SDOD: Real-time Segmenting and Detecting 3D Object by Depth

Caiyi Xu[1], Jianping Xing[1], Yafei Ning[1], YongHong Chen[2], and Yong Wu[2]

[1] School of Microelectronics, Shandong University, Jinan, China
[2] City Public Passenger Transport Management Service Center of Jinan, China
cyxu@mail.sdu.edu.cn, {xingjp, ningyafei}@sdu.edu.cn,
chenyhjn@foxmail.com, wu.100@163.com

Abstract. Most existing instance segmentation methods only focus on 2D objects and are not suitable for 3D scenes such as autonomous driving. In this paper, we propose a model that splits instance segmentation and object detection into two parallel branches. We discretize the objects depth into depth categories (background set to 0, objects set to [1, K]), then the instance segmentation task has been transformed into a pixel-level classification task. Mask branch predicts pixel-level depth categories, 3D branch predicts instance-level depth categories, we produce instance mask by assigning pixels which have same depth categories to each instance. In addition, in order to solve the problem of imbalanced between mask labels and 3D labels in the KITTI dataset (200 for mask, 7481 for 3D), we introduce coarse mask generated by auto-annotation model to increase samples.

Keywords: Real-time Instance Segmentation, 3D Objects Detection

1 Introduction

Instance segmentation and 3D object detection from a RGB image are key tasks for autonomous driving, they help autonomous vehicles perceive complex surroundings. Instance segmentation is a combination of object detection and semantic segmentation, so it can be perfectly integrated with 2D object detection tasks, and this is what Mask R-CNN [11] has done. What about merging instance segmentation with 3D object detection tasks?

Different from 3D instance segmentation based on point cloud, our goal is to propose a network that when we input a image it can outputs 3D location, 3D bounding box and instance mask,as shown in Fig. 1.

State-of-the-art approaches to instance segmentation like Mask R-CNN [11] and FCIS [17] are two-stages, they focus on performance over speed. We focus on speed over performance, and propose a network that can segment instance and detect 3D objects by depth in real-time.

It is known to all that 2D detection is faster than 3D detection, semantic segmentation is faster than instance segmentation. Some state-of-the-art 3D objects detection frameworks speed up by splitting 3D tasks into multiple 2D related

Fig. 1. Input and output Fig.(a) input of SDOD; Fig.(b) instance-level depth categories map; Fig.(c) pixel-level depth categories map; Fig.(d) fused instance segmentation and detection output. In Fig (b) and (c) darker the color of a pixel/instance is, the greater the depth value of a pixel is, and the farther the pixel/instance is from us.

subtasks. MonoGRNet [24] propose a network composed of four task-specific subnetworks, responsible for 2D object detection, instance depth estimation, 3D localization and local corner regression. Inspired by this idea, we split 3D detection task into these four sub-tasks.

Now, there are three challenges left: 1) how to transform instance segmentation tasks into semantic segmentation tasks 2) how to combine 3D network with instance network efficiently 3) how to train 3D networks and instance network together.

We use depth to connect the 3D network with the instance network, and at the same time we use depth to transform instance segmentation into semantic segmentation. We believe that different instances have different depths. If they have the same depth, they are spatially separated. For example, if two cars in the image are obscured or overlaped, they mask have different depth, if they have same depth, we can distinguish them by 2D bounding box.

As shown in Fig. 2, we divide the network into two parallel branches: 3D branch and mask branch. We discretize the objects depth into depth categories, 3D branch predicts instance-level depth categories, mask branch predicts pixel-level depth categories. In this way, mask brach classifies each pixel, this is similar to semantic segmentation. Finally, we produce instance mask by assigning pixels which have same depth categories to each instance.

The instance segmentation labels in the KITTI dataset are not balanced with the 3D detection labels(200:7841), so we cannot train 3D and mask branches diectly. First We introduce the auto-annotation model trained on Cityscapes provided by polygon-rnn++ [1] to generate coarse masks on the KITTI dataset. Then add real depth to these coarse masks. Finally we use these masks to train mask branch.

Different from two-stage instance segmentation method, the 3D branch and mask branch of our network are parallel and proposal-free. This is the main reason that our network is real-time. Experiments on the KITTI dataset demonstrate that our network is effectively and real-time.

In general, our contributions of this paper are three-fold:

- Transform instance segmentation task into semantic segmentation task by discreting depth.
- Propose a network that combines 3D detection and instance segmentation and set them as parallel branches to speed up.

– Combine coarse masks with real depth to train mask branch to solve the problem of imbalanced labels.

The remainder of this paper is organized as follows: related works are introduced in section 2. Our method is well illustrated in section 3. In section 4, we conduct experiments to verify the effectiveness of the proposed framework. Finally, we conclude this paper in section 5.

2 Related Work

Our work is related to 3D object detection, depth estimation and instance segmentation. We mainly focus on the works of studying 3D detection, depth estimation and instance segmentation, while 2D detection is the base for them.

2D Object Detection 2D object detection methods based on convolutional neural networks [15] are mainly divided into one stages and two stages. Two stages method Faster R-CNN [26] use region proposal to generate ROI, performance well but slowly. One stage method YOLO [25] and SSD [20] focus on speed over performance, they are faster but less accurate. Methods above are anchor based, FCOS [30] use no anchor and performance better than anchor based methods. Multinet [29] proposes a non-proposed approach similar to YOLO, and uses RoiAlign [11] in rescaling layer to narrow the gap, we use this as our 2D detector.

3D Object Detection Existing methods based on RGB image includes multi-view method MV3D [6], single-view method MonoGRNet [24], MonoFENet [2] and RGB-Depth method. multi-view RGB method MV3D [6] takes the birds eye view and front view of point cloud as well as an image as input, RGB-Depth method takes RGB image and point cloud depth as input, single-view method MonoGRNet [24] takes a single RGB image as input, and proposes a network composed of four task-specific subnetworks, responsible for 2D object detection, instance depth estimation, 3D localization and local corner regression. We need a simple and real-time 3D detection method, and the input must be a single RGB image. Inspired by MonoGRNet [24], we split 3D detection task into these four sub-tasks.

Depth Estimation Depth estimation is mainly divided into monocular depth estimation and binocular depth estimation. Binocular depth estimation[13] uses the disparity of two image to estimate depth. Monocular depth estimation directly estimates the depth of each pixel. DORN [8] proposes a spacing-increasing discretization method to discretize continuous depth values, and transform depth estimation task into classification task. This method estimates the depth of each pixel in the image, it may not suiltable for 3D object detection. In the 3D branch, we estimate instance-level depth, which is the center depth of the object, and in the mask branch, we estimate pixel-level depth.

Instance Segmentation Existing methods range from one-stage instance segmentation approach YOLACT [3]; SOLO [31] to two-stage instance segmentation approach Mask-RCNN [11], Mask scoring r-cnn [12]. Mask R-CNN [11] is a representative two-stage instance segmentation approach that first generates ROI(region-of-interests) and then classifies and segments those ROI in the second stage. Mask scoring r-cnn [12] is an improvement on Mask R-CNN. Add a new branch to Mask R-CNN [11] to score the mask to predict a more accurate score.

Similar to two-stage object detection, two-stage instances are based on proposal, they performance well but slowly. SOLO [31] distinguish different instances by 2D location. It divided an input image of $H \times W$ into $S_x \times S_y$ grids, and do semantic segmentation in each grid, this is similar to the main idea of YOLO[25]. However, it only uses 2D location to distinguish different instance, performance not good for overlaped instance.

We discretize the object depth into depth categories (background set to 0, objects set to [1, K]), 3D branch predicts instance-level depth categories, mask branch predicts pixel-level depth categories. In this way, mask brach classifies each pixel, this is similar to semantic segmentation. Finally, we produce instance mask by assigning pixels which have same depth categories to each instance.

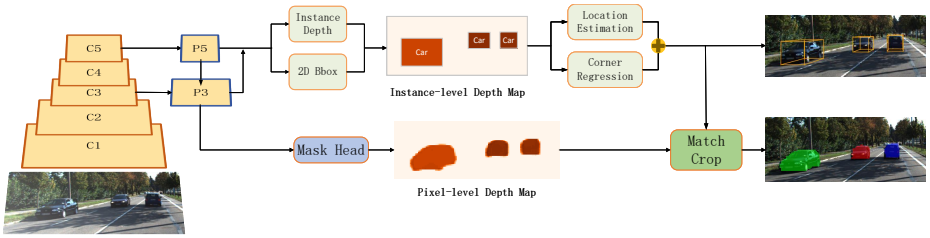

Fig. 2. SDOD Framework. SDOD consists of a backbone network and two parallel branches: a 3D branch and a mask branch. 3D branch includes four subnetworks: 2D detection, instance-level depth estimation, 3D location estimation and corners regression. We match and crop the instance-level depth category map generated by the 3D branch and the pixel-level depth category map generated by the mask branch, and finally get instance mask.

3 SDOD

In this section, we first introduce the overall structure of the proposed SDOD framework, as shown in Fig. 2, which consists of two parallel branches, a 3D branch and a mask branch, and then detail these two branches.

3.1 Depth Discretization

It is hard to directly regress the continue center depth g_d, we discretize continuous depth in to depth classes, and a particular class c_i can be assigned to each depth d. There are two discrete methods: linear method and non-linear method. Linear method means that the depth $d \in [d_{min}, d_{max}]$ is linearly divided into classes $c_i \in \{c_1, c_2, ..., c_K\}$. Note that the background is set to c_0 and the value is -1. Non-linear method choose a more complex mapping function for discretization e.g. SDNet [23] choose a logarithmic function and DORN [8] choose an exponential function.

Compared with the linear discrete method, the non-linear discrete method increases the proportion of difficult examples, making the model easier to train and converge. In this work, we spilt the depth into depth classes c_i with an exponential function 1 where K is the number of depth classes.

$$c_i = d_{min} \cdot \left(\frac{d_{max}}{d_{min}}\right)^{\frac{i-1}{K-1}} , i \in \{1, 2..., K\} \tag{1}$$

The left plot of Fig. 3 shows the linear and exponential discretization of the depths, the right plot shows the example frequency of linear and exponential discrete depth classes in the KITTI 3D object detection dataset. At the same time, we use depth error to measure the difficulty of the depth classes, and the red curve shows the depth class of the object is positive related to the difficulty of the object depth estimation. The exponential discrete method increases the proportion of hard examples, making the model easier to train and converge.

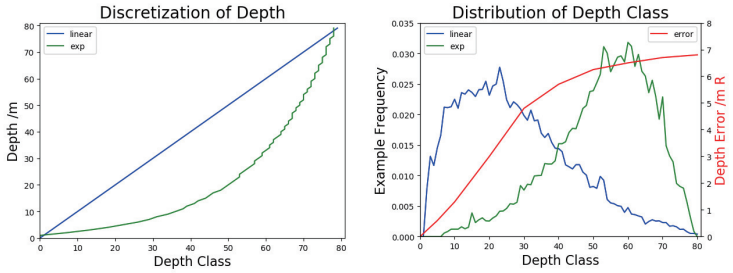

Fig. 3. The left plot shows the linear and exponential discretization of the depths with $K = 80$ in a depth interaval [2, 80]. The right plot shows the example frequency of linear and exponential discrete depth classes in the KITTI 3D object detection dataset. The depth error of example in different depth classes is also shown in the right plot. Depth error curve reflect the difficulty of the sample, and the the data comes from 3DOP [5].

3.2 3D Branch

We leverage the design of MonoGRNet [24], which decomposes the 3D objects detection into four subnetworks: 2D detection, instance-level depth estimation, 3D location estimation and corners regression.

2D Detection The 2D detection module is the basic module that extracts the region of interest from the feature map, classifies the object and regresses bounding boxes.

We use the design of detection in Multinet[29], which proposes a non-proposed approach similar to YOLO [25] and Overfeat [27]. To archive the good detection performance of proposal based detection systems, it uses RoiAlign [11] in rescaling layer. An input image of $H \times W$ is divided into $S_x \times S_y$ grids, and each grid is responsible for detecting objects whose center falls into the grid. Then each grid outputs the 2D bounding box B_{2d} and the class probabilities P_{cls}.

Instance-Level Depth Estimation Given a grid g, this module predicts the center depth g_d of the object in g and provides it for 3D location estimation and mask branch.

As shown in Fig. 2, the module takes P5 and P3 as the input feature map. Compared with P3, P5 has larger receptive field and lower resolution, it is less sensitive to location so we use P5 to generate a coarse depth estimation, and then fused with P3 to get accurate depth estimation. We apply several parallel atrous convolution with different rates to get multi-scale information, then fuse it with 2D bounding box to generate instance-level depth map.

Compared with the pixel-level depth estimation of the mask branch, the resolution of the module output is lower, which is an instance-level. For details of implementation, please refer to section 3.3.

3D Location Estimation The 3D location module uses the 2D coordinates (u, v) and the center depth d of the object to calculate the 3D location (x, y, z) by the following formula:

$$\begin{cases} u = x \cdot f_x + cx \\ v = y \cdot f_y + cy \\ d = z \end{cases} \tag{2}$$

f_x, f_y, cx, cy are camera parameters which can be obtained from the camera's internal parameter matrix C.

Corners Regression As illustrated in Fig.6, we first establish a coordinate system whose origin is the object center, and the x ,y ,z axis is parallel to the camera coordinate axis, and then regress the 8 corners of the object. Finally, we use the method of Deep3DBox [22] to calculate the object's length, width, height, and observation angle from 8 corner points. The length, width and observation angle will be used to calculate the depth threshold in Section 3.4.

3.3 Mask Branch

The Mask branch predicts pixel-level depth categories over the entire image, which is similar to semantic segmentation. Semantic segmentation classifies pixels based on the category they belong to, our mask branch classifies pixels based on the depth class to which the pixel belongs.

As shown in Fig. 4, the mask branch consists of atrous spatial pyramid pooling(ASPP) layers, fully convolutional (FCN) layers, fully connected layers(FC), and upsample layers. ASPP layers help to get multi-scale information, fully convolutional layers help to get semantic information, and fully connected layers help to transform semantic information into depth information. We have tried using convolutional layers instead of fully connected layers and encoding the depth category, but the performance is not good, refer to section 4.4 for details.

ASPP The input of the ASPP module is the P5 feature map, and its resolution is only 1/32 of the original image. In order to expand the receptive field of the input and obtain more semantic information, we use the ASPP module, inspired by dilated convolutions [32] and DeepLab v3++ [4].

As shown in Fig. 4, The ASPP module connects 1 convolutional layer and 3 atrous convolutional layers with rates of 2,4,8. The input size of the module is 39×12×512, after upsampled and concatenated it becomes to 156×48×256, then we throw the feature map into FCN layers.

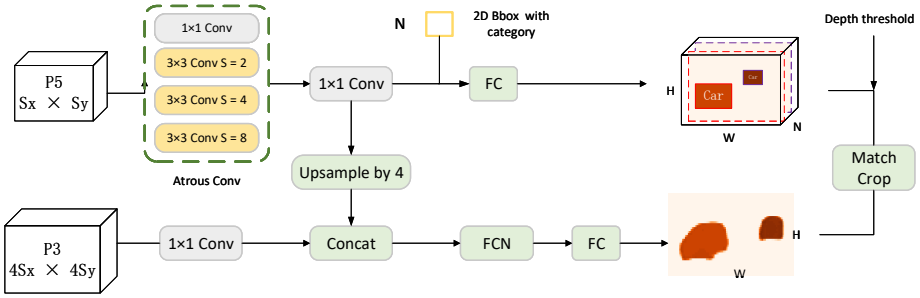

Fig. 4. Mask branch architecture. The mask branch consists of atrous spatial pyramid pooling(ASPP) layers, fully convolutional (FCN) layers, fully connected layers, and upsample layers. Note that the height H and width W in the picture are 1/4 of the original input picture.s

FCN and FC To get a pixel-level depth category map, we use a FCN module which is similar to mask branch in Mask R-CNN[11], proposed by FCN [21]. As shown in Fig 5, compared to Mask R-CNN [11] we have added a 1×1 convolution layer, which is responsible for depth classification of each pixel. K is the total number of depth categories in equation 1, we set it to 64. Mask branch does not

predict the target category of the pixel(car, pedestrian,cyclist), it is predicted in the 3D branch. In section 4.4, we have try to remove fully connected layers, but failed.

The FCN module finally outputs 1 pixel-level depth category map, as shown in Fig. 4. The darker the color of a pixel is, the greater the depth value of a pixel is, and the farther the pixel is from us. The size of the output image is 312×96, and the size of the original image is 1248×384.

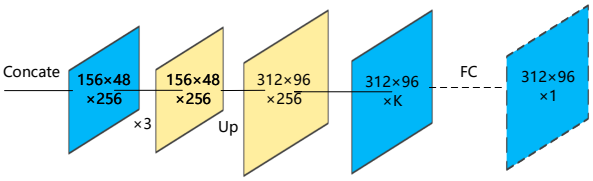

Fig. 5. FCN with FC The brown feature map is obtained by 3×3 conv, and the blue feature map is obtained by 1×1 conv. ×3 means that 3 conv layers are used, Up means that upsampling layer is used and K is the total number of depth categories. We apply fully connected layers to get the exact value of depth category that vary from 0 to K(background is 0). Note that mask branch does not predict the target category of the pixel(car, pedestrian,cyclist), it is predicted in the 3D branch.

3.4 Match And Crop

Instance segmentation requires each pixel to be assigned to a different instance. We need to assign each pixel in the pixel-level depth map $X = \{x_0, x_1, x_2...x_{N-1}\}$ to a set of instance $S = \{S_0, S_1, S_2...S_{M-1}\}$ in the 3D branch,and we treat this as a pixel matching task.

How to match pixels with instances? There are two conditions: first, the pixel must has the same depth category as the instance, and second, the pixel must has the same position as the instance. The first condition can be described by the following formula:

$$x_i \in S_k \Leftrightarrow \mid x_i - S_k \mid < \delta_k,\ i \in [0, N-1]\,,\ k \in [0, M-1] \tag{3}$$

x_i is the depth class of pixel i and $x_i \in [0, K]$, S_k is the depth class of the instance k, and δ_k is the depth threshold of the instance k.

As shown in Fig.6., each instance has only one depth class in the instance-level depth map, but each instance may has multiple depth classes in the pixel-level depth map. So we set a depth threshold δ_k for each instance, which is calculated by the following formula:

$$\delta_k = (K-1) \cdot \log_{d_{max}/d_{min}} \frac{c_k}{c_k - \triangle d_k} \tag{4}$$

$$\triangle d_k = \frac{1}{2} w_k \mid \cos\theta_k \mid + \frac{1}{2} l_k \mid \sin\theta_k \mid, \quad \theta_k \in [-\pi, \pi] \tag{5}$$

c_k is the depth class of instance k, δ_k is the depth margin and is shown in Fig.6. w_k, l_k and θ_k are the width length and observation angle of the instance, which can be obtained from the cornesr regression module. The derivation of equation 5 can be seen in the supplementary material.

The second condition can be transformed into a crop operation. Crop operation means using the 2D bounding box to crop the pixel-level depth map, and this can improve the accuracy of the mask. During training, we use truth bounding boxes to crop the depth map.

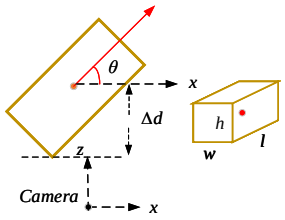

Fig. 6. Coordinate system and geometric constraints. The left one is a bird's eye view and shows the position of the target and the camera. $\triangle d$ is the depth threshold and can be calculated by equation 5 , θ is the deflection angle of the object.

3.5 Loss Function

Here we have determined independent loss functions for each module and joint loss functions for the entire network.

2D Detection Loss. 2D detection includes classification loss L_{cls} and box regression loss L_{box}, they are defined in the same as in Multinet [29]. Due to the imbalance of samples between classes in the KITTI dataset, we used focal loss [19] to modify the classification loss. The total loss of 2D inspection is

$$L_{2d} = w_1 L_{cls} + w_2 L_{box} \tag{6}$$

where w_1 and w_2 are weight coefficients, when we trained 2D detection only, we set $w_1 = w_2 = 1$.

Instance-Level Depth Loss. We use L1 loss as instance-level depth loss:

$$L_d = \sum_{i=1}^{n} \mid d_i - \hat{d}_i \mid \tag{7}$$

where n is the numbel of cell, di is the ground truth of cell i, $\hat{d}_i$ is the prediction of cell i.

3D Location and Corner Loss. We use L1 loss as corner loss and location loss, and they are the same as in MonoGRNet [24].

Pixel-Level Depth Loss. When fully connected layer is used, we use the L1 loss the same as the instance-level depth loss.

$$L_{mask} = \sum_{i=1}^{m} \mid M_i - \hat{p_i} \mid \tag{8}$$

where n is the numbel of pixel, M_i is the ground truth of pixel-level depth categories, which is deined by equation 9 , p_i is the prediction category of pixel i. We also tried L2 loss, CE loss and Focal loss, and finally we found that L1 loss performed better. We think that the smaller the object is, the farther it is, the greater its depth value is and the greater the loss is. And this is why long-distance object can be detected well.

4 Experiments

Note that only the KITTI dataset [9] has both 3D object detection and instance segmentation challenging in the main autonomous driving dataset, the Cityscapes dataset [7] and the BDDV dataset [33] are lack of 3D data. So we evaluate our network on the KITTI dataset.

4.1 Datasets and Coarse Mask

Datasets The KITTI dataset has 7841 training images and 7581 test images with calibrated camera parameters for 3D object detection challenge. However, due to the difficulty of instance segmentation labeling, there are only 200 labeled training images and 200 unlabeled testing images for instance segmentation challenge. In addition, the 3D object detection task evaluates on 3 types of targets(car,pedestrian,cyclist), and instance segmentation task evaluates on 8 types of targets(car,pedestrian,cyclist,truck,bus,train,bicycle,motorcycle).

We think that car is the main vehicle in the autonomous driving scenes, motorcycle,bicycle and cyclist can be merged into 1 category, so only car, cyclist, and pedestrian will be detected by our network. We evaluate cars, pedestrian, and cyclists on both 3D object detection and instance segmentation tasks.

The number of 3D objects in the KITTI dataset is slightly less than the number of instance masks. 3D detection datasets ignore targets that are beyond the Lidar detection range, and some of them are not ignored in instance segmentation dataset. In our work we take the 3D dataset as the benchmark, although this will bring some performance loss.

Coarse Mask In order to solve the problem of imbalanced between mask labels and 3D labels in the KITTI dataset, we introduce coarse mask generated by auto-annotation model to increase samples of instance segmentation. Polygon-RNN++ [1] is a state-of-the-art auto-annotation model which inputs 2D bounding boxes and outputs instance masks. It is trained on the instance-level semantic labeling task of the Cityscapes dataset. We use 200 labeled training images to evaluate the accuracy of the coarse mask, results are shown in Table 1.

We did not directly train the mask branch with coarse labels , but superimposed the real depth value with it for training:

$$p_k = i_k \times m_k,\ i_k \in [1, K]\,,\ m_k \in \{0, 1\} \tag{9}$$

i_k is the real depth category of instance k, which can be calculated from equation 1. m_k is the coarse mask of instance i, with a value of 0 or 1, p_k is the final label for mask branch.

When training the mask branch, first, we train mask branch and 3D branch together with coarse masks for 120K iterations, and then train mask branch only with fine masks for 40K iterations.

Table 1. Accuracy of coarse mask generated by Polygon-RNN++. Note that Polygon-RNN++ was trained on the Cityscapes dataset rather than the KITTI dataset.

	car	pedestrian	cyclist	average
AP	40.1	36.3	35.1	37.2
AP_{50}	56.7	50.6	50.3	52.5

Evaluation Metrics For evaluating 3D detection performance, we follow the official settings of KITTI benchmark to evaluate the 3D Average Precision(AP_{3d}). For evaluating instance segmentation performance, we follow the official settings of KITTI benchmark to evaluate the Average Precision on the region level (AP) and Average Precision with 50%(AP_{50}). Note that we only evaluate three types of object: car, pedestrian, and cyclist, the reason has been discussed in the Dataset section.

4.2 Implement Details

Network. The architecture of SDOD is shown in Figure 2. VGG-16[28] is employed as the backbone, but without the FC layers. FPN [18] is used to solve the problem of multi-scale detection.

Training. 2D detecor should be trained first, we set $w_1 = w_2 = 1$ in the loss functions and initialize VGG-16 with the pretrained weights on ImageNet. We trained 2D detector for 150K iterations with the Adam optimizer[14], and L2

regularization is used with a decay rate 1e-5. Then 3D branch and mask branch are trained for 120K iterations with the Adam optimizer. At this stage, coarse mask generated by Polygon-RNN++ [1] is used. Finally, we continue to train the network with fine masks for 40K iterations with SGD optimizer. We set batch size to 4 , learning rate to 1e-5 and dropout rate to 0.5 throughout the training. The network is trained using a single GPU of NVidia GTX 2080TI.

4.3 Experimental Results

In this section we do relevant experiments to verify the effectiveness of our method. We evaluate the AP and AP_{50} of instance segmentation tasks, and the results are shown in Table 2. We compare our method with Mask R-CNN [11] and Lklnet [10]. Mask R-CNN is the main method for two-stage instance segmentation, it has high accuracy but slow speed. Our method is a one-stage instance segmentation method, it has lower accuracy than the one-stage method, but it is faster and more suiltable for autonomous driving.

Table 2. Instance segmentation mask AP on KITTI. Mask R-CNN is trained on the KITTI dataset and inference with the environment of 1 core 2.5 Ghz (C/C++). All parameters are tuned in COCO dataset in Mask R-CNN*. Ours time is the total time of 3D detection and instance segmentation.

Method	Backbone	Training	AP_{50}	AP	Time
Mask R-CNN [11]	ResNet101+FPN	KITTI	39.14	20.26	1s
Mask R-CNN* [11]	ResNet101+FPN	COCO	19.86	8.80	0.5s
Lklnet [10]	ResNet101+FPN	KITTI	22.88	8.05	0.15s
Ours	VGG16+FPN	KITTI	37.35	20.38	**0.054s**

Specific to different objects, the results of our method are shown in Table 3. The car has the highest accuracy and the rider has the lowest accuracy.

Table 3. Specific accuracy for each category of our method. Note that our method only evaluate AP and AP_{50} of car, pedestrian, and cyclist.

	car	pedestrian	cyclist	average
AP	23.36	19.73	18.15	20.38
AP_{50}	48.59	33.21	30.26	37.35

We also evaluated our method on 3D object detection tasks, and the results are shown in Table 4. We compared our method with MonoFENet [2] and MonoPSR [16]. Thanks to splitting the 3D detection into four sub-networks, we can see our method is the fastest and real time.

Table 4. 3D detection performance. All results is evaluated using the AP_{3D} at 0.7 3D IoU threshold for car class. Difficulties are define in KITTI [9]. Ours-3D time only includes 3D inference time and does not include instance segmentation time.

	Easy	Moderate	Hard	Time
MonoFENet [2]	8.35%	5.14%	4.10%	0.15s
MonoPSR [16]	10.76%	7.25%	5.85%	0.20s
Ours-3D	9.63%	5.77%	4.25%	**0.035s**

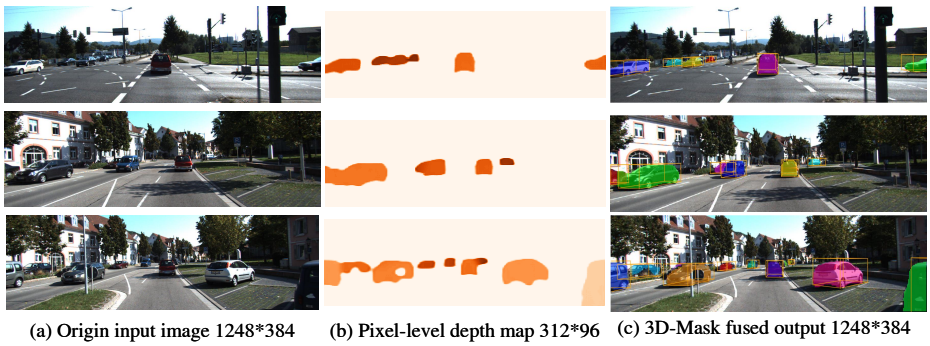

Fig. 7. Results Samples Results on KITTI datasets. Fig.(a) input of SDOD; Fig.(b) pixel-level depth categories map; Fig.(c) fused instance segmentation and detection output. We use random colors for instance segmentation. In Fig (b) the darker the color of a pixelis, the greater the depth value of a pixel is, and the farther the pixel is from us.

4.4 Ablation Study

Depth Category K In the instance-level and pixel-level depth estimation, we use equation 1 to discretize the depth into K categories. To illustrate the sensitivity to the number of categories, we set K to different values for comparison experiments, and the results are shown in Table5. We can see that neither too few nor too many depth categories are rational: too few depth categories cause large error, while too many depth categories lose the advantage of discretization.

Table 5. Ablation Study Results.

FC	Depth threshold	K	AP	AP_{50}	Time
Y	Y	32	18.90	33.23	0.049s
Y	Y	64	20.38	37.35	0.054s
Y	Y	96	19.84	37.34	0.058s
Y	N	64	19.27	35.84	0.054s
N	Y	64	18.81	32.15	0.052s

Depth threshold We use object's length and observation angle to calculate depth threshold, so 3D detection is necessary for instance segmentation. If we set length and observation angle to constant and then we can remove corners regression module and location estimation module. We use the typical values of the target: the length of car is set to 3.5m, person to 0.5m, cyclist to 1m and all objects observation angle is set to be 0. The result is shown in Fig.5, we can see that depth threshold helps to improve the mask AP from 19.27 to 20.38.

Fully connected layers We apply fully connected layers to get the exact value of depth category that vary from 0 to K(background is 0). We try to remove the FC layer and encode the depth category in one-hot form, e.g. if depth category is 3 just output $[1, 1, 1, 0, ..., 0]$. Then the size of pixel-level feature map is 312×96×K, and we use pixel-wise binary cross entropy(BCE) loss to replace L1 loss. The result is shown in Fig.5, we can see that fully connected layer is necessary and helps to improve the mask AP from 18.81 to 20.38.

4.5 Error Analysis

To quantitatively understand SDOD for mask prediction, we perform two error analysis. First, we replace the predicted pixel-level depth map with the ground truth value and coarse value to evaluate the effect of the mask branch on the result. Specifically, for each picture, we use equation 1 to convert the given masks into a pixel-level depth map. As shown in Table 6, if we replace the predicted pixel-level depth map with coarse mask the AP increase to 35.37, if we replace

with ground truth the AP increase to 61.18. This shows that there is still room for improvement in the mask branch.

Second, we replace the 3D predicted results with 3D ground truth values, which include 2D boxes, 3D depth values, and depth thresholds. As reported in in Table 6, the AP increase from 20.38 to 20.69, this shows that 3D branch has less effect on the final mask prediction.

Table 6. Error analysis.

	baseline	coarse mask	gt mask	gt 3D
AP	20.38	35.37	61.18	20.69
AP_{50}	37.35	49.25	61.42	37.98

5 Conclusion

In this paper, we propose the SDOD framework to combines 3D detection and instance segmentation by depth. In our framework we divide the network into two parallel branches: 3D branch and mask branch, and they are proposal-free. We transform instance segmentation tasks into semantic segmentation tasks by discreting the object depth into depth categories. We combine coarse masks with real depth to train mask branch to solve the problem of imbalanced labels. And our network conducts instance segmentation and 3D object detection in real time.

6 Supplementary material: Depth Treshold Derivation

In section 3.4 match and crop, we set a depth threshold δ_k for each instance, which is calculated by the following formula:

$$\delta_k = (K-1) \cdot \log_{d_{max}/d_{min}} \frac{c_k}{c_k - \triangle d_k} \tag{10}$$

$$\triangle d_k = \frac{1}{2} w_k \mid \cos\theta_k \mid + \frac{1}{2} l_k \mid \sin\theta_k \mid, \quad \theta_k \in [-\pi, \pi] \tag{11}$$

Depth discretization In section 3.1, we spilt the depth into depth classes c_i with an exponential formula 12 where K is the number of depth classes. Transforming formula 12 can get formula 13.

$$c_i = d_{min} \cdot \left(\frac{d_{max}}{d_{min}}\right)^{\frac{i-1}{K-1}}, i \in \{1, 2..., K\} \tag{12}$$

$$i = 1 + (K-1) \cdot \log_{d_{max}/d_{min}} \frac{c_i}{d_{min}} \tag{13}$$

Depth treshold Then we prove formula 11 with formula 13. Figure 1 shows the four possible positions of the object corresponding to the value of θ from $-\pi$ to π.

$$\begin{aligned} \triangle d_k &= AC + AB \\ &= OA\cdot \mid \cos\theta_k \mid + AD\cdot \mid \sin\theta_k \mid \\ &= \frac{1}{2} w_k \mid \cos\theta_k \mid + \frac{1}{2} l_k \mid \sin\theta_k \mid \end{aligned} \tag{14}$$

$$\begin{aligned} \delta_k &= i_O - i_D \\ &= 1 + (K-1)\cdot \log_{d_{max}/d_{min}} \frac{c_{i_O}}{d_{min}} - (1 + (K-1)\cdot \log_{d_{max}/d_{min}} \frac{c_{i_D}}{d_{min}}) \\ &= (K-1)\cdot \log_{d_{max}/d_{min}} \frac{c_{i_O}}{c_{i_D}} \\ &= (K-1)\cdot \log_{d_{max}/d_{min}} \frac{c_k}{c_k - \triangle d_k} \end{aligned} \tag{15}$$

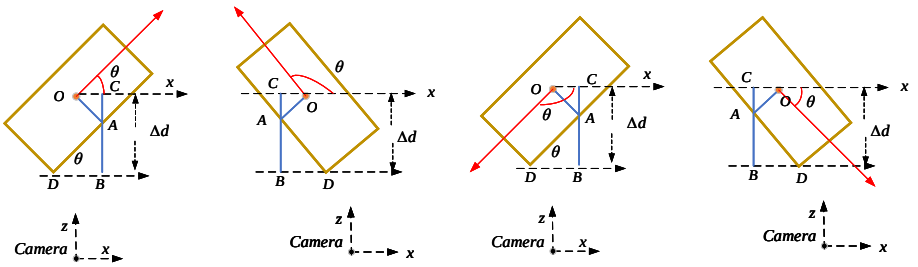

Fig. 8. Coordinate system and geometric constraints. It shows the position of the target and the camera. $\triangle d$ is the depth threshold and can be calculated by formula 11 , θ is the deflection angle of the object.

References

1. Acuna, D., Ling, H., Kar, A., Fidler, S.: Efficient interactive annotation of segmentation datasets with polygon-rnn++. In: CVPR (2018)
2. Bao, W., Xu, B., Chen, Z.: Monofenet: Monocular 3d object detection with feature enhancement networks. IEEE Transactions on Image Processing (2019)
3. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: Yolact++: Better real-time instance segmentation. arXiv preprint arXiv:1912.06218 (2019)
4. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV (2018)
5. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals for accurate object class detection. In: NIPS (2015)
6. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: CVPR (2017)
7. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
8. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: CVPR (2018)
9. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: ICCV (2012)
10. Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K.: Detectron. `https://github.com/facebookresearch/detectron` (2018)
11. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017)
12. Huang, Z., Huang, L., Gong, Y., Huang, C., Wang, X.: Mask scoring r-cnn. In: CVPR (2019)
13. Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A.: End-to-end learning of geometry and context for deep stereo regression. In: ICCV (2017)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
16. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In: CVPR (2019)
17. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: CVPR (2017)
18. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
19. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
20. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (2016)
21. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: ICCV (2015)
22. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR (2017)
23. Ochs, M., Kretz, A., Mester, R.: Sdnet: Semantically guided depth estimation network. In: German Conference on Pattern Recognition. pp. 288–302. Springer (2019)

24. Qin, Z., Wang, J., Lu, Y.: Monogrnet: A geometric reasoning network for monocular 3d object localization. In: AAAI (2019)
25. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
26. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS (2015)
27. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 (2013)
28. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
29. Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., Urtasun, R.: Multinet: Real-time joint semantic reasoning for autonomous driving. In: 2018 IEEE Intelligent Vehicles Symposium (IV). pp. 1013–1020. IEEE (2018)
30. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: ICCV (2019)
31. Wang, X., Kong, T., Shen, C., Jiang, Y., Li, L.: Solo: Segmenting objects by locations. arXiv preprint arXiv:1912.04488 (2019)
32. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
33. Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., Darrell, T.: Bdd100k: A diverse driving video database with scalable annotation tooling. arXiv preprint arXiv:1805.04687 (2018)