DOI:10.3969/j.issn.1003-5060.2024.12.010

用于 ICP 的近似 KD-Tree 搜索加速器设计及 FPGA 实现

郑凯磊，　陈　强，　肖　昊
(合肥工业大学 微电子学院,安徽 合肥　230601)

摘　要:为了加速迭代最近点(iterative closest point,ICP)算法中 k 近邻(k-nearest neighbor,KNN)搜索过程,文章根据近似 K 维树(K-dimensional tree,KD-Tree)数据结构,基于现场可编程门阵列(field programmable gate array,FPGA)提出一种高性能的 KNN 搜索加速器;分析近似 KD-Tree 数据结构的可行性,结果表明该数据结构能够满足 ICP 算法精度要求,并提高计算的并行度和性能;为了解决近似 KD-Tree 建树过程耗费时间长的问题,设计基于分治归并排序的具有反馈数据通路的树构建计算模块,该模块可在 8.95 ms 内计算出 256 个空间的树节点并完成树构建;为了优化点云暴力搜索过程,设计一种高吞吐率的点云搜索模块,可以在 0.49 ms 内完成近 30 000 个点的最近点搜索。研究结果表明,与相关的设计相比,该文提出的硬件加速方法可以有效降低 KNN 搜索时间复杂度,提高算法性能。

关键词:K 维树(KD-Tree);迭代最近点(ICP)算法;三维重建;硬件加速;现场可编程门阵列(FPGA)
中图分类号:TN791;TN402　　**文献标志码**:A　　**文章编号**:1003-5060(2024)12-1648-07

Design and FPGA implementation of approximate KD-Tree searching accelerator for ICP

ZHENG Kailei,　CHEN Qiang,　XIAO Hao
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: To shorten the time that k-nearest neighbor (KNN) searching process in iterative closest point (ICP) algorithm takes, a high-performance KNN accelerator is designed, with approximate K-dimensional tree (KD-Tree), on the field programmable gate array (FPGA). The feasibility of approximate KD-Tree is analyzed with the result showing that this data structure can meet the precision requirements of ICP algorithm and boost its parallel degree as well as performance. To improve the time-consuming tree building phase of approximate KD-Tree, a tree building calculation module, which is based on merge sort and equipped with feedback data channels, is made to be able to determine 256 space nodes and finish tree building in 8.95 ms. To better point cloud searching process, a point cloud searching module with high throughput rate is created so that it can execute the nearest search of almost 30 000 points in 0.49 ms. The research results show that, compared to other relevant designs, the hardware acceleration method in this paper can significantly lower the complexity of KNN searching time and upgrade the performance of ICP algorithm.

Key words: K-dimensional tree (KD-Tree); iterative closest point (ICP) algorithm; three-dimensional reconstruction; hardware acceleration; field programmable gate array (FPGA)

收稿日期:2022-12-30;**修回日期**:2023-02-23
基金项目:国家自然科学基金资助项目(61974039)
作者简介:郑凯磊(1997—),男,山东潍坊人,合肥工业大学硕士生;
陈　强(1962—),男,江苏江阴人,博士,合肥工业大学教授,博士生导师;
肖　昊(1982—),男,安徽合肥人,博士,合肥工业大学教授,博士生导师,通信作者,E-mail:xiaohao@hfut.edu.cn.

随着计算机视觉的飞速发展，三维点云配准在许多行业中愈发受到关注，如逆向工程、文化遗产保护、医学、机器人、自动驾驶等[1-5]。点云是一种可视化三维扫描仪（如激光雷达或深度图像传感器）输出物体表面点三维坐标的方法[6]。三维点云配准是通过求解最优刚性变换来实现点云与点云的匹配[7-9]，将一个物体不同视角下的点云合并，进而对齐到一个坐标系中，得到完整点云模型的过程[10]。

迭代最近点（iterative closest point，ICP）[11-12]算法是最主要、应用最广泛的点云配准算法[13]，它对异常数据值具有鲁棒性。ICP 算法主要包括 k 近邻（k-nearest neighbor，KNN）搜索、对物体位姿旋转矩阵和平移矩阵计算[14] 2 个过程。KNN 搜索算法是许多计算机视觉算法的基本组成部分，文献[15-16]对其在 ICP 中的应用进行研究和改进，结果表明 KNN 搜索步骤占 ICP 算法处理时间的 90%以上，成为 ICP 算法的计算瓶颈。

随着现场可编程门阵列（field programmable gate array，FPGA）器件工艺水平的发展，器件密度显著提高，对于一些计算量要求大或者是占用大量带宽的任务可以利用 FPGA 并行加速完成。KNN 搜索需要对大量的点对穷尽计算欧氏距离，可以利用 FPGA 对此过程进行并行化处理，以提高计算速度。

K 维树（K-dimensional tree，KD-Tree）[17]是最著名的最近点算法之一，文献[11]建议使用 KD-Tree 加速 KNN 搜索部分。KD-Tree 可以在数据维度不太高的情况下按照某种顺序将无序化的点云进行有序化的树结构构建，在海量的数据环境中有效搜索最近点，但此方法仍需要繁琐的回溯过程，降低了运行效率。文献[18]基于近似 KD-Tree 数据结构设计一种用于 ICP 算法的 KNN 搜索加速器，但该设计采用先计算树节点后进行树构建的方法，使得数据结构构建耗费时间长，同时在 KNN 搜索时，数据吞吐率较低；文献[15]最早在 FPGA 上采用并行暴力搜索的方法解决 ICP 算法中的 KNN 搜索问题，但该方法的时间复杂度与参考点数量呈线性关系增长，不适用于大规模点云配准；文献[16]设计一种基于 FPGA 层次图复用的 KNN 搜索的目标位姿估计加速器，但该设计层次图构建耗费时间长，同时需要大量的片上存储资源存储层次图数据结构，不利于在低成本 FPGA 上实现；文献[19]基于双分割体素数据结构（double segmentation voxel structure，DSVS）在 FPGA 上实现一种用于三维点云配准的高效 KNN 搜索加速器，但该设计的时间复杂度不随点云规模变化而变化，当点云规模逐步缩小时，该设计的性能降低。

文献[20]提出一种近似 KD-Tree 数据结构算法，本文基于该算法设计一种点云搜索加速器，用于加速 ICP 算法中的 KNN 搜索过程。对于该算法中的树构建和近似最近点（approximate nearest neighbor，ANN）搜索 2 个关键问题，本文分别设计一种基于归并排序的具有反馈数据通路的树构建计算模块和一种高吞吐率的最近点搜索计算模块。实验结果表明，当点云规模为 30 000 时，本文提出的加速器完成数据结构构建仅用 8.95 ms，并在 0.49 ms 内完成最近点搜索，同时保证 ICP 点云配准的准确性。

1　算法介绍

标准的 KD-Tree 使用试探性回溯搜索查询最近点，近似 KD-Tree 由 KD-Tree 演变而来，主要包括树构建和 ANN 搜索 2 个步骤。ANN 搜索公式为：

$$\| q - p_f \| \leqslant (1+\varepsilon) \| q - p_c \| ,\ \varepsilon \geqslant 0 \quad (1)$$

其中：q 为目标点；p_f 为在参考点云中找寻的近似最近点；ε 为误差值；p_c 为在参考点云中找寻的真实最近点。

近似 KD-Tree 算法是返回目标点在参考点云中的一个近似最近点，该值由树结构的初始深度决定，首先穷尽查找相应子空间中的参考点，如果没有找到真正的最近点，那么返回一个近似结果。当划分的子空间足够大时，由于空间内点的接近性，该算法计算的结果非常接近真实值。同时，该算法避免了回溯搜索开销，有效提高 ICP 算法的运行效率。

1.1　建树过程

近似 KD-Tree 的每个树节点都可以看作一个垂直于节点选定某长 x、宽 y、高 z 维度的超平面，将空间分割成 2 个子空间，左子空间各点选定维度的值不超过该节点在此维度的值，右子空间反之。三维空间近似 KD-Tree 算法如图 1 所示。

本文按照三维空间长、宽、高的维度顺序反复逐次迭代计算树节点，并进行近似 KD-Tree 数据结构构建。将点选定维度的各值进行排序，以中位数作为树节点。从图 1a 可以看出，对于树节点（8，9，1），左子空间（5，2，2）（1，7，6）（8，9，1）各点

长度值都不超过(8,9,1)该树节点的长度值,从三维空间来看都在树节点的左侧,右子空间反之。

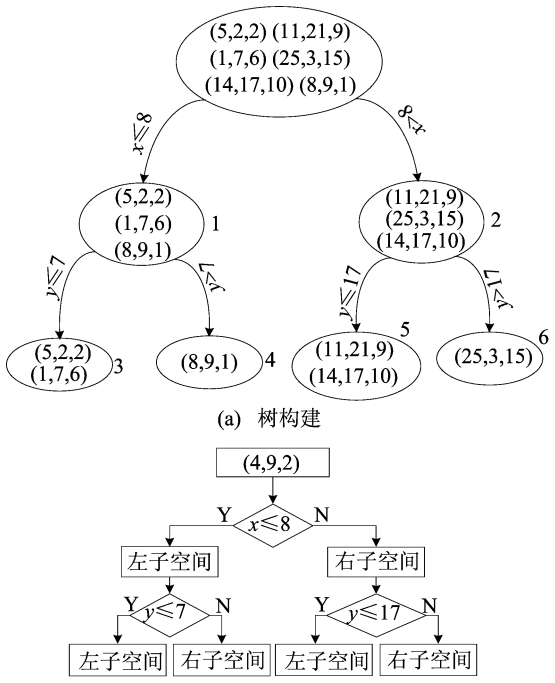


图1 近似KD-Tree算法

1.2 最近点搜索过程

以图1a构建的树为例,假设要对目标点(4,9,2)进行最近点(1-nearest neighbor,$k=1$)搜索。首先将该点长度值与第1层树节点长度值进行比较,此时$4<8$,确定进入左子空间查找,即1号子空间;然后将目标点宽度值与此子空间树节点宽度值进行比较,此时$9>7$,最终确定进入4号子空间,对此空间内的所有参考点与该目标点进行欧氏距离计算,穷尽搜索找出最近点为(8,9,1),最近点搜索完毕。对于其他目标点搜寻最近点的步骤也遵循上述方法。

2 硬件加速方法

2.1 KNN加速器整体电路架构

本文设计的KNN搜索加速器主要由处理系统(processing system,PS)和可编程逻辑(programmable logic,PL)2个部分组成,如图2所示。

第三代双倍速率同步动态随机存储器(third generation of double-data-rate synchronous dynamic random-access memory,DDR3 SDRAM)负责存储点云数据和最近点计算结果。PL端与PS端通过高级可扩展接口(advanced extensible interface,AXI)协议进行通信。顶层逻辑控制模块负责控制整个加速器的时序逻辑。当ICP算法的KNN搜索步骤开始时,树构建计算模块开始构建近似KD-Tree数据结构,并将树节点和构建好的数据结构存储到最近点搜索计算模块中。最近点搜索计算模块读取目标点云,负责计算查找其对应的参考最近点,并返回计算结果,完成ICP算法的KNN搜索过程。ICP算法的其他步骤则由PS端低功耗嵌入式中央处理器(central processing unit,CPU)进行处理。

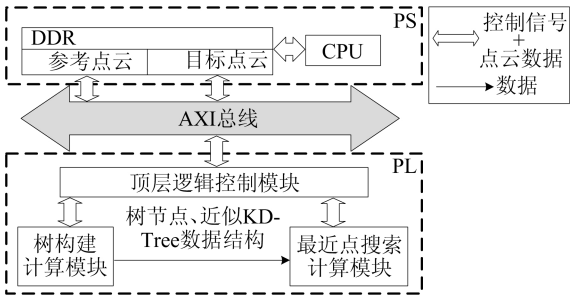


图2 KNN搜索加速器

2.2 具有反馈数据通路的树构建计算模块

树构建需要多次对点云进行排序和计算树节点,此过程耗费大量时间,分治归并排序算法如图3所示。

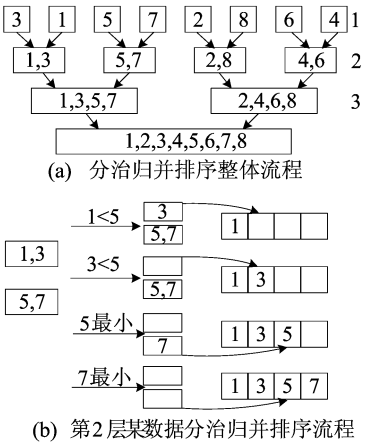


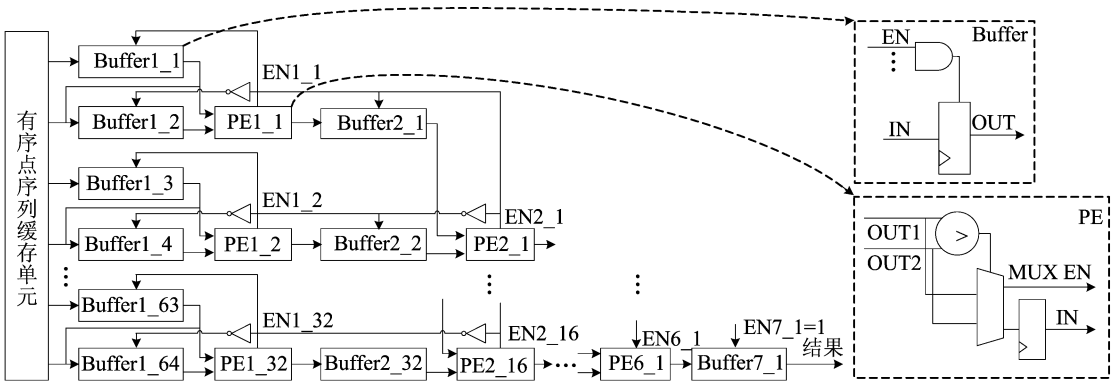
图3 分治归并排序算法

从图3a可以看出,本文根据分治归并排序算法,对点某个维度的值进行排序求中位数树节点,进行树构建。从图3b可以看出,以图3a第2层某两子序列数据分治归并排序为例,该算法需要花费两子序列总长度相应的时间,比较两子序列首位数字大小,对选定数字进行移动来完成排序操作,时间复杂度为$O(4)$,完成该层排序运行时间为$O(8)$。因此,完成1层m个数据排序的运行

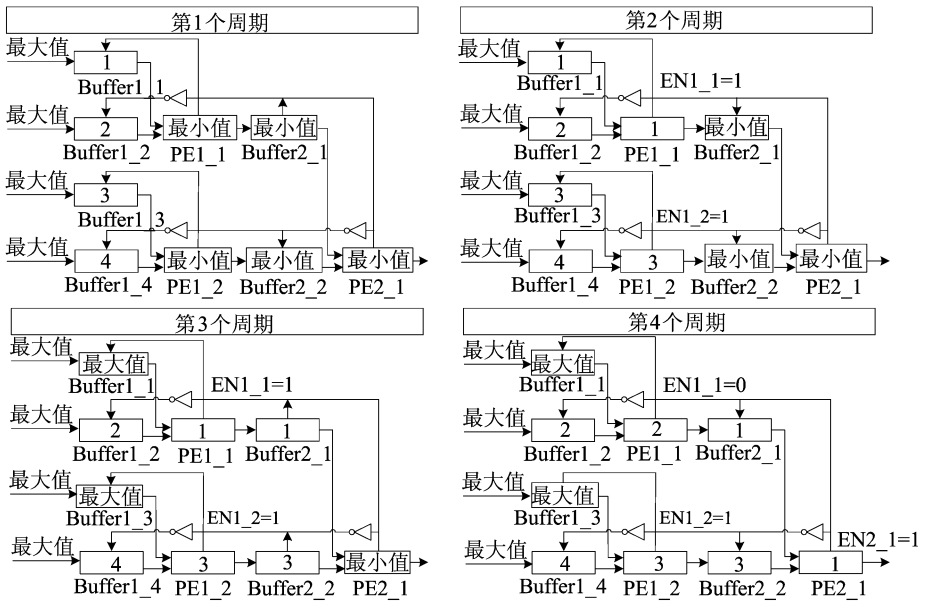
时间复杂度为 $O(m)$。对 n 个数据进行排序时，可以分为 lb n 层，每层 n 个数据，由于该过程为串行操作，因此该算法时间复杂度为 $O(n\text{lb}\ n)$。

本文根据分治归并排序设计了一种具有反馈数据通路的树构建计算模块电路架构，如图 4a 所示。设计将参考点进行长、宽、高维度标记，经过该模块可以得到所需长度的有序点序列，进行树节点计算和树结构构建。缓存器(Buffer)流水线式接收参考点，处理单元(process element，PE)只对参考点维度值进行大小比较操作，输出较小值对应的参考点。

以 Buffer1_1、Buffer1_2、Buffer1_3、Buffer1_4 输入通路为例，4 输入通路数据流如图 4b 所示。



(a) 树构建计算模块电路架构



(b) 4输入通路数据流

图 4　树构建计算模块

首先预读取参考点数据存入第 1 列 Buffer 中。本模块开始运行时，从第 1 个周期开始，PE1_1 执行 Buffer1_1 与 Buffer1_2 数值大小比较操作并从第 2 个周期生成输入数据写使能 EN1_1 信号，PE1_2 执行 Buffer1_3 与 Buffer1_4 数值大小比较操作并从第 2 个周期生成 EN1_2 信号。从第 3 个周期开始，PE2_1 执行 Buffer2_1 与 Buffer2_2 中的数值大小比较操作并从第 4 个周期生成 EN2_1 信号。从第 1～第 3 个周期，EN1_1 单独决定 Buffer1_1 或 Buffer1_2 是否写入新数据，EN1_2 单独决定 Buffer1_3 或 Buffer1_4 是否写入新数据，Buffer2_1 与 Buffer2_2 每个周期都写入新数据。从第 4 个周期开始，该模块开始计算出有序数值，EN2_1 单独决定 Buffer2_1 或 Buffer2_2 是否写入新数据，EN2_1 与 EN1_1 进行与操作决定 Buffer1_1 或 Buffer1_2 是否写入新数据，EN2_1 与 EN1_2 进行与操作决定 Buffer1_3 或 Buffer1_4 是否写

入新数据。图 4b 有 2 层 PE 计算单元,对应于图 3a可以并行完成 2 层子序列的比较移位排序,时间复杂度降为 $O(n(\mathrm{lb}\ n)/2)$。按图 4b 示例设计 6 层PE,并行完成归并排序 6 层数据的比较移位排序工作,进一步将时间复杂度降为 $O(n(\mathrm{lb}\ n)/6)$。

以 32 768 个三维参考点构成的无序点序列为例,第 1 次树构建过程如下:首先从 DDR3 SDRAM 中读取 64 个无序点并存入到图 4a 第 1 列Buffer 中,输入到该模块进行点排序操作,将会获得一个长度为 64 的由小到大有序点序列;重复该过程 512 次,将得到的 512 个有序点序列依次存入有序点序列缓存单元中;第 2 轮排序先将 64 个有序点序列的最小值分别输入到 64 个 Buffer,其余有序点根据反馈读写使能 EN 信号逐个判断,并输入到树构建模块中进行排序,本轮会得到 8 个长度为 4 096 的有序点序列;第 3 轮排序将数组分别输入 8 个排序数据通路中,关闭其余 56 个数据通路,排序完毕,得到 1 个中位数树节点和 2 个长度为 16 384 的有序点序列。第 2 次迭代分别对 2 个长度为 16 384 的有序点序列进行排序,得到 2 个树节点和 4 个长度为 8 192 的有序点序列。以此类推,多次迭代计算树节点并完成树构建,树构建模块每隔 2 个周期输出一个排序结果。最终将计算得到的树节点和构建好的近似 KD-Tree 数据结构分别写入图 2 最近点搜索计算模块中目标子空间的标志生成单元和树结构存储单元中。

在本文设计的电路架构中,基于点维度标记的方法使得在计算树节点的过程中可以完成树构建过程,提高计算效率。

2.3 最近点搜索计算模块

树构建计算模块获得多个参考点云子空间,每个子空间中又具有多个点需要穷尽计算求取最近点。为了最大提升计算效率,本文设计一种高吞吐率的最近点搜索计算模块,如图 5所示。

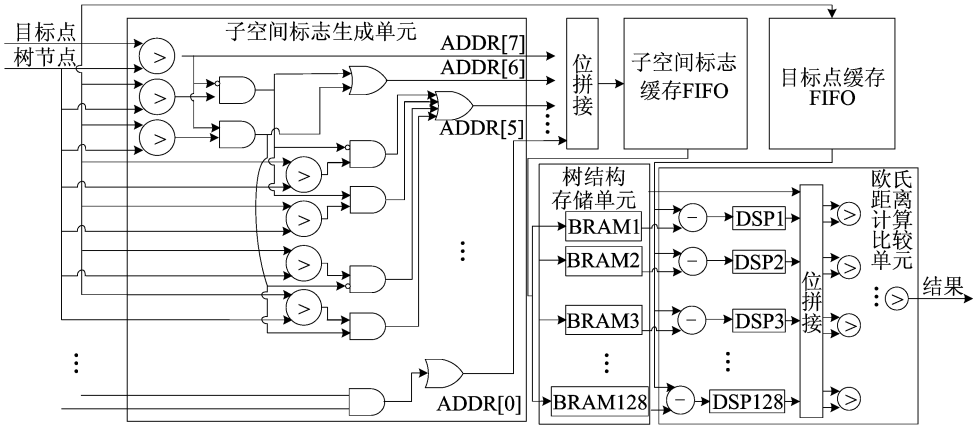


图 5 最近点搜索计算模块

最近点搜索时,为了同时读取各子空间参考点,本文将各随机存取内存块(block random access memory,BRAM)相同的地址表示同一个目标子空间的不同参考点某维度的值,每个参考点用 3 个地址空间存储。目标点进入子空间标志生成单元与已完成存储的树节点进行计算,每个周期生成 1 个存储单元子空间标志(address,ADDR),即 BRAM 读地址。计算比较单元中调用 FPGA 的数字信号处理(digital signal processing,DSP)资源计算参考点与目标点之间的欧氏距离。为了增大数据吞吐率,本文设计将欧氏距离与参考点进行位拼接(高 48 bit 为参考点,低 36 bit 为欧氏距离),后面步骤只对欧氏距离进行比较操作,使得该设计在进行欧氏距离比较的同时求得最近点。首先,128 个位拼接值流水线经过两两比较可以得到 64 个近似最近点,进一步两两比较计算得到 32 个近似最近点,以此类推,可以在一定周期内,以高吞吐率计算得到所需的多个近似最近点。最后,将计算结果返回存储到 DDR3 SDRAM 中,最近点搜索完毕。

3 实验分析

3.1 精度分析

实验的点云数据选用下采样到 32 768 个点的斯坦福兔子模型[21]。本文将原始模型作为参考点云,旋转 0°~360°的模型作为多个目标点数据集进行测试,以传统 KD-Tree 搜索 ICP 算法的均方根误差作为精度判断基准,精度分析曲线如

图 6 所示。

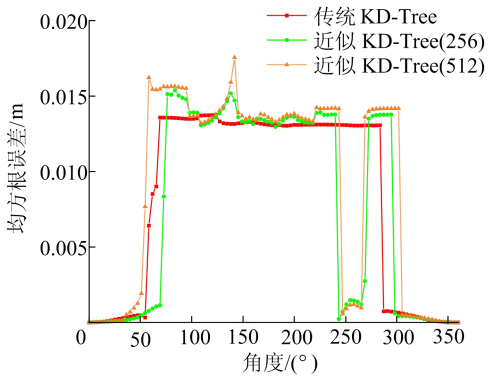


图 6　精度分析曲线

一般认为均方根误差在 10 mm 以下点云配准成功[16]，因此传统 KD-Tree 算法配准成功的角度为 0°～70°、300°～360°，ICP 算法适用于物体位姿变换较小的情况。对上述角度范围进行精度分析确定子空间大小，实验在构建子空间数量为 256 个时，精度曲线与基准曲线能够很好地拟合。当子空间数量达到 512 个时，精度曲线过早偏离基准曲线。为了满足 ICP 算法的精度要求，同时尽可能降低子空间中点的数量，使最近点搜索过程时间复杂度最小，最终确定将参考点云划分为 256 个子空间，每个子空间中有 128 个点。

3.2　加速性能对比及分析

本文的实验设备选用 XilinxZYNQ XC7Z035-FFG676-2 开发板。实验中，FPGA 以 200 MHz 时钟频率运行，最近点搜索运行时间和 FPGA 资源利用率分别见表 1、表 2 所列，KNN 搜索性能比较见表 3 所列。

表 1　最近点搜索运行时间

进程	运行时间/ms
计算树节点	8.46
树结构存储	0.49
最近点搜索	0.49

表 2　FPGA 资源利用率

资源	BRAMs/KiB	LUTs	FFs	DSP 数量
已使用	6 318	51 185	43 946	128
利用率/%	35.1	29.8	12.8	14.2

表 3　KNN 搜索性能比较

参数	文献[15]	文献[16]	文献[18]	文献[19]	本文方法		
参考点数量	32 768	30 000	32 768	30 000	32 768	32 768	32 768
目标点数量	32 768	65 000	65 536	10 000	32 768	65 536	10 000
K	1	30	30	5	1	30	5
搜索时间复杂度	32 768	90	128		128	128	128
时钟频率/MHz	125	250	200	200	200	200	200
DSP 数量	78		128	83	128	128	128
BRAMs/KiB	2 772	23 552	4 122	25 236	6 318	6 318	6 318
运行时间/ms	546	146	23.6	20	0.49	0.98	0.15

文献[15]采用并行暴力搜索的方法顺序地将目标点与每一个参考点计算欧氏距离并进行比较，以求解最近点，它的算法时间复杂度为参考点的总数量 32 768，对于 32 768 个目标点搜索最近点耗时长约 546 ms。而本设计采用近似 KD-Tree 数据结构，通过对参考点建立树结构，将算法时间复杂度降为 128，相同点云规模下，本文的最近点搜索运行时间仅为 0.49 ms。文献[16]设计的电路架构使得层次图数据结构构建这一过程耗时长约 289 ms，同时需要 23 552 KiB 的 BRAM 片上存储空间进行层次图预存储，需要部署在高成本的 Xilinx UltraScale＋ZU9EG 开发板上，而本设计中仅使用 6 318 KiB 的 BRAM 存储近似 KD-Tree 数据结构的子空间数据，得以在较低成本 FPGA 上实现。同时，对于 65 536 个目标点的 KNN 搜索，本设计仅用 0.98 ms，相比文献[16]性能提升 148 倍。文献[18]基于近似 KD-Tree 数据结构设计了一种 KNN 搜索加速器，耗费约 18.6 ms 的时间寻找树节点并构建树结构，而本文设计仅需 8.95 ms 就可以完成数据结构构建。文献[18]在计算最近点时每 72 个周期才计算得到 1 个目标点的 30 个近似最近点结果。而本文设计的欧氏距离比较单元可以高并行度计算最近点，在 K 为 30 时，本文的设计 4 个周期就可以计算出 1 个目标点的 30 个近似最近点结果，数据吞吐率大幅提升，KNN 搜索性能相比文献[18]提升近 24 倍。文献[19]基于 DSVS 数据结构，可以花费较少的时间进行数据构建，并且可以调节子空间点数量，减少冗余搜索。但是，该算法的时间复杂度不随点云规模的改变而改变，在点

云规模较大时,加速效果较为显著,在点云规模逐步缩小时,该设计的性能降低。在目标点云规模为 10 000 左右时本文的设计 KNN 搜索耗时 0.15 ms,相比文献[19]提高了近 130 倍。

4 结 论

本文基于近似 KD-Tree 数据结构提出一种用于 ICP 算法的 KNN 搜索加速器。针对树构建过程提出一种基于分治归并排序的具有反馈数据通路的树构建计算模块,在计算树节点的过程中完成树构建,从而避免额外的时间开销;同时提出一种高吞吐率的点云搜索模块,显著减少 KNN 搜索运行时间。实验结果表明,所提出的加速器能够有效提升 ICP 算法中 KNN 搜索性能。

[参 考 文 献]

[1] 陆尚鸿,李文国. 基于点云边界质心的粗配准方法[J]. 电子科技,2022,35(4):53-59.

[2] ABDONI F, MOUSAVINIA A, PAZOKI F. A Restricted 4PCS algorithm for registering low overlap point cloud [C]//2022 27th International Computer Conference, Computer Society of Iran (CSICC). [S. l. : s. n.], 2022: 1-5.

[3] SABARIEGO N, CENTENO J A S. Modeling of roofs from point clouds using genetic algorithms [J]. Boletim de Ciéncias Geodésicas, 2020, 26(1): 1-21.

[4] 孙培芪,卜俊洲,陶庭叶,等. 基于特征点法向量的点云配准算法[J]. 测绘通报,2019(8):48-53.

[5] 王松波,李马骁,李海瑞,等. 基于三维激光雷达技术的输电线路廊道障碍物检测研究[J]. 电子科技,2019,32(4):81-84.

[6] PAZOKI F, MOUSAVINIA A, ABDONI F. Accelerated 4PCS algorithm for point cloud registration[C]//2022 27th International Computer Conference, Computer Society of Iran (CSICC). [S. l. : s. n.], 2022: 1-5.

[7] SEGAL A V, HEAHNEL D, THRUN S. Generalized-ICP [M]//Robotics: science and systems. Cambridge, Mass. : MIT Press, 2009: 435.

[8] MA J, ZHAO J, JIANG J, et al. Locality preserving matching[J]. International Journal of Computer Vision, 2019, 127(5): 512-531.

[9] WANG X, LI Y, PENG Y, et al. A coarse-to-fine generalized-ICP algorithm with trimmed strategy[J]. IEEE Access, 2020, 8: 40692-40703.

[10] 李慧慧,刘超,陶远. 一种改进的 ICP 激光点云精确配准方法[J]. 激光杂志,2021,42(1):84-87.

[11] BESL P J, MCKAY N D. A method for registration of 3-D shapes[J]. IEEE Tranctions on Pattern Analysis and Machine Interlligence, 1992, 14(2): 239-256.

[12] CHEN Y, MEDIONI G. Object modelling by registration of multiple range images[J]. Image and Vision Computing, 1992, 10(3): 145-155.

[13] 张晗,康国华,张琪,等. 基于改进 SAC-IA 算法的激光点云粗配准[J]. 航天控制,2019,37(5):67-74.

[14] KOSUGE A, OSHIMA T. An object-pose estimation acceleration technique for picking robot applications by using graph-reusing k-NN search[C]//2019 First International Conference on Graph Computing (GC). [S. l. : s. n.], 2019: 68-74.

[15] BELSHAW M S, GREENSPAN M A. A high speed iterative closest point tracker on an FPGA platform[C]//2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops. [S. l.]: IEEE, 2009: 1449-1456.

[16] KOUSUGE A, YAMAMOTO K, AKAMINE Y, et al. An SoC-FPGA-based iterative-closest-point accelerator enabling faster picking robots[J]. IEEE Transactions on Industrial Electronics, 2020, 68(4): 3567-3576.

[17] BENTLEY J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM, 1975, 18(9): 509-517.

[18] LI Y, ZHENG K, XIAO H. A KNN accelerator based on approximate KD tree for ICP[C]//2022 International Conference on Image Processing and Media Computing (ICIPMC). [S. l. : s. n.], 2022: 124-128.

[19] SUN H, LIU X, DENG Q, et al. Efficient FPGA implementation of K-nearest-neighbor search algorithm for 3D LIDAR localization and mapping in smart vehicles[J]. IEEE Transactions on Circuits and Systems Ⅱ (Express Briefs), 2020, 67(9): 1644-1648.

[20] GREENSPAN M, YURICK M. Approximate kd tree search for efficient ICP[C]//Fourth International Conference on 3-D Digital Imaging and Modeling. [S. l. : s. n.], 2003: 442-448.

[21] Stanford University Computer Graphics Laboratory. The stanford 3D scanning repository[DB/OL]. (1996-08-05) [2021-08-24]. http://www.graphics.stanford.edu/data/3Dscanrep/1.

(责任编辑 张 锴)