Multi-Level Fusion based 3D Object Detection from Monocular Images

Bin Xu, Zhenzhong Chen*
School of Remote Sensing and Information Engineering, Wuhan University, China
{ysfalo,zzchen}@whu.edu.cn

Abstract

In this paper, we present an end-to-end multi-level fusion based framework for 3D object detection from a single monocular image. The whole network is composed of two parts: one for 2D region proposal generation and another for simultaneously predictions of objects' 2D locations, orientations, dimensions, and 3D locations. With the help of a stand-alone module to estimate the disparity and compute the 3D point cloud, we introduce the multi-level fusion scheme. First, we encode the disparity information with a front view feature representation and fuse it with the RGB image to enhance the input. Second, features extracted from the original input and the point cloud are combined to boost the object detection. For 3D localization, we introduce an extra stream to predict the location information from point cloud directly and add it to the aforementioned location prediction. The proposed algorithm can directly output both 2D and 3D object detection results in an end-to-end fashion with only a single RGB image as the input. The experimental results on the challenging KITTI benchmark demonstrate that our algorithm significantly outperforms monocular state-of-the-art methods.

1. Introduction

In recent years, with the development of technologies in computer vision and deep learning, numerous impressive methods are proposed for accurate 2D object detection. The results of 2D detection indicate the accurate locations for each object in image coordinate system and the object class for it. However, in several scenarios like robotic application and autonomous driving, it is not enough to describe objects in the 3D real world scene with 2D detection results only.

The focus of this paper is on 3D object detection utilizing only monocular images. We aim at extending existing 2D object detectors for accurate 3D object detection in the context of self-driving cars, without any help of expensive LIDAR systems, stereo information or hand-annotated maps of the environment. For 2D object detection, since the success of region-based convolutional neural networks (R-CNNs) [13], the advanced promising works like SPP-Net [17], Fast R-CNN [12], Faster R-CNN [31], R-FCN [7] and Mask R-CNN [16] mostly apply deep convolutional neural networks (CNNs) to learn features from region candidates over the image for accurate 2D object recognition. Here we would like to extend the existing image-based 2D detection algorithms for 3D object detection. Usually, a 2D object is described by its location in the image, which is quite different from the representation of a 3D object. Typically, a 3D object like a car in the real world is represented by its pose, 3D dimension and localization of its center in the camera coordinate system. It is much more complicated for 3D object detection with only monocular images. However, since the existence of imaging mechanism and geometric constraints, all the descriptors of a 3D object still have compact relations to the projected image content, it is possible to handle the 3D detection problem with only monocular images.

To deal with this, we propose a framework for 3D object detection by estimating the object class, 2D location, orientation, dimension, and 3D location based on a single monocular image in an end-to-end fashion. A region proposal network (RPN) is utilized to generate 2D proposals in the image, as RPN provides strong objectness confidence regions of interest (RoIs) with CNN features and it can share weights with the down-stream detection network [31, 36, 37, 19]. With features learned from the proposals, both object class confidence and 2D bounding box offset to the proposal are predicted, just like most region-based 2D object detectors. Two more branches are added for jointly learning of orientation and dimension. In addition, another module is introduced to estimate the disparity information and adopt multi-level fusion method for accurate 3D localization, constituting our 3D object detection procedure.

In this work, to get the pose of a 3D object, we follow a similar idea as the *MultiBin* architecture described in [26] by adopting discrete-continuous formulation for orientation estimation. For 3D object dimension, typical sizes made

*Corresponding author: Zhenzhong Chen

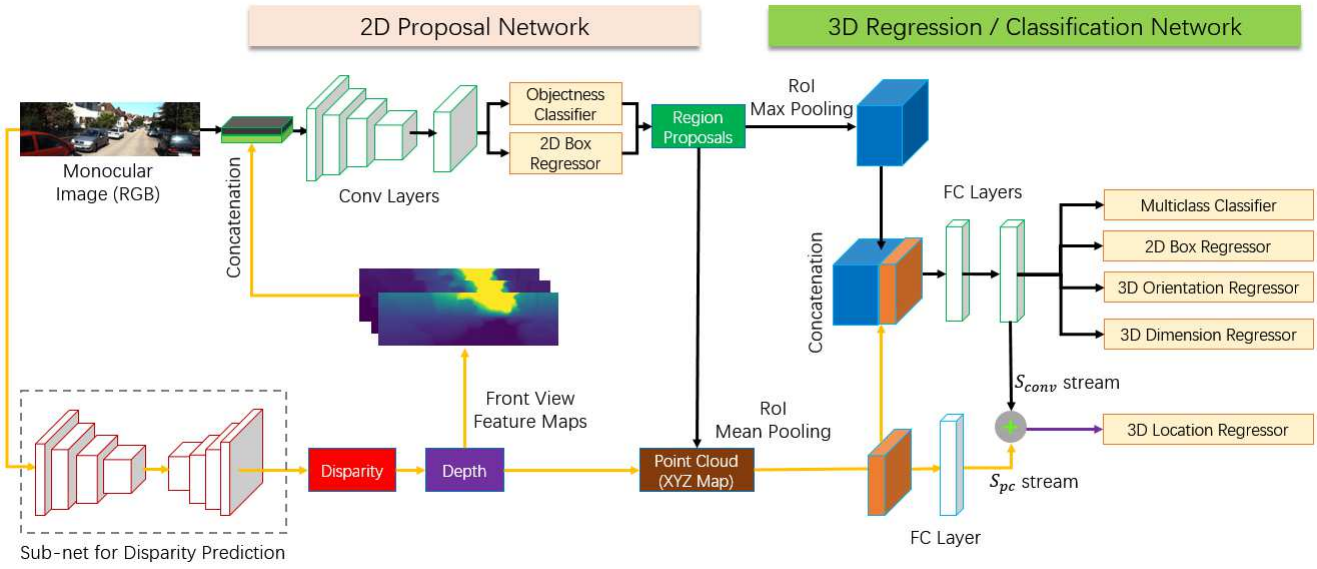

Figure 1. The proposed framework for 3D object detection.

up of *length*, *width* and *height* are accessed by analyzing the training labels for each class. The offset between actual dimension and typical dimension is estimated from the network. It is much more complicated for estimating the 3D coordinates (*X, Y, Z*) of object center, since only image appearance cannot decide the absolute physical location. To solve this, global context information needs to be considered as a prior for each region candidate. For the input monocular image, the disparity information of each pixel will be estimated through a fully convolutional network (FCN), thus the approximate depth and point cloud can be reached with the help of camera calibration files. Then we superadd the estimated information in multiple steps for 3D localization. A *RoI mean pooling* layer is introduced to convert the point cloud inside the proposal into a fixed-length feature vector through mean(average) pooling. 3D location of the object center will be estimated with features from point cloud and the original convolutional features. Besides, estimated depth is also encoded as front view feature maps and be fused with the RGB image to improve the performance. Thus all the 2D and 3D descriptors can be simultaneously predicted.

As just described, our work focuses on 3D object detection from monocular images with existing 2D object detectors. Our first contribution is an efficient multi-level fusion based method for 3D object detection with a stand-alone module for estimating the disparity information. Features from disparity and the original RGB image are fused in different levels, proposing a possibly effective method for accurate 3D localization. The second contribution is the introduction of a general framework that can directly extend existing region-based 2D object detectors for 3D object detection. End-to-end learning is applied for estimating a 3D object's full pose, dimension and location without any other additional annotations or 3D object models. The experimental evaluation shows that our approach is able to perform really well on the very challenging KITTI dataset [11], outperforming the state-of-the-art monocular methods and even some methods with stereo information on particular evaluation index.

The remainder of this paper is organized as follows: In the next section, we review the related literature. Section 3 explains our framework and exhibits more details of the proposed algorithm. After providing experimental results and comparisons on the very challenging KITTI dataset in Section 4, we conclude this paper.

2. Related Work

Our work is related to 2D object detection and monocular 3D object detection. The literature review will mainly be focused on 2D object detection algorithms and 3D object detection methods with only monocular images.

2D Object Detection. The majority of state-of-the-art 2D object detectors belong to deep learning methods. According to the detection procedure and mechanism, they can be categorized into two parts [23]. The first is one-stage detectors, they are applied over a dense sampling of possible object locations, such as OverFeat [32], YOLO [29, 30] and SSD [24, 10], that can provide promising results with relatively fast speed. These methods trailed in accuracy even with a larger compute budge [23, 18]. The other is two-stage, proposal-driven detectors that apply classification and regression over learned features within object

proposals. In the first stage, several methods are adopted for proposal generation. The widely-used ones include BING [5], Selective Search [35], EdgeBoxes [39], DeepMask [27, 28] and RPN [31]. The most famous two-stage detector is the aforementioned Faster R-CNN, which can generate proposals and apply object recognition in an end-to-end fashion. Typically, two-stage detectors need fewer data augmentation tricks and have more accurate results in most public benchmarks. Through a sequence of advances [12, 31, 22, 16], this two-stage framework consistently achieves top accuracy on the challenging COCO benchmark [21, 23]. According to [23], recent work on one-stage detectors demonstrates promising results, while the accuracy is within 10-40% relative to state-of-the-art two-stage methods.

Monocular 3D Object Detection. Both 2D object detection and monocular 3D object detection are adopted on a single RGB image. Unlike 2D object detection, it can be quite difficult for monocular 3D object detection, since the lack of stereo information and accurate laser points from other sensors. Like two-stage, region-based 2D detectors, several works make use of high quality 3D region proposals for further classification and detection. In [2], the author makes use of a general assumption that all the objects should lie close to the ground plane, which should be orthogonal to the image plane. The 3D object candidates are then exhaustively scored in the image plane by utilizing class segmentation, instance level segmentation, shape, contextual features and location priors. With the projected 2D proposals from 3D candidates, it uses Fast R-CNN to jointly predict category labels, bounding box offsets, and object orientation. It performs really well on KITTI and outperforms all published monocular object detectors at that time. Recently proposed works mainly focus on combining deep neural networks and geometric properties, providing more accurate results. Deep3DBox [26] introduces geometric constraints into 3D object detection scenario. It is based on the fact that the 3D bounding box should fit tightly into 2D detection bounding box, thus it requires each side of the 2D bounding box to be touched the projection of at least one of the 3D box corners. Deep3DBox combines visual appearance and geometric properties to find the 3D location. It also utilizes deep CNN features to estimate the pose and dimension of a 3D object, constituting the complete detection framework. Another recently introduced method for monocular 3D object detection is called Deep MANTA [1]. It takes a monocular image as input and can output vehicle part coordinates, 3D template similarity and part visibility properties, in addition to 2D scored bounding boxes. Deep MANTA encodes 3D vehicle information using characteristic points of vehicles, since they are rigid objects with well known geometry. Corresponding to this, deep MANTA uses a 3D vehicle dataset composed of 3D meshes with real dimensions and several vertices are annotated for each 3D model. Then the vehicle part recognition in Deep MANTA can be considered as extra key points detection, which will be adopted for 2D / 3D matching with the most similar 3D template, thus the 3D localization results can be achieved. Although it only needs a single RGB image as input, additional annotations like part locations and visibility for 3D objects and an extra dataset of 3D templates are necessary for the training and inference stages.

3. 3D Object Detection Framework

The proposed framework is an end-to-end network that takes a monocular image as input and output 2D/3D object representations. The system has two main parts: 2D region proposal generation and simultaneous 2D/3D parameters estimation. In particular, we adopt multi-level fusion methods for accurate 3D localization and system enhancement, constructing the robust detection pipeline. The whole framework is illustrated in Figure 1.

3.1. 2D Region Proposal Generation

In our implementations, we utilize region proposal network (RPN) introduced in Faster R-CNN to extract RoIs for further detection. In RPN, a set of rectangular object proposals with objectness scores are generated through a slided small network over the convolutional feature map and the *Anchors* mechanism [31]. 2D *Anchors* are generated with pre-defined scales and aspect ratios over a basic rectangle in each location. Then the network can output region proposals through objectness scores prediction and 2D bounding box regression.

3.2. 2D/3D Parameters Estimation

Based on the 2D region proposals, 2D/3D parameters for object description are estimated. For the 2D part, it consists of multi class classification and 2D box regression, our implementation is just like Faster R-CNN. For the 3D part, it is determined by orientation estimation, dimension estimation and 3D localization. Next we will describe our approach for estimating these different parameters.

With region proposals of different spatial sizes, the *RoI pooling* layer is introduced in [12] to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of $H \times W$ via max pooling. This layer is regarded as *RoI max pooling* to prevent confusion in this paper. With fixed-sized input features, then the fully connected layers can be added for estimating the corresponding parameters. In Fast R-CNN, features for region classifier and bounding box regressor are shared. Following this principle, we add two additional branches on the top of the shared features for object orientation and dimension regression at first.

Orientation Estimation. For the orientation branch, it is not possible to estimate the global orientation in the camera reference frame from only the contents of the region proposal [26]. Thus the local orientation is regressed with the state-of-the-art *MultiBin* architecture [26]. It discretizes an angle and divides it into *n* overlapping bins. With the input features, both confidence probabilities for each bin and the residual part to the center of bin are estimated. The two parts are regarded as angle confidence and angle localization, respectively.

The loss function for angle confidence equals to cross entropy (CE) with sigmoid function as probabilities, since there are overlaps between adjacent bins and an angle could belong to more than one bin. We adopt smooth L_1 defined in [12] for the regression loss of the residual angle. The smooth L_1 loss is a robust L_1 loss that is less sensitive to outliers than the L_2 loss, which is defined as follows:

$$SL_1(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{1}$$

Besides, instead of predicting the residual angle directly, we predict the sine and the cosine of it, just like [26]. Overall, the loss function for angle confidence L_{conf} is then defined as:

$$L_{conf} = CE(\sigma(P_{conf}), D^*_{conf}) \tag{2}$$

D^*_{conf} is the ground data for bin class. $\sigma(\cdot)$ indicates sigmoid function. For the residual part, the loss L_{loc} is defined as:

$$L_{loc} = \frac{1}{n} \cdot SL1(P_{loc} - D^*_{loc}) \tag{3}$$

D^*_{loc} are ground truth residual data, n is the number of bins that cover ground truth and $SL1(\cdot)$ indicates smooth L1 function. The total loss for orientation is thus:

$$L_\alpha = L_{conf} + w \cdot L_{loc} \tag{4}$$

It is made up of angle confidence loss and angle localization loss. w is used to balance the relative importance for the two parts.

Dimension Estimation. For dimension estimation, we do not regress the absolute dimensions directly. Instead, we compute the average *length*, *height* and *width* for each class over the training dataset at first to get the typical dimension. Then the offset to the typical size is estimated with the same shared features, just like the part for the regression of 2D bounding boxes. The loss function in this branch is defined as follows:

$$L_d = SL1(log(\frac{P_d}{D_t}) - log(\frac{D^*_d}{D_t})) \tag{5}$$

whereP_d, D^*_d means the prediction and ground truth, respectively. D_t indicates typical dimension that is computed from training labels.

3.3. Multi-Level Fusion and 3D Localization

It is much more complicated to estimate the 3D location of a 3D object. In our previous branches, only features inside region proposals are utilized for angle and dimension regression. However, estimating 3D location in the same way is difficult since the existence of *RoI max pooling*. Features generated from *RoI max pooling* have several drawbacks. First, it converts the features inside RoIs with different scales into fixed-sized feature tensors. Thus it partly eliminates a fundamental photography constraint that the RoI with bigger size should lie closer to the camera. Besides, the image coordinates of each RoI are only used to fetch the corresponding region on the feature map. It means that different spatial locations of RoIs may have similar output after *RoI max pooling*, while the actual 3D location can differ a lot. In 2D bounding boxes regression, the absolute coordinates are achieved by estimating the offset to the proposals. However, for 3D localization, the 2D proposals do not contain 3D information of the coordinates.

Another fact is that we human can tell the approximate 3D location of any object in a monocular image if we have seen the scene before. A rich understanding of the world can be developed through our past visual experience [38]. The understanding can be used as the prior knowledge for the whole image, which can be adopted to localize any object, even any pixel in it. To help localize the 3D object in our framework, the approximate layouts of objects or 3D locations of pixels can be modeled.

Disparity Estimation. To help understanding the whole scene in the image, a sub-net for disparity estimation from a single image is proposed. Learning based methods for disparity or depth estimation are mainly built with fully convolutional networks (FCNs). However, disparity or depth estimation with only monocular information can be quite difficult. Thanks for the remarkable work called *MonoDepth* in [14], which enforces left-right depth consistency for disparity estimation in an unsupervised way, we directly use the pre-trained weights provided by *MonoDepth* to initialize our sub-net and the weights won't be updated during training. Thus the sub-net can be regarded as a stand-alone off-line module to help understand the whole scene. With camera intrinsic parameters and estimated disparity, 3D coordinates of each pixel in camera coordinate system can be accessed, thus constructing the point cloud in the whole scene. Given a pixel $I = (I_x, I_y)$ with disparity I_d in the 2D image, the 3D coordinates (x, y, z) can be computed as:

$$\begin{cases} z = f * C_b / I_d \\ x = (I_x - C_x) * z / f \\ y = (I_y - C_y) * z / f \end{cases} \tag{6}$$

(a) Depth (b) Height (c) Distance

Figure 2. Front view feature maps.

where f is the focal length of the camera, (C_x, C_y) is the principal point, C_b is the baseline distance. In this paper, we always adopt the camera coordinate system in KITTI for 3D locations[1].

Estimation Fusion for 3D Localization. With the estimated point cloud in the whole image as our prior knowledge and the previously generated RoIs, here we introduce a *RoI mean pooling* layer for 3D localization. The principle is quite simple and very similar to *RoI max pooling*. *RoI max pooling* is proposed mainly to deal with high-level visual recognition tasks. The *max* operation aggregates multiple activations and outputs the max value, which has been demonstrated useful in numerous applications. Nowadays, the mostly used *pooling* operation in deep neural networks is also *max pooling*. In *RoI mean pooling*, we just replace the *max* operation with *mean* operation. Since original point cloud does not contain any high-level representations, the *max* pooling will get the maximum value for each axis, which is not reasonable. We simply adopt *RoI mean pooling* over the point cloud to get a fixed-sized *point cloud feature map*, which takes both global prior knowledge about locations and the region proposal into account. Here the point cloud is encoded as a 3-channel XYZ map which has the same size as the RGB image. With this representation, we can use ROI Pooling operation on the XYZ map, similar to the ROI Pooling on convolutional feature maps. Objects with the same appearance in the image probably have different point cloud features when they locate differently in 3D space. Therefore, the point cloud feature is complementary to the appearance feature and crucial to 3D localization.

Now we have two streams for 3D localization, one is from CNN features with *RoI max pooling* and the other is from point cloud with *RoI mean pooling*. The two parts are regarded as S_{conv} and S_{pc} for better description. For the S_{conv} stream, we estimate the 3D location, just like the orientation and dimension branches. For the S_{pc} stream, we only use one fully connected layer and then estimate the 3D location. Finally, location estimations from the two streams will be added, constructing the final 3D location estimation. In the training stage, smooth L1 function is applied again, so the loss for 3D localization can be represented as follow:

$$L_{loc} = SL1((P_{pc} + P_{conv}) - D^*_{loc}) \tag{7}$$

where P_{pc} indicates estimation from S_{pc} and P_{conv} indicates estimation from S_{conv}.

The joint estimation for 3D location can be seen as a late fusion between estimations from S_{conv} and S_{pc}. The convolutional features are learned from image content and mostly contain high-level semantic information. While the representations from point cloud can be seen as low-level manipulation about pixel locations. Estimations from different levels are fused for accurate 3D estimation. In another point of view, the point cloud that depends on estimated depth information can be regarded as global prior knowledge. It contains spatial information about the RoIs and can help compensate the information reduction after *RoI max pooling*. The late fusion ensures the accurate 3D localization in the network, which is the most important part of the whole 3D object detection framework.

Input Fusion with Front View Feature Maps Encoding. In addition to helping estimate the 3D locations, we also encode the estimated depth information with three-channel representations as the front view feature maps, which is similar to [4, 15, 20]. Given a pixel $I = (I_x, I_y)$ with depth I_{depth} in the 2D image, the three front view feature maps FV_1, FV_2, FV_3 are encoded as:

$$\begin{cases} FV_1 = I_{depth} \\ FV_2 = (I_y - C_y) * I_{depth}/f \\ FV_3 = \sqrt{(FV_1)^2 + (FV_2)^2 + ((I_x - C_x) * I_{depth}/f)^2} \end{cases} \tag{8}$$

where f is the focal length of the camera, (C_x, C_y) is the principal point. As we can see, FV_1 represents depth information, FV_2 denotes height information and FV_3 indicates distance information in the camera coordinate system. The encoded front view feature maps are visualized in Figure 2.

Since the sub-net for disparity estimation is an off-line module and will not be updated in the training stage, the three-channel front view feature maps and three-channel RGB images are concatenated as the input for RPN. This

[1]Details about the set up for camera coordinate system can be found in `http://www.cvlibs.net/datasets/kitti/setup.php`

can be regarded as an early fusion or a pre-processing step for enhancing the input.

Feature Fusion for Accurate Estimation. In the estimation of 3D location, another fusion for different feature maps is proposed. Here we regard feature maps after *RoI max pooling* as F_{max}(blue cube in Figure 1) and feature maps after *RoI mean pooling* as F_{mean}(orange cube in Figure 1). Then the two feature maps are concatenated to enhance the S_{conv} stream.

In total, there are three levels of fusion in the network. The earliest fusion is the concatenation between front view feature maps and the corresponding RGB image. The second one is the mergence of feature maps from the original input and the estimated point cloud inside each region proposal, with different types of *RoI pooling*. The last fusion is the joint estimation from two different types of data for the final 3D localization. Generally, the last fusion is necessary for the framework, while the other two can improve the whole performance to a certain extent. Details for comparisons are illustrated in Section 4.

3.4. Multi-task Loss

The whole loss L_{rcnn} for the 3D regression and classification network can be formulated as:

$$L_{rcnn} = w_{2D}L_{2D} + w_{\alpha}L_{\alpha} + w_{d}L_{d} + w_{loc}L_{loc} \quad (9)$$

where all the loss for 2D detection is regarded as L_{2D} in this paper, including multiclass classification loss and 2D bounding box loss, just like the description in [31]. For the RPN part, we use the same loss as the original paper [31]. In order to optimize the whole framework, joint training is adopted for the whole network, which is also introduced from the original Faster R-CNN implementations. Then the whole network is trained end-to-end.

4. Experimental Results

We evaluate our approach on the challenging KITTI dataset. It provides 7,481 images for training and 7,518 images for testing, along with the camera calibration files. Detection is evaluated in three regimes: *easy*, *moderate* and *hard*, according to the occlusion and truncation levels of objects. Since there are no ground truth for the testing images, we conduct 3D box evaluation on the validation set. We split the training set into train/val parts. For better comparisons with other start-of-the-art algorithms, we use two train/val splits: *train1/val1* from [2, 3] and *train2/val2* from [26].

Implementation Details. We choose Faster R-CNN with 16-layer VGG net [34] as our basic 2D object detectors. For *RoI max pooling*, the recently introduced operator called *Deformable RoI Pooling* described in [8] is adopted in our implementation. The *RoI mean pooling* operator is modified from *RoI align* described in [16]. By default, we use pre-trained weights learned on ImageNet [9] dataset for the initialization of our network. In addition, the input of models trained on ImageNet is a single three-channel RGB image. To handle the six-channel input in our framework (three-channel RGB image + three-channel front view feature maps), the weights in the first convolutional layer are just duplicated for initialization. To handle particularly small objects in KITTI images, the shorter side of the training images is upscaled to 512 pixels, which was found to be crucial to achieve very good performance. We don't apply any data augmentation methods, even the *flip* operation, which is usually used in most 2D detectors. For anchors used in RPN, 5 scales of 2, 4, 8, 16, 32 and 3 aspect ratios of 1:1, 1:2, 2:1 are used on the basic 16 $\times$ 16 box, where 16 is just the stride size on the utilized feature map according to the original implementation of Faster R-CNN. Mostly used sampling heuristics like the fixed foreground-to-background ratio and online hard example mining (OHEM) [33] are adopted to maintain a balance between foreground and background for good performance. Mini-batch stochastic gradient descent (SGD) is the optimizer of the network. We use a batch size of $N = 1$ for images and a batch size of $R = 256$ for proposals. Weights of the sub-net for disparity prediction are from the model which is trained on Cityscapes [6] and KITTI[2]. The network is trained with a learning rate of 0.0005 for 30K iterations. Then we reduce the learning rate to 0.00005 and train another 10K iterations. In our implementation, the whole network takes around 120ms per image on a single TITAN X GPU in inference stage.

Evaluation Metrics. We evaluate 3D object detection results using the official evaluation metrics from KITTI. 3D box evaluation is conducted on both two validation splits (different models are trained with the corresponding training sets). We focus our experiments on the car category as KITTI provides enough car instances for our method. Following the KITTI setting, we do evaluation on three difficulty regimes: *easy*, *moderate* and *hard*. In our evaluation, the 3D IoU threshold is set to 0.5 and 0.7 for better comparison. We compute Average Precision (AP_{loc}) for the bird's eye view boxes, which are obtained by projecting the 3D boxes to the ground plane. Average Precision (AP_{3D}) metric is also used to evaluate the full 3D bounding boxes. Table 1 shows AP_{loc} on *val1* and *val2* and Table 2 is the comparison results for AP_{3D}. Table 3 shows results on the KITTI testing set.

[2] https://github.com/mrharicot/monodepth

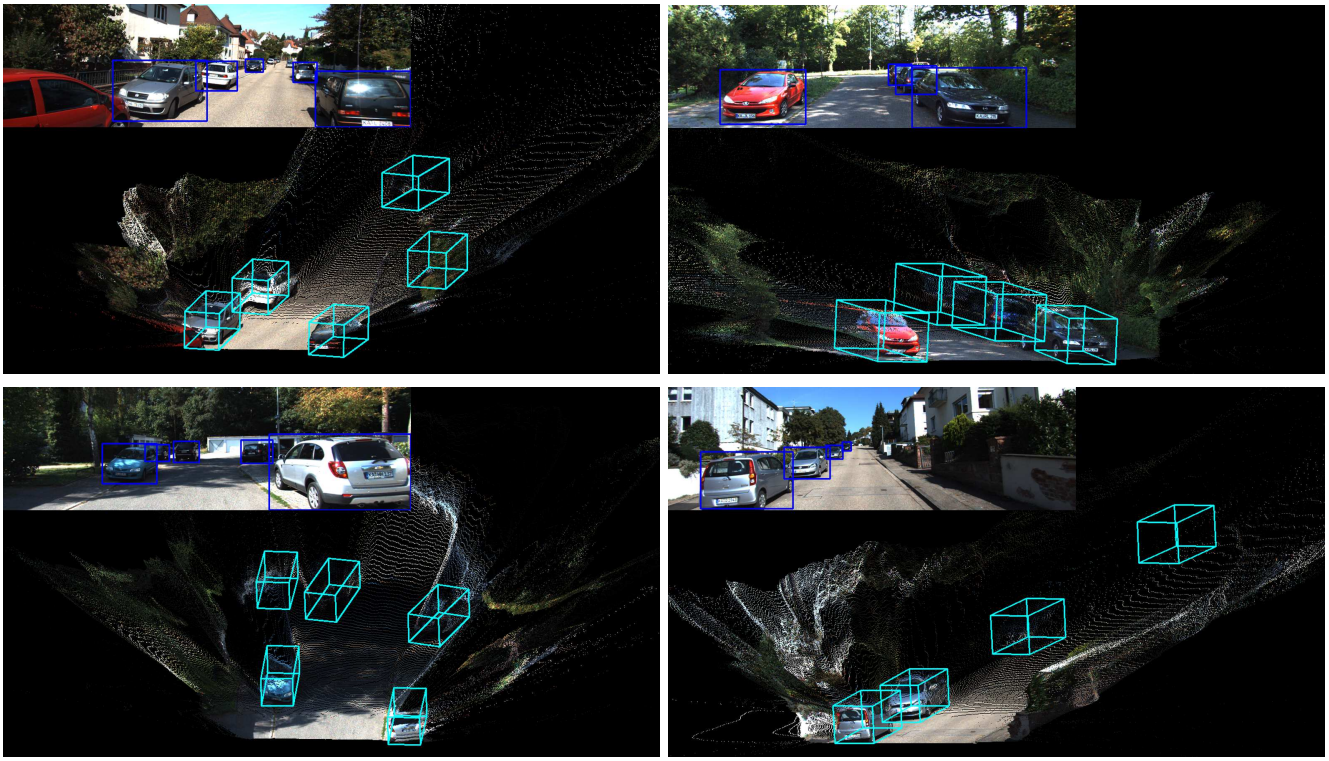

Figure 3. Visualization for 2D detection boxes, the projected 3D detection boxes on inferred point cloud from estimated disparity.

Comparison with Other Methods. As this work aims at monocular 3D object detection, our approach is mainly compared to other methods with only monocular images as input. Here three methods are chosen for comparisons: Mono3D [2], 3DOP [3] and Deep3DBox [26]. AP_{loc} and AP_{3D} for Mono3D [2] and 3DOP [3] are provided in [4]. For the state-of-the-art Deep3DBox method, we use the 3D detection results provided by [26] on *val2*[3]. The evaluation codes are from the official KITTI website[4]. As we can see the quantized results from Tables 1 and 2, our method outperforms the Mono3D and Deep3DBox methods. Compared to 3DOP, which uses stereo information, the proposed method provides better results in some circumstances. In addition, given an input image, the network directly output the 2D and 3D detection results without any extra computation. We also measure the effect of fusion methods in our framework. As we can see, both input fusion(**FV**) and feature fusion(**FF**) can improve the detection results. All experiments are done with estimation fusion for 3D localization, which is also the core part for the 3D detection pipeline. Typically, more fusion requires more parameters. However, the increased amount of parameters is a very small part compared to the total parameters of the network. In particular, changing 3-channel RGB input to 6-channel RGB+FV input only introduces 0.009% additional weights, and adding fusion of XYZ map only increases 0.79% weights. With the significant performance gain (5% - 8%), we think the increased parameters are negligible.

Disparity Effect. To show how the quality of the estimated disparity map might change the 3D object detection performance, we replace MonoDepth method with DispNet [25], which takes stereo pairs as input to estimate more accurate disparity. This stereo setting can be seen as a upper bound of the monocular input in our framework. As shown in Table 4, 3D detection accuracy increases significantly when using much more accurate disparity.

Qualitative Results. Apart from drawing the 2D detection boxes on images, we also project the 3D detection boxes on inferred point cloud from estimated disparity for better visualization. As shown in Figure 3, our approach can obtain accurate 2D bounding box, 3D orientation, dimension, and 3D location in various scenes with only monocular images.

5. Conclusion

We have proposed an approach for accurate 3D object detection with monocular images in this paper. We directly

[3] https://cs.gmu.edu/~amousavi/results/Output3DBoxes.zip

[4] http://kitti.is.tue.mpg.de/kitti/devkit_object.zip

Method	Type	IoU=0.5			IoU=0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [2]	Mono	30.50 / -	22.39 / -	19.16 / -	5.22 / -	5.19 / -	4.13 / -
3DOP [3]	Stereo	55.04 / -	41.25 / -	34.55 / -	12.63 / -	9.49 / -	7.59 / -
Deep3DBox [26]	Mono	- / 30.02	- / 23.77	- / 18.83	- / 9.99	- / 7.71	- / 5.30
Ours	Mono	46.69 / 50.99	28.69 / 30.52	26.18 / 24.44	11.14 / 12.69	6.59 / 8.03	5.43 / 5.99
Ours(+FF)	Mono	49.05 / 53.93	30.20 / 37.50	28.35 / 30.99	13.47 / 17.40	9.46 / 10.95	8.35 / 9.50
Ours(+FF+FV)	Mono	55.02 / 54.18	36.73 / 38.06	31.27 / 31.46	22.03 / 19.20	13.63 / 12.17	11.60 / 10.89

Table 1. **3D localization performance:** Average Precision (AP_{loc}) (in %) of bird's eye view boxes on KITTI val sets. Results on the two validation sets: *val1 / val2*. **FV** indicates the fusion between the front view feature maps and the RGB image. **FF** indicates the fusion between F_{max} and F_{mean}.

Method	Type	IoU=0.5			IoU=0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [2]	Mono	25.19 / -	18.20 / -	15.52 / -	2.53 / -	2.31 / -	2.31 / -
3DOP [3]	Stereo	46.04 / -	34.63 / -	30.09 / -	6.55 / -	5.07 / -	4.10 / -
Deep3DBox [26]	Mono	- / 27.04	- / 20.55	- / 15.88	- / 5.85	- / 4.10	- / 3.84
Ours	Mono	39.73 / 42.59	24.69 / 26.37	20.79 / 21.14	4.63 / 5.38	2.88 / 3.44	2.40 / 2.58
Ours(+FF)	Mono	41.47 / 43.91	26.77 / 29.29	22.45 / 23.58	6.26 / 7.60	4.57 / 4.84	4.13 / 4.42
Ours(+FF+FV)	Mono	47.88 / 44.57	29.48 / 30.03	26.44 / 23.95	10.53 / 7.85	5.69 / 5.39	5.39 / 4.73

Table 2. **3D detection performance:** Average Precision (AP_{3D}) (in %) of 3D boxes on KITTI val sets. Results on the two validation sets: *val1 / val2*. **FV** indicates the fusion between the front view feature maps and the RGB image. **FF** means the fusion between F_{max} and F_{mean}.

Metric	IoU=0.7		
	Easy	Moderate	Hard
2D AP	90.43	87.33	76.78
Orientation AP	90.35	87.03	76.37
Birdview's AP	13.73	9.62	8.22
3D AP	7.08	5.18	4.68

Table 3. Results for 2D, orientation, Bird's eye view and 3D AP(%) on the KITTI testing set.

Metric	Disparity	Data	IoU=0.5	IoU=0.7
Birdview's AP	MonoDepth	Mono	36.73	13.63
	DispNet	Stereo	53.65	19.54
3D AP	MonoDepth	Mono	29.48	5.69
	DispNet	Stereo	47.42	9.80

Table 4. Bird's eye view and 3D AP(%) on KITTI val1 sets(moderate).

extend the existing proposal-driven 2D detectors with the help of deep CNN features. With a single RGB image as input, the network can output all the descriptors for 2D and 3D objects in an end-to-end fashion. We have shown that our detection approach significantly outperforms state-of-the-art monocular approaches and even detectors with stereo information on particular evaluation index.

The main innovation in the framework is the multi-level fusion scheme. It utilizes a stand-alone module to estimate the disparity information, which ensures the accurate 3D localization and improve the detection performance. Although we only adopt region-based 2D detectors for extension in this paper, one-stage detector like SSD [24] utilizing the mechanism of *default box* also has a potential to be extended for 3D object detection, which deserves better research in the future.

Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant No. 2017YFB1002202, the National Natural Science Foundation of China under Grant No. 61471273 and No. 61771348, Wuhan Morning Light Plan of Youth Science and Technology under Grant No. 2017050304010302, and LIESMARS Special Research Funding.

References

[1] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image. In *CVPR*, 2017.

[2] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016.

[3] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *NIPS*, 2015.

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[5] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, 2014.

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[7] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[8] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.

[9] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Feifei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

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

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[14] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[15] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014.

[16] K. He, G. Gkioxari, P. Dollr, and R. Girshick. Mask R-CNN. In *ICCV*, 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[18] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017.

[19] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, 2016.

[20] B. Li, T. Zhang, and T. Xia. Vehicle detection from 3d lidar using fully convolutional network. In *Robotics: Science and Systems*.

[21] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[22] T. Y. Lin, P. Dollr, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollr. Focal loss for dense object detection. In *ICCV*, 2017.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[25] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[26] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *CVPR*, 2017.

[27] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *NIPS*, 2015.

[28] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016.

[29] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[30] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017.

[31] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[33] A. Shrivastava, A. Gupta, and R. B. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[35] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, pages 154–171, 2013.

[36] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *WACV*, 2017.

[37] F. Yang, W. Choi, and Y. Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016.

[38] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[39] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.