Shape-Aware Monocular 3D Object Detection
Wei Chen, Jie Zhao, Wan-Lei Zhao, and Song-Yuan Wu

Abstract—The 3D object detection is the key issue in the autonomous driving system. This issue is particularly challenging when the detection only relies on a single perspective camera. The anchor-free and keypoint-based models receive increasing attention recently due to their effectiveness and simplicity. However, most of these methods are vulnerable to the occlusion and truncation of objects. In this paper, a single-stage monocular 3D object detection model is proposed. An instance-segmentation head is integrated into the model training, which allows the model to be aware of the visible shape of a target object. Therefore, the detection largely avoids interference from irrelevant regions surrounding the target objects. In addition, we also reveal that the popular IoU-based evaluation metrics, which were originally designed for evaluating stereo or LiDAR-based detection methods, are insensitive to the improvement achieved by the monocular 3D object detection algorithms. A novel evaluation metric, namely average depth similarity (ADS) is proposed for the monocular 3D object detection models. Our method outperforms the comparison baseline in terms of both the popular and the proposed evaluation metrics while maintaining real-time efficiency.

Index Terms—Monocular 3D object detection, autonomous driving, occlusion, shape-aware.

I. INTRODUCTION

The three-dimensional visual object detector is a core component of an autonomous driving system. The quality of the detection results has a direct impact on the subsequent tasks, such as tracking and driving planning. Recently, we have witnessed the great advances in the design of 3D object detection models. Most of them [1], [2], [3] rely on 3D LiDAR laser scanners. Although LiDAR point clouds allow the detectors to achieve accurate 3D scene localization, the hardware is too expensive to be equipped with normal cars. Stereo camera rigs are the alternative option in many 3D object detection models [4], [5], [6], where depth information is estimated by building accurate pixel correspondences between the left and the right cameras. However, binocular methods are usually computationally expensive and come with higher bandwidth costs. The stringent conditions mentioned above hinder the practical application of these methods. To circumvent these difficulties, increasing efforts have been made in developing cheaper 3D object detectors. They aim at detecting the 3D pose of objects from a single 3-channel RGB image [7], [8], [9], [10], [11] that has been captured by a monocular camera.

Despite recent progress, the performance gap between monocular 3D object detectors and stereo-based or LiDAR-based detectors remains wide. Compared with stereo-based or LiDAR-based detectors, it is an ill-posed problem that recovers the depth information from a single image captured by a perspective camera. Due to the irreversible information loss introduced by the projection process of a perspective camera, it is challenging to recover the depth from a single image. Instead of directly regressing the object depth [10], [12], [13], most of the recent works [7], [9], [11] recover the depth of objects by modeling the relationship between the 3D geometric prior and the detected 2D keypoints. For instance, one can infer the depth of a target object by estimating the height of the 3D object and the height of the 2D projection, since the height of the 2D projection is inversely proportional to the depth when the 3D height is constant. These geometric-constraint-based methods [7], [9], [11] usually show stronger generalization ability than the one that regresses the depth of objects directly [10], [12], [13].

The depth estimation is particularly challenging when the object is occluded or truncated in the 2D view, as the surrounding 2D keypoints of the object cannot be fully recovered. Objects are only partially visible due to either the occlusion by other objects or the truncation due to the limited FoV (Field of View) of cameras. In this case, it is difficult to provide an accurate prediction for an object since its shape is largely missing. Firstly, information that can be extracted from the partially visible object is very limited. Secondly, the available occluded training samples are very limited compared to those fully-visible ones. For this reason, the occluded objects cannot be sufficiently trained. Finally, most of the existing object detection methods are center-based, for which an object center (instead of a 3D bounding box) is assigned to a detected object. However, for the occluded/truncated objects, their center points often lie on other irrelevant objects. In such cases, the features derived from

Fig. 1. Feature expansion versus RoIAlign. In order to be compatible with the center-based detector, feature input to the segmentation task is extracted from the center of an object (figure (b)), instead of RoIAlign (figure (a)), and expanded by broadcasting.
the center point and its neighboring area are mixed with irrelevant contents. In this paper, a novel auxiliary training task is designed to address the training failures caused by the object occlusion in the training process. Namely, an additional instance segmentation branch is added to the model. By learning to predict the visibility mask of objects, our model becomes “shape-aware”. The feature is therefore derived from a real target object region. Moreover, the segmentation head is designed to be center-based to be compatible with the detection heads. In contrast to popular segmentation methods, the features input to the segmentation head is extracted by expanding the center feature instead of RoIAlign (shown in Figure 1(b)).

Nevertheless, the training task of instance segmentation requires pixel-level annotation on the monocular image, which is a laborious task to undertake. To overcome this difficulty, the pixel-level object annotation is approximated by utilizing the 3D object-level annotation and the corresponding point cloud. We simply project the 3D points inside the 3D bounding box onto the image plane to form an instance mask used for training. Additionally, an uncertainty-weighted loss function is adopted to learn useful information from these noisy and sparse generated instance segmentation annotations.

In our paper, we also reveal the limitations of the current evaluation measure for 3D object detection. With the current IoU-based evaluation metrics, only the overlapping with the ground-truth objects is counted. The performance can be boosted by simply duplicating detected 3D bounding boxes and putting in different positions and layouts, which increases the chance of overlapping with the ground-truth while making no improvement in the 3D object detection. To address this issue, a new measure called average depth similarity (ADS) is proposed to evaluate the overlaps with the ground-truth in the object depth. In combination with 3D IoU-based metrics, a more objective evaluation can be made for 3D object detection methods that are based on the monocular image.

II. RELATED WORK

In this section, we briefly review the 3D object detection methods based on 3D point clouds and binocular images first. Then the review focuses on the methods built upon a monocular image since our method also falls in this category.

In the representative LiDAR-based 3D object detection method [1], 3D point clouds are encoded as 2D feature maps by projecting them into the bird-eye-view (BEV), then a 2D CNN is applied for detection. In VoxelNet [2], the neighboring cloud points are grouped into voxels. Thereafter, PointNet [14] is adopted in each voxel for feature encoding. The 3D convolution, which is used in VoxelNet for object detection, comes with an expensive computational cost. SECOND [3] improves the inference speed of VoxelNet by introducing sparse convolution based on the fact that many voxels are empty due to the point cloud sparsity. LiDAR-based detectors are able to provide accurate 3D detection results, however, the expensive sensor cost hinders the application of these methods.

Besides 3D point clouds, depth information can also be recovered from stereo cameras. Stereo R-CNN [4] extends the standard Faster R-CNN to detect and match the left and right objects, then refines the 3D detection with dense Roi (Region of Interest) alignment. In [6], pseudo-LiDAR points extracted from the disparity map are directly used as the input for 3D object detectors, achieving state-of-the-art performance. Stereo camera-based methods usually require dense disparity estimation on the full image or each Roi, which is computationally expensive. They also introduce extra bandwidth and calibration requirements compared to monocular image-based methods.

Different from the above methods, monocular-based 3D object detectors only take a single RGB image as input. Amongst these methods, recent models are mostly center-based [7], [8], [11], [12], where each object is represented with a representative center $x_c = (u_c, v_c)$. Depth, dimensions, and orientation are estimated as attributes of the detected center point. Theoretically speaking, depth estimation from a monocular image is ill-posed. There are various ways to undertake depth estimation. They can be roughly divided into three categories, namely the direct depth regression, depth-map-based, and geometric constraint-based methods.

There are many direct depth regression methods in the literature since the proposal of center-based detection network [12]. These methods are efficient as they do not rely on any additional depth-map estimation modules. The representative methods are SMOKE [10], MonoCon [8], and MonoDLE [15]. SMOKE [10] improves the estimation accuracy with a carefully designed attribute regression scheme, in which the loss functions are disentangled. MonoCon [8] attempted to improve the performance with simple 2D auxiliary training tasks on the monocular 3D object detection. In MonoDLE [15], the authors investigated the factors that impact the localization error. The direct regression method can also be integrated with the on-stage anchor-based networks. In M3DSSD [13], the performance of the detector is improved by introducing an asymmetric non-local attention block and a two-step feature alignment module. The ground information is discovered helpful in improving depth regression accuracy. In PGD [16], the fact “nearby objects are on the same ground” is utilized to refine the predicted depth with nearby objects. In Ground-Aware [17], the author mimics how human utilizes the ground plane information for depth inference with ground-aware convolutions. This work is further extended in [18] by introducing a novel architecture to exploit the information of line structures, which is supposed to be a useful cue for depth inference.

The depth map-based methods [6], [19] are also known as the pseudo-LiDAR (PL) based method, employs an off-the-shelf dense depth map estimation Convolutional Neural Network (CNN) as a substitute for LiDAR sensors. With the resulting pseudo-LiDAR, the detection task is usually achieved by applying a state-of-the-art LiDAR-based detector. Due to biased training protocol [19], superior performance for PL-based methods is only observed on the validation set. According to [19], the training set adopted by their depth estimation modules heavily overlaps with the val set of the PL-based detectors. Although PL-based methods are able to capitalize on the pre-training on raw scenes without object labeling, the discrepancy between pseudo-LiDAR and real LiDAR sensor cannot be overcome [20]. In [20], DD3D leverages the depth prediction for pre-training, which leads to the better detection accuracy.

Apart from the above two types of depth estimation, the task can also be implemented by leveraging 3D-2D geometric constraints. RTM3D [9] estimates nine predefined keypoints which introduce $18$ projection constraints ($9$ pairs of $x$ and $y$) for each object. Then the depth of each object can be recovered by solving a nonlinear least-squares optimization problem. Due to the variety of car layouts, the estimated $x$-coordinate is usually less reliable than the $y$-coordinate for one keypoint. Therefore, only the geometric constraint on the height of keypoints is utilized in MonoFlex [7]. In addition, it also divides $10$ keypoints into three groups to produce three independent depth predictions. The final depth is a weighted average of all the depth predictions. In GUPNet [21], the depth is estimated via the 3D-2D height relationship. A GUP module is designed to alleviate the amplified error caused by the height prediction error during the inference.

In the above geometric-constraint-based methods, objects are modeled as 3D bounding boxes, and keypoints are often defined...
As corners, centers, or top/bottom points of the bounding boxes. These keypoints mostly lie on the background, e.g., the sky or the ground. Deep MANTA [22] and AutoShape [11] address this issue by annotating the keypoints of the training data with CAD templates via a semi-autonomous/autonomous approach. However, due to the mismatch between the predefined CAD model and the real object, the improvement that these methods achieve is very limited. The constraint can also be defined densely on each pixel. MonoRUN [23] formulates the 3D pose prediction as a problem of minimizing the projection error of corresponding 3D-2D points in each ROI.

In the literature, there are also a few methods that are out of the aforementioned three categories. For instance, LPCG [24] studies the problem of improving the detection quality with unlabeled data, which is applicable to several kinds of detectors. CaDDN [25] transforms the image feature map to the BEV space with a predicted depth distribution. The BEV feature map can be further fed into a standard 3D detection head for 3D detection.

While impressive improvements have been achieved by recent works, they’re usually designed for fully-visible objects in an ideal environment. In this paper, an auxiliary training task is introduced to assist the learning of occluded objects. Namely, instance segmentation is incorporated as another task for the detection network. As the task itself is shape-aware, the detected object regions are expected to shrink from a bounding box to the real object area. Moreover, since the annotations on the 3D point cloud are wisely utilized to produce the segmentation mask, the introduced segmentation head boosts the overall detection performance while no laborious pixel-level annotation is required.

III. SHAPE-AWARE 3D OBJECT DETECTION

As aforementioned, the existing monocular 3D object detection models are not sufficiently trained on the occluded objects. First of all, in the monocular detectors, objects are modeled as representative centers. Unfortunately, for the occluded objects, their center may lie on other objects, leading to the polluted feature representation that is derived from the center. Moreover, it is challenging for the network to recover the 3D properties of an object when only a small area of the target object is visible. Furthermore, very few samples are available for the severely occluded objects in the training set.

In this section, our solution for monocular 3D object detection is presented. The general framework is illustrated in Figure 2. Our detection pipeline in general follows MonoFlex in [7]. Each target object is detected as a center point along with its attributes. As shown in the framework, the classification branch estimates the raw center of the 2D point. While the depth and other attributes are regressed by the regression branch. The novelty of our method lies in the segmentation branch. It is designed to estimate the shape of a target object. Such that both the classification and the regression are trained to focus on the visible areas of an object. In the following, we first present the details of the proposed segmentation branch. Thereafter, we show how the 3D object and its attributes are estimated with classification and regression branches. The implementation details are discussed in the last part of this section.

A. SHAPE-quette Auxiliary Training Task

As shown in Figure 2, attaching to the backbone network, a segmentation branch is added. It is parallel to the classification and regression branches. Since the object detection pipeline is center-based, this segmentation branch should also be center-based so that they are compatible with each other. We extract a feature vector \( f_i \in \mathbb{R}^{64} \) for the \( i \)-th object which is copied from the input feature map at the position of the object center. This feature from the same position is also shared with the classification and regression branches. Feeding \( f_i \) to the segmentation branch makes the feature adaptive to the visible object shape. Therefore, the feature vector is trained to be “shape-aware”. As this feature is shared with the classification and regression branches, the feature input to these two branches becomes “shape-aware” as well. Therefore, the problem that feature \( f_i \) suffers interference from the surrounding objects is alleviated.

Thereafter, \( f_i \) is added with a position embedding map \( f_{pos} \in \mathbb{R}^{s \times s \times 64} \) with broadcasting, which expands the shape of \( f_i \), so that the two tensors could have the same shape before addition. A feature map with size \( s \times s \times 64 \) is, therefore, produced (as illustrated in Figure 1(b)). The position embedding map \( f_{pos} \) is produced by applying convolution on a predefined image with two channels. Each pixel at position \((x_p, y_p)\) equals to a linear transformed coordinate \((\tilde{x}_p, \tilde{y}_p) = \left( \frac{x_p}{s} - 0.5, \frac{y_p}{s} - 0.5 \right)\).

In order to train this center-based segmentation branch, the instance masks must be ready for the training images. However, pixel-level annotation is expensive. To circumvent this difficulty, we alternatively extract the instance masks from the 3D bounding boxes and the original point cloud. Inspired by the ground-truth sampling strategy used by LiDAR-based 3D detection methods [3], [26], we treat points inside the bounding boxes as instance-level foreground points. When the whole cloud of points is projected onto the image plane, the pixel-level mask label is produced for each object by checking whether a point is inside the box or not. An \( s \times s \) square image is used to represent the ground-truth mask. If there are two or more points that fall into the same pixel of the image, we keep one of them at random. As illustrated in Figure 3, the generated ground-truth masks are coarse and sparse, however, they are sufficient to make the whole detection pipeline become aware of the object shapes.
of the detected center, we assign the instance labels to each point that is inside the 3 × 3 neighborhood of the object center point. As illustrated in Figure 4, we first collect all points that belong to the 3 × 3 neighborhood of representative points. These points share the same object label as the center closest to them in terms of Euclidean distance. The center sampling is adopted in the regression and segmentation heads. For regression heads, the center sampling operation is performed on regressed attributes. For the segmentation head, the center sampling is applied on original features and before the segmentation head to avoid producing abundant masks on background positions. The center sampling strategy does not affect the heatmap prediction. We still apply the penalty-reduced focal loss [12], [31], [32] to train the heatmap. In the test stage, we can simply extract the object predictions by finding the maximum points of the heatmap through a max-pooling operation.

For efficiency, the segmentation head is turned on only during the training. It is already sufficient since the instance segmentation training is designed to assist the training of detection and regression tasks. Note that this instance segmentation branch currently is only compatible with center-based monocular detectors.

B. Multi-Task Learning in 3D Object Detection

Similar to other center-based methods [12], [33], each target object is detected as a center point along with its attributes in our method. The detection is built upon the DLA-34 [34] backbone network, which has a broad receptive field thanks to the aggregated deep features from different levels and the deformable convolution modules. There are several target parameters to be estimated (listed in Table I) in our model. In general, it is a multi-task deep learning framework. The classification head estimates the raw location (u_f, v_f) of the 2D center point. Other attributes are regressed by several regression heads. The segmentation head is designed to boost the performance of other heads.

2D Detection Similar to FCOS [30], the 2D bounding box is represented as distances from the representative center to the four sides of the box. GIoU loss [35] is adopted to learn 2D bounding boxes since it is robust to object scale changes.

3D Detection Similar to [7], the keypoint regression head predicts 10 keypoints for each object, from which we can extract five vertical lines. These lines are strictly perpendicular to the horizontal plane since the roll and pitch angle of annotated 3D bounding boxes are assumed to be zero in the dataset. The depth value of vertical line l is estimated by utilizing the relative proportion between pixel height and estimated object height [7], [36]

\[
z_l = f \times h \times h_1, \tag{2}\]

where f is the focal length. h_l and h are the height of line l in pixel and the 3D height of the object respectively. Following with [7], five

Table I
Detection Heads and the Corresponding Parameters to Be Estimated

| Head       | Task              | Target Paras. |
|------------|-------------------|--------------|
| Classification | Heatmap           | u_f, v_f     |
| Segmentation | Shape-Aware Feat. | -            |

Regression

| Depth       | Orientation       | 2D Bounding Box |
|-------------|-------------------|-----------------|
| 3D Dimensions | 2b, lb, lb, lb     |
| Center Offset | l_h, w_l   |
| Keypoints   | x_kpt           |
| Uncertainties | σ₁, σ₂, σ₃, σ₄   |
vertical lines are divided into three groups, and the depth value of each vertical line is averaged within each group, resulting in three independent center depth values. In addition, another depth value is also estimated directly by regression. The resulting 4 depth values are weighted by their uncertainties (given in Eqn. 3).

\[
z = \left( \frac{\sum_{i=1}^{4} z_i}{\sum_{i=1}^{4} \sigma_i} \right)
\]

where the uncertainty estimation branch is trained under the Laplacian uncertainty loss [23], [27], [28], [29].

When an object is detected on the heatmap at position \((u_f, v_f)\), its center \(x_c\) is given by \((s_0 u_f + \delta_x, s_0 v_f + \delta_y)\), where \(s_0\) is the downsampling factor. With the predicted projected 3D center \(x_c = (u_c, v_c, \bar{e}_c)\). The object location can be decoded as:

\[
(x, y, z) = \left( \frac{(u_c - c_u)z}{f}, \frac{(v_c - c_v)z}{f} \right).
\]

where \((c_u, c_v)\) is the principle point.

For dimension estimation, the relative changes with respect to the average dimension is predicted. For each class \(c\), the average dimension is denoted as \((\bar{h}_c, \bar{w}_c, \bar{e}_c)\), the L1 loss for dimension regress is defined as:

\[
L_{dim} = \sum_{k \in \{h, w, l\}} |\bar{k}_c e^{k^*} - k^*|,
\]

where \(k^*\) is the ground-truth dimension, \(\delta_k\) is the relative offset to be regressed. The dimensions are estimated by scaling the average dimensions, namely \((h, w, l) = (\bar{h}_c e^{\delta_h}, \bar{w}_c e^{\delta_w}, \bar{e}_c e^{\delta_h})\).

MultiBin loss [37] is used to estimate the local orientation \(\alpha\). The global orientation \(r_y\) can be obtained by calculating:

\[
r_y = \alpha + \arctan(x/z).
\]

The final 3D bounding box is then encoded as \((x, y, z, w, h, l, r_y)\). We refer readers to [7] for more details of the 3D detection head.

C. Implementation Details

Following [7], [10], [38], a modified DLA-34 [12] is adopted as our backbone. For each input image, the backbone produces a feature map with the down-sampling ratio 4. Different headers are responsible for estimating different parameters. The parameters to be estimated and the corresponding header are shown in Table I. Every regression head consists of one \(3 \times 3 \times 256\) convolution layer, BatchNorm [39], ReLU, and another \(1 \times 1 \times c_0\) convolution layer, where \(c_o\) is the output channels. For heatmaps prediction, we use the same structure except that there is a sigmoid function padded after the final Conv layer. For center offset and heatmap prediction, the edge fusion [7] module is applied after the \(3 \times 5\) Conv layer to decouple the feature learning of truncated objects.

For the instance segmentation head, a two-layer CNN block with a ReLU activation function is used to process the predefined position embedding map. The processed position embedding map is then added with extracted feature vectors with broadcasting. The resulting tensor is then fed into another two-layer CNN, which has a GroupNorm [40] layer with the group number set to 8 embedded before each ReLU. The final mask is obtained by further applying one convolution layer with a Sigmoid activation function. In order to reduce the number of parameters, \(1 \times 1\) convolution is used in this head, except at the last layer, where a \(3 \times 3\) convolution is used. The final loss of our model is a weighted sum of multiple task losses.

1 All the input images are padded into the same size of 384 × 1280.

IV. AVERAGE DEPTH SIMILARITY

In the literature, KITTI [42], [43] is one of the most popularly used evaluation benchmarks. The performance of monocular 3D object detection is measured by the 3D IoU-based Average Precision (AP) on the 3D space (AP3D) and the BEV (APBEV). A detection is considered positive if it overlaps a ground-truth with an IoU larger than a threshold. Such metrics are reasonable to evaluate LiDAR-based or stereo-based methods. However, we find they could not reflect the real performance of a monocular object detector.

First of all, as pointed out in [15], the depth estimation error increases with respect to the object’s distance from the camera. The detected objects that are far away from the camera are largely ignored due to low 3D IoU. Figure 5 further shows another two scenarios in which we cannot have an honest observation about the detector with 3D IoU-based evaluation. In Figure 5(a), we can see the evaluation favors object in vertical layout even both of them are estimated due to low 3D IoU. The AP3D score is boosted by simply duplicating and shifting several distant predictions. As we will see later in the experiment, AP3D can be boosted by 33.33% by such simple “result sampling”. Although this trick is effective in producing better AP3D, it is not-pragmatic since it neither improves the quality of the results nor enhances the ability of the model.

To address this issue, average depth similarity (ADS) is proposed as a complementary to AP3D. Compared to AP3D which is based on 3D-IoU, ADS matches the 2D predictions with the ground-truth.
The depth similarity is defined as

\[ s_d(r) = \frac{1}{|\mathcal{D}(r)|} \sum_{i \in \mathcal{D}(r)} \exp\left(-\left|\Delta_d^{(i)}\right|\right) \delta_i, \]  

where \( \mathcal{D}(r) \) are the set of all detected objects at recall rate \( r \) and \( \Delta_d^{(i)} \) is the difference of the depth of detection \( i \). \( \delta_i \) is 1 if detection \( i \) has been associated with a ground-truth object and 0 otherwise. The definition of ADS follows the original design of AOS [42], [43], while the difference lies in the similarity definition. The error in depth \( \Delta_d^{(i)} \in [0, +\infty) \) is normalized into the range of \([0, 1]\).

As illustrated in Figure 6, the calculation of depth similarity solely depends on the absolute depth error along the z-axis. Unlike the 3D IoU, the calculation of depth similarity is independent of object attributes such as size, orientation, and position. In addition, ADS avoids penalizing predictions with small IoUs. Therefore, it enables a comprehensive observation of the behavior of a monocular detector when combined with traditional evaluation metrics AP\(_3D\) and AP\(_{BEV}\).

Although ADS provides a less biased evaluation of the depth quality of the detected objects, the other attributes of the detected objects are not considered in this measurement. For this reason, AP\(_3D\) is still indispensable as it reflects the detection performance of the object’s location, size, and orientation. In a more objective evaluation, we prefer methods that achieve both high AP\(_3D\) and ADS.

V. EXPERIMENTS

Our method is evaluated on the popular KITTI dataset [42], [43] in comparison to state-of-the-art methods. The dataset consists of 7,481 training images and 7,518 testing images. Due to the restrictions on the test set submission to the KITTI official site, the training images are split into train (3,712) and val (3,769) sets following [44]. All our models jointly predict three categories, including Car, Pedestrian, and Cyclist. Following the official settings specified in the KITTI dataset [42], [43], the objects are divided into three difficulty levels, namely Easy, Moderate, and Hard according to the box height, occlusion level, and truncation level of objects.

The methods we consider in the experiment include one-stage center-based methods such as AutoShape [11], LPCG-MonoFlex [24], MonoCon [8], MonoFlex [7] and GUPNet [27]. The comparison also covers one-stage anchor-based methods such as Ground-Aware [17], DLE [18], and LPCG-M3D-RPN [24] and one-stage FCOS-like method DD3D [20]. A two-stage method MonoRoUn [23] is incorporated in our comparative study as well. In the evaluation, MonoFlex [7] that is retrained with the same settings as our method is treated as the comparison baseline. For clarity, this run is given as “MonoFlex\(^*\)”, while the run that is loyal to the original paper is given as “MonoFlex”.

Before we proceed with the comprehensive empirical study, the proposed evaluation measure, namely average depth similarity (ADS) is validated. A simple sampling [45] is conducted on the results produced by our method. Namely, each detected 3D bounding box is duplicated into several copies. These copies are placed to different depths to allow them to sufficiently overlap with the ground-truth. As shown in Figure 7, the AP\(_3D\) score of our method increases from 18.41 to 24.52, while without any literal improvement. This is where ADS comes to complement. As shown in the figure, the detection results after the sampling trick remain poor when it is measured by ADS. This is because the ADS penalizes the blind predictions. Duplicating the detected bounding boxes makes no contribution to the quality of 2D bounding boxes or depth predictions. To this end, it is clear to see ADS is complement to AP\(_3D\). In the following experiments, both AP\(_3D\) and ADS scores are reported on the val set for all the methods.

A. Ablation Study

In our first experiment, an ablation study is conducted to show the contributions of center sampling (CS) and shape-aware training (SA), both of which are introduced by us. In order to investigate the role that the uncertainty weighting plays, another run “SA\(^-\)” is pulled out, in which the uncertainty weighting \(\sigma_{seg}\) (in Eqn. 1) is replaced by a fixed weight in the loss function. The detection results are reported in Table II. As can be observed, when applying center sampling alone, 6.27% and 1.05% improvement in terms of AP\(_3D\) and ADS respectively are observed on the moderate objects. When the shape-aware auxiliary training head is further integrated, extra 9.59% and 3.11% respectively are observed. As shown in Table II, this improvement is consistent on the Pedestrian and Cyclist categories. We can see that “SA\(^-\)” produces considerably poorer detection results,
TABLE II

**ABLATION STUDY ON THE KITTI val SET FOR THREE CATEGORIES, i.e., THE CAR, PEDESTRIAN, AND CYCLIST. ABBREVIATIONS: CS REFERS TO CENTER SAMPLING, SA REFERS TO SHAPE-AWARE FEATURE LEARNING, AND SA- REFERS TO REMOVING THE UNCERTAINTY FROM THE LOSS FUNCTION OF SA.**

| Methods  | Easy | Moderate | Hard | Easy | Moderate | Hard | Easy | Moderate | Hard | Easy | Moderate | Hard |
|----------|------|----------|------|------|----------|------|------|----------|------|------|----------|------|
| MonoFlex* | 21.74 | 15.79 | 81.16 | 13.32 | 53.02 | 4.40 | 63.84 | 3.47 | 11.12 | 5.24 | 36.41 | 2.69 | 21.28 | 2.59 | 19.94 |
| + CS | 23.25 | 20.00 | 67.79 | 18.81 | 54.96 | 7.54 | 52.37 | 5.54 | 43.77 | 4.32 | 36.45 | 3.62 | 37.82 | 3.17 | 23.57 | 2.72 | 22.15 |
| + CS + SA | 23.45 | 70.37 | 67.01 | 14.82 | 55.51 | 8.26 | 52.41 | 6.49 | 43.85 | 5.29 | 37.53 | 5.75 | 40.52 | 3.34 | 25.14 | 2.79 | 23.54 |
| + CS + SA | 24.92 | 73.53 | 62.68 | 15.56 | 56.43 | 8.40 | 53.34 | 6.88 | 44.56 | 5.78 | 37.13 | 6.77 | 44.18 | 3.24 | 26.86 | 3.12 | 24.93 |

Fig. 9. Qualitative results on KITTI val set. The 1st column shows the detected 3D boxes along with the objects. The 2nd column shows the visible area of objects predicted by the segmentation head. On the 3rd column, the detection results are shown along with the ground-truth in the BEV. The detected boxes and the ground-truth are in red and blue respectively.

B. Performance Analysis

In this section, the performance of our shape-aware detector is studied on both the test and val set of KITTI in comparison to state-of-the-art methods. The 3D average precision $\text{AP}_{3D}$ and the average depth similarity ADS are reported for all the methods. Note that we only evaluate ADS on the val set since the ground-truth of the test set is not available. Evaluation metrics are divided into easy, moderate, and hard levels according to the height, occlusion, and truncation level of objects. All evaluation metrics use 40 recall points instead of 11 recall points as recommended in [47]. As shown in Table IV, our method in general outperforms most of the recent methods in the literature. In particular, our method outperforms MonoFlex considerably and ranks the 1st on the test set while it shares the same detection head as MonoFlex. It does indicate the effectiveness of the proposed shape-aware learning task. Moreover, our method shows consistently better accuracy in depth estimation than the rest of methods. As illustrated in Figure 8, the ADS has a

which implies that weighting the auxiliary training task according to its uncertainty is important.

In our second experiment, we further confirm the choice of feature expanding (illustrated in Figure 1(b)) as the feature extractor in the segmentation head. In this ablation study, the performance of using feature expanding is compared to the configuration that uses the RoIAlign [46] in the segmentation head. As shown in Table III, the performance of the detector drops when RoIAlign is used as the feature extractor in the segmentation head. In contrast, considerable improvement (in comparison to MonoFlex*) is observed when the RoIAlign is replaced with feature expanding in the feature extractor. As discussed earlier, the misalignment between the RoI-based and center-based representation leads to potential conflicts between two training heads. The segmentation becomes harmful to the detection task, which leads to even poorer performance than the case it is, otherwise, not integrated.

TABLE III

**COMPARISON BETWEEN DIFFERENT ROI FEATURE EXTRACTORS. EXPERIMENTS ARE CONDUCTED ON THE KITTI val SET.**

| Methods | Easy | Moderate | Hard | Easy | Moderate | Hard |
|---------|------|----------|------|------|----------|------|
| MonoFlex* | 21.74 | 15.79 | 81.16 | 69.59 | 60.16 | 53.02 |
| + RoIAlign | 19.97 | 14.65 | 12.91 | 69.19 | 59.79 | 52.86 |
| + Ours | 24.92 | 18.39 | 15.56 | 73.53 | 62.68 | 56.61 |
Comparisons for the Car Category on the KITTI Benchmark. For Models Marked With *, We Evaluate Their Performance on the \textit{val} Set by Training With Official Codes. For Other Models, Their Results on the \textit{val} Set Are Either Cited From Their Papers or by Evaluating the Pre-Trained Model From the Authors. Methods Are Ordered by Their \textit{AP}_{3D} Values Under the Moderate Setting. For Clarity, the Best Results Are in \textbf{RED}, While the Second Is Highlighted in \textit{BLUE}.

| Name                  | Extra Dataset | \textit{AP}_{3D} (val/val)          | \textit{AP}_{3D} (val/val)          | \textit{AP}_{3D} (val/val)          | \textit{AP}_{3D} (val/val)          |
|-----------------------|---------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|
| AutoShape [11]        | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| LPCG-M3D-RPN [24]     | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| DD3D [20]             | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| LPCG-MonoFlex [24]    | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| PGD [16]              | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| MonoDLE [15]          | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| MonoRoL [23]          | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| Ground-Aware* [17]    | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| CaDNN [25]            | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| MonoFlex [7]          | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| DLE* [18]             | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| GUPNet [21]           | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| MonoCon [8]           | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |
| Ours                  | Stay          | Easy: 53.55                       | Moderate: 52.80                   | Hard: 52.50                       | Easy: 53.55                       |

VI. Conclusion

We have presented a simple but effective auxiliary training task named shape-aware feature learning. It aims to improve the performance of a monocular 3D detector on the occluded objects that is designed for the autonomous driving system. With uncertainty weighted loss function, our method is able to learn from sparse and noisy segmentation labels, relieving of the laborious manual mask annotation. Considerable improvements are observed on the full-view objects as well as the occluded objects in particular. In addition, a metric called average depth similarity that is complementary to the current popular evaluation protocol is proposed to measure the performance of a monocular 3D detector. It allows a more comprehensive understanding about the monocular 3D object detection models. Currently, this detection framework is only feasible for single image view. It is our future research direction to integrate the segmentation branch to multiple-view 3D detection framework.

References

[1] B. Yang, W. Luo, and R. Urtasun, “PIXOR: Real-time 3D object detection from point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 7652–7660, doi: 10.1109/CVPR.2018.00798.
[2] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 4490–4499, doi: 10.1109/CVPR.2018.00472.
[3] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," Sensors, vol. 18, no. 10, p. 3337, Oct. 2018, doi: 10.3390/s18103337. [Online]. Available: https://www.mdpi.com/1424-8220/18/10/3337.
[4] P. Li, X. Chen, and S. Shen, "Stereo R-CNN based 3D object detection for autonomous driving," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 7636–7644, doi: 10.1109/CVPR.2019.00783.
[5] J. Sun et al., “Disp R-CNN: Stereo 3D object detection via shape prior guided instance disparity estimation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 10545–10554, doi: 10.1109/CVPR42600.2020.01056.
[6] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 3289–3298, doi: 10.1109/CVPR46437.2021.00330.
