NVAutoNet: Fast and Accurate 360° 3D Visual Perception For Self Driving

Trung Pham[*], Mehran Maghoumi, Wanli Jiang, Bala Siva Sashank Jujjavarapu, Mehdi Sajjadi,
Xin Liu, Hsuan-Chu Lin, Bor-Jeng Chen, Giang Truong, Chao Fang, Junghyun Kwon, Minwoo Park
NVIDIA

Abstract

Achieving robust and real-time 3D perception is fundamental for autonomous vehicles. While most existing 3D perception methods prioritize detection accuracy, they often overlook critical aspects such as computational efficiency, onboard chip deployment friendliness, resilience to sensor mounting deviations, and adaptability to various vehicle types. To address these challenges, we present NVAutoNet: a specialized Bird's-Eye-View (BEV) perception network tailored explicitly for automated vehicles. NVAutoNet takes synchronized camera images as input and predicts 3D signals like obstacles, freespaces, and parking spaces. The core of NVAutoNet's architecture (image and BEV backbones) relies on efficient convolutional networks, optimized for high performance using TensorRT. Our image-to-BEV transformation employs simple linear layers and BEV look-up tables, ensuring rapid inference speed. Trained on an extensive proprietary dataset, NVAutoNet consistently achieves elevated perception accuracy, operating remarkably at 53 frames per second on the NVIDIA DRIVE Orin SoC. Notably, NVAutoNet demonstrates resilience to sensor mounting deviations arising from diverse car models. Moreover, NVAutoNet excels in adapting to varied vehicle types, facilitated by inexpensive model fine-tuning procedures that expedite compatibility adjustments.

1. Introduction

Autonomous vehicles must accurately understand their 3D surroundings, but achieving this by camera sensors only is complex. In early days, individual monocular camera perception modules are run separately, and their outputs are then merged to create a unified 3D picture. However, this approach faces challenges. Errors from independent monocular modules, like under or overestimation, vary and their error models are often unknown. Consequently, merging these results becomes tricky, often causing false positives. Nonetheless, maintaining camera independence enhances product safety by reducing shared failure risks.

Another vital consideration is production scalability. Car manufacturers typically produce various car models like SUVs, sedans, sport cars, and trucks, each with distinct sizes and camera positions. Thus, a camera perception system that can handle diverse camera angles, positions, radial distortions, and focal lengths becomes essential for scalability. Equally important, the system must operate in real-time, within a low-powered, shared compute budget on a System on Chip (SoC), especially since other independent radar or ultrasonic modules might run concurrently.

Design principles: We aim to design a perception network with the following principles: (1) *Precise 3D Perception*: The projected 3D predictions seamlessly align with the content of 2D image views. (2) *Extended Range Detection*: The network excels in detecting objects at considerable distances, reaching up to 200 meters. (3) *Efficiency at Its Core*: Operating in real-time on edge devices, such as the NVIDIA DRIVE Orin SoC, the network ensures seamless responsiveness. (4) *Resilience to Camera Variability*: The network maintains robust functionality even in the presence of camera dropouts. (5) *Holistic Machine Learning*: Requiring no post-processing, the network's performance scales effectively with the volume of data. (6) *Adaptive Scalability*: Straightforward customization for diverse vehicle platforms underscores the network's scalability.

Key components and features: Guided by the aforementioned design principles, we introduce NVAutoNet, a camera perception network featuring the following essential components and attributes. (1) CNN based image feature extractors are meticulously tailored using hardware-aware neural architecture search (NAS) to achieve both high accuracy and low latency. (2) Multi-camera fusion occurs at the BEV level, combining the strengths of early and late fusion methods. (3) Perspective-to-BEV view transformation is efficiently executed through column-wise MultiLayer Perceptron (MLP) layers and BEV look-up tables. (4) All perception tasks, including freespace perception, are structured as set prediction tasks, streamlining processes and negating the need for resource-intensive post-processing like clustering, boundary extraction, and curve fitting.

[*]corresponding author: trungp@nvidia.com

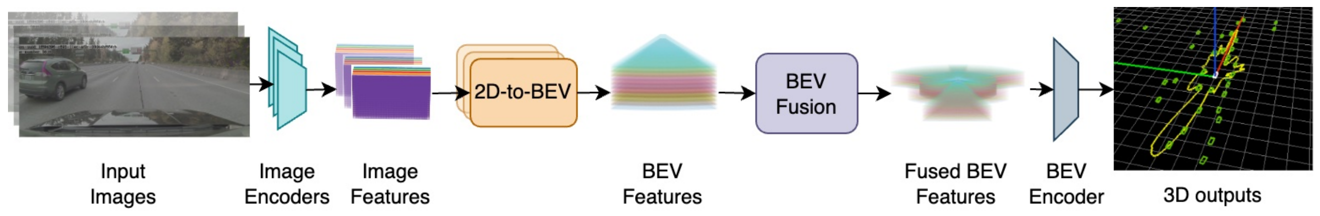

Figure 1. NVAutoNet overview. Surround view images are input to CNN backbones to extract 2D features, which are uplifted and fused into a unified BEV feature map. The generated BEV features are then encoded by a BEV backbone. Finally 3D signals are predicted by various 3D perception heads.

2. Related Work

Perception for Self-Driving. In the realm of self-driving, the perception module's responsibility encompasses the detection and identification of all static and moving objects in the surrounding environment. This roster includes obstacles, road markings, lanes, road boundaries, traffic lights, traffic signs, parking spaces, free spaces, and more. Existing research has predominantly focused on the 3D object detection task, with notable methodologies like DETR3D [25], PETR [9], BEVFormer [8], BEVDet [4], and BEVDepth [6]. In contrast, our NVAutoNet network tackles multiple tasks concurrently, including 3D obstacles, 3D freespaces, and 3D parking spaces. Modern 3D object detection methods have embraced set prediction approaches to obviate the need for NMS post-processing, a trend that NVAutoNet seamlessly follows. Recently, the domain of online mapping perception [10, 16] has also attracted considerable attention. NVAutoNet is well-equipped to extend its capabilities to address these tasks with ease.

Multi-camera Fusion and BEV Perception. Monocular camera perception tasks are well-established in computer vision, but multi-camera perception has recently gained attention [1, 13, 15, 18, 21, 22, 25]. Current methods, including ours, have shifted toward mid-level fusion, where information from various cameras is fused at the feature level. This demands a shared representation, often achieved through the BEV approach. BEV representation is advantageous for fusing data from multiple sensors and timestamps and can be directly used by downstream tasks like prediction, planning, and control.

Perspective to 3D/BEV View Transformation. Converting a perspective (image) view into a 3D/BEV perspective presents a challenging problem. Existing methods fall into four categories: homography-based, depth-based, MLP-based, and attention-based approaches. Homography-based methods (e.g., [18]) assume a flat-world model to shift pixels from a perspective view to a BEV view. However, this assumption often doesn't hold in real-world autonomous driving scenarios. Depth-based methods (e.g., [15,17,23]) require per-pixel depth information for transformation, but they face challenges. Inaccurate predicted depth can lead to poor-quality 3D feature maps, and the method's efficiency is hindered by high-dimensional 3D voxel feature maps. MLP-based methods (e.g., [2, 3, 12, 14, 26]) are popular due to MLP's ability to transform data between spaces. However, these methods are camera-dependent and neglect key geometric information like camera intrinsic and extrinsic parameters, limiting their application to new sensor configurations. Their computational expense stems from stretching 2D feature maps into 1D vectors and performing a full connection operation. Recent work (e.g., [20–22]) alleviates computational demands by independently applying MLP operations to image columns. In contrast, attention-based methods (e.g., [1, 22, 25]) work backward (3D to 2D). They construct 3D or BEV queries that cross-attend to image features for 3D or BEV feature creation. While effective, these methods can be costly for cost-sensitive autonomous driving systems, especially with dense queries. For a more comprehensive understanding of these view transformation methods, refer to [11].

In our quest to create a real-time self-driving car perception system, we use an MLP-based method for our 2D-to-BEV view transformation. However, we take a unique approach by independently elevating individual image columns, inspired by techniques found in [20–22]. Our method differs these references in several ways. Firstly, we don't assume that each image column corresponds to a BEV ray, which doesn't hold for wide-field-of-view cameras like fish-eye cameras. Secondly, we expand the application of these techniques to fuse BEV features from multiple cameras using camera intrinsic and extrinsic parameters. Lastly, to ensure efficiency, we avoid using attention-based layers (e.g., as in done [22]), opting instead for smaller MLP layers. Furthermore, in contrast to methods such as LSS [15], which initially construct a 3D volumetric feature map from 2D feature maps and depth information and subsequently reduce it to a BEV feature map, our approach generates a BEV feature map directly from 2D features. As a result, it is more computational and memory efficient. This strategy reflects our goal of achieving a balance between accuracy and computational efficiency.

3. NVAutoNet

3.1. Overview

NVAutoNet takes a set of N_{view} camera images $\{I_i\}_{i=1}^{N_{view}}$, covering a full 360-degree view of the surroundings, along with camera parameters. These images undergo 2D encoders to derive image features, which are then transformed into BEV features. BEV features specific to each camera are combined into a unified BEV feature map, fed into a shared BEV encoder to extract advanced features. Subsequently, this encoded BEV map enters task-specific encoders to generate 3D outputs such as obstacle cuboids, parking spaces, and driveable spaces. Refer to Figure 1 for an overview of the NVAutoNet process.

3.2. 2D Image Feature Extractors

Each *unrectified* input image $\mathbf{I}$ with dimensions $W \times H \times 3$ undergoes feature extraction, producing multiple feature maps F^k at various scales. These F^k have dimensions $C \times \frac{H}{2^{k+1}} \times \frac{W}{2^{k+1}}$. Our image feature extraction utilizes customized convolutional architectures, meticulously tailored for real-time operation. Comprising CNN blocks, the CNN backbone employs specific parameters like kernel size, stride, channel count, and repetitions. Network configurations for distinct camera groups are detailed in Table 1. These parameters are searched using hardware-informed neural architecture search [24], ensuring an optimal balance between accuracy and speed, with the omission of residual connections for faster processing [5]. The coarser maps merge with finer ones through upsampling, yielding multi-level feature maps enriched with semantic content. In the following sections, we delve into our approach for converting these 2D image feature maps into BEV representations.

3.3. Image-to-BEV Transformation and Fusion

3.3.1 BEV Plane and BEV Grid

The BEV representation is a popular choice in self-driving for efficiently depicting object positions, sizes, and aiding behavior prediction and planning. This representation involves a plane aligned with the vehicle's central axis, perpendicular to the Z axis. We divide this BEV plane into discrete sections using a grid $\mathbf{G}^{bev}$ with dimensions $M \times N$.

3.3.2 Polar vs Cartesian Coordinates

The resolution of the BEV grid $\mathbf{G}^{bev}$ significantly impacts detection range and accuracy. For safe and comfortable autonomous highway driving, a detection range of 200 meters is ideal. However, using a regular Cartesian grid with cells of 0.25 meters results in a large 1600×1600 grid. This consumes excessive memory and computational resources, making it impractical to train deep neural networks and deploy them on vehicle chips like NVIDIA DRIVE Orin. To balance close and far-range resolution, we adopt an irregular BEV grid using Polar coordinates as similar to PolarNet [27]. This grid is defined by angular samples (ranging from 0 to 360) denoted by M, and depth samples represented by N. For instance, by employing 1-degree angular resolution and logarithmic radial spacing, we efficiently represent the BEV plane with a compact 360×64 grid.

3.3.3 2D to BEV View Transformation

From a collection of 2D image feature maps originating from various cameras, our objective is to derive a singular BEV feature map, denoted as F_{bev}. In our approach, we adopt a data-centric strategy wherein we train a camera-to-BEV transformation function using a multilayer perceptron (MLP) network. Unlike prior MLP-based view transformation methods, our approach notably incorporates camera intrinsic and extrinsic parameters both during training and inference phases. Consequently, once trained, the model is capable of robustly adapting to different sensor configurations. A visual overview of our image-to-BEV view transformation module is provided in Figure 2.

A common observation is that there is a strong geometric relationship between row and column positions in the image plane and radial and angular positions in the BEV plane. In particular, each image column will correspond to a **curve** passing through the camera center on the BEV plane (see Figure 2 left). If camera images are rectified, these curves become polar BEV rays as utilized in the previous works [21, 22]. In this work, input images are not rectified because rectifying fisheyes camera images is often challenging due to nonlinear, non-uniform distortion. Our goal is to learn functions (represented by neural networks) to transform every image column features onto the BEV plane.

Formally, let $\mathbf{I}^c$ be a column from an image $\mathbf{I}$ (for brevity camera index is omitted), we first project every pixel location $\mathbf{p}_i = [u_i, v_i] \in \mathbf{I}^c$ onto the BEV plane (using camera intrinsic and extrinsic parameters), and convert them to polar coordinates (angles and distances), resulting in a list of polar BEV points $\{\mathbf{b}_i = [a_i, d_i]\}$. Here it is worth noting that we are not merely projecting pixels to the BEV plane for view transformation as we do not assume the world is flat. Instead, these polar BEV points are used to fit a polynomial curve (one per image column) $a = f^c(d)$ that allow us to deduce a BEV angular position from a BEV radial position. This is illustrated in Figure 2 (left), where examples of projected BEV points and fitted curves are presented. These fitted curves implicitly encapsulate camera intrinsic and extrinsic information.

For a given camera with a maximum detection range of r

	Input size	Blocks	Kernel sizes	Strides	Repeats	Channels
Front cam encoder	480x960	5	7-3-3-3-3	4-1-2-2-2	1-0-2-5-3	32-32-128-256-512
Side cam encoder	480x960	5	7-3-3-3-3	4-1-2-2-2	1-0-2-3-3	32-32-128-192-512
Fisheye cam encoder	480x960	5	7-3-3-3-3	4-1-2-2-2	1-0-2-3-3	32-32-64-96-512
BEV encoder	64x360	3	3-3-3	1-2-2	4-4-4	64-128-256

Table 1. Configurations of lightweight camera and BEV CNN backbones.

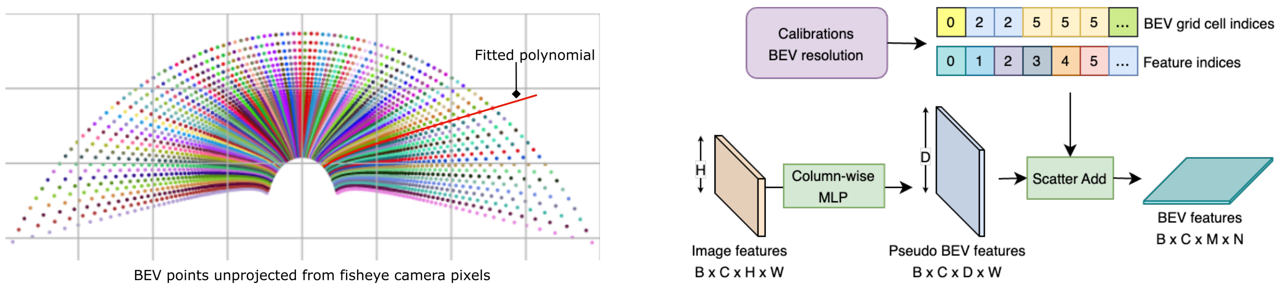

Figure 2. An overview of perspective to BEV view transformation. Left: Camera pixels are projected onto the BEV plane using camera intrinsic and extrinsic parameters. The resulting polar BEV points are then used to fit polynomial functions (one for each image column). These polynomial functions accept BEV radial distances as inputs and output corresponding BEV angular positions. Right: Image features are transformed into pseudo BEV features which are then transformed to the BEV features using BEV indices. The BEV indices are pre-computed using the fitted polynomial functions and a pre-defined BEV grid.

meters, we discretize the range $[0, r]$ into D logarithmically spaced bins. Let $\mathbf{c} \in \mathbb{R}^{H\times C}$ represent the encoded features corresponding to the image column $\mathbf{I}^c$ (again we dropped the feature level for notation simplicity), we define a BEV transformation network $\mathcal{B}(.)$ to yield $\mathbf{b} = \mathcal{B}(\mathbf{c})$ where $\mathbf{b} \in \mathbb{R}^{D\times C}$. Yet, these features aren't true BEV features at this stage; they are pseudo BEV features that must be aligned and combined into the global BEV feature map F_{bev}.

Each $\mathbf{b}$ feature comprises D vectors, each associated with a radial distance bin. Consequently, the BEV plane location of each cell in $\mathbf{b}$ is determined using its distance (center of the distance bin) and the fitted polynomial function f^c. The same procedure is extended to all image columns and images, facilitating the transformation of image features from the image plane to the BEV plane. To optimize computational efficiency, only one feature map level at stride 8 is transformed to the BEV plane per image.

BEV Pooling. Leveraging a predefined BEV grid $\mathbf{G}^{bev}$, we generate the final BEV feature map F_{bev} by combining all the computed BEV feature points described earlier. Specifically, the BEV feature points that belong to the same BEV grid cell are directly added together.

BEV Indexing. Assuming accurate camera calibration parameters, we can treat them as constants. In this context, we have the option to pre-calculate BEV indices, facilitating the mapping of pseudo BEV features to corresponding BEV grid cells. This BEV look-up-table expedites the image-to-BEV transformation process, ensuring remarkable efficiency for both training and inference operations. The computational overhead is notably minimal, requiring just 0.3ms per image.

3.3.4 BEV Transformation using MLP

The BEV transformation function $\mathcal{B}(.)$ is modeled using an MLP block with only one hidden layer, and the MLP parameters are shared by different image columns. Unlike convolutional layers, MLP layers have a capability to encode global contextual information (aka global attention), which we found very crucial for assigning image features into correct BEV positions as depth information has been lost and objects appears at different heights.

3.4. BEV Feature Extractor

Similar to 2D image feature extractors, we adopt a CNN backbone to extract high-level features from the fused BEV feature map. See Table 1 for network configration details. The output of the BEV feature extractor, dubbed $\hat{F}_{bev}$, will be consumed by different 3D detection heads.

4. Perception Tasks

3Dobject detection: A pivotal capability for autonomous driving, 3D object detection encompasses localizing, classifying, and estimating dimensions and orientations of objects in 3D space. Each object is uniquely defined by its category and a 3D cuboid.

3D freespace detection: While 3D obstacle detection traditionally covers specific categories like vehicles and vulnerable road users (VRUs), safe navigation demands broader insights. This encompasses addressing unforeseen road hazards such as tire debris, traffic cones, and immobile obstacles like dividers, curbs, and guardrails—scenarios that exceed standard 3D obstacle detection. To ensure secure movement within road limits, autonomous vehicles must adeptly navigate through various obstacles. The open space within these confines emerges as the drivable area, also known as the freespace region. In this work, the freespace regions are represented using radial distance maps (RDM), i.e., a representation that assigns a radial distance value to every angular ray extending from a central point.

3D parking space detection: Another essential facet of autonomous driving pertains to the accurate localization and classification of parking spaces. Each parking space is represented as an oriented rectangle, parameterized by values such as center coordinates, length, width, and orientation.

All three tasks are structured as set prediction tasks [19]. The network predicts a set of objects, subsequently matched against a corresponding set of ground truth objects to steer the training process. Due to space constraints, we direct readers to the supplementary material for comprehensive details concerning head network architectures, object representation, and loss calculations.

5. Multi-task Learning and Loss Balancing

NVAutoNet functions as a multitask network, adeptly handling various tasks simultaneously. These tasks inherently vary, making a uniform training approach less optimal. Additionally, distinct task losses occupy different terrains. A simple combination of loss components with equal weights might underemphasize certain tasks or let some tasks dominate the overall loss.

Adaptive Weight Adjustment: With T number of tasks, the combined training loss for batch b is $L_{total} = \sum_{t=1}^{T} w_t L_t^b$, where w_t is the task-specific loss weight, and L_t^b is the task loss for batch b. We introduce a straightforward yet powerful algorithm to dynamically modify the loss weights $\{w_t\}_{t=1}^{T}$ for each loss component. In a dataset of S samples, $L_{t,s}$ represents the loss for task t on sample s. At the start of each epoch, the total losses for each task t across all samples are computed. The loss weight w_t is then set as the reciprocal of this loss sum, proportionally adjusted by a predefined task loss prior c_t, i.e.,

$$L_t = \sum_{s=1}^{S} L_{t,s}, \quad \hat{w}_t = \frac{c_t}{L_t}, \quad w_t = \frac{\hat{w}_t}{\sum_{t=1}^{T} \hat{w}_t}. \qquad (1)$$

c_t is a configurable loss multiplier for task t that can be used to boost or reduce a task's loss. The approach of scaling losses by their reciprocal loss sums intuitively aligns all losses on a comparable scale, thereby minimizing disparities in gradient magnitudes. The inclusion of loss multipliers c_t becomes valuable when specific tasks are either more significant or present higher complexity.

In the initial epoch, we manually set all task loss weights $w_t = 1$ for all $t \in [1, T]$, since we lack previous epoch's loss sum data. After each epoch, we iteratively update w_t, ensuring a progressive refinement process.

Two-Stage Approach: Our approach unfolds in two stages. Initially, in the first training round, we uniformly establish c_t values at 1. Subsequently, we assess the outcomes of multi-task training in comparison to single-task training. By gauging improvements or declines in key performance indicators (KPIs) for individual tasks, we tune suitable c_t values. Equipped with these c_t values, we proceed to retrain the network in subsequent iterations. Notably, while c_t values are fine-tuned manually, our observations suggest that the pursuit of optimal c_t values is notably simpler compared to directly searching for w_t values.

6. Experimental Evaluation

This section highlights NVAutoNet's latency and accuracy achievements. Ideally, comparing NVAutoNet to state-of-the-art methods would be insightful. However, we encountered various challenges. Firstly, while there exist BEV object detection methods and benchmark datasets, freespace and parking space detection are rare. Secondly, many existing BEV perception methods, like 3D object detection, prioritize accuracy, unlike NVAutoNet which optimizes for both accuracy and latency. Thirdly, NVAutoNet is tailored for real self-driving applications with a required detection range of up to 200 meters. This makes some design choices (like image to BEV view transformation, polar BEV representation) less favorable for public datasets like nuScenes, which primarily consider shorter ranges (less than 70 meters). Consequently, our evaluation predominantly focuses on our in-house dataset and the NVIDIA DRIVE Orin SoC platform.

6.1. Datasets and Evaluation Metrics

6.1.1 Datasets

Our in-house datasets consist of real, simulated reality and augmented reality data. In total, there are 2.2M training scenes, 400K validation scenes and 177K testing scenes. The data was collected from more than 20 countries, and from different weather, lightning and road conditions. More detail information about the datasets such as percentage of each condition can be found in the supplementary material.

6.1.2 Evaluation Metrics

For obstacle and parking space detection, we use standard metrics such as F1 score, AP, mAP for evaluation. For true positive detections, we also provide regression errors such as position errors, orientation errors and shape errors. Regarding freespace detection, we calculate classification

Components	Latency (ms)	Total (ms)
Front Cam Encoder	1.80×2	3.61
Side Cam Encoder	1.60×2	3.21
Fisheye Cam Encoder	1.39×4	5.58
3D Uplifting + Fusion	0.30×8	2.40
3D Encoder + Heads	3.91×1	3.91
		18.72

Table 2. NVAutoNet latency (ms) measured on NVIDIA DRIVE Orin embedded GPU. The model runs at 53 FPS.

accuracy and regression error for individual angular bins. These outcomes are then averaged across angular bins and frames, resulting in the final metrics. A detailed elaboration of each metric's precise definition can be found in the supplementary material.

6.2. Latency Performance

NVAutoNet is deployed using NVIDIA TensorRT and evaluated on the NVIDIA DRIVE Orin platform. The total latency and individual latency figures for distinct NVAutoNet components are presented in Table 2. Notably, our network operates at an impressively low duration of just **18.7 ms** (for an 8-camera input), enabling a remarkable frame rate of 53 FPS.

In comparison to BEVDET [4], a highly efficient BEV object detection method, NVAutoNet showcases a 7.5x speed boost, even though BEVDET's latency measurement was conducted on the more potent NVIDIA GeForce RTX 3090 GPU, surpassing the NVIDIA DRIVE Orin SoC's capabilities. Compared to Fast-BEV [7], an approach highly tailored for real-time use cases, NVAutoNet still outperforms with 53 FPS against Fast-BEV's 45 FPS. Both NVAutoNet and Fast-BEV were evaluated on the identical NVIDIA DRIVE Orin SoC.

6.3. Qualitative Performance

Figure 3 shows visual results of a single NVAutoNet model tested on different scenarios (i.e., parking lot, urban and highway) and different car models (i.e., Sedan and SUV). More qualitative results can be found in the video.

6.4. Quantitative Performance

6.4.1 3D Obstacles

Presented in Table 3, the obstacle detection accuracy outcomes are showcased. The overall mAP score achieves 0.465. Notably, the network excels in detecting vehicles, attaining an AP score of 0.648, while it faces challenges in identifying persons/pedestrians, with an AP score of 0.351. This observation aligns with the typical size discrepancy between persons and vehicles. When focusing solely on a safety region (within 100 meters ahead and behind the ego vehicle, and 10 meters to its sides), the mAP score notably improves to 0.595.

	V	T	P	B	mAP	s-mAP
AP	0.63	0.38	0.35	0.48	0.46	0.59
PE (meters)	0.98	1.93	0.61	0.7	-	-
OE (degrees)	6.54	5.29	53.4	12.33	-	-

Table 3. Obstacle detection accuracy for different classes. PE=Position error, OE=Orientation error. V=Vehicle, T=Truck, P=Person, B=Bike. s-mAP is the mAP measured for the safety region only.

Class	Range (m)	F1	PE	OE	SE
Vehicle	0-50	0.715	0.741	7.213	0.173
Vehicle	50-100	0.481	2.231	11.295	0.246
Vehicle	100-150	0.371	4.098	12.304	0.306
Vehicle	150-200	0.250	6.647	9.193	0.339
Truck	0-50	0.481	1.099	8.463	0.362
Truck	50-100	0.430	2.269	7.064	0.453
Truck	100-150	0.419	3.554	6.854	0.420
Truck	150-200	0.386	5.334	6.778	0.411
Person	0-30	0.491	0.500	37.772	0.435
Person	30-50	0.413	1.259	51.483	0.480
Person	50-100	0.313	2.283	60.708	0.528
Bike/Rider	0-30	0.574	0.437	11.876	0.297
Bike/Rider	30-50	0.482	1.152	16.516	0.352
Bike/Rider	50-100	0.387	2.055	19.197	0.428

Table 4. Obstacle detection accuracy at different radial distance ranges. PE=Position error, OE=Orientation error, SE=Shape error.

The network demonstrates reasonable accuracy in estimating orientations for vehicles and trucks, displaying average errors below 7 degrees. However, orientation errors for bikes and persons are relatively high, averaging at 12.3 and 53.4 degrees, respectively. For more detailed insights into detection results, Table 4 provides a granular breakdown. Generally, detection accuracy drops steadily as distances increase.

6.4.2 3D Freespace

Table 5 offers a comprehensive overview of the freespace evaluation outcomes. For more detailed insights, Table 6 presents segmented regression metrics across various angular and radial sectors. Notably, regions in close proximity demonstrate higher accuracy compared to distant ones. Moreover, the front area exhibits greater accuracy than the rear section. This discrepancy arises from the fact that the front region benefits from coverage by three distinct cameras: 120FOV, 30FOV, and fisheye 200FOV.

Notably, precision and recall for the "Other" and "Vehicle" categories surpass those of the "VRU" category. This variance can be attributed to the inherent complexity of VRU classification. VRUs are often captured within fewer

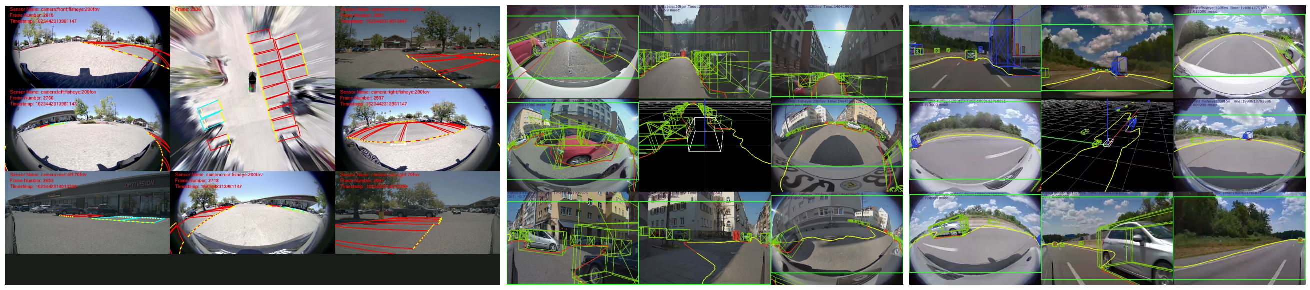
Figure 3. NVAutoNet's qualitative results. Left: Parking space detection in a parking lot. Middle: Obstacle and freespace detection in an urban area with a sedan car. Right: Obstacle and freespace detection in a highway area with a suv car.

angular bins compared to other categories, contributing to the increased difficulty in their precise classification.

Relative gap	Absolute gap	Success rate	Smoothness
44.14	1.95	77.59	0.77

	Vehicle	VRU	Other
Precision	0.92	0.73	0.98
Recall	0.92	0.66	0.98

Table 5. 3D freespace regression metrics (top) and classification metrics (bottom).

6.4.3 3D Parking space

Displayed in Table 7, the parking space detection outcomes are presented. It's noteworthy that the mean IoU for true positive detections hovers around 86%, suggesting a strong alignment between the majority of detections and the actual ground truth labels. However, it's observed that the parallel parking space category exhibits the lowest performance. This can be attributed to labeling intricacies. The dimensions of parallel parking spaces exhibit significant variability, contributing to difficulties in accurate labeling. Moreover, many curbside parking spaces lack precise width regulations, compelling labelers to rely on subjective judgments for width determination. Consequently, the labeling process for such spaces becomes inherently inconsistent.

6.5. Single task vs Multi-task Learning

For this experiment, we establish distinct c_t parameters (as introduced in Section 5) specifically as $[5, 3, 1]$ for the individual obstacle, parking space, and freespace tasks. The rationale behind this selection is to emphasize obstacle detection by assigning a higher weight, while relatively less emphasis is placed on the freespace task due to its relatively lower complexity. Performance comparison between the multi-task model and single task models is detailed in Table 8. It's observed that obstacle and parking detection accuracies hold up comparably to those achieved by the single task models. Although the freespace task experiences a 9.5% drop, its accuracy remains commendable. This outcome underscores the substantial efficacy of our proposed multi-task loss balancing algorithm.

Radial (meters)	Angular (degrees)	Success Rate (%)	Absolute gap (meters)
0-10	[-45, +45]	86.43	0.91
10-20	[-45, +45]	83.02	1.40
20-30	[-45, +45]	73.74	2.70
30-50	[-45, +45]	64.52	4.72
50-80	[-45, +45]	57.05	8.21
80-120	[-45, +45]	42.39	18.96
120-200	[-45, +45]	1.81	54.38
0-10	[-135, +135]	82.68	0.93
10-20	[-135, +135]	76.54	1.67
20-30	[-135, +135]	67.85	3.11
30-50	[-135, +135]	58.96	5.31
50-80	[-135, +135]	51.46	9.31
80-120	[-135, +135]	38.84	15.39
120-200	[-135, +135]	0.04	61.2

Table 6. Freespace regression metrics for different radial ranges and field-of-views.

	AP	mean IoU	F1
Angled	0.68	0.86	0.75
Parallel	0.19	0.82	0.37
Perpendicular	0.57	0.85	0.67
All	0.58	0.85	0.68

Table 7. Parking space detection performances.

	Obstacle	Freespace	Parking space
	mAP	Success rate	F1
Single task	0.46	77.59	0.68
Multi-task	0.46	70.19	0.68

Table 8. Single task vs. multi-task benchmarking results.

6.6. IPM vs MLP View Transformation

This study aims to showcase the advantages of employing a learning-based technique over an IPM method to convert 2D image features to BEV features. For this experiment, we focus on the obstacle detection task. The outcomes are summarized in Table 9, where our proposed 2D-to-BEV approach demonstrates a remarkable performance improvement over the traditional IPM method.

Method	Vehicle AP	Truck AP	Person AP	Bike-rider AP	mAP
IPM	0.49	0.32	0.27	0.33	0.30
MLP	0.63	0.38	0.35	0.48	0.46

Table 9. IPM versus MLP based 2D-to-BEV view transformation for obstacle detection.

	PT	T/F	Dataset size (Truck Platform)				
			50K	75K	100K	125K	150K
MA	✗	✓	0.14	0.18	0.19	0.20	0.21
MB	✓	✗	0.16	0.16	0.16	0.16	0.16
MC	✓	✓	0.28	0.29	0.29	0.29	0.30

Table 10. Transfer learning experiments for NVAutoNet. PT=Pretrained, T=Trained, F=Finetuned. mAP scores for obstacle detection are reported. Model-A (MA) trained from scratch using different dataset sizes collected from the truck platform; Model-B (MB) previously trained using data collected from ***the car platform***; Model-C (MC) fine-tuned from model-B using data collected from the truck platform.

6.7. Generalization to Different Vehicle Lines

In order to validate the robustness and scalability of our proposed architecture, we assess the performance of NVAutoNet on a truck platform, despite its initial development for a car platform. While there exist minor disparities in intrinsic parameters between the two platforms, the variations in extrinsic parameters are notably significant. Nevertheless, we demonstrate that deploying our perception model on a distinct platform doesn't necessitate extensive model redesign or extensive data collection. Remarkably, achieving satisfactory results only requires fine-tuning the model using a limited training dataset.

Datasets: This study focuses on the 3D obstacle detection task. We gathered data from both car and truck platforms for training purposes. Specifically, we collected 850K scenes for car model training (car-dataset-train) and 150K scenes for truck model training (truck-dataset-train), along with 26K scenes for truck model validation (truck-dataset-val). The label distribution is roughly similar between the two platforms. To explore the effect of training set size, we created smaller subsets from the 150K truck-dataset-train, ranging from 50K to 150K scenes.

Our comparison involves three models: Model-A trained solely with truck-dataset-train, Model-B pretrained with car-dataset-train, and Model-C fine-tuned from Model-B using truck-dataset-train. All models are tested on truck-dataset-val, and the results are presented in Table 10. As anticipated, Model-A's performance improves as the dataset size increases. Interestingly, even with only 50K scenes, Model-B, which has never seen data from a truck platform, outperforms Model-A. This illustrates the network's generalization capability. Notably, Model-C outperforms both Model-A and Model-B by a significant margin. It's worth noting that Model-C's performance improvement with larger fine-tuning datasets is not as substantial as Model-A's. This suggests that only a small amount of data is required for fine-tuning when deploying the NVAutoNet model, previously trained for one platform, onto another. These findings reaffirm the robustness and scalability of our proposed NVAutoNet, crucial qualities for practical production applications.

7. Conclusion

NVAutoNet emerges as a specialized perception network designed to meet the unique demands of autonomous vehicles. Unlike many existing 3D perception methods, it strikes a balance between accuracy and computational efficiency, onboard chip compatibility, and adaptability to diverse vehicle types. NVAutoNet processes synchronized camera images to predict 3D signals such as obstacles, freespaces, and parking spaces. Its architecture employs efficient convolutional networks for image and BEV backbones, optimized for efficiency with TensorRT. The image-to-BEV transformation uses simple linear layers and BEV look-up tables for rapid yet accurate inference.

Trained extensively on proprietary data, NVAutoNet consistently attains high perception accuracy while maintaining real-time performance, achieving 53 frames per second on the NVIDIA DRIVE Orin SoC. It excels in handling sensor mounting deviations across car models and adapts seamlessly to various vehicle types through streamlined model fine-tuning procedures. NVAutoNet stands as a significant advancement in autonomous vehicle perception, addressing key challenges and offering efficient and reliable 3D perception for real-world scenarios.

Acknowledgments We would like to thank the NVIDIA Maglev and ML foundation teams for the machine learning infrastructure support, thank the NAS team for the network optimization, thank Tero Kuosmanen for the initial idea of the loss balancing algorithm and special thank to the whole DRIVE AV team for data collection, network deployment, iteration and on-road testing.

References

[1] Yigit Baran Can, Alexander Liniger, Danda Pani Paudel, and Luc Van Gool. Structured bird's-eye-view traffic scene understanding from onboard images. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15641–15650, 2021. 2

[2] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15773–15783, 2021. 2

[3] Noureldin Hendy, Cooper Sloan, Fengshi Tian, Pengfei Duan, Nick Charchut, Yuxing Xie, Chuan Wang, and James Philbin. Fishing net: Future inference of semantic heatmaps in grids. *ArXiv*, abs/2006.09917, 2020. 2

[4] Junjie Huang, Guan Huang, Zheng Zhu, Ye Yun, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021. 2, 6

[5] Guilin Li, Junlei Zhang, Yunhe Wang, Chuanjian Liu, Matthias Tan, Yunfeng Lin, Wei Zhang, Jiashi Feng, and Tong Zhang. Residual distillation: Towards portable deep neural networks without shortcuts. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8935–8946. Curran Associates, Inc., 2020. 3

[6] 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. 2

[7] Yangguang Li, Bin Huang, Zeren Chen, Yufeng Cui, Feng Liang, Mingzhu Shen, Fenggang Liu, Enze Xie, Lu Sheng, Wanli Ouyang, and Jing Shao. Fast-bev: A fast and strong bird's-eye view perception baseline, 2023. 6

[8] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers, 2022. 2

[9] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. PETR: position embedding transformation for multi-view 3d object detection. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXVII*, volume 13687 of *Lecture Notes in Computer Science*, pages 531–548. Springer, 2022. 2

[10] Yicheng Liu, Tianyuan Yuan, Yue Wang, Yilun Wang, and Hang Zhao. Vectormapnet: End-to-end vectorized hd map learning. In *International Conference on Machine Learning*, 2023. 2

[11] Yuexin Ma, Tai Wang, Xuyang Bai, Huitong Yang, Yuenan Hou, Yaming Wang, Yu Qiao, Ruigang Yang, Dinesh Manocha, and Xinge Zhu. Vision-centric bev perception: A survey, 2023. 2

[12] Kaustubh Mani, Swapnil Daga, Shubhika Garg, N. Sai Shankar, Krishna Murthy Jatavallabhula, and K. Madhava Krishna. Mono lay out: Amodal scene layout from a single image. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1678–1686, 2020. 2

[13] Mong H. Ng, Kaahan Radia, Jianfei Chen, Dequan Wang, Ionel Gog, and Joseph Gonzalez. Bev-seg: Bird's eye view semantic segmentation using geometry and semantic point cloud. *ArXiv*, abs/2006.11436, 2020. 2

[14] Bowen Pan, Jiankai Sun, Ho Yin Tiga Leung, Alex Andonian, and Bolei Zhou. Cross-view semantic segmentation for sensing surroundings. *IEEE Robotics and Automation Letters*, 5:4867–4873, 2020. 2

[15] 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. 2

[16] Limeng Qiao, Wenjie Ding, Xi Qiu, and Chi Zhang. End-to-end vectorized hd-map construction with piecewise bezier curve. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13218–13228, June 2023. 2

[17] Cody Reading, Ali Harakeh, Julia Chae, and Steven L. Waslander. Categorical depth distribution network for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8555–8564, June 2021. 2

[18] Lennart Reiher, Bastian Lampe, and Lutz Eckstein. A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird's eye view. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, page 1–7. IEEE Press, 2020. 2

[19] H. Rezatofighi, T. Zhu, R. Kaskman, F. T. Motlagh, J. Shi, A. Milan, D. Cremers, L. Leal-Taixe, and I. Reid. Learn to predict sets using feed-forward neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9011–9025, dec 2022. 5

[20] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11135–11144, 2020. 2

[21] Avishkar Saha, Oscar Mendez, Chris Russell, and Richard Bowden. Enabling spatio-temporal aggregation in birds-eye-view vehicle estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5133–5139, 2021. 2, 3

[22] Avishkar Saha, Oscar Mendez, Chris Russell, and Richard Bowden. Translating images into maps. In *ICRA 2022*, 2022. 2, 3

[23] Samuel Schulter, Menghua Zhai, Nathan Jacobs, and Manmohan Chandraker. Learning to look around objects for top-view representations of outdoor scenes. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 815–831, Cham, 2018. Springer International Publishing. 2

[24] Linnan Wang, Chenhan Yu, Satish Salian, Slawomir Kierat, Szymon Migacz, and Alex Fit Florea. Gpunet: Searching the deployable convolution neural networks for gpus, 2022. 3

[25] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. DETR3D: 3d object detection from multi-view images via 3d-to-2d queries. *CoRR*, abs/2110.06922, 2021. 2

[26] Weixiang Yang, Qi Li, Wenxi Liu, Yuanlong Yu, Yuexin Ma, Shengfeng He, and Jia Pan. Projecting your view attentively: Monocular road scene layout estimation via cross-view transformation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15531–15540, 2021. 2

[27] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3