CAMPER: Exploring the Potential of Content Addressable Memory for 3D Point Cloud Efficient Range Search

Jiapei Zheng, Lizhou Wu, Yutong Su, Jingyi Wang, Zhangcheng Huang, Chixiao Chen, Qi Liu
State Key Lab of Integrated Chips and Systems, Frontier Institute of Chip and System, Fudan University
Shanghai, China
cxchen@fudan.edu.cn

ABSTRACT

The use of Light Detection and Ranging (LiDAR) for sensing has continuously improved the precision and performance of autonomous driving. At the same time, the large number of high-precision point clouds generated by LiDAR require real-time processing, and the range search is the key part of the processing pipeline. Content-Addressable Memory (CAM) has proven its efficiency for search tasks on switches and routers, but so far, there is still a lack of exploration on its application in point cloud range search. In this work, we propose CAMPER, a CAM-centered accelerator, aiming to explore the potential of CAM for point cloud range search. We developed a ripple comparison 13T (RC-13T) CAM cell for distance comparison, designed a spatial approximation search algorithm based on Chebyshev distance, and discussed the flexibility and scalability of the architecture. The results show that in the range search task of 64k@64k points, CAMPER achieves a latency of 0.83ms and a power consumption of 114.6mW. Compared with GPU, the throughput is increased by 10.4×; compared with SOTA accelerator, the energy efficiency is increased by about 228×.

CCS CONCEPTS

• **Hardware** → **Application specific integrated circuits**; • **Computer systems organization** → *Special purpose systems.*

KEYWORDS

Point Cloud, Content Addressable Memory, Range Search, LiDAR, Autonomous Driving

1 INTRODUCTION

With the rapid advancements in sensors, algorithms, and hardware, the accuracy and performance of autonomous driving are continually improving [13]. Light Detection And Ranging (LiDAR) is gaining increasing attention, which can generate high-accuracy point cloud data with depth information. In autonomous driving, point cloud data is widely employed in tasks such as object detection, vehicle localization, map construction, and so on.

The neighbor relationships within point clouds are the foundation of many algorithms. Therefore, range search, which involves finding several points adjacent to a given point in the point cloud, is introduced into the point cloud processing pipeline. For example, in point-based backbones [9], the information derived from neighboring points is used as input for feature calculation, and in point cloud denoising [2], it is used to determine whether a point is an outlier. In processing a frame of point cloud data, the range search algorithm is called billions of times. Therefore, a slight increase in the execution time of each call can significantly impact the overall efficiency. Recent studies [3, 14] have shown that the latency of range searches accounts for over 50% of the point cloud processing pipeline. Facing the demand for increasingly high throughput, the performance of range search has become an important issue.

Several advanced neighbor query accelerators, such as QuickNN [8], KD Bonsai [4], and ParallelNN [3], have been proposed. These accelerators aim to address the problem of frequent and irregular memory access. Most of these accelerator designs are based on traditional tree methods. They constructed KD tree or octree and exploited the parallelism of the tree algorithm to accelerate the non-strict range search. Nonetheless, these approaches often grapple with high memory bandwidth requirements, and the need for data movement is difficult to eliminate, hindering further efficiency improvements.

A content-addressable memory (CAM) compares input search data against a table of stored data and returns the address of the matching data [7]. The high speed and low power consumption capabilities of CAM in search tasks have been fully demonstrated on switches and routers [6]. In this work, we propose a point cloud range search accelerator, CAMPER, which uses the characteristics of point cloud data and search patterns, aiming to explore the application potential of CAM in point cloud range search. The main contributions of this paper are summarized as follows:

- We propose CAMPER, a point cloud range search accelerator based on CAM. It can complete a range search of the entire point cloud in two cycles and saves data movement, thereby greatly improving search performance. The CAM-centered architecture eliminates irregular main memory access, simplifies adding and deleting point cloud data, and supports scaling.
- We present a ripple-comparison 13-transistor (RC-13T) CAM cell. Compared to conventional CAM, RC-13T supports range comparison operations. It leverages the characteristic of point cloud range searches where the data is stored within CAM, and search rules are applied as input.
- We employ a comparison based on the Chebyshev distance, which decouples the search dimensionally to take full advantage of the parallel characteristics of CAMPER.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3656248

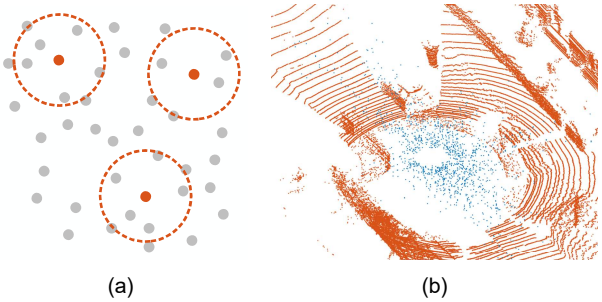


Figure 1: (a) Schematic illustration of point cloud range search. (b) A point cloud image containing snow points.

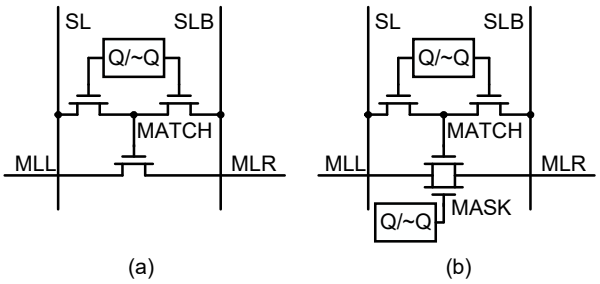


Figure 2: CAM core cells for (a) 9T NAND-type BCAM and (b) 16T NAND-type TCAM [7]. The cells are shown using SRAM-based data-storage cells. The SRAM storage and access transistors account for six of the cell transistors.

We conducted transistor-level simulations and tested CAMPER on several datasets. The results show that in the range search task of 64k@64k points, CAMPER achieved a latency of 0.83ms and power consumption of 114.6mW, which is 10.4× speedup and 228× power efficiency compared to the GPU and SOTA accelerator.

The rest of this paper is organized as follows. Section 2 introduces the background and motivation. Section 3 presents the proposed RC-13T CAM cell and array. Section 4 describes the proposed approximate search algorithm and the architecture of CAMPER. Section 5 provides experiments and results. Finally, the conclusion is given in Section 6.

2 BACKGROUND AND MOTIVATION

2.1 Point Cloud and Neighbor Search

A point cloud is a set of points P, each of which includes its spatial position (x, y, z). Point clouds are typically stored in a 1D list, and their data structures are unordered and devoid of spatial adjacency information. Therefore, to extract spatial information from a point cloud set, range search is introduced into the point cloud processing pipeline. As illustrated in Figure 1(a), for a given point cloud set P, a central point cloud p_c, and a range r, the objective of the range search is to find a neighboring point set N for p_c such that the distance between the points in N and p_c is less than r. There are many point cloud processing algorithms that use range search.

Feature extraction of point cloud backbone network. PointNet++ [9] is a classic point cloud feature extraction backbone network. In these applications, the purpose of range search is to feed the points and their features within a fixed region to the PointNet layer to obtain local region features.

Point cloud denoising/de-snowing/isolation. Due to sensor or environmental weather factors, outliers inevitably exist in point clouds [2]. These outliers could be noise or snow, as shown in Figure 1(b). In such applications, the purpose of the nearest neighbor search is to determine the number of neighbors each point in the point cloud has and, accordingly, decide whether the point is an outlier that needs to be removed.

Point cloud compression. The Moving Picture Experts Group (MPEG) is developing a point cloud compression standard [12] to support the widespread application of point clouds. The purpose of the nearest neighbor search here is to identify whether a point is an isolated point, thereby reducing the number of bits required for encoding.

Point cloud registration. Point cloud registration is the process of aligning different point clouds into a unified coordinate system. Corresponding point pairs [11] are used to estimate the transformation parameters (such as rotation and translation) needed to align the point clouds. In these applications, the purpose of the nearest neighbor search is to find corresponding point pairs between two point clouds.

In processing a single frame of point cloud data, billions of computations and comparisons may be involved. Hence, the performance of range search significantly affects point cloud processing speed.

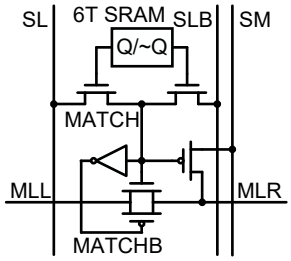
2.2 Content Addressable Memory

CAM can quickly search an input search data from the entire memory and notifies the address of the data in a single clock cycle [7]. The cell structures of the classical NAND-type Binary CAM (BCAM) and Ternary CAM (TCAM) are illustrated in Figure 2. In BCAM, when the data is matched, Match Line Left (MLL) and Match Line Right (MLR) are connected. When all bits are the same, the precharged node will be discharged to the ground by the connected Match Line (ML). For TCAM, because an additional storage unit is added, the "don't care" function can be implemented. Since the Search Line (SL) takes effect simultaneously for each row, CAM can complete the search of all rows in one cycle.

The parallel search of CAM also eliminates the need to read data for comparison, making it faster and more efficient than other hardware and software-based search systems. The main commercial applications of CAM today are for classifying and forwarding Internet Protocol (IP) data packets in network routers [10]. Software Defined Network (SDN) schemes rely on TCAM as the primary hardware of their data paths. Due to the rapid search capability of CAM, it is a good choice for implementing search operations.

2.3 Motivation

There have been various approaches to the neighbor query problem, including the brute force (BF) search, spatial partitioning using 2D grids or 3D voxels, as well as schemes based on KD tree/Octree. The



(a)

(SL, Q)	GE	LE	EQ
(0, 0)	PASS	PASS	PASS
(0, 1)	PU	PD	PU
(1, 0)	PD	PU	PU
(1, 1)	PASS	PASS	PASS

(b)

Figure 3: Proposed RC-13T cell with (a) schematic and (b) truth table.

BF search algorithm must traverse the entire point cloud, resulting in an O(n) time complexity. However, due to its independence from complex data structures, it can easily be paralleled and may perform better on GPUs than more complex algorithms. The time complexity of tree-based searches is O($\log_2$n), but they are essentially sequential and require frequent and irregular memory access. Some accelerators have been proposed to solve these problems. However, in the implementation of these solutions, the demand for irregular memory access and data movement is still challenging to eliminate.

The advantages of CAM have been fully demonstrated in switches and routers. It can perform parallel search operations, providing higher speed and lower power consumption than traditional methods. Some research has been conducted on the range search of CAM [1, 6], but most of these studies are designed based on IP distribution applications, which store search rules and input target data. For the scenario of point cloud search, where storing point cloud data and inputting search rules, there is a lack of relevant research on utilizing the parallel search capability and low latency characteristics of CAM to achieve efficient point cloud range search.

3 RIPPLE COMPARISON 13T CAM CELL FOR RANGE SEARCH

In this section, we introduce the RC-13T CAM Cell designed for point cloud range search tasks. We first introduce the structure and function of the cell (section 3.1), then discuss the working mode of the array (section 3.2).

3.1 Ripple Comparison 13T CAM Cell

Figure 3(a) shows the proposed RC-13T cell for range search. The RC-13T is composed of a 9T NAND-type CAM used to store one bit of point cloud data, an additional 3T inverter and transmission gate circuit, and a 1T pull-up/pull-down circuit. When SL == Q, the MATCH node will be pulled high, the transmission gate will open, M1 will close, and MLR will be connected to MLL. When SL != Q, the MATCH node remains low, the transmission gate closes, M1 turns on, and MLL is connected to SM.

The truth table of RC-13T for comparison is shown in Figure 3(b). We define that MLR being pulled down represents a match, and being pulled up represents a mismatch. When the data of the search pattern is equal to the data stored in CAM, that is, (SL, Q) == (0,0) or (1,1), the match status of the entire word cannot be asserted, and this cell behaves as a pass. When SL is different from Q, that is, (SL, Q) == (0,1) or (1,0), the match status of the entire word can be asserted, and the single cell will behave as a match or mismatch, which will be determined by the mode of comparison. RC-13T supports three comparison modes: greater than or equal to input (GE), less than or equal to input (LE), and equal to input (EQ). When the mode is GE, MLR is pulled up when SL==0, and MLR is pulled down when SL==1. When the mode is LE, MLR is pulled down when SL==0, and MLR is pulled up when SL==1. When the mode is EQ, MLR is always pulled up. According to this rule, SM can just be assigned by SL and mode, as shown in Eq. 1.

$$SM = GE \cdot (\neg SL) + LE \cdot SL + EQ \cdot 1 \tag{1}$$

Mode	LE	LE	GE	GE
Search Data (SL)	1 1 0 0 (3)	1 1 0 0 (3)	1 1 0 0 (3)	1 1 0 0 (3)
(SM)	0 0 1 1	0 0 1 1	1 1 0 0	1 1 0 0
Point Data (Q)	0 0 1 0 (4)	1 0 0 0 (1)	0 0 1 0 (4)	1 1 0 0 (3)
Cell Status	PD/PD/PU/PS	PS/PD/PS/PS	PU/PU/PD/PS	PS/PS/PS/PS
Final Result	PU: mismatch	PD: match	PD: match	PD: match

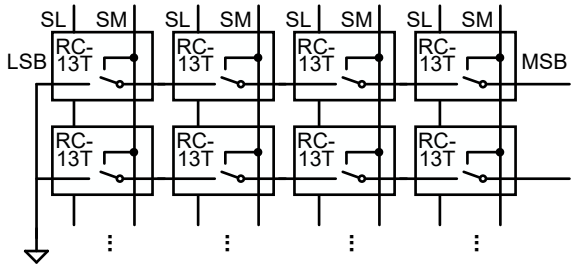


Figure 4: (a) Example of a 4-bit range search. (b) Array structure of RC-13T.

3.2 CAM Array

Figure 4 illustrates a case analysis of an RC-13T array. The comparison process proceeds from the most significant bit (MSB) to the least significant bit (LSB). When the MSB cannot be determined, the transmission gate is turned on. After the final result is determined, the state propagates from left to right, like a ripple effect. Figure 4(a) shows four different cases where the results are determined at different positions and transmitted to the rightmost end through cells with the state pass (PS). The MLL of the LSB is connected to the ground and will be pulled down when all the cells are the same.

When storing multiple point cloud data, it is straightforward to form an array by combining multiple RC-13T cells. Figure 4(b) illustrates the structure of such an array. Each row of the CAM stores a single point cloud data, and the search rules are applied in parallel to each row through SL/SLB and SM. Since SM is only dependent on the search data (SL) and the search mode and is independent of the stored data, SM can be driven outside the cell, which makes it easy to implement RC-13T. This is the advantage of point cloud range search, as it stores the data and inputs search rules.

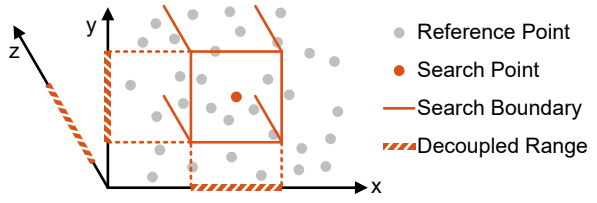


Figure 5: Approximate spatial search algorithm based on the Chebyshev distance.

4 RANGE SEARCH ALGORITHM AND CAMPER ARCHITECTURE

In order to maximize the potential of the proposed RC-13T, we first optimize the distance calculation operation (section 4.1) and then discuss the overall architecture of CAMPER (section 4.2).

4.1 Range Search Algorithm

In most algorithms, the distance between two points is typically defined as the Euclidean distance $L_2 = [\sum (x_i - y_i)^2]^{1/2}$. However, the computation of Euclidean distance depends on the information on each dimension, which would lead to difficulty in mapping to the design of CAMPER. Furthermore, Euclidean distance computation requires a square operation, which adds extra computational overhead.

In this paper, the Chebyshev distance $L_\infty = [\sum (x_i - y_i)^\infty]^{1/\infty}$ is used as the definition of distance to perform the range search. Compared to the L_2 distance, the error range of L_k distance is: $1 \leqslant e \leqslant d^{1/2-1/k}$. When demension $d = 3$, and $k = \infty$, the error range is $[1{:}\sqrt{3}]$. The Chebyshev distance can be rewritten as:

$$(\sum_{i=1}^{n} (x_i - y_i)^\infty)^{1/\infty} = \max_{1 \leqslant i \leqslant n} (|x_i - y_i|) \leqslant R \tag{2}$$

$$\begin{cases} |x_1 - y_1| & \leqslant R \\ |x_2 - y_2| & \leqslant R \\ & \cdots \\ |x_n - y_n| & \leqslant R \end{cases} \tag{3}$$

Hence, the Chebyshev distance separates the distance calculation among dimensions and can be easily extended to high-dimensional spaces, as shown in Figure 5. In addition, it only requires comparison operation, simplifying hardware implementation.

For a three-dimensional point cloud, as shown in Algorithm 1, the range search comparison will be decoupled into three dimensions. Range search on each dimension can also be decoupled into upper-bound search and lower-bound search. There is no interdependence between dimensions, which implies that the calculation will be completely restricted to a single dimension, and it also means that the calculation of each dimension can be fully executed in parallel.

4.2 CAMPER Architecture

The overall architecture of CAMPER, featuring the RC-13T cell array as its core module, is shown in Figure 6. The inputs of the CAM array include point cloud data and search patterns, while the outputs include point cloud information and the addresses of matching point cloud data.

Algorithm 1 Range search based on the Chebyshev distance

Input: Point cloud set $\mathbb{P}$, range search pattern $\mathbb{S}$
Output: Point flag list $\mathbb{F}$

```
Decouple range search pattern 𝕊 = (x_lb, x_ub, y_lb, y_ub, z_lb, z_ub)
Initial flag 𝔽, F_i = True|i = 0, 1..n × N
for (search data S, search mode M) ∈ 𝕊 do
    for all P ∈ ℙ do
        Determine whether it meets (S, M), save to F_i^t
        Update flag list F_i = F_i & F_i^t
    end for
end for
return Point flag list 𝔽
```

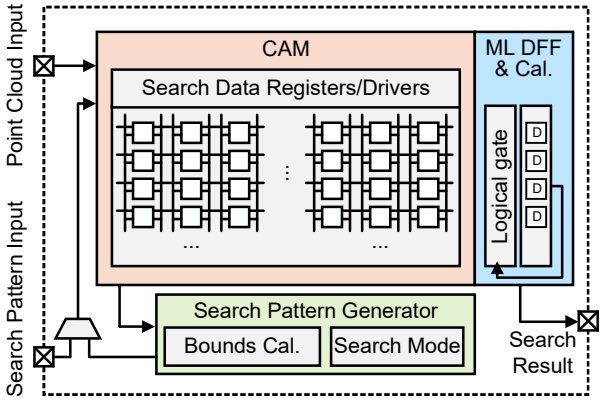


Figure 6: Overall architecture of CAMPER.

In preparation for the search, the reference point cloud data needs to be moved from DRAM to CAM. CAM enables parallel searching and comparison of entire point clouds, eliminating the need for a specific storage order. Thus, the point cloud data can be sequentially read from DRAM and loaded into CAMPER.

Once all reference point clouds are successfully loaded, the search can start. The search patterns can be externally input or generated on the chip by reading the stored point cloud and calculating. Leveraging the parallel search capability of the RC-13T cell, CAMPER can complete the search in two cycles. In the first cycle, the inputs x_{ub}, y_{ub}, z_{ub} are given, and CAMPER outputs the addresses of point cloud data that satisfy these conditions simultaneously. In the second cycle, the inputs x_{lb}, y_{lb}, z_{lb} are given, and CAMPER outputs the addresses of point cloud data that satisfy these conditions simultaneously. CAMPER performs an "AND" operation on the addresses, outputting the addresses that satisfy both the upper- and lower-range search conditions.

After completing the search, these outputs will be used for counting, flagging, or other operations according to the specific requirements of the algorithm.

The capacity of the CAM array needs to be consistent with the number of point clouds in the reference. In the typical range search

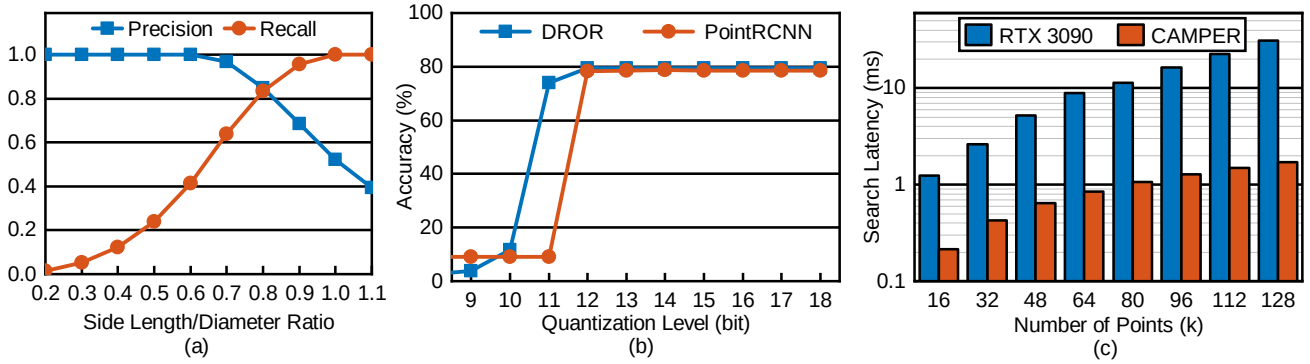


Figure 7: (a) Variation of precision and recall with the change in the side length/diameter ratio. (b) Variation of the accuracy of DROR and PointRCNN with the change in search rules quantization level. (c) Variation of search latency of GPU and CAMPER with the change in number of points per frame.

task of 64k@64k, the required capacity for CAM is 64k entries, with a size of each word size of 18b × 3. The CAMPER architecture and search scheme allow for the flexible updating, addition, and deletion of point cloud data, even when only one point is added or deleted. Since the search scheme does not require any order of entries, it is as simple as removing the corresponding entries from the CAM or adding the entries of the corresponding point cloud in an empty CAM row. In contrast, adding and removing points in a KD tree is a relatively complex process and often requires rebuilding the entire tree structure to ensure balance.

5 EXPERIMENTS AND RESULTS

5.1 Experimental Setup

Models and Datasets. We conducted object detection and snow removal algorithm tests on the KITTI dataset [5] and snowyKITTI dataset [2] to evaluate the impact of Chebyshev distance on accuracy. The algorithms used are the DROR [2] and PointRCNN [13], with the original parameter settings maintained. The range of the point clouds in the KITTI dataset is about 80 meters, so we quantized the geometric information (X, Y, Z) to 1mm and stored it in three 18-bit fixed-point numbers, totaling 54 bits.

Platform and Implementation. For the GPU implementation of BF, well-designed CUDA is used to allocate tasks to each thread, compiled under the environment of PyTorch 1.10.2 and CUDA 11.3, and ran on RTX 3090. CAMPER was implemented under the 28nm process. The custom CAM array was simulated and tested using Spectre, and the digital part was synthesized using Design Compiler.

5.2 Results and Analysis

We first perform an approximate analysis of the range search task mathematically, when the distance of a point > L_2, and < L_∞, it is defined as false positive, and distance < L_2, and > L_∞, is defined as false negative. The relationship between accuracy and recall is shown in Figure 7(a). When L=0.81D, both accuracy and recall can reach 0.84. Therefore, the algorithm does not need to retrain parameters or set hyperparameters. It only needs to use L=0.81D as the range search conversion rule and for subsequent tests.

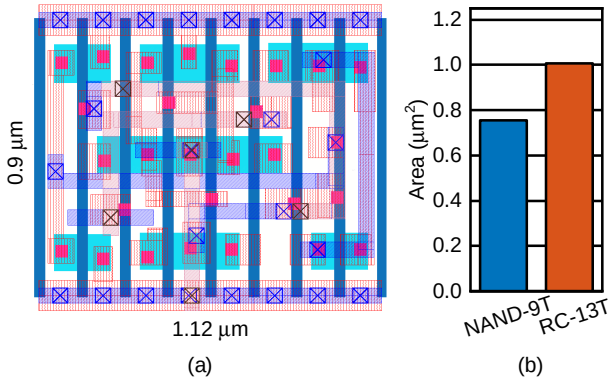


Figure 8: (a) Layout of RC-13T cell. (b) Increase in area relative to traditional NAND-type CAM.

As the quantization level decreases, the accuracy of the search will also decrease, but most downstream applications can tolerate this situation. Figure 7(b) shows the impact of range search with different quantization levels on downstream algorithms. It can be seen that when the accuracy is above 12 bits, the loss of accuracy can be controlled within 0.4%. It is worth noting that improving accuracy does not come at the expense of latency because the delay of CAMPER is relatively fixed.

Figure 7(c) represents the detailed results of the latency of the GPU implementation and CAMPER under varying point number conditions. On the GPU platform, search tasks can be allocated to tens of thousands of computing cores, but as the number of points grows, compute cores become saturated, and the search time also begins to increase with the square of the number of points. By storing point cloud data in RC-13T, CAMPER can achieve almost linear latency growth. In a range search task of 64k search point cloud at 64k reference point cloud, the GPU latency is 8.9ms, while the latency of CAMPER is 0.83ms. Compared with the GPU implementation, CAMPER can achieve approximately 10.4× acceleration.

We conducted a more detailed evaluation of RC-13T. Figure 8 shows the layout of RC-13T. The layout is drawn referring to the

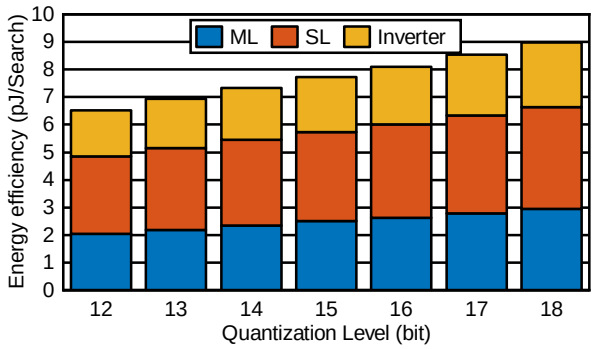


Figure 9: Energy efficiency of search operation with the change in search rules quantization level.

Table 1: Comparison Table

	GPU	QuickNN [8]	ParallelNN [3]	**This Work**
Data structure	Brute-force	KD-tree	Octree	**Brute-force**
Test Case	64k@64k	30k@30k	50k@50k	**64k@64k**
Approximation	100%	75%	85%	**84%**
Frame rate	112.4fps	110.1fps	1250.0fps	**1171.2fps**
Power	255W	4.7W	28.4W	**74.1mW(CAM) 114.6mW(Sys.)**
Speedup	1×	0.98×	11.1×	**10.4×**
Efficiency	0.019×	1×	1.88×	**428.8×**

standard cell library, which has an area of $1.008\mu m^2$. Figure 9 shows the relationship between the power consumption of each part of RC-13T and the number of search bits. We only change the number of bits in each search pattern and maintain the quantization level of the point cloud stored in CAMPER. The overall search power consumption increases linearly with the number of SL/ML. At the quantization level of 15b, the proportions of ML and SL are 32.3% and 41.9%, respectively. The inverter added to build the transmission gate also consumes some power, reaching 25.8%.

Table 1 compares CAMPER with GPU implementation and several existing accelerators. CAMPER operates at a clock frequency of 300MHz and can handle 64k@64k point cloud queries. The capacity of RC-13T CAM is 432KB. Thanks to the parallelism of CAM, the CAMPER architecture is naturally suitable for unordered point cloud data, eliminating the need for external random access, does not require additional assumptions about the shape and uniformity of the point cloud, and can be easily expanded. In addition, the near-memory architecture centered on CAM means that data can move as little as possible in the search, and the core power consumption is only 74.2mW. Compared with the SOTA accelerator, the energy efficiency of the system can achieve an increase of about 228×.

6 CONCLUSION

In this paper, we explored the potential of CAM for 3D point cloud range search and proposed an accelerator for point cloud range search, namely CAMPER. To support search operations more efficiently and utilize the characteristics of point cloud search patterns, we propose RC-13T, which implements ripple comparison by adding a transmission gate and a PU/PD PMOS. In addition, we adopted a comparison based on Chebyshev distance and decoupled it in dimensions to utilize the parallel characteristics of the CAMPER architecture fully. Finally, we conducted simulation and testing at the transistor level. The results show that compared to the GPU and the state-of-the-art accelerator, CAMPER can improve throughput and energy efficiency by 10.4× and 228×, respectively. The CAM-centered CAMPER architecture supports point-level updates and modifications, and can be easily scaled to support a larger number of points.

ACKNOWLEDGMENTS

This work was supported by the National Science and Technology Major Project (No. 2021ZD0114400) and the National Natural Science Foundation of China (No. 62322404).

REFERENCES

[1] Anat Bremler-Barr, Yotam Harchol, David Hay, and Yacov Hel-Or. 2016. Encoding short ranges in TCAM without expansion: Efficient algorithm and applications. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*. 35–46.

[2] Nicholas Charron, Stephen Phillips, and Steven L Waslander. 2018. De-noising of lidar point clouds corrupted by snowfall. In *2018 15th Conference on Computer and Robot Vision (CRV)*. IEEE, 254–261.

[3] Faquan Chen, Rendong Ying, Jianwei Xue, Fei Wen, and Peilin Liu. 2023. ParallelNN: A Parallel Octree-based Nearest Neighbor Search Accelerator for 3D Point Clouds. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 403–414.

[4] Pedro H. E. Becker, José-María Arnau, and Antonio González. 2023. KD Bonsai: ISA-Extensions to Compress KD Trees for Autonomous Driving Tasks. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)*. 1–13.

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3354–3361.

[6] Young-Deok Kim, Hyun-Seok Ahn, Joon-Young Park, Suhwan Kim, and Deog-Kyoon Jeong. 2006. A storage-and power-efficient range-matching TCAM for packet classification. In *2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers*. IEEE, 587–596.

[7] Kostas Pagiamtzis and Ali Sheikholeslami. 2006. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE journal of solid-state circuits* 41, 3 (2006), 712–727.

[8] Reid Pinkham, Shuqing Zeng, and Zhengya Zhang. 2020. Quicknn: Memory and performance optimization of kd tree based nearest neighbor search for 3d point clouds. In *2020 IEEE International symposium on high performance computer architecture (HPCA)*. IEEE, 180–192.

[9] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* 30 (2017).

[10] VC Ravikumar, Rabi N Mahapatra, and Laxmi Narayan Bhuyan. 2005. EaseCAM: An energy and storage efficient TCAM-based router architecture for IP lookup. *IEEE Trans. Comput.* 54, 5 (2005), 521–533.

[11] Szymon Rusinkiewicz and Marc Levoy. 2001. Efficient variants of the ICP algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 145–152.

[12] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. 2018. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)* 9, 1 (2018), 133–148.

[13] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 770–779.

[14] Tiancheng Xu, Boyuan Tian, and Yuhao Zhu. 2019. Tigris: Architecture and algorithms for 3d perception in point clouds. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 629–642.