Supplementary Material for MonoRUn: Monocular 3D Object Detection by Reconstruction and Uncertainty Propagation

Hansheng Chen, Yuyao Huang, Wei Tian*, Zhong Gao, Lu Xiong
Institute of Intelligent Vehicles, School of Automotive Studies, Tongji University
hanshengchen97@gmail.com {huangyuyao, tian_wei, 1931604, xiong_lu}@tongji.edu.cn

As mentioned in the main paper, this supplementary material discusses the epistemic uncertainty, end-to-end training and Monte Carlo scoring in detail. Apart from that, it also provides the complete evaluation results on the official KITTI benchmark, including precision-recall plots.

1. Details on Epistemic Uncertainty

For the estimation of epistemic uncertainty, we adopt the Monte Carlo dropout approach described in [2]. By sampling the reconstruction network, we can estimate the epistemic uncertainty of object coordinates $\mathbf{x}^{\text{OC}}$. The predictive mean and variance are approximated by:

$$\mathrm{E}[\mathbf{x}^{\text{OC}}] \approx \overline{\mathbf{x}^{\text{OC}}} = \frac{1}{N_{\text{MC}}} \sum_{i}^{N_{\text{MC}}} \mathbf{x}_i^{\text{OC}}, \tag{1}$$

$$\mathrm{Var}[\mathbf{x}^{\text{OC}}] \approx \frac{1}{N_{\text{MC}} - 1} \sum_{i}^{N_{\text{MC}}} (\mathbf{x}_i^{\text{OC}} - \overline{\mathbf{x}^{\text{OC}}})^2, \tag{2}$$

where $\mathbf{x}_i^{\text{OC}}$ is the output of the i-th sampled network, N_{MC} is the number of Monte Carlo samples (set to 50).

Next, we need to transform the variances of 3D object coordinates into the variances of 2D reprojected coordinates. Strict 3D-2D variance projection requires knowing the object pose in advance, which is unavailable before the PnP module. Thus, we use the following approximation:

$$\begin{cases} \mathrm{Var}[u_{\text{rp}}^{\text{norm}}] \approx \frac{1}{2}(\mathrm{Var}[x^{\text{OC}}] + \mathrm{Var}[z^{\text{OC}}]), & \text{(3a)} \\ \mathrm{Var}[v_{\text{rp}}^{\text{norm}}] \approx \mathrm{Var}[y^{\text{OC}}], & \text{(3b)} \end{cases}$$

where $u_{\text{rp}}^{\text{norm}}, v_{\text{rp}}^{\text{norm}}$ are the normalized reprojected coordinates (invariant to depth). Note that this approximation does not take object orientation into consideration, thus Eq. 3a simply averages the horizontal variances.

*Corresponding author: Wei Tian.

Finally, the reconstruction module outputs the combined uncertainty of reprojected 2D coordinates:

$$\begin{cases} \sigma^2_{u_{\text{rp}}^{\text{norm}},\text{comb}} \approx \frac{1}{N_{\text{MC}}} \sum_{i}^{N_{\text{MC}}} \sigma^2_{u_{\text{rp}}^{\text{norm}},i} + \mathrm{Var}[u_{\text{rp}}^{\text{norm}}], & \text{(4a)} \\ \sigma^2_{v_{\text{rp}}^{\text{norm}},\text{comb}} \approx \frac{1}{N_{\text{MC}}} \sum_{i}^{N_{\text{MC}}} \sigma^2_{v_{\text{rp}}^{\text{norm}},i} + \mathrm{Var}[v_{\text{rp}}^{\text{norm}}], & \text{(4b)} \end{cases}$$

where $\sigma^2_{u_{\text{rp}}^{\text{norm}},i}, \sigma^2_{v_{\text{rp}}^{\text{norm}},i}$ is aleatoric uncertainty predicted by the i-th sampled network. It is to be observed that all the 2D variances above are in the normalized scale. The actual variances should be further multiplied by a scale factor $(f/t_{\text{z}})^2$, where f is camera focal length in pixels and t_{z} is the z-component of object pose. Since t_{z} is unknown beforehand, we only apply this factor to the final pose covariance as a correction.

2. Details on End-to-End Training

End-to-end training is only investigated in the ablation studies, and is not included in our final training setup. Nevertheless, here we elaborate on the details of differentiable PnP and end-to-end training loss.

2.1. Back-Propagating the Uncertainty-Driven PnP

Generally, we follow the approach in BPnP [1], with the code completely re-implemented for higher efficiency and uncertainty awareness. The details are as follows.

Differentiating the PnP Algorithm The derivative of the PnP result $\mathbf{p}^*$ is as follows:

$$\frac{\partial \mathbf{p}^*}{\partial (\cdot)^{\text{T}}} = -\mathbf{H}^{-1} \frac{\partial \mathbf{J}^{\text{T}} \mathbf{r}_{\text{all}}}{\partial (\cdot)^{\text{T}}} \Big|_{\mathbf{p}^*}, \tag{5}$$

where $(\cdot)$ stands for PnP inputs $\mathbf{x}^{\text{OC}}$ and $\boldsymbol{\sigma}$.

Proof. Recall the MLE of object pose:

$$\begin{aligned}\mathbf{p}^* &= \arg\min_{\mathbf{p}} \frac{1}{2} \sum_{(u,v)\in RoI} \mathbf{r}_{(u,v)}^{\mathrm{T}} \boldsymbol{\Sigma}_{(u,v)}^{-1} \mathbf{r}_{(u,v)} \\ &= \arg\min_{\mathbf{p}} \frac{1}{2} \mathbf{r}_{\text{all}}^{\mathrm{T}} \mathbf{r}_{\text{all}}. \end{aligned} \tag{6}$$

The gradient of the NLL function ($\frac{1}{2}\mathbf{r}_{\text{all}}^{\mathrm{T}}\mathbf{r}_{\text{all}}$) w.r.t. $\mathbf{p}$ is as follows:

$$\mathbf{g} = \mathbf{J}^{\mathrm{T}} \mathbf{r}_{\text{all}} \tag{7}$$

with $\mathbf{J} = \frac{\partial \mathbf{r}_{\text{all}}}{\partial \mathbf{p}^{\mathrm{T}}}$. When the optimization (Eq. 6) converges to $\mathbf{p}^*$, the gradient always satisfies

$$\mathbf{g}_{\mathbf{p}^*} = \mathbf{0}. \tag{8}$$

Therefore, the total derivative of $\mathbf{g}$ w.r.t. any PnP input equals zero:

$$\frac{\mathrm{D}\mathbf{g}_{\mathbf{p}^*}}{\mathrm{D}(\cdot)^{\mathrm{T}}} = \frac{\partial \mathbf{g}_{\mathbf{p}^*}}{\partial \mathbf{p}^{*\mathrm{T}}} \frac{\partial \mathbf{p}^*}{\partial (\cdot)^{\mathrm{T}}} + \frac{\partial \mathbf{g}_{\mathbf{p}^*}}{\partial (\cdot)^{\mathrm{T}}} = \mathbf{0}, \tag{9}$$

which implies that

$$\frac{\partial \mathbf{p}^*}{\partial (\cdot)^{\mathrm{T}}} = -\left(\frac{\partial \mathbf{g}_{\mathbf{p}^*}}{\partial \mathbf{p}^{*\mathrm{T}}}\right)^{-1} \frac{\partial \mathbf{g}_{\mathbf{p}^*}}{\partial (\cdot)^{\mathrm{T}}} = -\mathbf{H}^{-1} \frac{\partial \mathbf{J}^{\mathrm{T}} \mathbf{r}_{\text{all}}}{\partial (\cdot)^{\mathrm{T}}} \Big|_{\mathbf{p}^*}. \tag{10}$$

□

Implementation Details The PnP forward process is implemented with Ceres Solver on CPU, while the backward process is implemented with the Autograd package of PyTorch on GPU. With our efficient implementation, the backward overhead of computing the exact second derivatives is negligible for training.

2.2. End-to-End Training Loss

Leveraging the differentiable PnP, we apply smooth L1 loss on the Euclidean errors of estimated translation vector $\mathbf{t}^*$ and yaw angle β^*:

$$L_{\text{trans}} = L_{\text{SmoothL1}}(\|\mathbf{t}^* - \mathbf{t}_{\text{gt}}\|), \tag{11}$$

$$L_{\text{rot}} = L_{\text{SmoothL1}}\left(\left\| \begin{bmatrix} \cos\beta^* \\ \sin\beta^* \end{bmatrix} - \begin{bmatrix} \cos\beta_{\text{gt}} \\ \sin\beta_{\text{gt}} \end{bmatrix} \right\|\right). \tag{12}$$

The overall loss for end-to-end training is:

$$\begin{aligned} L_{\text{e2e}} =& L_{\text{2D}} + L_{\text{dim}} + L_{\text{score}} + \lambda L_{\text{calib}} \\ &+ L_{\text{trans}} + L_{\text{rot}} + L_{\text{NOC}}, \end{aligned} \tag{13}$$

where the reprojection loss is replaced by translation and rotation losses, and the NOC loss is added as regularization.

3. Details on Monte Carlo Scoring

By sampling $\mathbf{p}_i$ from the distribution $\mathcal{N}(\mathbf{p}^*, \boldsymbol{\Sigma}_{\mathbf{p}^*})$, the 3D localization score can be computed using Monte Carlo integration:

$$c_{\text{3DL}} = \frac{1}{N_{\text{MC}}} \sum_{i=1}^{N_{\text{MC}}} f\left(IoU_{\text{3D}}\left(\begin{bmatrix}\mathbf{p}^* \\ \mathbf{d}\end{bmatrix}, \begin{bmatrix}\mathbf{p}_i \\ \mathbf{d}\end{bmatrix}\right)\right), \tag{14}$$

where vectors in the form $\begin{bmatrix}\mathbf{p} & \mathbf{d}\end{bmatrix}^{\mathrm{T}}$ represent 3D boxes used for computing 3D IoU; $f(\cdot)$ is technically a step function with a hard IoU threshold, which is 1 for $IoU_{\text{3D}} \geq IoU_{\text{threshold}}$ and 0 otherwise. In practice, we use the clamped linear function:

$$f(IoU_{\text{3D}}) = \max(0, \min(1, 2IoU_{\text{3D}} - 0.5)). \tag{15}$$

Table 1 illustrates the performance of Monte Carlo and MLP scoring methods, along with the baseline of using only 2D detection score. We observe that, although Monte Carlo scoring proves effective compared to the baseline, its performance is still much lower than the MLP scoring network. This validates that fusing both pose uncertainty and network feature can produce a more reliable confidence score.

Scoring Method	mAP
2D score only	25.40
Monte Carlo	28.19
MLP	31.47

Table 1. Comparison between different scoring methods, based on the evaluation results on KITTI validation set.

4. Complete Evaluation Results on the Official KITTI Benchmark

The official KITTI-Object benchmark consists of four tasks (2D detection, orientation similarity, 3D detection,

Benchmark	Easy	Mod.	Hard
Car (2D det.)	95.48	87.91	78.10
Car (orientation)	95.44	87.64	77.75
Car (3D det.)	19.65	12.30	10.58
Car (BEV det.)	27.94	17.34	15.24
Ped. (2D det.)	73.05	56.40	51.40
Ped. (orientation)	63.28	47.82	43.23
Ped. (3D det.)	10.88	6.78	5.83
Ped. (BEV det.)	11.70	7.59	6.34
Cyclist (2D det.)	67.47	49.13	43.41
Cyclist (orientation)	49.04	34.36	30.22
Cyclist (3D det.)	1.01	0.61	0.48
Cyclist (BEV det.)	1.14	0.73	0.66

Table 2. With LiDAR supervision.

Benchmark	Easy	Mod.	Hard
Car (2D det.)	95.65	87.76	80.12
Car (orientation)	95.48	87.33	79.51
Car (3D det.)	16.04	10.53	9.11
Car (BEV det.)	24.02	15.98	13.52
Ped. (2D det.)	71.27	55.80	49.47
Ped. (orientation)	60.13	46.00	40.61
Ped. (3D det.)	11.18	6.53	5.73
Ped. (BEV det.)	12.03	7.27	6.20
Cyclist (2D det.)	68.85	50.32	44.36
Cyclist (orientation)	37.29	27.95	25.27
Cyclist (3D det.)	0.69	0.38	0.40
Cyclist (BEV det.)	0.94	0.55	0.42

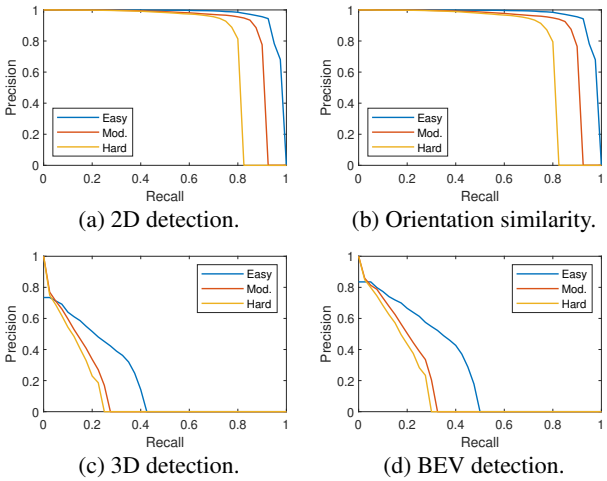
Table 3. Without LiDAR supervision (Fully self-supervised reconstruction).

BEV detection) on three classes (Car, Pedestrian, Cyclist). All metrics are based on the precision-recall curves given the IoU threshold or rotation threshold. We tested two variants of our model: with LiDAR supervision and without LiDAR supervision. The results are shown in Tables 2 and 3 and Figures 1 to 6.

References

[1] Bo Chen, Alvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

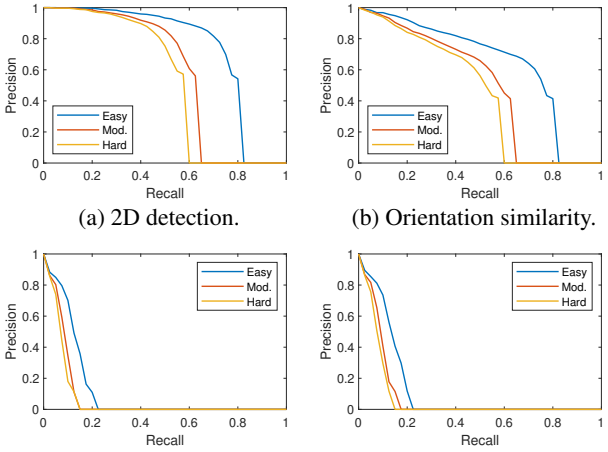
[2] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Conference on Neural Information Processing Systems (NIPS)*, 2017. 1



(a) 2D detection. (b) Orientation similarity.

(c) 3D detection. (d) BEV detection.

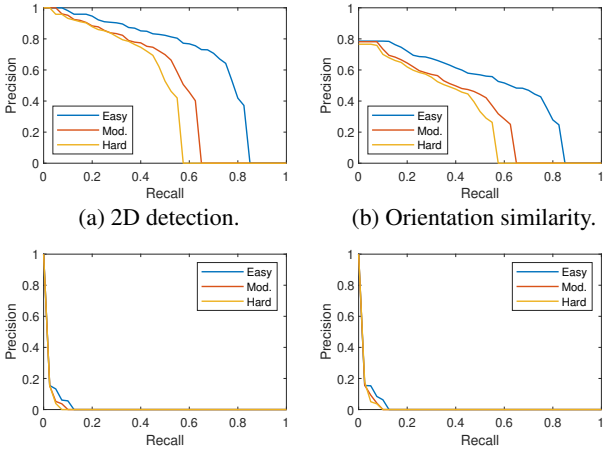
Figure 1. Car, with LiDAR supervision.



(a) 2D detection. (b) Orientation similarity.

(c) 3D detection. (d) BEV detection.

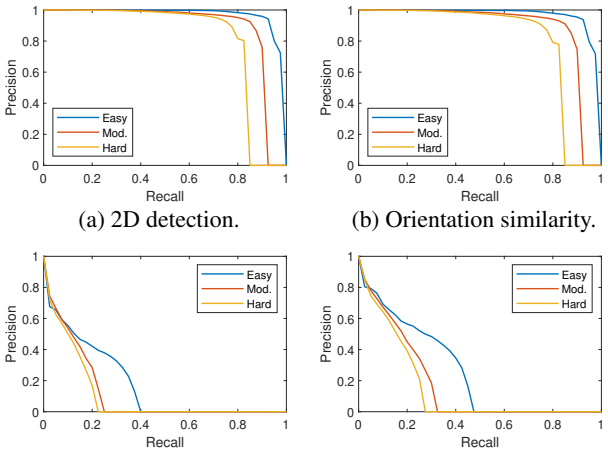
Figure 2. Pedestrian, with LiDAR supervision.



(a) 2D detection. (b) Orientation similarity.

(c) 3D detection. (d) BEV detection.

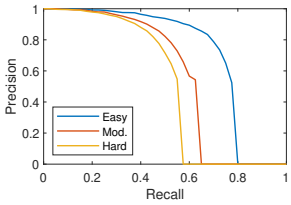
Figure 3. Cyclist, with LiDAR supervision.



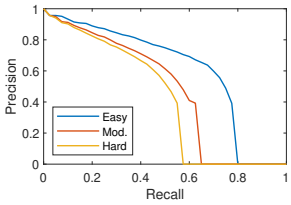
(a) 2D detection. (b) Orientation similarity.

(c) 3D detection. (d) BEV detection.

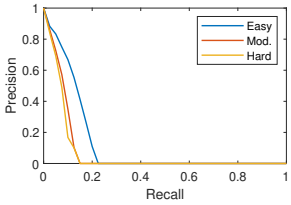
Figure 4. Car, without LiDAR supervision (Fully self-supervised reconstruction).



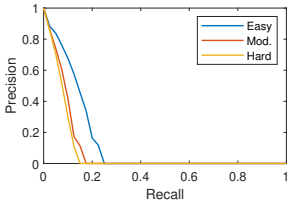
(a) 2D detection.



(b) Orientation similarity.

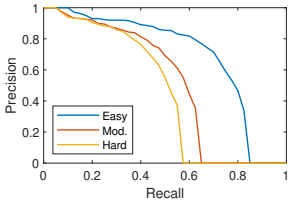


(c) 3D detection.

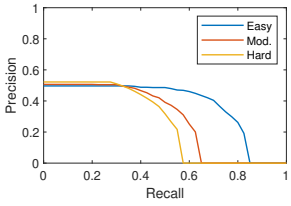


(d) BEV detection.

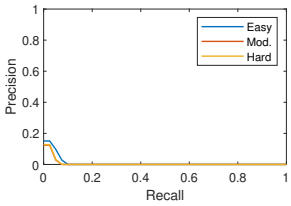
Figure 5. Pedestrian, without LiDAR supervision (Fully self-supervised reconstruction).



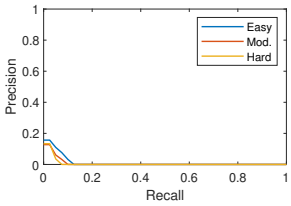
(a) 2D detection.



(b) Orientation similarity.



(c) 3D detection.



(d) BEV detection.

Figure 6. Cyclist, without LiDAR supervision (Fully self-supervised reconstruction).