Geometry-based Distance Decomposition for Monocular 3D Object Detection

Xuepeng Shi[1] Qi Ye[3] Xiaozhi Chen[2] Chuangrong Chen[2] Zhixiang Chen[1] Tae-Kyun Kim[1,4]
[1] Imperial College London [2] DJI [3] Zhejiang University
[4] Korea Advanced Institute of Science and Technology
{x.shi19, zhixiang.chen, tk.kim}@imperial.ac.uk {qi.ye}@zju.edu.cn

Abstract

Monocular 3D object detection is of great significance for autonomous driving but remains challenging. The core challenge is to predict the distance of objects in the absence of explicit depth information. Unlike regressing the distance as a single variable in most existing methods, we propose a novel geometry-based distance decomposition to recover the distance by its factors. The decomposition factors the distance of objects into the most representative and stable variables, i.e. the physical height and the projected visual height in the image plane. Moreover, the decomposition maintains the self-consistency between the two heights, leading to robust distance prediction when both predicted heights are inaccurate. The decomposition also enables us to trace the causes of the distance uncertainty for different scenarios. Such decomposition makes the distance prediction interpretable, accurate, and robust. Our method directly predicts 3D bounding boxes from RGB images with a compact architecture, making the training and inference simple and efficient. The experimental results show that our method achieves the state-of-the-art performance on the monocular 3D Object Detection and Bird's Eye View tasks of the KITTI dataset, and can generalize to images with different camera intrinsics [1].

1. Introduction

Object detection is a fundamental and challenging problem in computer vision. With the emergence of deep learning [43, 16], 2D object detection has achieved great progress in the past years [13, 12, 36, 29, 26, 25]. However, it is yet insufficient for applications requiring 3D spatial information like autonomous driving. 3D object detection, which detects objects as 3D bounding boxes, has drawn much attention. Comparing with 3D object detection methods relying on expensive LiDAR sensors to provide depth information [47, 38, 51, 22], monocular 3D object detection [5, 37] infers depth from monocular images with low computation and energy cost. The core challenge of monocular 3D object detection is inferring the distance of objects in the absence of explicit depth information. Given the visual appearance of an object, its spatial location can be inferred based on the imaging geometry [14] as an inverse problem. Thus, the priors of object physical size, scene layout, and the imaging process of cameras are essential to exploit to recover the distance.

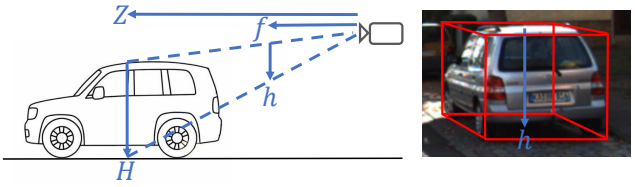


Figure 1. Our distance decomposition is based on the imaging geometry [14] of a pinhole camera. The distance from the center of an object to the camera, denoted as Z, can be calculated by $Z = \frac{fH}{h}$, where f denotes the focal length of the camera, H denotes the physical height of the object, and h denotes the length of the projected central line (PCL). The PCL represents the projection of the vertical line at the center of the 3D bounding box. This equation shows the distance of objects is determined by the physical height and projected visual height in the image plane.

On one hand, such geometric priors have been exploited to predict the pose or distance of objects by their factors. In 6D object pose estimation, PVNet [35] and SegDriven [17] regress the 2D keypoints of objects. In category-level 6D object pose and size estimation, NOCS [44] uses the normalized object coordinate space map. In stereo 3D object detection, Stereo R-CNN [23] uses sparse 2D keypoints, yaw angle, object physical size, and a region-based photometric alignment using the left and right RoIs. These works [35, 17, 23, 44] recover the pose or distance by several factors, such as 2D keypoints, 2D bounding boxes, and object physical size, which achieves interpretable and robust pose or distance estimation.

On the other hand, most monocular 3D object detection methods deal with the challenging distance prediction by regressing it as a single variable. The learning-based meth-

[1] https://github.com/Rock-100/MonoDet

ods [50, 7, 41, 42] directly learn a mapping from input images to the distance. The pseudo-LiDAR based methods [45, 31, 48] first regress the depth map of an input image and then predict the distance of objects with the depth map. The 3D-anchor-based methods [2, 3, 9] break the distance prediction into the region proposal and the offset regression. The only exceptions are [33, 19, 24], which recover the distance by minimizing the re-projection error between 3D bounding boxes and 2D bounding boxes or 2D keypoints. However, [33, 19, 24] still lag behind those methods which regress the distance as a single variable.

Aiming to close this gap, we propose a novel geometry-based distance decomposition to recover the distance by its factors. Different from [33, 19, 24], we abstract objects as vertical lines at the center of 3D bounding boxes, and their visual projection as the projection of these vertical lines, then recover the distance by them based on the imaging geometry [14], as shown in Fig. 1. The decomposition is designed to be *as simple as possible, yet effective and efficient* to extract the most representative and stable factors of the distance of objects, i.e., the physical height and the projected visual height. The advantages of the decomposition are four-fold. 1) It makes the distance prediction interpretable. The physical height can be interpreted as an intrinsic attribute of objects, and the visual height can be interpreted as the extrinsic position in a scene. 2) The physical height and projected visual height are easy to estimate as revealed by our observations. 3) The decomposition maintains the self-consistency between the two heights, leading to robust distance prediction when both predicted heights are inaccurate. 4) The decomposition enables us to trace and interpret the causes of the distance uncertainty for different scenarios, by introducing an uncertainty-aware regression loss for the decomposed variables. In addition, our method can generalize to images with different camera intrinsics, because it reasons the distance only by the local information of objects and decouples the focal length from the distance prediction. This generalization ability is crucial to facilitate the deployment of the machine learning models of monocular 3D vision [10].

The contributions of our method are summarized below:

1. A novel geometry-based distance decomposition makes the distance prediction interpretable, accurate and robust.

2. Based on the decomposition, our method originally traces the causes of the distance uncertainty.

3. Our method directly predicts 3D bounding boxes from RGB images with a compact architecture, making the training and inference simple and efficient.

4. Our method achieves the state-of-the-art (SOTA) performance on the monocular 3D Object Detection and Bird's Eye View tasks of the KITTI dataset [11], and can adapt to images with different camera intrinsics.

2. Related Work

2.1. 2D Object Detection

2D object detection has achieved sustainable improvements [13, 12, 36, 29, 26, 25] in the past years. Notably, the two-stage frameworks, such as Faster R-CNN [36] and Mask R-CNN [15], achieve dominated performance on several challenging datasets [27, 11]. Feature pyramid networks (FPN) [25] has also been proposed to improve the 2D object detection performance. We adopt Faster R-CNN [36] with FPN [25] as our 2D object detection framework, because of its high accuracy and flexibility.

2.2. Monocular 3D Object Detection

Most monocular 3D object detection methods deal with the challenging distance prediction by regressing it as a single variable. The learning-based methods [50, 7, 41, 42] directly regress the distance of objects by adding distance branches to 2D object detectors, which are simple and efficient. The pseudo-LiDAR-based methods [45, 31, 48] first predict the depth map of an input image using an external monocular depth estimator, then predict the distance of objects from the estimated depth map using a point-cloud-based 3D object detector. Though the explicit depth cues from the the estimated depth map can ease the distance prediction, the generalization of these methods is bounded by that of the monocular depth estimators [40]. The 3D-anchor-based methods [2, 3, 9] extend the 2D anchor boxes [36] to the 3D anchor boxes by supplementing 3D bounding box templates, then predict the transformations from the 3D anchor boxes to the ground-truth 3D bounding boxes. The 3D anchor boxes can ease the distance learning. Unlike regressing the distance as a single variable in these methods, we propose a novel geometry-based distance decomposition to recover the distance by its factors.

A few works [33, 19, 24] predict the distance of objects by its factors. Deep3Dbox [33] abstracts objects as 3D bounding boxes and their visual projection as the four boundaries of projected 3D bounding boxes, then recovers the distance by minimizing the re-projection error between the four boundaries of projected 3D bounding boxes and 2D bounding boxes. The keypoint-based methods [19, 24] abstract objects as 3D bounding boxes and their visual projection as the eight projected corners of 3D bounding boxes, then recovers the distance by minimizing the re-projection error between the eight projected corners of 3D bounding boxes and the predicted eight projected corners. However, [33, 19, 24] still lag behind those methods which regress the distance as a single variable. Similar to [33, 19, 24],

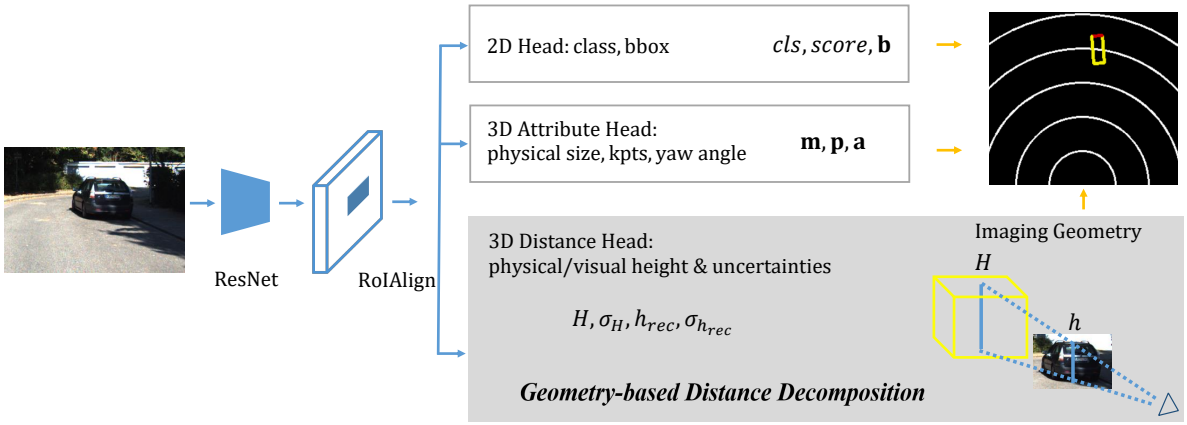


Figure 2. The main architecture of MonoRCNN. MonoRCNN is built upon Faster R-CNN [36] and adds the carefully designed 3D distance head. The 3D distance head is based on our geometry-based distance decomposition. Specifically, our method regresses H, $h_{rec} = \frac{1}{h}$, and their uncertainties, then recovers the distance by $Z = fHh_{rec}$. Blue arrows represent operations in the network during training and inference, and orange arrows represent operations to recover 3D bounding boxes during inference.

our method also recovers the distance by its factors, but our decomposition is more simple and effective.

2.3. Geometry-based Object Pose Estimation

Geometric priors have been exploited to predict the pose or distance of objects by their factors. In 6D object pose estimation, PVNet [35] and SegDriven [17] regress 2D keypoints of objects, then optimize the estimation of the 6D pose by solving a Perspective-n-Point (PnP) problem. In category-level 6D object pose and size estimation, NOCS [44] uses the normalized object coordinate space map together with the depth map in a pose fitting algorithm, to estimate the 6D pose and physical size of unseen objects. In stereo 3D object detection, stereo R-CNN [23] first calculates the coarse distance from sparse 2D keypoints, yaw angle, and object physical size, then recovers the accurate distance by a region-based photometric alignment using the left and right RoIs. Inspired by these works, we propose a geometry-based distance decomposition for monocular 3D object detection, to recover the distance by its factors.

2.4. Uncertainty Estimation

There are two seminal works [20, 21] exploring uncertainties in deep learning for computer vision. The uncertainty-aware regression loss [21] enables networks to re-balance samples and re-focus on more reasonable samples, which improves the overall accuracy. MonoLoco [1], MonoPair [7] and UR3D [39] regress the distance of objects with the uncertainty-aware regression loss [21] to improve the distance regression accuracy. MonoDIS [41] proposes a self-supervised confidence score to re-sort the predicted 3D bounding boxes. Kinematic3D [3] proposes a self-balancing 3D confidence loss to both improve the 3D box regression accuracy and re-sort the predicted 3D bounding boxes. [1, 7, 39, 41, 3] directly apply the uncertainty-aware losses to the distance. Instead, we apply the uncertainty-aware regression loss [21] to the decomposed variables of the distance, which enables us to trace the causes of distance uncertainty for different scenarios.

3. Proposed MonoRCNN

We first present the basic framework, then two 3D-related detection heads, i.e., the 3D distance head and 3D attribute head. We detail the geometry-based distance decomposition and uncertainty-aware regression in the 3D distance head. We term our method as MonoRCNN, and the main architecture is illustrated in Fig. 2.

3.1. Basic Framework

We address monocular 3D object detection, which predicts the 3D bounding boxes of objects from monocular RGB images. Two common assumptions [11] are 1) only considering the yaw angle of 3D bounding boxes and setting the roll and pitch angle as zero, 2) per-image camera intrinsics are available both during training and inference. For a given RGB image, MonoRCNN reports all objects within concerned categories, and the output for each object is

1. class label cls and confidence $score$,
2. 2D bounding box represented by the top-left and bottom-right corners, denoted as $\mathbf{b} = (x_1, y_1, x_2, y_2)$,
3. the 2D projected center of the 3D bounding box, denoted as $\mathbf{p} = (p_1, p_2)$,
4. the physical size of the 3D bounding box, denoted as $\mathbf{m} = (W, H, L)$, where W, H, L are the physical width, height, and length, respectively,

Figure 3. Comparisons between the predicted eight projected corners (red boxes) and predicted visual height (blue lines). Predicting the eight projected corners fails under challenging cases, such as occlusion, truncation, and extreme lighting conditions, while predicting the visual height is more simple and robust. The images are from the val subset of the KITTI validation split [6].

5. the yaw angle of the 3D bounding box, denoted as $\mathbf{a} = (\sin(\theta), \cos(\theta))$, where θ is the allocentric pose of the 3D bounding box,

6. the distance of the center of the 3D bounding box, denoted as Z.

MonoRCNN predicts the 3D center (p_1, p_2, Z) in pixel coordinates, and convert it to camera coordinates using the projection matrix $\mathbf{P}$ during inference, formulated as

$$\begin{bmatrix} p_1 \cdot Z \\ p_2 \cdot Z \\ Z \end{bmatrix}_{\mathrm{P}} = \mathbf{P} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{\mathrm{C}}. \tag{1}$$

For the yaw angle prediction, MonoRCNN predicts $\sin(\theta)$ and $\cos(\theta)$, and convert them to θ during inference.

MonoRCNN is built upon Faster R-CNN [36]. We use a ResNet-50 [16] with FPN [25] as the backbone and RoIAlign [15] to extract the crops of object features. For the training and inference of the 2D object detection network, we follow the pipelines in [36, 15]. To adapt to monocular 3D object detection, we add the 3D distance head and 3D attribute head.

3.2. 3D Distance Head

3.2.1 Geometry-based Distance Decomposition

The 3D distance head recovers the distance of objects and is based on our geometry-based distance decomposition. Specifically, we decompose the distance of an object Z, into the physical height H, and the reciprocal of the projected visual height $h_{rec} = \frac{1}{h}$, which is formulated as

$$Z = \frac{fH}{h} = fHh_{rec}, \tag{2}$$

where f denotes the focal length of the camera. We regress H and h_{rec} separately and recover Z by them.

The decomposition makes the distance prediction interpretable. H can be interpreted as an intrinsic attribute of objects, the estimation of which can be regarded as a fine-grained object classification problem. While given an object, h_{rec} can be interpreted as the extrinsic position in a scene, the estimation of which is a 2D regression problem in the image plane.

	Mean prediction error (meters) ↓		
	S & F & B	S	F & B
Length	0.293	0.276	0.296
Width	0.071	0.078	0.070
Height	0.078	0.076	0.078

Table 1. The prediction error of the physical size within different yaw angle ranges on the val subset of the KITTI validation split [6]. 'S', 'F', and 'B' denote cars whose visible parts are side, front, and back, respectively.

The physical height and visual height required by our decomposition are easy to estimate. Predicting the eight projected corners of 3D bounding boxes [19, 24] is challenging due to the occlusion, truncation, yaw angle variations, and extreme lighting conditions. As shown in Fig. 3, the visual height prediction is accurate in different challenging cases but the projected corner prediction fails. Moreover, the physical height is the simplest and the most stable variable among the physical size, according to the prediction error of the physical size of cars on the val subset of the KITTI validation split [6], shown in Tab. 1. The mean prediction error of the physical height is much smaller than that of the physical length. In addition, the prediction error of the physical length and width are influenced by the yaw angle due to single-view ambiguity, while the prediction error of the physical height is not. Our decomposition only uses the physical height, instead of the full physical size [19, 24], to recover the distance, which improves the distance prediction accuracy.

The decomposition can also maintain the self-consistency during inference, leading to robust distance prediction when both predicted heights are inaccurate. With the objective of predicting the distance, the neural network can learn the correlation between H and h_{rec} during training, then the learned correlation can serve as the self-consistency during inference. This correlation is detailed as below. In the context of monocular 3D object detection task, the physical height H is a constant for a given object. The distance of the object to the camera Z can be modeled as a random variable since the object can appear in different locations in a scene. Similarly, the reciprocal of the length of the PCL of the object h_{rec} is also a random variable. While the variable Z is random for a specified object, we notice that this variable is expected to follow a same distribution for different objects, denoted as $\mathcal{D}$. This is because the distribution of the location of an object is irrelevant to its fine-grained object type. For example, the spatial positions of cars on a street are not influenced by their car types. We formulate this as

$$Z = fHh_{rec} \sim \mathcal{D}. \tag{3}$$

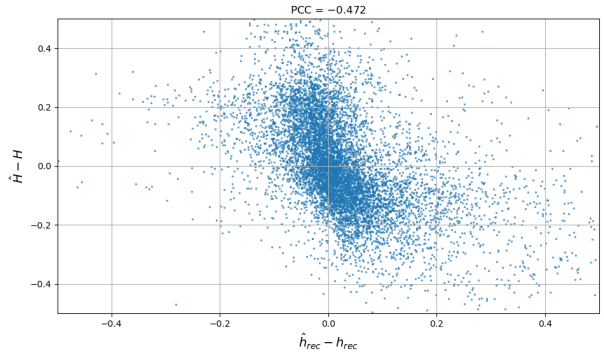


Figure 4. The learned correlation between H and h_{rec} on the val subset of the KITTI validation split [6]. The Pearson correlation coefficient (PCC) of $\hat{h}_{rec} - h_{rec}$ and $\hat{H} - H$ is -0.472. Errors are normalized to $[-0.5, 0.5]$.

By taking expectation on Eq. (3), we have

$$H\mathbb{E}[h_{rec}] = \frac{\mathbb{E}[Z]}{f}. \tag{4}$$

Eq. (4) shows that, for the training labels of different objects, the product between their H and their expectation of h_{rec} is a constant. In other words, for different objects, the expectation of h_{rec} decreases with the increase of H. *Intuitively speaking, the larger the physical height of an object, the larger the average projected visual height of this object*. This is the correlation between the training labels of H and h_{rec}. The neural network can learn this correlation during training, as shown in Fig. 4. During inference, if the predicted H is larger than the groundtruth, the learned correlation pushes the predicted h_{rec} to become smaller on average, and vice versa. Thus, our method can recover the accurate distance Z with the inaccurate H and h_{rec}, i.e., maintain the self-consistency during inference.

3.2.2 Uncertainty-aware Regression

Based on the decomposition, we further trace and interpret the causes of the distance uncertainty for different scenarios. We modify the uncertainty-aware regression loss [21] to regress H and h_{rec}. The loss functions for H and h_{rec} can be formulated as

$$L_H = \frac{L_1(\hat{H}, H)}{\sigma_H} + \lambda_H log(\sigma_H), \tag{5}$$

$$L_{h_{rec}} = \frac{L_1(\hat{h}_{rec}, h_{rec})}{\sigma_{h_{rec}}} + \lambda_{h_{rec}} log(\sigma_{h_{rec}}), \tag{6}$$

where $\hat{H}$ and $\hat{h}_{rec}$ are the groundtruths, H and h_{rec} are the predictions, λ_H and $\lambda_{h_{rec}}$ are the positive parameters to balance the uncertainty terms, and σ_H and $\sigma_{h_{rec}}$ are the learnable variables of uncertainties.

In Fig. 5, we show the uncertainties of the physical height and the projected visual height, i.e., σ_H and $\sigma_{h_{rec}}$,

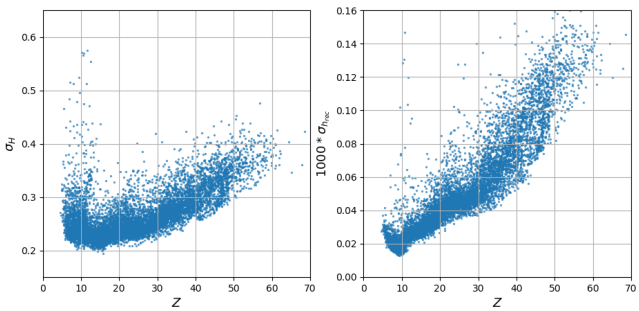


Figure 5. The uncertainties of H (left) and h_{rec} (right) vs. distance Z (meters) on the val subset of the KITTI validation split [6].

for objects at different distances on the val subset of the KITTI validation split [6]. For both H and h_{rec}, the uncertainties first decrease and then increase when objects go far away from the camera. At a close distance, the high uncertainties are mainly caused by the truncated views of objects. It is hard to make accurate prediction with partial observations. At a far distance, the high uncertainties are mainly caused by the coarse views of objects with fewer pixels representing them in images. The truncated and coarse views result in comparable increases for the uncertainties of H. However, h_{rec} is with noticeable higher uncertainties for the coarse views than the truncated views. In other words, the accuracy of the distance estimation for faraway objects is heavily influenced by the accuracy of h_{rec}.

$\sigma_{h_{rec}}$ is more discriminative for objects at different distance than σ_H, and $fH\sigma_{h_{rec}}$ can represent the distance uncertainty. We use $\frac{score}{fH\sigma_{h_{rec}}}$, instead of $score$, to sort the predicted boxes to improve the 3D object detection accuracy.

3.3. 3D Attribute Head

3D attribute head predicts the physical size, yaw angle, and 2D keypoints, i.e., the projected center and corners of 3D bounding boxes. We use the L_1 loss to directly regress the physical size and yaw angle, formulated as

$$L_{size} = L_1(\hat{\mathbf{m}}, \mathbf{m}), \tag{7}$$

$$L_{yaw} = L_1(\hat{\mathbf{a}}, \mathbf{a}), \tag{8}$$

where $\hat{\mathbf{m}}$ and $\hat{\mathbf{a}}$ are the groundtruths, and $\mathbf{m}$ and $\mathbf{a}$ are the predictions. For the keypoint regression, we normalize the keypoints by their proposal size. Let (x_1, y_1, x_2, y_2) denote the top-left and bottom-right corners of a proposal, and $\hat{\mathbf{p}} = (\hat{p}_1, \hat{p}_2)$ and $\mathbf{p} = (p_1, p_2)$ denote the groundtruth keypoint and the predicted keypoint, respectively. Let $\hat{\mathbf{t}}$ and $\mathbf{t}$ denote the normalized groundtruth keypoint and the normalized predicted keypoint, respectively, and $\hat{\mathbf{t}}$ is defined as

$$\hat{\mathbf{t}} = (\frac{\hat{p}_1 - x_1}{x_2 - x_1}, \frac{\hat{p}_2 - y_1}{y_2 - y_1}). \tag{9}$$

The keypoint loss function can be formulated as

$$L_{kpt} = L_1(\hat{\mathbf{t}}, \mathbf{t}). \tag{10}$$

During inference, we transform the normalized predicted keypoint $\mathbf{t}$ to the predicted keypoint $\mathbf{p}$. We only use the projected center of a 3D bounding box during inference, the losses of the eight projected corners are auxiliary losses during training.

3.4. Overall Loss

The overall training loss function for the detection heads is

$$L = \lambda_{cls}L_{cls} + \lambda_{bbox}L_{bbox} + \lambda_{size}L_{size} + \lambda_{yaw}L_{yaw} + \lambda_{kpt}L_{kpt} + L_H + L_{h_{rec}}, \quad (11)$$

where λ_{cls} is 1, λ_{bbox} is 1, λ_{size} is 3, λ_{yaw} is 5, λ_{kpt} is 5, λ_H is 0.25, and $\lambda_{h_{rec}}$ is 1.

3.5. Implementation Details

The backbone of MonoRCNN is ResNet-50 [16] with FPN [25] and is pre-trained on ImageNet [8]. We extract ROI features from P2, P3, P4 and P5 of the backbone, as defined in [25]. We use five scale anchors of $\{32, 64, 128, 126, 512\}$ with three ratios $\{0.5, 1, 2\}$, and tile the anchors on P4. Images are scaled to a fixed height of 512 pixels for both training and inference. During training the batch size is 4, and the total iteration number is 1×10^5 and 2×10^5 on the training subset of the KITTI validation split [6] and the KITTI official test split [11], respectively. We adopt the step strategy to adjust the learning rate. The initial learning rate is 0.01 and reduced by 10 times after 60%, 80%, 90% iterations. During training random mirroring is used as augmentation, and during inference no augmentation is used. We implement our method with PyTorch [34] and Detectron2 [46]. All the experiments run on a server with 2.2 GHz CPU and GTX Titan X.

4. Experiments

We first analyze the ablation studies and self-consistency on the KITTI validation split [6]. Then we comprehensively benchmark MonoRCNN on the KITTI official test dataset [11]. We also present the cross-dataset test results using a nuScenes [4] cross-test set. Finally, we visualize qualitative examples on the KITTI dataset [11] in Fig. 6, and the nuScenes [4] cross-test set in Fig. 7 [2].

4.1. Datasets

The KITTI dataset [11] provides multiple widely used benchmarks for computer vision problems in autonomous driving. The Bird's Eye View (BEV) and 3D Object Detection tasks are used to evaluate the 3D localization performance. These two tasks are characterized by 7481 training and 7518 test images with 2D and 3D annotations for cars, pedestrians, cyclists, etc. Each object is assigned with a difficulty level, i.e., easy, moderate or hard, based on its visual size, occlusion level and truncation degree. We conduct experiments on two common data splits, the val split [6] and the official test split [11]. We only use the images from the left cameras for training. We report the $\text{AP}|_{R_{40}}$ [41] to compare the accuracy. We use the car class, the most representative class, and the official IoU criteria 0.7 for cars.

The nuScenes [4] 3D object detection task requires detecting 10 object classes in terms of full 3D bounding boxes, attributes and velocities. In this work, we focus on detecting the 3D bounding boxes of cars, to test the cross-dataset performance from KITTI [11] to nuScenes [4]. We use the script [3] for converting nuScenes data to KITTI format to generate a cross-test set. The cross-test set consists of 6019 front camera images from the official val subset.

	$\text{AP}\|_{R_{40}}$ [Easy / Mod / Hard] ↑	
	AP_{3D}	AP_{BEV}
L	15.01 / 10.52 / 8.45	21.03 / 14.84 / 11.44
K	13.58 / 8.96 / 7.06	19.39 / 13.59 / 10.54
D	15.78 / 10.97 / 8.15	22.06 / 15.52 / 11.82
D+U	16.94 / 12.00 / 9.46	24.60 / 17.23 / 13.38
D+U+S	16.61 / 13.19 / 10.65	25.29 / 19.22 / 15.30

Table 2. **Ablation Studies** on the val subset of the KITTI validation split [6]. 'L' means directly regressing the distance. 'K' means using the eight projected corners and physical size to recover the distance, similar to [19, 24]. 'D' means using our decomposition. 'U' means adding the uncertainty-aware regression loss [21]. 'S' means sorting the predicted boxes by $\frac{score}{fH\sigma_{h_{rec}}}$.

4.2. Ablation Studies

We conduct ablation studies to examine how each proposed component affects the final performance. We evaluate the performance by first setting a baseline which utilizes our decomposition, then adding the uncertainty-aware loss [21], finally sorting the predicted boxes by $\frac{score}{fH\sigma_{h_{rec}}}$, as shown in Tab. 2. We also implement a model which directly regresses the distance to compare our method with the learning-based methods [50, 7, 41, 42], and a model which recovers the distance by the eight projected corners and physical size to compare our method with the keypoint-based methods [19, 24]. From Tab. 2, we can see:

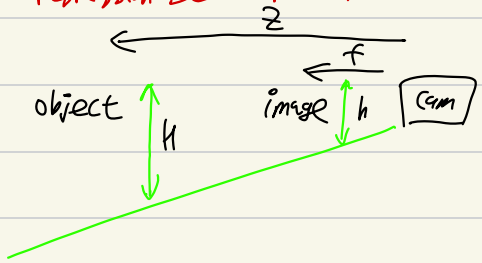
1) The geometry-based distance decomposition is effective. Comparing 'D' with 'K', we can see the geometry-based distance decomposition outperforms the keypoint-based model by a large margin, which supports its effectiveness.

[2] More qualitative examples can be found in the supplementary file.

[3] https://github.com/nutonomy/nuscenes-devkit/blob/master/python-sdk/nuscenes/scripts/export_kitti.py

◦ Mono RCNN

Regression 으로 depth 정보 얻기 ⇒ geometry 로 decompose



$z : f = H : h$

$z = \frac{f \cdot H}{h}$

H: physical size (object), h: image size (object)

Pre trained Faster RCNN + 3D Attribute Head + 3D Distance Head

Input → ResNet → RoI Align → { 2D Head, 3D Attribute Head, 3D Distance Head }

① 2D Head = cls, score, box

② 3D Attribute Head = M, P, a

$M = (W, h, l)$

$P = (P_1, P_2)$ ⇒ 3D Bbox center → 2D projected center

a = yaw angle → $(\cos\theta, \sin\theta)$

③ 3D Distance Head = depth

$z = \frac{f \cdot H}{h}$

⇒ Inference, 3D Bbox recover

Method	Input	Time (ms)	$AP\|_{R_{40}}$ [Easy / Mod / Hard] ↑ AP_{3D}	AP_{BEV}
ROI-10D (CVPR 19) [32]	image + depth	200	4.32 / 2.02/ 1.46	9.78 / 4.91/ 3.74
AM3D (ICCV 19) [31]	image + depth	400	16.50 / 10.74 / 9.52	25.03 / 17.32/ 14.91
D^4LCN (CVPR 20) [9]	image + depth	200	16.65 / 11.72 / 9.51	22.51 / 16.02/ 12.55
DA-3Ddet (ECCV 20) [48]	image + depth	400	16.77 / 11.50 / 8.93	23.35 / 15.90/ 12.11
PatchNet (ECCV 20) [30]	image + depth	400	15.68 / 11.12 / 10.17	22.97 / 16.86/ 14.97
Kinematic3D (ECCV 20) [3]	image + video	120	19.07 / 12.72 / 9.17	26.69 / 17.52/ 13.10
FQNet (CVPR 19) [28]	image	500	2.77 / 1.51 / 1.01	5.40 / 3.23/ 2.46
M3D-RPN (ICCV 19) [2]	image	160	14.76 / 9.71 / 7.42	21.02 / 13.67/ 10.23
MonoPair (CVPR 20) [7]	image	60	13.04 / 9.99 / 8.65	19.28 / 14.83/ 12.89
MoVi-3D (ECCV 20) [42]	image	–	15.19 / 10.90 / 9.26	22.76 / 17.03/ 14.85
RTM3D (ECCV 20) [24] †	image	50	14.41 / 10.34 / 8.77	19.17 / 14.20/ 11.99
MonoRCNN (Ours)	image	70	18.36 / 12.65 / 10.03	25.48 / 18.11/ 14.10

Table 3. **Comparisons on the KITTI benchmark [11]**. 'Input' means the input data modality used during training and inference. The inference time is reported from the official leaderboard with slight variances in hardware. Red / blue indicate the best / second, respectively. † denotes methods using additional images from right cameras for training.

	$AP\|_{R_{40}}$ [Easy / Mod / Hard] ↑ AP_{3D}	AP_{BEV}
K (P)	13.58 / 8.96 / 7.06	19.39 / 13.59 / 10.54
K (G)	13.31 / 9.51 / 7.45	21.13 / 14.98 / 11.44
Ours (P)	16.94 / 12.00 / 9.46	24.60 / 17.23 / 13.38
Ours (G)	14.45 / 10.82 / 8.62	22.10 / 16.45 / 12.81

Table 4. **Self-Consistency Comparisons** on the val subset of the KITTI validation split [6]. 'K' means using the eight projected corners and physical size to recover the distance, similar to [19, 24]. 'P' means using the predicted physical height or size when recovering the distance, 'G' means using the groundtruth instead.

2) The uncertainty-aware regression loss [21] improves the accuracy. Comparing 'D+U' with 'D', we can see using the uncertainty-aware regression loss [21] for H and h_{rec} improves the accuracy.

3) Sorting by $\frac{score}{fH\sigma_{h_{rec}}}$ is effective. Comparing 'D+U+S' with 'D+U', we can see sorting by $\frac{score}{fH\sigma_{h_{rec}}}$ improves the accuracy, showing that $\frac{score}{fH\sigma_{h_{rec}}}$ represents the 3D prediction quality better than $score$.

4.3. Self-Consistency Comparisons

We analyze the self-consistency as shown in Tab. 4. From 'Ours', we can see that, the accuracy decreases if the predicted physical height for recovering the distance is replaced with the groundtruth physical height. This supports our method can maintain the self-consistency during inference. From 'K', we can see that, the accuracy increases if the predicted physical size for optimizing the distance is replaced with the groundtruth physical size. This shows there is no correlation, i.e., self-consistency, between the predicted projected corners and physical size.

4.4. Comparisons on the KITTI benchmark

We comprehensively benchmark MonoRCNN on the KITTI official test dataset [11] in Tab. 3, and show the advantages of our method compared with existing methods. From Tab. 3, we can see:

1) MonoRCNN achieves the SOTA accuracy. Existing image-only methods are not able to emulate the methods using extra depth input, while our method pushes the forefront of image-only methods [42], by $3.17/1.75$ in AP_{3D} and $2.72/1.08$ in AP_{BEV} on the easy and moderate subset, and surpasses those depth-based methods by $1.59/0.93$ in AP_{3D} and $0.45/0.79$ in AP_{BEV} on the easy and moderate subset. With only single frame as input, our method is even comparable with a video-based method [3]. Notice that MonoRCNN uses a ResNet-50 backbone [16], while [2] uses a DenseNet-121 backbone [18] and [7, 24] use DLA-34 backbones [49]. Though DenseNet-121 [18] and DLA-34 [49] are more advanced than our ResNet-50 [16], MonoRCNN still outperforms [2, 7, 24]. We also emphasize that MonoRCNN significantly outperforms RTM3D [24], a keypoint-based method, though [24] uses additional images from right cameras for training. This supports that our distance decomposition is much better than the keypoint-based distance decomposition.

2) MonoRCNN is simple and efficient. MonoRCNN is an image-only method, thus is more simple and efficient than those depth-based and video-based methods. The monocular depth estimator in [9, 31, 48, 30] uses a heavy ResNet-101 backbone [16], and MonoRCNN runs 3 times faster than [9], and 5 times faster than [31, 48, 30].

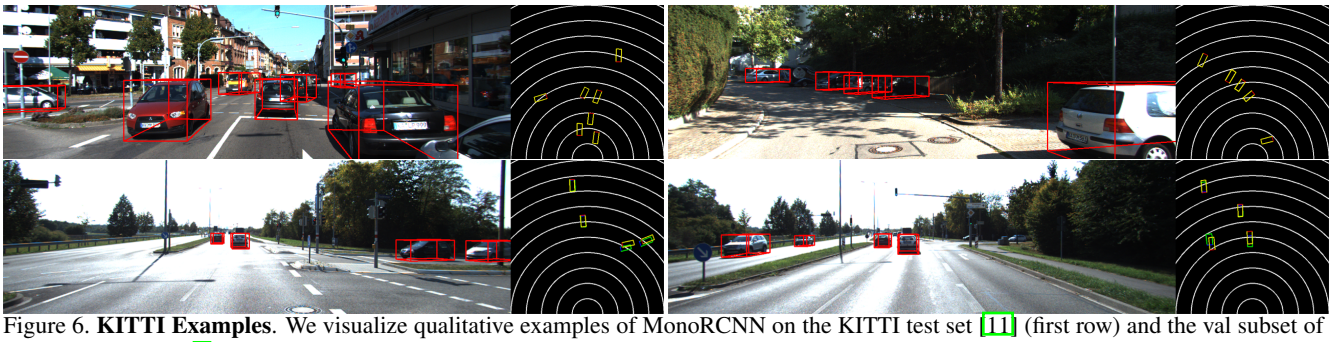


Figure 6. **KITTI Examples**. We visualize qualitative examples of MonoRCNN on the KITTI test set [11] (first row) and the val subset of the validation split [6] (second row). The red boxes in the image planes represent the 2D projections of the predicted 3D bounding boxes. The yellow / green boxes in the bird's eye view results represent the predictions and groundtruths, respectively, and the red / blue lines indicate the yaw angle of cars. The radius difference between two adjacent white circles is 5 meters. All images are not used for training.

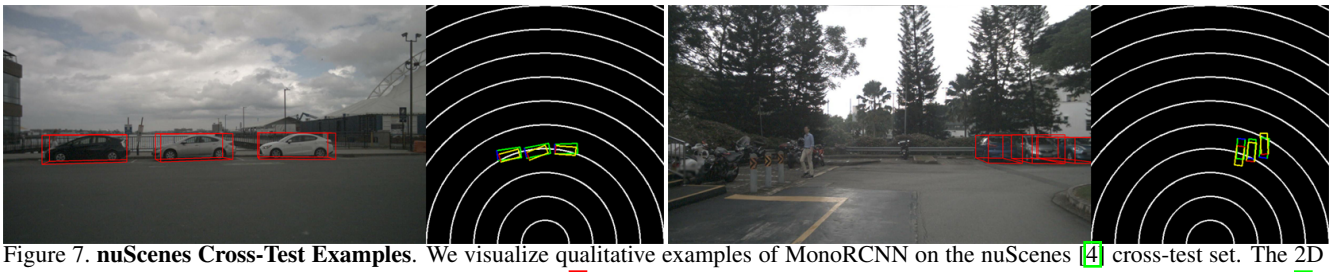


Figure 7. **nuScenes Cross-Test Examples**. We visualize qualitative examples of MonoRCNN on the nuScenes [4] cross-test set. The 2D projections and bird's eye view results are shown as in Fig. 6. Our model is only trained with the training subset of the KITTI val split [6], and can generalize to images in the nuScenes [4] cross-test set with different camera intrinsics.

4.5. Cross-Dataset Test

To evaluate the ability to generalize to images with different camera intrinsics, we conduct cross-dataset test by applying the model trained with the training subset of the KITTI val split [6] to the nuScenes [4] cross-test set. Since this paper is the first paper presenting cross-dataset test in monocular 3D object detection, we also provide the results of M3D-RPN [2] using its official model [4] as a comparison. To focus on the accuracy of the distance prediction, we report the mean error of the distance prediction within different distance ranges for both methods in Tab. 5. The errors are calculated for recalled objects. The results show that our method achieves lower distance prediction errors. From errors in different distance intervals, we can see that the further the objects are, the better generalization of our method is. This is because our method reasons the distance by the local geometric variables of objects. We also visualize some qualitative examples in Fig. 7, and we can see our method achieves accurate distance prediction. The pseudo-LiDAR-based methods [45, 31, 48] suffer from the difficulty of generalizing to images with different camera intrinsics, because it is difficult for monocular depth estimators to generalize to images with different camera intrinsics [10].

[4] https://github.com/garrickbrazil/M3D-RPN

	Distance prediction mean error (meters) ↓			
	$[0, +\infty)$	$[0, 20)$	$[20, 40)$	$[40, +\infty)$
[2] (T)	1.26	0.56	1.33	2.73
[2] (N)	2.75	1.04	3.29	10.73
Ours (T)	1.14	0.46	1.27	2.59
Ours (N)	2.39	0.94	2.84	8.65

Table 5. **Cross-Dataset Test** on the nuScenes [4] cross-test set. We show the mean error of the distance prediction within different distance ranges on different test datasets. 'T' means the val subset of the KITTI val split [6], and 'N' means the nuScenes [4] cross-test set. All models are only trained with the training subset of the KITTI val split [6].

5. Conclusion

We have proposed a novel geometry-based distance decomposition which makes the distance prediction interpretable, accurate, and robust. Our method directly predicts 3D bounding boxes from RGB images with a compact architecture, thus is simple and efficient. The experimental results show that our method achieves the SOTA performance on the monocular 3D Object Detection and Bird's Eye View tasks of the KITTI dataset, and can generalize to images with different camera intrinsics.

References

[1] Lorenzo Bertoni, Sven Kreiss, and Alexandre Alahi. Monoloco: Monocular 3d pedestrian localization and uncertainty estimation. In *ICCV*, 2019.

[2] Garrick Brazil and Xiaoming Liu. M3D-RPN: monocular 3d region proposal network for object detection. In *ICCV*, 2019.

[3] Garrick Brazil, Gerard Pons-Moll, Xiaoming Liu, and Bernt Schiele. Kinematic 3d object detection in monocular video. In *ECCV*, 2020.

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[5] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016.

[6] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G. Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *NeurIPS*, 2015.

[7] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang Li. Monopair: Monocular 3d object detection using pairwise spatial relationships. In *CVPR*, 2020.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[9] Mingyu Ding, Yuqi Huo, Hongwei Yi, Zhe Wang, Jianping Shi, Zhiwu Lu, and Ping Luo. Learning depth-guided convolutions for monocular 3d object detection. In *CVPR*, 2020.

[10] José M. Fácil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. Camconvs: Camera-aware multi-scale convolutions for single-view depth. In *CVPR*, 2019.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012.

[12] Ross B. Girshick. Fast R-CNN. In *ICCV*, 2015.

[13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[17] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *CVPR*, 2019.

[18] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[19] Eskil Jörgensen, Christopher Zach, and Fredrik Kahl. Monocular 3d object detection and box fitting trained end-to-end using intersection-over-union loss. *CoRR*, 2019.

[20] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 2017.

[21] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

[22] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[23] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo R-CNN based 3d object detection for autonomous driving. In *CVPR*, 2019.

[24] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. RTM3D: real-time monocular 3d detection from object keypoints for autonomous driving. In *ECCV*, 2020.

[25] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[26] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[27] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014.

[28] Lijie Liu, Jiwen Lu, Chunjing Xu, Qi Tian, and Jie Zhou. Deep fitting degree scoring network for monocular 3d object detection. In *CVPR*, 2019.

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016.

[30] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. Rethinking pseudo-lidar representation. In *ECCV*, 2020.

[31] Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, and Xin Fan. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In *ICCV*, 2019.

[32] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. ROI-10D: monocular lifting of 2d detection to 6d pose and metric shape. In *CVPR*, 2019.

[33] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[35] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019.

[36] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[37] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. In *BMVC*, 2019.

[38] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[39] Xuepeng Shi, Zhixiang Chen, and Tae-Kyun Kim. Distance-normalized unified representation for monocular 3d object detection. In *ECCV*, 2020.

[40] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, and Elisa Ricci. Demystifying pseudo-lidar for monocular 3d object detection. *CoRR*, 2020.

[41] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Manuel Lopez-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *ICCV*, 2019.

[42] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Elisa Ricci, and Peter Kontschieder. Towards generalization across depth for monocular 3d object detection. In *ECCV*, 2020.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[44] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *CVPR*, 2019.

[45] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark E. Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019.

[46] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[47] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: real-time 3d object detection from point clouds. In *CVPR*, 2018.

[48] Xiaoqing Ye, Liang Du, Yifeng Shi, Yingying Li, Xiao Tan, Jianfeng Feng, Errui Ding, and Shilei Wen. Monocular 3d object detection via feature domain adaptation. In *ECCV*, 2020.

[49] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *CVPR*, 2018.

[50] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, 2019.

[51] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.

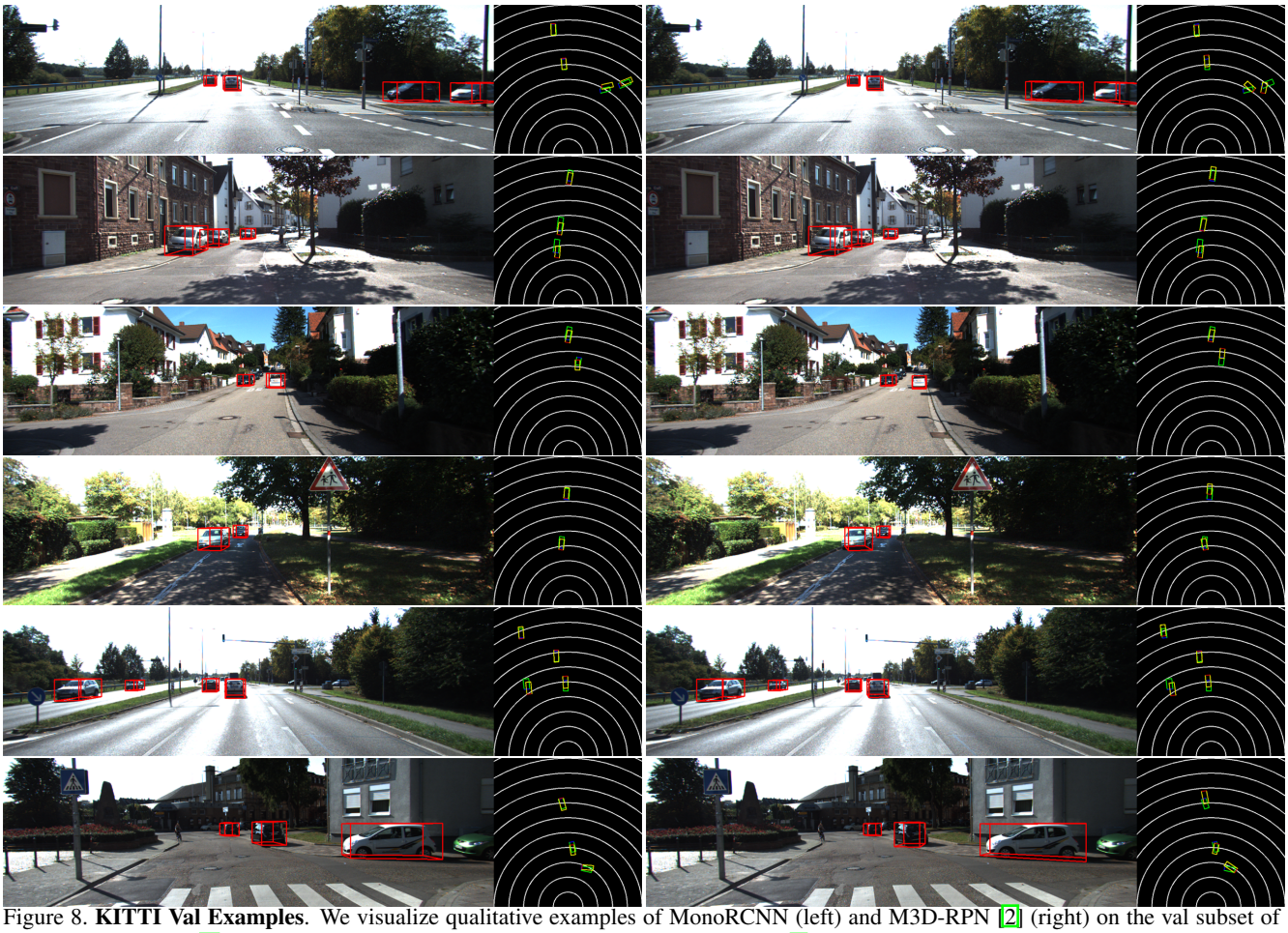


Figure 8. **KITTI Val Examples**. We visualize qualitative examples of MonoRCNN (left) and M3D-RPN [2] (right) on the val subset of the KITTI val split [6]. We can see our method is more accurate than M3D-RPN [2]. The red boxes in the image planes represent the 2D projections of the predicted 3D bounding boxes. The yellow / green boxes in the bird's eye view results represent the predictions and groundtruths of the 3D bounding boxes, respectively, and the red / blue lines indicate the yaw angle of cars. The radius difference between two adjacent white circles is 5 meters. All illustrated images are not used for training.

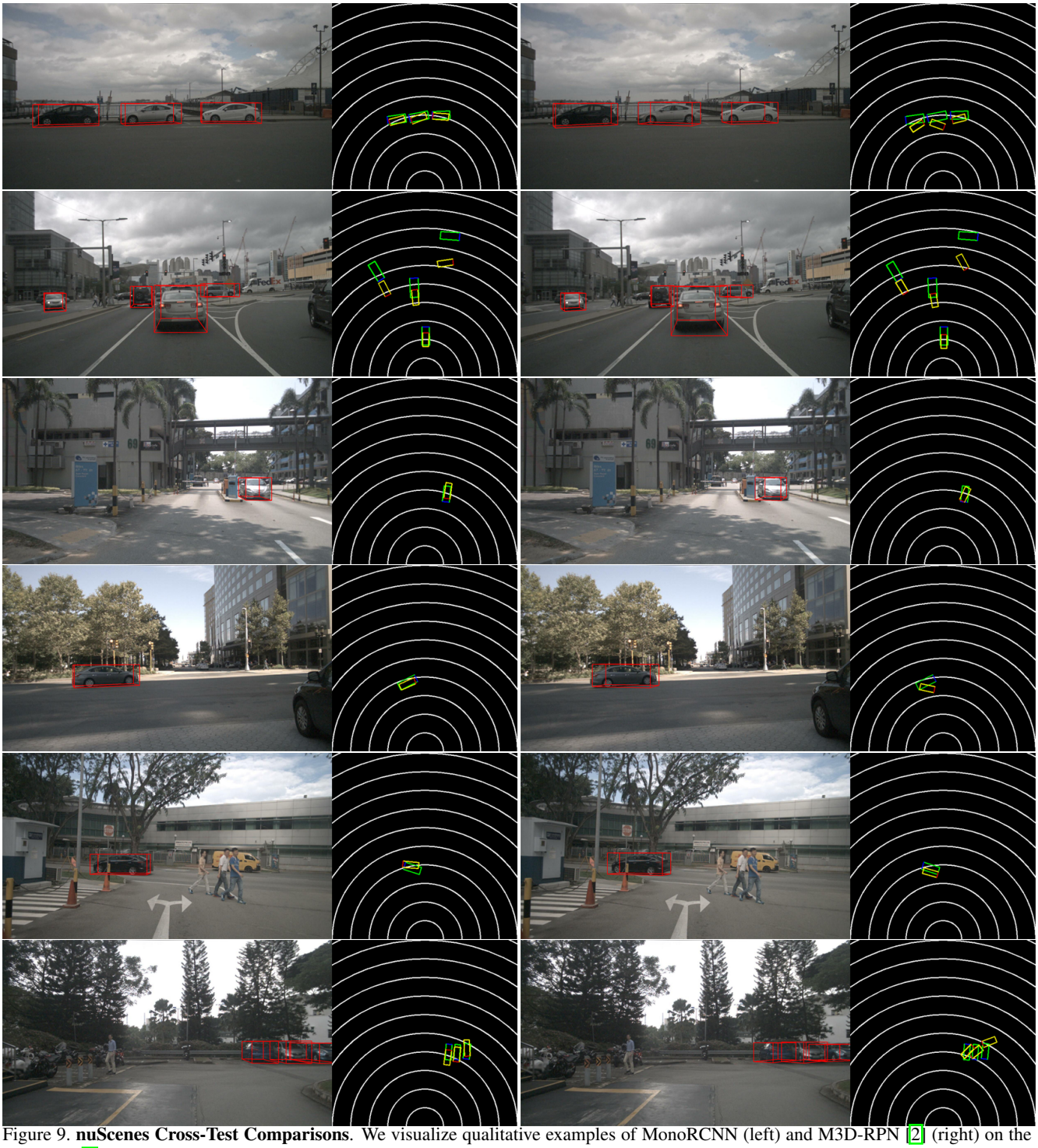


Figure 9. **nuScenes Cross-Test Comparisons**. We visualize qualitative examples of MonoRCNN (left) and M3D-RPN [2] (right) on the nuScenes [4] cross-test set. We can see our method achieves more accurate distance prediction. The 2D projections and bird's eye view results are shown as in Fig. 8. All models are only trained with the training subset of the KITTI val split [6].