# 3D Object Detection from Consecutive Monocular Images

Chia-Chun Cheng[1][0000−0002−0706−2808] and
Shang-Hong Lai[1,2][0000−0002−5092−993X]

[1] National Tsing Hua University, Hsinchu, Taiwan
ms0365647@gmail.com, lai@cs.nthu.edu.tw
[2] Microsoft AI R&D Center, Taipei, Taiwan
shlai@microsoft.com

**Abstract.** Detecting objects in 3D space plays an important role in scene understanding for real applications, such as autonomous driving and mobile robot navigation. Many image-based methods have been proposed due to the high cost of LiDAR. However, monocular images are lack of depth information and difficult to detect objects with occlusion. In this paper, we propose to integrate 2D/3D object detection and 3D motion estimation for consecutive monocular images to overcome these problems. Additionally, we estimate the relative motion of the object between frames to reconstruct the scene in the previous timestamp. To learn motion estimation from unlabeled data, we propose an unsupervised motion loss which learns 3D motion estimation from consecutive images. Our experiments on KITTI dataset show that the proposed method outperforms the state-of-the-art methods for 3D Pedestrian and Cyclist detection and achieves competitive results for 3D Car detection.

## 1 Introduction

Detecting objects and their motions in 3D space plays an important role in scene understanding for applications such as autonomous driving[1] and mobile robot navigation[2]. Previous methods[3] on 3D object detection rely heavily on LiDAR device which provides precise depth information. However, due to the high cost of LiDAR, many monocular image-based approaches were proposed. Specifically, image-based approaches can be categorized into image-only[4] and pseudo-LiDAR methods[5]. The image-only methods typically leverage geometric constraints such as object shape and key points. Pseudo-LiDAR approaches relied on external sub-networks such as state-of-the-art depth prediction network to obtain depth cues. The drawback of these approaches is the upper-bound for the framework limited by the additional network due to the persistent output noise. On the other hand, unsupervised depth prediction methods[6] using sequential monocular frames have achieved state-of-the-art performance compared to the supervised approaches, which makes it possible to solve the core problem in 3D object detection for the lack of depth information. Furthermore, due to the

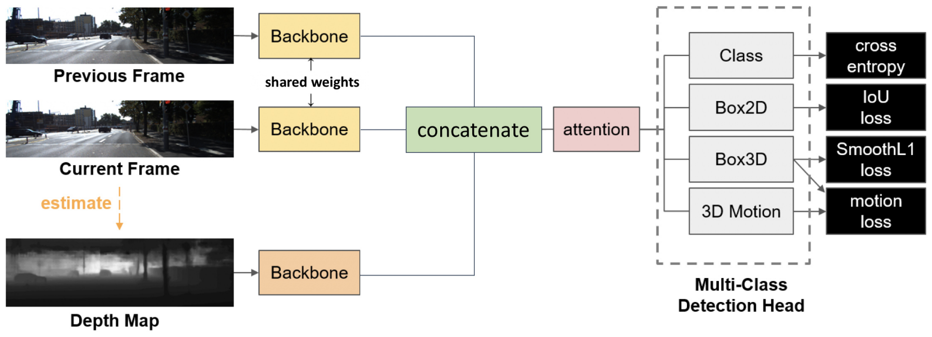

**Fig. 1.** The overall architecture of the proposed network that integrates 2D/3D object detection and 3D object motion estimation for multi-frame monocular images. The outputs are for current frame.

single view observation, monocular image-based approaches typically fail to detect objects with occlusion and truncation, which may occur in real-world driving scenarios. Therefore, the lack of depth information and occlusion cause large performance gap between LiDAR-based methods and monocular approaches, thus making 3D object detection from monocular images still a challenging problem.

The 3D object detection aims to classify the object category and estimate 3D bounding boxes of physical objects. Specifically, for each detected 3D bounding box, 3D centroid coordinates (x, y, z), orientation (rotation y) and sizes (l,h,w) are given according to the camera coordinate system. Particularly, monocular image-based approaches only take monocular frames as input instead of 3D point cloud data, and the calibration parameters for each image are usually provided.

In this work, we focus on addressing two key challenges in monocular 3D object detection. First, depth prediction is considerably the most challenging problem in monocular image-only 3D object detection. Previous approaches typically use projection geometric constraint[7], prior knowledge[4][8] or external pre-trained depth estimation network[9] to solve this problem. In addition, occlusion or truncation often occurs in driving scenes, especially in the scenarios with lots of cars on the road. Thus, an accurate 3D object detector is highly demanded for real-world autonomous driving applications.

Motivating by [6], which learns depth prediction from unlabeled monocular videos, we propose another way to address the above problems by using sequential frames from a single monocular camera. Specifically, we use consecutive frames to detect objects in 3D space. To utilize consecutive frames, we not only predict 3D bounding boxes, but also their 3D relative motions to reconstruct their 3D positions in the previous timestamp. Thus, we can recover depth cues from multi-view geometric constraints. Overall, we propose a novel three-streamed convolutional neural network based on [4], which takes consecutive frames and an estimated depth map as inputs and additionally estimates

3D relative motion for each object. The overall architecture of the proposed integrated network is depicted in figure 1.

To better capture object motion between consecutive frames, we adopt channel attention mechanism[10] in our proposed network. Due to the lack of labels in previous frames, unsupervised training is adopted to learn 3D object motion. Notably, we propose a motion loss which projects the 3D object onto the current frame and the previous frame according to the estimated depth and motion, and then calculate the absolute error and structural similarity error[11] between two projected patches.

The main contributions of this work include: (1) We propose a three-streamed network which combines the local features and motion features to address the depth estimation problem in 3D object detection. (2) Channel attention is applied to capture motion features between consecutive frames. (3) Moreover, we additionally estimate the relative 3D motions for objects in 3D space to help us better understand the surroundings in real-world driving scenes. (4) To estimate 3D motions, we propose a unsupervised motion loss which projects the 3D object onto both current and previous frames, and then computes the structure similarity error. In the experimental evaluation, we show that our method outperforms the state-of-the-art methods in both 3D Pedestrian and Cyclist detection and achieves competitive results for 3D Car detection on the KITTI[12] dataset.

## 2 Related Work

### 2.1 LiDAR-based 3D Object Detection

Typically, LiDAR-based 3D object detection can be categorized into point-based and voxel-based according to the feature representations. Zhou *et al.* [13] divided point clouds into equally spaced 3D voxels and transform a group of points within each voxel into a unified feature representation. In [14], Wang *et al.* implemented FPN technique on voxel-based networks. On the other hand, PointNets[15] can directly extract features from raw point clouds, which leads to many point-based approaches being proposed. Qi *et al.* [16] leveraged mature 2D object detectors and extracted point clouds from 2D detection frustum. In [3], Shi *et al.* proposed to segment point clouds into foreground and background and generate high-quality 3D proposals. The drawbacks of LiDAR are its high cost and sensitivity to adverse weather conditions, which limit its application to real-world scenarios.

### 2.2 Pseudo LiDAR-based 3D Object Detection

Ma *et al.* [17] first transformed the input data from 2D image plane to 3D point clouds space for a better input representation and embedded the complementary RGB cue into the generated point clouds representation. In [18], Wang *et al.* converted image-based depth maps to pseudo-LiDAR, which can apply different existing LiDAR-based detection algorithms. Following the pipeline of two-stage 3D detection algorithms, Weng *et al.* [19] detected 2D object proposals in the input image and extracted a point cloud frustum from the pseudo-LiDAR for each proposal. However, pseudo-LiDAR relies heavily on the additional predicted depth map, which limits the upper bound of the performance.

### 2.3 Image-based Monocular 3D Object Detection

Liu *et al.* [20] predicted a 3D bounding box for each detected object by combining a single keypoint estimate with regressed 3D variables and proposed a multi-step disentangling approach for constructing the 3D bounding box, which significantly improves both training convergence and detection accuracy. Qin *et al.* [21] proposed a unified network composed of four task-specific subnetworks, responsible for 2D object detection, instance depth estimation (IDE), 3D localization and local corner regression. In [4], Brazil *et al.* proposed 2D-3D anchor architecture for region proposal network, which initializes all anchors with prior statistics for each of its 3D parameters. Additionally, depth-aware convolutional layers are proposed to better capture depth information. To increase 2D-3D consistency and achieve better orientation prediction, a post-optimization algorithm is applied after region proposal network. Ding *et al.* [9] indicated that conventional 2D convolutions are unsuitable for 3D object detection due to the failed capture of local object and its scale information. Therefore, a new local convolutional network (LCN), termed Depth-guided Dynamic-Depthwise-Dilated LCN, was proposed, where the filters and their receptive fields can be automatically learned from image-based depth maps, making different pixels of different images have different filters. However, single 2D image can not fully represent the 3D structure of each object and fail to restore the accurate 3D information.

In this paper, we use consecutive monocular images to recover depth information through multi-view geometric constraints. To reconstruct the previous scene, we not only predict 3D bounding box for each object but also predict its 3D relative motion. In addition, we use channel attention mechanism to capture motion information between consecutive frames. Notably, we do not need extra labels to train the estimation of relative object motions. The motion is trained through unsupervised learning.

## 3 Proposed Method

As a multi-class 3D object detector, the proposed network is comprised of three key components: a backbone, an attention module and a multi-class detection head. Following [9], we apply DORN[22] to generate our depth maps. Fig. 1 illustrates the overall architecture of the proposed network. First, our network takes two consecutive RGB frames and an estimated depth map of the current frame as inputs. Then, the features are concatenated along the channel dimension after the backbone feature extraction. The channel attention mechanism is applied after the feature concatenation. Last, the attention module is followed by a multi-class detection head. Details of the three components are described below.

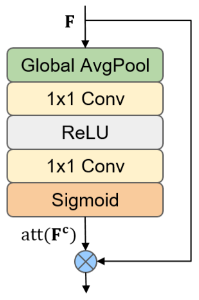

**Fig. 2.** Illustration of channel attention block

## 3.1 Backbone

To better utilize previous frames and depth maps, we design the backbone with a three-branch network. The first two branches are the RGB images feature extraction networks with shared weights for the previous frame and current frame. The third branch is the depth feature extraction network which takes the estimated depth map of the current image as input. After the feature extraction, these three feature maps are merged via concatenating along the channel dimension. Inspired by [9], the backbone of the feature extraction network is ResNet-50[23] without the FC and average pooling layers. The all subsequent convolutional layers in block4 are replaced by dilated convolutional layers (the dilation rate is 2) to obtain a larger receptive field.

## 3.2 Attention Module

Attention mechanism helps the neural network to highlight distinctive features. Consecutive frames contain redundant information such as background in both spatial and temporal dimensions, which should not be treated as equally important in our detection network. In addition, attention mechanism allows our network to focus on object motions and bring out better depth reconstruction effects. Thus, we adopt the channel attention[10] mechanism to predict relative object motion and aggregate more robust features in consecutive frames. The output of the attention module can be considered as the weighted sum of the feature maps along the channel dimension.

The backbone features are followed by three residual groups[10], each of which consists of 12 residual attention blocks, having totally 36 channel attention blocks. Fig. 2 illustrates the architecture of the channel attention block. First, global average pooling is applied on the input feature map to aggregate the statistics of each channel. Then, we used two 1x1 convolutional layers to capture the inter-channel relationship. To make sure the weights are in the range between 0 and 1, sigmoid is applied. The output of channel attention block is calculated as an element-wise product of the input feature $\mathbf{F}$ and att($\mathbf{F^c}$) along the channel dimension.

## 3.3 Multi-Class Detection Head

Inspired by [9, 4], the proposed framework is based on the region proposal network (RPN) with shared 2D-3D anchor boxes. The region proposal network locates the predefined anchors at every spatial location of an image and generates proposals where the object matches the templates of the predefined anchors. Then, the parameters of the estimated objects are regressed from the generated proposals.

**Formulation:** The input feature maps of our detection head has a network stride factor of 16 to the input image. Following the practice, the 3D-to-2D projection matrix for each image is given both at training and test time. The projection matrix $\mathbf{P} \in \mathbb{R}^{3x4}$ can be written as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_P \cdot z_{3D} = \mathbf{P} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{3D} , \tag{1}$$

where $[x, y, z]_{3D}$ denotes the 3D point in camera coordinates, and $[x, y]_P$ denotes the projected 2D point of the 3D point in image coordinates.

Let us denote $n_a$ to be the number of anchors, $n_c$ the number of classes, $h$ and $w$ the height and width of the input feature maps, respectively. The output represents the anchor-based transformation. Our detection head predicts $n_c$+14 parameters per anchor for each position $(i, j)$ in the input feature maps as: $\mathbf{c}, [t_x, t_y, t_w, t_h]_{2D}, [t_x, t_y]_P,$
$[t_w, t_h, t_l, t_z, t_\alpha]_{3D}, [t_x, t_y, t_z]_{motion}$ , where $\mathbf{c}$ denotes the confidence score of each class, and $[t_x, t_y, t_w, t_h]_{2D}$ is the estimated 2D box, $[t_x, t_y]_P$ denotes the projected 2D point of the 3D object center, $[t_w, t_h, t_l, t_z, t_\alpha]_{3D}$ denotes the estimated 3D shape, depth and rotation, $[t_x, t_y, t_z]_{motion}$ denotes the 3D relative motion of the object between two frames in camera coordinates. The total size of the output is $w \times h \times n_a \times (14 + n_c)$.

**2D-3D Anchor:** We utilize our 2D anchors with 3D parameters. Specifically, the 2D-3D anchors are defined on the 2D space. For each anchor, the default values of the 3D parameters are the mean statistics calculated from the training dataset as the corresponding priors. A template anchor is defined using parameters of both 2D and 3D spaces: $[x, y, w, h]_{2D}, [x, y]_P, [w, h, l, z, \alpha]_{3D}$, where $[x, y, w, h]_{2D}$ denotes the 2D box, $[x, y]_P$ denotes the projected 3D center in image coordinates, $[w, h, l]_{3D}$ denotes the shape of the 3D box, $z_{3D}$ denotes the depth, and $\alpha_{3D}$ denotes the observation viewing angle.

For 2D anchors, we use 12 anchor scales ranging from 30 to 400 pixels and 3 ratios (0.5, 1.0, 1.5) to define our total 36 anchors for each cell in the output feature maps. Note that $[x, y]_{2D}$ and $[x, y]_P$ share the same anchor parameters. To calculate the default priors for the 3D parameters, we project all 3D ground truth boxes to the 2D space. Then, for each projected box, we assign it to the 2D anchors whose intersection over union (IoU) with it are greater than 0.5. Afterwards, the 3D parameters $[w, h, l, z, \alpha]_{3D}$ are the statistics across all matching 3D ground truth boxes.

**Motion Scale Parameter:** For the relative object motion, since the 2D convolution is a spatially-invariant operation and does not consider the depth dimension, it is difficult for convolutional kernels to predict 3D object motions between frames. On the other hand, due to the perspective projection, the moving distance for an object in the monocular view would be different.

To address these problems, we predefine our motion scale parameter $s_{motion}$ for each anchor based on its depth prior. Specifically, the motion scale parameter is applied to the estimated relative object motion to obtain the final object motion in the 3D space. The transformation is further detailed in Data Transformation. Generally, the anchor with the larger depth prior has larger motion scale parameter. To calculate the motion scale parameter, we first define two fixed points in image coordinates and back-project these two points from image coordinates to camera coordinates based on the depth prior of each anchor. Then, we calculate the distance between the two points in the 3D space. We thus define a motion scale parameter for each anchor based on the ratio of its distance difference in the 3D space to the smallest one among all anchors.

**Data Transformation:** The output of our network represents the anchor-based transformation. Following [4], the data transformation is applied to obtain the final results:

$$\begin{aligned}
&[x', y']_{2D} = [x, y]_{2D} + [t_x, t_y]_{2D} \cdot [w, h]_{2D} \\
&[w', h']_{2D} = [w, h]_{2D} \cdot exp([t_w, t_h]_{2D}) \\
&[x', y']_{P} = [x, y]_{P} + [t_x, t_y]_{P} \cdot [w, h]_{2D} \\
&[w', h', l']_{3D} = [w, h, l]_{3D} \cdot exp([t_w, t_h, t_l]_{3D}) \\
&[z', \alpha']_{3D} = [z, \alpha]_{3D} + [t_z, t_\alpha]_{3D} \\
&[x', y', z']_{motion} = [s_{motion}, 1, s_{motion}] \cdot [t_x, t_y, t_z]_{motion},
\end{aligned} \tag{2}$$

where $[*']$ denotes the transformed parameter, $[*]$ denotes the parameter of the predefined anchor, $s_{motion}$ denotes the motion scale parameter, $[t_*]$ is the output of our network. Notably, since the difference in vertical motion of an object is usually small, the motion scale parameter is applied to the horizontal and depth axes in camera coordinates.

## 3.4 Loss Function

The loss of our network is formed as a multi-task learning task, which contains a classification loss $L_{class}$, a 2D box regression loss $L_{2D}$, a 3D box regression loss $L_{3D}$ and a relative object motion regression loss $L_{motion}$. Let $b'_{2D}$ denote

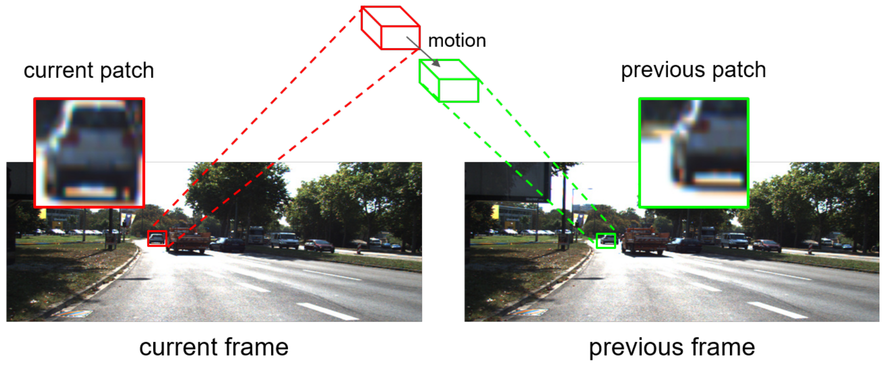

**Fig. 3.** Illustration of motion loss

$[x', y', w', h']_{2D}$, $b'_{3D}$ denote $[x', y']_P$ and $[w', h', l', z', \alpha']_{3D}$. Following [24], to generate the ground truth target $\hat{b}_{2D}, \hat{b}_{3D}$ for each anchor, we check if any ground truth 2D box matches the predefined 2D anchors with at least 0.5 IoU. We thus define the target of the anchor with the best matching ground truth.
For the classification loss, we employ the cross-entropy loss:

$$L_{class} = -log(softmax(\mathbf{c})). \tag{3}$$

For 2D box, we use the negative logistic IoU loss for transformed $b'_{2D}$:

$$L_{2D} = -log(IoU(b'_{2D}, \hat{b}_{2D})). \tag{4}$$

We use Smooth L1 loss for each transformed parameter in $b'_{3D}$:

$$L_{3D} = SmoothL1(b'_{3D}, \hat{b}_{3D}). \tag{5}$$

For the target of relative 3D object motion, due to the lack of labels, we adopt an unsupervised learning mechanism to learn 3D object relative motion in camera coordinates. As shown in Fig. 3, for each foreground anchor, we project the target 3D bounding box onto the image in the current timestamp to obtain a 2D image patch $\hat{patch}_{curr}$. Then, we shift the estimated 3D box with estimated 3D relative motion to reconstruct the object location in the previous timestamp, and obtain another 2D patch $patch'_{prev}$ through 3D-to-2D projection. We finally calculate the SSIM loss and the L1 loss between the two patches to supervise the motion learning.
The motion loss $L_{motion}$ is denoted as:

$$L_{motion} = SSIM(\hat{patch}_{curr}, patch'_{prev}) + L1(\hat{patch}_{curr}, patch'_{prev}). \tag{6}$$

Overall, the weighted loss of our multi-task network is defined as:

$$L = \lambda_1 L_{class} + \lambda_2 L_{2D} + L_{3D} + \lambda_3 L_{motion}. \tag{7}$$

## 4 Experimental Results

### 4.1 Dataset

**KITTI:** The KITTI[12] dataset is a widely used autonomous driving dataset, which provides the 3D object detection and the bird's eye view (BEV) benchmarks for 3D localization evaluation. In BEV task evaluation, all 3D boxes are projected to the ground plane, and 2D object detection is applied to compute the evaluation results. The official dataset consists of 7,481 training samples and 7,518 test samples, including images, corresponding point clouds, 3 temporally preceding frames, camera calibration matrices, and 2D-3D annotations for three object classes: Car, Pedestrian, and Cyclist. For each object in annotations, one of the three difficulty levels (easy, moderate, hard) is given based on the occlusion and truncation levels of the object.

Since the annotations of the official test dataset are not available, we split the training dataset into a *training* set and a *validation* set described in [25] for comprehensive evaluation. The split renders 3,712 *training* and 3,769 *validation* samples, and avoids overlapping sequences in both sets.

### 4.2 Evaluation Metrics

Following [12], for each difficulty level and class, we compute the precision-recall curve and average precision (AP) for evaluation with IoU threshold 0.7 (Car) and 0.5 (Pedestrian, Cyclist). Notably, all methods on the official KITTI leaderboard are ranked based on the moderately difficult results. Before Aug. 2019, 11 recall points are applied to compute interpolated average precision, denotes as $\text{AP}|_{R_{11}}$. After [26] released, the official evaluation uses 40 recall points instead of 11 recall points in the benchmark, denoted as $\text{AP}|_{R_{40}}$. Particularly, since the early methods only report the results under old metric, we compare our results on *validation* set using $\text{AP}|_{R_{11}}$ for fair comparison.

### 4.3 Implementation Details

In our motion loss, we generate two projection patches in the previous frame to supervise both motion and depth. Particularly, to better supervise motion, we use the ground truth 3D box $\hat{b}_{3D}$ and the estimated motion $[x', y', z']_{motion}$ to obtain the patch in the previous frame. On the other hand, to supervise both depth and motion, we replace the depth parameter in $\hat{b}_{3D}$ with estimated depth $z'_{3D}$ and obtain another previous patch with the estimated motion $[x', y', z']_{motion}$. Both patches are used to compute SSIM and L1 loss with the current patch. All patches are scaled to 32×32. We set $\lambda_1 = \lambda_2 = 1$, $\lambda_3 = 0.05$.

A dropout layer with a dropout rate of 0.2 is applied after the channel attention module. We adopt non-maximum suppression with an IoU threshold of 0.4 on our output in 2D space. Horizontal flipping is used for data augmentation. The input images are scaled to 512×1760. Following [4], a hill climbing post-processing is applied for optimizing the estimated rotation $\alpha'_{3D}$.

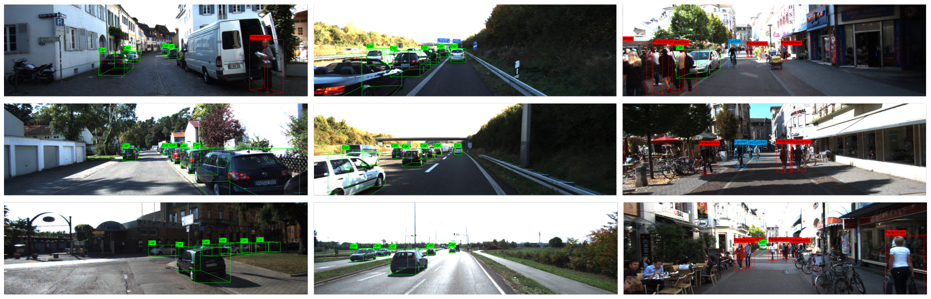

**Fig. 4.** Qualitative Examples

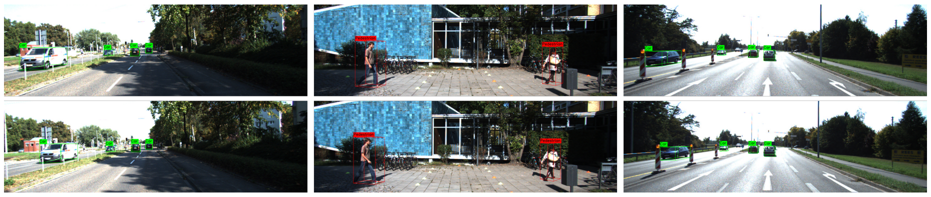

**Fig. 5.** Visualization results of reconstructed object positions in previous frames.

The model is trained with SGD optimizer with a learning rate 0.01, a momentum 0.9, a weight decay 0.0005, and mini-batches of size 8. We adopt the 'one-cycle' learning policy[27] with a maximum learning rate 0.01 and a minimum learning rate 0.000001. The model has 263,283,080 trainable parameters and the inference time is 0.4s. We train our network on 4 NVIDIA Tesla V100 GPUs for 50,000 iterations.

## 4.4 Experimental Results

In this section, we present evaluation results on the KITTI *validation* set[25]. We compare our results with existing monocular methods in 3D object detection and bird's eye view (BEV) benchmarks. For fair comparison, all results are reported in $\mathrm{AP}|_{R_{11}}$. We visualize our qualitative examples in Fig. 4. In Fig. 5, the second row shows the scene in the previous timestamp corresponding to the first row. Since we only predict the results in the current timestamp, the reconstructed results from the estimated object motions in the previous timestamp show that our model can estimate the object motion.

**Car:** Table 1 and Table 2 show the comparative results in Car at IoU threshold 0.7 and 0.5, respectively. Our methods achieve competitive results both in $\mathrm{AP}_{3D}$ and $\mathrm{AP}_{BEV}$. As mentioned in [9], pseudo-LiDAR based methods [19, 17] fail to detect other classes in their single model, while our model is a multi-class detector. In addition, [19, 17] used the additional 2D detector to obtain the better 2D results in their model and resort to multi-stage training. However, our

**Table 1.** Comparative results on *validation* set at 0.7 IoU threshold for Car.

| Method | AP$_{3D}$ | | | AP$_{BEV}$ | | |
|---|---|---|---|---|---|---|
| | Easy | **Moderate** | Hard | Easy | **Moderate** | Hard |
| Shift R-CNN[28] | 13.84 | 11.29 | 11.08 | 18.61 | 14.71 | 13.57 |
| MonoGRNet[21] | 13.88 | 10.19 | 7.62 | 24.97 | 19.44 | 16.30 |
| MonoPSR[29] | 12.75 | 11.48 | 8.59 | 20.63 | 18.67 | 14.45 |
| Multi-Fusion[30] | 10.53 | 5.69 | 5.39 | 22.03 | 13.63 | 11.60 |
| Mono3D-PLiDAR[19] | 31.50 | 21.00 | 17.50 | 41.90 | 28.30 | **24.50** |
| AM3D[17] | **32.23** | 21.09 | 17.26 | **43.75** | **28.39** | 23.87 |
| MonoDIS[26] | 18.05 | 14.98 | 13.42 | 24.26 | 18.43 | 16.95 |
| M3D-RPN[4] | 20.27 | 17.06 | 15.21 | 25.94 | 21.18 | 17.90 |
| SMOKE[20] | 14.76 | 12.85 | 11.50 | 19.99 | 15.61 | 15.28 |
| D4LCN[9] | 26.97 | **21.71** | **18.22** | 34.82 | 25.83 | 23.53 |
| **Ours** | 25.49 | 18.86 | 17.15 | 33.69 | 25.02 | 20.38 |

**Table 2.** Comparative results on *validation* set at 0.5 IoU threshold for Car.

| Method | AP$_{3D}$ | | | AP$_{BEV}$ | | |
|---|---|---|---|---|---|---|
| | Easy | **Moderate** | Hard | Easy | **Moderate** | Hard |
| MonoGRNet[21] | 50.51 | 36.97 | 30.82 | 54.21 | 39.69 | 33.06 |
| MonoPSR[29] | 49.65 | 41.71 | 29.95 | 56.97 | 43.39 | 36.00 |
| Multi-Fusion[30] | 47.88 | 29.48 | 26.44 | 55.02 | 36.73 | 31.27 |
| Mono3D-PLiDAR[19] | 68.40 | 48.30 | **43.00** | 72.10 | **53.10** | **44.60** |
| AM3D[17] | **68.86** | **49.19** | 42.24 | **72.64** | 51.82 | 44.21 |
| M3D-RPN[4] | 48.96 | 39.57 | 33.01 | 55.37 | 42.49 | 35.29 |
| **Ours** | 59.00 | 44.29 | 36.72 | 66.38 | 46.32 | 38.48 |

multi-task model is trained end-to-end. The 2D detection results are shown in Table 3.

**Pedestrian & Cyclist:** For Pedestrian and Cyclist, our method achieves the state-of-the-art results in both 3D object detection and bird's eye view benchmarks (Table 4 and Table 5). Notably, due to the non-rigid body and small size, pedestrians and cyclists are particularly hard to detect in 3D space. Most methods fail to detect these two classes in their models. However, our proposed multi-class 3D detection model outperforms all previous methods with a large margin. Particularly, we achieve 10.46% relative improvement (12.46 vs. 11.23) in Pedestrian AP$_{3D}$ under the moderate setting.

### 4.5 Ablations

To verify the effect of each component in our model, we conduct ablation study on *validation* set. We only modify the target components and keep other settings

**Table 3.** Comparative 2D detection results on *validation* set at 0.7 IoU threshold for Car.

| Method | AP$_{2D}$ | | |
|---|---|---|---|
| | Easy | **Moderate** | Hard |
| AM3D[17] | 90.50 | **89.90** | 80.70 |
| Mono3D-PLiDAR[19] | **96.50** | **90.30** | **87.60** |
| **Ours** | 87.32 | 82.42 | 66.51 |

**Table 4.** Comparative results on *validation* set at 0.5 IoU threshold for Pedestrian.

| Method | AP$_{3D}$ | | | AP$_{BEV}$ | | |
|---|---|---|---|---|---|---|
| | Easy | **Moderate** | Hard | Easy | **Moderate** | Hard |
| Shift R-CNN[28] | 7.55 | 6.80 | 6.12 | 8.24 | 7.50 | 6.73 |
| MonoPSR[29] | 10.64 | 8.18 | 7.18 | 11.68 | 10.05 | 8.14 |
| Mono3D-PLiDAR[19] | 11.60 | 11.20 | 10.90 | 14.40 | **13.80** | 12.00 |
| AM3D[17] | 11.29 | 9.01 | 7.04 | 14.30 | 11.26 | 9.23 |
| MonoDIS[26] | 10.79 | 10.39 | 9.22 | 11.04 | 10.94 | 10.59 |
| M3D-RPN[4] | - | 11.28 | - | - | 11.44 | - |
| D4LCN[9] | 12.95 | 11.23 | 11.05 | - | - | - |
| **Ours** | **14.19** | **12.46** | **11.82** | **14.98** | 12.73 | **12.40** |

in our study. We evaluate the performance in both AP$_{3D}$ and AP$_{BEV}$ using AP$|_{R_{11}}$.

**Effect of Multi-frame Input:** In Table 6, we compare models trained with the different numbers of input frames. The model trained with 2 frames and 3 frames which additionally use one and two preceding frames as inputs, respectively. The model with 2-frame input outperforms others in Car and Pedestrian. In Cyclist, 3-frame input achieves the best. We present the contribution of consecutive frames in monocular 3D object detection task. Particularly, for hard difficulty, we observe that the performance increases when using more frames as input, which shows the effect of preceding frames on highly occluded objects as well.

**Effect of Motion Loss:** To better utilize two consecutive frames, we add motion loss $L_{motion}$ to train our model. Therefore, we ablate $L_{motion}$ in Table 7 to observe the contribution in our model. We show that $L_{motion}$ improves the performance with large margin especially for Pedestrian and Cyclist. Since the movement of pedestrians and cyclists is much smaller than cars, it is easier to predict their motions and achieve large improvement.

**Table 5.** Comparative results on *validation* set at 0.5 IoU threshold for Cyclist.

| Method | $AP_{3D}$ | | | $AP_{BEV}$ | | |
|---|---|---|---|---|---|---|
| | Easy | **Moderate** | Hard | Easy | **Moderate** | Hard |
| Shift R-CNN[28] | 1.85 | 1.08 | 1.10 | 2.30 | 2.00 | 2.11 |
| MonoPSR[29] | 10.88 | 9.93 | 9.93 | 11.18 | 10.18 | 10.03 |
| Mono3D-PLiDAR[19] | 8.50 | 6.50 | 6.50 | 11.00 | 7.70 | 6.80 |
| AM3D[17] | 8.90 | 4.81 | 4.52 | 10.12 | 6.39 | 5.63 |
| MonoDIS[26] | 5.27 | 4.55 | 4.55 | 5.52 | 4.66 | 4.55 |
| M3D-RPN[4] | - | 10.01 | - | - | 10.13 | - |
| D4LCN[9] | 5.85 | 4.41 | 4.14 | - | - | - |
| **Ours** | **10.94** | **10.39** | **10.43** | **12.31** | **10.76** | **10.76** |

**Table 6.** Comparisons of different number of frames as input on *validation* set

| Class | Input | $AP_{3D}$ | | | $AP_{BEV}$ | | |
|---|---|---|---|---|---|---|---|
| | | Easy | **Moderate** | Hard | Easy | **Moderate** | Hard |
| Car | 1-frame | 24.09 | 18.05 | 15.08 | 31.23 | 23.36 | 18.95 |
| | 2-frame | **25.49** | **18.86** | **17.15** | **33.69** | **25.02** | **20.38** |
| | 3-frame | 23.88 | 18.05 | 16.46 | 30.82 | 23.73 | 19.30 |
| Pedestrian | 1-frame | 7.15 | 5.84 | 5.28 | 8.01 | 6.29 | 6.22 |
| | 2-frame | **14.19** | **12.46** | **11.82** | **14.98** | **12.72** | **12.40** |
| | 3-frame | 13.14 | 11.51 | 11.12 | 13.46 | 11.73 | 11.48 |
| Cyclist | 1-frame | 4.06 | 3.52 | 3.60 | 5.71 | 3.96 | 3.86 |
| | 2-frame | 10.84 | 10.39 | 10.43 | 12.31 | 10.76 | **10.76** |
| | 3-frame | **12.50** | **10.44** | **10.46** | **12.87** | **10.83** | 10.74 |

**Effect of Attention Module:** In this part, we compare the model trained with the different numbers of attention groups in Table 8. We report the results at moderate difficulty level for better viewing. We observe that the performance of Car detection in both $AP_{3D}$ and $AP_{BEV}$ becomes larger as the number of attention groups increases . Notably, due to the small shapes of pedestrians and cyclists, it is hard to detect them in 3D space and is more sensitive to the hyperparameters. Overall, all categories achieve the best performances under 3 attention groups.

**Effect of Patch Size:** Table 9 shows the effect of different patch sizes used in motion loss. We observe that the performance of Pedestrian detection and Cyclist detection becomes better as the patch size increases. All categories achieve the best performance when patch size is 32×32.

**Table 7.** Comparisons of the effect of motion loss $L_{motion}$ on *validation* set

| Class | $L_{motion}$ | AP$_{3D}$ | | | AP$_{BEV}$ | | |
|---|---|---|---|---|---|---|---|
| | | Easy | **Moderate** | Hard | Easy | **Moderate** | Hard |
| Car | ✗ | 23.93 | 18.13 | 15.87 | 32.10 | 23.92 | 19.58 |
| | ✓ | **25.49** | **18.86** | **17.15** | **33.69** | **25.02** | **20.38** |
| Pedestrian | ✗ | 6.86 | 6.06 | 5.51 | 7.61 | 5.97 | 6.02 |
| | ✓ | **14.19** | **12.46** | **11.82** | **14.98** | **12.72** | **12.40** |
| Cyclist | ✗ | 6.83 | 5.97 | 6.02 | 9.25 | 6.68 | 6.39 |
| | ✓ | **10.84** | **10.39** | **10.43** | **12.31** | **10.76** | **10.76** |

**Table 8.** Comparisons of the number of attention groups in the moderate difficulty on *validation* set

| Num of Attention Groups | AP$_{3D}$ | | | AP$_{BEV}$ | | |
|---|---|---|---|---|---|---|
| | Car | Pedestrian | Cyclist | Car | Pedestrian | Cyclist |
| 0 | 17.66 | 10.99 | 10.15 | 23.31 | 11.36 | 10.23 |
| 1 | 18.11 | 5.81 | 5.49 | 23.75 | 11.96 | 5.65 |
| 2 | 18.54 | 11.33 | 3.75 | 24.21 | 12.09 | 4.09 |
| 3 | **18.86** | **12.46** | **10.39** | **25.02** | **12.72** | **10.76** |

## 5 Conclusions

In this paper, we proposed a three-streamed network which additionally estimates relative object motions to recover depth cues for 3D object detection from monocular images. We observe that depth prediction and occlusion are the most important issues in monocular-based methods. Thus, we take consecutive monocular images as inputs to overcome the above problems. To better utilize consecutive frames, we proposed an unsupervised motion loss and applied the attention mechanism. Our model is a multi-class detector which is more robust in real-world scenes than existing pseudo LiDAR-based methods. Additionally, our model also estimates the relative 3D motions for the detected objects, which provides more information for the surrounding environment in driving scenarios. Our experiments show that the proposed model outperforms existing methods with a large margin for Pedestrian and Cyclist detection and achieves competitive results for Car detection on the KITTI dataset.

**Table 9.** Comparisons of patch sizes in the moderate difficulty on *validation* set

| Patch Size | AP$_{3D}$ | | | AP$_{BEV}$ | | |
|---|---|---|---|---|---|---|
| | Car | Pedestrian | Cyclist | Car | Pedestrian | Cyclist |
| 8x8 | 18.80 | 5.16 | 3.82 | 24.81 | 6.27 | 4.64 |
| 16x16 | 18.61 | 5.43 | 4.71 | 24.80 | 7.47 | 5.11 |
| **32x32** | **18.86** | **12.46** | **10.39** | **25.02** | **12.72** | **10.76** |

## References

1. Behl, A., Hosseini Jafari, O., Karthik Mustikovela, S., Abu Alhaija, H., Rother, C., Geiger, A.: Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 2574–2583
2. Guerry, J., Boulch, A., Le Saux, B., Moras, J., Plyer, A., Filliat, D.: Snapnet-r: Consistent 3d multi-view semantic labeling for robotics. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. (2017) 669–678
3. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 770–779
4. Brazil, G., Liu, X.: M3d-rpn: Monocular 3d region proposal network for object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9287–9296
5. You, Y., Wang, Y., Chao, W.L., Garg, D., Pleiss, G., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. arXiv preprint arXiv:1906.06310 (2019)
6. Casser, V., Pirk, S., Mahjourian, R., Angelova, A.: Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 33. (2019) 8001–8008
7. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 7074–7082
8. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2147–2156
9. Ding, M., Huo, Y., Yi, H., Wang, Z., Shi, J., Lu, Z., Luo, P.: Learning depth-guided convolutions for monocular 3d object detection. arXiv preprint arXiv:1912.04799 (2019)
10. Choi, M., Kim, H., Han, B., Xu, N., Lee, K.M.: Channel attention is all you need for video frame interpolation, AAAI (2020)
11. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13** (2004) 600–612
12. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE (2012) 3354–3361
13. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 4490–4499
14. Wang, B., An, J., Cao, J.: Voxel-fpn: multi-scale voxel feature aggregation in 3d object detection from point clouds. arXiv preprint arXiv:1907.05286 (2019)
15. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 652–660
16. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 918–927

17. Ma, X., Wang, Z., Li, H., Zhang, P., Ouyang, W., Fan, X.: Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 6851–6860
18. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 8445–8453
19. Weng, X., Kitani, K.: Monocular 3d object detection with pseudo-lidar point cloud. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. (2019) 0–0
20. Liu, Z., Wu, Z., Tóth, R.: Smoke: Single-stage monocular 3d object detection via keypoint estimation. arXiv preprint arXiv:2002.10111 (2020)
21. Qin, Z., Wang, J., Lu, Y.: Monogrnet: A geometric reasoning network for monocular 3d object localization. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 33. (2019) 8851–8858
22. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 2002–2011
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
24. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. (2015) 91–99
25. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals for accurate object class detection. In: Advances in Neural Information Processing Systems. (2015) 424–432
26. Simonelli, A., Bulo, S.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 1991–1999
27. Smith, L.N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications. Volume 11006., International Society for Optics and Photonics (2019) 1100612
28. Naiden, A., Paunescu, V., Kim, G., Jeon, B., Leordeanu, M.: Shift r-cnn: Deep monocular 3d object detection with closed-form geometric constraints. In: 2019 IEEE International Conference on Image Processing (ICIP), IEEE (2019) 61–65
29. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 11867–11876
30. Xu, B., Chen, Z.: Multi-level fusion based 3d object detection from monocular images. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 2345–2353