IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Adaptive-Cloud: Dynamic Computation Control for 3D Object Detection From LIDAR Point Clouds

Mir Sayeed Mohammad, Uday Kamal, and Saibal Mukhopadhyay, *Fellow, IEEE*

***Abstract*—In this work, we introduce an adaptive hierarchical framework for efficient 3D object detection from point cloud data, designed to dynamically balance computational efficiency and detection performance. Our approach employs a shared feature extractor and multiple detector backbones of varying widths, enabling selective activation of models based on the complexity of the input scene. A novel feature gating mechanism dynamically determines the most relevant features for reduced-width backbones, while a surrogate loss prediction module ranks models in real-time, ensuring optimal backbone selection with minimal overhead. This adaptive strategy reduces compute costs by 41.4% while maintaining a negligible 2.44% reduction in detection accuracy across a range of real-world driving scenes (urban, highway, residential, campus, person) from the KITTI dataset. By addressing runtime adaptability—a critical gap in existing 3D detection frameworks—our method provides a significant algorithmic improvement for high-performance detection models in resource-constrained environments.**

***Index Terms*—Object detection, path planning, real-time systems, edge AI, adaptive systems, computer vision.**

I. Introduction

THE rapid advancements in 3D object detection have played a pivotal role in autonomous systems such as self-driving cars, drones, and mobile robots. Among the different sensing modalities, LIDAR-generated point cloud data [1] provides a precise environment perception due to its rich spatial and geometric information. Processing the high dimensional and unstructured data efficiently presents significant computational challenges [2]. Existing 3D object detection frameworks rely on converting point cloud data into a voxel format, followed by feature extraction and detection in a neural network. While these methods have shown promising results in accuracy, they incur a high computational overhead, particularly in scenarios with dense environments and large point cloud volumes [3]. Traditional models also utilize the on-board resources at a fixed rate even in simple scenes where full model capacity might be unnecessary, necessitating adaptive frameworks that can dynamically adjust computation based on scene complexity.

Received 19 December 2024; accepted 10 April 2025. Date of publication 9 May 2025; date of current version 20 May 2025. This article was recommended for publication by Associate Editor J. Li and Editor A. Valada upon evaluation of the reviewers' comments. This work was supported in part by the CogniSense, one of seven centers in JUMP 2.0 and in part by DARPAthrough Semiconductor Research Corporation (SRC) Program. *(Corresponding author: Mir Sayeed Mohammad.)*

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30318 USA (e-mail: mirsayeed-mohammad@gatech.edu).

Digital Object Identifier 10.1109/LRA.2025.3568566

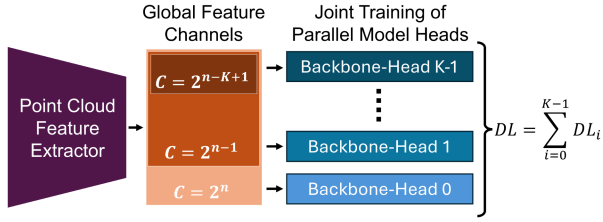


Fig. 1. Overview of our hierarchical feature learning for point cloud feature extraction in 3D object detection. Consecutively smaller backbone sub-networks use a smaller subset of features from the point cloud feature extractor. Total loss DL is calculated as the sum of individual model losses.

In this letter, we propose Adaptive-Cloud, a novel paradigm for efficient and adaptive computation in LIDAR-based 3D object detection. The key insight behind Adaptive-Cloud is a hierarchical feature learning method that dynamically selects feature subsets and model backbones of varying capacities as highlighted in Fig. 1 based on input scene, significantly reducing computation without sacrificing detection performance. By integrating a multi-scale feature extraction network and a surrogate loss ranking model, our approach balances computational efficiency and detection accuracy in real-time.

The key contributions of our work are as follows:

- **Hierarchical Feature Learning:** Our approach incorporates a shared feature extraction backbone with multiple detector heads of varying channel widths that enables adaptive feature usage.
- **Feature Gating Mechanism:** We choose feature subsets of reduced size to optimize computation in subsequent stages using a feature gating module to choose the most relevant features.
- **Surrogate Loss Ranking:** A surrogate loss ranking module works on the point cloud feature space to adapt to the current input scene and choose the most optimal backbone for reducing computation.
- **Runtime Adaptability:** Unlike existing static pruning methods that drop model weights during training or post training, our model retains all the weights and adapts to the scene during runtime instead to reduce computational overhead.

We show the application of the proposed approach on the KITTI [4] multi-object detection data set for autonomous driving and demonstrate state-of-the-art accuracy. Finally, we compare object detection performance and computational overhead of some of the existing single stage object detection methods. We show that our model can reduce 41.4% computational overhead

on average for only 2.44% reduction of mean average precision (mAP) during runtime. We also show the effects of computation reduction on downstream tasks, such as driving path planning.

II. Related Work

3D object detection from LIDAR point cloud data is a vital task for autonomous systems, offering precise depth perception compared to traditional camera-based methods. Monocular [5], [6] and stereo vision [7], [8], [9], [10] techniques often suffer from calibration sensitivity, limited information, or lighting conditions. LIDAR, though more robust to lighting, struggles with sparsity in distant objects and environmental noise, leading recent methods [11], [12], [13] to use multimodal fusion for performance gains.

Our focus is on optimizing LIDAR-only detection for low-power platforms, where real-time efficiency is critical. LIDAR sensors generate 3D point cloud data as an unordered set of points by emitting light pulses from laser arrays and measuring their return times. LIDAR point clouds, being unstructured, require conversion to a structured format such as voxels or pillars. Two dominant approaches exist: point-based (e.g., PointRCNN [14], 3D-SSD [15]), Graph-encoding [16] and voxel-based (e.g., VoxelNet [3], Cylinder3D [17]). Hybrid models like PV-RCNN [18], CenterPoint [19], and PointPillars [20] combine the benefits of both. Multi-stage detectors like PV-RCNN or Fast PointRCNN offer high accuracy but at the cost of latency and compute, making them less suitable for edge deployment. Single-stage detectors such as PointPillars, SA-SSD [21], and 3D-SSD prioritize speed, and our work builds on this family.

Numerous strategies have been proposed for optimizing 3D detection models: sparse convolution reduces redundancy by skipping zero-valued voxels, dynamic pruning [22] and weight pruning [23] selectively deactivate layers or regions, and knowledge distillation [24] transfers knowledge from large models to compact ones. However, these techniques primarily operate post-training and do not adapt model capacity dynamically during inference.

Unlike existing dynamic neural networks that reduce computation sample-wise [25], spatially [26], or temporally [27], our approach explores dynamic computation along the model width, specifically tailored for LIDAR data. Works like [28], [29] aim to dynamically compress point cloud videos by reducing temporal and spatial redundancy in point/voxel space, but do not handle the redundancy across the channels of model layers-an orthogonal direction for optimization. Early exit strategies such as BranchyNet [30], AdaDeT [31], and dynamic token halting [32] operate along the model depth by skipping deeper layers when possible. This is complementary to our work on reducing model width.

Recent works like MMFG [33] explore feature gating in multimodal settings, while HAPGN [34] and SAT3D [35] apply attention mechanisms to LIDAR segmentation. However, these attention-based gating mechanisms typically incur $O(n^2)$ complexity, making them less suitable for deployment on edge hardware. Additionally, while surrogate models have been used in applications such as safety performance prediction [36], and cooperative perception [37], [38], our work uniquely focuses on ranking perception quality using low-level feature representations. We do not statically prune or compress a single model. Instead, we propose a dynamic backbone selection strategy guided by scene complexity—an underexplored yet critical direction for efficient LIDAR-based detection systems.

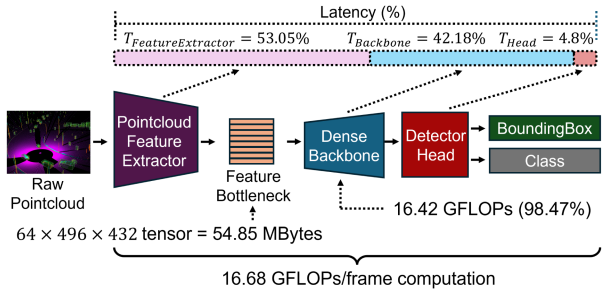


Fig. 2. Computation and latency in different modules of the PointPillars Architecture - the dense backbone of the network carries out most of the computation, leaving room for optimization in this part of the network.

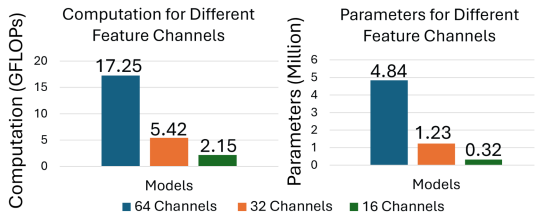


Fig. 3. Computation for different levels of bottleneck feature volume. Reducing the feature volume by 75% can save up to 87.5% compute.

III. Methodology

Our approach builds on the PointPillars architecture [20] for 3D object detection from LIDAR point clouds. PointPillars is a single-stage detector comprising three components: a pillar feature extractor, a dense convolutional backbone with FPN, and a detection head. The feature extractor converts raw point clouds into a 2D BEV representation with 64 channels, which the backbone processes before passing to the head for bounding box regression and classification.

As shown in Fig. 2, the backbone dominates computation, while the feature extractor and head together account for only 1.53% of total FLOPs. This highlights the potential for optimizing model efficiency by reducing feature volume and backbone width, as outlined in Fig. 3. Such scaling improves data rates and exponentially reduces computation in downstream components—a core motivation for our method. To enable adaptive computation, we restructure the pipeline into a hierarchical framework with three key modules: feature gating, parallel model backbones, and a surrogate loss predictor. Their interaction is illustrated in Fig. 4(a).

A. Network Architecture

We retain the original PointPillars feature extractor and share it across K downstream modules. Instead of forwarding the full 64-channel BEV feature to a single backbone, we introduce

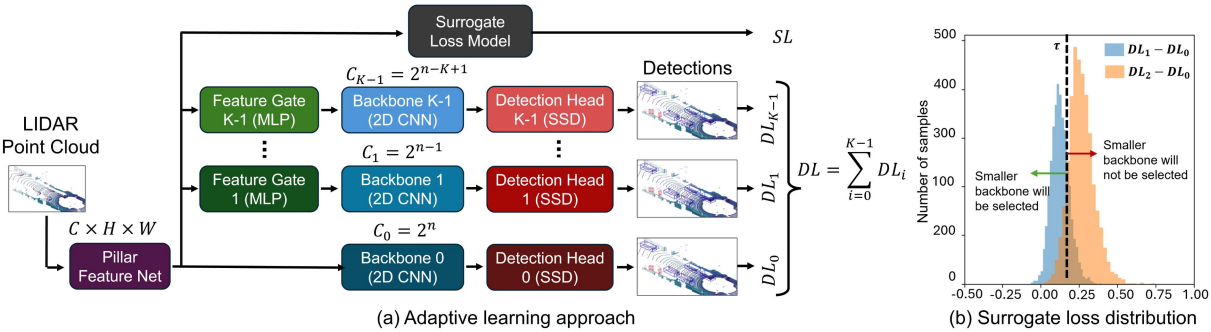


Fig. 4. (a) Hierarchical Point Cloud Representation Learning Scheme. Pillar feature network encodes the point cloud information into 2D birds eye view features with C channels. Gating is done on the encoded feature channels to extract the necessary subset of point cloud features that are passed to the remaining detector network. A surrogate loss prediction model tries to rank the performance of different models from the input features. The detection losses (DL) of each separate backbone-head pair are added together along with the prediction loss (SL) of the surrogate loss prediction model to obtain the total model loss. (b) Ground truth relative loss distribution for different samples after training. If the predicted relative loss is less than the parameter τ, smaller model is selected during inference. Otherwise, larger model is used.

parallel backbones of varying widths (e.g., $64/2^i, i\epsilon\{0, 1 \ldots K-1\}$), each paired with an independent detection head. Inspired by Matryoshka Representation Learning [39], we hypothesize that not all scenes require full feature capacity. Each backbone processes a subset of the original features and is trained jointly, enabling runtime switching based on scene complexity. Independent heads support flexible switching, while the shared extractor ensures efficient and consistent early-stage computation.

B. Feature Gating Module

To determine which subset of channels from the shared pillar features is most informative for the i-th ($i\epsilon\{0, 1 \ldots K-1\}$) reduced-width backbone, we introduce a feature gating module. This module analyzes the full 64-channel feature map and selects the most relevant C_i channels (where $C_i = 64/2^i$) for each backbone path. Specifically, a small convolutional network followed by an MLP generates a channel importance score vector:

$$\mathbf{s} = g(\mathbf{X}) \in \mathbb{R}^{64}$$

where $\mathbf{X} \in \mathbb{R}^{64\times H\times W}$ is the shared feature map. s_j is the score for the j-th channel. We define mask $\mathbf{m} \in \{0, 1\}^{64}$:

$$m_j = \begin{cases} 1, & \text{if } s_j \text{ is among the top } C_i \text{ values in } \mathbf{s}, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathcal{J} = \{j \mid m_j = 1\}$ denote the set of selected channel indices. The output feature map is then obtained by gathering the corresponding channels from $\mathbf{X}$:

$$\mathbf{X}' = \{\mathbf{X}_j\}_{j\in\mathcal{J}} \in \mathbb{R}^{C_i\times H\times W}.$$

This gated feature subset is then forwarded to the corresponding reduced backbone, ensuring each model receives the most relevant information for detection.

C. Surrogate Loss Prediction Model

Feature gating helps customize the input to each backbone, but choosing the optimal backbone for a given scene still requires inference. To avoid evaluating all backbones during runtime, we introduce a lightweight surrogate loss prediction model. This model takes the full 64-channel pillar feature map $\mathbf{X} \in \mathbb{R}^{64\times H\times W}$ as input and predicts the expected detection loss of each backbone relative to the full model, i.e., $RD_{pred,i} = \overline{DL_i - DL_0}$. Here $RD_{pred,i}$ is the relative detection loss prediction between the i-th head and 0-th head. During inference, user-defined parameter, τ (refer to Fig. 4(b)) controls the selection of the i*-th backbone.

$$i^* = \begin{cases} \min\{i \in \{1, \ldots, K-1\} \mid \\ \quad RD_{pred,i} < \tau\}, & \text{if such } i \text{ exists} \\ 0, & \text{otherwise} \end{cases}$$

A higher tolerance allows frequent use of low-computation backbones, saving resources at the cost of some detection accuracy. This balances performance and efficiency dynamically. The surrogate model enables early model selection, before activating any heavy backbone based on the scene complexity embedded in the shared features. The results in Table I are reported for EfficientNetB0 [40] surrogate model with 1.345 GFLOPs compute for the given feature volume.

D. Training Scheme

1) Defining the Loss: Each model head independently predicts detection outputs. The detection loss for the i-th head is defined as a weighted sum of classification, regression, and direction losses:

$$DL_i = (w_1 L_{\text{class}} + w_2 L_{\text{reg}} + w_3 L_{\text{dir}})_i$$

$$DL = \sum_{i=0}^{K-1} DL_i$$

DL is the total loss across K parallel heads. For adaptive inference, we train a surrogate model to predict the relative detection loss between each lightweight head and the full model (head 0). The ground truth relative loss is defined as:

$$RD_{\text{gt}} = \{DL_i - DL_0;\ 1 \le i \le K-1\}$$

TABLE I
3D OBJECT DETECTION RESULTS (CARS-EASY, KITTI)

Model	Compute GFLOPs	Params (M)	Latency (ms)	mAP (%)
VoxelNet [3]	-	-	33.0	77.82
Dynamic VoxelNet [41]	34.45	4.58	94.3	78.83
SECOND [42]	34.45	5.30	100.5	82.02
ContFuse [43]	-	-	-	83.68
SP-CenterPoint [22]	34.30	-	-	84.90
CenterPoint [19]	62.90	-	76.9	85.00
SA-SSD [21]	34.92	5.33	99.0	88.75
HybridPillars [44]	31.70	7.10	45.8	90.06
CasA++ [45]	47.50	11.32	86.0	90.68
TSSTDet [46]	-	-	-	91.84
3ONet [47]	-	23.10	136.8	92.03
PointPillars (Baseline) [20]	16.68	4.81	35.8	82.58
Shared Feature Extractor (Ours)				
H. PointPillars 64C	17.25	4.84	34.4	84.04
H. PointPillars 32C	5.42	1.23	38.3	72.34
H. PointPillars 16C	2.15	0.32	35.5	70.26
Adaptive Backbone Selection (Ours)				
Adaptive-Cloud (Min)	**8.10**	**10.42**	**46.1**	**80.84**
Adaptive-Cloud (Max)	**13.83**	**10.42**	**45.4**	**83.28**

Params: Number of model parameters (in millions).
Latency (ms): Model latency (in milliseconds).
mAP: Mean Average Precision (%) for 3D bounding box detection on the KITTI car (easy) split.
H. PointPillars: Hierarchical PointPillars (feature widths-C).
Adaptive: Dynamically selects backbones during inference.

$$SL = \sum e^{-|RD_{\text{gt}} \cdot a|} \cdot |RD_{\text{gt}} - RD_{\text{pred}}|$$

The surrogate model estimates these values as RD_{pred}. The surrogate loss is then defined as a weighted L1-distance. Here, the exponential weight (controlled by hyperparameter $a = 2$) prioritizes samples where detection heads perform similarly (i.e., $RD_{\text{gt}} \approx 0$), since these allow greater flexibility in runtime backbone selection. The surrogate model is not penalized for whether a lightweight model performs better or worse—it only learns to accurately predict the relative ranking. The final training loss is defined as:

$$\text{Loss} = DL + SL$$

2) Training Strategy: We begin with end-to-end training from scratch. Once converged, we freeze the pillar encoder and fine-tune each backbone and its gating module. Finally, we freeze all modules except the surrogate model and fine-tune it to accurately predict the relative loss differences.

IV. RESULTS

A. Experiments

We evaluate our method on KITTI object detection dataset [4], which contains 3712 training and 3769 validation samples across diverse driving scenes such as urban, residential, campus, and highway. Each sample includes stereo camera images and 64-channel Velodyne LIDAR data. We preprocess the point cloud to retain only the forward-facing region for detection. We also test the impact of detection quality on downstream path planning using an offline planner on the same dataset. Perception-planning latency tests are done on NVIDIA Tesla T4 unit and a Ryzen-7 CPU.

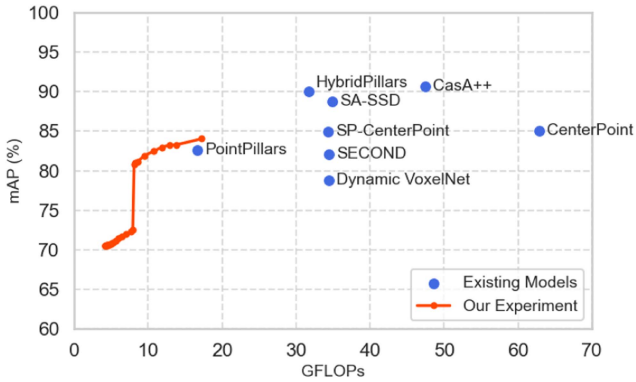


Fig. 5. Computation vs. 3D AP for various single-stage object detection architectures. We sweep the tunable parameter τ across its full range. The red curve spans configurations from the most lightweight (lower left) to the most accurate but compute-intensive (upper right), corresponding to models with minimal and maximal backbone usage, respectively.

Our model includes 3 parallel backbones using 64, 32, and 16 channel subsets (i.e., $K = 3$ in Fig. 4(a)). In the ablation studies (Table IV), we compare the performance across these variants. Naïvely reducing model width degrades accuracy, but sharing a common 64-channel feature extractor improves generalization for the smaller models. Further gains are achieved by introducing feature gating, which enables each backbone to focus on its most informative subset.

We also test several surrogate model architectures and loss functions, finding that deeper models like EfficientNetB0 [40] outperform simple MLPs (as used in [48]) for predicting detection loss differences.

B. 3D Object Detection Performance

We compare our adaptive framework against state-of-the-art single-stage 3D detectors in terms of accuracy, computational overhead, and memory usage. Table I summarizes these results. Initially, we report the performance of baseline models, followed by our hierarchical variants using a shared feature extractor and separate backbones. Lastly, we present the performance under varying tolerance levels of the surrogate model controlling backbone selection at inference.

Average computation ranges from 8.10 to 13.83 GFLOPs, with corresponding mAP values between 80.84 and 83.28. The full 64-channel backbone yields the highest accuracy, while the 16-channel variant underperforms due to limited feature representation. Our adaptive strategy strikes a balance by using smaller backbones in simpler scenes and switching to the full model in complex cases. Fig. 5 illustrates the trade-off between computation and detection accuracy as the tolerance parameter is varied. Notably, accuracy drops sharply when average FLOPs fall below 8 GFLOPs, as the smallest backbone dominates and introduces more false detections (further explained in Section IV-D).

In Table II, we see that for reducing model sizes, the mAP decreases rapidly in the pedestrian class. But our adaptive model retains detection performance for both cyclist and pedestrian classes with negligible drops in mAP. The adaptive method

TABLE II
MAP(%) COMPARISON ACROSS OBJECT CATEGORIES

Model	Car	Cyclist	Pedestrian
Shared Feature Extractor (Ours)			
H. PointPillars 64C	84.04	86.99	52.12
H. PointPillars 32C	72.34	84.65	42.80
H. PointPillars 16C	70.26	83.77	40.27
Adaptive Backbone Selection (Ours)			
Adaptive-Cloud (Min)	**80.84**	**86.99**	**51.97**
Adaptive-Cloud (Max)	**83.28**	**86.67**	**53.11**

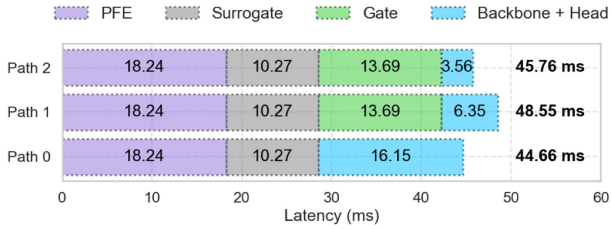


Fig. 6. Latency breakdown of the 3 path options using backbone 0, 1 and 2 respectively in the adaptive pipeline on Tesla T4 unit. Note that feature gating is not utilized in execution of the largest backbone (full feature-set).

deploys larger backbones in safety critical dense object situations, ensuring a lower false positives even for less average computation.

Fig. 6 shows the latency in different parts of the network for the three possible computational routes. We can see that the smaller backbones have lower latency, but because of the feature gates, the overall latency of the possible computation routes remain almost similar. This implies the adaptive pipeline can introduce computational savings without significant latency overhead.

C. Effect of Object Detection Performance on Path Planning

In this section, we evaluate how adaptive 3D object detection barely impacts downstream path planning performance in autonomous driving. Using the Frenet Optimal Trajectory algorithm [49], a simulated vehicle follows the KITTI ground truth trajectory while avoiding obstacles detected from the perception pipeline. The planning algorithm requires the global trajectory, local position of the ego vehicle, obstacle map (detector outputs), and a desired target velocity along with hyperparameters related to the vehicle description. The planned trajectory is compared with the ground truth obstacles to check for potential collisions.

In Table III, we report collision rates and lateral deviation for different target velocities and compare our adaptive model to the baseline 64-channel detector. Collision rate is defined as the percentage of frames where planned trajectories intersect with ground truth obstacles. Lateral deviation measures the average distance from the ground truth path. As shown in Fig. 7, at higher speeds, the ego vehicle adjusts its path more aggressively around merging objects. The adaptive model consistently shows lower collision rates despite slightly higher deviation. This is attributed to its looser bounding boxes, which create safer margins for planning. When the baseline bounding boxes are padded with +0.1 m safety region, the planning performance improves for the baseline too, yet the adaptive model achieves competitive performance for less computation.

TABLE III
PATH PLANNING PERFORMANCE

Vel	Collision (%)			Deviation (m)			Latency (ms)			T_r
km/h	BL	BL*	Adp	BL	BL*	Adp	Min	Max	Avg	(ms)
10.0	0.00	0.00	0.00	0.31	0.31	0.31	43.2	96.4	63.9	2750
12.5	0.03	0.00	0.03	0.31	0.31	0.31	42.1	99.8	64.5	2200
15.0	0.57	0.40	0.07	0.32	0.32	0.32	41.9	99.7	65.9	1830
17.5	1.34	1.14	1.21	0.32	0.32	0.33	57.1	136.7	92.0	1570
20.0	3.36	2.18	2.62	0.36	0.36	0.37	59.9	144.2	96.1	1380
22.5	3.53	2.35	3.73	0.42	0.43	0.42	54.7	131.3	92.8	1220
25.0	6.01	3.59	3.83	0.45	0.47	0.46	56.8	151.4	94.1	1100

BL: PointPillars model using a fixed 64-channel backbone.
BL*: BL detections with a +0.1m padding around boundingbox.
Adp: Adaptive backbone selection at runtime.
Collision (%): Percentage of frames with collision trajectories.
Deviation (m): Average lateral deviation from the GT trajectory.
Latency (ms): Framewise min, max, and average latency.
$\mathbf{T_r}$: Safety constraint for perception-to-planning loop, based on worst case stopping distance of 7.64m from KITTI dataset.

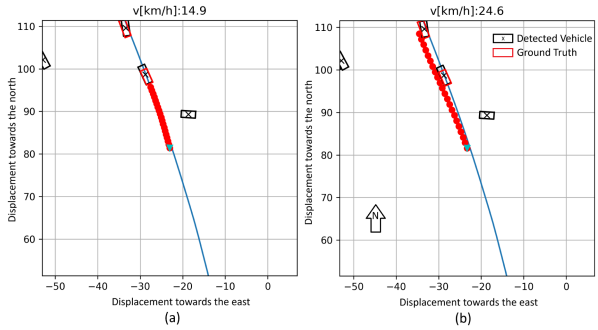


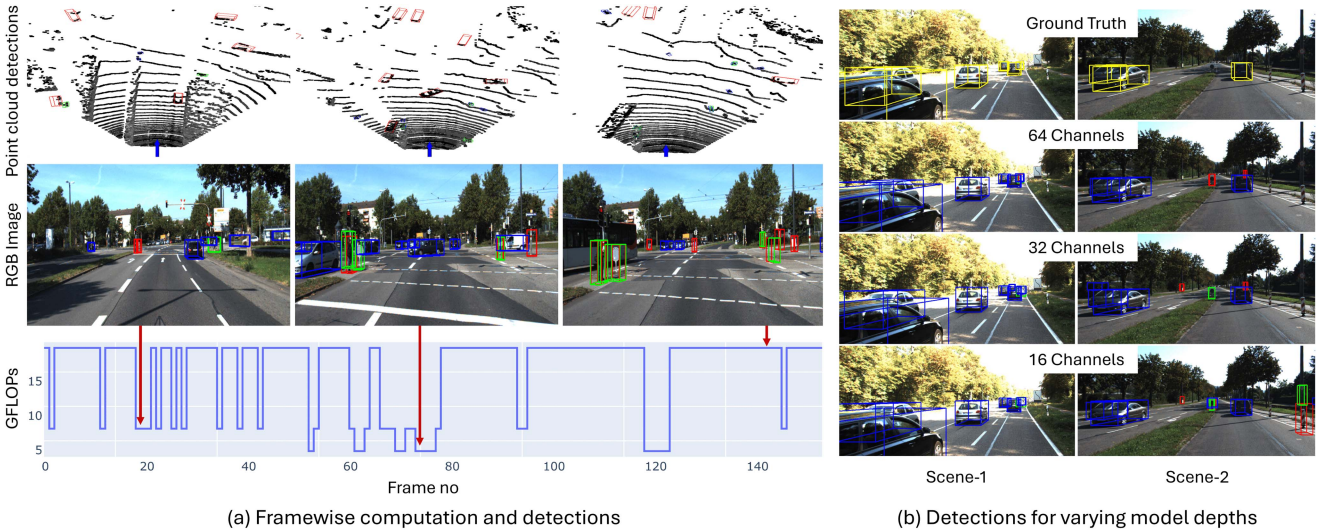
Fig. 7. Path planning in the driving scene for different target velocities of (a) 15 km/h (b) 25 km/h.

For testing real-time applicability, we take the reaction time as the stopping distance to the nearest vehicle divided by the velocity of the ego vehicle. For the highest test velocity at 25 km/h, the maximum reaction time is found to be 1100 ms for the KITTI dataset. We show that the full loop runs within the reaction time upper bound, with a worst case latency of 46.1 ms (Table I) + 151.4 ms (Table III) = 197.5 ms. While we run the planner offline on the KITTI dataset due to experimental limitations, our core finding is that adaptive detection preserves planning performance in real-time, making it suitable for real-world integration.

D. Qualitative Results

Fig. 8(a) shows the dynamic computation in continuous driving scenes. In early frames with few objects, the smallest model is used since larger and smaller models perform similar. As more vehicles appear closer to the ego vehicle, detection improves even with reduced channels. In later frames, distant objects need the largest model, as smaller models cannot detect them reliably. Scene-1 of Fig. 8(b) shows that larger objects are detected accurately across all widths. Vehicle-class objects rarely have false negatives, ensuring reliability in urban and highway scenarios. While tuning the NMS threshold could reduce false positives, we kept it constant in our experiments for simplicity.

In Fig. 9, we see that feature relevance changes with the input scene, necessitating the need for a feature gating module. We also notice that the two different backbones utilize the feature channels independently. Allowing for feature independence helps the



(a) Framewise computation and detections (b) Detections for varying model depths

Fig. 8. (a) Runtime computation at different timeframes. The larger model captures objects that are further away, whereas the smaller model captures objects that are closer to the ego vehicle. This enables scene dependent model selection and adaptive computation. (b) For varying model widths, in scene-1, only vehicles close to the ego vehicle are successfully detected despite reduced channel usage, giving less number of false positive detections. In scene-2, small objects of pedestrian and cyclist classes, and objects further away from the sensor generate more false detections in reduced number of channels.

TABLE IV
ABLATION STUDY ON FEATURE SELECTION STRATEGIES IN HIERARCHICAL POINTPILLARS

Training Scheme	Compute	Params	Birds Eye View BBox (mAP-Car)			3D BBox (mAP-Car)			3D BBox (mAP-3 Class)		
	GFLOPs	Million	Easy	Medium	Hard	Easy	Medium	Hard	Easy	Medium	Hard
Baseline PointPIllars	17.1635	4.835	89.0716	83.5149	72.4902	84.3744	66.1146	55.4714	74.2610	54.6611	48.5299
PointPillars 32C	4.496	1.217	86.8854	72.0485	68.3678	66.0029	49.9175	40.4964	50.4930	35.9390	30.0419
PointPillars 16C	1.227	0.309	84.5329	69.9189	65.9489	56.7212	40.7338	37.2514	45.1259	30.2942	28.1003
Ours (w/o gating) 64C	17.1635	4.835	88.4251	82.6212	71.6003	82.4792	56.6701	52.4682	73.5228	51.3505	47.4952
Ours (w/o gating) 32C	4.496	1.217	87.9506	82.6170	71.4959	71.9505	55.0457	51.3291	69.5679	50.2244	46.6178
Ours (w/o gating) 16C	1.227	0.309	87.4919	70.7406	67.0832	68.6458	43.1177	39.0694	59.4255	41.0386	35.8110
Ours (w gating) 64C	17.1635	4.835	88.4008	83.8012	80.6011	84.0366	66.8784	56.2205	74.3863	54.5583	48.6982
Ours (w gating) 32C	5.42	1.231	88.2485	82.3146	71.1266	72.3370	54.6106	50.7251	66.5958	49.3573	44.0282
Ours (w gating) 16C	2.151	0.323	87.2663	72.4787	68.9499	70.2569	51.8630	41.7364	64.7629	45.0128	39.5820

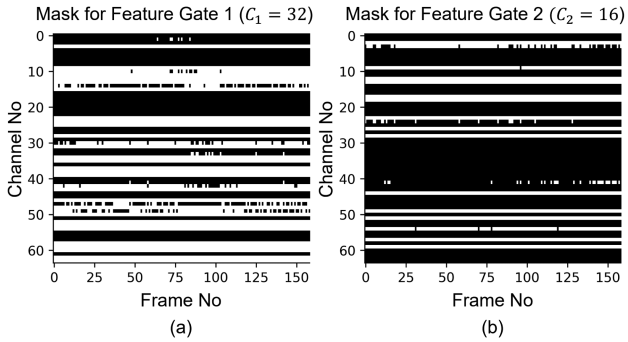


Fig. 9. Gate activation maps for (a) 32 and (b) 16 channels in the scene from Fig. 8(a). Each backbone selects its own feature subset; most channels remain consistent after convergence, while a few vary based on scene-specific content.

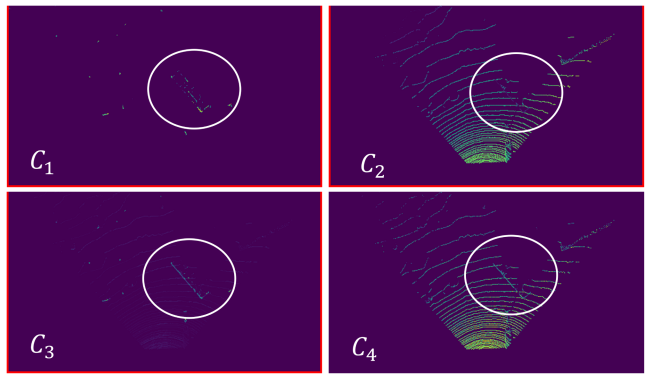


Fig. 10. BEV projection of 4 feature channels from one frame of Fig. 9(b). C_1 and C_2 are consistently selected, C_3 is selected occasionally, and C_4 is consistently ignored. C_2 covers background (ground plane), while C_1 and C_3 highlight objects. C_4 has redundant mixture of multiple features.

feature extractor to learn better, and improves the performance of the hierarchical approach as shown in Table IV. Fig. 10 shows the actual information being passed/blocked by the gating module.

Figs. 8(b) and 10 illustrate how false positives and negatives arise across model backbones of different widths. As described in [20], the 64-dimensional pillar feature vectors are generated by aggregating all points in a vertical grid via a PointNet [50], forming a BEV feature map. Some feature channels learn to highlight objects of interest, while others capture background elements like ground or trees. Large objects- even when spatially downsampled in the encoder, retain more features in the important channels. The full 64-channel backbone processes all features—including noise—leading to better detections. In contrast, the 16-channel model uses gated, informative features,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

missing details of small or distant objects due to spatial downsampling and limited input. This trade-off explains why scenes with lot of objects or distant objects benefit from the larger backbone, while scenes with closer objects perform well even with the smaller backbone. Thus the larger backbone is used in more complex situations, while the small model takes over the simpler driving scenes.

E. Ablation Studies

We summarize the progressive development of the adaptive hierarchical learning approach in Table IV. The baseline PointPillars [20] model achieves a mAP of 84.37 for the KITTI cars-easy class and an average mAP of 74.26 across cars, pedestrians, and cyclists, but offers no runtime computation optimization. Reducing the number of channels in the pillar feature extractor (PFE) and using smaller backbones in the next two rows degrades detection performance, emphasizing the challenge of downsizing without sacrificing accuracy.

Next, we explore joint optimization of three model backbones using a shared 64-channel feature extractor. For the 32- and 16-channel models, subsets of the original features are selected, representing half and a quarter of the channels, respectively. This setup boosts the performance of smaller models by 8%-10%, with a marginal 1.9% drop in mAP for the largest backbone. These results highlight the benefit of joint optimization in enabling smaller models to learn better while maintaining accuracy in larger backbones.

Feature gating further enhances performance by dynamically selecting the most informative channels for the smaller models. This refinement improves accuracy by 1%–2% for the 32- and 16-channel backbones. Finally, the addition of the surrogate loss prediction model allows the framework to adaptively select the optimal backbone based on scene complexity. Sweeping the tolerance parameter during inference, as illustrated in Fig. 5, demonstrates the trade-off between computational efficiency and detection performance, showcasing the effectiveness of this adaptive approach.

Finally, we implemented our network on an Edge AI platform (Nvidia Jetson Nano) to evaluate it's feasibility for real-time application. Using 16 b floating-point TensorRT optimization, the largest path runs at 3.4 fps on 3.89 Watts (1.069 J/sample, 294.23 ms) and the smallest path runs at 6.7 fps on 3.63 Watts (0.557 J/sample, 143.35 ms). The adaptive approach minimizes energy consumption from 0.944 J to 0.576 J, enabling a 38.93% energy savings.

F. Limitations

The proposed computation reduction method relies on a preset parameter to balance performance and efficiency at runtime, without directly optimizing for downstream path planning. We used identical pruned paths for all backbones, though custom architectures for smaller models may improve results. Scalability on edge devices may be limited due to memory overhead and latency constraints, which could be addressed through parameter sharing. Spatio-temporal optimization and model distillation could improve the performance of smaller backbones. Lastly, while the method works well on KITTI, broader adaptability to diverse scenes and integration with closed-loop planning remain areas for future work.

V. Conclusion

This work presents an adaptive hierarchical feature learning method optimized for low-power applications in autonomous driving, achieving 41.4% computational savings with only a 2.44% reduction in detection accuracy. By employing adaptive model backbones and a surrogate loss prediction model, the approach dynamically adjusts the computation during runtime, demonstrating benefits for downstream tasks like path planning. The framework currently focuses on optimizing the detection stage, and closed-loop perception-planning integration can be done to further improve efficiency and robustness in autonomous systems.

References

[1] Y. Li and J. Ibanez-Guzman, "LiDAR for autonomous driving: The principles, challenges, and trends for automotive LiDAR and perception systems," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 50–61, Jul. 2020.
[2] Y. Li et al., "Deep learning for LiDAR point clouds in autonomous driving: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3412–3432, Aug. 2021.
[3] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4490–4499.
[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[5] S. Zhu and X. Liu, "LightedDepth: Video depth estimation in light of limited inference view angles," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 5003–5012.
[6] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 10371–10381.
[7] G. Xu, J. Cheng, P. Guo, and X. Yang, "Attention concatenation volume for accurate and efficient stereo matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12981–12990.
[8] L. Lipson, Z. Teed, and J. Deng, "RAFT-stereo: Multilevel recurrent field transforms for stereo matching," in *Proc. Int. Conf. 3D Vis.*, 2021, pp. 218–227.
[9] X. Cheng et al., "Hierarchical neural architecture search for deep stereo matching," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 22158–22169.
[10] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr, "GA-Net: Guided aggregation net for end-to-end stereo matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 185–194.
[11] H. Wu, C. Wen, S. Shi, X. Li, and C. Wang, "Virtual sparse convolution for multimodal 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 21653–21662.
[12] X. Li et al., "Logonet: Towards accurate 3 D object detection with local-to-global cross-modal fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17524–17534.
[13] Y. Tian et al., "ACF-Net: Asymmetric cascade fusion for 3D detection with LiDAR point clouds and images," *IEEE Trans. Intell. Veh.*, vol. 9, no. 2, pp. 3360–3371, Feb. 2024.
[14] S. Shi, X. Wang, and H. Li, "PointrCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 770–779.
[15] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11040–11048.
[16] P. D. O. Rente, C. Brites, J. Ascenso, and F. Pereira, "Graph-based static 3D point clouds geometry coding," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 284–299, Feb. 2019.

[17] X. Zhu et al., "Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9939–9948.

[18] S. Shi et al., "Pv-rCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10529–10538.

[19] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11784–11793.

[20] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12697–12705.

[21] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3D object detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11873–11882.

[22] J. Liu, Y. Chen, X. Ye, Z. Tian, X. Tan, and X. Qi, "Spatial pruned sparse convolution for efficient 3D object detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 6735–6748.

[23] K. Xu et al., "DM3D: Distortion-minimized weight pruning for lossless 3D object detection," 2024, *arXiv:2407.02098*.

[24] J. Yang, S. Shi, R. Ding, Z. Wang, and X. Qi, "Towards efficient 3D object detection with knowledge distillation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 21300–21313.

[25] Y. Wang et al., "Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference," *IEEE J. Sel. Top. Signal Process.*, vol. 14, no. 4, pp. 623–633, May 2020.

[26] W. Xia, H. Yin, X. Dai, and N. K. Jha, "Fully dynamic inference with deep neural networks," *IEEE Trans. Emerg. Top. Comput.*, vol. 10, no. 2, pp. 962–972, Apr.–Jun. 2022.

[27] Y.-D. Zheng, Z. Liu, T. Lu, and L. Wang, "Dynamic sampling networks for efficient action recognition in videos," *IEEE Trans. Image Process.*, vol. 29, pp. 7970–7983, 2020.

[28] Y. Zhou, X. Zhang, X. Ma, Y. Xu, K. Zhang, and L. Zhang, "Dynamic point cloud compression with spatio-temporal transformer-style modeling," in *Proc. Data Compression Conf.*, 2024, pp. 53–62.

[29] C. Chen, S. Xie, J. Liu, Z. Cheng, Z. Zeng, and H. Zhuang, "Efficient point cloud video recognition via spatio-temporal pruning for MEC-based consumer applications," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 4108–4119, Feb. 2024.

[30] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit.*, 2016, pp. 2464–2469.

[31] L. Yang, Z. Zheng, J. Wang, S. Song, G. Huang, and F. Li, "AdaDet: An adaptive object detection system based on early-exit neural networks," *IEEE Trans. Cogn. Devel. Syst.*, vol. 16, no. 1, pp. 332–345, Feb. 2024.

[32] M. Ye, G. P. Meyer, Y. Chai, and Q. Liu, "Efficient transformer-based 3D object detection with dynamic token halting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 8438–8450.

[33] W. Xu and Z. Fu, "MMFG: Multimodal-based mutual feature gating 3D object detection," *J. Intell. Robotic Syst.*, vol. 110, no. 2, 2024, Art. no. 85.

[34] C. Chen, S. Qian, Q. Fang, and C. Xu, "HAPGN: Hierarchical attentive pooling graph network for point cloud segmentation," *IEEE Trans. Multimedia*, vol. 23, pp. 2335–2346, 2021.

[35] M. Ibrahim, N. Akhtar, S. Anwar, and A. Mian, "SAT3D: Slot attention transformer for 3D point cloud semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5456–5466, May 2023.

[36] Y. Wang, R. Yu, S. Qiu, J. Sun, and H. Farah, "Safety performance boundary identification of highly automated vehicles: A surrogate model-based gradient descent searching approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 23809–23820, Dec. 2022.

[37] K. Qu, W. Zhuang, Q. Ye, W. Wu, and X. Shen, "Model-assisted learning for adaptive cooperative perception of connected autonomous vehicles," *IEEE Trans. Wireless Commun.*, vol. 23, no. 8, pp. 8820–8835, Aug. 2024.

[38] Q. Xie, X. Zhou, T. Qiu, Q. Zhang, and W. Qu, "Soft actor–critic-based multilevel cooperative perception for connected autonomous vehicles," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21370–21381, Nov. 2022.

[39] A. Kusupati et al., "Matryoshka representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 30233–30249.

[40] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2019, pp. 6105–6114.

[41] Y. Zhou et al., "End-to-end multi-view fusion for 3D object detection in LiDAR point clouds," in *Proc. Conf. Robot Learn.*, 2020, pp. 923–932.

[42] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3337.

[43] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 641–656.

[44] Z. Huang, Y. Huang, Z. Zheng, H. Hu, and D. Chen, "Hybridpillars: Hybrid point-pillar network for real-time two-stage 3D object detection," *IEEE Sensors J.*, vol. 24, no. 22, pp. 38318–38328, Nov. 2024.

[45] H. Wu, J. Deng, C. Wen, X. Li, C. Wang, and J. Li, "CasA: A cascade attention network for 3-D object detection from LiDAR point clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–11, 2022.

[46] H. A. Hoang, D. C. Bui, and M. Yoo, "TSSTDet: Transformation-based 3-D object detection via a spatial shape transformer," *IEEE Sensors J.*, vol. 24, no. 5, pp. 7126–7139, Mar. 2024.

[47] A. Hiep Hoang and M. Yoo, "3ONet: 3-D detector for occluded object under obstructed conditions," *IEEE Sensors J.*, vol. 23, no. 16, pp. 18879–18892, Aug. 2023.

[48] I. Kim, Y. Kim, and S. Kim, "Learning loss for test-time augmentation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 4163–4174.

[49] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 987–993.

[50] C. R. Qi, H. Su, K. Mo, and L. J Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.