软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(9):3485−3497 [doi: 10.13328/j.cnki.jos.006293]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于六边形的自适应层次网格及其应用于点在球面多边形内的判断*

李 静[1], 王文成[2,3]

[1](中国科学院 动物研究所 动物进化与系统学院重点实验室, 北京 100101)

[2](计算机科学国家重点实验室 (中国科学院 软件研究所), 北京 100190)

[3](中国科学院大学, 北京 100049)

通信作者: 李静, E-mail: jingli2018@ioz.ac.cn

摘 要: 点在球面多边形内的判定计算, 在全球数据处理中有大量的需求. 为此, 提出一种基于六边形的自适应层次网格, 克服了已有六边形层次网格难以自适应划分组织的不足, 并应用于点在球面多边形内的判定. 首先, 基于正二十面体对球面进行均匀划分, 形成初始网格. 然后, 根据球面多边形的边与六边形网格的相交情况, 对六边形网格单元进行自适应的细分处理, 形成层次化的网格, 使得各个没有细分的单元不包含或仅包含少量的多边形的边, 并预计算这样单元本身或其中心点位于多边形内/外的属性. 在此, 记录相邻层次的六边形网格之间关联的点边面的拓扑结构, 由此可快速地从初始网格检索到没有细分的网格单元. 对于一个测试点, 检索到其所在的没有细分的六边形单元, 再依据该单元关于多边形的局部情况即可判定该测试点是否位于多边形内. 实验表明, 所提方法较以往方法具有更稳定而高效的判定性能.

关键词: 六边形网格; 球面; 自适应; 点在球面多边形内判定

中图法分类号: TP391

中文引用格式: 李静, 王文成. 基于六边形的自适应层次网格及其应用于点在球面多边形内的判断. 软件学报, 2022, 33(9): 3485–3497. http://www.jos.org.cn/1000-9825/6293.htm

英文引用格式: Li J, Wang WC. Adaptive Hexagonal Hierarchical Grid for Point-in-spherical-polygon Tests. Ruan Jian Xue Bao/Journal of Software, 2022, 33(9): 3485–3497 (in Chinese). http://www.jos.org.cn/1000-9825/6293.htm

Adaptive Hexagonal Hierarchical Grid for Point-in-spherical-polygon Tests

LI Jing[1], WANG Wen-Cheng[2,3]

[1](CAS Key Laboratory of Zoological Systematics and Evolution, Institute of Zoology, Chinese Academy of Sciences, Beijing 100101, China)

[2](State Key Laboratory of Computer Science (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

[3](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Point-in-spherical-polygon tests are highly required in global data processing. For this reason, this study proposes an adaptive hexagonal hierarchical grid, which overcomes the difficulty of existing hexagonal hierarchical grids in adaptively subdividing grid cells, and applies it to point-in-spherical-polygon tests. First, the initial spherical hexagonal grid is built by uniformly partitioning a sphere using a regular icosahedron. Then, hierarchical grids are constructed by adaptively subdividing hexagonal cells according to whether a grid contains many polygon edges. As a result, the cells not subdivided contain no or only a few edges, called leaf cells. Finally, pre-computing is performed to determine the location attributes (inside/outside the polygon) of such cells or their center points. In the hierarchical structures, the topologies of related points, edges and faces between adjacent hexagonal grid levels are recorded, by which the leaf cells can be quickly located. For a test point, the leaf cell containing it is found quickly, and then whether it is located in the polygon

* 基金项目: 国家自然科学基金 (61872348, 62072446, 31961143002)
收稿时间: 2020-09-02; 修改时间: 2020-11-10; 采用时间: 2020-12-07; jos 在线出版时间: 2022-07-15

is determined according to the local situation of the cell. Experimental results show that the proposed method has more stable and efficient performance than the existing methods.

Key words: hexagonal grid; spherical; adaptive; point-in-spherical-polygon test

1 简 介

全球数据处理以虚拟球体表达作为参考模型对地理空间数据进行组织、分析和可视化. 它在空间探索、移动通讯、网络监控、环境监测、天气预报等诸多领域中应用广泛, 其高效计算日益重要. 点在球面多边形内的判定是全球数据处理中的基础操作, 其效率的高低对于上层应用的性能有很大影响. 实际应用中, 球面多边形通常具有较多的边, 而相关的被测点也很多. 这些因素再叠加上球面本身非欧氏空间的性质, 使得点在球面多边形内的高效判定是一个极具挑战性的问题.

尽管平面上的点在多边形内的判定已有很多研究, 但它们不能直接在球面的非欧氏空间使用. 虽然可以通过投影方法将球面多边形转换到平面, 进而使用平面上的检测算法, 但投影会使多边形的边产生形变, 继而导致判定错误[1]. 球面 ray-crossing 方法[2]将平面上的经典 ray-crossing 方法[3]移植到球面空间, 以球面弧段代替直线段进行求交计算, 能够避免因投影导致的问题. 但它每次判定都需要检测多边形所有的边, 因而速度很慢. 平面上基于空间划分的加速策略, 如均匀网格[4,5], 也可用于球面的处理, 它们首先定位被测点所在单元, 然后将判定操作局限在被定位的单元及其附近几个单元内, 可大幅提高判定速度, 但由于球面为非欧空间, 其表面的划分与组织比平面上的相关处理更加复杂, 计算开销不小.

经纬度网格和六边形网格是目前使用最为广泛的两种球面空间划分结构. 经纬度网格依据经度和纬度的数值范围的均匀划分来形成网格. 其创建简单、便于基于经纬度的数值进行快速的网格单元的检索, 很早便在地理信息等领域得到广泛使用[6]. 但这种网格结构的单元大小随纬度变化, 特别是在极点处会出现聚集为一点的奇异情况. 这严重影响多边形的边在网格单元内的分布, 使得一些本应细分的单元不再细分, 会包含过多的边, 因而影响判定效率.

六边形网格是近年来得到广泛关注和使用的球面划分结构[7−9], 因为各个正六边形网格单元对应的球面区域是大小一致的, 可有效避免经纬度网格的那种单元大小不均衡对局部化操作的影响. 目前球面六边形层次网格的生成方法均首先创建初始正多面体, 随后在该正多面体的各个平面片上建立平面六边形层次网格, 最后将平面六边形层次网格投影到球面得到球面六边形层次网格. 这类方法均需在每层相对于整个球面进行细分, 因为细分的下一层网格中的六边形单元, 有的单元所覆盖的球面区域属于上一层网格中多个六边形单元的球面覆盖区域. 这使得层次化六边形网格的管理比较复杂, 分辨率较大的层次将包含大量的单元. 因此, 在实现中通常并不存储球面单元顶点的实际几何坐标, 而是对各个单元进行编码, 根据单元的编码即时计算其顶点坐标[10]. 但由于计算过程中的投影操作通常会涉及三角函数计算, 因此速度较慢, 严重影响判定速度.

本文提出一种新的基于六边形的自适应层次网格组织方式, 简称 AMH (adaptive multi-level hexagons), 它记录单元顶点坐标位置, 并且对于相邻层次的六边形单元, 记录它们之间相关联的点、边、面拓扑结构, 由此实现层次结构中由粗到细的检索. 如此, 我们无需在平面片上建立层次结构再投影回球面, 无需细分各个层次上的所有六边形网格, 也无需全局性的编码处理, 可自适应地优化局部处理, 提升处理效率. 具体地, 我们先基于正二十面体创建初始全球六边形网格, 随后只对包含较多多边形的边的网格单元进行局部的迭代细分, 形成层次化网格, 使得各个没有细分的单元 (称为叶子单元) 不包含或只包含少量的边. 于是, 对于一个测试点, 先基于我们的层次结构快速找到包含它的叶子单元, 再根据该单元相关的多边形局部情况, 即可判断该测试点是否位于多边形内. 为提高非空叶子单元内的局部计算的效率, 我们依据文献 [1] 的工作, 先预计算各个非空叶子单元的中心点是否位于多边形内的属性, 然后在检测时, 将测试点与其所在的叶子单元的中心点进行连线, 统计该线与该单元内的多边形的边的相交情况, 即可完成检测处理. 图 1 为本文方法的工作流图. 图中, 蓝色实线为球面多边形, 灰色粗实线和灰色细实线分别为第 1 层和第 2 层的网格边框. 绿色和红色单元分别为位于多边形内和多边形外的单元. 黑色单元为非空

单元, 它们位于多边形内和多边形外的中心点分别以绿色和红色点表示. Q_1 和 Q_2 为 2 个测试点, 其属性分别通过其所落入的叶子单元本身的属性和单元中心点属性获得.

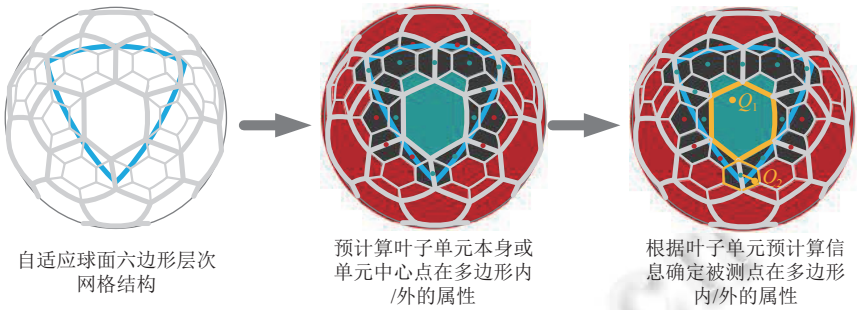


图 1 本文方法的工作流程图

得益于六边形网格的均衡性, 我们基于 AMH 的方法可稳定地处理球面上各处的计算, 并由于 AMH 可局部自适应地优化处理, 节省了大量的全局性计算, 我们方法可很好地提高计算效率. 实验结果显示, 相比于目前最快的基于层次经纬度网格的判定算法[1], 基于 AMH 的判定方法能够对具有不同特征的多边形保持稳定的判定效率, 特别是对于具有较长边以及靠近极点的多边形, 其效率提升更加明显. 而相比于基于目前业界流行的六边形层次网格索引方法 H3[11]的点在球面多边形内的近似判定实现, 本文方法的判定效率可提高 3.0 倍, 并且为精确判定.

本文第 2 节简述相关工作. 之后在第 3 节声明一些必要的定义和约定. 在第 4 节和第 5 节分别详述基于六边形的自适应层次网格和基于它的点在球面多边形内的判定方法. 在第 6 节的实验结果与分析之后, 在第 7 节进行总结.

2 相关工作

2.1 经纬度网格

经纬度球面网格利用经线和纬线在球面上进行分割构成网格. 它分为等角经纬度网格和等积经纬度网格两种[12]. 等角经纬度网格对经纬度按照固定的间隔进行划分. 其优点是便于实现, 因而被广泛采用[13]. 其缺点是网格单元的面积和形状随纬度变化, 这不但会降低多分辨率数据操作的准确性和处理效率, 而且还会导致高纬度地区的数据冗余. 为此, 人们提出使用等积经纬度网格, 在同一层次上使用面积近似相等的单元[14,15]. 但这种结构难以进行层次化组织, 近邻搜索复杂. 在工程应用中, Google 的 Google Earth[16], NASA 的 World Wind[17], 微软的 Virtual Earth[18]均采用经纬度球面网格, 但这些应用更侧重于可视化, 对空间数据的组织与分析支持不多.

2.2 六边形球面网格

六边形球面网格是一种重要的球面划分结构, 是数字地球[19]的核心结构——离散全球网格系统 (discrete global grid system, DGGS) 的一种重要类型. 全球六边形网格将球面离散化为无缝无叠的多分辨率六边形网格层次结构, 球面信息被指派到其位置对应的网格单元内, 借助于网格结构的层次性和均匀性, 实现球面信息的高效组织、存取与处理. 其创建过程遵循 DGGS 的通用创建过程, 即首先创建拟合球体的初始多面体, 然后将初始多面体的平面片进行细分形成多分辨率的平面单元, 再通过投影将平面单元映射到球面形成球面单元. 此外, 还需要设置单元的索引机制以便于实现对单元的查询访问. 目前已有多种全球六边形网格结构, 其区别主要在于多面体初始化、细分方法和单元索引方法等方面. 六边形层次网格的初始多面体通常为正八面体[20−22]或正二十面体[23,24]. 其中, 正二十面体在对球体近似表达时的角度和面积的形变更小. 在细分方法方面, 依据[25]的定义, 面积为 A 的单元被细分为面积为 A/k 的一组单元被称为 1-to-k 细分, k 为细分孔径. 六边形单元的细分孔径可为 3[21,24,26], 4[22,27]或 7[10](图 2, 其中父单元与子单元分别用蓝色和橙色边框表示), 以及基于它们的混合孔径[28]. 六边形单元不存在叠合细分, 即一个层次的两个或多个单元会与下一细分层次的一个单元有共享的球面区域, 这给上、

下层之间的搜索处理带来了困难. 孔径为 3 和 7 的细分都会使相邻两层网格的方向发生旋转[29], 而孔径为 4 的细分则可以保证所有层次的网格方向一致, 可简化层次分析[30], 有利于单元定位操作. 在单元索引方面, 通常是为每个单元进行编码并以其作为单元的唯一标识. 编码方法可分为基于层次的[24,31,32]、基于空间填充曲线的[33]和基于坐标的[34,35]这 3 种类型. H3 是美国运输网络公司——优步公司 (Uber Technologies Inc.) 的全球离散网格系统所采用的索引方法[10], 应用广泛. H3 针对基于正二十面体的孔径为 7 的 16 层全球六边形层次网格结构进行编码, 采用基于层次的编码方式, 每个编码长度为 8 个字节.

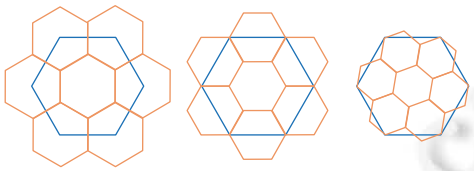


图 2　六边形单元孔径 3, 4 和 7 的细分 (从左至右)

对于基于编码的索引方法, 单元的编码包含了单元所在网格层次以及在该层次中的逻辑位置信息, 因而可以通过编码计算出单元在球面的位置. 但根据 DGGS 的创建过程, 这一计算不但需要平面片上的多种坐标转换, 而且需要将多面体平面片上的局部坐标通过某种投影方法 (通常为保面积的施耐德投影[36]) 投影到球面, 计算量很大, 严重影响单元的快速定位. 预计算并存储所有单元顶点坐标可以缓解这一问题, 但目前的结构均对每层网格进行完全细分, 高分辨率层次上的单元数量很多 (例如, H3 最高分辨率层次包含 569 T 个单元), 存储开销巨大.

本文提出的 AMH 结构也基于正二十面体创建, 且采用基于孔径 4 的细分, 但为了支持快速球面单元定位, 其结构与现有的六边形层次结构有如下 3 点不同: (1) 根据多边形的边自适应对单元进行迭代细分, 即仅细分包含较多多边形边的单元; (2) 不在多面体的平面片上进行迭代细分, 而是直接对球面进行迭代细分, 即逐层确定并存储球面单元顶点坐标, 依据上层球面单元顶点, 直接生成下层球面单元顶点. 这样不但可以避免三角函数计算, 而且可以保证同层单元大小基本一致, 不影响单元内边数的分布; (3) 不对单元进行编码, 而是存储同层及相邻层单元间的拓扑连接关系, 以此实现快速单元查找.

2.3　点在球面多边形内的判定

点在球面多边形内的判定问题是球面信息处理中的基本操作. 相对于平面上的点在多边形内检测算法, 点在球面多边形内的判定问题研究得较少. 已有的方法可分为两类, 即转换到平面空间进行处理和直接在球面空间处理. 第 1 类方法通过某种投影方法将球面多边形投影到平面上, 再使用平面上的高效算法进行判定. 由于投影和平面判定方法均可采用已有的成熟算法, 这类方法易于实现. 但球面弧线投影到平面变为直线, 会有形变而可能导致判定错误[1]. 第 2 类方法直接在球面空间计算, 如 Bevis 等人[2]在球面空间直接进行 ray-crossing 计算. 其过程是找到一个位于球面多边形内的已知点 X, 在球面上连接被测点 Q 与 X, 计算与 QX 相交的球面多边形的边的个数. 如果相交数为偶数则 Q 位于球面多边形内部, 否则位于外部. 这种方法不但需要检测球面多边形所有的边, 而且在判定两条球面弧线段是否相交时需要大量的三角函数计算, 因此速度很慢, 难以满足实际应用需求. 2017 年的一个工作[1]提出建立层次化的球面经纬度网格结构, 预计算叶子单元中心点位于多边形内/外的属性, 在判定时将被测点与其所在叶子单元中心点连线并使用局部 ray-crossing 方法来实现. 由于利用了局部化操作, 并在弧线求交的基本操作上避免使用三角函数计算, 该方法的速度很快. 但经纬度网格单元大小随纬度变化, 使得单元内所含边数分布不均, 从而导致某些区域缺乏细分, 使得一些叶子单元的相关计算量上升, 增加了判定开销. 在本文, 我们将基于六边形的自适应层次网格结构与[1]提出的球面网格中心点判定方法相结合, 具有与目前最快算法相若的判定效率, 但提高了对不同类型多边形的判定稳定性.

3　约定和定义

在详述算法前, 我们首先明确一些本文涉及的定义和约定. 在球面几何中, 大圆是指将球体分为两个相等半球

的圆. 若球面上的两点 P 和 Q 不是对极点 (在球面上相反的两点), 则它们可确定唯一一个大圆. 该大圆被点 P 和 Q 划分为两个弧段, 较长的弧段被称为主弧段, 较短的弧段被称为次弧段. 本文中的弧均指大圆的次弧段. 若一个弧段给定起始顶点和终止顶点, 则称该弧段为从起始点到终止点的有向弧段. 我们定义本文研究的球体为球心位于直角坐标系的原点处且半径为 1 的单位球. 球面网格的所有顶点均位于球体表面. 所有网格单元的边均为连接单元顶点的有向弧段. 从球体外部观察一个单元时, 单元的边以逆时针顺序首尾相连. 按此序遍历单元的边时, 左侧为单元的内侧, 右侧为单元的外侧. 两个同层的单元通过一对重叠且方向相反的边相邻接, 这一对边互为对边. 我们定义球面多边形 S 由 N 个球面上的点以及依次连接这些点的弧构成. 这些弧被称为球面多边形的边. 在后续描述中我们简称球面多边形为多边形. 从球体外部观察多边形, 当多边形的边以逆时针顺序被遍历时, 定义边左侧为多边形的内侧, 右侧为多边形的外侧. 点在球面多边形内的判定是指给定球面上的一点 Q 和球面多边形 S, 判定 Q 是否位于 S 内.

4 自适应细分的六边形层次网格

首先, 我们借助正二十面体生成首层六边形网格. 然后从首层网格开始, 依据球面多边形的边与六边形网格单元的相交情况进行网格单元的自适应细分, 即对包含较多多边形的边的单元进行细分、对包含较少多边形的边或不包含多边形的边的单元不进行细分; 细分过程迭代进行, 直至每个没有细分的单元均只含少量的多边形的边或不含多边形的边. 下面分别进行详细说明.

4.1 首层六边形网格

首层六边形网格基于正二十面体构建. 其过程是首先创建一个中心位于球心 (即直角坐标系原点) 的任意大小的正二十面体[37], 随后依据正二十面体的顶点生成首层网格的顶点, 然后按照给定规则连接这些顶点生成五边形或六边形单元, 最后设置边之间的拓扑关系. 下面分别进行说明.

对于正二十面体的每个三角面, 我们首先在其上取 4 个点, 分别为三角面的重心以及该重心与三角面 3 个顶点连线的中点. 我们称这些点为首层网格单元顶点对应的平面顶点. 这些平面顶点的球面投影点即为该三角面对应的首层网格单元顶点. 这里采用的投影方法是球心投影, 即做从球心开始经过平面点的射线, 该射线与球面的交点即为投影点. 由于我们设定球体为球心位于直角坐标系原点的单位球, 因此只要将这 4 个平面顶点的坐标进行归一化即可得到投影点坐标.

在得到首层网格单元的顶点之后, 我们按照如下步骤生成首层网格单元: (1) 对于正二十面体的每个顶点, 找到围绕它的 5 个最近平面顶点, 逆时针连接这 5 个平面顶点对应的首层网格单元顶点, 形成一个球面五边形单元. (2) 对于正二十面体的每条边, 找到位于其两侧的各 3 个总共 6 个最近平面顶点, 逆时针连接这 6 个平面顶点对应的首层网格单元顶点, 形成一个六边形单元. 在拓扑关系的实现上, 我们规定网格单元的边均为单向边; 每个单元记录构成它的单向边序列闭环; 具有相邻关系的单元通过一组对边相邻接. 图 3 展示了首层六边形网格在正二十面体平面片上的对应顶点及单元 ((a) 是平面顶点, (b) 是绿色填充的平面五边形单元, (c) 是紫色填充的平面六边形单元).

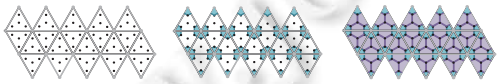


图 3 首层六边形网格在正二十面体面片上的对应平面顶点及单元

4.2 相邻层次之间的关联

在生成首层球面六边形网格之后, 我们将其作为当前层次网格, 通过自适应迭代细分建立相邻层次间的关联, 完成多层六边形网格的创建. 自适应细分采用受限的孔径为 4 的细分结构, 存储同层和相邻层单元间的拓扑连接关系以实现单元的检索定位, 并直接在球面上实现细分, 具体如下.

受限的孔径为 4 的细分结构在孔径为 4 的细分结构基础上限制子单元的范围, 以减少单元顶点数目. 在这种结构中, 一个被细分单元 (父单元) 的子单元包括一个中心子单元和若干个边界子单元. 中心子单元位于父单元内部, 且与父单元形状一致. 边界子单元则分别与父单元的一条边相关联, 且根据邻接单元是否为空单元而具有不同的形状: 若邻接单元为非空单元, 则该边对应的边界子单元为横跨该边的六边形单元, 即一半位于父单元内部一半位于邻接单元内部; 若邻接单元为空单元或无相邻单元, 则对应的边界子单元仅为位于父单元内的一半六边形单元 (即四边形单元). 图 1 展示了一个基于球面三角形创建的两层自适应细分结构.

由此可见, 自适应层次网格结构中存在 3 种类型的单元, 即六边形单元、五边形单元和四边形单元, 并且相邻层网格单元间存在错杂对应的关系. 为了能对这种复杂结构实现快速的单元检索, 我们存储同层及相邻层网格单元间的拓扑连接关系, 即每个单元由单向边序列的闭环构成; 相同层次的两个具有相邻关系的单元通过一对对边相邻接; 父单元仅存储中心子单元; 父单元的每条边存储该边对应的边界子单元.

与 DGGS 的细分过程不同, 我们不在初始多面体表面进行细分, 而是直接在球面上迭代细分. 也就是对于被细分的每一层网格, 我们首先为其所有需要细分的非空单元创建中心子单元, 之后再为所有需要细分的非空单元创建边界子单元, 下面分别进行说明.

中心子单元的创建过程是首先确定中心子单元的顶点, 然后依次逆时针连接这些顶点. 中心子单元顶点的确定方法是取父单元中心点 (若是四边形单元则取最长边的中点), 将该中心点与父单元的各个顶点以直线连接, 取这些直线段的中点, 再以球心投影方式投影到单位球表面即可. 图 4(a)–(c) 展示了六边形、五边形和四边形 3 种类型单元的中心子单元 (以紫色表示) 的情况.

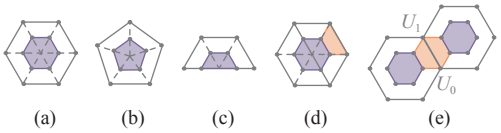


图 4　不同类型的子单元

边界子单元的创建过程是遍历每个非空单元的每条边, 若当前边已存在对应的边界子单元, 则不做处理; 否则, 按照本节第 2 段所述的规则生成边界子单元. 无论何种类型的边界子单元, 它们的顶点均为中心子单元顶点或父单元顶点. 新生成的边界子单元被记录为当前边的子单元. 若当前边存在对边, 则也将这个边界子单元记录为对边的子单元, 也就是说, 一对对边共享一个边界子单元. 此外, 需要注意的是四边形单元的中心子单元与父单元的一条边相邻接, 则这条边不参与边界子单元的生成, 即没有与之相关的边界子单元. 图 4(d) 展示了一个四边形边界子单元 (以橙色表示). 图 4(e) 则展示了一个六边形的边界子单元 (以橙色表示), 它同时属于一对对边 U_0U_1 和 U_1U_0. 图 5 为一个边界子单元的生成示例, 其中, (a) 为需要被依次处理生成边界子单元的ⓐ、ⓑ 和ⓒ 3 个单元; (b) 为处理单元ⓐ 时生成的边界子单元, 以橙色单元表示; (c) 为处理单元ⓑ 时生成的边界子单元, 以绿色单元表示; (d) 为处理单元ⓒ 时生成的边界子单元, 以红色单元表示.

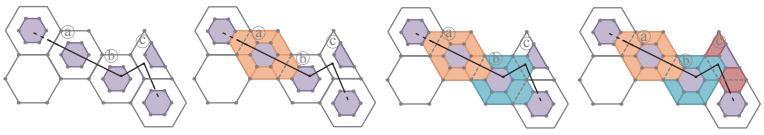


图 5　边界子单元的创建过程示例

当满足截止条件时, 迭代细分过程将结束. 截止条件通常被设置为网格已达到最大层次数. 最大层次数可被简单的设置为固定值, 也可根据多边形进行动态调整以获得更佳的性能. 例如将其设置为使得最大层次上的单元大小与多边形边的平均长度相当时所需的层次数. 除了设置最大深度, 还可同时设置单元内包含的最多边数, 即当非空单元内包含的边数小于此值时, 该单元可提前结束细分. 该参数的设定可根据应用进行相应的调整. 在我们的实

现中, 自动确定最大深度并设单元内最多边数为 5.

4.3 基本操作

自适应层次网格的建立过程主要涉及两个基本操作: 判定点是否位于网格单元内和判定多边形的边是否与网格单元的边相交. 这两个操作均可通过直角坐标空间内的计算实现. 由于避免了球面空间操作中耗时的三角函数运算, 因此速度很快.

第 1 个基本操作可概括为设 P_0 为给定点, A 为一给定单元, 现判定 P_0 是否位于 A 内. 判定方法是逆时针遍历单元 A 的所有边, 若点 P_0 位于当前边与球心构成平面的右侧, 则点 P_0 必然位于单元 A 的外侧, 可以不再继续检测, 提前结束判定. 而若 P_0 位于平面左侧, 则以同样的方法检测下一条边. 当单元 A 的所有边都通过检测, 则说明点 P_0 位于单元 A 内.

第 2 个基本操作可概括为设 P_0P_1 为一条多边形的边, U_0U_1 为网格单元的一条边, 现判定 P_0P_1 是否与 U_0U_1 相交. 由定义可知, 多边形的边与网格单元的边均为球面大圆的一部分, 因此判定多边形的边是否与网格单元的边相交的问题等同于判定两条球面弧是否相交. 为此, 我们采用文献 [1] 中的方法进行判定, 也就是若 P_0 和 P_1 位于 U_0U_1 与球心构成平面的两侧, 并且 U_0 和 U_1 也位于 P_0P_1 与球心构成平面的两侧, 则多边形的边与单元的边相交. 图 6 为上述两种基本操作的示意图, 其中, O 为球心, 弧段 P_0P_1 与弧段 U_0U_1 相交于球面上一点 W. 点 P_0、P_1 和 O 构成的平面和点 U_0、U_1 和 O 构成的平面的分别以橙色和蓝色表示. 这两个平面相交于直线 WO.

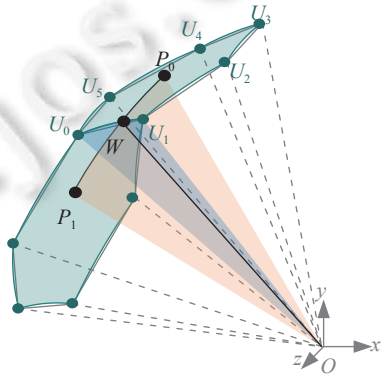


图 6 判定点是否位于网格单元内以及判定多边形的边是否与网格单元的边相交的示意图

5 点在球面多边形内的快速判定

在 AMH 结构创建完成后, 还需要预计算其叶子单元的信息, 随后可以实现点在球面多边形内的快速判定, 即先为被测点定位其所在的叶子单元, 再根据叶子单元的预计算信息确定被测点属性. 前者为寻找局部化操作区域的过程, 后者则为执行局部化操作的过程. 下面分别进行说明.

5.1 预计算叶子单元信息

为提高计算效率, 我们提出一种自底向上逐层预计算叶子单元信息的方法, 它自最底层 (即最精细层) 网格开始到首层 (即最粗略层) 网格为止, 自下而上地逐层对叶子单元进行处理. 对于每层网格分别处理其非空叶子单元和空叶子单元. 对于非空叶子单元借助单元内的边判定其中心点的内外属性. 对于空叶子单元则使用 floodfill 方法 [38]在同层及相邻层中相邻的空单元间以传播方式确定单元本身的内外属性, 由此可去除大量冗余的判定操作, 有效提高效率. 具体讲就是我们根据非空叶子单元中心点的属性, 可判断其相邻的空叶子单元位于多边形内/外的属性, 然后以其为种子单元, 迭代地将所有与其相邻的同层网格的空单元均设置为与其相同的属性, 重复该过程, 直到当前层网格中所有的空单元均被设置为止. 在这一过程中, 当前层的空单元可能与属于上层网格的空单元相邻接, 它们具有与这些当前层空单元相同的属性. 我们将这些属于上层网格的空单元作为种子单元记录下来, 用于

自底向上逐层设置过程中后续相应层次上的空单元的 floodfill 计算. 因此, 对于当前层次, 种子单元按照来源可分为两类: 一类是根据本层非空单元得到属性的空单元; 另一类是通过下层网格的空单元传导得到属性的空单元, 它们在处理下层网格的时候被记录下来, 并在处理当前层网格时被使用. 算法 1 描述了这一过程.

算法 1. 预计算 AMH 叶子单元信息.

```
1 setAMHLeafInfo(grids[0...maxdepth – 1]:AMH 结构)
2 begin
3   upseedlists[0..maxdepth – 1][]: 由下层网格传导得到的种子单元数组
4   for(i=maxdepth – 1;i≥0;i – –)
5       grid←grids[i]
6       for(j=0;j<grid 的单元总数;j++)
7           cell←grid.cells[j]
8           If(cell 为非空叶子单元) then
9               判定 cell 的中心点属性
10              seeds←与 cell 相邻的同层次上未设置属性的空叶子单元集合
11              for(k=0;k<seeds 中的单元数;k++)
12                  依据 cell 的中心点属性判定 seeds[k] 本身的属性
13                  floodfill(seeds[k], upseedlists)
14      for(j=0;j<upseedlists[i] 中的单元数;j++)
15          floodfill(upseedlist[i][j], upseedlists)
16 end
17 floodfill(startseed, upseedlists[0...maxdepth – 1][])
18 begin
19   cells←与 startseed 相邻的同层次上未设置属性的空叶子单元
20   for(i=0;i<cells 中的单元数;i++)
21       将 cells[i] 的属性设置为 startseed 的属性
22       upseeds←与 cells[i] 相邻的属于上层网格的未设置属性的单元集合
23       for(j=0;j<upseeds 中的单元数;j++)
24         d=upseeds[j] 的层次
25         将 upseeds[j] 添加到 upseedlists[d] 中
26      floodfill(cells[i], upseedlists)
27 end
```

在确定非空单元中心点属性时 (算法 1 第 9 行), 由于六边形网格单元交错排列, 因此不便于使用类似经纬度网格中心点算法[1]中的通过传递临近点属性来快速确定单元中心点属性的方法. 在此, 我们借助位于单元内的边来完成这一任务, 即第 1 步查找当前单元内离中心点最近的边; 第 2 步通过中心点与该边的相对位置来判定中心点的内外属性. 在第 1 步中需要注意 3 点. (1) 在计算中心点到多边形边的距离时, 由于多边形的边为弧段不便于计算, 我们计算中心点到该边两个顶点间直线段的距离. (2) 当垂足不在当前边对应直线段上时, 我们需要取该边距离中心点最近的顶点, 并将其与中心点的直线距离作为中心点到该边的最近距离. (3) 若对于两条相连的多边形的边, 它们距离中心点最近的点均为这两条边的公共顶点, 则需要进一步确定哪一条边离中心点更近. 在此, 我们比较该公共顶点与中心点的连线与这两条边分别构成的夹角 (取小于 180 度的角), 对应夹角更小的边则是更近一些的边. 在第 2 步中, 我们沿着最近边的方向, 判定中心点位于最近边与球心构成的平面的哪一侧. 若为左侧, 则中心点位于多边形内; 若为右侧, 则位于多边形外; 若位于该平面上, 即该边穿过中心点, 则标记其为特殊中心点, 以

便在判定时进行不同的处理. 在图 7 中, 对于非空单元ⓐ, 由于 e_k 为距离 C_a 最近的边且 C_a 位于边 e_k 与球心构成平面的左侧, 因此的中心点 C_a 位于多边形内.

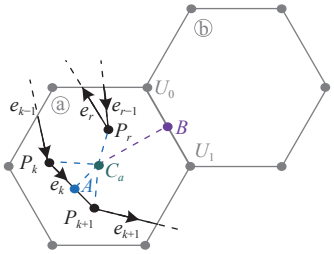


图 7 判定非空单元中心点属性及判定与非空单元邻接的空单元属性示例

在设置种子单元属性时 (算法 1 第 12 行), 我们可以借助非空单元的中心点使用局部 ray-crossing 方法来加速判定, 即取非空单元与空单元邻接边的中点, 计算该点与非空单元中心点连接弧段与多边形边的相交数. 若为偶数, 则空单元的属性与该中心点属性相同; 否则, 与之相异. 这里, 如果中心点为特殊中心点, 则我们仍借助点与最近边的相对位置关系进行判定. 在图 7 中, 对于空单元ⓑ, 由于邻接边 U_0U_1 的中点 B 与 C_a 的连线与单元ⓐ 内的多边形的边无交, 因此 B 位于多边形内, 故单元ⓑ 也位于多边形内.

5.2 点在球面多边形内的判定

点在球面多边形内的判定分为两步: 定位被测点所在的叶子单元和根据该单元的预计算信息确定被测点属性.

定位包含被测点的叶子单元从首层六边形网格开始, 即依次遍历首层网格单元, 直到找到包含被测点的单元为止. 由于首层网格覆盖整个球面, 因此一定可以找到一个包含被测点的单元. 从该单元开始自上而下地逐层确定点所在的细分单元, 即依次检测当前单元的中心子单元和边界子单元, 寻找包含被测点的子单元. 由细分过程可知, 父单元被其子单元完全覆盖, 因此一定可以找一个包含被测点的子单元. 将该子单元作为当前单元, 重复上述操作直到当前单元为叶子单元为止.

找到叶子单元后, 若该叶子单元为空单元, 则该单元的属性即为被测点的属性; 若其为非空单元, 则根据该单元的中心点使用局部 ray-crossing 得到被测点属性. 也就是连接被测点与叶子单元中心点, 计算该弧段与单元内多边形边的交点数, 若为偶数则被测点与中心点具有相同的属性, 否则具有相异的属性. 若中心点为特殊中心点, 则通过被测点与单元内最近边的相对位置关系进行判定.

在图 8 中, Q_0 落入位于多边形内的空叶子单元ⓐ 中 (以绿色表示), 因此 Q_0 也在多边形内. Q_1 落入非空单元中 (以白色表示), 其中心点 C_a 位于多边形外 (红色点) 且 Q_1C_a 与多边形的边无交, 因此 Q_1 也位于多边形外. Q_2, 所在的叶子单元的中心点 C_b 为特殊中心点 (蓝色点), e 为距离 C_b 最近的多边形的边. 由于 Q_2 位于 e 与球心所成平面的左侧, 故它位于多边形内.

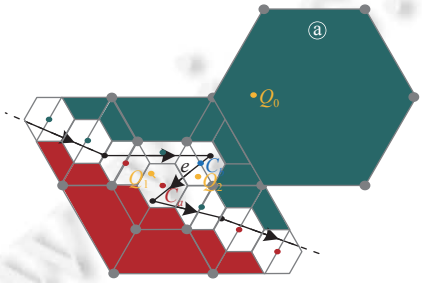


图 8 判定点是否位于多边形内过程的示例

5.3 复杂度分析

如前所述, 层次结构的最大细分层次数以自适应方式确定, 即最大层次上的单元大小与多边形边的平均长度

相当, 因此最大层次上的单元总数为 O(N), N 为多边形边数. 由于细分采用固定细分分辨率, 若不考虑极端情况, 则最大层次数为 O(log$N$). 对于预处理, 由于每层均需要处理 O(N) 条边且每层的单元数不超过 O(N), 总共 O(log$N$) 层, 因此复杂度为 O(Nlog$N$). 判定被测点的操作包括寻找叶子单元和依据叶子单元进行判定. 前者的时间复杂度为 O(log$N$). 对于后者, 若叶子单元为空单元, 则复杂度为 O(1); 若为非空单元, 叶子单元内的判定时间复杂度也为 O(1)[1], 因此判定时间复杂度为 O(log$N$). 空间开销主要用于存储网格单元和单元内的边指针, 两者的复杂度均为 O(N), 因此空间复杂度是 O(N). 综上, 本文方法的判定时间, 预处理时间和空间的复杂度分别为 O(log$N$), O(Nlog$N$) 和 O(N).

6 实验结果和分析

我们用 C++实现了本文提出的 AMH 结构以及基于它的点在球面多边形内的判定方法 (AMH-PISP), 并在配置有 2.1 GHz CPU, 16 GB RAM 的台式机上进行了实验以验证其性能. 实验分为两组, 下面分别进行说明.

在第 1 组实验中, 我们选取 3 个具有不同特征的多边形用于实验. 如表 1 所示, P0 为不包含空洞的具有密集刺状边的多边形; P1 为围绕极点的多边形, 其边长相对较长; P2 为包含 484 个空洞的复杂多边形, 其边长相对较短. 我们为每个多边形在其最小经纬度包围盒内生成 1 000 000 个随机均匀分布的被测点. 我们将 AMH-PISP 与其他 3 种判定算法进行了对比. 第 1 种为球面 ray-crossing 算法 (SRC)[2]. 它是目前普遍使用的非投影类判定方法. 由于文献 [2] 中的弧段求交过于缓慢, 我们将其替换为第 4.3 节中的方法. 第 2 种为基于层次经纬度网格中心点的方法 (LL-PISP)[1]. 它是目前速度最快的精确判定算法. 第 3 种为一种基于现有六边形 DGGS 系统索引方法——H3[10]的点在多边形内的判定实现 (H3-PISP). 它首先预计算所有位于多边形内的单元, 然后在判定时定位被测点所在单元并以该单元属性作为被测点属性. 由于六边形单元不能完全拟合多边形的形状, 因此该实现为近似判定. 此外, 由于 AMH 和 H3 的相同层次的单元大小不同, 为了对比公平, 我们为 H3-PISP 寻找其单元大小与 AMH 最大层次的单元大小基本一致的层次, 并将该层次作为 H3 的细分层次. H3-PISP 的预处理和点的单元定位操作分别由 H3 开源代码中的 polyfill 和 geoToH3 函数实现. 表 2 为统计结果. 其中, T_p 为预处理时间, T_q 为判定 1 000 000 个点的总时间, T_i 为 1 000 000 个点的单元定位总时间, 即将点的定位到最精细单元的总时间, 它是 T_q 的一部分, S 为辅助结构的空间开销, L 为网格最大层次数. T_p, T_i 和 T_q 的单位为秒, S 的单位为 KB.

表 1 用于第 1 组实验的 3 个球面多边形

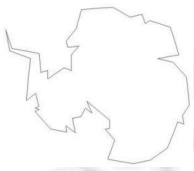
名称	P0	P1	P2
形状			
边数	1 000	54	2 927

表 2 本文方法与其他方法的对比实验结果

多边形	SRC	LL-PISP					H3-PISP					AMH-PISP				
	T_q	T_p	S	L	T_i	T_q	T_p	S	L	T_i	T_q	T_p	S	L	T_i	T_q
P0	355.1	0.033	307	4	0.063	1.21	0.11	3	3	0.77	0.95	0.097	274	5	0.30	0.66
P1	18.6	0.000 5	10	4	0.027	0.93	N/A	N/A	3	N/A	N/A	0.002 7	41	4	0.15	0.27
P2	1 052.9	0.046	246	4	0.055	0.46	22 384	12 815	7	1.29	1.79	0.27	2116	10	0.34	0.45

由表 2 可知, SRC 的判定速度大幅低于其他方法. 这是因为 SRC 方法需要检测多边形所有边, 而其他方法借助于球面划分结构仅需要检测很少的边, 因此速度快很多. AMH-PISP 的判定速度高于或与 LL-PISP 持平. 但对于

不同多边形, 速度提高的幅度差异较大. 其原因之一是由于 LL-PISP 的单元定位操作在整个判定过程中时间占比很小 (T_l/T_q≤12%), 因此其判定效率取决于单元中所包含多边形边的数量. 而由于经纬度网格单元的大小变化较大, 这使得每个单元内包含的多边形边数差异较大, 当以单元内包含的边数为参考进行细分时, 容易造成欠细分, 使得很多被测点落入非空叶子单元内, 从而使求交计算量上升继而影响判定效率. 这点在多边形的平均边长较长, 如 P0, 和在多边形靠近极点时, 如 P1, 更为明显. 而当多边形的平均边长相对较短时, 则能充分细分单元, 大多数被测点落入无需求交计算的空叶子单元内, 从而加快了判定速度, 如 P2. 由此可见, 得益于六边形网格的均匀特性, AMH-PISP 较 LL-PISP 具有更佳的稳定性. 对于 H3-PISP, 除 P1 因 polyfill 函数运行结果错误而没有数据外, 其他两个多边形的判定速度均低于本文方法. 由数据可知, H3-PISP 的超过 70% 的判定时间被用于单元定位, 因此 H3-PISP 的判定效率取决于单元定位的效率. 而 AMH-PISP 的单元定位时间仅为 H3-PISP 的 26%–38%. 这使得 AMH-PISP 具有更高的判定效率. 此外, H3-PISP 为近似计算而 AMH-PISP 是精确计算.

在所需空间方面, 由于 LL-PISP 和 H3-PISP 不存储网格单元顶点, 因此 AMH-PISP 相对需要更多的空间, 但其开销在可控的范围内 (见第 2 组实验). 由于 H3-PISP 需要存储单元索引, 因此其空间开销与多边形内的单元数密切相关. P0 的 L 仅为 3, 因此位于其内的单元较少, 空间开销很少. 但在此细分程度下六边形单元不能很好地拟合多边形, 误判较多. 与精确判定结果对比显示 H3-PISP 对 P0 的判定正确率仅为 76%. 增加 L, 可以提高正确率, 但空间开销会迅速增加. 比如 P2 在 L=7 时正确率为 99%, 但其空间开销大幅超过 AMH-PISP, 如表 2 所示.

在预处理速度方面, 由于 LL-PISP 能够方便的以传递方式确定单元中心点属性, 因此速度快于 AMH-PISP. 而对于 H3-PISP, 由于其判定单元是否位于多边形内的操作不但使用 ray-crossing 算法而且需要投影到多面体平面片上计算, 因此速度很慢. 对于边较多的 P2, 其预处理时间是 AMH-PISP 的 82 903 倍.

在第 2 组实验中我们对 AMH-PISP 的时空复杂度进行了验证. 我们生成了 12 个形状类似且边数从 100 变化到 102 400 的球面多边形. 被测点的生成方法与数量与第一组实验相同. 图 9 展示了其中 4 个被测多边形以及预处理时间 (包括创建 AMH 结构和预计算单元信息)、AMH 占用空间和判定所有被测点的总时间随边数的变化曲线. 由图 9 可见, 判定时间和空间开销的曲线显示出 O(log$N$) 和 O(N) 的趋势, 而预处理时间的曲线则不超过 O(Nlog$N$). 这与第 5.3 节中的分析结论是相符合的.

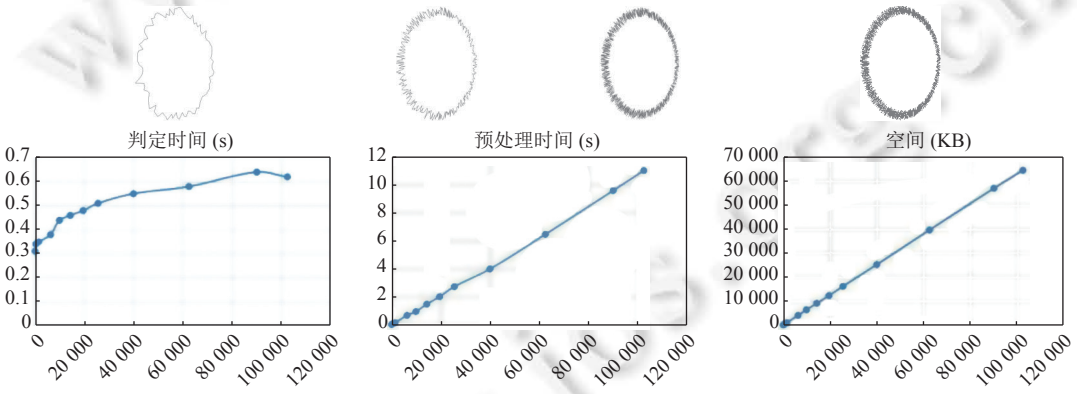


图 9 本文方法的判定时间、预处理时间和空间开销随多边形边数的变化曲线

7 结 语

本文提出了一种球面基于六边形的自适应层次网格的组织方式, 克服了已有六边形层次网格组织需要全局性处理的不足. 由此, 我们提出一种新的点在球面多边形内的快速判定方法. 该层次网格基于相邻层次的网格单元之间的拓扑关系进行上下层之间的搜索关联可快速找到叶子单元. 基于该层次网格结构, 我们对多边形的边进行管理, 使得各个叶子单元不包含或只包含少量的边. 这样, 检测计算时, 只需对测试点所在的叶子单元中的局部情况

进行处理即可. 实验表明, 本文方法比已有方法能更稳定而快速地判断测试点是否位于一个球面多边形内.

References:

[1] Li J, Zhang H, Wang WC. Fast and robust Point-in-Spherical-Polygon tests using multilevel spherical grids. In: Proc. of the 3rd Int'l Workshop on Next Generation Computer Animation Techniques. Bournemouth: Springer, 2017. 56–66. [doi: 10.1007/978-3-319-69487-0_5]

[2] Bevis M, Chatelain JL. Locating a point on a spherical surface relative to a spherical polygon of arbitrary shape. Mathematical Geology, 1989, 21(8): 811–828. [doi: 10.1007/BF00894449]

[3] Manber U. Introduction to Algorithms: A Creative Approach. Reading: Addison-Wesley, 1989. 266–270.

[4] Žalik B, Kolingerova I. A cell-based point-in-polygon algorithm suitable for large sets of points. Computers & Geosciences, 2001, 27(10): 1135–1145. [doi: 10.1016/S0098-3004(01)00037-1]

[5] Li J, Wang WC. Point-in-polygon tests by determining grid center points in advance. In: Proc. of the 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conf. IEEE, 2013. 1–7. [doi: 10.1109/APSIPA.2013.6694387]

[6] Brooks DR. Grid systems for earth radiation budget experiment applications. Technical memorandum, Hampton: National Aeronautics and Space Administration, 1981.

[7] Sahr K. Hexagonal discrete global grid systems for geospatial computing. Archives of Photogrammetry, Cartography and Remote Sensing, 2011, 6(22): 363–376.

[8] Xie JR, Yu HF, Ma KL. Interactive ray casting of geodesic grids. Computer Graphics Forum, 2013, 32(3): 481–490. [doi: 10.1111/cgf.12135]

[9] Xie JR, Yu HF, Ma KL. Visualizing large 3D geodesic grid data with massively distributed GPUs. In: Proc. of the 4th IEEE Symp. on Large Data Analysis and Visualization. Paris: IEEE, 2014. 3–10. [doi: 10.1109/LDAV.2014.7013198]

[10] Brodsky I. H3: Uber's hexagonal hierarchical spatial index. 2018. https://eng.uber.com/h3/

[11] H3: Hexagonal hierarchical geospatial indexing system. https://github.com/mayhemheroes/h3-1

[12] Wei C. Research on the representation of spatial data based on spherical hexagonal grid system [MS. Thesis]. Nanjing: Nanjing Normal University, 2008 (in Chinese with English abstract).

[13] Hastings DA, Dunbar PK. Development & assessment of the global land one-km base elevation digital elevation model (GLOBE). In: Proc. of the ISPRS Commission IV Symp. on GIS— Between Visions and Applications. Stuttgart: ISPRS, 1998. 218–221.

[14] Tobler W, Chen ZT. A quadtree for global information storage. Geographical Analysis, 1986, 18(4): 360–371. [doi: 10.1111/j.1538-4632.1986.tb00108.x]

[15] Bjørke JT, Grytten JK, Hæger M, Nilsen S. A global grid model based on "constant area" quadrilaterals. In: Proc. of the 9th Scandinavian Research Conf. on Geographical Information Science. Espoo: Helsinki University of Technology, 2003. 239–250.

[16] Google Inc. Google earth. https://google-earth.gosur.com/cn/

[17] NASA. WorldWind Java. http://worldwind.arc.nasa.gov/java/

[18] Schwartz J. Bing maps tile system. 2018. https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

[19] Mahdavi-Amiri A, Alderson T, Samavati F. A survey of digital earth. Computers & Graphics, 2015, 53: 95–117. [doi: 10.1016/j.cag.2015.08.005]

[20] White D. Global grids from recursive diamond subdivisions of the surface of an octahedron or icosahedron. Environmental Monitoring and Assessment, 2000, 64(1): 93–103. [doi: 10.1023/A:1006407023786]

[21] Vince A. Indexing the aperture 3 hexagonal discrete global grid. Journal of Visual Communication and Image Representation, 2006, 17(6): 1227–1236. [doi: 10.1016/j.jvcir.2006.04.003]

[22] Goodchild MF, Shiren Y. A hierarchical spatial data structure for global geographic information systems. CVGIP: Graphical Models and Image Processing, 1992, 54(1): 31–44. [doi: 10.1016/1049-9652(92)90032-S]

[23] Tong XC, Ben J, Zhang YS. The subdivision of global multi-resolution hexagonal grid and the rules of address coding. Acta Geodaetica et Cartographica Sinica, 2007, 36(4): 428–435 (in Chinese with English abstract). [doi: 10.3321/j.issn:1001-1595.2007.04.012]

[24] Sahr K. Location coding on icosahedral aperture 3 hexagon discrete global grids. Computers, Environment and Urban Systems, 2008, 32(3): 174–187. [doi: 10.1016/j.compenvurbsys.2007.11.005]

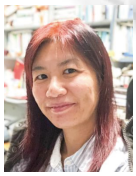
[25] Sahr K, White D, Kimerling JA. Geodesic discrete global grid systems. Cartography and Geographic Information Science, 2003, 30(2): 121–134. [doi: 10.1559/152304003100011090]

[26] Amiri AM, Alderson T, Samavati F. Geospatial data organization methods with emphasis on aperture-3 hexagonal discrete global grid systems. Cartographica: The Int'l Journal for Geographic Information and Geovisualization, 2019, 54(1): 30–50. [doi: 10.3138/cart.54.1.

2018-0010]
[27] Wang R, Ben J, Du LY, Zhou JB, Li ZX. Encoding and operation for the planar aperture 4 hexagon grid system. Acta Geodaetica et Cartographica Sinica, 2018, 47(7): 1018–1025 (in Chinese with English abstract). [doi: 10.11947/j.AGCS.2018.20170374]
[28] Wang R, Ben J, Zhou JB, Zheng MY. Indexing mixed aperture icosahedral hexagonal discrete global grid systems. ISPRS Int'l Journal of Geo-Information, 2020, 9(3): 171. [doi: 10.3390/ijgi9030171]
[29] Ivrissimtzis IP, Dodgson NA, Sabin MA. A generative classification of mesh refinement rules with lattice transformations. Computer Aided Geometric Design, 2004, 21(1): 99–109. [doi: 10.1016/j.cagd.2003.08.001]
[30] Thuburn J. A PV-based shallow-water model on a hexagonal-icosahedral grid. Monthly Weather Review, 1997, 125(9): 2328–2347. [doi: 10.1175/1520-0493(1997)125<2328:APBSWM>2.0.CO;2]
[31] Vince A, Zheng X. Arithmetic and Fourier transform for the PYXIS multi-resolution digital earth model. Int'l Journal of Digital Earth, 2009, 2(1): 59–79. [doi: 10.1080/17538940802657694]
[32] Tong XC, Ben J, Wang Y, Zhang YS, Pei T. Efficient encoding and spatial operation scheme for aperture 4 hexagonal discrete global grid system. Int'l Journal of Geographical Information Science, 2013, 27(5): 898–921. [doi: 10.1080/13658816.2012.725474]
[33] Uher V, Gajdoš P, Snášel V, Lai YC, Radecký M. Hierarchical hexagonal clustering and indexing. Symmetry, 2019, 11(6): 731. [doi: 10.3390/sym11060731]
[34] Patel A. Red blob games—Hexagonal grids. http://www.redblobgames.com/grids/hexagons/
[35] Mahdavi-Amiri A, Harrison E, Samavati F. Hexagonal connectivity maps for digital earth. Int'l Journal of Digital Earth, 2015, 8(9): 750–769. [doi: 10.1080/17538947.2014.927597]
[36] Harrison EE. Equal area spherical subdivision [MS. Thesis]. Calgary: University of Calgary, 2012.
[37] Regular icosahedron. https://www.simpleplanes.com/a/04ZPeo/Regular-Icosahedron
[38] Sun JG. Computer Graphics. 3rd ed., Beijing: Tsinghua University Press, 1998. 185–187 (in Chinese).

附中文参考文献:

[12] 韦程. 基于球面六边形网格系统的空间数据表达研究[硕士学位论文]. 南京: 南京师范大学, 2008.
[23] 童晓冲, 贲进, 张永生. 全球多分辨率六边形网格剖分及地址编码规则. 测绘学报, 2007, 36(4): 428–435. [doi: 10.3321/j.issn:1001-1595.2007.04.012]
[27] 王蕊, 贲进, 杜灵瑀, 周建彬, 李祝鑫. 平面四孔六边形格网系统编码运算. 测绘学报, 2018, 47(7): 1018–1025. [doi: 10.11947/j.AGCS.2018.20170374]
[38] 孙家广. 计算机图形学. 第3版, 北京: 清华大学出版社, 1998. 185–187.



李静(1972－), 女, 博士, 工程师, CCF 高级会员, 主要研究领域为计算机图形学, 计算几何.

王文成(1967－), 男, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为计算机图形学, 虚拟现实, 图像编辑.