DAWN: Accelerating Point Cloud Object Detection via Object-Aware Partitioning and 3D Similarity-Based Filtering

Dongdong Tang[1], Yu Mao[1]*,Weilan Wang[1], Nan Guan[1], Tei-Wei Kuo[2], and Chun Jason Xue[3]
[1]City University of Hong Kong, [2]National Taiwan University, [3]MBZUAI

***Abstract*—As a fundamental perception task, 3D point cloud detection has become essential for applications in autonomous driving and robotics. However, point cloud detection faces significant challenges of high computational cost due to complex point processing operations. To address this issue, we propose DAWN, an acceleration framework for point cloud object detection that identifies partial similarities between adjacent frames and reduces computational cost by filtering redundant points. DAWN uses object-aware partitioning that defines boundaries based on previous detection results for localized similarity analysis. Additionally, it applies axis-sorted point selection to refine partitioning for point clouds with non-uniform distribution. An efficient 3D similarity algorithm then filters redundant points to reduce computational load. DAWN enables flexible latency-accuracy trade-offs by tuning point filtering ratios. Experimental results show that DAWN achieves a $1.59\times$ average speedup and up to $1.70\times$ on state-of-the-art detection networks by filtering more than 50% of points on average, with negligible impact on accuracy.**

I. Introduction

With the rapid advancement of 3D sensing technology, point cloud data has become increasingly popular for its detailed 3D geometric information [1], [2]. A point cloud is a set of 3D points generated by sensing devices such as LiDAR (Light Detection and Ranging) sensors [3], [4]. Point cloud data processing requires complex computations, posing significant challenges for real-time autonomous driving and robotics implemented on embedded devices. To address this, we propose DAWN, an acceleration framework that reduces computation by filtering redundant points, decreasing processing latency while preserving accuracy.

Point cloud detection can be broadly categorized into voxel-based and point-based methods. Voxel-based detection methods convert the point cloud to 3D grids for efficient 3D CNN processing [5], [6] but suffer from loss of 3D spatial information during the voxelization process. Point-based detection methods process raw points directly, achieving higher accuracy by preserving fine-grained patterns and local structures. However, point-based detection methods suffer from significant processing latency due to substantial computational costs. Point-voxel based detection networks process both voxel-based and point-based features for detection [10], [26]. Some of these methods achieve high accuracy but face the same latency challenges as point-based methods due to intensive point processing.

*Corresponding author

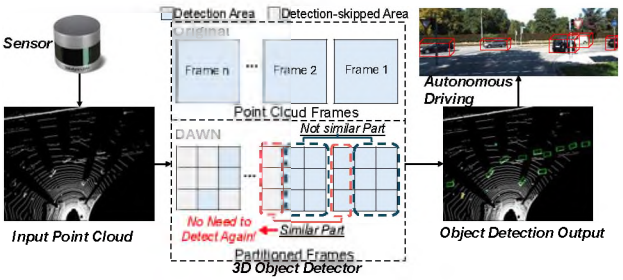


Fig. 1: Original point cloud detection processes entire frames. DAWN accelerates detection by identifying partial similarities and filtering redundant points.

This paper proposes DAWN, an acceleration framework that reduces the inference latency of point cloud detection methods while preserving accuracy. DAWN is based on the observation that points in similar areas across adjacent frames typically produce similar detection outcomes. As shown in Figure 1, DAWN partitions frames and identifies areas that are similar to their counterparts in the previous frame through localized similarity analysis. DAWN skips detection for these areas by reusing previous results and only processes those with significant changes. Unlike traditional methods that perform detection on entire frames, DAWN's selective processing strategy reduces points for computation. However, implementing this strategy presents three key challenges. First, DAWN requires frame partitioning for localized similarity analysis, but defining partitioning boundaries is non-trivial. Naive solutions without considering object positions can fragment objects across partitions, leading to detection errors. Second, while DAWN needs fine-grained partitions for partial similarity analysis, the non-uniform point cloud distribution poses challenges. The partitioning result may be skewed, where some partitions are empty while others become overcrowded with points. Third, DAWN should precisely identify truly redundant points that can be safely skipped. Otherwise, the speedup would come at the cost of significant accuracy loss.

To address these challenges, we propose three key design elements. First, we propose a dynamic object-aware partitioning that defines boundaries based on previous detection results to avoid object fragmentation. Second, we present the axis-sorted point selection to refine the partitioning, enabling localized frame comparisons for better partial similarity analysis. Third,

we introduce a GPU-efficient 3D similarity algorithm that accurately filters redundant points through parallel computation of point distances. Based on the experiment results, DAWN achieves a 1.59× average speedup and up to 1.70× on state-of-the-art detection networks, with minimal accuracy loss. Additionally, DAWN enables flexible latency-accuracy trade-offs to meet different system constraints and application requirements. Our work makes the following contributions:

- We propose a dynamic object-aware partitioning method that maintains object integrity by defining boundaries based on previous detection results.
- We introduce axis-sorted point selection to further refine the partitioning for more effective partial similarity analysis between adjacent frames.
- We present an efficient 3D similarity algorithm to reduce detection workload by accurately filtering the redundant points.
- We implement DAWN, a framework providing flexible latency-accuracy trade-offs, on three state-of-the-art detection networks. Experiment results show that DAWN accelerates these networks by 1.59× on average and up to 1.70× by filtering more than 50% of points on average, with negligible impact on accuracy.

II. Background and Related Work

A. Point Cloud Object Detection

A point cloud is a set of sparse, unstructured 3D data points defined by their (X, Y, Z) coordinates. Point cloud data is widely used in various fields, including autonomous driving, robotics, and remote sensing [1]. Among point cloud applications, object detection is a crucial perception task. It identifies and locates objects for environmental perception. Point cloud detection methods can be categorized into point-based and voxel-based detection networks. Voxel-based detection networks transform point clouds into regular 3D grids for efficient 3D Convolutional Neural Networks (CNNs) processing [5]–[7]. However, this voxelization process can lose fine-grained spatial information as points within each voxel are summarized into single feature representations. Point-based detection networks directly process raw points and preserve more spatial information, offering higher detection accuracy at the cost of longer inference time [8], [9]. Point-voxel based detection networks process both voxel-based and point-based features for detection [10], [11]. While some methods achieve low processing latency, others targeting high accuracy face computational challenges due to intensive point processing.

B. Acceleration of Point Cloud Detection

Numerous acceleration methods have been developed to enhance the performance of point cloud detection [12], [13], [15], [27]–[30], [32]–[39]. Mesorasi [16] achieves faster and more energy-efficient detection on GPU by applying delayed aggregation to the detection network. Point-X [17] proposes a spatial locality-aware accelerator, in which a speculative breadth-first search (BFS) graph traversal method is applied to extract the locality of input. PointAcc [18] presents a versatile design supporting efficient nonzero mapping operations, effectively addressing input sparsity and improving processing speed. Mars [14] is a memory access reduced and scalable accelerator for point-based and voxel-based 3D point cloud neural networks.

III. Motivation

Our preliminary study evaluates the performance limitations of point-based detection. Using the KITTI dataset [19], we compare different detection methods on a desktop platform equipped with an Intel(R) Core(TM) i7-9700K CPU@3.6GHz, 64GB DDR4 RAM and NVIDIA GTX 2080Ti. As shown in Figure 2, point-based and point-voxel based detection networks process fewer than 10 frames per second. The processing speed is insufficient for real-time operation because most LiDAR sensors used in autonomous driving generate point cloud data at 10 Hz [19]–[21]. Voxel-based detection networks achieve faster processing but suffer from lower accuracy due to voxelization's spatial information loss.

Figure 3 identifies point processing as the primary computational bottleneck in both point-based and point-voxel based detection networks. Point processing steps are operations on raw points that include feature extraction through set abstraction and subsequent processing. Point-based and point-voxel based detection networks, sharing the point processing steps, suffer from the same latency challenges. To address this challenge, we propose DAWN, an acceleration framework that speeds up the detection while maintaining accuracy by filtering redundant points based on spatio-temporal frame similarity.

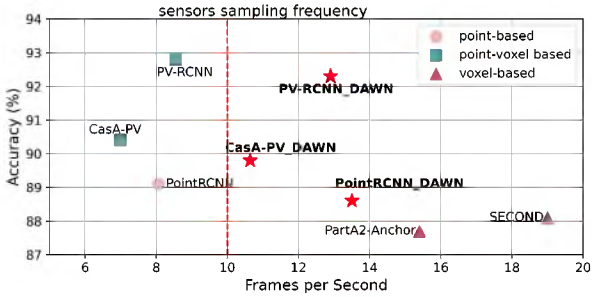


Fig. 2: Detection speed of different methods. The point-based and point-voxel based detection networks process fewer than 10 frames per second. The voxel-based detection networks are faster but have a larger accuracy loss.

IV. Methodology

A. System Overview

DAWN focuses on continuous point cloud frames. As Figure 5 shows, objects can move or remain stationary between frames. Consequently, regions that remain similar across adjacent frames typically produce similar detection results. Based on this observation, DAWN leverages these partial similarities to reduce computation by skipping redundant detection. Figure 4 shows the three key components of DAWN. First,

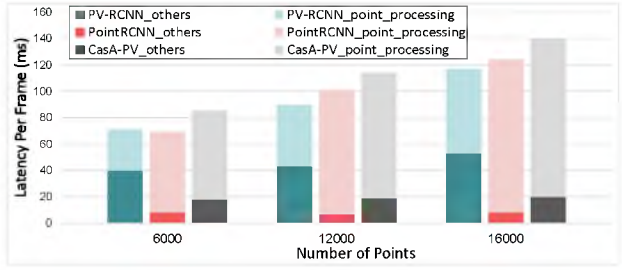


Fig. 3: Latency breakdown. The breakdown reveals that point processing is the main computational bottleneck.

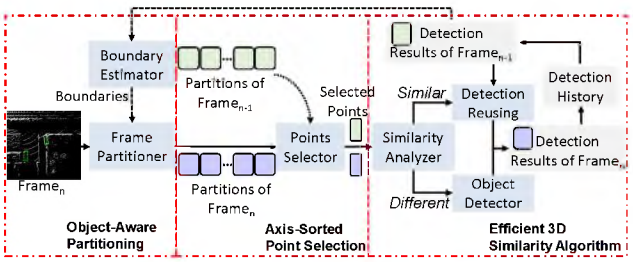


Fig. 4: System Overview: DAWN initially partitions a frame using object-aware partitioning (Section IV-C). It selects points using axis-sorted point selection (Section IV-D) and compares them to the corresponding part in the previous frame. The detector processes points sufficiently different from their previous frame counterparts, filtering redundancies through similarity analysis (Section IV-E).

object-aware partitioning divides each frame using boundaries estimated from previous detection results to avoid object fragmentation. Second, axis-sorted point selection selects points at fine-grained granularity for better similarity analysis. The granularity level defines how many points are processed for the similarity comparison to decide if detection can be skipped. Point selection enables localized comparison between the current (Frame$_n$) and the previous frame (Frame$_{n-1}$) using an efficient 3D similarity algorithm. The detector processes points that differ significantly from their counterparts in previous frames while reusing detection results for sufficiently similar regions. The detection results of the current frame are then stored in the detection history for subsequent frame processing.

B. Design Challenges

DAWN is designed to speed up point cloud detection while maintaining accuracy. However, the design faces several challenges.

Challenge 1: Preserving Object Integrity Maintaining object integrity in frame partitioning is crucial for accurate detection. Naive frame partitioning using fixed boundaries introduces problems. When partition boundaries intersect with objects without considering their positions, the objects' constituent points become fragmented across different partitions. The fragmented point sets cannot provide complete spatial information for accurate detection, resulting in detection errors.

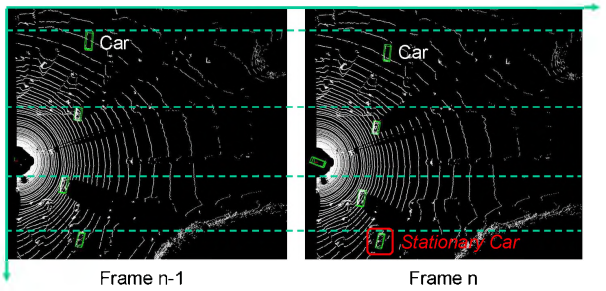


Fig. 5: Partial similarity between adjacent frames. Objects in a frame may have varying degrees of motion.

Challenge 2: Achieving Fine-grained Partitioning Granularity Sufficiently fine-grained partitioning granularity is crucial for utilizing spatial similarity effectively. The initial partitioning may not achieve optimal granularity due to non-uniform point cloud distribution. The partitioning methods may result in skewed point distributions, with some partitions containing no points while others are excessively dense. As a result, the partial similarity cannot be well explored.

Challenge 3: Filtering by Similarity The challenge of similarity-based filtering lies in accurately identifying partial similarity. The method should correctly identify points that are sufficiently different from their counterparts in the previous frame. Otherwise, the speedup would come at the cost of significant accuracy degradation. Additionally, the similarity calculation should be fast to avoid introducing significant overhead.

C. Object-Aware Partitioning

Objects in point cloud frames show varying motion: some move significantly, others remain stationary. Figure 5 illustrates these motion variations between adjacent frames. Stationary regions across frames are highly similar and produce similar detection results, while regions with significant movement require new detection. To identify these partial similarities between frames, developing a partitioning method for localized comparison is essential.

To minimize detection errors, it's essential to maintain object integrity when partitioning frames. A naive approach involves partitioning the frames with fixed boundaries. However, this can lead to the problem of fragmenting the objects across the boundaries. Object detection algorithms struggle to recognize objects from these fragmented sets due to the lack of complete spatial information needed for accurate detection.

To tackle this challenge, we propose object-aware partitioning to split the frame while preserving object integrity. This partitioning defines boundaries based on partial similarities between adjacent frames. This approach is based on the principle that the objects maintain spatial continuity across adjacent frames. The position of objects changes gradually between adjacent frames, rather than exhibiting unexpected movements to distant locations.

Based on this observation, we utilize the predictions from the previous frame, denoted as $Pred = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ to determine the partitioning of the current frame. The predictions of previous frames are denoted as $[x, y, z, w, l, h]$, where $[x, y, z]$ is the position and $[w, l, h]$ is the dimension of objects. We then define a series of partitioning planes $a_i x + b_i y + c_i z + d_i = 0$ that are positioned between these objects without fragmenting the objects. With the partitioning planes established, we determine how to partition the frame and assign each point to its appropriate partition. Each point (x_j, y_j, z_j) in the frame is compared with the partitioning planes to determine its relative position. This comparison is performed by substituting the coordinates of the point (x_j, y_j, z_j) into the equation of each plane. Each comparison will return a result r_i, which equals 1 if the point lies on or above the plane ($a_i x_j + b_i y_j + c_i z_j + d_i \geq 0$), and 0 if the point lies below the plane ($a_i x_j + b_i y_j + c_i z_j + d_i < 0$).

Results from comparing all partitioning planes determine the appropriate partition for points. The point with comparison results $r_1, r_2, r_3, r_4, \ldots, r_n$ will be assigned to partition k, where $k = (r_1 r_2 r_3 r_4 \ldots r_n)_2$. Here, $(\ldots)_2$ denotes the binary representation of the partition index. Figure 6 compares fixed boundary partitioning and dynamic object-aware partitioning. Fixed boundaries fragment objects across different partitions, resulting in detection errors. Conversely, object-aware partitioning maintains object integrity within partitions, shown by uniform color in bounding boxes. By maintaining object integrity, object-aware partitioning better preserves detection accuracy.

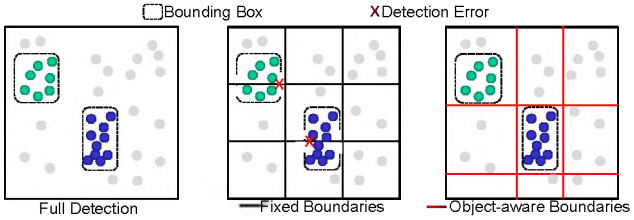


Fig. 6: Object-aware partitioning. The partitioning boundaries are defined based on detection results from previous frames.

D. Axis-Sorted Point Selection

The object-aware partitioning method effectively splits the frame while maintaining object integrity. However, the partitioning may not achieve fine-grained granularity due to non-uniform point cloud distribution. If partitions lack ideal granularity, exploring partial similarity becomes challenging. Directly applying similarity calculations on these partitions does not work well.

To address this challenge, we propose axis-sorted point selection. The point selection step is added before the similarity calculation to select the points at a fine-grained granularity level. Since DAWN leverages partial similarities between adjacent frames within specific regions, our point selection method should ensure that the selected points form cohesive regions rather than scattering across the space. To maintain the spatial continuity of selected points, we implement a two-step sorting process before selection.

The detailed steps of the point selection are illustrated in Algorithm 1. This step selects points from the current frame and the previous frame at an optimal granularity level. First, we sort point sets from both current and previous frames along the x-axis and divide them into groups. To avoid fragmenting objects across different groups, we determine ranges along the x-axis that are likely to contain objects based on previous detection results. The first n_{g_1} points in both point sets are assigned with $id.x = 0$, and the next n_{g_2} points in both point sets are assigned with $id.x = 1$. This assignment continues with each subsequent group of n_{g_i} points receiving an incrementally higher $id.x$ value. The group sizes n_{g_i} are adjusted to avoid objects being fragmented across different groups. The same sorting and assignment steps are then repeated in the y-direction. After the sorting and assignment, each point has a pair of indices $(id.x, id.y)$. The final selection index is computed as $id = id.x + id.y * u_y$, where u_y is the number of unique elements in $id.y$. The points with the same id are selected together, ensuring that the selected points form cohesive regions. Figure 7 illustrates the point selection process with a simplified example, where $id.x$ and $id.y$ range from 0 to 2. We skip selection along the z-axis since objects are primarily distributed on the road plane, having much smaller differences in height than in their x and y positions.

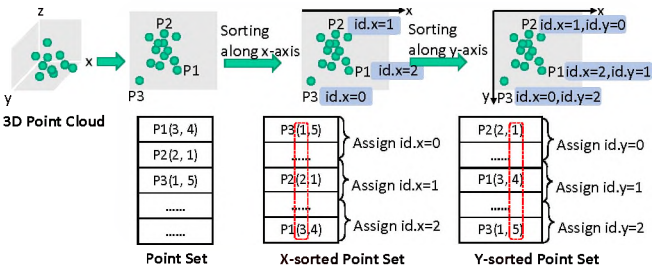


Fig. 7: Axis-sorted point selection selects points at fine-grained granularity to better explore the partial similarity.

E. Efficient 3D Similarity Algorithm

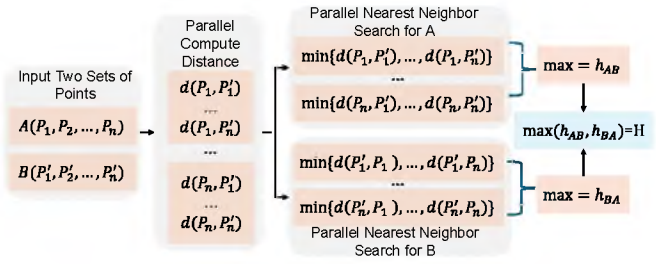


Fig. 8: The 3D similarity calculation provides efficient similarity analysis to identify points to be filtered out.

To accelerate detection while maintaining accuracy, DAWN identifies partial similarities between adjacent frames and

Algorithm 1 Axis-sorted Point Selection.

Input: N Points P_1, P_2 from current and previous frames.
Output: k division of points at optimal granularity levels.

Sort points P_1, P_2 in x direction;
Object ranges $\mathbb{X}_i$=(x^i_{min}, x^i_{max}) from previous frame;
$j = 0$;
while $n_x \leq N$ **do**
  Select $P_2[n_x : n_x + n_{g_i}]$ with range $\mathbb{X}_s$;
  if $\mathbb{X}_i \cap \mathbb{X}_s \neq \emptyset$ **then**
    Adjust n_{g_i} until $\mathbb{X}_i \in \mathbb{X}_s$;
  end if
  points in $P_1[n_x : n_x + n_{g_i}]$ have $id.x = j$;
  points in $P_2[n_x : n_x + n_{g_i}]$ have $id.x = j$;
  $n_x = n_x + n_{g_i}$,$j = j + 1$;
end while
Repeat the division in y direction and assign $id.y$;
Points are indexed with division $id = id.x + id.y * u_y$, u_y is count of unique $id.y$ values;

accurately filters redundant points. We propose an efficient 3D similarity algorithm using the Hausdorff distance to identify points requiring detection. The Hausdorff distance is widely used to measure the degree of similarity between two sets of points. The original Hausdorff algorithm is implemented as follows. Let A and B be two non-empty subsets in 3D space. $h(A, B)$ is the directed Hausdorff distance from A to B. First, for each element in A, it finds its nearest neighbour in B and the corresponding distance. Next, it finds the maximum value among these minimum distances to obtain $h(A, B)$. Then, repeat the same process by swapping sets A and B to get $h(B, A)$. Finally, the Hausdorff distance $H(A, B)$ is determined by taking the larger value between $h(A, B)$ and $h(B, A)$.

Our method proposes an efficient implementation of the Hausdorff algorithm on GPU. In our design, the distance calculation of the Hausdorff algorithm is implemented in parallel on GPU. The implementation is illustrated in Figure 8. The process begins with pairwise distance computation, where the distance between any pair of points P_i and P'_j is calculated in parallel. Then, the nearest neighbour search can be implemented in parallel for each point, obtaining distance from each point in one set to its closest point in the other set. The maximum distance of P_i in A to its nearest neighbour in B is the directed Hausdorff distance from A to B $h(A, B)$. Similarly, the maximum distance of P'_i in B to its nearest neighbour in A is the directed Hausdorff distance from B to A $h(B, A)$. The Hausdorff distance $H(A, B)$ is the maximum of $h(A, B)$ and $h(B, A)$.

DAWN uses distance thresholds in localized similarity analysis to identify redundant points. Partitions with Hausdorff distances below the threshold are considered similar, and the corresponding detection is skipped. Otherwise, a new detection is necessary. A smaller threshold imposes stricter similarity criteria, resulting in fewer partitions being considered similar and more frequent new detections. The similarity analysis starts with an initial threshold value t_0. This threshold is adaptively adjusted as $t'_0 = t_0 + \Delta t$, where Δt is the scene adjustment factor based on scene complexity. The similarity criteria become more stringent for partitions containing multiple objects in the previous frame and less stringent for those with fewer objects. The filtering ratio can be adjusted by tuning t_0 and Δt, enabling flexible latency-accuracy trade-offs for different application requirements.

V. Evaluation

A. Experiment Setup

The experiments are conducted on a high-performance desktop computer equipped with an Intel(R) Core(TM) i7-9700K CPU operating at 3.6 GHz, 64 GB of DDR4 RAM, and an NVIDIA GeForce RTX 2080Ti graphics card. We implement our method on OpenPCDet, a toolbox for point cloud object detection [25]. Our experiments are conducted on the dataset of KITTI [19]. We evaluate DAWN's performance by applying it to detection networks PointRCNN [8], PV-RCNN [10] and CasA-PV [26] using mean average precision (mAP) as the metric. The accuracy is calculated at the moderate difficulty level for the car class with 40 recall positions.

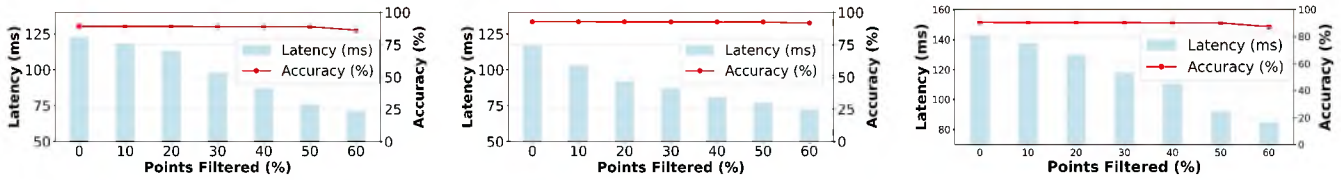
B. Performance Evaluation

Speedup and Accuracy Figure 9 shows the improvement of DAWN on networks PointRCNN [8],PV-RCNN [10] and CasA-PV [26]. A 50% filtering ratio means on average 50% of the input points require processing in each frame. The filtering ratio is adjusted by tuning the threshold in the similarity analysis. With 50% points filtered, PointRCNN [8] achieves 1.70× speedup with 0.3 accuracy loss. PV-RCNN [10] demonstrates a 1.51× speedup with 0.2 accuracy degradation when filtering 50% of points. CasA-PV [26] is accelerated by 1.56× with 0.4 accuracy loss when filtering 50% of points. The detection accuracy shows significant degradation when more than 60% of points are filtered.

System Performance Analysis We evaluate the system efficiency improvements of networks optimized by DAWN. As demonstrated in Figure 10, PV-RCNN [10], CasA-PV [26], and PointRCNN [8] optimized by DAWN show lower GPU memory usage and GPU utilization. This improvement can be attributed to the filtering of redundant points and reduced computational workload.

C. Comparison with Related Work

We compare DAWN with QuickFPS [31] and Mesorasi [16] in Table I. The performance of DAWN is evaluated with 50% of points filtered. Prior works show limited applicability: QuickFPS [31] supports only point-based and point-voxel based detection networks, while Mesorasi [16] is confined to point-based detection networks. In contrast, DAWN is compatible with voxel-based, point-based, and point-voxel based networks. The dash ('-') in Table I indicates that the corresponding method cannot be applied to that network. On PointRCNN [8], DAWN achieves 13.6 FPS with 0.3



(a) PointRCNN [8] (b) PV-RCNN [10] (c) CasA-PV [26]

Fig. 9: Latency-accuracy trade-off under different point filtering ratios for PointRCNN [8], PV-RCNN [10] and CasA-PV [26].

accuracy loss, outperforming Mesorasi [16] (10.3 FPS with 1.0 accuracy loss) and QuickFPS [31] (9.9 FPS with 0.7 accuracy loss). For PV-RCNN [10], DAWN reaches 13.0 FPS with 0.2 accuracy loss, better than QuickFPS [31] (11.0 FPS with 0.7 loss). On CasA-PV [26], DAWN achieves 10.6 FPS with 0.4 accuracy loss, outperforming QuickFPS [31] which achieves 8.8 FPS with 0.8 accuracy loss. Additionally, DAWN enhances SECOND's performance [5] to 21.7 FPS with 0.5 accuracy loss, demonstrating its effectiveness across different network architectures.

Method	Voxel-based			Point-based			Point-voxel based					
	SECOND [5]			PointRCNN [8]			PV-RCNN [10]			CasA-PV [26]		
	Acc.	Lat.	FPS	Acc.	Lat.	FPS	Acc.	Lat.	FPS	Acc.	Lat.	FPS
Original	88.1	52ms	19.2	89.1	124ms	8.1	92.8	117ms	8.5	90.4	142ms	7.0
Mesorasi [16]	-	-	-	88.1	97ms	10.3	-	-	-	-	-	-
QuickFPS [31]	-	-	-	88.4	102ms	9.9	92.1	91ms	11.0	89.6	113ms	8.8
DAWN	87.6	46ms	**21.7**	88.8	74ms	**13.6**	92.6	77ms	**13.0**	90.0	94ms	**10.6**

TABLE I: Performance comparison with related work. Acc. stands for Accuracy, Lat. stands for Latency and FPS denotes frames processed per second. DAWN is evaluated with 50% of points filtered. Dash (-) indicates that the corresponding method cannot be applied to that network.

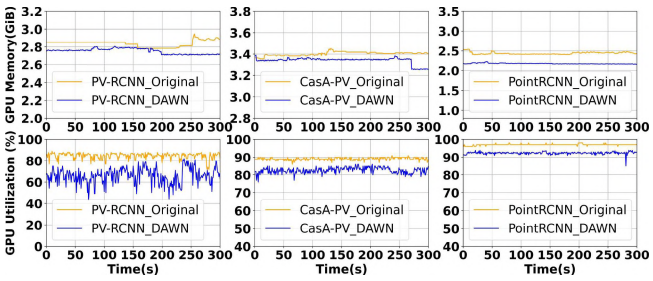


Fig. 10: Memory usage and utilization comparison between original and DAWN-optimized networks.

D. Partitioning Method Analysis

Distribution Analysis Figure 11 shows the granularity levels and accuracy performance of different partitioning methods. The granularity level defines how many points are processed for the similarity analysis. The histograms display the normalized point distribution on the log scale. In Figure 11 (a), fixed partitioning results in many empty partitions and other partitions containing the majority of points. Figure 11 (b) demonstrates that object-aware partitioning reduces large granularity (3000 to 6000). Figure 11 (c) shows that object-aware partitioning with axis-sorted selection ($n_g = 600$) minimizes mid-range granularity (2000 to 4000 points) while concentrating points around the target of $n_g = 600$.

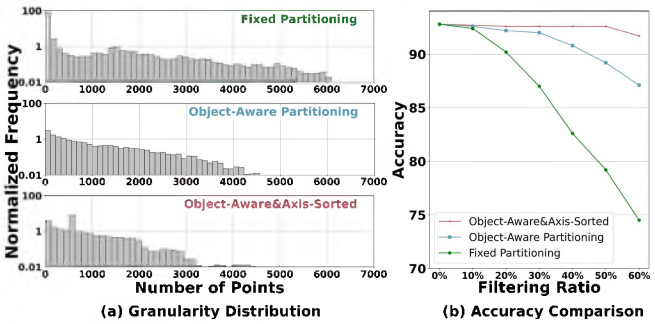


Fig. 11: The granularity distribution and accuracy comparison of different partitioning methods. The granularity is the number of points after partitioning and point selection.

Fragmentation Reduction Experimental results show that fixed partitioning results in 34.8% of objects being fragmented. Object-aware partitioning, defining boundaries based on previous detection results, reduces this fragmentation rate to 2.6%.

Accuracy Analysis Figure 11 (b) compares the accuracy of different partitioning methods to evaluate their ability to capture partial similarities. Filtering using object-aware partitioning demonstrates a much slower accuracy drop than filtering using fixed partitioning. The filtering with object-aware partitioning and axis-sorted point selection achieves the best performance, maintaining stable accuracy.

VI. Conclusion

This paper addresses the latency challenge of point cloud detection. We propose DAWN to leverage partial similarity between adjacent point cloud frames to accelerate detection. The object-aware partitioning and axis-sorted point selection optimize partitioning granularity to enable localized similarity comparison. DAWN filters redundant points by an efficient 3D similarity algorithm. It provides a flexible latency-accuracy trade-off to meet different application requirements. The experiment results show that DAWN improves state-of-the-art detection networks by 1.59× on average and up to 1.70×, with minimal accuracy loss.

REFERENCES

[1] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. and Bennamoun, M. (2020). Deep learning for 3d point clouds: A survey. transactions on pattern analysis and machine intelligence, 43(12), 4338-4364.

[2] Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In Proceedings of the conference on Computer Vision and Pattern Recognition (pp. 1907-1915).

[3] Mao, J., Shi, S., Wang, X., & Li, H. (2023). 3D object detection for autonomous driving: A comprehensive survey. International Journal of Computer Vision, 131(8), 1909-1963.

[4] Qian, R., Lai, X., & Li, X. (2022). 3D object detection for autonomous driving: A survey. Pattern Recognition, 130, 108796.

[5] Yan, Y., Mao, Y., & Li, B. (2018). Second: Sparsely embedded convolutional detection. Sensors, 18(10), 3337.

[6] Zhou, Y., & Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the conference on computer vision and pattern recognition (pp. 4490-4499).

[7] Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., & Li, H. (2021, May). Voxel r-cnn: Towards high performance voxel-based 3d object detection. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 2, pp. 1201-1209).

[8] Shi, S., Wang, X., & Li, H. (2019). Pointrcnn: 3d object proposal generation and detection from point cloud. In Proceedings of the 'CVF conference on computer vision and pattern recognition (pp. 770-779).

[9] Yang, Z., Sun, Y., Liu, S., & Jia, J. (2020). 3dssd: Point-based 3d single stage object detector. In Proceedings of the 'CVF conference on computer vision and pattern recognition (pp. 11040-11048).

[10] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In Proceedings of the 'CVF conference on computer vision and pattern recognition (pp. 10529-10538).

[11] Sheng, H., Cai, S., Liu, Y., Deng, B., Huang, J., Hua, X. S., & Zhao, M. J. (2021). Improving 3d object detection with channel-wise transformer. In Proceedings of the 'CVF international conference on computer vision (pp. 2743-2752).

[12] Lyu, D., Li, Z., Chen, Y., Xu, N., & He, G. (2023, July). FLNA: An energy-efficient point cloud feature learning accelerator with dataflow decoupling. In 2023 60th ACM/ Design Automation Conference (DAC) (pp. 1-6).

[13] Xu, T., Tian, B., & Zhu, Y. (2019, October). Tigris: Architecture and algorithms for 3d perception in point clouds. In Proceedings of the 52nd Annual 'ACM International Symposium on Microarchitecture (pp. 629-642).

[14] Yang, X., Fu, T., Dai, G., Zeng, S., Zhong, K., Hong, K., & Wang, Y. (2023, July). An efficient accelerator for point-based and voxel-based point cloud neural networks. In 2023 60th ACM/ Design Automation Conference (DAC) (pp. 1-6).

[15] Ying, Z., Bhuyan, S., Kang, Y., Zhang, Y., Kandemir, M. T., & Das, C. R. (2023, June). EdgePC: Efficient deep learning analytics for point clouds on edge devices. In Proceedings of the 50th Annual International Symposium on Computer Architecture (pp. 1-14).

[16] Feng, Y., Tian, B., Xu, T., Whatmough, P., & Zhu, Y. (2020, October). Mesorasi: Architecture support for point cloud analytics via delayed-aggregation. In 2020 53rd Annual 'ACM International Symposium on Microarchitecture (MICRO) (pp. 1037-1050).

[17] Zhang, J. F., & Zhang, Z. (2021, October). Point-x: A spatial-locality-aware architecture for energy-efficient graph-based point-cloud deep learning. In MICRO-54: 54th Annual 'ACM International Symposium on Microarchitecture (pp. 1078-1090).

[18] Lin, Y., Zhang, Z., Tang, H., Wang, H., & Han, S. (2021, October). Pointacc: Efficient point cloud accelerator. In MICRO-54: 54th Annual 'ACM International Symposium on Microarchitecture (pp. 449-461).

[19] Geiger, A., Lenz, P. & Urtasun, R. (2012, June). Are we ready for autonomous driving? the kitti vision benchmark suite. In 2012 conference on computer vision and pattern recognition (pp. 3354-3361).

[20] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., ... & Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In Proceedings of the 'CVF conference on computer vision and pattern recognition (pp. 11621-11631).

[21] Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., ... & Hays, J. (2023). Argoverse 2: Next generation datasets for self-driving perception and forecasting. arXiv preprint arXiv:2301.00493.

[22] Yang, Z., Sun, Y., Liu, S., Shen, X., & Jia, J. (2018). Ipod: Intensive point-based object detector for point cloud. arXiv preprint arXiv:1812.05276.

[23] Shi, W., & Rajkumar, R. (2020). Point-gnn: Graph neural network for 3d object detection in a point cloud. In Proceedings of the 'CVF conference on computer vision and pattern recognition (pp. 1711-1719).

[24] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the conference on computer vision and pattern recognition (pp. 652-660).

[25] OpenPCDet Development Team. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds, 2020.

[26] Wu, H., Deng, J., Wen, C., Li, X., Wang, C., & Li, J. (2022). CasA: A cascade attention network for 3-D object detection from LiDAR point clouds. Transactions on Geoscience and Remote Sensing, 60, 1-11.

[27] Feng, Y., Hammonds, G., Gan, Y., & Zhu, Y. (2022, June). Crescent: taming memory irregularities for accelerating deep point cloud analytics. In Proceedings of the 49th Annual International Symposium on Computer Architecture (pp. 962-977).

[28] Feng, X., Tang, C., Zhang, Z., Sun, W., & Liu, Y. (2023, January). Semantic Guided Fine-grained Point Cloud Quantization Framework for 3D Object Detection. In Proceedings of the 28th Asia and South Pacific Design Automation Conference (pp. 390-395).

[29] Li, W., Qu, Y., Chen, G., Ma, Y., & Yu, B. (2021, January). TreeNet: Deep point cloud embedding for routing tree construction. In Proceedings of the 26th Asia and South Pacific Design Automation Conference (pp. 164-169).

[30] Lee, M., Park, S., Kim, H., Yoon, M., Lee, J., Choi, J. W., ... & Choi, J. (2024, March). SPADE: Sparse Pillar-based 3D Object Detection Accelerator for Autonomous Driving. In 2024 International Symposium on High-Performance Computer Architecture (HPCA) (pp. 454-467).

[31] Han, M., Wang, L., Xiao, L., Zhang, H., Zhang, C., Xu, X. & Zhu, J. (2023). QuickFPS: Architecture and algorithm co-design for farthest point sampling in large-scale point clouds. Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42(11), 4011-4024.

[32] Zhao, P., Yuan, G., Cai, Y., Niu, W., Liu, Q., Wen, W., ... & Lin, X. (2021, December). Neural pruning search for real-time object detection of autonomous vehicles. In 2021 58th ACM/ Design Automation Conference (DAC) (pp. 835-840).

[33] Zheng, J., Jiang, H., Nie, X., Huang, Z., Chen, C., & Liu, Q. (2023, July). TiPU: A Spatial-Locality-Aware Near-Memory Tile Processing Unit for 3D Point Cloud Neural Network. In 2023 60th ACM/ Design Automation Conference (DAC) (pp. 1-6).

[34] Tang, D., Sun, X., Guan, N., Kuo, T. W., & Xue, C. J. (2022, December). *p*LPAQ: Accelerating LPAQ Compression on FPGA. In 2022 International Conference on Field-Programmable Technology (ICFPT) (pp. 1-6).

[35] Mao, Y., Cui, Y., Kuo, T. W., & Xue, C. J. (2022, April). Trace: A fast transformer-based general-purpose lossless compressor. In Proceedings of the ACM Web Conference 2022 (pp. 1829-1838).

[36] Liu, X., Song, Z., Chen, H., Li, X., & Liang, X. (2024, June). MoC: A Morton-Code-Based Fine-Grained Quantization for Accelerating Point Cloud Neural Networks. In Proceedings of the 61st ACM/ Design Automation Conference (pp. 1-6).

[37] Yoon, H., & Kim, J. J. (2024, June). Fused Sampling and Grouping with Search Space Reduction for Efficient Point Cloud Acceleration. In Proceedings of the 61st ACM/ Design Automation Conference (pp. 1-6).

[38] Mao, Y., Li, J., Cui, Y., & Xue, J. C. (2023, July). Faster and stronger lossless compression with optimized autoregressive framework. In 2023 60th ACM/ Design Automation Conference (DAC) (pp. 1-6).

[39] Li, Jingzong, et al. "Moby: Empowering 2d models for efficient point cloud analytics on the edge." Proceedings of the 31st ACM International Conference on Multimedia. 2023.