Neurocomputing 562 (2023) 126894

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

GPro3D: Deriving 3D BBox from ground plane in monocular 3D object detection

Fan Yang [a,b,1,2], Xinhao Xu [a,b,1,2], Hui Chen [b], Yuchen Guo [b], Yuwei He [a,b], Kai Ni [c], Guiguang Ding [a,b,*]

[a] School of Software, Tsinghua University, Beijing, China
[b] BNRist, Tsinghua University, Beijing, China
[c] HoloMatic Technology, Beijing, China

ARTICLE INFO

Keywords:
Computer vision
Object detection
Monocular images
Autonomous driving

ABSTRACT

Considering the inherent ill-posed nature, monocular 3D object detection (M3OD) is extremely challenging. The ground plane prior is a highly informative geometry clue in M3OD. However, it has been neglected by most mainstream methods. This paper introduces an original M3OD framework that leverages the ground plane to directly derive the object's 3D Bounding Box (BBox) and 3D attributes geometrically. We identify and tackle three key factors that limit the applicability of the ground plane: the projection point localization issue, the ground plane tilt issue, and the lack of ground plane annotation issue. For the projection point localization issue, we propose leveraging the car's explicit and salient wheel pixels, which are easier for the neural network to detect compared to the bottom vertices or the bottom center of the 3D BBox. To tackle the ground plane tilt problem, we propose a vertical-edge-enhanced horizon line detection algorithm to precisely deduce the ground plane equation. Moreover, using only M3OD labels, wheel pixel and horizon line pseudo-labels can be easily generated to train the network without extra data or annotation cost. Extensive experiments demonstrate the effectiveness and superiority of our framework over previous methods.

1. Introduction

3D object detection plays a crucial role in autonomous driving systems [1–3]. Monocular 3D object detection (M3OD) utilizes a single image captured by a single camera as an input to determine the 3D bounding box of objects, which is commonly employed in scenarios involving automated driving [4–9]. However, due to the lack of 3D spatial information, M3OD faces limitations that result in suboptimal detection accuracy. To address this issue, it is important to incorporate the projections and geometrical cues in the scenes [10–14].

In this study, we leverage the ground plane to extract valuable geometric information. Essentially, each pixel in the image corresponds to a ray in space. For any pixel, which is the image projection of a point on the ground, we can ascertain its spatial location through the intersection of the corresponding ray with the ground plane, as depicted in Fig. 1.

However, the spatial position obtained directly from the inverse projection is not accurate enough. We identify three issues in this procedure: (1) The projection point localization issue. Specifically, when performing pixel inverse projection, using the 3D BBox bottom vertices or the bottom center point as the projection pixels may be a natural choice. Nevertheless, as illustrated in Fig. 2, these points lack clear semantics in images and are often too rough to be learned precisely by neural networks. (2) The ground plane tilt problem. Simply employing a preset fixed ground plane to inverse projection is a straightforward approach. However, in autonomous driving scenes, the ground plane in the camera coordinate system (CCS) is not static and may rotate due to bumps in the ego car. It can change dramatically when the ego car or the road fluctuates. Consequently, using an inaccurate ground plane reference can result in large errors when inferring objects' spatial positions (Fig. 3). (3) The lack of ground plane annotation issue. In general, there are no annotations related to the ground plane in most M3OD datasets.

In our approach, we propose a novel **G**round plane **Pro**jection Mono**3D** detection framework, namely GPro3D. We tackle the above three issues in M3OD and fully leverage the ground plane to derive objects' 3D BBoxes and 3D attributes geometrically.

* Correspondence to: School of Software, Tsinghua University, Beijing, 100084, China.
E-mail addresses: yfthu@outlook.com (F. Yang), xxh22@mails.tsinghua.edu.cn (X. Xu), jichenhui2012@gmail.com (H. Chen), yuchen.w.guo@gmail.com (Y. Guo), heyuwei403@gmail.com (Y. He), nikai@holomatic.com (K. Ni), dinggg@tsinghua.edu.cn (G. Ding).

[1] Fan Yang and Xinhao Xu have equal contribution.
[2] Fan Yang and Xinhao Xu are also with Hangzhou Zhuoxi Institute of Brain and Intelligence.

https://doi.org/10.1016/j.neucom.2023.126894
Received 24 June 2023; Received in revised form 10 September 2023; Accepted 3 October 2023
Available online 11 October 2023

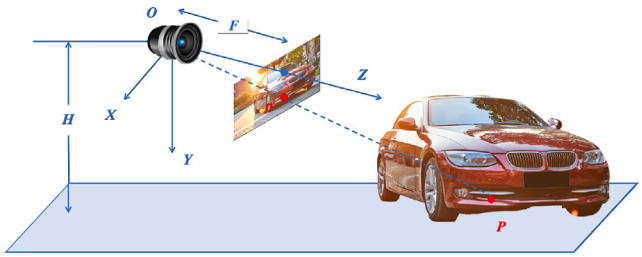

Fig. 1. Illustration of the inverse projection with the ground plane. H is the camera's height, and F is the camera's focal length. p is a pixel in the image, and P is the spatial position after the inverse projection. The blue plane in the figure represents the ground plane.

Fig. 2. Comparison between the 3D BBox's bottom vertices (orange dots) and the wheel pixels (blue dots). Obviously, the wheel pixels have stronger semantic information than the bottom vertices.

Fig. 3. The ground plane tilt problem in inverse projection. The blue plane is the preset fixed ground plane, neglecting the ground plane's tilt. From the bird's eye view (BEV), the inferred 3D BBox (blue box) using the fixed ground plane will gradually drift from the ground truth (red box).

Firstly, for the projection point localization issue, compared to the implicit 3D BBox bottom vertices or center, we observe that the wheel pixels of vehicles are explicit and salient pixels in images (Fig. 2). They are rich in semantics, easy for neural networks to detect, and could be inversely projected directly with the ground plane. In addition, for pedestrians and other objects that do not have wheels, we choose the pixels connected to the ground as projection points, like feet pixels for pedestrians.

Secondly, to address the problem of ground plane tilt, we propose a vertical-edge-enhanced horizon line detection method to deduce the precise ground plane equation. Note that directly inferring the ground plane equation from the 2D image is considerably difficult for the neural network. We find that, in principle, to capture such a dynamic ground plane, we can leverage a point, i.e., the camera's optical center, and a line, i.e., the horizon line,[3] to deduce such a plane mathematically. Intuitively, compared to the ground plane, the horizon line is easier for the neural network to deduce. However, in practical situations, another problem arises: the horizon is usually occluded (Fig. 4 c). In this case, the neural network may not be able to detect the horizon line effectively. To address this problem, we further design a novel vertical edge mining algorithm that enhances the horizon line detection by leveraging salient vertical edges in the image (such as building edges, cement columns, lamp posts, etc.) (Fig. 4). With the improved horizon line obtained through this algorithm, we can derive an accurate and dynamic mathematical equation for the ground plane, which, in turn, mitigates the deviation arising from ground plane tilt problems.

Thirdly, most M3OD datasets have no wheel pixels and ground plane annotations. AVOD [15] uses extra sensor data (including IMU) to generate plane labels in KITTI datasets, which is not feasible in other M3OD datasets and practical situations. And the authors of AVOD [15] also acknowledge that it is a time-consuming procedure, only applicable for beating on the KITTI benchmark. In contrast, we propose a pseudo-label generation method for ground and wheel pixels, which is applicable to various M3OD datasets and real-world scenarios. The pseudo-labels work adequately for training wheel pixels and horizon line using only M3OD labels, with no other labeling costs incurred.

We notice that MonoGround [16] and MoGDE [17] also use the ground. But they only use a fixed ground [16] or use external sensor data to roughly infer camera pose [17]. And the ground serves as additional auxiliary information for the neural network to infer depth. Objects' 3D attributes like 3D size, orientation, and depth are inferred from neural networks. They do not fully leverage the geometric projection in the scene. In contrast, we leverage a more accurate ground plane equation and wheel pixels to directly deduce objects' 3D BBoxes and attributes geometrically, which makes more thorough use of the ground plane equation with better interpretability and controllability. Furthermore, with our vertical-edge-enhanced horizon detection algorithm, the ground plane equation could be deduced precisely and robustly. At the same time, pseudo-labels for wheel pixels and ground equation could be generated without needing any other data or annotation beyond the M3OD dataset.

In summary, our main contributions are as follows.

(1) We introduce a novel monocular 3D object detection (M3OD) framework that leverages the ground plane and inverse projection to derive objects' 3D BBoxes geometrically accurately. In our introduced M3OD framework, various 3D attributes of the objects, including spatial location, 3D size, and orientation could be determined geometrically. Our framework resolves three key issues: the projection point localization issue, the ground plane tilt issue, and the lack of ground plane annotation issue.

(2) For the projection point localization issue, we propose leveraging the wheel pixels of the objects, which are explicit and salient pixels in images. Such points are more suitable for neural networks to localize precisely.

(3) To address the ground plane tilt issue, we propose determining the ground plane equation based on the vertical-edge-enhanced horizon detection algorithm. The innovative algorithm extracts vertical edges in images to enhance the horizon line detection, which could solve the occlusion problem of the horizon line and accomplish a robust ground plane equation deduction.

(4) In order to train the wheel pixel and the horizon detection, we propose a simple but very effective pseudo-label generation scheme, which only uses M3OD labels and does not need extra annotation.

(5) Extensive experiments demonstrate the effectiveness of our framework and emphasize the importance of the ground plane, which could be a promising and potential geometry clue for M3OD. Our method outperforms previous methods without using extra data, demonstrating the extraordinary capacity of our framework.

2. Related work

2.1. Keypoints in monocular 3D object detection

Monocular 3D object detection aims to infer the 3D BBoxes of the objects from an image [18–21]. To obtain the accurate 3D bounding box, keypoints have been employed as auxiliary information in M3OD, which can also assist in inferring the shape of occluded and truncated objects. Points can effectively enhance feature extraction [7,22–24].

[3] The horizon is defined as the line of intersection of the projection plane and the plane parallel to the ground plane. And it refers to the perceived line where the sky meets the ground.

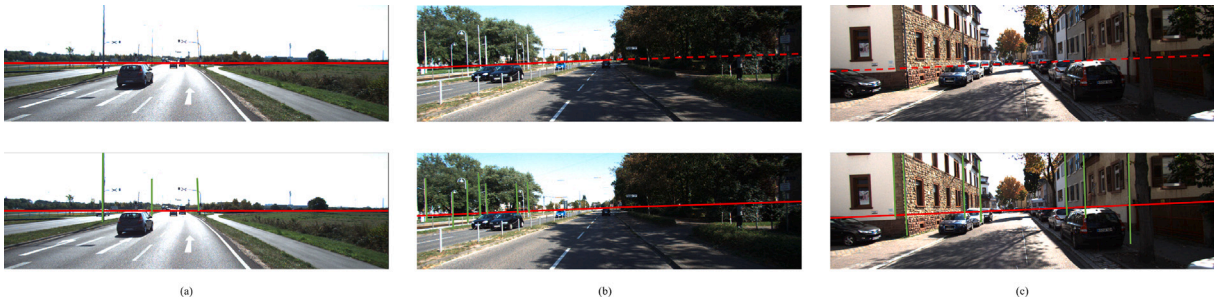

Fig. 4. Horizon line detection. Our vertical edge detection (in green) can be applied as an assistant to the horizon line detection.

F. Chabot et al. [25] use a 3D CAD model with a fixed number of keypoints. They predict the CAD template similarity and match the model with the highest similarity to reconstruct the object's shape and orientation. CAD models have shown an effect on shape detection. But other 3D attributes' results, especially the depth prediction, are still unsatisfactory.

Some works use the keypoints to conduct projection. For example, RTM3D [22] uses a feature pyramid network (FPN) as the feature extraction backbone [26], and a multi-task [27] detection head to predict different attributes of objects. RTM3D uses nonlinear least square optimization to minimize the distance between the 3D position projection and the 2D keypoints predicted by the neural network. KM3D [7] improves RTM3D by embedding the 2D-3D geometric constraints into the training process of the neural network, and the matrix calculus is used for gradient back propagation to minimize the re-projection loss. MonoFlex [28] uses the keypoints to assist the depth estimation process and adopts an uncertainty-guided ensemble method to improve its accuracy.

2.2. Geometry projection in monocular 3D object detection

Recently, many works have attempted to involve geometric priors in M3OD, achieving encouraging results [29,30]. Deep learning can handle spatial information very effectively [31,32]. Deep3DBox [33] uses neural networks to predict an object's orientation, dimension, and 2D bounding box to provide constraints for the 3D bounding box. Some works [16,34] use the ground plane to assist M3OD, but they overlook the ground plane tilt problem and use the rough 3D BBox bottom center to conduct projection. In addition, MonoRCNN [35] introduces the geometric information between 2D BBox height and 3D height to estimate the objects' depth. GUPNet [11] also estimates objects' depth with objects' 2D and 3D height projection and uses uncertainty loss to determine objects' scores more precisely. MonoEF [36] focuses on the changes of camera external parameters and uses the extra data, i.e., the odometry datasets to train the network. It designs a feature transfer network to rectify the feature disorder. Unlike these implicit feature transfer networks, our method finds a way to calculate the ground plane equation directly.

3. Approach

3.1. Overview

On the whole, our procedure is shown in Fig. 5. For the projection point localization issue, we propose the wheel pixels guided 2D object localization method (Section 3.2). In order to solve the ground plane tilt problem, we design a novel vertical-edge-enhanced horizon line detection algorithm and propose estimating the exact ground plane equation mathematically based on the horizon line (Section 3.3). Furthermore, we design a 3D attribute deduction method based on dynamic inverse projection, which can achieve accurate 2D-3D inference and deduce objects' 3D BBoxes and 3D attributes geometrically (Section 3.4). Additionally, we invent the wheel pixel and horizon line pseudo-label generation method, which enables our framework to operate without the need for any extra data (Section 3.2.2 and Section 3.3.5).

Note that the horizon refers to the horizon line in the image coordinate by default, and the ground plane refers to the ground plane in 3D coordinates by default. The image coordinate system, the camera coordinate system (CCS), and their axis' direction are shown in Fig. 6.

3.2. 2D object localization guided by wheel pixels

3.2.1. Wheel pixel detection

We begin with the localization of objects in the 2D image. Rather than the 3D BBox bottom vertices or bottom center used by most previous works [7,22,33], we propose using the wheel pixels of the vehicles as the projection points, because they have clear semantics in the image (Fig. 2) and are much easier to be detected by neural networks. For pedestrians and other objects, we detect the pixels that connected to the ground. The wheel pixel detection is comprised of three detection heads [37]. The first head's output $\hat{Y}$ is the heatmap of all objects' wheel pixels in the image. The second head outputs the local offset $\hat{Y}_{of}$ to remedy the downsampling error, and the third head's output $\hat{Y}_c$ represents the indication from the object center. Defining Y, Y_{of}, Y_c as their ground truths, the keypoints loss is as follows:

$$\begin{aligned} L_{kps} &= L_{kps,hm} + L_{kps,of} + L_{kps,C} \\ &= \text{Focal}(\hat{Y}, Y) + \text{L}_1(\hat{Y}_{of}, Y_{of}) + \text{L}_1(\hat{Y}_c, Y_c) \end{aligned} \tag{1}$$

where $L_{kps,hm}$ is the focal loss [38] of the keypoint heatmaps, $L_{kps,of}$ is the L_1 loss of the points' local offset and $L_{kps,C}$ is the L_1 loss of the indication from the object center.

The focal loss [38] is as follows:

$$\begin{aligned} &\text{Focal}(\hat{T}, T) \\ &= \frac{-1}{UVC} \sum_{uvc} \begin{cases} (1-\hat{T}_{uvc})^{\alpha} \log(\hat{T}_{uvc}) & \text{if } T_{uvc} = 1 \\ (1-T_{uvc})^{\beta} \hat{T}_{uvc}^{\alpha} \log(1-\hat{T}_{uvc}) & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

where $\hat{T}$ is the predicted tensor and T is the ground truth. U, V, C are the tensor's size and u, v, c are the indexes. We set focal loss parameters $\alpha = 2, \beta = 4$. The L_1 loss is as follows:

$$\text{L}_1(\hat{T}, T) = \text{mean}(\left|\hat{T} - T\right|) \tag{3}$$

where $\hat{T}$ is the predicted value and T is the ground truth.

Benefiting from the well-designed keypoint detection heads, our method can detect the wheel pixels accurately even when they are occluded. We use $(u_{CP}, v_{CP})^{\top}$ to denote one of the detected wheel pixels in the image, and we will inversely project $(u_{CP}, v_{CP})^{\top}$ to the 3D space in Section 3.4.2.

3.2.2. Wheel pixel pseudo-label generation

One non-negligible issue in wheel pixel detection is that the wheel pixel annotation is inaccessible in M3OD datasets. Thus, we design a wheel pixel pseudo-label generation scheme to train our network. Specifically, we first localize the wheel point 3D coordinates guided by the object's 3D BBox in the CCS. We determine the spatial position

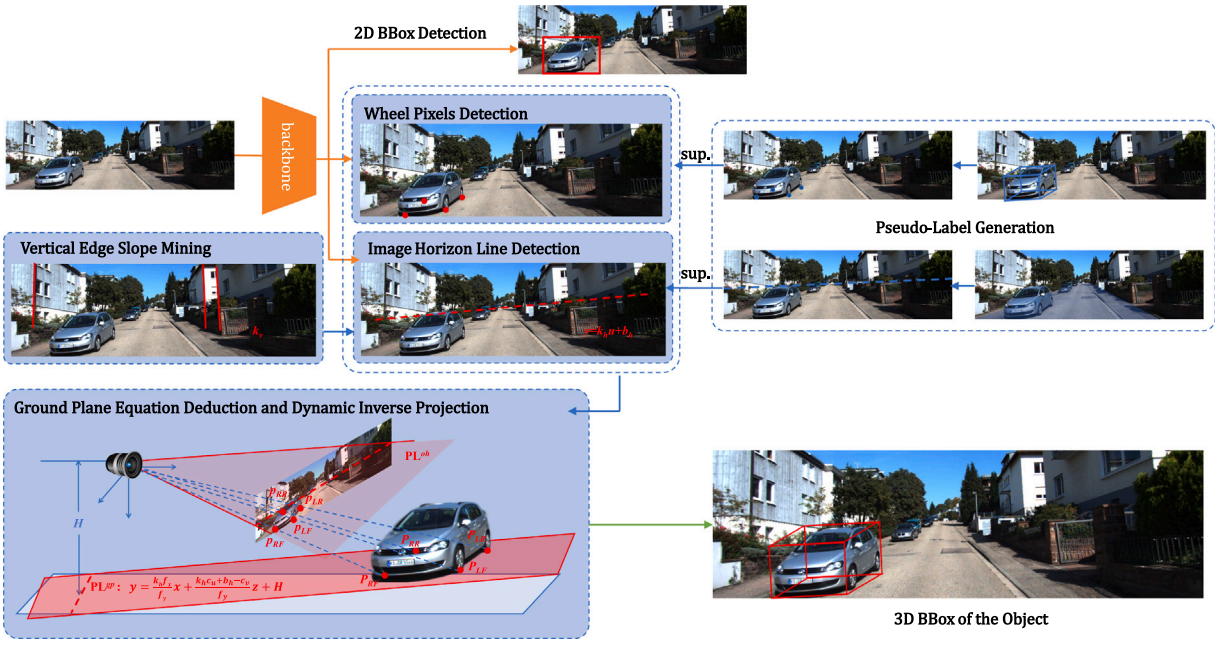

Fig. 5. The Dynamic Ground Plane Projection (GPro3D) Framework. Firstly, we predict the 2D BBox, the wheel pixels and the horizon line from the input image. Next, with our proposed vertical edge and horizon line detection, the ground plane equation PL^{gp} (Eq. (19)) is derived mathematically. Then with the wheel pixels and the ground plane equation, dynamic inverse projection is conducted to infer a geometry-based 3D BBox. Moreover, we design the pseudo-label generation scheme using only M3OD labels to train the wheel pixel and the horizon line detection head without any extra data.

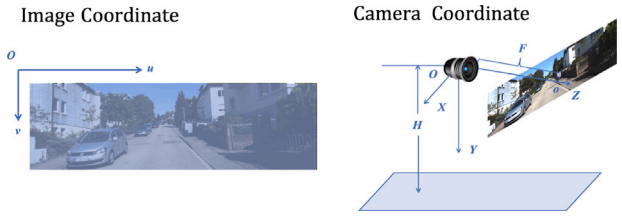

Fig. 6. The image coordinate and the camera coordinate. The left subfigure is the image coordinate: the origin point is in the upper left corner of the image with the u-axis right and v-axis down. The right subfigure is the camera coordinate system (CCS): the origin point is the camera's optical center with the X-axis right, Y-axis down, and Z-axis forward along the camera's optical axis.

of 3D BBox's bottom face and then utilize the wheelbase information to localize the wheel points. We define k_l as the ratio of the front and rear wheels' distance to the length of the 3D BBox and k_w as the ratio of the left and right wheels' distance to the width of the 3D BBox, as shown in Fig. 7. We use prior knowledge of vehicle expertise to determine k_l and k_w. The k_l and k_w may slightly differ from the ground truth, but our experimental results have shown that such pseudo-labels significantly benefit the 3D BBox derivation.

We define $P^o_{LF}, P^o_{RF}, P^o_{RR}, P^o_{LR}$ (The superscript "o" represents the object's local 3D coordinate system. LF, RF, RR, and LR represent the left-front, right-front, right-rear and left-rear point, respectively) as the wheel points in the object's local 3D coordinate system[4]:

$$\begin{aligned} P^o_{LF} &= [\frac{k_l}{2}l, 0, \frac{k_w}{2}w]^\top,\ P^o_{RR} = [-\frac{k_l}{2}l, 0, -\frac{k_w}{2}w]^\top \\ P^o_{LR} &= [-\frac{k_l}{2}l, 0, \frac{k_w}{2}w]^\top,\ P^o_{RF} = [\frac{k_l}{2}l, 0, -\frac{k_w}{2}w]^\top \end{aligned} \tag{4}$$

[4] We define that object local coordinate's axis direction is the same as the camera coordinate, i.e., X-axis right, Y-axis down, Z-axis forward along the optical axis of the camera. The origin of the object's local coordinate is the center of the bottom face, and the front of the object faces the positive direction of the X-axis.

where l and w are the length and width of the object, respectively.

Given one coordinate P^o in the object's local coordinate system, the position P^c in the CCS can be obtained by using the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{T}$:

$$\begin{aligned} P^c &= \mathbf{R}P^o + \mathbf{T} \\ \mathbf{R} &= \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix},\quad \mathbf{T} = [x, y, z]^T \end{aligned} \tag{5}$$

where x, y, z are the position of the object's bottom center in the camera coordinate and θ is the corresponding rotation angle around Y-axis. With the camera intrinsic matrix $\mathbf{K}$, we can derive the wheel pixel pseudo-label $p = [u, v]^\top$:

$$\begin{aligned} z_p[u, v, 1]^\top &= \mathbf{K}P^c = \mathbf{K}(\mathbf{R}P^o + \mathbf{T}) \\ \mathbf{K} &= \begin{bmatrix} f_x & 0 & c_u \\ 0 & f_y & c_v \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{6}$$

where z_p is the Z-axis coordinate of P^c in the CCS and f_x, f_y, c_u, c_v are camera intrinsic parameters. When P^o's value is set to $P^o_{LF}, P^o_{RF}, P^o_{RR}$ or P^o_{LR}, the corresponding wheel pixel pseudo-labels in image p_{LF}, p_{RF}, p_{RR} or p_{LR} can be derived.

3.2.3. 2D bounding box detection

In the 2D BBox detection part, we predict the object's center heatmap $\hat{B}$, the center offset $\hat{B}_{of}$, and the 2D BBox size $\hat{B}_s$ [37,39]. We use focal loss [38] for the center heatmap, L_1 loss for center offset and box size. The 2D BBox detection loss L_{2d} is as follows:

$$\begin{aligned} L_{2d} &= L_{center} + L_{2d\_of} + L_{2d\_size} \\ &= \mathrm{Focal}(\hat{B}, B) + \mathrm{L}_1(\hat{B}_{of}, B) + \mathrm{L}_1(\hat{B}_s, B) \end{aligned} \tag{7}$$

where B, B_{of}, B_s are the center heatmap, center offset and 2D BBox size ground truths, respectively.

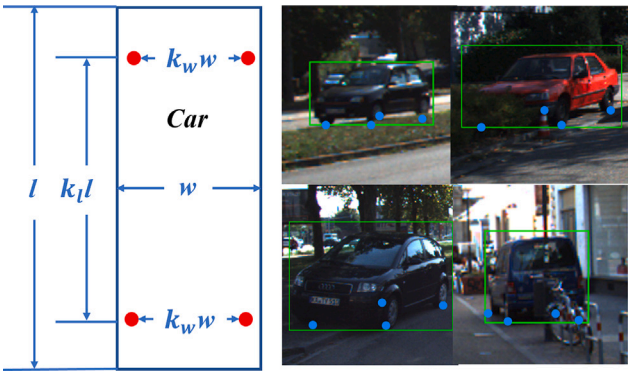

Fig. 7. The wheel pixels pseudo-label generation. We design the pseudo-label generation scheme using 3D BBox labels. Specifically, the position of the wheel pixels satisfies a proportional relationship with the size of the 3D BBox. The left figure shows the relationship in BEV, and the right one is the visualization of pseudo-labels.

3.3. Ground plane estimation based on the horizon line

3.3.1. Analysis

For the second issue hindering the use of the ground plane, i.e., the ground plane tilt issue, we estimate a mathematical formulation of the ground plane to solve the problem. Note that directly estimating such tilt from the image is too obscure for neural networks, and we propose a vertical-edge-enhanced horizon detection algorithm to deduce the ground plane equation. As shown in Fig. 4, the horizon is the image projection of the ground plane at infinity. Such horizon information can reflect the ego pose of the camera relative to the ground. When the pitch angle of the camera changes, the horizon moves up and down in the image. When the roll angle of the camera changes, the horizon rotates in the image. Therefore, we aim to leverage such horizon information to mathematically derive an accurate ground plane equation.

Note that naive image horizon detection does not work in an environment where the horizon is unclear. As shown in Fig. 4(c), the horizon line in the image is severely occluded owing to the obstacles in the scene, which makes the horizon implicit and difficult to detect. In order to estimate the ground plane in the image with many obstacles, we draw inspiration from human recognition patterns. The pattern is that humans usually utilize large objects in the scene, like buildings, cement columns, and lamp poles as guidance to infer the ego posture and the tilt of the ground plane. We imitate such human capacity and go deep into the details. Intuitively, the salient edges in images, especially vertical edges, can reflect the rotation of the scenes. Specifically, the man-made buildings tend to be upright, so most vertical edges of these buildings are orthogonal to the horizon line. Therefore, we design an ingenious unsupervised digital image processing algorithm to extract salient vertical edges in images and dig out the valuable edges. Then we deduce the slope of the vertical edges and the horizon line. Finally, we can complete the horizon line detection and ground plane estimation whether it is occluded or not.

Table 1 shows the meaning of symbols in the derivation of the following formula.

3.3.2. Image horizon line detection

To accomplish the task of 2D horizon line detection, we introduce a horizon line heatmap head into the neural network. This heatmap represents the probability, ranging from 0 to 1, of a pixel belonging to the horizon line. During training, we adopt the focal loss [38] for each pixel.

$$L_{hor} = \text{Focal}(\widehat{M}, M) \tag{8}$$

Table 1
The meanings of some symbols.

Symbol	Meaning
CCS	camera coordinate system
HL_{nn}^{2d}	2D horizon line predicted by neural network
HL_{v}^{2d}	vertical-edge enhanced 2D horizon line
k_h, b_h	slope and intercept of the vertical-edge enhanced 2D horizon line
u, v	2D pixel coordinates in images
x^c, y^c, z^c	3D point coordinates in camera coordinate system
f_x, f_y	focal length of the camera
c_u, c_v	principal point of the camera
H	the camera's height to the ground

where $\widehat{M}$ is the predicted horizon heatmap and M is the ground truth. The focal loss is defined as follows:

$$\text{Focal}(\widehat{M}, M) = \frac{-1}{UV} \sum_{uv} \begin{cases} (1-\widehat{M}_{uv})^{\alpha} \log(\widehat{M}_{uv}) & \text{if } M_{uv} = 1 \\ (1-M_{uv})^{\beta} \widehat{M}_{uv}^{\alpha} \log(1-\widehat{M}_{uv}) & \text{otherwise} \end{cases} \tag{9}$$

where U, V denote the size of the horizon heatmap, and u, v indicates the coordinates of individual pixels within the heatmap. We set the focal loss parameters as $\alpha = 2, \beta = 4$.

In the post-processing stage, we extract the points with the maximum activation value from each column and utilize the Least Squares Method to fit a horizon line. If we can obtain the slope of the horizon line, denoted as k_h, from the vertical edge detection in Section 3.3.3 (i.e., when the output of Algorithm 1 is not None), we only need to calculate the bias b_h of the horizon line. Otherwise, both the slope k_h and the bias b_h of the horizon line need to be calculated using the Least Squares Method.

The detected 2D horizon line generated by the neural network can be represented as follows:

$$\text{HL}_{nn}^{2d}: \quad v = k_h^{nn} u + b_h^{nn} \tag{10}$$

where u and v denote the pixel coordinates in the image, and k_h^{nn} and b_h^{nn} represent the slope and intercept of the horizon line detected by the neural network, respectively.

3.3.3. Image vertical edge detection

In real scenes, the horizon line may be occluded by buildings or other obstacles, which makes it very hard for neural networks to detect. Though the vertical edges could greatly assist the horizon line detection and ground plane estimation, there are no annotations of vertical edges in M3OD datasets. Considering that the vertical edges are low-level features, we adopt an unsupervised vertical edge slope mining algorithm based on the traditional digital image processing technique to implement vertical line detection.

Our vertical edge slope mining algorithm can be summarized in Algorithm 1. The input is the original image and the output is the slope of the largest cluster k_v or None. Firstly, we conduct a GaussianBlur algorithm on the input image, which can exclude tiny textures while not affecting the large color patches. Secondly, a Canny algorithm extracts edge points from blurred images. Next, a probabilistic Hough transform algorithm is utilized to detect lines. The GaussianBlur, Canny, and HoughLinesP functions are implemented with OpenCV [40,41]. Afterward, we filter the vertical edges with the inclination angle between 70° and 110°. Then we employ the Birch [42] algorithm to angles of all vertical edges and find the largest cluster. In this process, noise vertical edges in the image can be eliminated, and the vertical edge slope, denoted as k_v, can be derived from the centroid slope of the largest cluster. Thr_N and Thr_S are the threshold that can be set manually. If the number of the vertical edges $N_V > Thr_N$ and the standard deviation of the slopes $S_V < Thr_S$, we can trust the slope result of the vertical edges. Then the horizon line is perpendicular to these vertical lines and

Algorithm 1 Vertical Edge Slope Mining

Input: an RGB image, denoted as img
Output: the slope of the largest cluster, denoted as k_v, or None

```
BlurImg = GaussianBlur(img, ksize = (13, 13), σ_X = 4, σ_Y = 4)
CannyEdges = Canny(BlurImg, threshold1 = 50, threshold2 = 100, apertureSize=3)
HoughEdges = HoughLinesP(CannyEdges, ρ = 1, θ = π / 180, threshold = 5, minLineLength=40, maxLineGap=10)
VerticalEdgesSlopes = FilterVertical(HoughEdges)
N_V = VerticalEdgesSlopes.number
S_V = VerticalEdgesSlopes.StandardDeviation
LargestCluster = Birch(VerticalEdgesSlopes)
k_v = LargestCluster.slope
if N_V > Thr_N and S_V < Thr_S then
    return k_v
else
    return None
end if
```

its slope: $k_h = -\frac{1}{k_v}$. If the vertical edges are too few or the standard deviation of their slope is too large, the output of the Algorithm 1 would be nothing. In this case, we give up the vertical edge detection and predict the horizon directly. In summary, the vertical-edge-enhanced image horizon line HL_v^{2d} is as follows[5]:

$$k_h = \begin{cases} -\frac{1}{k_v} & \text{if } N_V > Thr_N \text{ and } S_V < Thr_S \\ k_h^{nn} & \text{otherwise} \end{cases}, \tag{11}$$

$$b_h = b_h^{nn} \tag{12}$$

$$\mathrm{HL}_v^{2d}: \quad v = k_h u + b_h \tag{13}$$

where u and v are pixel coordinates in images. k_h^{nn} and b_h^{nn} are the slope and intercept of the horizon line detected by the neural network, from Eq. (10). Note that the slope k_h is derived from the vertical edges in the case $N_V > Thr_N$ and $S_V < Thr_S$ and the intercept $b_h = b_h^{nn}$ comes from the horizon line detection network.

3.3.4. Ground plane estimation

We aim to acquire the ground plane equation in the camera coordinate from the horizon line in the image. Given the camera intrinsic matrix $\mathbf{K}$, we get the relationship between pixel coordinates $(u, v)^\top$ and the camera coordinates $(x^c, y^c, z^c)^\top$ as follows:

$$z_c[u, v, 1]^\top = \mathbf{K}[x^c, y^c, z^c]^\top$$
$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_u \\ 0 & f_y & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

From Eq. (14), we can obtain:

$$u = \frac{f_x}{z^c} x^c + c_u, \quad v = \frac{f_y}{z^c} y^c + c_v \tag{15}$$

Considering a pixel $(u, v)^\top$ in the horizon line HL_v^{2d}, we insert Eq. (15) into Eq. (13):

$$\frac{f_y}{z^c} y^c + c_v = k_h(\frac{f_x}{z^c} x^c + c_u) + b_h \tag{16}$$

In the image, the horizon line's depth value z^c in the CCS is equal to the focal length f of the camera. Therefore, with Eq. (16) and $z^c = f$, the horizon line equation in the CCS HL^{3d} can be derived as:

$$\mathrm{HL}^{3d}: \quad y^c = \frac{k_h f_x}{f_y} x^c + \frac{k_h c_u + b_h - c_v}{f_y} z^c, \quad z^c = f \tag{17}$$

which represents the image projection of the ground plane when the depth approaches infinity. Based on HL^{3d}, we discover an ingenious corollary: the plane (denoted as PL^{oh}) passing through the horizon line HL^{3d} and the camera's optical center is parallel to the ground plane (denoted as PL^{gp}). In the CCS, the intercept between two planes along the Y-axis is the camera's height to the ground, denoted as H ($H = 1.65m$ in KITTI [43]).

Considering that a plane can be uniquely determined by a line and a point in 3D space, we can derive the equation of plane PL^{oh} from the horizon line HL^{3d} (Eq. (17)) and the camera's optical center (the origin point of the CCS) as followings:

$$\mathrm{PL}^{oh}: \quad y = \frac{k_h f_x}{f_y} x + \frac{k_h c_u + b_h - c_v}{f_y} z \tag{18}$$

By translating the plane PL^{oh} along the Y-axis for H, the real-world ground plane PL^{gp} can be derived as:

$$\mathrm{PL}^{gp}: \quad y = \frac{k_h f_x}{f_y} x + \frac{k_h c_u + b_h - c_v}{f_y} z + H \tag{19}$$

Additionally, the roll angle θ_{roll} and pitch angle θ_{pitch} of the ground plane in the CCS can be derived as:

$$\theta_{roll} = \arctan(\frac{k_h f_x}{f_y})$$
$$\theta_{pitch} = \arctan(\frac{k_h c_u + b_h - c_v}{f_y}) \tag{20}$$

The previous works [16,34] preset a fixed ground plane equation. In contrast, we propose estimating a dynamic ground plane. Our main focus is on the ground plane tilt issue, and the experimental results prove that correcting the tilt issue is critical for inverse projection. In theory, solving more problems, for example, modeling the ground plane as a curved surface, could lead to further improvements and could become a promising future research direction.

3.3.5. Horizon line pseudo-label generation

The horizon line ground truth is indispensable for training the horizon line detection network. However, the wide-used dataset for M3OD, e.g., KITTI [43] and nuSences [44], does not have the horizon line label annotations. Here, we use M3OD dataset annotations to generate pseudo-labels for horizon line detection, which is cost-efficient and easy to operate.

We derive the ground plane pseudo-label from the object's 3D bounding box ground truth. Specifically, we use the bottom centers of the objects' 3D bounding boxes to fit a plane, which can be regarded as the ground plane in the scene. The horizon line pseudo-label in the image can be obtained by projecting the infinite position of the ground plane onto the image, which is the inverse process of Ground Plane Estimation, seeing Section 3.3.4.

Vertical edge and horizon line detection techniques are utilized during both the training and inference stages. The horizon line pseudo-label generation method is employed specifically during the training stage, which aims to provide supervision for the neural networks.

3.4. 3D bounding box deduction based on dynamic inverse projection

3.4.1. Analysis

Each pixel in the image can be mapped to a ray in space. If we know a point along the ray is on the specific ground, we can determine its spatial position. Therefore, we design the following dynamic inverse projection algorithm with our estimated precise wheel pixels in Section 3.2 and dynamic ground plane equation in Section 3.3, then we derive objects' 3D BBoxes and 3D attributes geometrically.

[5] The vertical edges have been filtered, and their inclination angles are between 70° and 110°, so k_v cannot be 0, and k_h is assigned 0 if k_v is infinity.

3.4.2. Dynamic inverse projection of wheel pixels with ground plane equation

Given the detected wheel pixel $(u_{CP}, v_{CP})^\top$ in the image (from Section 3.2.1) and the camera intrinsic parameter matrix, we can inversely project the wheel pixel to the spatial position in the CCS:

$$z^c_{CP}[u_{CP}, v_{CP}, 1]^\top = \begin{bmatrix} f_x & 0 & c_u \\ 0 & f_y & c_v \\ 0 & 0 & 1 \end{bmatrix} [x^c_{CP}, y^c_{CP}, z^c_{CP}]^\top \tag{21}$$

Since the wheel point $(x^c_{CP}, y^c_{CP}, z^c_{CP})^\top$ in the CCS is located on the ground plane, we can conveniently re-formulate Eq. (19) as follows:

$$y^c_{CP} = \frac{k_h f_x}{f_y} x^c_{CP} + \frac{k_h c_u + b_h - c_v}{f_y} z^c_{CP} + H \tag{22}$$

By solving Eqs. (21) and (22), the wheel point coordinate $(x^c_{CP}, y^c_{CP}, z^c_{CP})^\top$ in the CCS can be derived using the image coordinate $(u_{CP}, v_{CP})^\top$, the horizon line parameters k_h, b_h, the camera's height H and the camera's intrinsic parameters as follows:

$$\begin{aligned} x^c_{CP} &= \frac{u_{CP} - c_u}{\lambda} \cdot \frac{f_y}{f_x} \\ y^c_{CP} &= \frac{v_{CP} - c_v}{\lambda} \\ z^c_{CP} &= \frac{f_y}{\lambda} \end{aligned} \tag{23}$$

where $\lambda = (v_{CP} - k_h u_{CP} - b_h)/H$.

3.4.3. 3D attribute derivation

In this section, we will introduce the process of deriving all 3D attributes of an object, i.e., depth, dimension, and rotation. Taking the car category as an example, a car has four wheel pixels with spatial coordinates P_{LF}, P_{RF}, P_{RR}, and P_{LR}, derived by Eq. (23) (LF, RF, RR and LR represent the left-front, right-front, right-rear and left-rear point, respectively.). Then the estimated bottom center coordinate can be calculated as:

$$P_{BC} = \frac{1}{4}(P_{LF} + P_{RF} + P_{LR} + P_{RR}) \tag{24}$$

The depth of the car d_g from the ground plane prior can be derived as:

$$d_g = z_{BC} = \frac{1}{4}(z_{LF} + z_{RF} + z_{LR} + z_{RR}) \tag{25}$$

where $z_{BC}, z_{LF}, z_{RF}, z_{LR}, z_{RR}$ are the Z-axis coordinates of $P_{BC}, P_{LF}, P_{RF}, P_{LR}, P_{RR}$ in the CCS, respectively.

In the process of wheel pixel pseudo-label generation (Section 3.2.2), we have defined k_l as the ratio of the front and rear wheels' distance to the 3D BBox's length and k_w as the ratio of the left and right wheels' distance to the 3D BBox's width. Then the length l_{3D} and the width w_{3D} of the 3D BBox can be derived from the wheels' distance:

$$\begin{aligned} l_{3D} &= \frac{\|(P_{LF} + P_{RF}) - (P_{LR} + P_{RR})\|_2}{2k_l} \\ w_{3D} &= \frac{\|(P_{RF} + P_{RR}) - (P_{LF} + P_{LR})\|_2}{2k_w} \end{aligned} \tag{26}$$

The height of the 3D BBox is $h_{3D} = d_g \cdot h_{2D}/f_y$, where f_y is one of camera intrinsic parameters, d_g is the depth of the object from Eq. (25) and h_{2D} is the predicted object's 2D BBox height by neural network.[6]

The rotation angle of the object in the BEV (X-Z plane) from the X-axis is as follows:

$$r_o = \arctan\left(\frac{z_{LF} + z_{RF} - 2z_{BC}}{x_{LF} + x_{RF} - 2x_{BC}}\right), \quad r_o \in [-\frac{\pi}{2}, \frac{\pi}{2}] \tag{27}$$

[6] For other categories which have less than four wheel pixels , we use the average width and length to estimate the geometry based w_{3D} and l_{3D}. Apart from that, the process of acquiring 3D attributes is similar to that of the Car category.

The final rotation result r_g is r_0 after simple adjustments according to the car's orientation, i.e., $r_g = r_0$, $r_g = r_0 + \pi$ or $r_g = r_0 - \pi$, where $r_0 \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $r_g \in [-\pi, \pi]$.

From the above procedure, we can derive all required 3D attributes for M3OD, i.e., depth, dimension (l_{3D}, w_{3D} and h_{3D}) and rotation. Note that in most M3OD works, the 3D attributes of the object are usually predicted by neural networks. By contrast, we could determine the 3D BBox more accurately through the geometry derivation.

Additionally, we also design a 3D BBox refinement head. Following previous works [11,29], we use RoIAlign [45] to extract the single object's RoI feature. Then we use several layers of convolution to obtain 3D bias. The refined 3D attribute could be obtained by adding the geometry estimation and the 3D bias.

4. Experiments

4.1. Setup and implementation details

We conduct experiments on the KITTI [43] and nuScenes [44] datasets to verify the effectiveness and generalization of the proposed method. The KITTI 3D dataset is the most popular benchmark in M3OD. Following [56,58], we split the training images into training and validation sets and use the standard AP_{40} and AP_{11} [43,46] to evaluate our method. Following KITTI's official metrics [43] and previous works, the IoU threshold for cars is set to 0.7, while the thresholds for pedestrians and cyclists are set to 0.5. In addition, we also conduct experiments on the nuScenes [44] to demonstrate the generalization capacity of our approach.

Our code is implemented with PyTorch 1.7.1 [59] and CUDA 11.0. The input images are resized to 1280 × 384. We utilize the popular DLA34 [60] as the backbone with a downsampling ratio of 4. We use two 3 × 3 convolution layers with batch normalization and ReLU for all detection heads. We train our network with the batch size 16 and Adam [61] optimizer for 200 epochs. The initial learning rate is 1.25×10^{-3}, and we use the cosine learning rate warmup schedule of 5 epochs and learning rate decay of 0.1 in the training process.

4.2. Main results

As indicated in Table 2, we use the KITTI official metrics AP_{40} [43] to evaluate GPro3D's performance [56]. Our method achieves the best performance in all twelve metrics of the 3D and BEV detection without any extra data. As we can see, our approach outperforms previous state-of-the-art methods in both 3D and BEV metrics. Moreover, Table 3 shows the AP_{11} results of 3D and BEV detection. Our method ranks top in various metrics, demonstrating that our approach does work and yields better results.

Table 4 is 3D and BEV results for the Car category at IoU 0.7 on the KITTI test set. Table 5 shows our results of all three categories on the KITTI test set with official metrics [11,28,43]. Our method has achieved excellent performance.

Note that some other works [62–65] use extra data during the training stage or the inference stage, such as LiDAR points, depth maps, and external data, which requires much higher annotation costs and makes neural networks even more complex.

4.3. The depth and dimensions L_1 errors on the KITTI validation set

We evaluate the L_1 errors of the most important 3D attributes in M3OD, including object depth, 3D height, 3D length, and 3D width on the KITTI validation set, as shown in Table 6. The improvement over our baseline method [11] demonstrates the effectiveness of our approach.

Table 2
AP_{40} scores on the KITTI 3D object detection *validation* set for Car category. We highlight the best results in bold.

Method	3D@IoU=0.7			BEV@IoU=0.7			3D@IoU=0.5			BEV@IoU=0.5		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MonoGRNet [29]	11.90	7.56	5.76	19.72	12.81	10.15	47.59	32.28	25.50	48.53	35.94	28.59
MonoDIS [46]	11.06	7.60	6.37	18.45	12.58	10.66	–	–	–	–	–	–
M3D-RPN [47]	14.53	11.07	8.65	20.85	15.62	11.88	48.53	35.94	28.59	53.35	39.60	31.76
MoVi-3D [48]	14.28	11.13	9.68	22.36	17.87	15.73	–	–	–	–	–	–
MonoPair [49]	16.28	12.30	10.42	24.12	18.17	15.76	55.38	42.39	37.99	61.06	47.63	41.92
MonoDLE [50]	17.45	13.66	11.68	24.97	19.33	17.01	55.41	43.42	37.81	60.73	46.87	41.89
MonoFENet [50]	17.54	11.16	9.74	30.21	20.47	17.58	59.93	42.67	37.50	66.43	47.96	43.73
GrooMeD-NMS [51]	19.67	14.32	11.27	27.38	19.75	15.92	55.62	41.07	32.89	61.83	44.98	36.29
MonoFlex [52]	23.64	17.51	14.83	–	–	–	–	–	–	–	–	–
GUPNet [11]	22.76	16.46	13.72	31.07	22.94	19.75	57.62	42.33	37.59	61.78	47.06	40.88
DDCDC [4]	23.83	16.00	12.04	–	–	–	60.04	42.75	33.85	–	–	–
VoPoints [5]	20.81	16.92	16.79	25.83	23.41	21.24	56.12	42.75	37.03	58.65	45.72	43.31
DID-M3D [53]	22.98	16.12	14.03	31.10	22.76	19.50	–	–	–	–	–	–
MoGDE [17]	23.35	20.35	17.71	–	–	–	–	–	–	–	–	–
DEVIANT [54]	24.63	16.54	14.52	32.60	23.04	19.99	61.00	46.00	40.18	65.28	49.63	43.50
MonoDTR [55]	24.52	18.57	15.51	33.33	25.35	21.68	64.03	47.32	42.20	69.04	52.47	45.90
MonoGround [16]	25.24	18.69	15.58	32.68	24.79	20.56	62.60	47.85	41.97	67.36	51.83	45.65
GPro3D (Ours)	**26.93**	**20.41**	**17.74**	**34.62**	**25.18**	**22.71**	**64.97**	**49.02**	**44.12**	**69.84**	**54.14**	**49.01**

Table 3
AP_{11} scores on the KITTI 3D object detection *validation* set for Car category. We highlight the best results in bold.

Method	3D@IoU=0.7			BEV@IoU=0.7			3D@IoU=0.5			BEV@IoU=0.5		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Mono3D [56]	2.53	2.31	2.31	5.22	5.19	4.13	–	–	–	–	–	–
Deep3DBox [33]	5.85	4.19	3.84	9.99	7.71	5.30	27.04	20.55	15.88	30.02	23.77	18.83
Mono3D++ [19]	10.60	7.90	5.70	16.70	11.50	10.10	42.00	29.80	24.20	46.70	34.30	28.10
M3D-RPN [47]	20.27	17.06	15.21	25.94	21.18	21.18	48.96	39.57	33.01	53.35	39.60	31.76
RTM3D [22]	20.77	20.77	16.63	25.56	22.12	20.91	54.36	41.90	35.84	57.47	44.16	42.31
RARNet [57]+M3D-RPN [47]	23.12	19.82	16.19	29.16	22.14	18.78	51.20	44.12	32.12	57.12	44.41	37.12
GUPNet [11]	25.76	20.48	17.24	34.00	24.81	22.96	59.36	45.03	38.14	62.58	46.82	44.81
GPro3D (Ours)	**29.91**	**24.84**	**21.37**	**38.13**	**28.95**	**27.11**	**65.84**	**49.78**	**46.79**	**69.98**	**56.04**	**49.51**

Table 4
3D and BEV AP_{40} scores on the KITTI 3D object detection *test* set for Car category at IoU 0.7. We highlight the best results in bold and underline the second best results.

Method	Extra data	AP_{3D}			AP_{BEV}		
		Easy	Moderate	Hard	Easy	Moderate	Hard
MonoPSR [62]	LiDAR	10.76	7.25	5.85	18.33	12.58	9.91
AM3D [63]	Depth	16.50	10.74	9.52	25.03	17.32	14.91
Decoupled-3D [64]	Depth	11.08	7.02	5.63	23.16	14.82	11.25
PatchNet [30]	Depth	15.68	11.12	10.17	22.97	16.86	14.97
D4LCN [18]	Depth	16.65	11.72	9.51	22.51	16.02	12.55
Kinem3D [65]	Multi-frames	19.07	12.72	9.17	26.69	17.52	13.10
CaDDN [66]	LiDAR	19.17	13.41	11.46	27.94	18.91	17.19
DFR-Net (I+D) [67]	Depth	19.40	13.63	10.35	28.17	19.17	14.84
MonoEF [36]	External	21.29	13.87	11.71	29.03	19.70	17.26
MonoDIS [46]	None	10.37	7.94	6.40	17.23	13.19	11.12
M3D-RPN [47]	None	14.76	9.71	7.42	21.02	13.67	10.23
SMOKE [68]	None	14.03	9.76	7.84	20.83	14.49	12.75
MonoPair [49]	None	13.04	9.99	8.65	19.28	14.83	12.89
MonoRCNN [35]	None	18.36	12.65	10.03	25.48	18.11	14.10
DDMP-3D [69]	None	19.71	12.78	9.80	28.08	17.89	13.44
MonoFlex [52]	None	19.94	13.89	12.07	28.23	19.75	16.89
Fine-Grained [20]	None	20.28	13.12	9.56	–	–	–
MonoGround [16]	None	21.37	14.36	12.62	<u>30.07</u>	<u>20.47</u>	<u>17.74</u>
MonoDTR [55]	None	<u>21.99</u>	<u>15.39</u>	<u>12.73</u>	28.59	20.38	17.14
GPro3D (Ours)	None	**22.41**	**15.44**	**12.84**	**30.31**	**20.79**	**18.21**

4.4. Ablation study

To verify the effectiveness of our approach's architecture, we conduct a detailed ablation study on the KITTI validation set, as shown in Tables 7 and 8.

4.4.1. Effectiveness of the ground plane estimation

In Table 7, Exp (a) is the baseline in which the object's 3D attributes are predicted directly. Exp (b) uses the preset fixed ground plane to inversely project the wheel pixels and deduce the 3D BBox. By comparing the experimental results (a → b), we discover that introducing the inverse projection with the ground plane improves accuracy marginally. This observation proves our motivation that directly using the preset fixed ground plane is likely to encounter the problem of ground plane tilt, thus resulting in non-negligible errors. Exp (c) accomplishes ground plane estimation by directly predicting the horizon line through the network without the vertical edges' assistance. From Exp (b) to Exp (c), the 3D AP_{40} increases by 1.26% on the moderate level. Exp (d) uses vertical edge detection in the image to calculate the roll angle of the ground plane and assumes that the pitch angle of the ground plane is zero. After the vertical edge detection is equipped, the 3D AP_{40}

Table 5
AP_{40} scores on the KITTI 3D object detection *test* set for Car, Pedestrian and Cyclist category. We highlight the best results in bold and underline the second-best results.

Method	Extra data	Car@IoU=0.7			Pedestrian@IoU=0.5			Cyclist@IoU=0.5		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
D4LCN [18]	Depth	16.65	11.72	9.51	4.55	3.42	2.83	2.45	1.67	1.36
Kinematic [65]	Multi-frames	19.07	12.72	9.17	–	–	–	–	–	–
MonoEF [36]	External	21.29	13.87	11.71	4.27	2.79	2.21	1.80	0.92	0.71
MonoPair [49]	None	13.04	9.99	8.65	10.02	6.68	5.53	3.79	2.12	1.83
M3DSSD [70]	None	17.51	11.46	8.98	5.16	3.87	3.08	2.10	1.51	1.58
DFR-Net (I) [67]	None	17.30	11.89	9.32	6.62	4.58	4.17	1.63	1.01	1.02
MonoDLE [50]	None	17.23	12.26	10.29	5.34	3.28	2.83	4.59	2.66	2.45
GrooMeD-NMS [51]	None	18.10	12.32	9.65	–	–	–	–	–	–
MonoFlex [52]	None	19.94	13.89	12.07	9.43	6.31	5.26	4.17	2.35	2.04
GUPNet [11]	None	20.11	14.20	11.77	**14.72**	**9.53**	7.87	4.18	2.65	2.09
MonoGround [16]	None	21.37	14.36	12.62	12.37	7.89	7.13	4.62	2.68	2.53
GPro3D (Ours)	None	**22.41**	**15.44**	**12.84**	14.61	9.36	**7.91**	**5.45**	**3.05**	**2.56**

Table 6
The depth and dimensions L_1 errors for KITTI validation set (meters). The lower the better.

Method	Depth	3D height	3D length	3D width
Baseline [11]	1.362	0.084	0.324	0.087
Ours	0.991	0.073	0.288	0.075

Table 7
Influence of ground plane estimation on M3OD AP_{40} results for Car category. Exp (a) is the baseline in which the object's 3D attributes are predicted directly. IP here represents the inverse projection with the ground plane prior. The horizon here represents the ground plane estimation by directly predicting the horizon through the network. VerE means vertical edge detection.

Exp	IP	Horizon	VerE	3D@IoU=0.7			BEV@IoU=0.7		
				Easy	Mod.	Hard	Easy	Mod.	Hard
(a)	–	–	–	17.36	12.97	10.74	24.80	18.38	16.33
(b)	✓	–	–	17.62	14.04	11.88	24.74	19.12	17.15
(c)	✓	✓	–	20.33	15.30	13.24	28.52	21.69	18.33
(d)	✓	–	✓	22.71	15.64	13.51	29.52	21.45	18.40
(e)	✓	✓	✓	**26.93**	**20.41**	**17.74**	**34.62**	**25.18**	**22.71**

Table 8
Influence of projection points on M3OD AP_{40} results. We compare objects' wheel pixels and the bottom vertices of the BBoxes.

Keypoint	Car@IoU=0.7 AP_{3D}/AP_{BEV}		
	Easy	Mod.	Hard
bottom vertices	18.43/25.91	14.84/21.51	11.53/18.36
wheel pixels	26.93/34.62	20.41/25.18	17.74/22.71

Keypoint	Cyclist@IoU=0.5 AP_{3D}/AP_{BEV}		
	Easy	Mod.	Hard
bottom vertices	7.84/7.07	2.44/3.84	1.95/3.56
wheel pixels	7.85/9.09	4.48/5.53	3.97/4.74

on the moderate setting increases by 1.60% (b → d). Exp (e) is our complete GPro3D. It uses vertical-edge-enhanced horizon line detection to estimate the ground plane equation, which achieves the best result.

4.4.2. Influence of the projection point

As shown in Table 8, we conduct experiments using objects' wheel pixels and the bottom vertices. We use our estimated accurate ground plane equation in dynamic inverse projection for both wheel pixels and bottom vertices. From the results, the AP_{40} increases dramatically when using wheel pixels as the projection points.

Moreover, we analyzed the position selection for wheel pixel pseudo-labels. Specifically, we experimented with k_l, i.e., the ratio of the front and rear wheels' distance to the length of the 3D bounding box. For k_l, we took values of 0.6, 0.7, and 0.8 to generate wheel pixel pseudo-labels. The experimental results, presented in Table 9, demonstrate that the optimal performance was achieved when k_l was set to 0.7. Consequently, we employ 0.7 as k_l's value.

Table 9
Influence of wheel pixel pseudo-labels selection on M3OD AP_{40} results. We compare the location selection of wheel pixel pseudo-labels.

wheel pixel pseudo label	Car@IoU=0.7 AP_{3D}/AP_{BEV}		
	Easy	Mod.	Hard
0.6 length	23.81/30.52	16.94/22.07	13.55/18.71
0.7 length	26.93/34.62	20.41/25.18	17.74/22.71
0.8 length	25.37/32.88	18.54/23.61	15.29/19.05

Table 10
Cross-dataset evaluation on the KITTI and nuScenes frontal validation dataset.

Dataset	Method	Depth prediction mean error ↓		
		[0, 20)	[20, 40)	[40, +∞)
KITTI	M3D-RPN [47]	0.56	1.33	2.73
	MonoRCNN [35]	**0.46**	1.27	2.59
	GUPNet [11]	0.54	1.21	2.49
	GPro3D (Ours)	**0.46**	**1.12**	**2.22**
nuScenes	M3D-RPN [47]	1.04	3.29	10.73
	MonoRCNN [35]	0.94	2.84	8.65
	GUPNet [11]	0.83	2.14	5.98
	GPro3D (Ours)	**0.72**	**2.11**	**5.79**

4.5. Cross-dataset evaluation

To demonstrate the generalization capacity of our approach, we conduct cross-dataset evaluation. As shown in Table 10, all models are trained on the KITTI train set. We evaluate the models on the KITTI validation set and the nuScenes frontal set. Since depth prediction is the most important metric in M3OD, we follow [35,47] and evaluate the depth errors in different ranges. We introduce more ground plane geometrical clues in the scenes, which are robust and generalize better across different datasets and domains.

4.6. Visualization analysis

We visualize some examples to further demonstrate the effectiveness of the proposed GPro3D. As shown in Fig. 8, GPro3D tends to predict

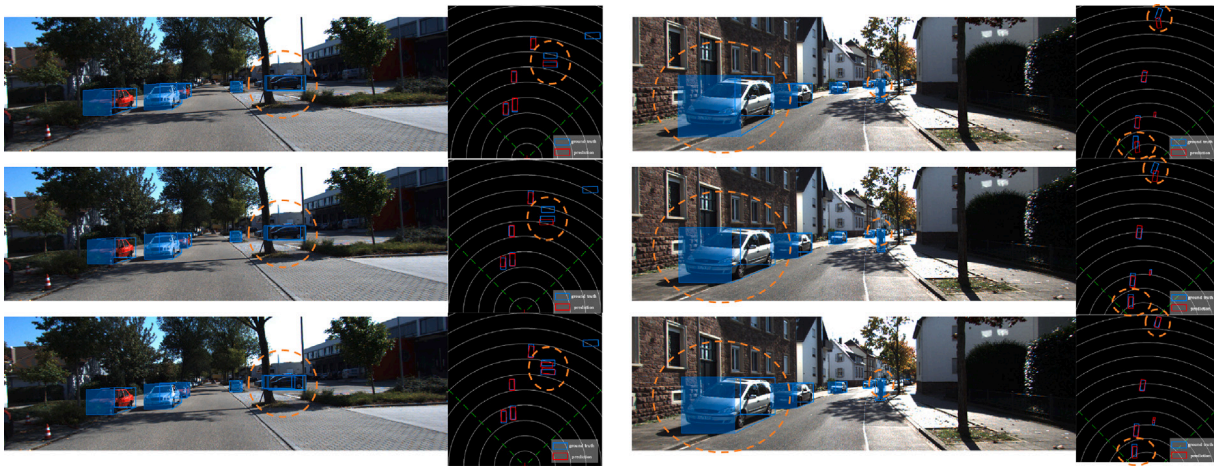

Fig. 8. Visualization results. From top to bottom, we show examples generated by using 3D boxes' bottom vertices as projection points, by using a preset fixed ground plane, and by our GPro3D, respectively.

Fig. 9. Examples of the vertical edge and horizon line detection. The green lines are detected vertical edges, and the red lines represent the detected horizon lines in the image. The proposed vertical-edge-enhanced horizon line detection method can work effectively even if the horizon line is heavily occluded by obstacles.

Fig. 10. The 2D BBox, wheel pixel and 3D BBox detection results. They are represented by yellow, green, and blue, respectively. Although some wheel pixels are blocked in the image, our model can still work well.

a more accurate spatial position compared with two competitors. For example, in the example on the right, the car circled by the bigger orange ellipse has the best-fitting 3D boxes generated by our GPro3D, which reveals a more accurate 3D size estimation. Meanwhile, as shown in the bird's-eye view, the distance between the prediction result and the ground truth is apparently closer, especially for distant objects (circled by the smaller orange ellipse). This observation indicates that the proposed GPro3D can produce higher quality 3D bounding boxes. Fig. 9 is our vertical edge and horizon line detection result. As we can see, the proposed vertical-edge-enhanced horizon line detection method can work effectively even if the horizon is heavily occluded by obstacles. Fig. 10 shows the 2D BBox, wheel pixels , and 3D BBox detection results. Note that our model can still work when some wheel pixels are occluded in the image, demonstrating our approach's robustness.

5. Conclusions

In this paper, we identify three key issues that highly hinder the application of the ground plane prior in monocular 3D object detection, i.e., the projection point localization issue, the ground plane tilt problem, and the lack of ground plane annotation issue. To tackle the issues, we propose a GPro3D framework. Specifically, for the projection point localization issue, we adopt the objects' wheel pixels as explicit and salient signals. Such wheel pixels can be well projected into the 3D space with the ground plane equation, resulting in a more accurate geometry estimation. To deal with the ground plane tilt problem, we leverage the horizon line in the image to estimate the ground plane equation. We also propose a vertical-edge-enhanced method for the horizon line detection when the horizon is highly occluded. Pseudo-labels for wheel pixels and horizon lines are generated using only

M3OD labels. With the wheel pixels and the accurate ground plane equation, we could fully use the ground plane and directly derive objects' 3D BBoxes and 3D attributes geometrically. Extensive experiment results show that the proposed GPro3D can significantly outperforms baseline methods, demonstrating the proposed framework's effectiveness and superiority. In the future, we will explore object detection tasks in more challenging scenarios, such as long-tail distribution, category imbalance scenarios [71], etc.

CRediT authorship contribution statement

Fan Yang: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Xinhao Xu:** Methodology, Software, Visualization, Writing – original draft. **Hui Chen:** Methodology, Software, Writing – review & editing. **Yuchen Guo:** Methodology, Visualization, Writing – review & editing. **Yuwei He:** Methodology, Writing – review & editing. **Kai Ni:** Conceptualization, Investigation, Writing – review & editing. **Guiguang Ding:** Conceptualization, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

This work was supported by National Natural Science Foundation of China (Nos. U1936202, 61925107, 62271281, 62021002, 61971260), Zhejiang Provincial Natural Science Foundation of China (No. LDT23F 01013F01), and Zhejiang Lab Open Research Project (No. K2022KI 0AB01).

References

[1] D. Dai, J. Wang, Z. Chen, H. Zhao, Image guidance based 3D vehicle detection in traffic scene, Neurocomputing 428 (2021) 1–11.
[2] G. Tian, L. Liu, J. Ri, Y. Liu, Y. Sun, ObjectFusion: An object detection and segmentation framework with RGB-D SLAM and convolutional neural networks, Neurocomputing 345 (2019) 3–14.
[3] Y. Wu, Y. Pang, B. Gao, J. Han, Complementary features with reasonable receptive field for road scene 3D object detection, in: 2019 IEEE International Conference on Image Processing, ICIP, IEEE, 2019, pp. 3905–3909.
[4] X. Wu, D. Ma, X. Qu, X. Jiang, D. Zeng, Depth dynamic center difference convolutions for monocular 3D object detection, Neurocomputing 520 (2023) 73–81.
[5] H. Chu, L. Mo, R. Wang, T. Hu, H. Ma, Visibility of points: Mining occlusion cues for monocular 3D object detection, Neurocomputing 502 (2022) 48–56.
[6] W. Chen, P. Li, H. Zhao, MSL3D: 3D object detection from monocular, stereo and point cloud for autonomous driving, Neurocomputing 494 (2022) 23–32.
[7] P. Li, H. Zhao, Monocular 3D object detection using dual quadric for autonomous driving, Neurocomputing 441 (2021) 151–160.
[8] J. Zhang, Q. Su, C. Wang, H. Gu, Monocular 3D vehicle detection with multi-instance depth and geometry reasoning for autonomous driving, Neurocomputing 403 (2020) 182–192.
[9] W. Li, J. Gu, B. Chen, J. Han, Incremental instance-oriented 3D semantic mapping via RGB-d cameras for unknown indoor scene, Discrete Dyn. Nat. Soc. 2020 (2020) 1–10.
[10] J. Zhang, G. Fang, B. Wang, X. Zhou, Q. Pei, C. Chen, Monocular 3D object detection with pseudo-lidar confidence sampling and hierarchical geometric feature extraction in 6G network, Digit. Commun. Netw. (2022).
[11] Y. Lu, X. Ma, L. Yang, T. Zhang, Y. Liu, Q. Chu, J. Yan, W. Ouyang, Geometry uncertainty projection network for monocular 3d object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 3111–3121.
[12] Y. Ming, X. Meng, C. Fan, H. Yu, Deep learning for monocular depth estimation: A review, Neurocomputing 438 (2021) 14–33.
[13] L. Fang, D. Zhu, J. Yue, B. Zhang, M. He, Geometric-spectral reconstruction learning for multi-source open-set classification with hyperspectral and LiDAR data, IEEE/CAA J. Autom. Sin. 9 (10) (2022) 1892–1895.
[14] D. Meng, L. Li, X. Liu, L. Gao, Q. Huang, Viewpoint alignment and discriminative parts enhancement in 3d space for vehicle reid, IEEE Transactions on Multimedia (2022).
[15] J. Ku, M. Mozifian, J. Lee, A. Harakeh, S. Waslander, Joint 3D proposal generation and object detection from view aggregation, in: IROS, 2018.
[16] Z. Qin, X. Li, MonoGround: Detecting monocular 3D objects from the ground, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3793–3802.
[17] Y. Zhou, Q. Liu, H. Zhu, Y. Li, S. Chang, M. Guo, Mogde: Boosting mobile monocular 3d object detection with ground depth estimation, Adv. Neural Inf. Process. Syst. 35 (2022) 2033–2045.
[18] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, P. Luo, Learning depth-guided convolutions for monocular 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 1000–1001.
[19] T. He, S. Soatto, Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, 2019, pp. 8409–8416.
[20] H. Liu, H. Liu, Y. Wang, F. Sun, W. Huang, Fine-grained multi-level fusion for anti-occlusion monocular 3D object detection, IEEE Trans. Image Process. (2022).
[21] X. Xu, Z. Chen, F. Yin, Multi-scale spatial attention-guided monocular depth estimation with semantic enhancement, IEEE Trans. Image Process. 30 (2021) 8811–8822.
[22] P. Li, H. Zhao, P. Liu, F. Cao, Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving, in: European Conference on Computer Vision, Springer, 2020, pp. 644–660.
[23] L. Wu, L. Fang, J. Yue, B. Zhang, P. Ghamisi, M. He, Deep bilateral filtering network for point-supervised semantic segmentation in remote sensing images, IEEE Trans. Image Process. 31 (2022) 7419–7434.
[24] W. Lao, J. Han, et al., 3D modeling for capturing human motion from monocular video, in: Proceedings of the 27th Symposium on Information Theory in the Benelux, 8-9 June 2006, Noordwijk, the Netherlands, Werkgemeenschap voor Informatie-en Communicatietheorie (WIC), 2006, pp. 299–306.
[25] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, T. Chateau, Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2040–2049.
[26] J. Xie, Y. Pang, J. Nie, J. Cao, J. Han, Latent feature pyramid network for object detection, IEEE Trans. Multimed. (2022).
[27] H. Wang, Z.-J. Zha, L. Li, X. Chen, J. Luo, Semantic and relation modulation for audio-visual event localization, IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).
[28] Y. Zhang, J. Lu, J. Zhou, Objects are different: Flexible monocular 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2021, pp. 3289–3298.
[29] Z. Qin, J. Wang, Y. Lu, Monogrnet: A general framework for monocular 3d object detection, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[30] X. Ma, S. Liu, Z. Xia, H. Zhang, X. Zeng, W. Ouyang, Rethinking pseudo-lidar representation, in: European Conference on Computer Vision, Springer, 2020, pp. 311–327.
[31] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, J.A. Benediktsson, Deep learning for hyperspectral image classification: An overview, IEEE Trans. Geosci. Remote Sens. 57 (9) (2019) 6690–6709.
[32] L. Li, X. Gao, J. Deng, Y. Tu, Z.-J. Zha, Q. Huang, Long short-term relation transformer with global gating for video captioning, IEEE Transactions on Image Processing 31 (2022) 2726–2738.
[33] A. Mousavian, D. Anguelov, J. Flynn, J. Kosecka, 3D bounding box estimation using deep learning and geometry, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7074–7082.
[34] Y. Liu, Y. Yixuan, M. Liu, Ground-aware monocular 3d object detection for autonomous driving, IEEE Robot. Autom. Lett. 6 (2) (2021) 919–926.
[35] X. Shi, Q. Ye, X. Chen, C. Chen, Z. Chen, T.-K. Kim, Geometry-based distance decomposition for monocular 3d object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 15172–15181.
[36] Y. Zhou, Y. He, H. Zhu, C. Wang, H. Li, Q. Jiang, Monocular 3d object detection: An extrinsic parameter free approach, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 7556–7566.
[37] X. Zhou, D. Wang, P. Krähenbühl, Objects as points, 2019, arXiv preprint arXiv:1904.07850.
[38] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2980–2988.
[39] X. Liu, L. Li, S. Wang, Z.-J. Zha, Z. Li, Q. Tian, Q. Huang, Entity-enhanced adaptive reconstruction network for weakly supervised referring expression grounding, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (3) (2022) 3003–3018.
[40] G. Bradski, A. Kaehler, Opencv, Dr. Dobb's J. Softw. Tools 3 (2000) 2.

[41] M. Naveenkumar, A. Vadivel, Opencv for computer vision applications, in: Proceedings of National Conference on Big Data and Cloud Computing, NCBDC'15, 2015, pp. 52–56.
[42] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, ACM Sigmod Rec. 25 (2) (1996) 103–114.
[43] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 3354–3361.
[44] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, Nuscenes: A multimodal dataset for autonomous driving, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11621–11631.
[45] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.
[46] A. Simonelli, S.R. Bulo, L. Porzi, M. López-Antequera, P. Kontschieder, Disentangling monocular 3d object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1991–1999.
[47] G. Brazil, X. Liu, M3d-rpn: Monocular 3d region proposal network for object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9287–9296.
[48] A. Simonelli, S.R. Bulo, L. Porzi, E. Ricci, P. Kontschieder, Towards generalization across depth for monocular 3d object detection, in: European Conference on Computer Vision, Springer, 2020, pp. 767–782.
[49] Y. Chen, L. Tai, K. Sun, M.M. Li, Monocular 3D object detection using pairwise spatial relationships, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 2020, pp. 14–19.
[50] X. Ma, Y. Zhang, D. Xu, D. Zhou, S. Yi, H. Li, W. Ouyang, Delving into localization errors for monocular 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 4721–4730.
[51] A. Kumar, G. Brazil, X. Liu, Groomed-nms: Grouped mathematically differentiable nms for monocular 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8973–8983.
[52] Y. Zhang, J. Lu, J. Zhou, Objects are different: Flexible monocular 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3289–3298.
[53] L. Peng, X. Wu, Z. Yang, H. Liu, D. Cai, DID-M3D: Decoupling instance depth for monocular 3D object detection, in: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I, Springer, 2022, pp. 71–88.
[54] A. Kumar, G. Brazil, E. Corona, A. Parchami, X. Liu, Deviant: Depth equivariant network for monocular 3d object detection, in: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX, Springer, 2022, pp. 664–683.
[55] K.-C. Huang, T.-H. Wu, H.-T. Su, W.H. Hsu, MonoDTR: Monocular 3D object detection with depth-aware transformer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4012–4021.
[56] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, R. Urtasun, Monocular 3d object detection for autonomous driving, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2147–2156.
[57] L. Liu, C. Wu, J. Lu, L. Xie, J. Zhou, Q. Tian, Reinforced axial refinement network for monocular 3d object detection, in: European Conference on Computer Vision, Springer, 2020, pp. 540–556.
[58] X. Chen, H. Ma, J. Wan, B. Li, T. Xia, Multi-view 3d object detection network for autonomous driving, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1907–1915.
[59] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019).
[60] F. Yu, D. Wang, E. Shelhamer, T. Darrell, Deep layer aggregation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2403–2412.
[61] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[62] J. Ku, A.D. Pon, S.L. Waslander, Monocular 3d object detection leveraging accurate proposals and shape reconstruction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11867–11876.
[63] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, X. Fan, Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6851–6860.
[64] Y. Cai, B. Li, Z. Jiao, H. Li, X. Zeng, X. Wang, Monocular 3d object detection with decoupled structured polygon estimation and height-guided depth estimation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 07, 2020, pp. 10478–10485.
[65] G. Brazil, G. Pons-Moll, X. Liu, B. Schiele, Kinematic 3d object detection in monocular video, in: European Conference on Computer Vision, Springer, 2020, pp. 135–152.
[66] C. Reading, A. Harakeh, J. Chae, S.L. Waslander, Categorical depth distribution network for monocular 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8555–8564.
[67] Z. Zou, X. Ye, L. Du, X. Cheng, X. Tan, L. Zhang, J. Feng, X. Xue, E. Ding, The devil is in the task: Exploiting reciprocal appearance-localization features for monocular 3D object detection, in: ICCV, 2021, pp. 2713–2722.
[68] Z. Liu, Z. Wu, R. Tóth, Smoke: Single-stage monocular 3d object detection via keypoint estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 996–997.
[69] L. Wang, L. Du, X. Ye, Y. Fu, G. Guo, X. Xue, J. Feng, L. Zhang, Depth-conditioned dynamic message propagation for monocular 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 454–463.
[70] S. Luo, H. Dai, L. Shao, Y. Ding, M3DSSD: Monocular 3D single stage object detector, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2021, pp. 6145–6154.
[71] L. Fang, Q. Tang, L. Ouyang, J. Yu, J. Lin, S. Ding, L. Tang, Long-tailed object detection of kitchen waste with class-instance balanced detector, Sci. China Technol. Sci. (2023) 1–12.

Fan Yang received his B.E. degree from School of Software, Tsinghua University, China, in 2021. He is currently a Ph.D. student at School of Software, Tsinghua University. His research interests include artificial intelligence, computer vision, and autonomous driving.

Xinhao Xu received his B.E. degree from School of Software, Tsinghua University, China, in 2022. He is currently a Master student at School of Software, Tsinghua University. His research interests include computer vision and artificial intelligence.

Hui Chen received his Ph.D. and B.E. degrees from Tsinghua University, Beijing, China in 2021 and 2016 respectively. He was a Postdoc researcher at Department of Automation, Tsinghua University, between 2021 and 2023. Now he is an assistant researcher at Beijing National Research Center for Information Science and Technology, Tsinghua University. His research interests focus on artificial intelligence.

Yuchen Guo received his Ph.D. degree and B.E. degree from School of Software, Tsinghua University, Beijing, China in 2018 and 2013 respectively. He was a Postdoc researcher in Department of Automation, Tsinghua University, between 2018 and 2020. Now he is an associate researcher at Beijing National Research Center for Information Science and Technology, Tsinghua University. His research interests focus on artificial intelligence.

Yuwei He received his Ph.D. degree from School of Software, Tsinghua University, Beijing, China in 2021. Now he is a Postdoc researcher at School of Software, Tsinghua University. His research interests include computer vision and medical imaging.

Kai Ni received his Ph.D. degree from College of Computing, Georgia Institute of Technology. He is the founder and CEO of HoloMatic Technology Co., Ltd. His research interests include autonomous driving, computer vision, robotics, and machine learning.

Guiguang Ding is currently a full professor in the School of Software, Tsinghua University, China. His research interests include the areas of multimedia information retrieval, computer vision, and machine learning.