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

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

1. Perception Tasks

1.1. 3D Object Detection

3D object detection is a key capability for autonomous driving. The goal is to localize, classify and estimate dimensions and orientations of objects in 3D space. Each object is represented by its category and 3D cuboid. In particular, each 3D cuboid has 9 degree-of-freedom (DOF) representing position, dimension, and orientation. In this work, we adopt a set prediction approach to remove the need for a non-maximum suppression (NMS) post-processing.

The 3D object detection network includes five lightweight heads (implemented by a couple of convolutional layers), which takes the bottleneck feature map $\hat{F}_{bev}$ as input and predicts object class distributions and 3D cuboid parameters. Formally, let us denote $C \times M \times N$ be the dimension of $\hat{F}_{bev}$, where $M \times N$ is the spatial dimension and C is the number of channels, the 3D detection network will output $\hat{K} = M \times N$ objects — one object per grid cell. The model employs one head to predict the object classification scores, and three other heads for 3D cuboid parameters (position, dimensions and orientation) regression. There is an additional head for predicting uncertainty of cuboid parameters.

Classification. For k number of objects, the network outputs $k + 1$ classification channels, where the first channel represents object existence and the other k channels represent a categorical distribution over k classes.

Position. The network predicts a tuple $[r, a, e]$, where r is radial distance, a is azimuth angle, and e is elevation. Note that the network actually predicts radial and angular offset values, which are then added to grid cell positions to form final radial and angular positions.

Dimensions. The network predicts three scalars $[d_x, d_y, d_z]$ which are absolute values in meters.

Orientation. We represent object orientation using a full rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$, as opposed to previous works only estimating yaw angle. However, rotation matrix prediction is nontrivial because not every 3×3 matrix is a valid rotation matrix. Here we propose to train the network to predict sine and cosine values of yaw (ψ), pitch (θ), and roll (ϕ) angles respectively, which are later used to construct a rotation matrix by applying matrix multiplication of three axis rotation matrices together.

$$R = R_z(\psi)R_y(\theta)R_x(\phi), \tag{1}$$

$$\begin{aligned} R_z &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ R_y(\theta) &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \\ R_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \end{aligned} \tag{2}$$

Training Losses. For each scene, let $\mathcal{G} = \{\mathbf{g}_i\}_{i=1}^{K}$ be a set of K ground-truth objects, and let $\mathcal{P} = \{\mathbf{d}_i\}_{i=1}^{\hat{K}}$ be a set of $\hat{K}$ predicted objects. The training loss is computed in a two-step fashion where in the first step, we found the best one-to-one matching between $\mathcal{G}$ and $\mathcal{P}$. In the second step, we compute the final loss based on the matching result. Although the idea sounds very simple, the training process will be highly affected by the matching quality — if the matching algorithm returns wrong or sub-optimal matches, the network training won't be successful.

To ensure high quality matches, we constructed a matching cost function considering classification, position, dimension, and orientation costs between each candidate-ground truth pair. In the early phase of training, however the network prediction is quite noisy, often leading to bad assignments. This issue can be mitigated by setting a higher weight for the position cost. Alternatively, inspired by [1], we proposed to limit the matching candidates for each ground truth object by its corresponding coverage on the BEV grid. This will prohibit matching pairs in which the ground truth object and the candidate are far apart. Moreover, the matching optimization problem is now simplified,

we adopt a greedy matching algorithm instead of the well-known Hungarian algorithm without losing accuracy, but more efficient.

Let $\mathcal{P}_{pos}$ be the set of positive object candidates matched to the ground truth objects, and $\mathcal{P}_{neg} = \mathcal{P} \setminus \mathcal{P}_{pos}$ be a set of negative objects, the training loss is as below:

$$\begin{aligned} L_{obs}(\mathcal{G}, \mathcal{P}) &= L_{obs}^{pos}(\mathcal{G}, \mathcal{P}_{pos}) + L_{obs}^{neg}(\mathcal{P}_{neg}) \\ &= \sum_{\mathbf{d}_i \in \mathcal{P}_{pos}} L(\mathbf{g}_i, \mathbf{d}_i) + \sum_{\mathbf{d}_i \in \mathcal{P}_{neg}} L(\mathbf{d}_i), \end{aligned} \tag{3}$$

where $L(\mathbf{g}_i, \mathbf{d}_i)$, and $L(\mathbf{d}_i)$ are loss functions per each predicted candidate. If $\mathbf{d}_i$ is a negative candidate, the loss function $L(\mathbf{d}_i)$ is simply a (focal) binary cross entropy loss — pulling its objectness score to zero. If $\mathbf{d}_i$ is a positive candidate, the loss function $L(\mathbf{g}_i, \mathbf{d}_i)$ composes of classification and regression losses. While the classification loss is still a (focal) cross-entropy loss, the regression loss is more complex as explained below.

To regress 3D cuboid parameters, ideally we should compute a global loss such as intersection-over-union (IoU) score which considers all parameters together as there is a strong correlation between cuboid parameters. However there is no closed-form solution to compute IoU between two 3D cuboids. Here, we propose to decompose the 3D cuboid regression loss into position (location) loss, shape (size) loss and orientation (rotation) loss, as below:

$$L^{reg}(\mathbf{g}_i, \mathbf{d}_i) = L^{loc}(\mathbf{g}_i, \mathbf{d}_i) + L^{size}(\mathbf{g}_i, \mathbf{d}_i) + L^{rot}(\mathbf{g}_i, \mathbf{d}_i) \tag{4}$$

$$\begin{aligned} L^{loc}(\mathbf{g}_i, \mathbf{d}_i) = \frac{|\mathbf{g}_i^r - \mathbf{d}_i^r|}{\sigma_r} + \frac{|\mathbf{g}_i^a - \mathbf{d}_i^a|}{\sigma_a} + \frac{|\mathbf{g}_i^e - \mathbf{d}_i^e|}{\sigma_e} \\ + \log(2\sigma_r) + \log(2\sigma_a) + \log(2\sigma_e) \end{aligned} \tag{5}$$

$$L^{size}(\mathbf{g}_i, \mathbf{d}_i) = \frac{1}{\sigma_s}(1 - \prod_{d \in \{d_x, d_y, d_z\}} \frac{\min(\mathbf{g}_i^d, \mathbf{d}_i^d)}{\max(\mathbf{g}_i^d, \mathbf{d}_i^d)}) + \log(2\sigma^s) \tag{6}$$

$$L^{rot}(\mathbf{g}_i, \mathbf{d}_i) = \frac{1}{\sigma_o} \sum_{r \in R} |\mathbf{g}_i^r - \mathbf{d}_i^r| + \log(2\sigma^o) \tag{7}$$

where σ_r,σ_a, σ_e, σ_s, σ_o are uncertainty values for position, shape and orientation respectively. These values are predicted by the network.

1.2. 3D Freespace

3D obstacle detection generally covers category-wise classifiable vehicles and vulnerable road users (VRU). In a driving scenario, there is a lot more information which is relevant for safe driving beyond the predictions of 3D obstacle detection. For example, there can be random hazard obstacles like tyres, traffic cones lying on the road. Additionally, there are a lot of static obstacles like road-divider, road-side curb, and guard rails which are not covered by 3D obstacle detection. An autonomous vehicle has to drive safely within the boundaries of the road by avoiding all kinds of obstacles. The region within the boundaries of the road which is not occupied by any obstacle could be carved out as a driveable region. The driveable region is used interchangeably as the freespace region. The down-stream behaviour planner would consume the freespace region information to plan a safe trajectory for the AV. So it is essential to have a perception component which predicts this freespace region. The freespace region is represented as a radial distance map (RDM). The representation is explained in more details in the next section.

Radial Distance Map. While polygons are used for labeling of 3D freespace, we use the RDM representation due to its higher efficiency. RDM is composed of equiangular bins and radial distance values for each angular bin to denote spatial locations. For autonomous driving applications, the distance to the closest freespace boundary is the most important one and thus we use a single scalar for each angular bin to represent the closest freespace boundary. In order to create ground-truth RDM for 3D freespace, we simply shoot a ray from the center of BEV plane at each angular bin direction and compute the intersection between the ray and the ground-truth polygon. The most important benefit of using RDM to represent 3D freespace is it can be directly used to create 3D boundary points without additional post-processing. In addition to the radial distance, we also have, for each angular bin, boundary semantic labels such as vehicle, VRU and others. Each freespace ground truth label becomes $(\mathbf{r}, \mathbf{c})$, where $\mathbf{r}$ is a radial distance vector, and $\mathbf{c}$ is a boundary semantic vector.

Model. The 3D freespace detection network consists of a shared neck and two separate heads. The shared neck includes a couple of convolutional layers on top of the bottleneck feature map $\hat{F}_{bev}$. It is later extended into two heads which predict radial distance and classification maps.

Training Losses. For each scene, let $\mathcal{G} = (\mathbf{r}, \mathbf{c})$ be the ground-truth, let $\mathcal{P} = (\hat{\mathbf{r}}, \hat{\mathbf{c}})$ be the prediction. The overall loss function for the 3D freespace detection task is defined as:

$$L_{fsp}(\mathcal{G}, \mathcal{P}) = L_{fsp}^{reg}(\hat{\mathbf{r}}, \mathbf{r}) + L_{fsp}^{cls}(\hat{\mathbf{c}}, \mathbf{c}) \tag{8}$$

$$L_{fsp}^{reg}(\hat{\mathbf{r}}, \mathbf{r}) = L_{fsp}^{iou}(\hat{\mathbf{r}}, \mathbf{r}) + L_{fsp}^{sim}(\hat{\mathbf{r}}, \mathbf{r}) \tag{9}$$

where $L_{fsp}^{reg}(\hat{\mathbf{r}}, \mathbf{r})$ is the regression loss which is a combination of radius loss $L_{fsp}^{iou}(\hat{\mathbf{r}}, \mathbf{r})$ and the similarity loss $L^{sim}(\hat{\mathbf{r}}, \mathbf{r})$; $L_{fsp}^{cls}(\hat{\mathbf{c}}, \mathbf{c})$ is the classification loss. The radius loss is a polar intersection-over-union loss (IoU) computed between the prediction and ground truth labels, defined as

below:

$$L_{fsp}^{iou}(\hat{\mathbf{r}}, \mathbf{r}) = 1.0 - \prod_{i=1}^{N_{bins}} \frac{\min(r_i, \hat{r_i})}{\max(r_i, \hat{r_i})}. \quad (10)$$

The similarity loss is computed between the line segments formed by joining the end points from the consecutive angular bins in the prediction and ground truth labels. This loss helps in reducing the noise in the predicted RDM.

$$L_{fsp}^{sim}(\hat{\mathbf{r}}, \mathbf{r}) = \sum_{i=1}^{N_{bins}} (1.0 - \frac{\hat{l}_i^{i+1} \cdot l_i^{i+1}}{\left\|\hat{l}_i^{i+1}\right\| \left\|l_i^{i+1}\right\|}), \quad (11)$$

where $\hat{l}_i^{i+1}$ is the line segment formed by joining the end points of $\hat{r}_i$ and $\hat{r}_{i+1}$. l_i^{i+1} is defined similarly. Finally, the classification loss $L_{fsp}^{cls}(\hat{\mathbf{c}}, \mathbf{c})$ is the standard focal loss, i.e.,

$$L_{fsp}^{clc}(\hat{\mathbf{c}}, \mathbf{c}) = \sum_{i=1}^{N_{bins}} \sum_{j=1}^{C} (1 - p_{ij})^{\gamma} \log(p_{ij}), \quad (12)$$

where γ is the focal loss parameter.

1.3. 3D Parking Space

Another important aspect of autonomous driving is the ability to localize and classify parking spaces. Each parking space is represented as an oriented rectangle, parameterized by $[cx, cy, l, w, \theta]$, where cx and cy are the center coordinates of the box, l and w are the length and the width of the box in meters respectively, and θ is the orientation of the box (yaw angle in radians) in the range $[0, \pi)$. Note that an oriented box angled at π visually appears the same as a box oriented at 0, thus the orientation value θ need not cover the entire angular range of $[0, 2\pi)$. Knowing the *profile* of every parking space is important for planning and control purposes. As such, every prediction output by our model will be assigned a parking profile. In the current system we support three different parking profiles: *angled, parallel and perpendicular*. As their name suggests, the profiles denote the types of planning and control maneuvering required to successfully park the car in the parking space. *Parallel* parking spaces are ones that typically appear on the side of the street and require a parallel parking maneuver. Conversely, *angled* and *perpendicular* parking spots are ones where the car can be parked straight in (or backed in).

Model. The parking space detection task follows the same design (head, training strategy and losses) of the obstacle detection task. The parking detection network consists of classification and regression heads. The classification head predicts per-profile confidence scores. We rely on each parking spot's profile to implicitly encode its existence score. The regression head predicts the parking space oriented bounding boxes as discussed above.

	Amount
Number of cameras per scene	8
Real training samples	2M
Sim training samples	200k
Validation samples	400K
Test samples	177K
Number of countries	20
Number of obstacle classes	5
Number of freespace classes	3
Number of parking classes	3
Percentage of dry roads	95.17%
Percentage of wet roads	4.83%
Bright light condition	49.5%
Diffused light condition	31.4%
Poor light condition	19.1%

Table 1. In-house dataset summary.

Training Losses. The training loss is similar to that of the obstacle detection task, but its regression loss function is much simpler. Let $\mathbf{g}_i$ and $\mathbf{d}_i$ be a matched pair of ground truth and detection, the regression loss is defined as:

$$L_{prk}^{reg}(\mathbf{g}_i, \mathbf{d}_i) = \sum_{s \in \{cx,cy,l,w,\theta\}} (\mathbf{g}_i^s - \mathbf{d}_i^s)^2. \quad (13)$$

2. 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. Table 1 summarizes our datasets. Lidar data was used to generate ground truth labels. Our data contains a fair amount of noisy labels due to view point differences between Lidar and camera sensors. For example, Lidar is mounted at a higher position than cameras, thus there are obstacles, which may be visible by Lidar, are hardly visible by cameras. These issue not only affect model training but also model evaluation (e.g., low recall rates).

3. Evaluation Metrics

3.1. 3D Obstacles

We calculate obstacle detection metrics based on identifying true positives (TP), false positives (FP), and false negatives (FN) from detection outputs and ground truth labels. For each class, we find one-to-one matching between detection output and ground truth using greedy algorithm with the Euclidean distance between their centroids. A match is valid if the relative radial distance between a prediction and ground truth objects is less than 10%, and their absolute azimuth error is less than 2 degrees. All unmatched detections become FP while all unmatched ground truth becomes FN. Once TP, FP, and FN have been identified, we compute precision, recall, F1-score, AP and mAP KPIs. Moreover, we

search for the best confidence threshold that maximizes F1-score, and compute regression errors for all true positive detections. *Position error* measures relative radius error (%), absolute azimuth error (degrees) and absolute elevation error (meter). *Orientation error* is defined as $||\log(R^{-1}\hat{R})||$, where R and $\hat{R}$ are ground truth and prediction rotation matrices. *Shape error* measures relative error for length, width, and height (%). We also define the safety mAP based on a safety zone. The safety zone is defined as a rectangular region around the ego vehicle, i.e., 100 meters ahead and behind the ego vehicle and 10 meters left and right of the vehicle.

3.2. 3D Freespaces

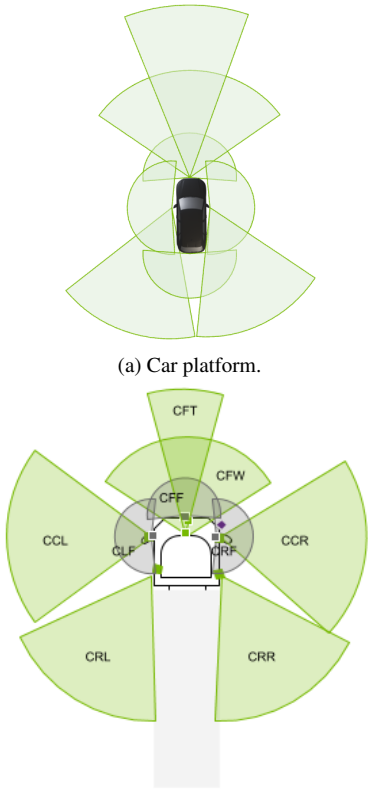
Given a pair of ground truth and prediction freespace RDMs, we compute the following metrics (averaged over angular bins and frames). *Relative gap* measures the relative radial distance errors (%). *Absolute gap* measures the absolute radial distance errors (meters). *Success rate* measures the percentages of successfully estimated angular bins. Angular bins are considered as successfully estimated when the relative gap is less than 10%. *Smoothness* measures the total variation of radial distance maps defined as $\sum_{i=1}^{N_{bins}} |r_i - r_{i-1}|$. *Classification error* measures precision and recall for each label class.

3.3. 3D Parking Spaces

Similar to object detection, we compute precision, recall, F1 and AP metrics. Intersection over union (IoU) scores are used to match predictions to ground truth labels. A match is valid if the IoU $\geq$ 70%. This strict criteria is necessary for real-world applications of autonomous parking as small misalignment between the detection and the actual parking space position can lead to imperfect parking. We also compute mean IoU values for all true positive detections.

4. Sensor Configurations for Car and Truck platform.

Figure 1 shows the differences of sensor mounting positions between a car and truck platforms.



(a) Car platform.

(b) Truck platform.

Figure 1. Different camera sensor setups for cars and trucks.

References

[1] Peize Sun, Yi Jiang, Enze Xie, Wenqi Shao, Zehuan Yuan, Changhu Wang, and Ping Luo. What makes for end-to-end object detection? In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9934–9944. PMLR, 2021. 1