分内容. 另外本文更侧重于对 IFC 格式下, 大场景中冗余数据的去除和渐进式加载等, 还未考虑场景的纹理贴图以及场景加载后的各种高级渲染等功能, 后续将继续考虑对这些方面进行分析和改进.

最后, 由于目前 BIM 场景方面的公开数据集还较少, 希望收集更多的实验数据来完善现有的公开数据集或直接创建新的数据集等.

参考文献

- 1 Donkers S, Ledoux H, Zhao J, et al. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Trans GIS*, 2016, 20: 547–569
- 2 Thomas L. IFC Office Release 4. Model Support Group (MSG) of BuildingSMART, 2013. <http://www.buildingsmart-tech.org/specifications/ifc-releases/ifc4-release>
- 3 Johansson M, Roupé M, Bosch-Sijtsema P. Real-time visualization of building information models (BIM). *Automat Constr*, 2015, 54: 69–82
- 4 Pasewaldt S, Trapp M, Döllner J. Multi-perspective detail overview visualization for 3D building exploration. In: *Proceedings of the 11th Theory and Practice of Computer Graphics (TPCG)*, 2013. 57–64
- 5 Karsch K, Golparvar-Fard M, Forsyth D. ConstructAide: analyzing and visualizing construction sites through photographs and building models. *ACM Trans Graph*, 2014, 33: 176
- 6 Sun J, Liu Y S, Gao G, et al. IFCCompressor: a content-based compression algorithm for optimizing industry foundation classes files. *Automat Constr*, 2015, 50: 1–15
- 7 Arthaud G, Lombardo J C. Automatic semantic comparison of STEP product models. In: *Innovations in Design & Decision Support Systems in Architecture and Urban Planning*. Berlin: Springer, 2006. 447–463
- 8 Lee G, Won J, Ham S, et al. Metrics for quantifying the similarities and differences between IFC files. *J Comput Civil Eng*, 2011, 25: 172–181
- 9 Zhang L, Issa R R A. Development of IFC-based construction industry ontology for information retrieval from IFC models. In: *Proceedings of the EG-ICE Workshop*. The Netherlands: University of Twente, 2011. 6–8
- 10 Gao G, Liu Y S, Wang M, et al. A query expansion method for retrieving online BIM resources based on industry foundation classes. *Automat Constr*, 2015, 56: 14–25
- 11 Liu H, Liu Y S, Pauwels P, et al. Enhanced explicit semantic analysis for product model retrieval in construction industry. *IEEE Trans Ind Inf*, 2017, 13: 3361–3369
- 12 Gao G, Liu Y S, Lin P, et al. BIMTag: concept-based automatic semantic annotation of online BIM product resources. *Adv Eng Inf*, 2017, 31: 48–61
- 13 Shi X, Liu Y S, Gao G, et al. IFCdiff: a content-based automatic comparison approach for IFC files. *Automa Constr*, 2018, 86: 53–68
- 14 Guo J, Yan D M, Li E, et al. Illustrating the disassembly of 3D models. *Comput Graph*, 2013, 37: 574–581
- 15 Laga H, Mortara M, Spagnuolo M. Geometry and context for semantic correspondences and functionality recognition in man-made 3d shapes. *ACM Trans Graph*, 2013, 32: 150
- 16 Zheng Y, Cohen-Or D, Averkiou M, et al. Recurring part arrangements in shape collections. *Comput Graph Forum*, 2014, 33: 115–124
- 17 Cohen-Or D, Chrysanthou Y L, Silva C T, et al. A survey of visibility for walkthrough applications. *IEEE Trans Visual Comput Graph*, 2003, 9: 412–431
- 18 Mattausch O, Bittner J, Wimmer M. CHC++: coherent hierarchical culling revisited. *Comput Graph Forum*, 2008, 27: 221–230
- 19 Jia J Y, Wang W, Wang M F, et al. Multi-layered incremental & scalable sector of interest (MISSOI) based efficient progressive transmission of large-scale DVE scenes. *Chinese J Comput*, 2014, 37: 1324–1334 [贾金原, 王伟, 王明飞, 等. 基于多层增量式可扩展扇形兴趣区域的大规模 DVE 场景对等渐进式传输机制. *计算机学报*, 2014, 37: 1324–1334]
- 20 Gu W, Wang J, Shi H, et al. Research on a hybrid spatial index structure. *J Comput Inf Syst*, 2011, 7: 3972–3978
- 21 Terry J, Stantic B. Indexing method for multidimensional vector data. *Comput Sci Inf Syst*, 2013, 10: 1077–1104
- 22 Varduhn V, Mundani R P, Rank E. Real time processing of large data sets from built infrastructure. *J Syst Cybernet Inf*, 2011, 9: 63–67
- 23 Kang H, Lee G. Development of an object-relational IFC server. In: *Proceedings of ICCEM/ICCPM*, Jeju, 2009
- 24 Liu X J, Xie N, Tang K, et al. Lightweighting for Web3D visualization of large-scale BIM scenes in real-time. *Graph Model*, 2016, 88: 40–56
- 25 Volk R, Stengel J, Schultmann F. Building information modeling (BIM) for existing buildings — literature review and future needs. *Automat Constr*, 2014, 38: 109–127
- 26 Van Kaick O, Zhang H, Hamarneh G, et al. A survey on shape correspondence. *Comput Graph Forum*, 2011, 30:

- 1681–1707
- 27 Kong Y, Dong W, Mei X, et al. SimLocator: robust locator of similar objects in images. *Visual Comput.* 2013, 29: 861–870
- 28 Zhu C, Yi R, Lira W, et al. Deformation-driven shape correspondence via shape recognition. *ACM Trans Graph*, 2017, 36: 51
- 29 Bouaziz S, Tagliasacchi A, Pauly M. Sparse iterative closest point. *Comput Graph Forum*, 2013, 32: 113–123
- 30 Lin Y H, Liu Y S, Gao G, et al. The IFC-based path planning for 3D indoor spaces. *Adv Eng Inf*, 2013, 27: 189–205
- 31 Amor R, Dimyadi J. An open repository of IFC data models and analyses to support interoperability deployment. In: *Proceedings of the 27th Annual International CIB W78 Conference*, Cairo, 2010

Mobile web-based lightweight and real-time roaming algorithm for large-scale WebBIM scenes

Xiaojun LIU^{1,2} & Jinyuan JIA^{1*}

1. *School of Software Engineering, Tongji University, Shanghai 201804, China;*

2. *Mathematical and Information Engineering Department of Nanhu College, Jiaxing University, Jiaxing 314001, China*

* Corresponding author. E-mail: jyjia@tongji.edu.cn

Abstract With the popularity of mobile terminals, including smart phones, there is a great push for users to share large-scale data on phone webpages. This study considers building information modeling (BIM) as the core smart construction technology and leverages, its classical data format — industry foundation classes (IFC) as the input data format, and presents a mobile & web-based lightweight and real-time roaming algorithm for large-scale WebBIM scenes. To comply with the semantics information in IFC files and with a building's structure, our method focuses on creating a more lightweight product, by not relying exclusively on geometry faces as ordinary visualization programs do. Our main idea for optimizing mobile, web-based, lightweight and real-time roaming is threefold: reduce redundancies, generate an efficient hybrid data structure and design dynamic load/unload strategies. In the study, a prototype system was integrated and leveraged for experimentation. The final performance analysis shows that we can significantly reduce the redundancy to about 70 percent of the raw BIM scenes. Meanwhile, our method can maintain the FPS at around 30 and memory consumption at about 400 MB with an increasing scene scale.

Keywords mobile web3D, BIM/IFC, lightweight, building space analysis, index structure

Xiaojun LIU was born in 1981. He is currently a Ph.D candidate at the Department of Software Engineering in the School of Software Engineering of Tongji University. His research interests include 3D modeling and geometry computing. E-mail: xjliu1204@126.com.

Jinyuan JIA was born in 1963. He is currently a professor at the Department of Software Engineering in the School of Software Engineering of Tongji University. His research interests include virtual reality and web graphics. E-mail: jyjia@tongji.edu.cn.