DOI: 10.16452/j.cnki.sdkjzk.2025.02.006 文章编号:1672-3767(2025)02-0055-10

基于 CUDA 的多波束测深数据海底地形模型并行构建算法

李展鹏[1],崔晓东[1],云天宇[1],李晓勇[3],亓 超[1],阳凡林[1,2]
(1.山东科技大学 测绘与空间信息学院,山东 青岛 266590;
2.自然资源部海洋测绘重点实验室,山东 青岛 266590;
3.中国人民解放军 91001 部队,北京 100071)

摘 要:多波束测深系统(MBES)具有探测范围广、测量效率高的特点,面向海量多波束测深数据的高效快速地形建模与表达成为亟待解决的技术难点。为此,本研究提出一种基于通用并行计算架构(CUDA)的自适应分辨率规则格网并行构建算法,首先对区域进行四叉树的构建,然后针对分割后的点云计算自适应最优分辨率,利用图形处理器(GPU)并行加速构建格网,最后通过光照阴影计算和高程渲染,构建海底地形模型。本研究以水下大规模多波束测深点云数据为研究对象,进行了 GPU 加速与中央处理单元(CPU)环境下构网任务的测试和对比分析。结果显示,基于 CPU 构网耗时 22.753 s,本算法构网耗时缩短至 9.023 s,加速 2.52 倍,实现了海底地形模型的快速构建。

关键词:多波束测深系统;四叉树;图形处理器;规则格网;自适应分辨率
中图分类号:P229 **文献标志码**:A

Parallel construction algorithm of seafloor topography model for multibeam bathymetric data based on CUDA

LI Zhanpeng[1], CUI Xiaodong[1], YUN Tianyu[1], LI Xiaoyong[3], QI Chao[1], YANG Fanlin[1,2]
(1. College of Geodesy and Geomatics, Shandong University of Science and Technology, Qingdao 266590, China;
2. Key Laboratory of Oceanic Surveying and Mapping, Ministry of Natural Resources of the People's Republic of China, Qingdao 266590, China;
3. Unit 91001 of the Chinese People's Liberation Army, Beijing 100071, China)

Abstract: Multibeam echo sounder system(MBES) is characterized by wide detection range and high measurement efficiency, and the efficient and fast terrain modeling and expression for massive multibeam echo sounder data have become a technical difficulty to be solved. To this end, this study proposed a parallel construction algorithm of adaptive resolution regular grid based on common parallel computing architecture (CUDA). Firstly, a quadtree was constructed for the region. Then, the adaptive optimal resolution for the segmented point cloud was calculated and the grid using the parallel acceleration of the graphic processor (GPU) was constructed. Finally, the seafloor topography model was constructed through the calculation of light shading and the rendering of elevation. Taking

收稿日期:2024-10-04
基金项目:国家自然科学基金项目(52201400);中国博士后科学基金项目(2023M733686);山东省自然科学基金青年基金项目(ZR2022QD043);浙江省水利河口研究院院长科学基金项目(ZIHE21Y005)
作者简介:李展鹏(2000—),男,湖北荆州人,硕士研究生,主要从事海洋测绘数据管理与成图方面的研究.
E-mail:zpli0904@163.com
崔晓东(1992—),男,山东青岛人,副教授,博士,主要从事海洋测绘方面的研究,本文通信作者.
E-mail:cuixiaodong@sdust.edu.cn

underwater large-scale multibeam bathymetric point cloud data as the research object, this study conducted tests and comparative analyses of the grid construction task under GPU acceleration and central processing unit (CPU) environment. The results show that it takes 22.753 s to construct the network based on CPU, but the time taken by the proposed algorithm is shortened to 9.023 s, with an acceleration ratio of 2.52 times, achieving the rapid construction of the seabed terrain model.

Key words: multibeam echo sounder system (MBES); quadtree; graphics processing unit (GPU); regular grid; adaptive resolution

海底地形测量是海洋测绘的基础性任务,其获取的海底地形地貌信息是海洋开发活动的基本支撑资料[1]。由于多波束测深系统(multibeam echo sounder system,MBES)具有宽覆盖、高精度的优势,能够大幅提高海底地形的测量效率和质量,已成为水深测量任务中的主流探测设备[2-3]。基于当前主流声呐探测系统获取的海量数据,需要利用计算机资源快速构建海底地形模型并可视化后,方可精确表达复杂的海底地形特征,为海洋资源管理、环境监测和海上安全提供全面且可靠的支持。

目前,针对大规模海底地形点云数据,为有效地管理和处理海量的点云数据,便于数据分析和可视化,通常选择规则格网构建模型。吴焕萍等[4]介绍的基于分块索引的规则格网构建算法和基于扫描线填充的规则格网构建算法,两种算法在一定程度上提高了构网的效果与效率。Agarwal 等[5]提出一种可扩展的算法生成规则格网数字高程模型(digital elevation model,DEM),通过引入一种 I/O 效率算法提升构网的速度。这些算法在一定程度上能加快规则格网的构建速度,但对于大规模海底点云数据,仍需要更高效的算法来进一步提高构网效率。

早期,构网算法多基于中央处理单元(central processing unit,CPU)的串行算法,随着对构网速度需求的不断提高,通过并行处理技术进行加速成为必然。近年来,随着图形处理器(graphics processing unit,GPU)的出现和性能的不断提升,越来越多的计算密集型任务开始转向 GPU 平台并行加速。作为 NVIDIA 公司发布的并行计算编程模型,通用并行计算架构(compute unified device architecture,CUDA)可以有效地调度 GPU 并行计算资源,使得开发者可以将 GPU 的强大计算能力应用于图形处理之外的广泛领域。许多研究者利用 CUDA 平台来加速与优化计算密集型任务,彭敬等[6]实现了基于 CUDA 的伽马射线定位计算,处理 10 000 组数据时,对比基于单 CPU 的伽马射线定位计算,加速 364 倍。Rubio 等[7]引入一种单精度和双精度的混合算法,然后基于 CUDA 快速计算图形处理中的地势梯度,处理 3 200 个对象时,对比单 CPU 加速 645 倍。Schütz 等[8]采用 CUDA 实现了点云细节层次(level of detail,LOD)的实时生成和渲染,不考虑加载和渲染时,比非增量、基于 CPU 的网页版 Potree 加速 15.8～66 倍。郭大佑等[9]提出利用 CUDA 实现逆信噪比-复值退相关光学相干层析血流造影(optical coherence tomography angiography based on the inverse SNR and decorrelation features,ID-OCTA)的实时信号处理与图像显示。韩丰等[10]提出采用 CPU+GPU 混合架构设计并行雷达拼图算法来提高组网拼图算法的效率,在全国 1 km 和 500 m 分辨率的拼图任务上,分别加速 3.52 和 6.82 倍。熊超等[11]利用 GPU 处理航空 γ 能谱数据,确定航空 γ 能谱数据的最佳 block 尺寸在 64×64 到 128×128 之间,91%的小波基函数比 CPU 端加速 90 倍以上。李朝奎等[12]利用 GPU 加速计算进行图形和图像数据的快速可视化,在三角形面片较少时有一定的加速效果,三角形面片超过 40 000 后,加速比趋于平稳(50～60 倍)。由于海底地形点云数据规模达到 TB 级,传统的规则格网串行算法不能满足高效构网需求,而且当前基于 GPU 平台利用 CUDA 处理大规模海底点云数据建模过程的研究尚未开展。

因此,本研究给出一种基于 CUDA 的自适应分辨率规则格网并行构建算法,包括四叉树构建、格网的自适应分辨率计算和基于 CUDA 的快速构网三部分,实现针对海量声呐点云数据的规则格网快速构建。

1 串行算法及基于 CUDA 的并行化方案

海底地形点云数据规则格网构建串行算法主要包括以下步骤。

1) 格网分辨率选择。根据点云的密度和精度要求,选择适当的格网分辨率(即格网单元的大小),通常,分辨率越高,构网后的结果越精细,但计算量也会越大。

2) 确定格网范围。根据点云数据的空间分布,确定构建规则格网的空间范围。可以选择点云的最小外接矩形或立方体区域作为格网的覆盖范围。

3) 初始化格网。基于选择的格网分辨率,计算网格坐标系的起始点(通常是数据的最小坐标)和格网单元大小。

4) 填充格网。对于每个格网单元,统计该单元内的点数并计算单元内的水深平均值。用计算得到的水深平均值填充每个格网单元,如果格网内没有点,则将该格网单元值设置为空。

5) 精度验证。通过可视化工具或误差分析,验证构建的规则格网是否符合要求。

利用 CUDA 进行规则格网构建,首先要实现并行加速,加快构网速度。此外,由于 GPU 的存储空间有限,当数据量过大时,无法将整个数据集加载到 GPU 内存中,部分数据需要反复从 CPU 端传输到 GPU 端,造成频繁的内存交换和数据传输延迟,降低 GPU 的计算效率。此外,存储空间不足可能迫使 GPU 处理部分数据时进行分块或精简,这会影响计算精度,导致构建的网格模型不完整或不准确。

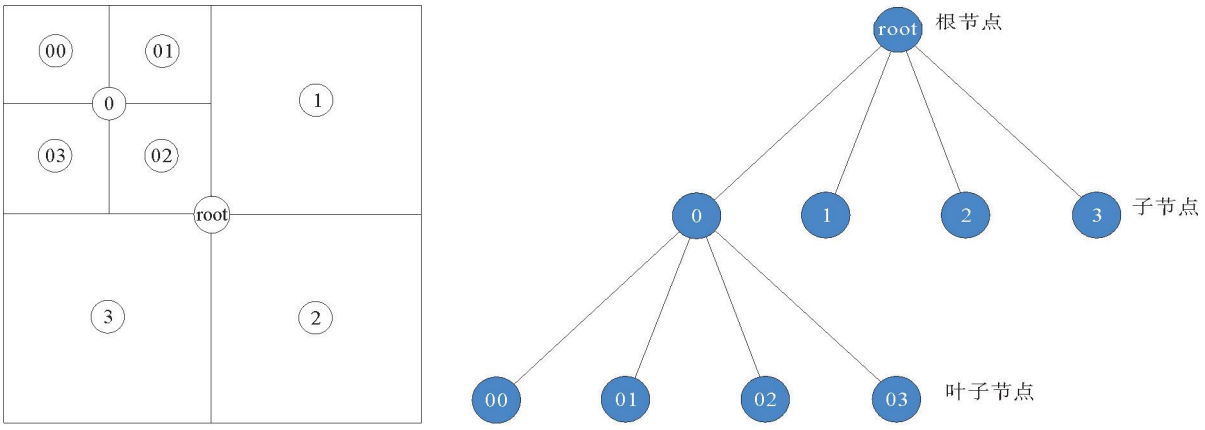
为了解决 GPU 存储受限问题,本研究借鉴分治法[13]的思想,构建四叉树,将数据划分成多个子区域,减少每次计算需要处理的数据量,减轻 GPU 内存负担,有效避免了内存溢出并提高了计算效率。此外,本研究基于四叉树设计一种自适应分辨率建模算法,该算法根据数据点密度和四叉树深度动态调整各区域分辨率,能够提高构网效率、减少数据冗余并保证构网精度,最后在 GPU 上实现规则格网并行构建。

2 基于 CUDA 的自适应分辨率规则格网并行构建算法

本节提出一种基于 CUDA 的自适应分辨率规则格网并行构建算法,通过四叉树管理海量海底点云数据,结合点密度与四叉树深度自适应调整格网分辨率,并利用 GPU 并行化实现格网初始化与填充,提高大规模海底地形构网效率,最后开发了一个可视化软件,实现大规模海底地形可视化。

2.1 四叉树构建过程

四叉树是一种用于海量空间数据管理的数据结构,常用于空间数据的分割和索引。可递归地将空间划分为四个子区域,将每个区域中的数据分配到相应的子节点,从而逐步形成一个层次化的树状结构,如图 1 所示。通过四叉树管理点云数据,可以实现空间数据高效插入和查询,因此常被应用于 LOD 动态调整分辨率。



(a) 四叉树空间结构示意图　　(b) 四叉树节点示意图

图 1　四叉树结构示意图

Fig. 1　Schematic illustration of the quadtree structure

对海量点云数据构建四叉树,具体过程如下:

1) 初始化根节点。在平面直角坐标系下,基于海底地形数据确定数据集在 x、y 方向上的最值,即 $X_{\min}$、$X_{\max}$、$Y_{\min}$、$Y_{\max}$。以 $[X_{\min}, Y_{\min}]$ 为原点,构建二维坐标系,根节点的包围盒大小为 $[X_{\min}, X_{\max}]$ 和 $[Y_{\min}, Y_{\max}]$ 的区域范围。

2) 划分节点。将根节点的区域划分为四个子节点,每个子节点的边界由根节点的中心点确定。子节点的边长是原区域的 $\frac{1}{2}$,根据当前节点中心点将区域分为四个象限。

3) 插入点云。对于每一个待插入的海底点云数据,根据其在 x、y 方向上的坐标值,确定其所在的子节点并将其插入该子节点中。

4) 分割条件。当一个节点中的点数超过了预定阈值且没有达到最大深度时,执行步骤 2)。

5) 递归调用。对于每一个点云数据点,递归执行步骤 3)和步骤 4),直到节点满足分割条件或达到最大深度为止。

6) 终止条件。当每个叶子节点中的点数小于预定阈值或达到最大深度时,四叉树构建完毕。

2.2 基于 CUDA 的大规模海底地形自适应分辨率建模

由于数据量较大,为了提高建模效率,本研究首先对海底地形点云数据构建四叉树,然后基于 CUDA 并行构建规则格网,格网示意图如图 2 所示。

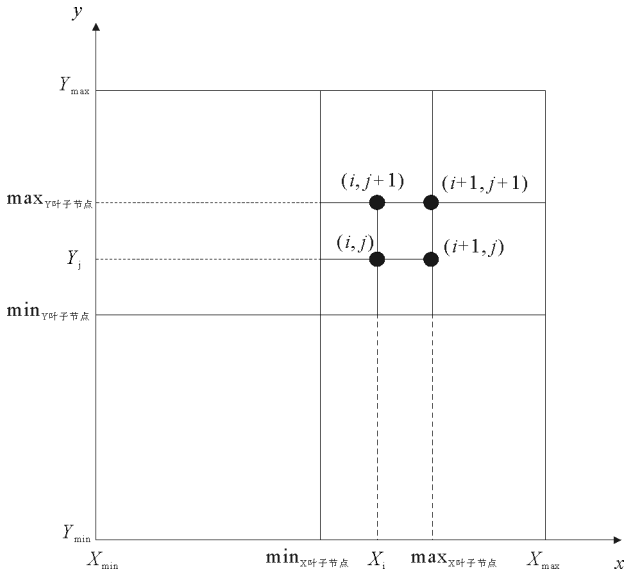


图 2 格网示意图

Fig. 2 Grid illustration

图 2 中,$\max_{X\text{叶子节点}}$、$\min_{X\text{叶子节点}}$、$\max_{Y\text{叶子节点}}$、$\min_{Y\text{叶子节点}}$ 是对应四叉树节点包围盒 x、y 方向的最值,X_i 和 Y_j 分别表示格网点 (i, j) 的 x 坐标和 y 坐标,其中 $i=1, 2, \cdots, \text{ceil}\left(0.5+\frac{\max_{X\text{叶子节点}}-\min_{X\text{叶子节点}}}{a_{\text{utostep}}}\right)$,$j=1,2,\cdots, \text{ceil}\left(0.5+\frac{\max_{Y\text{叶子节点}}-\min_{Y\text{叶子节点}}}{a_{\text{utostep}}}\right)$,ceil 是向上取整函数。

对应的叶子节点:

$$i=\frac{X_i-\min_{X\text{叶子节点}}}{a_{\text{utostep}}}, \tag{1}$$

$$j=\frac{Y_j-\min_{Y\text{叶子节点}}}{a_{\text{utostep}}}, \tag{2}$$

$$i_{\text{ndex}(i,j)}=j\times\left(\frac{\max_{X\text{叶子节点}}-\min_{X\text{叶子节点}}}{a_{\text{utostep}}}+1\right)+i。 \tag{3}$$

式中:a_{utostep} 是叶子节点分辨率大小,$i_{\text{ndex}(i,j)}$ 是(i,j)对应点索引。

基于 CUDA 的自适应分辨率规则格网并行构建算法具体步骤为:

1) 设置四叉树的节点阈值,遍历所有点云构建四叉树。

2) 计算各个叶子节点的点密度

$$\rho=\frac{C_{\text{loud}\,\text{叶子节点}}}{(\max_{X\text{叶子节点}}-\min_{X\text{叶子节点}})\times(\max_{Y\text{叶子节点}}-\min_{Y\text{叶子节点}})}。 \tag{4}$$

式中:$C_{\text{loud}\,\text{叶子节点}}$ 是对应四叉树节点的点数量。

考虑到四叉树包围盒以及四叉树节点中点云个数非常少时,计算得到的分辨率过小导致模型出现空洞现象。为此,在计算过程中跳过点云数量过少的节点,并将这类节点的分辨率设定为计算得到的最高分辨率。

3) 通过叶子节点的四叉树深度 L、点密度 ρ 确定叶子节点的分辨率

$$a_{\text{utostep}}=\frac{\rho}{L\times P}。\tag{5}$$

式中,P 是点云数据构建四叉树的叶子节点总数,计算得到的自适应分辨率与点密度呈正相关关系,与四叉树深度成负相关关系。

4) 根据各个叶子节点的最适分辨率初始化并填充格网:①遍历叶子节点中所有离散点,根据点坐标将其分配到相应的格网单元中;②对于每个格网单元,统计该单元内的点数并计算单元内的水深平均值,用计算得到的水深平均值填充每个格网单元,如果格网内没有点,则将该格网单元值设置为空;③遍历所有格网单元,跳过空值格网单元,通过式(3)构建格网。

根据各个叶子节点的分辨率构建规则格网时,由于 GPU 可以针对不同任务进行独立计算,因此在考虑 GPU 显存容量的前提下,动态的调度四叉树管理的点云数据进行规则格网构建:在 CPU 端定义并调用 CUDA 核函数完成格网初始化和格网填充,在调用核函数时通过设置线程块尺寸(blockDim)和格网尺寸(gridDim)来指定核函数在 GPU 上的执行配置,分配 GPU 的计算资源,确保充分利用 GPU 的并行计算能力。

5) 当四叉树的全部叶子节点均参与计算之后,将不同线程块处理的格网子块进行合并,完成对海底地形的建模。

该算法根据四叉树叶子节点的点密度和深度计算最适分辨率。在同一深度时,点云密度大的节点分辨率较高,点云密度小的节点分辨率较低。当点云密度相同时,四叉树深度越深,节点分辨率越小,从而将分辨率控制在合理范围内。通过创建 CUDA 核函数来初始化和填充格网,可以提高海底地形模型的构建效率。

基于 CUDA 的自适应分辨率规则格网并行构建的 CPU 端和 GPU 端流程如图 3 所示。

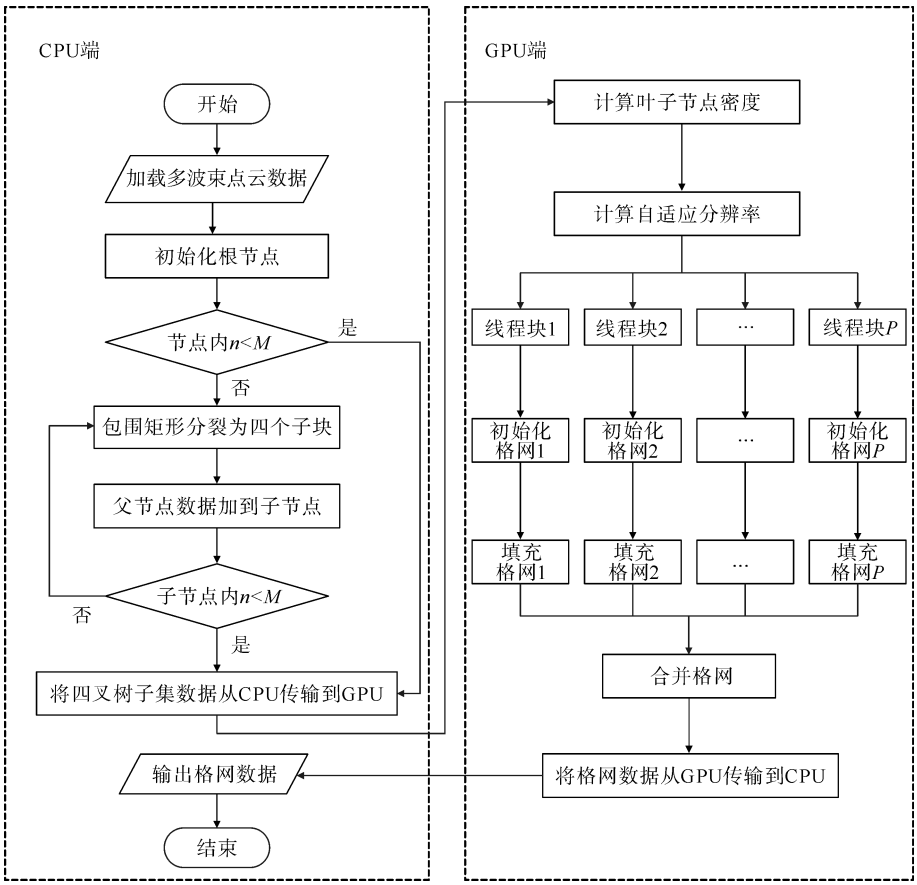


图 3 基于 CUDA 的自适应分辨率规则格网并行构建流程图

Fig. 3 Flowchart of CUDA-based parallel construction of adaptive resolution regular grid

基于 CUDA 的自适应分辨率规则格网并行构建伪代码如算法 1 所示。

算法 1　基于 CUDA 的自适应分辨率规则格网并行构建算法

```
输入：多波束测深点云数据 pointCloud
输出：规则格网数据 gridData
1) Begin
2) function CPU_Processing(pointCloud)  // CPU 端处理函数,接受点云数据作为输入
3)     rootNode = InitializeRootNode(pointCloud)  //使用点云数据初始化四叉树的根节点
4)     while rootNode.nodeCount < threshold N:  //如果根节点的节点数小于阈值 N,进行分裂
5)         childNodes = SplitNodeIntoQuads(rootNode)  //将当前节点分裂为四个子节点
6)     for childNode in childNodes:
7)         AddParentDataToChild(childNode, rootNode)  //将父节点的数据添加到子节点中
8)     for childNode in childNodes:
9)         if childNode.nodeCount < threshold N:  //如果子节点的数据量小于阈值
10)            ProcessNodeRecursively(childNode)  //递归处理该子节点
11)    quadTreeData = cudaMemcpy(rootNode, cudaMemcpyHostToDevice)  //将处理好的四叉树数据从 CPU 传输到 GPU
12) return quadTreeData  //返回四叉树数据
13) function GPU_Processing(quadTreeData):  // GPU 端处理函数,接受四叉树数据作为输入
14)    leafDensities = ComputeLeafNodeDensity(quadTreeData)  //计算每个叶子节点的数据密度
15)    resolutions = ComputeAdaptiveResolutions(leafDensities)  //根据叶子节点的密度计算适应的分辨率
16)    parallel for i in range(numBlocks):  //在多个线程中并行初始化每个格网
17)        grid[i] = InitializeGrid(resolutions[i])  //根据自适应分辨率初始化格网
18)        FillGrid(grid[i], quadTreeData[i])  //填充每个格网
19)    finalGrid = MergeGrids(grid)  //合并所有计算得到的格网数据
20)    finalGridData = cudaMemcpy(finalGrid, cudaMemcpyDeviceToHost)  //将计算完成的格网数据从 GPU 传回到 CPU
21) return finalGridData  //返回最终的格网数据
22) End
```

2.3　海底地形可视化软件开发

为了体现渲染的效果,本研究开发了一个可视化软件,为了增强地形模型的真实感,首先计算模型的光照阴影。此过程通过模拟光源对物体表面的影响来提升地形模型的真实感。将输入的光照天顶角和方位角转换为弧度,遍历具有有效水深值的格网单元,使用 3×3 格网单元计算 x 方向和 y 方向的梯度,得到地形的坡度和坡向,然后基于地形坡度与光源方向之间的角度余弦值计算光照阴影值,并将其限制在 0～1,其中 1 表示完全照明,0 表示完全阴影。

光照阴影值计算完成后,进行高程渲染:遍历所有格网点,通过水深最值将水深值映射到 0～1 之间,并根据映射区间转换为颜色值,然后将光照阴影值应用于颜色值,实现对地形的直观可视化。再基于 C++和 QT 平台实现海底地形模型的可视化。

3　实验与分析

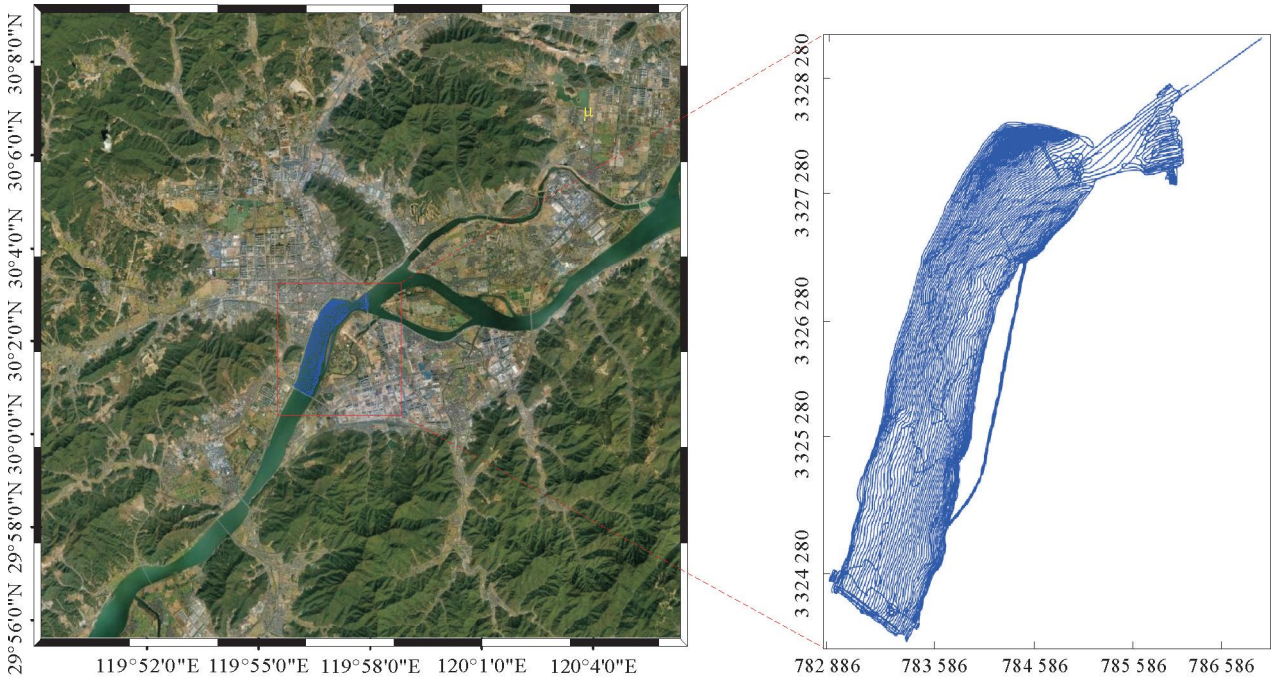
硬件平台配置和实验平台软件环境如表 1 所示。

为验证本算法对大规模海底地形快速建模的效率,实验选取的多波束数据由 R2Sonic2020 型号的多波束测深系统采集,于 2022 年 5 月 26 日完成外业测量,点云总数为 4.031 亿个。测区位于中国浙江省钱塘江上游富春江富阳段河道区域,测区长度为 4.019 km,宽度为 4.673 km,共布设 211 条测线,测线分布如图 4 所示。

表 1 实验平台软硬件配置

Table 1 Hardware and software configuration of experimental platform

	参数	配置
硬件环境	CPU	13th Gen Intel(R) Core(TM) i7-13700KF @ 3.40 GHz
	Architecture	x86_64
	CUDA Cores	9 728
	内存	64 GB
	GPU	NVIDIA GeForce RTX 4080
	总显存	16 GB
软件环境	Microsoft Visual Studio	2022
	CUDA	12.3



(a) 区域示意图 (b) 多波束测线示意图

图 4 实验区域示意图

Fig. 4 Schematic illustration of the experimental area

3.1 CPU 与 GPU 单线程下构网对比

为验证自适应分辨率并行构网算法的优势,对多波束测深数据,使用串行程序分别在 CPU 和 GPU 单线程下构建分辨率为 0.2 m 的规则格网,格网数量为 2.040 亿个,实验结果得到 CPU 端的构网耗时为 37.634 s,而 GPU 端则为 47.407 s。

实验结果表明,在单线程下,基于 GPU 的规则格网并行构建算法的构建速度低于基于 CPU 的规则格网串行构建算法。原因在于 GPU 的存储空间有限,在处理大规模数据时,无法将整个数据集一次性加载到 GPU 内存中,需要将数据反复从 CPU 端传输到 GPU 端,造成频繁的内存交换和数据传输延迟,从而导致 GPU 的构建速度较低。

3.2 基于四叉树的并行构网效率对比

为验证采用分治策略构网的优势,在 CPU 端对原始多波束数据构建四叉树,构树耗时 7.684 s。对

子集数据并行构建分辨率为 0.2 m 的规则格网,实验结果表明,基于四叉树的构网方法,在 CPU 端的构网耗时为 22.100 s,GPU 端为 2.815 s。整个规则格网并行构建算法总耗时,在 CPU 端为 29.784 s,在 GPU 端为 10.499 s。

结合 3.1 节的实验结果可以看出,基于 GPU 采用分治思想并行构建规则格网能显著加快构网效率,因为每次计算仅需处理局部数据集,减少对显存的直接压力,提高了算法的整体效率。虽然引入四叉树增加了构建过程中的时间开销(构建四叉树耗时为 7.684 s),但是整体的构网效率显著提升。

3.3 基于 CUDA 的自适应分辨率并行构网

本研究基于 CUDA 实现自适应分辨率并行构网,旨在提高建模效率并满足分辨率需求。利用本算法对四叉树管理的点云数据计算自适应分辨率,得到四叉树叶子节点的分辨率为 0.22～2.03 m,其中分辨率为 0.22～0.50 m、0.50～1.00 m、1.00～1.50 m、1.50～2.03m 的四叉树节点占比分别为 38.46%、34.62%、19.23%、7.69% 。

为验证本研究算法的构网效率和精度,实验设置 4 档分辨率(0.2、0.5、1.0、1.5 m)构建海底地形模型进行对比,选择平均绝对误差 M_{AE} 和均方根误差 R_{MSE} 两个评价指标评估构网精度。计算公式如下:

$$M_{AE}=\frac{\sum_{k=1}^{N}|H_k-h_k|}{N}, \tag{6}$$

$$R_{MSE}=\sqrt{\frac{\sum_{k=1}^{N}(H_k-h_k)^2}{N}}。 \tag{7}$$

式中:H_k 为第 k 个数据点的实际水深值,h_k 为第 k 个数据点的插值水深值。

比较表 2 数据可以发现,随着分辨率增大,构网时间逐渐减少,误差值增大。在最高分辨率(0.2 m)时,构网时间最长,但模型精度最高;在最低分辨率(1.5 m)时,构网时间最短,但模型精度最低,表明海底地形模型的构网时间与分辨率大小呈正相关性。低分辨率有更高的构网效率,但会降低模型的细致度,影响区域细节的表达;反之,较高的分辨率虽然能提供更多细节,但效率较低,并且更容易导致海底地形模型出现漏洞,影响模型的完整性。实验得到 CPU 端 0.5 m 固定分辨率格网的串行构网时间为 22.753 s,GPU 端 0.5 m 固定分辨率格网的并行构网时间为 1.458 s(见表 2)。本研究算法为自适应分辨率,分辨率范围在 0.22～2.03 m,其中 0.22～1.00 m 的节点占所有节点的绝大部分,本算法在 GPU 上的并行构网时间为 1.339 s,比 GPU 端 0.5 m 固定分辨率格网的并行构网时间(1.458 s)少。本并行算法的构网时间与 CPU 端 0.5 m 固定分辨率格网的串行构网时间(22.753 s)相比,如果仅考虑构网效率不考虑四叉树构建耗时,加速 16.99 倍;如果同时考虑构网效率和四叉树构建时间(7.684 s),则加速 2.52 倍。

表 2 基于 CUDA 的不同分辨率模型构网效率和精度对比

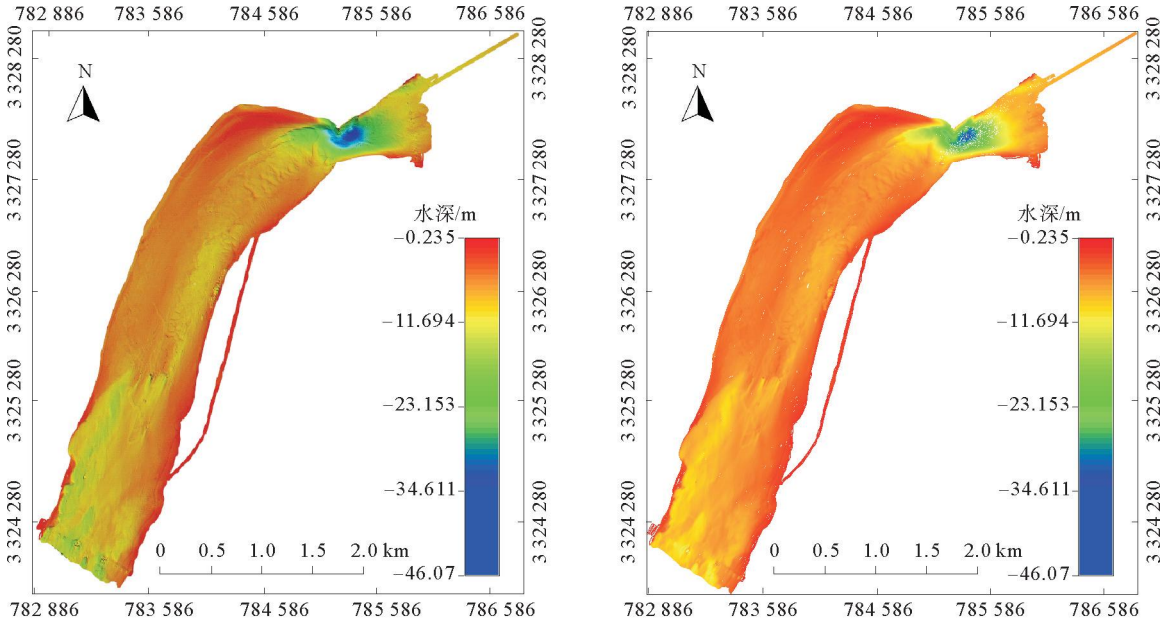
Table 2 Comparison of grid construction efficiency and accuracy for different resolutions based on CUDA

分辨率/m	格网数/亿个	构网耗时/s	总耗时/s	M_{AE}/m	R_{MSE}/m
0.2	2.040	2.815	10.499	0.112	0.204
0.5	0.330	1.458	9.142	0.245	0.315
1.0	0.084	1.096	8.780	0.422	0.484
1.5	0.038	0.770	8.454	0.609	0.672
自适应	0.383	1.339	9.023	0.343	0.405

3.4 海底地形模型可视化对比

为进一步验证本算法的构网效率和海底地形模型可视化软件的渲染效果,将自研的可视化软件与 Cloud Compare 软件构建的海底地形模型进行对比(图 5)。Cloud Compare 构建分辨率为 0.5 m 的海底地

形模型耗时 10.051 s,构网效率低于本算法,且从视觉角度观察,自研软件生成的海底地形模型在地形可视化上更为真实,且 Cloud Compare 构建的模型存在明显空洞,特别是在右上角的蓝色区域。



(a) 自研海底地形可视化软件　　(b) Cloud Compare

图 5　自研海底地形可视化软件与 Cloud Compare 光照效果对比图

Fig. 5　Comparison of lighting effects between self-developed seafloor terrain visualization software and Cloud Compare

综上所述,本研究算法能够提高大规模海底地形构网效率,且自研可视化软件生成的海底地形模型具有较好的可视化效果。

4　结论

针对多波束海底地形数据量大导致的构网效率低的问题,提出基于 CUDA 的自适应分辨率规则格网并行构建算法。实验结果表明:

1) 与基于单核 CPU 的串行构建算法相比,本算法耗时明显缩短,如果仅考虑构网效率不考虑四叉树构建,加速 16.99 倍。如果同时考虑四叉树构建时间,则加速 2.52 倍;

2) 通过光照阴影计算和高程渲染构建海底地形模型,实现的自研海底地形可视化软件比 Cloud Compare 软件构网效率略优,且在可视化效果上更具优势;

3) 通过合理设计四叉树深度和充分利用 GPU 并行计算资源,能显著提升多波束测深点云数据的处理效率。

后续可以利用 GPU 加速海底地形点云数据的四叉树构建过程。

参考文献:

[1] 赵建虎,欧阳永忠,王爱学. 海底地形测量技术现状及发展趋势[J]. 测绘学报,2017,46(10):1786-1794.
ZHAO Jianhu, OUYANG Yongzhong, WANG Aixue. Status and development tendency for seafloor terrain measurement technology[J]. Acta Geodaetica et Cartographica Sinica, 2017, 46(10): 1786-1794.

[2] 阳凡林,韩李涛,王瑞富,等. 多波束声纳水柱影像探测中底层水域目标的研究进展[J]. 山东科技大学学报(自然科学版),2013,32(6):75-83.
YANG Fanlin, HAN Litao, WANG Ruifu, et al. Progress in object detection in middle and bottom-water based on multibeam water column image[J]. Journal of Shandong University of Science and Technology(Natural Science), 2013, 32

(6):75-83.

[3] 陈嘉阳,卜宪海,陈殿称,等. 顾及地形复杂度因子权重的多波束点云抽稀算法[J]. 山东科技大学学报(自然科学版),2022,41(5):21-29.
CHEN Jiayang,BU Xianhai,CHEN Diancheng,et al. A thinning algorithm of multibeam point cloud considering weight of terrain complexity factor[J]. Journal of Shandong University of Science and Technology(Natural Science),2022,41(5):21-29.

[4] 吴焕萍,潘懋,胡金星等. 规则格网 DTM 快速构建算法研究[J]. 计算机应用研究,2004(6):26-28.
WU Huanping,PAN Mao,HU Jinxing et al. Study on quick algorithm for generating regular grid DTM[J]. Application Research of Computers,2004(6):26-28.

[5] AGARWAL P K,ARGE L,DANNER A. From point cloud to grid DEM:A scalable approach[C]//Progress in Spatial Data Handling:12th International Symposium on Spatial Data Handling,Berlin,Heidelberg:Springer. 2006:771-788.

[6] 彭敬,王磊,卢位,等. 基于神经网络的伽马射线定位算法及 CUDA 计算[J]. 核电子学与探测技术,2024,44(4):663-672.
PENG Jing,WANG Lei,LU Wei,et al. Gamma ray positioning algorithm based on neural network and CUDA calculation [J]. Nuclear Electronics and Detection Technology,2024,44(4):663-672.

[7] RUBIO C,GONZALO J,SIMINSKI J,et al. Efficient computation of the geopotential gradient in graphic processing units [J]. Advances in Space Research,2024,74:332-347.

[8] SCHÜTZ M,HERZBERGER L,WIMMER M. SimLOD:Simultaneous LOD generation and rendering for point clouds [J/OL]. Proceedings of the ACM on Computer Graphics and Interactive Techniques,2024,7(1). DOI:10.1145/3651287.

[9] 郭大佑,刘开元,章慧英,等. 基于 GPU 高速并行计算实现逆信噪比-复值退相关 OCTA 实时成像[J/OL]. 中国激光,2024,51(9). DOI:10.3788/CJL231299.
GUO Dayou,LIU Kaiyuan,ZHANG Huiying,et al. Inverse SNR and complex-valued decorrelation OCTA real-time imaging based on GPU high-speed parallel computing [J/OL]. Chinese Journal of Lasers,2024,51(9). DOI:10.3788/CJL231299.

[10] 韩丰,高嵩,薛峰,等. 基于 CUDA 的并行雷达拼图算法研究[J]. 气象,2023,49(10):1246-1253.
HAN Feng,GAO Song,XUE Feng,et al. Study of algorithms for radar networking based on CUDA[J]. Meteorological Monthly,2023,49(10):1246-1253.

[11] 熊超,王欣,王鑫杰,等. 基于 CUDA 的航空 γ 能谱数据小波降噪并行加速算法[J/OL]. 核技术,2024,47(4). DOI:10.11889/j.0253-3219.2024.hjs.47.040201.
XIONG Chao,WANG Xin,WANG Xinjie,et al. CUDA-based parallel acceleration algorithm for wavelet denoising of airborne γ-ray spectrometry data[J/OL]. Nuclear Techniques,2024,47(4). DOI:10.11889/j.0253-3219.2024.hjs.47.040201.

[12] 李朝奎,方军,肖克炎,等. 基于 GPU 的地形可视化加速算法研究[J]. 地球学报,2020,41(2):303-308.
LI Chaokui,FANG Jun,XIAO Keyan,et al. Research on acceleration algorithm of terrain visualization based on GPU[J]. Acta Geoscientica Sinica,2020,41(2):303-308.

[13] LEE D T,SCHACHTER B J. Two algorithms for constructing a Delaunay triangulation[J]. International Journal of Computer and Information Science,1980,9(3):219-242.

(责任编辑:高丽华)