3D Visual Object Detection from Monocular Images

Anonymous Authors

Anonymous Institutes

Abstract. 3D visual object detection is a fundamental requirement for autonomous vehicles. However, accurately detecting 3D objects was until recently a quality unique to expensive LiDAR ranging devices. Approaches based on cheaper monocular imagery are typically incapable of identifying 3D objects. In this paper, we propose a novel approach to predict accurate 3D bounding box locations on monocular images. We first train a generative adversarial network (GAN) to perform monocular depth estimation. The ground truth training depth data is obtained via depth completion on LiDAR scans. Next, we combine both depth and appearance data into a birds-eye-view representation with height, density and grayscale intensity as the three feature channels. Finally, We train a convolutional neural network (CNN) on our feature map leveraging bounding boxes annotated on corresponding LiDAR scans. Experiments show that our method performs favorably against baselines.

Keywords: 3D Object Detection · Depth Estimation · Monocular Vision.

1 Introduction

In the past few years, new types of LiDAR (Light Detection And Ranging) sensors have been developed for autonomous vehicles. These sensors provide an accurate 3D perception of the surrounding environment in real-time. As a result, several LiDAR-based classification, detection, and segmentation datasets are made available to public [10]. LiDAR is popular and advantageous compared to traditional stereo or multi-camera ranging devices for a variety of reasons. Firstly, LiDAR is able to give accurate measurements invariant of the ego car distance, while camera based ranging algorithms typically give a degraded performance on distant objects. This is because the object size reduces quadratically with distance to the camera for most imaging sensors. Secondly, LiDAR is an active time-of-flight (ToF) sensing device which works on a variety of objects including specular/metallic surfaces and textureless regions. Also, depending on the wavelength, LiDAR devices have certain levels of see-through capability on transparent objects (e.g. cloud, rain, snow). On the contrary, computer vision algorithms operating on camera sensors will start to fail when reflective/textureless/transparent regions increase. Finally, most LiDAR devices give 360-degree surrounding scans and immediate reading of orientation and distance to the object, whereas camera sensors usually have limited field-of-view (FOV) and multi-camera calibration issues, plus additional computation overheads to produce depth maps from raw input images. In the 2007 DARPA Urban Challenge, a team [20] finished in the second place using LiDAR alone with no camera sensors involved. Despite its advantages, there are a few major

drawbacks of LiDAR sensors. Firstly, they are typically bulky and expensive for wide use and deployment. Secondly, even top-of-the-line LiDAR sensors only provide 64 or 128 sparse scanlines across the 3D space, while camera sensors operate at a much higher resolution (typically ranging from 5 to 20 megapixels). Finally, LiDAR signals are inherently limited to spatial information and do not provide what cameras can typically see, such as words on the traffic sign, color, and pattern of the vehicle, etc. Therefore, it is still important to keep the camera sensors as a supplementary/fall-back option.

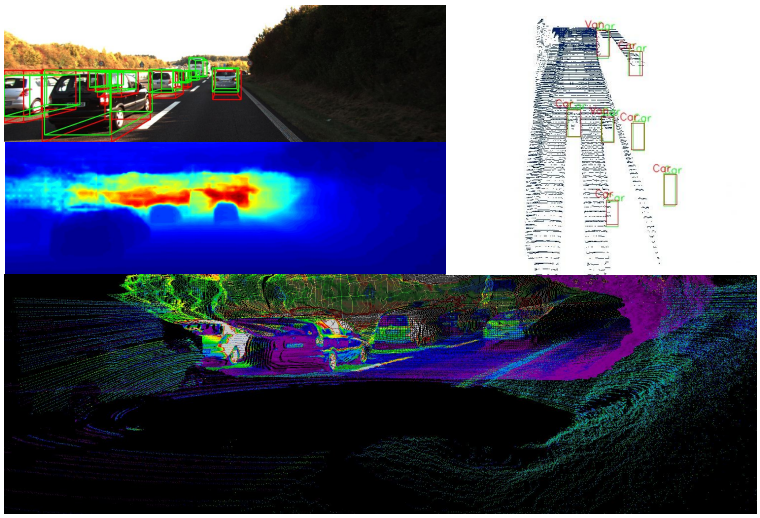

Fig. 1: Sample output and intermediate results from our pipeline (Best viewed electronically). Top left: Our predicted 3D bounding boxes (red) *vs.* ground-truth annotations on the KITTI dataset (green). Top middle: predicted depth map. Top right: 2D detection results on our depth map projected to the birds-eye-view (BEV) map. Bottom: Our transformed point cloud aligned with LiDAR scanlines. The intensity values on our point cloud are calculated using grayscale intensity values from the input RGB image.

Intuitively, depth data provide more useful descriptions of spatial information, while appearance data provide more visual cues to identify objects into different categories. Therefore, when combining semantically rich appearance data with depth information, one can improve the performance of both locating and categorizing objects in an image. Early research attempts to combine simple depth cues with image features for richer representation [25]. However, due to difficulty in propagating gradients in the model, simply stacking features from different modalities could not give satisfactory performance. Gupta *et al.* proposed to use horizontal disparity, height above ground, and angle with the direction of gravity to form another 3 channel image for training [13]. Due to difficulty on training these type of feature maps, it is a common practice to finetune on existing models trained on RGB images [26]. However, it is questionable whether this way of inter-model fusion is reasonable as depth features seldom resemble

shape, color, and appearance from the visible light spectrum. Lenz *et al.* proposes to learn features from RGB and depth images separately and then fuse at a higher level [17]. This method is termed *Late Fusion* by Eitel *et al.* [7]. Most work on 3D detection, on the other hand, are either purely based on LiDAR data [4, 24] or simply use visual cues to supplement LiDAR data [21, 18].

In this paper, we propose a novel approach to leverage both depth and visual cues for 3D object detection on monocular images. At the core of our technique is to integrate appearance and structural cues for better object detection. Our method contains three stages. Firstly, we use an unpaired image to image translation network to learn bi-directional transformations from RGB images to depth maps. Secondly, we calculate height, density and grayscale intensity as 3 feature channels and project the feature map to a birds-eye-view representation. Finally, we take advantage of 3D bounding box annotations on LiDAR data and train our object detection model on the feature map. The rest of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we demonstrate different components of the proposed method. We show experimental results and analysis in Section 4, 5 and draw conclusions in Section 6.

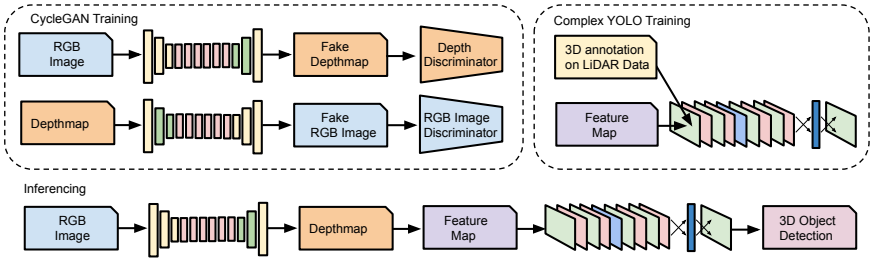

Fig. 2: Architecture of our network. Top left: our CycleGAN based depth prediction network. Top right: our 3D detection network based on Complex YOLO. Bottom: our network for inference. Note that for training our method requires both monocular images and aligned LiDAR scans. However, for inference we only need monocular images to predict 3D object locations and categories. See Section. 3 for details.

2 Related Work

Dual Learning The idea of using forward and backward consistency to improve training has a long history [5]. Recently, He *et al.* [14] proposed the concept of dual learning to improve the performance of machine translation systems. The proposed mechanism can be viewed as a two-agent communication game. The two agents may not be able to translate one language to another, but are still able to evaluate and collectively improve the quality of the two translation models by going through the full forward-backward translation cycle. This procedure can be performed by an arbitrary number of

rounds until the two models are fully converged. This idea inspired conditional GANs based cross-domain translation tasks [29] and improves performance in image-based depth/shape estimation tasks. [11, 28]. In this paper, we choose to use GAN for depth prediction because it allows unpaired image domain transfer, while other depth prediction models are usually dependent on LiDAR input guidance.

Image-to-Image Translation The idea of learning from a pair of images and then apply the model at inference time to produce an analogous target image from the input image dates back to [15]. More recently, Isola *et al.* [16] proposed a method which exploits conditional adversarial networks as a unified framework for image to image translation. It uses the L1 loss function to enforce generated synthetic images to be similar to ground truth training images while letting GANs to only hallucinate high-frequency details in the image. This is because the L1 loss can already guarantee similarity at low frequencies. Therefore, instead of processing the whole image, the discriminator only attempts to classify if a $N \times N$ image patch is correct or not. This method produces remarkable results on a variety of tasks, including photographs from sketches, automatic colonization of black and white images, raw images to label maps, thermal to visible light images, and so on.

3D Object Detection on LiDAR data Recent advance in sensor and computing technology enables 3D object detection on structural data. Due to the difficulty in processing large-scale point cloud data, most works preprocess the raw input data into either voxels or birds-eye-view maps (BEV). Chen *et al.* converts LiDAR data to a BEV representation for 3D object detection in the road scene [6]. Liang *et al.* develop a 3D object detector that reasons in BEV space and integrates visual cues by learning to project camera-based features into the BEV space [18]. YOLO3D [4] extends the 2D YOLOv2 object detector [22] to the BEV map and achieves real-time performance on the KITTI dataset. Complex-YOLO [24] also operates on the BEV map by running an E-RPN that estimates object orientations by both imaginary and real numbers.

3D Object Detection on RGB Images More recent publication [27] introduces the concept of *pseudo LiDAR*, arguing that by converting the image-based depth maps to a representation that closely mimics the LiDAR signal, one could obtain state-of-the-art results on stereo vision based 3D object detection. Our method is along the lines of performing 3D object detection on RGB images. However, our method differs from the pseudo-LiDAR approach in a few aspects. Firstly, our detection is performed on a feature map consists of height, density, and grayscale intensity information. This feature map combines both depth and visual cues and is not intended to mimic the LiDAR signal. Secondly, our method leverages unpaired adversarial learning to predict the depth map, eliminating the need for collecting pairwise-aligned RGB and depth data, thus making it much easier to apply to use cases other than autonomous driving (*e.g.* indoor scenes, close-up scenes, top-down surveillance videos, etc.)

3 Approach

3.1 Depth Estimation

We adopt the CycleGAN [29] for depth estimation from monocular images. We use the sparse to dense [19] depth completion results on KITTI LiDAR scans as groudtruth for training. The learning objective contains 2 terms: an adversarial loss and a cycle consistency loss:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = \\ \gamma_{adv}\mathcal{L}_{adv}(G, D_X, D_Y, X, Y) + \gamma_{cyc}\mathcal{L}_{cyc}(G, F)\end{aligned} \tag{1}$$

Where γ_{adv}, γ_{cyc} are the hyper-parameters to adjust loss on each term and are empirically set during the experiment. $\{x_i \in X\}_{i=1}^{N}$ and $\{y_i \in Y\}_{i=1}^{N}$ are N training images from the RGB dataset and depth dataset, respectively. G and F are mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$ to transform RGB images to depth maps or vice versa. D_X and D_Y are adversarial discriminators to distinguish between real images $\{x\}$ and synthetic images $\{F(y)\}$, or $\{y\}$ with $\{F(x)\}$. The cycle consistency loss is defined as:

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) =& \mathbb{E}_x \sim p_{data}(x)[\|F(G(x)) - x\|] \\ &+ \mathbb{E}_y \sim p_{data}(y)[\|G(F(y)) - y\|]\end{aligned} \tag{2}$$

It can be viewed as translating an RGB image into a depth image and then translate it back to compare with the original using L1 norm. Based on the cycle consistency loss, two discriminators D_X and D_Y are introduced to calculate the adversarial loss [12]. This term enforces the distribution of translated images to be as close to the training images as possible. The adversarial loss is defined as:

$$\begin{aligned}&\mathcal{L}_{adv}(G, F, D_X, D_Y, X, Y) = \\ &= \mathbb{E}_y \sim p_{data}(y)[logD_Y(y)] + \mathbb{E}_x \sim p_{data}(x)[logD_X(x)] \\ &+ \mathbb{E}_x \sim p_{data}(x)[log(1 - D_Y G(x))] \\ &+ \mathbb{E}_y \sim p_{data}(y)[log(1 - D_X G(y))]\end{aligned} \tag{3}$$

By using the above two loss terms we aim to minimize adversarial discriminator errors of both visual and structural cues, as well as L1 error of predicted images *vs.* original images.

3.2 Feature Map Generation

Once we obtained the trained GAN model for depth prediction, we would like to transform the depth map from camera coordinate system to the LiDAR coordinate system for alignment with ground-truth bounding box annotations. In order to do this, we first transform the depth map to the rectified (rotated) camera coordinate system:

$$\begin{aligned}z_{rect} &= D(u, v) \\ x_{rect} &= \frac{(u - c_u) \times z_{rect}}{f_u} + b_x \\ y_{rect} &= \frac{(v - c_v) \times z_{rect}}{f_v} + b_y\end{aligned} \tag{4}$$

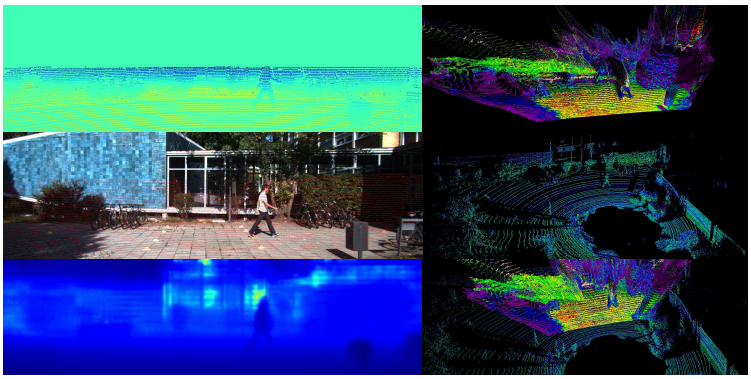

Fig. 3: Bidirectional transforms between LiDAR and camera coordinates (Best viewed electronically). Top left: LiDAR scans provided by the KITTI dataset projected to the camera imaging plane, color-coded by depth. Middle left: LiDAR scans projected to the corresponding RGB image. Bottom left: predicted depth map with one-to-one mappings to the input image. Top right: predicted depth map transformed to the LiDAR coordinates, color-coded by one channel grayscale intensity. Middle right: LiDAR scan color-coded by intensity/reflectivity. Bottom right: our transformed point cloud aligned with the LiDAR scan. Note the LiDAR has a much wider field of view (FOV).

Where $(x_{rect}, y_{rect}, z_{rect})$ is the 3D point coordinate in the rectified camera coordinates. (u, v) denotes a pixel location in the predicted depth map. (c_u, c_v) is the pixel location corresponding to the imaging center, f_u, f_v are the horizontal and vertical focal length and b_x, b_y are the baselines with respect to reference camera. The camera intrinsic can be obtained from the projection matrix provided by [10]:

$$\mathbf{P}_{rect} = \begin{pmatrix} f_u & 0 & c_u & -f_u b_x \\ 0 & f_v & c_v & -f_v b_y \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{5}$$

Next, we transform the 3D point cloud from rectified camera coordinates to reference camera coordinates and then to the LiDAR coordinates by calling the KITTI utility library[2]. Let $\mathbf{T}_{cam}^{velo}$ be the 4×4 transformation matrix from the camera coordinate system to the LiDAR coordinates, $\mathbf{R}_{rect}$ be the 4×4 rectifying rotation matrix converted from Cartesian to homogeneous coordinates by adding a fourth zero row and setting $\mathbf{R}_{rect}(4, 4) = 1$, $\mathcal{P}_{velo}$ and $\mathcal{P}_{rect} \in \mathbb{R}^3$ be the 3D point coordinates in the LiDAR and rectified camera coordinates, we can write the transformation as:

$$\mathcal{P}_{velo} = \mathbf{T}_{cam}^{velo} \, \mathbf{R}_{rect}^{-1} \, \mathcal{P}_{rect} \tag{6}$$

Note that we also store the RGB value and index of each point in another table to obtain the RGBXYZ representation of $\mathcal{P}_{velo}$. Next, we perform preprocessing in a fashion similar to Complex YOLO [24] to transform $\mathcal{P}_{velo}$ into the BEV feature map. The only difference between Complex YOLO and our method is that we are using grayscale

intensity (visual) as the blue channel of the image while Complex YOLO sets the blue channel to LiDAR intensity (reflectivity). More formally, let $\mathcal{S}$ be the mapping function to map each point in $\mathcal{P}_{velo}$ to a grid cell $\mathcal{S}_{bev}$ [24], we can formulate the transformation as:

$$
\begin{aligned}
f_g\left(\mathcal{S}_{bev}^j\right) &= \max\left(\mathcal{P}_{velo\to bev}^i \cdot [0,0,1]^T\right) \\
f_b\left(\mathcal{S}_{bev}^j\right) &= \max\left(I\left(\mathcal{P}_{velo\to bev}^i\right)\right) \\
f_r\left(\mathcal{S}_{bev}^j\right) &= \min(1.0, \log(\left|\mathcal{P}_{velo\to bev}^i\right|+1)/64)
\end{aligned}
\tag{7}
$$

Where f is the resulting 3-channel feature map. f_g, f_b, f_r denotes height map, grayscale intensity map and density map, respectively. I is the grayscale intensity of $\mathcal{P}_{velo}$ calculated from the RGB values. $\mathcal{P}_{velo\to bev}$ denotes the 3D points mapped to the grid cell $\mathcal{S}_{bev}$. To this end, we have constructed the feature map which is aligned with LiDAR 3D object bounding box annotations ready for training. We visualize the height, density and intensity maps of both LiDAR data and predicted depth data in Fig. 5. We also show the alignment of LiDAR data *vs.* our transformed point cloud in Fig. 3. As can be seen from Fig. 3, the LiDAR scanlines are accurately projected onto the camera coordinates. Also, the transformed point cloud is well-aligned with LiDAR data. It is worth-noting that the field of view (FOV) of LiDAR is much larger than the camera. This is reflected in both Fig. 3 (row 1 column 2 *vs.* row 2 column 2) and Fig. 5 (first row *vs.* second row). Therefore, unlike other methods, during mAP evaluation we only compare with ground-truth bounding box annotations that falls within the camera FOV.

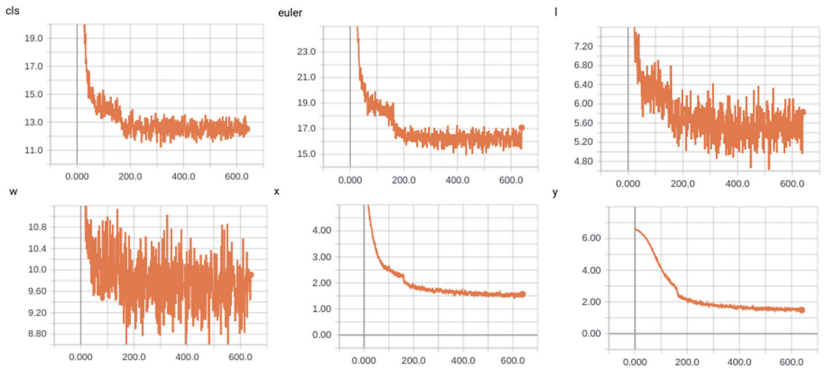

Fig. 4: Training loss visualization using TensorBoard [3]. Top: training loss on class labels, Euler region proposals and object length. Bottom: training loss on object width, horizontal and vertical locations. See Section. 4 for details.

3.3 3D Object Detection

We follow the Complex YOLO model architecture put forth by [24] to train the 3D object detector. This detector takes the BEV feature map mentioned in Section. 3.2 as input, and extends the YOLOv2 detector [22] by a complex angle regression and a Euler region proposal networks (E-RPN). The E-RPN is a direct extension of the region proposal networks (RPN) proposed by Ren *et al.* [23]. Specifically, consider (x, y, w, l, ϕ) as a vector describing 2D locations, size and orientations of 3D objects in the BEV coordinates, the parameterizations of the 5 coordinates can be obtained as [23, 24]:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x, b_y = \sigma(t_x) + c_y \\ b_w &= p_w e^{t_w}, b_l = p_l e^{t_l} \\ b_\phi &= \arg\left(|z| e^{ib_\phi}\right) = \arctan_2(t_{Im}, t_{Re}) \end{aligned} \tag{8}$$

The loss function of Complex YOLO is defined as a multi-part loss. The first part is the YOLOv2 loss [22] and the second part is an Euler regression loss:

$$\begin{aligned} L_{\text{Total}} &= L_{\text{YOLO}} + L_{\text{Euler}} \\ L_{\text{Euler}} &= \lambda_{\text{coord}} \sum_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} \left| e^{ib_\phi} - e^{i\hat{b}_\phi} \right| \end{aligned} \tag{9}$$

According to the authors, the Euler loss leads to a closed-form space eliminating singularities. This leads to state-of-the-arts results on the KITTI 3D object detection dataset, while achieving real-time performance on the embedded NVIDIA TX2 platform [24].

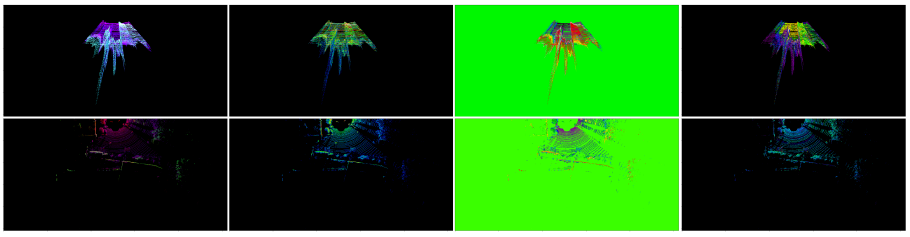

Fig. 5: Feature map visualization (Best viewed electronically). Top: Our combined BEV feature map, density map, height map and grayscale intensity map. Bottom: Feature map, density map, height map and intensity map on corresponding LiDAR scans used by Complex YOLO [24]. See Section. 3.2 for details.

4 Experiments

We use the KITTI 3D object detection benchmark suite for training and evaluation. The KITTI 3D object detection benchmark consists of 7481 training images as well as the

corresponding point clouds, training labels and camera calibration files. We first use the supervised model [19] provided by the author to obtain depth maps for all training images. We subsequently perform a random train (60%) / validation (25%) / test (15%) split and use the code provided by the author [29] for training the CycleGAN model. The model is trained from scratch with random weight initialization. We set base learning rate = 0.0002, gamma = 0.5, momentum = 0.5. We train for 200 epochs and use the CycleGAN model for constructing BEV feature maps.

Next, we modify the Complex YOLO framework by constructing feature maps on-the-fly during training. We still use the same train/validation sets and construct the BEV feature map for every image. We run CycleGAN inference on every image to obtain the depth map, then follow the transformations outlined in Section. 3.2 to obtain the 3-channel feature map as input for the Complex YOLO framework. Our implementation is based on the open source code provided by [1]. The ground-truth labels are obtained by converting 3D bounding box labels to 2D bounding boxes in the BEV coordinates. We set the base learning rate = 0.0001, gamma = 0.5, momentum = 0.9 and batch size = 32. We train for 700 epochs with four NVIDIA Titan V GPUs. The training losses mentioned in Section 3.3 are visualized in Fig. 4 using TensorBoard [3] until the 600th step ($\sim$ 4 epochs).

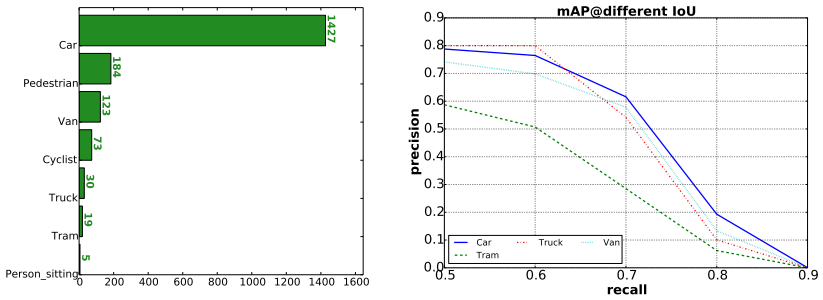

Fig. 6: Dataset statistics and precision-recall curve. Left: Number of objects per class in the KITTI dataset. Right: mean Average Precision (mAP) values on the car, truck, van and tram classes across varying Intersection-over-Union (IoU) values. Note that the performance of our model is robust to stricter IoU criterias and the performance only begins to significantly degrade when IoU is bigger than 0.6.

Similar to [24] which perform PASCAL VOC [9] style mean Average Precision (mAP) evaluation on its own test set split, we run mAP on 2D bounding boxes in the BEV space. There is one difference from our evaluation scheme *vs.* LiDAR-based methods: since LiDAR data has much wider FOV, we only evaluate against ground-truth bounding boxes that fall within our camera FOV. This means that we are not considering ground-truth labels that are too far away, or outside of our camera view frustum. Also, since our predicted depth map is not able to capture fine structures of small objects like LiDARs do, we only evaluate on four categories including car, tram, truck and

Method	Train	Test	Car	Tram	Truck	Van
MV3D [6]	LiDAR+RGB	LiDAR+RGB	86.02	N/A	N/A	N/A
C-YOLO [24]	LiDAR	LiDAR	85.89	N/A	N/A	N/A
Ours	LiDAR+RGB	RGB	78.78	58.70	80.00	74.14

Table 1: Quantitative mAP results. Note that we only evaluate within the visible range of the predicted depth map/point cloud, whereas all other methods evaluate on the full LiDAR scan. Also, our method and ComplexYOLO report scores on the random test split while MV3D evaluate on the test set. Both MV3D and ComplexYOLO scores are reported under the easy category of the BEV evaluation task. See Section. 4 for details.

van. As shown in Fig. 6, these four (out of seven) categories consist of more than 85% of objects in the KITTI 3D object detection dataset. We also vary the IoU threshold from 0.5 to 0.9 with a 0.1 interval and recalculate the mAP scores. We show the mAP scores at different IoU thresholds in Fig. 6. Compared to existing methods, our framework is one of the few that directly performs inference on RGB images. We compare with MV3D [6] and Complex YOLO [24] results in Table 1. The scores of MV3D and Complex YOLO are adopted from the original paper in the BEV category with easy difficulty. Easy difficulty is defined according to the bounding box height and occlusion/truncation levels. In general, the easy task corresponds to cars within 30 meters of the ego-car distance, according to [27]. Note that the effective range of our transformed point cloud is shorter than this 30-meter range. Also, the Complex YOLO scores are reported on the test split (similar to our evaluation) whereas the MV3D reports on the KITTI test set.

5 Discussion

We show quantitative PR-curve evaluations in Fig. 6 and compare with other methods in Table 1. As can be seen from Fig. 6, our method achieves satisfactory results on car, truck, van and tram categories, and the car category demonstrates the highest mAP scores across varying IoUs. This may due to the fact that cars are more common in real-life scenes and thus easier to recognize. Also, because the categories in the KITTI dataset is highly imbalanced, it is possible that the car class is over-represented and the classifier is biased towards this single class. In the future, we plan to test our approach on an evenly sampled 3D object detection dataset with more diverse examples. According to Fig. 6, the mAP starts to dramatically decrease only when the IoU value is more than 0.6. This shows that our detector is robust to stricter evaluation criteria, which is generally more desirable for complex real-life scenes. Also, according to Table 1, our method is competitive when compared to other LiDAR-based methods, even though our network only uses RGB images as input to perform forward inference. We visualize the qualitative results in Fig. 7. Our approach works well in cluttered scenes (*e.g.* row 1 and 2). It might be difficult for appearance-based methods to separate vehicles parked closely together (row 2 column 1), but our method makes accurate depth predictions and the BEV map (row 2 column 2) makes it much easier to learn the relative

locations of the vehicles. However, for small objects and thin structures (*e.g.* pedestrian in row 3 and 4), our network is not able to capture, as the predicted depth maps are not as accurate as LiDAR scans. Also, our method takes both structural (*e.g.* height) and visual cues for inference. For example, in the last row, the closest and farthest objects are wrongly classified as vans while the middle object is correctly classified as a car. This is because our feature map also contains the height map. The SUV and MPV in the front and back are taller than the sedan in the middle, which possibly leads to the wrong classification result. In general, Fig. 7 demonstrates that the bounding box predictions (structural) are more accurate than class predictions (visual appearance). This implies that our network is good at localizing objects but is still having difficulties learning visual features of an object. This can also be observed in Fig. 4, where the classification loss curve shows more oscillations than bounding box coordinates (x and y). Although the learning objective is designed to minimize both classification and localization errors, it is interesting to see what roles the structural and visual cues play, and when one overwhelms the other. In the future, we plan to train and test on more datasets and visualize neuron activation heatmaps in each channel (height, density and color intensity).

6 Conclusion

In this paper, we have presented a framework to detect and classify 3D objects from monocular images. Experiments show that our approach performs favorably against competitive methods trained on LiDAR data. Our method leverages generative adversarial networks to perform monocular depth estimation. The training groudtruth are obtained by completing LiDAR scans. The GAN approach is more flexible in terms of extending to other computer vision tasks. On the contrary, traditional monocular depth prediction networks are heavily dependent on pair-wise color-to-depth alignment and LiDAR input. Also, we integrate both visual and structural cues into the feature map representation, which distinguishes our method from those purely operating on LiDAR data, and those who learn depth from a monocular image but still perform detection on the pseudo LiDAR data (ignoring visual information). Our system can be used to add visual intelligence to smart vehicles, which is particularly useful for improving camera-based advanced driver-assistance systems (ADAS) for L3 level autonomy. Also, our system could be used as a supplementary or fall-back option to LiDAR sensors. In the future, we plan to include spatiotemporal data to improve both depth prediction (*e.g.* optical flow) and object detection (*e.g.* YOLO4D [8]).

References

1. Complex yolo with uncertainty. https://github.com/wl5/complex_yolo_3d
2. pykitti open source utility library. https://github.com/utiasSTARS/pykitti
3. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: USENIX Symposium). pp. 265–283 (2016)
4. Ali, W., Abdelkarim, S., Zidan, M., Zahran, M., El Sallab, A.: Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In: ECCV. pp. 0–0 (2018)

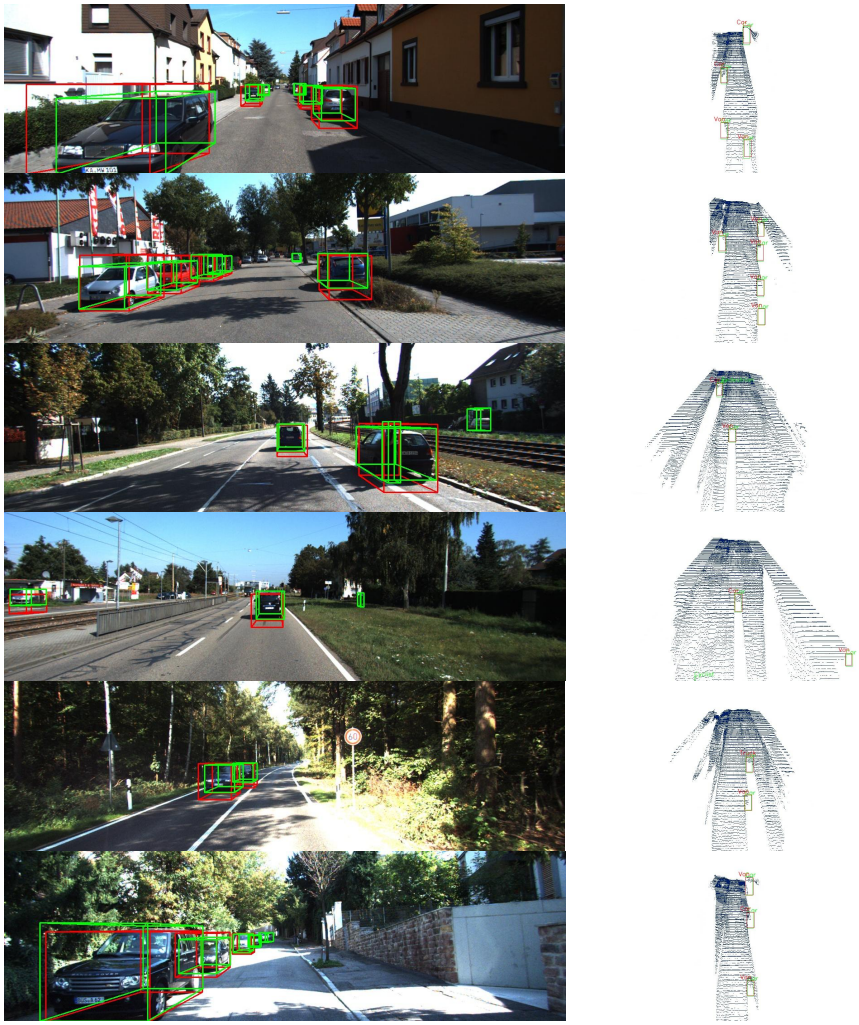

Fig. 7: Qualitative results on the KITTI dataset. Left: our 3D bounding box predictions (red) *vs.* ground-truth (green) annotations projected to the camera imaging plane. Right: our 2D bounding box predictions (red) on the BEV map vs ground-truth (green) annotations. Note that the camera optical axis is facing down on the BEV map for better visualization. See Section. 5 for details.

5. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings annual conference on Computational learning theory. pp. 92–100. ACM (1998)
6. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: CVPR. pp. 1907–1915 (2017)
7. Eitel, A., Springenberg, J.T., Spinello, L., Riedmiller, M., Burgard, W.: Multimodal deep learning for robust rgb-d object recognition. In: IROS. pp. 681–687. IEEE (2015)
8. El Sallab, A., Sobh, I., Zidan, M., Zahran, M., Abdelkarim, S.: Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds (2018)
9. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV **88**(2), 303–338 (2010)
10. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. IJRR **32**(11), 1231–1237 (2013)
11. Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: CVPR. vol. 2, p. 7 (2017)
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS. pp. 2672–2680 (2014)
13. Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from rgb-d images for object detection and segmentation. In: ECCV. pp. 345–360. Springer (2014)
14. He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T., Ma, W.Y.: Dual learning for machine translation. In: NIPS. pp. 820–828 (2016)
15. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: Conference on Computer graphics and interactive techniques. ACM (2001)
16. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arXiv preprint (2017)
17. Lenz, I., Lee, H., Saxena, A.: Deep learning for detecting robotic grasps. IJRR **34**(4-5), 705–724 (2015)
18. Liang, M., Yang, B., Wang, S., Urtasun, R.: Deep continuous fusion for multi-sensor 3d object detection. In: ECCV. pp. 641–656 (2018)
19. Mal, F., Karaman, S.: Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In: ICRA. pp. 1–8. IEEE (2018)
20. Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., et al.: Junior: The stanford entry in the urban challenge. Journal of field Robotics **25**(9), 569–597 (2008)
21. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: CVPR. pp. 918–927 (2018)
22. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: CVPR. pp. 7263–7271 (2017)
23. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS. pp. 91–99 (2015)
24. Simony, M., Milzy, S., Amendey, K., Gross, H.M.: Complex-yolo: an euler-region-proposal for real-time 3d object detection on point clouds. In: ECCV. pp. 0–0 (2018)
25. Socher, R., Huval, B., Bath, B., Manning, C.D., Ng, A.Y.: Convolutional-recursive deep learning for 3d object classification. In: NIPS. pp. 656–664 (2012)
26. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: CVPR. vol. 5, p. 6 (2015)
27. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. arXiv preprint arXiv:1812.07179 (2018)
28. Zhou, T., Krahenbuhl, P., Aubry, M., Huang, Q., Efros, A.A.: Learning dense correspondence via 3d-guided cycle consistency. In: CVPR. pp. 117–126 (2016)
29. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017)