DOI:10.3969/j.issn.1003-5060.2025.02.006

面向点云识别的最近邻搜索硬件加速器

陈 立, 李桢旻, 马宇晴
(合肥工业大学 微电子学院,安徽 合肥 230601)

摘 要:动态图卷积神经网络(dynamic graph convolutional neural network,DGCNN)作为点云识别主流算法之一,主要由边缘卷积层构成,而最近邻搜索操作占据边缘卷积层 63%的计算时间。文章针对现有的最近邻搜索加速器准确率较低、速度较慢的问题,设计一种面向点云识别的最近邻搜索硬件加速器。该加速器采用基于点云分割的并行双调流水排序结构进行 2 轮双调排序,并用曼哈顿距离替代欧氏距离衡量点与点距离的远近。实验结果表明,在同样的实验环境配置下,相较于其他点云最近邻搜索加速器,文章设计的最近邻搜索加速器速度提升了 3.6 倍。

关键词:最近邻搜索;硬件加速器;边缘卷积;双调排序;曼哈顿距离
中图分类号:TN47 **文献标志码**:A **文章编号**:1003-5060(2025)02-0179-06

Nearest neighbor search hardware accelerator for point cloud recognition

CHEN Li, LI Zhenmin, MA Yuqing
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: As one of the mainstream algorithms for point cloud recognition, dynamic graph convolutional neural network(DGCNN) is mainly composed of edge convolutional layers, and the nearest neighbor search takes up 63% of the computing time of edge convolutional layers. Aiming at the problems of low accuracy and slow speed of existing nearest neighbor search accelerators, this paper proposes a design of nearest neighbor search hardware accelerator for point cloud recognition. In this accelerator, a parallel bitonic flow sorting structure based on point cloud segmentation is adopted for two rounds of bitonic sort, and Manhattan distance is used instead of Euclidean distance to measure the distance between points. Experimental results show that the speed of the proposed nearest neighbor search accelerator is 3.6 times faster than that of the existing point cloud nearest neighbor search accelerator under the same experimental environment configuration.

Key words: nearest neighbor search; hardware accelerator; EdgeConv; bitonic sort; Manhattan distance

0 引 言

近年来,随着 3D 数据采集设备如激光雷达、扫描仪等 3D 点云获取技术的成熟,点云[1]成为一种常见的图像类型。点云是 3D 空间中点的集合,每个点包括空间坐标、强度、颜色等信息,而获得的点云数据常用于神经网络的识别。

文献[2]提出的动态图卷积神经网络(dynamic graph convolutional neural network,DGCNN)使用边缘卷积(EdgeConv)算法来获取点云特征,边缘卷积需要获取点云的最近邻(k-nearest neighbor,KNN)图来揭示点的空间关系。

最近邻搜索常用于获得点与点之间的空间几何结构信息。文献[3]提出基于哈希散列的最近邻搜索算法,是一种将高维数据映射到低维空间进行搜索的算法;文献[4]提出基于树的搜索算

收稿日期:2023-03-23;**修回日期**:2023-04-12
基金项目:国家重点研发计划资助项目(2018YFB2202604);安徽省高校协同创新资助项目(GXXT-2019-030)
作者简介:陈 立(1997—),男,湖北武汉人,合肥工业大学硕士生;
李桢旻(1982—),男,安徽合肥人,博士,合肥工业大学讲师,硕士生导师,通信作者,E-mail:zhenmin.li@hfut.edu.cn.

法,通过将点分割到不同区域实现快速搜索;文献[5]提出基于层次图的搜索算法,通过构建联通图进行搜索。以上搜索算法在点云数量较多时搜索速度较快,但由于使用近似最近邻搜索,搜索最近邻点准确率无法达到100%。在硬件方面,文献[6]和文献[7]分别基于近似KD树(k-dimension tree)和小世界图算法进行硬件实现。

本文主要工作如下:

1) 最近邻搜索占据边缘卷积大部分运算时间。针对时间消耗较大的问题,本文设计最近邻搜索硬件加速器,该加速器基于双调排序和曼哈顿距离计算,使用流水结构以提高资源利用率。

2) 现有的最近邻搜索加速器基于近似最近邻搜索算法实现,存在精度较低的问题,不利于在神经网络中应用;而本文提出的最近邻搜索加速器使用一种基于分割的方式,通过2轮排序实现最近邻搜索,由于采用精确搜索,准确率没有下降,仍为100%。

1 点云识别理论基础

1.1 边缘卷积

DGCNN是一种应用于点云识别的神经网络,通过边缘卷积构建局部邻域图获取局部信息和全局信息。边缘卷积的计算流程如图1所示。

从图1可以看出,通过最近邻搜索算法找到某点最近的k个点,构建KNN图,随后进行卷积激活与池化,其中最近邻搜索占据边缘卷积63%的计算时间。

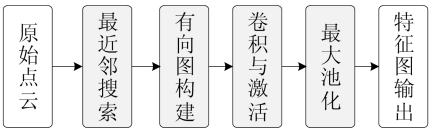

图1 边缘卷积流程

1.2 最近邻搜索

对于F维点云中的n个点,定义为:

$$X=\{\boldsymbol{X}_1,\cdots,\boldsymbol{X}_n\}\subseteq\mathbf{R}^F \tag{1}$$

设置$F=3$,维度F表示给定的特征维度。每个点云中的点都包含坐标$\boldsymbol{X}_i=(x_i,y_i,z_i)$。最近邻搜索一般是为了获取点与周围点的局部特征,用于点集的分类和分割。

最近邻点设置为3时的最近邻搜索过程如图2所示。

图2中的白色点首先计算与周围点的距离。常用的距离有欧氏距离、曼哈顿距离和切比雪夫距离,通过比较距离来查询距离最近的多个点,通过建立点与点之间的关系构造KNN图。

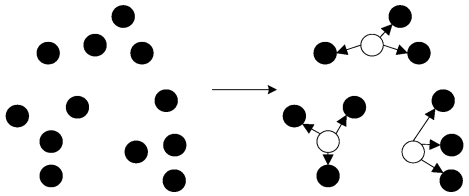

图2 最近邻点为3时的搜索过程

2 最近邻搜索硬件加速器设计

2.1 系统整体架构

最近邻搜索硬件加速器的整体结构如图3所示,主要由全局控制模块、点云分割模块、距离计算模块和排序电路模块组成。

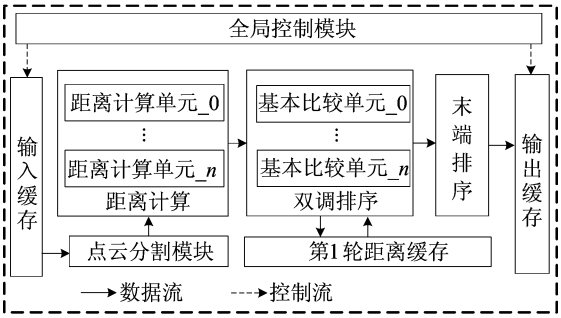

图3 最近邻搜索加速器结构

为了处理原始点云,首先将带标签的点进行分割,将1 024个点分割到不同区域;然后对分割到不同区域的点云数据进行距离计算,将计算完毕的距离进行第1轮双调排序,获取距离查询点的1 024个距离数值中较小的128个距离;然后对这128个距离进行第2轮排序,获得128个距离中最小的16个距离;之后对乱序的16个距离从大到小排序得到顺序的16个距离,则这16个数对应的点也就是查询点在1 024个点中找到的精确的16个最近邻点;最后从16个距离中选出最小的k个距离,取出距离的高位作为索引并对索引进行缓存,缓存1 024组索引后进行拼接并输出。

2.2 点云分割模块设计

本文最近邻搜索硬件加速器输入一张点云图像,包含1 024个点。每个点云数据包含标签以及x、y、z坐标值。由于一次并行输入过大不利于资源节省,本文将1 024个点分8个周期取出。点云分割模块电路如图4所示。

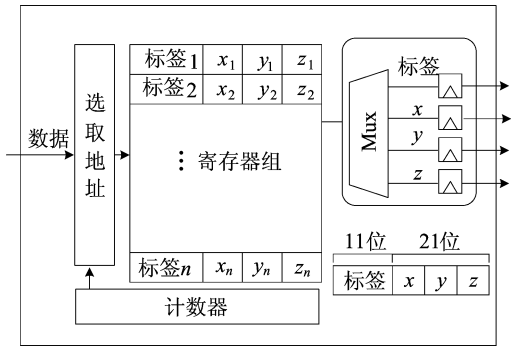

图 4　点云分割模块电路图

32 位宽的点被存入外部存储器,通过计数器计数产生外部存储器片选信号,计数器控制片选信号范围在信号 1～信号 8 之间,取址范围随着 Mux 的片选信号递增。片选信号为 1 时,取址范围为 1 到 128,从外部存储地址为 1 到 128 范围内取出 128 个 32 位点。

每次片选信号加 1,地址范围加 128,经过 8 个周期取出所有 128 个数据点,然后将取出的点放入寄存器组中,每个 32 位点中高 11 位为标签位,20 位～14 位为 x 值,13 位～7 位为 y 值,6 位～0 位为 z 值。将存入寄存器组中的点使用划分单元按位宽分割为标签、x、y、z 值并输出,1 个周期取出 128 组这样的数据。

2.3　距离计算模块设计

距离计算电路主要是计算查询点与 1 024 个待查询点的距离,从而判断最近邻点的远近,以获取最近的点。一般距离搜索算法中使用欧氏距离来衡量距离的远近,但在 k 近邻的搜索过程中,距离仅用于衡量点与点的远近,并不需要参与边的特征计算,因此本文使用曼哈顿距离替代欧氏距离对距离进行衡量。曼哈顿距离计算公式为:

$$d = |x_i - x_j| + |y_i - y_j| + |z_i - z_j| \quad (2)$$

而欧氏距离计算公式为:

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \quad (3)$$

对比可知,曼哈顿距离不需要进行乘法计算,而欧氏距离需要开方,逻辑复杂,计算消耗大。曼哈顿距离计算在硬件电路中不需要使用乘法器,故具有计算简单、在硬件电路中资源消耗少的特点,由此确定曼哈顿距离计算的计算结构、输入数据格式及基本单元,如图 5 所示。

本文距离计算电路的输入是 128 个待查询点和 1 个查询点,输出是 21 位带有标签的距离,以一个查询点(x_q,y_q,z_q)与一个待查询点(x,y,z)为例,查询点的 x_q、y_q、z_q 值与待查询点的 x、y、z 值,并对应相减。

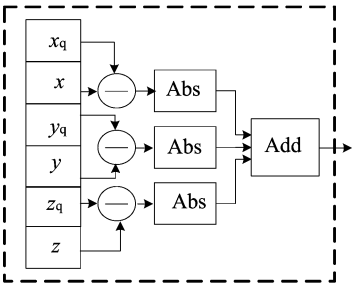

图 5　曼哈顿距离计算单元

将计算的 3 个差分别求绝对值后相加,再与待查询点的标签拼接,最后输出到双调排序电路。以此类推,1 024 个点分 8 次输入,共使用 128 个这样的计算单元对输入数据进行并行的曼哈顿距离计算。待查询点与查询点仅进行 x、y、z 的计算,标签值不参与距离计算。

2.4　排序模块设计

2.4.1　双调排序基本比较单元

文献[8]对基于冒泡排序的最近邻搜索进行了硬件实现,冒泡排序时间复杂度为 $O(n^2)$;而双调排序[9]并行实现时间复杂度仅为 $O((\mathrm{lb}\ n)^2)$,它是一种比较顺序与数据无关的排序方式,是适合并行计算的排序算法,在硬件实现上有诸多优势。双调排序与输入数据无关,电路所做的比较都是固定的,在任意一个步骤中,比较顺序都是预先确定好的。

一个基本的比较单元(compare unit,CU)如图 6 所示。

输入 2 个相对大小未知的值 x 和 y,较小值由上输出口输出,较大值由下输出口输出。

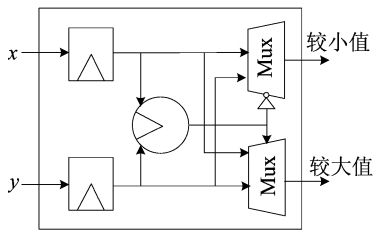

图 6　基本比较单元电路图

8 输入并行双调排序电路如图 7 所示,共使用 6 级比较单元构成排序电路。

输入为乱序的 8 个数字。第 1 层以间隔 1 位的 2 个输入进行比较并交换;第 2 层首先以间隔 2 位的位置进行比较并交换,然后以间隔 1 位的

2个输入进行比较,这些比较同时进行;第3层与第1层相同,按照间隔1位的2个输入进行比较。按照此规律一直交换下去,直到第6层共需要6个周期,可以排序获得顺序的8个数字。

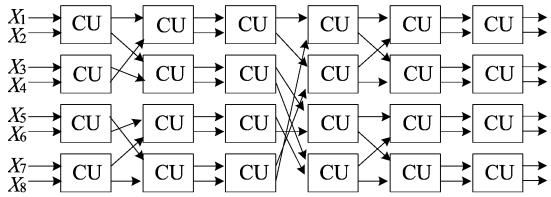

图7 8输入并行双调排序电路图

2.4.2 2轮双调排序结构

在DGCNN[10]中,若最近点的数量k值太大,则会破坏图像块的几何结构。

在k值从1开始递增的过程中:$k=16$时排序结果能够达到89%的准确率;$k=20$时达到最高平均类别准确率;k值超过20时平均类别准确率反而有所下降。因此以近邻点$k=16$作为硬件设计标准。

为了加速排序的速度,针对DGCNN最近邻搜索[11],本文提出并行双调流水排序结构,如图8所示。

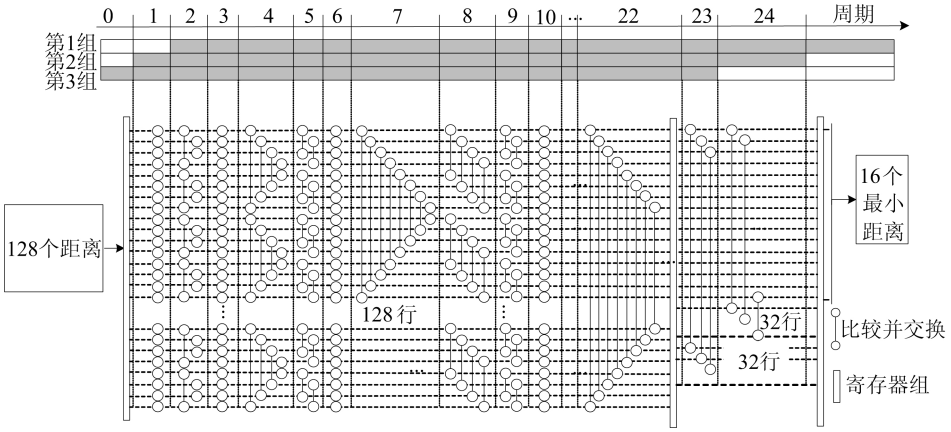

图8 128输入并行双调流水排序结构示意图

该加速排序电路基于双调排序算法设计,采用128个32位数据并行输入、16个22位数据并行输出的并行流水结构,并采用25级寄存器组和1组参数缓存组成。

在构建查询点的KNN图时,由于需要从1 024个距离中找出最小的16个距离,交换方式类似于8输入的双调排序方式,第1层按照间隔为1的顺序交换,第2层按照间隔为1和间隔为2的顺序进行交换,始终保证交换的2个数中较小的数在上,较大的数在下。按照此规律一直交换下去,共需要24个周期,可以排序获得128个距离中较小的16个距离,然后将输出进行缓存。

因为最后仅需要最小的16个距离,所以经过128排序电路时,在最后3个周期会减小面积。由于排序顺序与输入无关,在第2个周期时即可输入第2组距离数据,实现流水线的结构,共使用24级寄存器组,最大流水级数为24级。输入数据为n组时,消耗的周期为$n+24$即可得到排序结果,因此第8组排序完毕的数据消耗周期为32。排序完毕后,相当于取出8组128个距离中最小的16个距离。

每个周期输出128($16\times8=128$)个数,使用计数器等待8个周期后,将这8组距离缓存并拼接,形成128的输入,通过控制器控制时序,将这128个数再次输入128取16的双调排序电路。再次排序完毕后的16个距离即1 024个距离中最小的16个距离。2轮排序方式如图9所示。

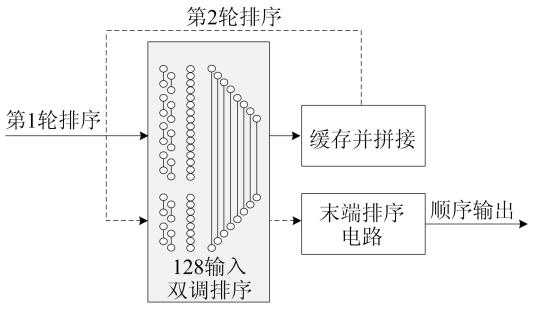

图9 2轮排序示意图

由于得到的最小的16个距离是乱序的,为了获得顺序的16个距离,还需要输入到末端排序电路进行排序。

在末端排序电路中,对乱序的16个距离使用16输入、16输出的并行双调排序电路进行排序,

而这些距离中包含的高 11 位即为某个点的 16 个最近邻点的索引,仅需取出高位数据即可得到顺序的 k 个最近邻点的索引。

在 DGCNN 中,即使数据集输入变为2 048个点,只要是 128 整数倍的点集输入数量,该电路仅需控制输入周期,依旧可以输出 $128n$ 个输入点的最近的 16 个点。

3　实验结果及性能分析

3.1　实验设置

为了衡量本文设计的最近邻搜索硬件加速器的性能,从准确率、资源、频率等方面进行分析。在 Xilinx 的 Kintex-7 xc7k325tffg900-2 FPGA 开发板上对本文设计的最近邻硬件加速器进行实验。

最近邻点的数量对时间和准确率均有影响,软件实验在 Rasberry Pi 4B 上进行,使用 Modelnet40 数据集。该数据集的测试集共有 2 468 个点云,分类为 40 种物品。设置每个点云 1 024 个点,在 batchsize 设置为 8 的情况下进行实验。DGCNN 的各层结构见表 1 所列。

表 1　DGCNN 模型

层数	结构	输入量	输出量
1	K-G-C-B-A-P	3×1 024×16	64×1 024×16
2	K-G-C-B-A-P	64×1 024×16	64×1 024×16
3	K-G-C-B-A-P	64×1 024×16	64×1 024×16
4	K-G-C-B-A-P	64×1 024×16	128×1 024×16

注:K 表示最近邻 KNN;G 表示图像(Graph);C 表示卷积(Conv);B 表示批量归一化(BN);A 表示激活(Activation);P 表示池化(Pooling)。

3.2　排序单元性能测试

为了验证本文双调排序算法的性能,本节参考文献[8]中的冒泡排序搜索方法,以 8 个数据输入、8 个数据输出的冒泡排序为例进行硬件实现。冒泡排序法的思路是:遍历原始数据,从第 1 个数开始,到倒数第 2 个数结束,比较这个数与下一个数的大小,若这个数比下一个数大,则交换这 2 个数。冒泡排序单元与本文 8 输入、8 输出双调排序单元的资源对比见表 2 所列。

本文以面积时间积[12](area time product,ATP)作为性能测试指标,其中面积以 FPGA 中一个 slice(包含若干查找表 LUTs 和触发器 Flip-Flop)作为一个基本单元。ATP 的计算公式为:

$$X_{\mathrm{ATP}} = X_{\mathrm{Area}} T \tag{4}$$

其中,T 为时间。

表 2　不同排序单元的资源及周期对比

排序方法	文献[8]	本文
查找表	307	290
触发器	182	354
功耗/W	0.348	0.424
频率/MHz	333	500
周期	28	7
基本单元	107	138
ATP	2 996	966

由表 2 可知,相较于文献[8]冒泡排序单元,本文设计的双调排序单元的触发器资源有所增加,但搜索周期仅为冒泡排序单元的 25%,且频率提升了 50%,因此搜索时间大幅缩小;综合考虑时间与面积,本文的双调排序单元 ATP 为 966,优于冒泡排序单元的 2 996。综上可知,在并行电路中双调排序是一种优于冒泡排序的方式。

3.3　最近邻搜索硬件加速器性能测试

将本文设计的最近邻搜索加速器与 3 种基于近似搜索的 FPGA 硬件加速器进行对比,结果见表 3 所列。

表 3　加速器性能对比

加速器	文献[7]	文献[13]	文献[14]	本文		
最近邻结构	HNSW	A-KDtree	A-KDtree	Split	Split	Split
FPGA 平台	Xilinx ZU9EG	Xilinx ZCU102	Xilinx XC7Z035	Xilinx ZU9EG	Xilinx ZCU102	Xilinx XC7Z035
查找表		49 805	41 967	64 206	65 131	64 206
触发器		65 024	63 280	103 915	103 863	103 915
准确率/%	<100	<100	91	100	100	100
总耗时/ms	435.0	20.0	20.9	5.7	5.7	5.7
频率/MHz	250	200	200	200	200	200
最近邻点数	30	5	1	16	16	16
总点数	30 000	30 000	32 768	32 768	32 768	32 768
ATP/10^3		249.0	219.0	91.5	92.8	91.5

文献[13]加速器输入点数为 30 000,文献[14]加速器输入的点云图像点数为 32 768,本文加速器输入点数为 32 768,包含 32 张图片,单张图片包含 1 024 个点。以搜索每张图像中所有点的 k 个最近邻点所消耗的时间作为最近邻搜索总时间。从表 3 可以看出,本文最近邻搜索单元查找表消耗是文献[14]的 1.5 倍,触发器消耗是文献[14]的 1.6 倍,但消耗的时间仅为文献[14]的 27%。

由于以上文献均使用近似的最近邻搜索方案,准确率不足 100%,如文献[14]仅有 91%的准确率,这在神经网络中会降低分类准确率的结果,而本文的搜索准确率为 100%;在 ATP 的比较上,本文加速器的 ATP 仅为文献[14]的 42%;相较于文献[14],本文提出的双调排序算法时间复杂度更低,且由于每级排序结构中下一级数据输出仅受上一级寄存器组输入的影响,流水线结构降低了排序时间,而基于小世界图或近似 KD 树算法在点数较少时构建过程花费了过多时间。

4 结 论

本文针对边缘卷积中使用蛮力最近邻搜索算法消耗时间较大的问题,设计了一种最近邻搜索硬件加速器。该加速器使用不完全双调排序方式和 2 轮流水排序结构,并采用了曼哈顿距离计算的方案。实验结果表明,相较于现有最近邻搜索加速器,本文加速器有明显的速度提升。但本文方法仍有提升空间,对于更大的最近邻值,选出最小的距离以及完整的边缘卷积硬件方案设计是下一步研究的方向。

[参 考 文 献]

[1] GOLOVINSKIY A, KIM V G, FUNKHOUSER T. Shape-based recognition of 3D point clouds in urban environments [C]//2009 IEEE 12th International Conference on Computer Vision. [S. l.]: IEEE, 2009: 2154-2161.

[2] WANG Y, SOLOMON J M. Object DGCNN: 3D object detection using dynamic graphs[J]. Advances in Neural Information Processing Systems, 2021, 34: 20745-20758.

[3] ANDONI A, INDYK P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions[J]. Communications of the ACM, 2008, 51(1): 117-122.

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

[5] MALKOV Y A, YASHUNIN D A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 42(4): 824-836.

[6] GREENSPAN M, YURICK M. Approximate KD tree search for efficient ICP[C]//4th International Conference on 3-D Digital Imaging and Modeling. [S. l.]: IEEE, 2003: 442-448.

[7] KOSUGE 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, 2021, 68(4): 3567-3576.

[8] PU Y, PENG J, HUANG L, et al. An efficient KNN algorithm implemented on fpga based heterogeneous computing system using opencl[C]//2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines. [S. l.]: IEEE, 2015: 167-170.

[9] NASSIMI D, SAHNI S. Bitonic sort on a mesh-connected parallel computer[J]. IEEE Transactions on Computers, 1979, 28(1): 2-7.

[10] PINKHAM R, ZENG S, ZHANG Z. QuickNN: memory and performance optimization of k-d tree based nearest neighbor search for 3D point clouds[C]//2020 IEEE International Symposium on High Performance Computer Architecture. [S. l.]: IEEE, 2020: 180-192.

[11] OMOHUNDRO S M. Five balltree construction algorithms [M]. Berkeley: International Computer Science Institute, 1989.

[12] LI X W, JAIN A, MASKELL D, et al. An area-efficient FPGA overlay using DSP block based time-multiplexed functional units [EB/OL]. (2016-06-21). https://doi.org/10.48550/arXiv.1606.06460.

[13] 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.

[14] LI Y M, ZHENG K L, XIAO H. A KNN accelerator based on approximate K-D Tree for ICP[C]//2022 International Conference on Image Processing and Media Computing. [S. l.]: IEEE, 2022: 124-128.

(责任编辑 胡亚敏)