Rethink 3D Object Detection from Physical World

Satoshi Tanaka,

Koji Minoda, Fumiya Watanabe, TIER IV, Inc

Takamasa Horibe

satoshi.tanaka@tier4.jp

Abstract

High-accuracy and low-latency 3D object detection is essential for autonomous driving systems. While previous studies on 3D object detection often evaluate performance based on mean average precision (mAP) and latency, they typically fail to address the trade-off between speed and accuracy, such as 60.0 mAP at 100 ms vs 61.0 mAP at 500 ms. A quantitative assessment of the trade-offs between different hardware devices and accelerators remains unexplored, despite being critical for real-time applications. Furthermore, they overlook the impact on collision avoidance in motion planning, for example, 60.0 mAP leading to safer motion planning or 61.0 mAP leading to high-risk motion planning. In this paper, we introduce latency-aware AP (L-AP) and planning-aware AP (P-AP) as new metrics, which consider the physical world such as the concept of time and physical constraints, offering a more comprehensive evaluation for real-time 3D object detection. We demonstrate the effectiveness of our metrics for the entire autonomous driving system using nuPlan dataset, and evaluate 3D object detection models accounting for hardware differences and accelerators. We also develop a state-of-the-art performance model for real-time 3D object detection through latency-aware hyperparameter optimization (L-HPO) using our metrics. Additionally, we quantitatively demonstrate that the assumption "the more point clouds, the better the recognition performance" is incorrect for real-time applications and optimize both hardware and model selection using our metrics.

1. Introduction

As autonomous driving technology advances, the ability to perceive and understand the environment is crucial. While 2D object detection has been a mainstay in computer vision, 3D object detection is vital for autonomous vehicles to navigate complex environments safely. 3D object detection is more challenging than 2D object detection because it requires understanding both position and orientation with more dimensions, which is critical for tasks such as collision avoidance in motion planning. Research in 3D object detection has focused on the average precision (AP) metric, with state-of-the-art models being evaluated based on mean average precision (mAP) using open datasets such as nuScenes [7] and Waymo Open Dataset [38].

However, when considering real-time applications in robotics, such as autonomous vehicles, traditional evaluation metrics like mAP may not accurately reflect real-world performance. As shown in Fig. 1, a model might achieve a high mAP yet still perform poorly in practice. Fig. 1 (a) shows the case where detection accuracy is high, but inference time is too long. This is primarily because 3D object detection metrics have been adapted from those used in 2D object detection, which do not account for the dynamics of robotic systems. Evaluation in computer vision is typically conducted in a static environment, whereas robotics operates in a time-dependent context. As shown in Fig. 1, while the error in image space is small even with 500 ms latency and IoU (Intersection over Union) is sufficient in 2D object detection, the results show discrepancies, as the IoU in the BEV (Bird's Eye View) space approaches 0 in 3D object detection. Moreover, Fig. 1 (b) shows the model may treat yaw measurements for large and small vehicles identically, even though larger vehicles' yaw fluctuations have a greater impact. Additionally, Fig. 1 (c) shows failure to detect critical objects for motion planning can lead to unsafe conditions, highlighting the gap between traditional mAP metrics and real-world safety requirements.

For issue of latency, while previous works on 3D object detection often evaluate performance based on mean average precision (mAP) and latency, they typically fail to address the trade-off between speed and accuracy, such as 60.0 mAP at 100 ms vs 61.0 mAP at 500 ms. A quantitative assessment of the trade-offs between device variations and accelerators has yet to be researched, despite being critical for real-time applications. Moreover, even in cases where performance improves using large data inputs, such as merge of multi-frame LiDAR pointcloud, a trade-off between performance and inference time exists, which has yet to be quantitatively assessed. In practical applications, while using more powerful GPUs or employing accelerators like Ten-



Figure 1. The cases where a high score on existing mAP does not correspond to practical performance. Grey objects represent ground truths (GTs), while color objects represent prediction (Pred) results. The image of (a) shows inference latency for 2D and 3D object detection. Blue objects represent the results with a latency of 0 ms, while orange objects represent the results with a latency of 500 ms.

sorRT to reduce latency can sometimes "improve recognition performance," evaluations have thus far been limited to assessments along the latency axis only. In addition to latency, they leave the impact on collision avoidance in motion planning, for example, 60.0 mAP leading to safer motion planning or 61.0 mAP leading to high-risk motion planning. Even if the mAP is high, a model becomes unusable if it fails to detect objects that are critical for motion planning.

To address these issues, we rethink about the metrics for 3D object detection in the context of the physical world, especially "time" and "spatial perception". It becomes evident that current evaluation methods in 3D object detection do not account for "time," a fundamental concept in physics with a long research history. Moreover, current evaluation methods do not account for "spatial perception", a fundamental for motion planning which need the information to avoid collision for planning-aware objects. In this study, we proposes metrics for 3D object detection in real-time applications, by incorporating the concept of physicality. The primary contribution of this work is to bridge the gap between academic algorithms in robotics perception and those suitable for real-world implementation, advancing the development of robotics perception. Specifically, we make three key contributions:

1. We introduce latency-aware AP (L-AP) and planningaware AP (P-AP) as new evaluation metrics, and demonstrate their effectiveness as a metrics for motion planning in real-world scenarios using nuPlan dataset.

- 2. We evaluate current baseline 3D object detection models accounting for device differences and the use of accelerators using our metrics, and develop a high-performance model for real-time 3D object detection that surpasses existing models by performing latency-based hyperparameter optimization.
- 3. We show the optimal amount of point cloud in real-time 3D object detection and the optimization for hardware and model selections through quantitative evaluation.

2. Related Work

Real-time 3D object detection. 3D object detection [9, 30] is mainly divided into LiDAR-based and camera-based approaches. Researchers have developed 3D object detection models using LiDAR point cloud [33, 34, 36, 37, 46, 47, 49, 53]. PointPillars [21] stands out as a model with performance and speed, representing a breakthrough for LiDAR-based 3D object detection in real-time robotics. Center-Point [51] simplifies the architecture by using a centerbased representation, making it a popular baseline and practical choice in industry applications. There has also been research into 3D object detection models based on camera-LiDAR fusion [10, 13, 15, 16, 22, 24, 25, 27, 35, 40, 41]. TransFusion [4] is a baseline model using an attention-based method. BEVFusion [29] unifies bird's-eye view

(BEV) representation from multiple sensors and improves the performance. Camera-based 3D object detection also has been developed in computer vision researches [3, 39]. LSS [32] is commonly used as a baseline method in camerabased 3D object detection, where it lifts 2D image features to a BEV grid in 3D space. Recently, multi-camera 3D object detection models have gained attention [14, 19, 23, 43, 45, 48, 52]. BEVFormer [26] represents a breakthrough in multi-camera 3D object detection using the transformer mechanism and BEV representation, which transforms local image features from a 2D image encoder into BEV space. Several studies have focused on improving real-time performance in 3D object detection. [44] proposes an evaluation method for LiDAR-based 3D object detection that considers computational costs and adjusts the model accordingly. LidarNAS [28] uses neural architecture search (NAS) to enhance real-time 3D object detection. UTR3D [11] aims to achieve real-time 3D object detection with a cost-effective sensor kit.

Metrics for 3D object detection. As metrics for 3D object detection, the most common evaluation metric is Average Precision (AP), which extends the 2D mAP metric by assessing object detection accuracy in 3D space, accounting for both spatial overlap and object localization across all three dimensions. nuScenes detection score (NDS), proposed in the nuScenes dataset [7], combines various performance metrics including AP, orientation, and velocity to evaluate detection performance. In Waymo Open Dataset [38], not only AP but also AP weighted by heading (APH) is used for 3D object detection.

In addition to the major metrics proposed from datasets, several metrics have been reported to improve detection performance. Stability index [42] provides the metrics of consistency of 3D object detection models to create stable 3D object detection model. Pl-metrics [18] introduce a planning cost function into perception evaluation and analyze the relationship between perception failures and motion planning. OMNI3D [6] raises concerns about the adequacy of IoU as the sole evaluation metric for 3D object detection performance. LET-3D-AP [17] introduces a modified 3D AP that accounts for longitudinal errors, improving the evaluation of camera-only 3D object detection. Longrange Detection Score (LDS) [50] focuses on long-range 3D object detection. MultiCorrupt [5] evaluates the robustness of LiDAR-camera fusion models for 3D object detection, with an emphasis on diverse sensor data corruptions. Object Criticality Model [8] shows the importance of assessing object detection systems not only for accuracy but also for their safety and reliability. MTBF model [31] aims to construct safety evaluation considering perception error.

[8, 18, 31] analyze the relationship between task-driven metrics, however, they do not focus on developing higherperformance models while considering latency. [5, 6, 17,



Figure 2. The metric of L-AP. L-AP measures the accuracy of detection at the moment when the inference is completed, considering the latency involved.

Table 1. Evaluation results with P-mAP for Fig. 3 (c). d_e represents the error in the distance to the nearest surface. We use distance thresholds for P-mAP as (0.5, 1.0, 1.5, 2.0) [m].

	(a)	(b)	(c)
d_e	+0.25	+0.75	-0.75
Is matching?	Yes	No	Yes
P-mAP	1.0	0.0	0.75

Table 2. Comparison of different metrics across various cases in Fig. 4. We use distance thresholds for mAP as (0.5, 1.0, 1.5, 2.0) [m]. IoU is 0.55 in (b) and 0.27 in (d) and we use IoU thresholds for mAP as (0.3, 0.5).

	(a)	(b)	(c)	(d)	(e)
mAP@IoU	0.5	1.0	0.5	0.0	0.0
Planning-aware?	Yes	No	Yes	Yes	No
mAP@Center	1.0	0.25	0.5	1.0	1.0
Planning-aware?	No	Yes	Yes	No	Yes
mAHS@Center	1.0	0.25	0.5	0.83	0.83
Planning-aware?	No	Yes	Yes	No	Yes
mAP@Corner	0.75	0.25	0.5	0	0.75
Planning-aware?	Yes	Yes	Yes	Yes	Yes

42, 50] apply their proposed metrics to high-performance models but do not examine the relationship between these metrics and downstream tasks. Furthermore, to the best of our knowledge, no metric has been reported that incorporates latency and enables single-axis evaluation.

3. Method

3.1. Latency-aware Average Precision (L-AP)

In this work, we introduce Latency-aware Average Precision (L-AP) in the evaluation of 3D object detection, with the goal of integrating the concept of "time" into real-time 3D object detection. As shown in Fig. 2, we measure how accurate the detection is at the moment the inference is com-



Figure 3. (a) Mis-detection towards unsafe and safe. If a 3D object detection model misidentifies objects as being farther away than their actual positions, it can lead to planning errors, increasing the risk of collisions by causing inaccurate collision avoidance. In contrast, misidentifying objects as being closer than their actual positions in 3D object detection generally results in a lower risk of collisions in planning, showing behavior that appears to stop earlier. (b) Occlusion filtering for planning-aware objects. Occluded objects do not significantly impact for motion planning in many cases. (c) Matching algorithm in planning-aware objects. d_m is the margin parameter used in planning. If the distance error exceeds d_m and the object is farther than the ground truth, the object is considered not to match.



Figure 4. The case to evaluate of corner distance. Grey objects represent ground truths, while blue objects represent prediction results. (a) Varying vehicle sizes. (b) Different vertical positions of large vehicles. (c) Different horizontal positions of large vehicles. (d) Yaw deviation of $\pi/6$ in a large vehicle. (e) Yaw deviation of $\pi/6$ in a bicycle.

pleted. Specifically, we evaluate true positive (TP) metrics by adjusting the evaluation based on the velocity multiplied by latency, which effectively shifts the detection results in time. To begin, $\hat{v_t}^{GT}$, the velocity of the ground truth object at time t is calculated by

$$\hat{v_t}^{GT} = \frac{x_t^{GT} - x_{t-\delta t}^{GT}}{\delta t},\tag{1}$$

where x_t^{GT} represents the position of the object in ground truth at time t, and δt is the time interval between annotations. Next, we estimate the position of the object at time $(t + \Delta t)$ by

$$x_{t+\Delta t}^{GT} = x_t^{GT} + \hat{v_t}^{GT} \cdot \Delta t, \qquad (2)$$

where Δt is the inference time. Note that we will discuss the accuracy of $x_{t+\Delta t}^{GT}$ in Appendix 6.2. The predicted position at time $(t + \Delta t)$ is calculated by

$$x_{t+\Delta t}^{Pred} = x_t^{Pred} + v_t^{Pred} \cdot \Delta t - v_t^{Ego} \cdot \Delta t, \qquad (3)$$

where x_t^{Pred} represents the position of the predicted object at time t, v_t^{Pred} is the predicted velocity from the result of 3D object detection at time t, and v_t^{Ego} is the velocity of the ego vehicle at time t.

By incorporating latency into the evaluation, we provide a more practical metric for real-world applications of 3D object detection. Specifically, accounting for latency allows for a more accurate assessment of detection performance in operational settings. First, latency highlights that evaluation results can vary depending on the deployment hardware. For instance, upgrading the hardware can improve detection performance, as reduced latency leads to faster inference. However, traditional AP metrics do not directly consider latency, meaning these hardware improvements may not be fully captured in conventional evaluations. Furthermore, introducing latency enables a more consistent assessment of the benefits provided by acceleration libraries, such as TensorRT. These libraries are designed to optimize inference speed, and by incorporating latency into the evaluation, we can more uniformly measure their impact on overall detection performance.

3.2. Planning-aware Average Precision(P-AP)

In the context of perception systems, mis-detection can be categorized into two distinct types: safe mis-detection and unsafe mis-detection as shown in Fig. 3 (a). Safe misdetections are those where the system's errors do not lead to hazardous consequences. On the other hand, unsafe misdetections are the ones which have the potential to cause serious errors, such as collisions or failure to perform critical tasks. Typically, false negatives or overestimates of the distance from an ego-vehicle to a target vehicle turn out to be critical for planning algorithms. While traditional evaluation metrics often fail to differentiate between these types of mis-detections, it is critical for autonomous systems, particularly in real-time applications, to properly account for unsafe mis-detections, as they pose significant safety risks.

Therefore, we propose a metric of Planning-aware Average Precision (P-AP). First, we apply occlusion filtering to focus on planning-aware objects, as shown in Fig. 3(b). Objects that are not occluded and have a significant impact on motion planning are considered planning-aware objects and included in the evaluation. Additionally, P-AP regards mis-detections occurring at farther distances as overestimation, reflecting the increased risk posed by such errors, as illustrated in Fig. 3(c). While mis-detections closer to the object tend to result in the system stopping safely at the front, those occurring at greater distances can lead to dangerous collisions. In robotics, it is common practice to plan trajectories with a margin, which is often set as a parameter. In this work, we define the margin parameter d_m for planning as and set it to 0.5 [m]. In our proposed matching process, we use the distance to the nearest surface instead of the center position, which is often used in the matching step in AP calculation, because motion planning should focus on the surface closest to the ego vehicle rather than the center position. If the error of the distance of the nearest surface d_e exceeds d_m and the object is further than ground truth, the object is considered unsafe and not matching to ground truth. We calculate the scores with the penalty of farther distance in cases like those illustrated in Fig. 3 (c) and summarized in Table 1.

In addition to filtering and matching for planning-aware objects, we replace matching metrics from center distance to corner distance. This modification is motivated by principles from motion planning. In motion planning, the primary concern is not overlap-based metrics such as Intersection over Union (IoU) or the center position but rather the safety margin required to ensure a collision-free trajectory. Specifically, the focus is on determining the necessary clearance to avoid collisions, rather than merely assessing object proximity in the detection space. In practice, the center-based evaluation used in AP for nuScenes does not penalize minor misalignments, such as diagonal displacements, making it suboptimal for planning purposes. Moreover, evaluation metrics that account for yaw errors, such as AHS used in Waymo Open Dataset [38], excessively penalizes from yaw errors for small objects such as pedestrians and bicycles in the same way as for trailers.

Instead of center distance, we calculate the average displacement of the four corner points. First, the corner distance for each corner point $d_c(i)$ is calculated by $d_c(i) = \sqrt{(x_i^{\text{Pred}} - x_i^{\text{GT}})^2}$ $(i \in P)$, where *i* represents each corner point, *P* is set of corner points of the bounding box, x_i^{Pred} is the position of the predicted object's corner, and x_i^{GT} is the position of the ground truth object's corner. The overall corner distance d_c is then computed as the average of the

distances across all four corners by $d_c = \frac{1}{4} \sum_{i \in P} d_c(i)$.

In Fig. 4, we compare the performance of different metrics across various real-world cases, including mAP(IoU), mAP(Center distance), mAHS, and P-mAP as shown in Table 2. For the cases in Fig. 4 (a) and (d), P-mAP is able to handle mis-detections of size, while AP and AHS fail to account for this factor. In the cases of Fig. 4 (b) and (e), P-mAP does not overestimate or underestimate the value, showing a reasonable decrease. However, AP(IoU) tends to produce a high score when size detection fails, or it underestimates the detection of small objects that are not significantly misaligned in absolute terms.

3.3. Latency-aware Hyperparameter Optimization

Building on our previous proposals, we further conduct latency-aware hyperparameter optimization (L-HPO) to enhance real-time 3D object detection. By incorporating latency into the optimization process, we can achieve optimal performance while ensuring timely decision-making and maintaining system reliability in applications like robotics and autonomous vehicles. Existing methods typically optimize by assigning weights to both latency and metrics such as mAP. However, by using the proposed metric, optimization can be performed directly along an evaluation axis, enabling a more efficient search for the optimal model. In this paper, we use the CenterPoint [51] model as base model to optimize hyperparameters such as the number of layers, the number of channels in each layer, the number of multiframe to merge point clouds, and the voxel size, aiming to find the model for the highest L-mAP.

4. Experiment

4.1. Preparation

In Sec. 4.2, to evaluate the effectiveness of our metrics as a comprehensive performance indicator for the application based on 3D object detection, we evaluate the overall behavior of an autonomous driving system using nuScenes dataset [7] and nuPlan dataset [20]. For 3D object detection, we use ground truth as prediction results and added errors in latency of inference, yaw degree of objects, and the position of objects to them in nuPlan dataset. To evaluate the contribution of the metrics to motion planning, we use PDM-Hybrid [12] as the baseline with nuPlan dataset. In our experiments, we use the closed-loop metric as nuPlan score used in [12].

From Sec. 4.3 to Sec. 4.5, we evaluate 3D object detection based on our metrics using the validation dataset of the nuScenes dataset. The hardware devices used for testing by an RTX3090 and an RTX4060Ti. The software backend includes PyTorch and TensorRT [2], an optimized deep learning inference library by NVIDIA that accelerates latency reduction on GPUs. For the 3D object detection models, we use PointPillars [21], CenterPoint[51], TransFusion-L [4] (LiDAR-only model), BEVFusion [29], and BEVFormer [26] (base model). LiDAR-only model of TransFusion is prepared for comparison with BEVFusion. The inference time for reference is based on the respective papers and their TensorRT implementation [1].

4.2. Evaluation for Metrics

Table 3 (a) presents the results of L-mAP as latency of 3D object detection increases. We observe that nuPlan score decreases in accordance with the increase in the latency, and L-mAP also correspondingly declines. Table 3 (b) presents the results of P-mAP with the error in yaw angle of objects. We observe that nuPlan score decreases in accordance with the increase in yaw difference, and P-mAP also correspondingly declines. Notably, in both cases, mAP remains consistently at 100.0, whereas L-mAP and P-mAP provide a more relevant evaluation metric by incorporating planning-based considerations. These results demonstrate that L-mAP and P-mAP better reflect the actual performance of real-time 3D object detection than traditional mAP.

Table 3 (c) presents the results of P-mAP with the difference in object positions. When errors are introduced to objects close to the ego vehicle, the value of P-mAP decreases, and nuPlan score similarly decreases. For objects close to the ego vehicle, when they are shifted slightly, mAP drops from 83.2 to P-mAP 76.6, and nuPlan score increases because the metrics related to collision risk improve. However, when these objects are displaced farther away, mAP drops from 83.3 to P-mAP 24.7, and nuPlan score also decreases accordingly. This quantitatively demonstrates the dangerous scenario of "mis-detection of nearby objects as distant" and shows that this metric can effectively reflect performance degradation in the planning metric. Moreover, when objects are displaced over longer distances, the impact on nuPlan score is smaller compared to objects displaced nearby. This behavior is well-represented by P-mAP, which aligns closely with the planning metric.

4.3. Benchmark with Our Metrics

In this study, we conducted an evaluation with our metrics to assess the real-time performance of various 3D object detection models, including our proposed model in Sec. 3.3. The results are shown in Table 4. L-HPO outperforms other models in L-mAP and demonstrates high performance for real-time 3D object detection applications. Specifically, L-HPO increase L-mAP by +8.4 from CenterPoint, by +1.5 from TransFusion-L, by +5.9 from BEVFusion. L-HPO surpasses P-mAP by +9.6 from TransFusion-L and +10.6 from BEVFusion despite L-HPO being lightweight similar to PointPillars. In addition to it, it is comparable performance in P-mAP to CenterPoint despite L-HPO being lighter than CenterPoint. These results suggest that L-HPO is a better choice in the context of real-time robotics.

As for the analysis of other models, it is noteworthy that, when comparing TransFusion-L and BEVFusion in terms of L-mAP, the L-mAP of BEVFusion decreased by -4.4 (from 53.6 to 49.2) while its mAP increased by +4.1 (from 64.3 to 68.4). This indicates that the additional latency introduced by Camera-LiDAR fusion led to a performance drop in real-time 3D object detection. When comparing BEVFormer with PointPillars between Camera-based detection and LiDAR-based detection, mAP of BEVFormer exceed the performance of PointPillars (41.7 vs 39.0), however, BEVFormer do not exceed in L-mAP (18.3 vs 38.0), meaning that BEVFormer can be considered unsuitable for real-time applications. CenterPoint provides highest performance in terms of P-mAP, and both TransFusion-L and BEVFusion were unable to achieve high scores in P-mAP unlike mAP, making CenterPoint a more favorable model for motion planning tasks in real-time applications.

Additionally, we evaluated the impact of weaker GPUs and acceleration libraries on performance in Table 5. Device performance improvements, such as reducing latency, directly contribute to better perception performance in autonomous driving systems in practical scenarios. By upgrading from RTX4060Ti to RTX3090, we observed improvements in L-mAP as +15.1 in CenterPoint, +17.7 in TransFusion-L, +18.0 in BEVFusion, and +14.1 in L-HPO. Regarding the contribution of TensorRT acceleration, using accelerators to reduce latency improved real-time performance significantly. TensorRT improved the following models in L-mAP as +23.4 in CenterPoint, +27.1 in TransFusion-L, +33.8 in BEVFusion, and +16.1 in L-HPO. By using a single-axis metric for evaluation, it becomes possible to compare performance across different devices and accelerators. For example, "RTX4060Ti + TensorRT + CenterPoint" outperforms "RTX3090 + Pytorch + TransFusion-L" in L-mAP (55.0 vs 53.6) and it suits for application with real-time 3D object detection.

4.4. Rich Input Data Makes Better Perception?

In previous studies, merging several frame point clouds as one input has primarily been used to improve the accuracy of 3D object detection models. However, these approaches have often overlooked the real-time performance requirements critical in real-world applications. The assumption that a higher density of LiDAR point clouds automatically leads to better recognition accuracy is a common, yet oversimplified. This assumption fails to account for the concept of "time", which are fundamental in real-time applications. Despite the widespread nature of this misconception, there is a noticeable lack of studies that quantitatively demonstrate its impact. To address this gap, we evaluate the optimal number of LiDAR points required for real-time robotics. In this study, we focus on measuring latency as Table 3. Evaluation in L-mAP and P-mAP with latency, yaw difference, and position difference using nuPlan dataset. NuPlan refers to the score based on the closed-loop metric for nuPlan dataset. (a) Verification of effectiveness in L-mAP for latency. "Lat." refers to latency. (b) Verification of effectiveness in P-mAP for yaw difference. "Yaw" refers to the yaw difference from the ground truth. (c) Verification of effectiveness in P-mAP for the position error of planning-aware object. "Difference" refers to the positional difference between the predicted and ground truth object positions. A positive value means the predicted objects are placed farther, and a negative value means the predicted objects is placed closer.

Lat.	L-mAP	nuPlan	Yav	v P-mAP	nuPlan		Difference	mAP	P-mAP	nuPlan
$0\mathrm{ms}$	100.0	0.9277	0	100.0	0.9277	_	0 m	100.0	100.0	0.9277
$50\mathrm{ms}$	98.6	0.9274	$\pi/2$	4 91.1	0.8539		$-0.6 \mathrm{m}(< 20 \mathrm{m})$	83.2	76.6 (-6.6)	0.9354
$100\mathrm{ms}$	94.3	0.9267	$\pi/1$	2 78.5	0.7976		$+0.6 \mathrm{m}(< 20 \mathrm{m})$	83.3	24.7 (-58.6)	0.8578
$200\mathrm{ms}$	88.5	0.9143	$\pi/6$	60.2	0.7183		$-0.6 \mathrm{m}(> 20 \mathrm{m})$	77.3	79.2 (+1.9)	0.9310
$500\mathrm{ms}$	78.2	0.8978	$\pi/2$	49.7	0.6544		$+0.6 \mathrm{m}(> 20 \mathrm{m})$	77.3	25.0 (-52.3)	0.9248
	(a)			(b)		_		(c)		

Table 4. Evaluation results for 3D object detection on the nuScenes validation dataset using an RTX3090 GPU and the PyTorch environment. We recalculated mAP using the MMDetection3D base framework, establishing a baseline for comparison.

	Modality	Latency	GPU memory	NDS	mAP	L-mAP	P-mAP
PointPillars [21]	LiDAR	$34\mathrm{ms}$	7512MiB	52.6	39.0	38.0	8.4
CenterPoint[51]	LiDAR	$80\mathrm{ms}$	4285MiB	64.8	56.3	46.7	36.9
TransFusion-L [4]	LiDAR	$80\mathrm{ms}$	4412MiB	69.1	64.3	53.6	27.0
BEVFusion [29]	Camera, LiDAR	$119\mathrm{ms}$	8246MiB	71.2	68.4	49.2	26.0
BEVFormer [26]	Camera	$416\mathrm{ms}$	5435MiB	51.8	41.7	18.3	18.4
L-HPO (Ours)	LiDAR	$45\mathrm{ms}$	4136MiB	64.8	57.7	55.1	36.6

Table 5. Performance comparison of different models with latency and various environments. (P) refers to the PyTorch backend, and (T) refers to the TensorRT backend. "Lat." stands for latency.

Environment	Model	Lat.	L-mAP
RTX4060Ti(P)	CenterPoint	$197\mathrm{ms}$	31.6
RTX4060Ti(P)	TransFusion-L	$202\mathrm{ms}$	35.9
RTX4060Ti(P)	BEVFusion	$277\mathrm{ms}$	31.2
RTX4060Ti(P)	L-HPO (Ours)	$123\mathrm{ms}$	41.0
RTX3090(P)	CenterPoint	$80\mathrm{ms}$	46.7
RTX3090(P)	TransFusion-L	$80\mathrm{ms}$	53.6
RTX3090(P)	BEVFusion	$119\mathrm{ms}$	49.2
RTX3090(P)	L-HPO (Ours)	$45\mathrm{ms}$	55.1
RTX4060Ti(T)	CenterPoint	$33\mathrm{ms}$	55.0
RTX4060Ti(T)	TransFusion-L	$34\mathrm{ms}$	63.0
RTX4060Ti(T)	BEVFusion	$47\mathrm{ms}$	64.8
RTX4060Ti(T)	L-HPO (Ours)	$20\mathrm{ms}$	57.1
RTX3090(T)	CenterPoint	$10\mathrm{ms}$	56.2
RTX3090(T)	TransFusion-L	$9\mathrm{ms}$	64.3
RTX3090(T)	BEVFusion	$20\mathrm{ms}$	68.0
RTX3090(T)	L-HPO (Ours)	$7\mathrm{ms}$	57.5

the sum of preprocessing time to merge point clouds and the model inference time. Since each frame contains approximately 34k LiDAR points in nuScenes dataset, the merged point clouds with 9 frames increases to about 300k points. At this case, the processing time to merge multi-frame point

Table 6. Performance evaluation of multi-frame point clouds for CenterPoint with RTX3090 GPU and Pytorch backend. N is the number of frames to merge point clouds. P_N is the number of merged point clouds.

N	P_N	Latency	mAP	L-mAP	P-mAP
1	34k	$31\mathrm{ms}$	46.7	45.5	28.5
3	102k	$44\mathrm{ms}$	52.5	50.3	35.9
6	204k	$67\mathrm{ms}$	55.4	48.3	35.9
9	306k	$99\mathrm{ms}$	56.3	43.7	36.9

clouds takes 19 ms, making the total inference time 99ms with 80ms of the model's own inference time.

Table 6 shows the result for multi-frame point clouds. As the number of input points increased, we observed improvements in mAP, P-mAP, indicating a positive correlation between point cloud density and detection accuracy. However, when considering L-mAP, a different trend emerged. Specifically, the highest L-mAP was achieved 50.3 at 3 frames, but it decreased as the frames increased: 48.3 at 6 frames and 43.7 at 9 frames. This indicates that the increased latency due to the larger point cloud size, had a greater negative impact on the model's performance than the improvements in accuracy brought by denser point clouds. Our findings demonstrate that, in real-time environments where low latency is critical, indiscriminately increasing the input data size offers diminishing returns. This

Table 7. Cost comparison for different models and backends with different devices. "Cen." represents CenterPoint and "Trans." represents TransFusion-L. "(P)" refers to the PyTorch backend, and "(T)" refers to the TensorRT backend. "C-1" represents cost for 1 systems, "C-10" for 10 systems, "C-100" for 100 systems (\$).

Model	Device	L-mAP	C-1	C-10	C-100
Cen.	4060Ti(P)	31.6	21k	30k	120k
Cen.	3090(P)	46.7	24k	60k	420k
Cen.	4060Ti(T)	55.0	41k	50k	140k
Cen.	3090(T)	56.2	44k	80k	440k
Trans.	4060Ti(P)	35.9	41k	50k	140k
Trans.	3090(P)	53.6	44k	80k	440k
Trans.	4060Ti(T)	63.0	61k	70k	160k
Trans.	3090(T)	64.3	64k	100k	460k

study provides a quantitative metric to show that a balance must be struck between accuracy and latency to achieve optimal performance in real-time 3D object detection for robotics.

4.5. Optimize development cost with hardware

Furthermore, as an industry case study, we explore the optimization of development costs and hardware selection using L-mAP. By using L-mAP, we enable quantitative comparisons that extend beyond differences in hardware devices and acceleration libraries. In this study, we formulate an optimization problem where L-mAP serves as the objective function, considering both hardware and development costs. The hardware cost is defined as \$4k for a system with an RTX3090 GPU and \$1k for a system with an RTX4060Ti GPU. The initial development cost is defined as \$20k for implementing CenterPoint in PyTorch (three months of work for a single engineer). Upgrading the algorithm from CenterPoint to TransFusion-L in PyTorch requires an additional cost of \$20k, while developing a TensorRT version of either CenterPoint or TransFusion-L requires an additional \$20k. The experiment considers a business scenario in which 1, 10, or 100 perception systems are deployed, with the goal of determining the most costeffective approach to maximizing L-mAP.

For a budget of \$60k allocated to deploying 10 perception systems, we compare three strategies: upgrading the hardware by running CenterPoint in PyTorch on an RTX3090, using an accelerator by implementing Center-Point in TensorRT on an RTX4060Ti, and upgrading the algorithm by employing TransFusion-L in PyTorch on an RTX4060Ti. The results indicate that upgrading the hardware leads to an L-mAP improvement to 46.7, using TensorRT improves it to 55.0, while upgrading the algorithm to TransFusion-L results in a lower L-mAP of 35.9. These findings suggest that, within this budget constraint, implementing TensorRT for CenterPoint provides the best costperformance.

When considering the case of deploying a single perception system, development costs become a more dominant factor compared to the deployment of multiple systems. In this scenario, upgrading from an RTX4060Ti to an RTX3090 is the most cost-effective option, as it improves L-mAP to 46.7 at a relatively low additional cost. In contrast, when deploying 100 perception systems, hardware costs become the primary concern. Upgrading from an RTX4060Ti to an RTX3090 across all systems incurs a total cost of \$400k, leading to an L-mAP of 46.7. However, switching from CenterPoint to TransFusion-L and utilizing TensorRT achieves a significantly higher L-mAP of 63.0 at a lower total cost of \$140k, making it the more cost-efficient choice. These results demonstrate that the optimal strategy depends on the scale of deployment. For small-scale deployments, upgrading the hardware provides the best balance between cost and performance, whereas for large-scale deployments, algorithmic optimization and acceleration using TensorRT yield the highest cost efficiency. By quantitatively comparing different configurations using L-mAP, this study provides insights into the tradeoffs between hardware selection, algorithmic improvements, and deployment scale in industrial applications.

5. Conclusion

In this paper, we propose latency-aware AP (L-AP) and planning-aware AP (P-AP) as new metrics, which consider the physical world such as the concept of time and physical constraints, offering a more comprehensive evaluation for real-time 3D object detection. Using nuPlan dataset, we evaluate the effectiveness of these metrics for the whole autonomous driving system and demonstrate their utility in capturing planning-relevant aspects of 3D object detection. Furthermore, we develop a state-of-the-art performance model for real-time 3D object detection through latency-aware hyperparameter optimization (L-HPO) based on our proposed metrics. Our metrics also enable quantitative evaluation along a single axis, accounting for hardware differences and accelerators. Additionally, we quantitatively optimize the number of frames for point clouds, as well as hardware and model selection, using our metrics.

We hope that this paper contributes to advancements in both algorithmic development within the research community and performance optimization for real-world applications in the industry. As future work, we aim to further develop state-of-the-art models for real-time 3D object detection using the proposed metrics. Additionally, we plan to refine these metrics for 3D object tracking, motion prediction, and motion planning to enhance real-time performance across a broader range of tasks.

References

- [1] Nvidia-ai-iot. https://github.com/NVIDIA-AI-IOT/Lidar_AI_Solution. 6
- [2] Tensorrt. https://github.com/NVIDIA/ TensorRT.5
- [3] Waleed Ali, Sherif Abdelkarim, Mohamed Zahran, Mahmoud Zidan, and Ahmad El Sallab. Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In ECCV Workshops, 2018. 3
- [4] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1080–1089, 2022. 2, 6, 7
- [5] Till Beemelmanns, Quan Zhang, Christian Geller, and Lutz Eckstein. Multicorrupt: A multi-modal robustness dataset and benchmark of lidar-camera fusion for 3d object detection. In 2024 IEEE Intelligent Vehicles Symposium (IV), pages 3255–3261, 2024. 3
- [6] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3D: A large benchmark and model for 3D object detection in the wild. In CVPR, Vancouver, Canada, 2023. IEEE. 3
- [7] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1, 3, 5
- [8] Andrea Ceccarelli and Leonardo Montecchi. Evaluating object (mis)detection from a safety and reliability perspective: Discussion and measures. *IEEE Access*, 11:44952–44963, 2023. 3
- [9] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl K. Wellington. 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Processing Magazine*, 38:68–86, 2020. 2
- [10] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6526–6534, 2016. 2
- [11] Xuanyao Chen, Tianyuan Zhang, Yue Wang, Yilun Wang, and Hang Zhao. Futr3d: A unified sensor fusion framework for 3d detection. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 172–181, 2022. 3
- [12] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learningbased vehicle motion planning. In *Conference on Robot Learning (CoRL)*, 2023. 5
- [13] Florian Drews, Di Feng, Florian Faion, Lars Rosenbaum, Michael Ulrich, and Claudius Gläser. Deepfusion: A robust and modular 3d object detector for lidars, cameras and radars. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 560–567, 2022. 2

- [14] Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *ArXiv*, abs/2112.11790, 2021. 3
- [15] Junjie Huang, Yun Ye, Zhujin Liang, Yi Shan, and Dalong Du. Detecting as labeling: Rethinking lidar-camera fusion in 3d object detection. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29 – October* 4, 2024, Proceedings, Part XXII, page 439–455, Berlin, Heidelberg, 2024. Springer-Verlag. 2
- [16] Keli Huang, Botian Shi, Xiang Li, Xin Li, Siyuan Huang, and Yikang Li. Multi-modal sensor fusion for auto driving perception: A survey. ArXiv, abs/2202.02703, 2022. 2
- [17] Wei-Chih Hung, Vincent Casser, Henrik Kretzschmar, Jyh-Jing Hwang, and Dragomir Anguelov. Let-3d-ap: Longitudinal error tolerant 3d average precision for camera-only 3d detection. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 8272–8279, 2024. 3
- [18] B. Ivanovic and Marco Pavone. Injecting planningawareness into prediction and detection evaluation. 2022 IEEE Intelligent Vehicles Symposium (IV), pages 821–828, 2021. 3
- [19] Xiaohui Jiang, Shuailin Li, Yingfei Liu, Shihao Wang, Fan Jia, Tiancai Wang, Lijin Han, and Xiangyu Zhang. Far3d: Expanding the horizon for surround-view 3d object detection. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024*, pages 2561–2569. AAAI Press, 2024. 3
- [20] Napat Karnchanachari, Dimitris Geromichalos, Kok Seang Tan, Nanxiang Li, Christopher Eriksen, Shakiba Yaghoubi, Noushin Mehdipour, Gianmarco Bernasconi, Whye Kit Fong, Yiluan Guo, and Holger Caesar. Towards learningbased planning: The nuplan benchmark for real-world autonomous driving. 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 629–636, 2024. 5
- [21] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12689–12697, 2018. 2, 6, 7
- [22] Hongyang Li, Chonghao Sima, Jifeng Dai, Wenhai Wang, Lewei Lu, Huijie Wang, Enze Xie, Zhiqi Li, Hanming Deng, Haonan Tian, Xizhou Zhu, Li Chen, Tianyu Li, Yulu Gao, Xiangwei Geng, Jianqiang Zeng, Yang Li, Jiazhi Yang, Xiaosong Jia, Bo Yu, Y. Qiao, Dahua Lin, Siqian Liu, Junchi Yan, Jianping Shi, and Ping Luo. Delving into the devils of bird's-eye-view perception: A review, evaluation and recipe. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:2151–2170, 2022. 2
- [23] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jian-Yuan Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. ArXiv, abs/2206.10092, 2022. 3
- [24] Yingwei Li, Adams Wei Yu, Tianjian Meng, Ben Caine, Jiquan Ngiam, Daiyi Peng, Junyang Shen, Yifeng Lu, Denny Zhou, Quoc V. Le, Alan Yuille, and Mingxing Tan. Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. In 2022 IEEE/CVF Conference on Computer Vi-

sion and Pattern Recognition (CVPR), pages 17161–17170, 2022. 2

- [25] Yingyan Li, Lue Fan, Yang Liu, Zehao Huang, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Fully sparse fusion for 3d object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2
- [26] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. pages 1–18, 2022. 3, 6, 7
- [27] Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi Tang. Bevfusion: A simple and robust lidar-camera fusion framework. *ArXiv*, abs/2205.13790, 2022. 2
- [28] Chenxi Liu, Zhaoqi Leng, Pei Sun, Shuyang Cheng, Charles R. Qi, Yin Zhou, Mingxing Tan, and Dragomir Anguelov. Lidarnas: Unifying and searching neural architectures for 3d point clouds. In *Computer Vision – ECCV* 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI, page 158–175, Berlin, Heidelberg, 2022. Springer-Verlag. 3
- [29] Zhijian Liu, Haotian Tang, Alexander Amini, Xingyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multitask multi-sensor fusion with unified bird's-eye view representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 2, 6, 7
- [30] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A comprehensive survey, 2023. 2
- [31] Fabian Oboril, Cornelius Bürkle, Alon Sussmann, Simcha Bitton, and Simone Fabris. Mtbf model for avs - from perception errors to vehicle-level failures. 2022 IEEE Intelligent Vehicles Symposium (IV), pages 1591–1598, 2022. 3
- [32] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV*, page 194–210, Berlin, Heidelberg, 2020. Springer-Verlag. 3
- [33] C. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 77–85, 2016. 2
- [34] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. 2
- [35] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 918–927, 2018. 2
- [36] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point

cloud. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–779, 2018. 2

- [37] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10526–10535, 2019. 2
- [38] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 2443–2451. Computer Vision Foundation / IEEE, 2020. 1, 3, 5
- [39] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 9626–9635, 2019. 3
- [40] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4603–4611, 2019. 2
- [41] Chunwei Wang, Chao Ma, Ming Zhu, and Xiaokang Yang. Pointaugmenting: Cross-modal augmentation for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11794– 11803, 2021. 2
- [42] Jiabao Wang, Qiang Meng, Guochao Liu, Liujiang Yan, Ke Wang, Ming-Ming Cheng, and Qibin Hou. Towards stable 3d object detection. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October* 4, 2024, Proceedings, Part L, page 197–213, Berlin, Heidelberg, 2024. Springer-Verlag. 3
- [43] Shihao Wang, Yingfei Liu, Tiancai Wang, Ying Li, and Xiangyu Zhang. Exploring object-centric temporal modeling for efficient multi-view 3d object detection. 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pages 3598–3608, 2023. 3
- [44] Xiaofang Wang and Kris M. Kitani. Cost-aware evaluation and model scaling for lidar-based 3d object detection. 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 9260–9267, 2022. 3
- [45] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin M. Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *The Conference on Robot Learning (CoRL)*, 2021. 3
- [46] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors (Basel, Switzerland)*, 18, 2018. 2
- [47] Binh Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Realtime 3d object detection from point clouds. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7652–7660, 2018. 2

- [48] Chenyu Yang, Yuntao Chen, Hao Tian, Chenxin Tao, Xizhou Zhu, Zhaoxiang Zhang, Gao Huang, Hongyang Li, Yu Qiao, Lewei Lu, Jie Zhou, and Jifeng Dai. BEVFormer v2: Adapting Modern Image Backbones to Bird's-Eye-View Recognition via Perspective Supervision . pages 17830–17839, 2023. 3
- [49] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 11037–11045, 2020. 2
- [50] Zetong Yang, Zhiding Yu, Chris Choy, Renhao Wang, Anima Anandkumar, and Jose M. Alvarez. Improving distant 3d object detection using 2d box supervision. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 14853–14863, 2024. 3
- [51] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Centerbased 3d object detection and tracking. *CVPR*, 2021. 2, 5, 6, 7
- [52] Yunpeng Zhang, Zheng Hua Zhu, Wenzhao Zheng, Junjie Huang, Guan Huang, Jie Zhou, and Jiwen Lu. Beverse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving. *ArXiv*, abs/2205.09743, 2022.
 3
- [53] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4490–4499, 2017. 2

Rethink 3D Object Detection from Physical World

Supplementary Material

6. Appendix

6.1. L-AP for Yaw Flipping



Figure 5. The case to evaluate for yaw flipping. (a) The velocity estimation is sufficiently accurate. (b) The direction is estimated in the opposite orientation. (c) The object is almost stationary. (d) The object is almost stationary but estimated in the opposite direction.

Table 8. Comparison of mAP and L-mAP at different inference time with π yaw difference. The thresholds are (0.5, 1.0, 1.5, 2.0).

	(a)	(b)	(c)	(d)
mAP	1.0	1.0	1.0	1.0
L-mAP for 100ms	1.0	0.25	1.0	1.0
L-mAP for 200ms	0.75	0.0	1.0	1.0
L-mAP for 1000ms	0.0	0.0	1.0	0.25

By introducing latency into the evaluation, it becomes possible to assess objects whose yaw orientation is incorrectly estimated in the opposite direction. In the cases of Fig. 5, we compare the performance of L-mAP with different inference time as shown in Table 8. Comparing (a) and (c), the greater the relative velocity of the target, the more significantly latency impacts recognition performance degradation. In (b), where the yaw is reversed by 180 degrees, the AP remains 1.0 when latency is not considered. However, when latency is taken into account, the model properly reflects the performance degradation. In (d), since the velocity is small, the performance degradation is less severe compared to (b).

6.2. Annotation Frequency

Here, we consider whether the annotation frequency is appropriate. We assume the velocity of ego vehicle v_t^{ego} as

 $0 \,\mathrm{m/s}$. This assumption does not lose generality, as we can consider it in a relative coordinate system.

First, we analyze the estimation error of $x_{t+\Delta t}^{GT}$. The ground truth of the position x at the time $(t + \Delta t)$, $x_{t+\Delta t}^{*GT}$ is calculated by

$$x_{t+\Delta t}^{*GT} = x_t^{GT} + v_t^{GT} \Delta t + \frac{1}{2} a_t^{GT} \Delta t^2 + \frac{1}{6} j \Delta t^3 \qquad (4)$$

, where a_t^{GT} is acceleration at the time t,j is jerk. In this work, The jerk is assumed to be constant. The error of the position E_x between $x_{t+\Delta t}^{*GT}$ and $x_{t+\Delta t}^{GT}$ is calculated by

$$E_x = x_{t+\Delta t}^{*GT} - x_{t+\Delta t}^{GT}$$

$$= \frac{(a_{t-\delta t}^{GT}\delta t + j\delta t^2)}{2}\Delta t + \frac{(a_{t-\delta t}^{GT} + j\delta t)}{2}\Delta t^2 + \frac{j}{6}\Delta t^3.$$
(6)

Eq. (6) indicates that the estimation error decreases as δt becomes smaller and Δt becomes smaller. Additionally, the estimation error decreases as $a_{t-\delta t}^{GT}$ becomes smaller and j becomes smaller.

For each Δt and δt , we visualized the relationship between estimation error $E_x = (0.05 \text{ m}, 0.1 \text{ m}, 0.2 \text{ m}, 0.5 \text{ m})$ and the corresponding values of j and $a_{t-\delta t}^{GT}$. As reference values, we also plotted the cases of normal driving $(j = 1 \text{ m/s}^3, a_{t-\delta t}^{GT} = 0.2 \text{ m/s}^2)$ and emergency case $(j = 3 \text{ m/s}^3, a_{t-\delta t}^{GT} = 0.6 \text{ m/s}^2)$. Fig. 6 shows the estimation error of $x_{t+\Delta t}^{GT}$ for inference time $\Delta t = 0.2 \text{ s}$. Fig. 7 shows the estimation error of $x_{t+\Delta t}^{GT}$ for inference time $\Delta t = 0.5 \text{ s}$.

In the nuScenes annotation, the annotation frequency is set as $\delta t = 0.5$ s (2 Hz annotation). When $\Delta t = 0.2$ s, the estimation error is approximately 0.05 m in a normal driving scenario. This error is sufficiently small compared to commonly used thresholds (0.5 m, 1.0 m, 1.5 m, 2.0 m), indicating that the annotation frequency is adequate for overall evaluation using mAP. Even in emergency scenarios, the estimation error remains within 0.2 m, which suggests that the overall evaluation is still valid. However, when $\Delta t =$ $0.5 \,\mathrm{s}$, the estimation error increases to approximately $0.2 \,\mathrm{m}$ in normal driving scenarios. In contrast, in emergency scenarios, the estimation error exceeds 0.5 m, surpassing the threshold margin. This indicates that an annotation interval of $\delta t = 0.5 \,\mathrm{s}$ is insufficient. To ensure an estimation error within 0.2 m, as achieved when $\Delta t = 0.2$ s, a higher annotation frequency of $\delta t = 0.1 \text{ s}$ (10 Hz annotation) is required. This corresponds to a fivefold increase in annotation volume. Thus, the real-time constraint Δt significantly impacts the annotation cost to maintain an acceptable level of



Figure 6. The estimation error of $x_{t+\Delta t}^{GT}$ for inference time $\Delta t = 0.2$ s.

estimation error. Conversely, given the inference time δt and acceptable error in a system, the required annotation interval can be determined as a specification requirement.

Next, we analyze the estimation error of $v_{t+\Delta t}^{GT}$. The ground truth of the velocity v at the time $(t + \Delta t)$, $v_{t+\Delta t}^{*GT}$ is calculated by

$$v_{t+\Delta t}^{*GT} = v_t^{GT} + a_t^{GT}\Delta t + \frac{1}{2}j\Delta t^2.$$
 (7)

The error of the velocity E_y between $v_{t+\Delta t}^{*GT}$ and $v_{t+\Delta t}^{GT}$ is calculated by

$$E_v = v_{t+\Delta t}^{*GT} - \hat{v_t}^{GT}$$

$$= \frac{2a_{t-\delta t}^{GT}\delta t + j\delta t^2}{4} + a_t^{GT}\Delta t + \frac{j}{2}\Delta t^2$$
(8)
(9)

For each Δt and δt , we visualized the relationship between estimation error $E_y = (0.2 \text{ m/s}, 0.5 \text{ m/s}, 1.0 \text{ m/s}, 2.0 \text{ m/s})$ and the corresponding values of j and $a_{t-\delta t}^{GT}$.

In the nuScenes annotation, for velocity estimation errors at $\Delta t = 0.2$ s, the error remains within 0.2 m/s in normal scenarios and within 1.0 m/s in emergency scenarios. However, in emergency scenarios at $\Delta t = 0.5$ s, the velocity estimation error reaches approximately 1.0 m/s, which may negatively impact downstream tracking and planning algorithms. Additionally, if the velocity estimation accuracy of the model is comparable to this E_v , it becomes difficult to ensure a reliable evaluation of velocity estimation performance. For instance, if $E_v < 0.20 \text{ m/s}$ and the mean velocity estimation error from model inference is 0.3 m/s, it is reasonable to assume a velocity error of 0.5 m/s and incorporate this assumption into the parameter settings of downstream tracking and planning modules.

Note that, in practical cases, annotation itself contains position errors. As illustrated in Fig. 10, there are cases where only a part of a vehicle is visible in a given frame, and a 3D bounding box is annotated accordingly. In the next annotated frame, the entire vehicle becomes visible, leading to a different annotation. Since 3D bounding box annotations are based on LiDAR point clouds, occlusions or missing point clouds can result in incorrect annotations. In such cases, velocity is calculated using the position difference between frames as:

$$v^{an} = \frac{x_{t+\Delta t}^{an} - x_t^{an}}{\Delta t},\tag{10}$$

where the annotation position error x_t^{an} propagates into the velocity estimation v^{an} . For instance like Fig. 10, if the





Figure 8. The estimation error of $v^{GT}_{t+\Delta t}$ for inference time Δt = 0.2 s.



Figure 9. The estimation error of $v_{t+\Delta t}^{GT}$ for inference time Δt = 0.5 s.



Figure 10. The error of annotation.

annotation position error is 0.5 m and the annotation interval is 0.5 s (2 Hz annotation), the resulting velocity estimation error can reach 1.0 m/s. Therefore, it is crucial to take such annotation-induced errors into account when evaluating velocity estimation accuracy.