Multi-View 3D Object Detection Network for Autonomous Driving

Xiaozhi Chen[1], Huimin Ma[1], Ji Wan[2], Bo Li[2], Tian Xia[2]
[1]Department of Electronic Engineering, Tsinghua University
[2]Baidu Inc.
{chenxz12@mails., mhmpub@}tsinghua.edu.cn, {wanji, libo24, xiatian}@baidu.com

Abstract

This paper aims at high-accuracy 3D object detection in autonomous driving scenario. We propose Multi-View 3D networks (MV3D), a sensory-fusion framework that takes both LIDAR point cloud and RGB images as input and predicts oriented 3D bounding boxes. We encode the sparse 3D point cloud with a compact multi-view representation. The network is composed of two subnetworks: one for 3D object proposal generation and another for multi-view feature fusion. The proposal network generates 3D candidate boxes efficiently from the bird's eye view representation of 3D point cloud. We design a deep fusion scheme to combine region-wise features from multiple views and enable interactions between intermediate layers of different paths. Experiments on the challenging KITTI benchmark show that our approach outperforms the state-of-the-art by around 25% and 30% AP on the tasks of 3D localization and 3D detection. In addition, for 2D detection, our approach obtains 14.9% higher AP than the state-of-the-art on the hard data among the LIDAR-based methods.

1. Introduction

3D object detection plays an important role in the visual perception system of Autonomous driving cars. Modern self-driving cars are commonly equipped with multiple sensors, such as LIDAR and cameras. Laser scanners have the advantage of accurate depth information while cameras preserve much more detailed semantic information. The fusion of LIDAR point cloud and RGB images should be able to achieve higher performance and safty to self-driving cars.

The focus of this paper is on 3D object detection utilizing both LIDAR and image data. We aim at highly accurate 3D localization and recognition of objects in the road scene. Recent LIDAR-based methods place 3D windows in 3D voxel grids to score the point cloud [25, 6] or apply convolutional networks to the front view point map in a dense box prediction scheme [16]. Image-based methods [4, 3] typically first generate 3D box proposals and then perform region-based recognition using the Fast R-CNN [9] pipeline. Methods based on LIDAR point cloud usually achieve more accurate 3D locations while image-based methods have higher accuracy in terms of 2D box evaluation. [10, 7] combine LIDAR and images for 2D detection by employing early or late fusion schemes. However, for the task of 3D object detection, which is more challenging, a well-designed model is required to make use of the strength of multiple modalities.

In this paper, we propose a Multi-View 3D object detection network (**MV3D**) which takes multimodal data as input and predicts the full 3D extent of objects in 3D space. The main idea for utilizing multimodal information is to perform region-based feature fusion. We first propose a multi-view encoding scheme to obtain a compact and effective representation for sparse 3D point cloud. As illustrated in Fig. 1, the multi-view 3D detection network consists of two parts: a *3D Proposal Network* and a *Region-based Fusion Network*. The 3D proposal network utilizes a bird's eye view representation of point cloud to generate highly accurate 3D candidate boxes. The benefit of 3D object proposals is that it can be projected to any views in 3D space. The multi-view fusion network extracts region-wise features by projecting 3D proposals to the feature maps from mulitple views. We design a deep fusion approach to enable interactions of intermediate layers from different views. Combined with drop-path training [14] and auxiliary loss, our approach shows superior performance over the early/late fusion scheme. Given the multi-view feature representation, the network performs *oriented* 3D box regression which predict accurate 3D location, size and orientation of objects in 3D space.

We evaluate our approach for the tasks of 3D proposal generation, 3D localization, 3D detection and 2D detection on the challenging KITTI [8] object detection benchmark. Experiments show that our 3D proposals significantly outperforms recent 3D proposal methods 3DOP [4] and Mono3D [3]. In particular, with only 300 proposals, we obtain 99.1% and 91% *3D recall* at Intersection-over-Union (IoU) threshold of 0.25 and 0.5, respectively. The

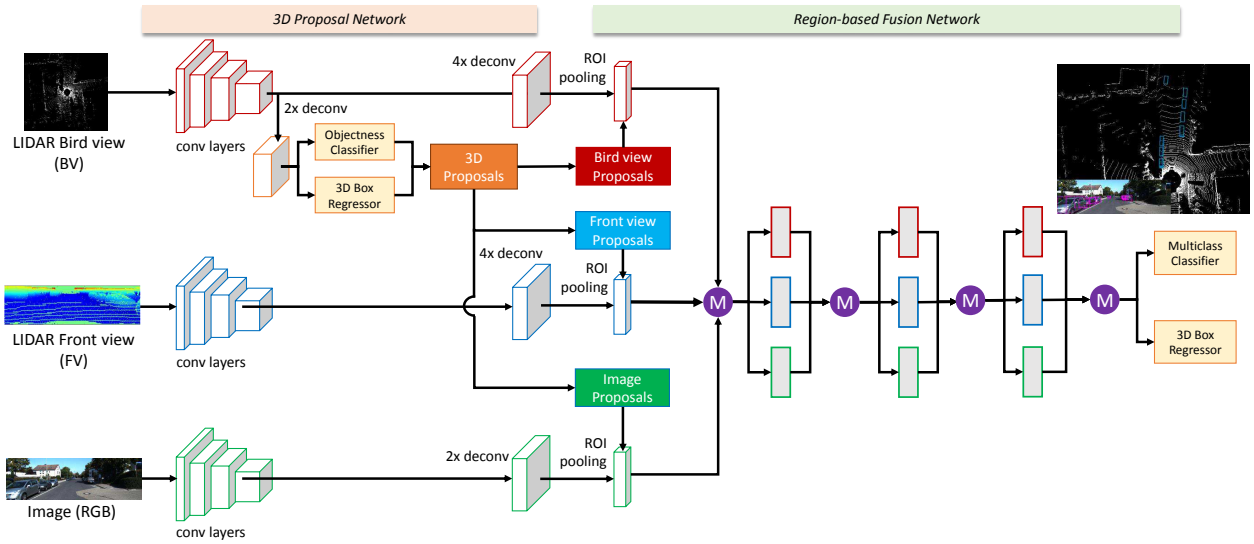


Figure 1: **Multi-View 3D object detection network (MV3D):** The network takes the bird's eye view and front view of LIDAR point cloud as well as an image as input. It first generates 3D object proposals from bird's eye view map and project them to three views. A deep fusion network is used to combine region-wise features obtained via ROI pooling for each view. The fused features are used to jointly predict object class and do oriented 3D box regression.

LIDAR-based variant of our approach achieves around 25% higher accuracy in 3D localization task and 30% higher 3D Average Precision (AP) in the task of 3D object detection. It also outperforms all other LIDAR-based methods by 14.9% AP for 2D detection on KITTI's hard test set. When combined with images, further improvements are achieved over the LIDAR-based results.

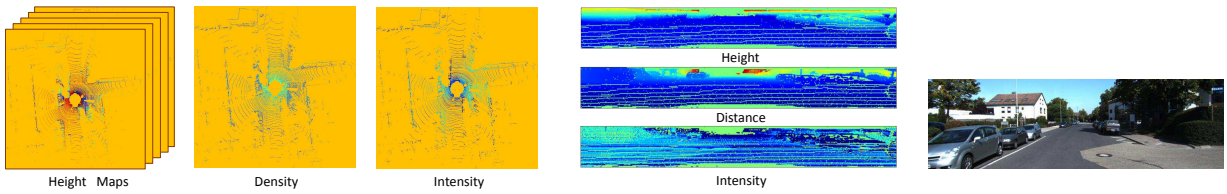
2. Related Work

We briefly review existing work on 3D object detection from point cloud and images, multimodal fusion methods and 3D object proposals.

3D Object Detection in Point Cloud. Most existing methods encode 3D point cloud with voxel grid representation. Sliding Shapes [21] and Vote3D [25] apply SVM classifiers on 3D grids encoded with geometry features. Some recently proposed methods [22, 6, 15] improve feature representation with 3D convolutions.networks, which, however require expensive computations. In addition to the 3D voxel representation, VeloFCN [16] projects point cloud to the front view, obtaining a 2D point map. They apply a fully convolutional network on the 2D point map and predict 3D boxes densely from the convolutional feature maps. [23, 17, 11] investigate volumetric and multi-view representation of point cloud for 3D object classification. In this work, we encode 3D point cloud with multi-view feature maps, enabling region-based representation for multimodal fusion.

3D Object Detection in Images. 3DVP [27] introduces 3D voxel patterns and employ a set of ACF detectors to do 2D detection and 3D pose estimation. 3DOP [4] reconstructs depth from stereo images and uses an energy minimization approach to generate 3D box proposals, which are fed to an R-CNN [9] pipeline for object recognition. While Mono3D [3] shares the same pipeline with 3DOP, it generates 3D proposals from monocular images. [30, 31] introduces a detailed geometry representation of objects using 3D wireframe models. To incorporate temporal information, some work[5, 20] combine structure from motion and ground estimation to lift 2D detection boxes to 3D bounding boxes. Image-based methods usually rely on accurate depth estimation or landmark detection. Our work shows how to incorporate LIDAR point cloud to improve 3D localization.

Multimodal Fusion Only a few work exist that exploit multiple modalities of data in the context of autonomous driving. [10] combines images, depth and optical flow using a mixture-of-experts framework for 2D pedestrian detection. [7] fuses RGB and depth images in the early stage and trains pose-based classifiers for 2D detection. In this paper, we design a deep fusion approach inspired by FractalNet [14] and Deeply-Fused Net [26]. In FractalNet, a base module is iteratively repeated to construct a network with exponentially increasing paths. Similarly, [26] constructs deeply-fused networks by combining shallow and deep subnetworks. Our network differs from them by using the same base network for each column and adding auxiliary paths and losses for regularization.



(a) Bird's eye view features (b) Front view features (c) RGB image

Figure 2: Input features of the MV3D network.

3D Object Proposals Similarly to 2D object proposals [24, 32, 2], 3D object proposal methods generate a small set of 3D candidate boxes in order to cover most of the objects in 3D space. To this end, 3DOP [4] designs some depth features in stereo point cloud to score a large set of 3D candidate boxes. Mono3D [3] exploits the ground plane prior and utilizes some segmentation features to generate 3D proposals from a single image. Both 3DOP and Mono3D use hand-crated features. Deep Sliding Shapes [22] exploits more powerful deep learning features. However, it operates on 3D voxel grids and uses computationally expensive 3D convolutions. We propose a more efficient approach by introducing a bird's eye view representation of point cloud and employing 2D convolutions to generate accurate 3D proposals.

3. MV3D Network

The MV3D network takes a multi-view representation of 3D point cloud and an image as input. It first generates 3D object proposals from the bird's eye view map and deeply fuses multi-view features via region-based representation. The fused features are used for category classification and oriented 3D box regression.

3.1. 3D Point Cloud Representation

Existing work usually encodes 3D LIDAR point cloud into a 3D grid [25, 6] or a front view map [16]. While the 3D grid representation preserves most of the raw information of the point cloud, it usually requires much more complex computation for subsequent feature extraction. We propose a more compact representation by projecting 3D point cloud to the bird's eye view and the front view. Fig. 2 visualizes the point cloud representation.

Bird's Eye View Representation. The bird's eye view representation is encoded by height, intensity and density. We discretize the projected point cloud into a 2D grid with resolution of 0.1m. For each cell, the height feature is computed as the maximum height of the points in the cell. To encode more detailed height information, the point cloud is devided equally into M slices. A height map is computed for each slice, thus we obtain M height maps. The intensity feature is the reflectance value of the point which has the maximum height in each cell. The point cloud density indicates the number of points in each cell. To normalize the feature, it is computed as $min(1.0, \frac{log(N+1)}{log(64)})$, where N is the number of points in the cell. Note that the intensity and density features are computed for the whole point cloud while the height feature is computed for M slices, thus in total the bird's eye view map is encoded as $(M+2)$-channel features.

Front View Representation. Front view representation provides complementary information to the bird's eye view representation. As LIDAR point cloud is very sparse, projecting it into the image plane results in a sparse 2D point map. Instead, we project it to a cylinder plane to generate a dense front view map as in [16]. Given a 3D point $p = (x, y, z)$, its coordinates $p_{fv} = (r, c)$ in the front view map can be computed using

$$\begin{aligned} c &= \lfloor \text{atan2}(y, x)/\Delta\theta \rfloor \\ r &= \lfloor \text{atan2}(z, \sqrt{x^2 + y^2})/\Delta\phi \rfloor, \end{aligned} \tag{1}$$

where $\Delta\theta$ and $\Delta\phi$ are the horizontal and vertical resolution of laser beams, respectively. We encode the front view map with three-channel features, which are height, distance and intensity, as visualized in Fig. 2.

3.2. 3D Proposal Network

Inspired by Region Proposal Network (RPN) which has become the key component of the state-of-the-art 2D object detectors [18], we first design a network to generate 3D object proposals. We use the bird's eye view map as input. In 3D object detection, The bird's eye view map has several advantages over the front view/image plane. First, objects preserve physical sizes when projected to the bird's eye view, thus having small size variance, which is not the case in the front view/image plane. Second, objects in the bird's eye view occupy different space, thus avoiding the occlusion problem. Third, in the road scene, since objects typically lie on the ground plane and have small variance in vertical location, the bird's eye view location is more crucial to obtaining accurate 3D bounding boxes. Therefore, using explicit bird's eye view map as input makes the 3D location prediction more feasible.

Given a bird's eye view map. the network generates 3D box proposals from a set of 3D prior boxes. Each 3D box

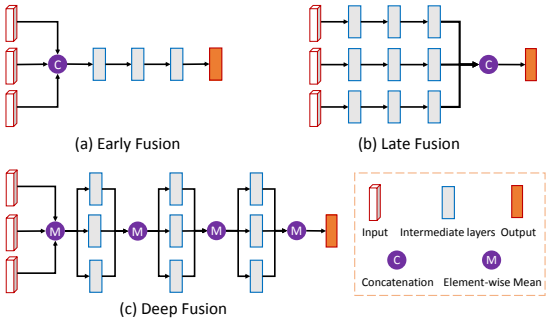


Figure 3: **Architectures of different fusion schemes:** We instantiate the join nodes in early/late fusion with *concatenation* operation, and deep fusion with *element-wise mean* operation.

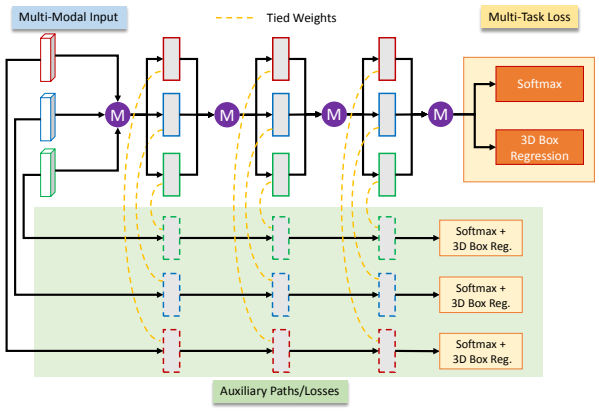


Figure 4: **Training strategy for the Region-based Fusion Network:** During training, the bottom three paths and losses are added to regularize the network. The auxiliary layers share weights with the corresponding layers in the main network.

is parameterized by (x, y, z, l, w, h), which are the center and size (in meters) of the 3D box in LIDAR coordinate system. For each 3D prior box, the corresponding bird's eye view anchor $(x_{bv}, y_{bv}, l_{bv}, w_{bv})$ can be obtained by discretizing (x, y, l, w). We design N 3D prior boxes by clustering ground truth object sizes in the training set. In the case of car detection, (l, w) of prior boxes takes values in $\{(3.9, 1.6), (1.0, 0.6)\}$, and the height h is set to 1.56m. By rotating the bird's eye view anchors 90 degrees, we obtain $N = 4$ prior boxes. (x, y) is the varying positions in the bird's eye view feature map, and z can be computed based on the camera height and object height. We do not do orientation regression in proposal generation, whereas we left it to the next prediction stage. The orientations of 3D boxes are restricted to $\{0°, 90°\}$, which are close to the actual orientations of most road scene objects. This simplification makes training of proposal regression easier.

With a disretization resolution of 0.1m, object boxes in the bird's eye view only occupy 5∼40 pixels. Detecting such extra-small objects is still a difficult problem for deep networks. One possible solution is to use higher resolution of the input, which, however, will require much more computation. We opt for feature map upsampling as in [1]. We use 2x bilinear upsampling after the last convolution layer in the proposal network. In our implementation, the front-end convolutions only proceed three pooling operations, i.e., 8x downsampling. Therefore, combined with the 2x deconvolution, the feature map fed to the proposal network is 4x downsampled with respect to the bird's eye view input.

We do 3D box regression by regressing to $\mathbf{t} = (\Delta x, \Delta y, \Delta z, \Delta l, \Delta w, \Delta h)$, similarly to RPN [18]. $(\Delta x, \Delta y, \Delta z)$ are the center offsets normalized by anchor sizes, and $(\Delta l, \Delta w, \Delta h)$ are computed as $\Delta s = log\frac{s_{\text{GT}}}{s_{\text{anchor}}}, s \in \{l, w, h\}$. we use a multi-task loss to simultaneously classify object/background and do 3D box regression. In particular, we use class-entropy for the "objectness" loss and Smooth ℓ_1 [9] for the 3D box regression loss. Background anchors are ignored when computing the box regression loss. During training, we compute the IoU overlap between anchors and ground truth bird's eye view boxes. An anchor is considered to be positive if its overlap is above 0.7, and negative if the overlap is below 0.5. Anchors with overlap in between are ignored.

Since LIDAR point cloud is sparse, which results in many empty anchors, we remove all the empty anchors during both training and testing to reduce computation. This can be achieved by computing an integral image over the point occupancy map.

For each non-empty anchor at each position of the last convolution feature map, the network generates a 3D box. To reduce redundancy, we apply Non-Maximum Suppression (NMS) on the bird's eye view boxes. Different from [22], we did not use 3D NMS because objects should occupy different space on the ground plane. We use IoU threshold of 0.7 for NMS. The top 2000 boxes are kept during training, while in testing, we only use 300 boxes.

3.3. Region-based Fusion Network

We design a region-based fusion network to effectively combine features from multiple views and jointly classify object proposals and do oriented 3D box regression.

Multi-View ROI Pooling. Since features from different views/modalities usually have different resolutions, we employ ROI pooling [9] for each view to obtain feature vectors of the same length. Given the generated 3D proposals, we can project them to any views in the 3D space. In our case, we project them to three views, i.e., bird's eye view (BV), front view (FV), and the image plane (RGB). Given a 3D proposal p_{3D}, we obtain ROIs on each view via:

$$\text{ROI}_v = \mathbf{T}_{\text{3D}\to v}(p_{\text{3D}}), v \in \{\text{BV}, \text{FV}, \text{RGB}\} \qquad (2)$$

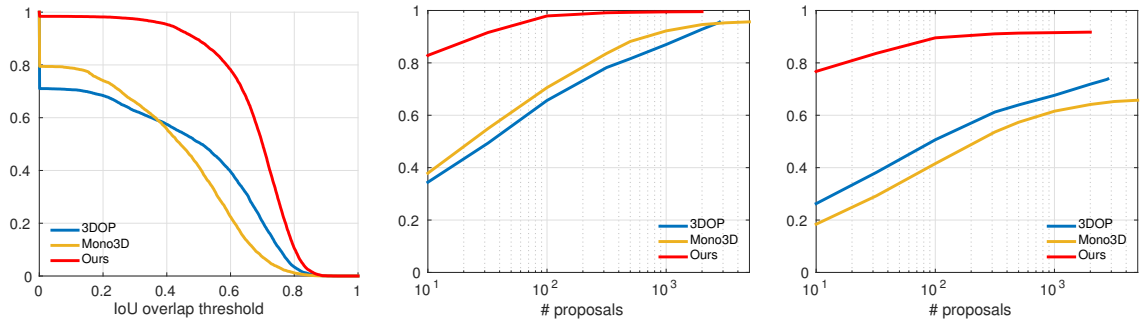


Figure 5: **3D bounding box Recall:** From left to right: Recall vs IoU using 300 proposals, Recall vs #Proposals at IoU threshold of 0.25 and 0.5 respectively. Recall are evaluated on moderate data of KITTI *validation* set.

where $\mathbf{T}_{3\mathrm{D}\to v}$ denotes the tranformation functions from the LIDAR coordinate system to the bird's eye view, front view, and the image plane, respectively. Given an input feature map x from the front-end network of each view, we obtain fixed-length features f_v via ROI pooling:

$$f_v = \mathbf{R}(x, \mathrm{ROI}_v), v \in \{\mathrm{BV}, \mathrm{FV}, \mathrm{RGB}\}. \tag{3}$$

Deep Fusion. To combine information from different features, prior work usually use *early fusion* [1] or *late fusion* [22, 12]. Inspired by [14, 26], we employ a *deep fusion* approach, which fuses multi-view features hierarchically. A comparison of the architectures of our deep fusion network and early/late fusion networks are shown in Fig. 3. For a network that has L layers, *early fusion* combines features $\{f_v\}$ from multiple views in the input stage:

$$f_L = \mathbf{H}_L(\mathbf{H}_{L-1}(\cdots \mathbf{H}_1(f_{BV} \oplus f_{FV} \oplus f_{RGB}))) \tag{4}$$

$\{\mathbf{H}_l, l = 1, \cdots, L\}$ are feature transformation functions and $\oplus$ is a join operation (e.g., concatenation, summation). In contrast, *late fusion* uses seperate subnetworks to learn feature transformation independently and combines their outputs in the prediction stage:

$$\begin{aligned} f_L = &(\mathbf{H}_L^{BV}(\cdots \mathbf{H}_1^{BV}(f_{BV})))\oplus \\ &(\mathbf{H}_L^{FV}(\cdots \mathbf{H}_1^{FV}(f_{FV})))\oplus \\ &(\mathbf{H}_L^{RGB}(\cdots \mathbf{H}_1^{RGB}(f_{RGB}))) \end{aligned} \tag{5}$$

To enable more interactions among features of the intermediate layers from different views, we design the following *deep fusion* process:

$$\begin{aligned} f_0 =& f_{BV} \oplus f_{FV} \oplus f_{RGB} \\ f_l =& \mathbf{H}_l^{BV}(f_{l-1}) \oplus \mathbf{H}_l^{FV}(f_{l-1}) \oplus \mathbf{H}_l^{RGB}(f_{l-1}), \\ & \forall l = 1, \cdots, L \end{aligned} \tag{6}$$

We use element-wise mean for the join operation for deep fusion since it is more flexible when combined with drop-path training [14].

Oriented 3D Box Regression Given the fusion features of the multi-view network, we regress to *oriented* 3D boxes from 3D proposals. In particular, the regression targets are the 8 corners of 3D boxes: $\mathbf{t} = (\Delta x_0, \cdots, \Delta x_7, \Delta y_0, \cdots, \Delta y_7, \Delta z_0, \cdots, \Delta z_7)$. They are encoded as the corner offsets normalized by the diagonal length of the proposal box. Despite such a 24-D vector representation is redundant in representing an oriented 3D box, we found that this encoding approach works better than the centers and sizes encoding approach. Note that our 3D box regression differs from [22] which regresses to axis-aligned 3D boxes. In our model, the object orientations can be computed from the predicted 3D box corners. We use a multi-task loss to jointly predict object categories and oriented 3D boxes. As in the proposal network, the category loss uses cross-entropy and the 3D box loss uses smooth ℓ_1. During training, the positive/negative ROIs are determined based on the IoU overlap of brid's eye view boxes. A 3D proposal is considered to be positive if the bird's eye view IoU overlap is above 0.5, and negative otherwise. During inference, we apply NMS on the 3D boxes after 3D bounding box regression. We project the 3D boxes to the bird's eye view to compute their IoU overlap. We use IoU threshold of 0.05 to remove redundant boxes, which ensures objects can not occupy the same space in bird's eye view.

Network Regularization We employ two approaches to regularize the region-based fusion network: *drop-path* training [14] and *auxiliary losses*. For each iteration, we randomly choose to do global drop-path or local drop-path with a probability of 50%. If global drop-path is chosen, we select a single view from the three views with equal probability. If local drop-path is chosen, paths input to each join node are randomly dropped with 50% probability. We ensure that for each join node at least one input path is kept. To further strengthen the representation capability of each view, we add auxiliary paths and losses to the network. As shown in Fig. 4, the auxiliary paths have the same number of layers with the main network. Each layer in the auxiliary paths shares weights with the corresponding layer in the

Method	Data	IoU=0.5			IoU=0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [3]	Mono	30.5	22.39	19.16	5.22	5.19	4.13
3DOP [4]	Stereo	55.04	41.25	34.55	12.63	9.49	7.59
VeloFCN [16]	LIDAR	79.68	63.82	62.80	40.14	32.08	30.47
Ours (BV+FV)	LIDAR	95.74	88.57	88.13	86.18	77.32	76.33
Ours (BV+FV+RGB)	LIDAR+Mono	**96.34**	**89.39**	**88.67**	**86.55**	**78.10**	**76.67**

Table 1: **3D localization performance:** Average Precision (AP_{loc}) (in %) of bird's eye view boxes on KITTI *validation* set.

Method	Data	IoU=0.25			IoU=0.5			IoU=0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [3]	Mono	62.94	48.2	42.68	25.19	18.2	15.52	2.53	2.31	2.31
3DOP [4]	Stereo	85.49	68.82	64.09	46.04	34.63	30.09	6.55	5.07	4.1
VeloFCN [16]	LIDAR	89.04	81.06	75.93	67.92	57.57	52.56	15.20	13.66	15.98
Ours (BV+FV)	LIDAR	96.03	88.85	88.39	95.19	87.65	80.11	71.19	56.60	55.30
Ours (BV+FV+RGB)	LIDAR+Mono	**96.52**	**89.56**	**88.94**	**96.02**	**89.05**	**88.38**	**71.29**	**62.68**	**56.56**

Table 2: **3D detection performance:** Average Precision (AP_{3D}) (in %) of 3D boxes on KITTI *validation* set.

main network. We use the same multi-task loss, i.e. classification loss plus 3D box regression loss, to back-propagate each auxiliary path. We weight all the losses including auxiliary losses equally. The auxiliary paths are removed during inference.

3.4. Implementation

Network Architecture. In our multi-view network, each view has the same architecture. The base network is built on the 16-layer VGG net [19] with the following modifications:

- Channels are reduced to half of the original network.
- To handle extra-small objects, we use feature approximation to obtain high-resolution feature map. In particular, we insert a 2x bilinear upsampling layer before feeding the last convolution feature map to the 3D Proposal Network. Similarly, we insert a 4x/4x/2x upsampling layer before the ROI pooling layer for the BV/FV/RGB branch.
- We remove the 4th pooling operation in the original VGG network, thus the convolution parts of our network proceed 8x downsampling.
- In the muti-view fusion network, we add an extra fully connected layer $fc8$ in addition to the original $fc6$ and $fc7$ layer.

We initialize the parameters by sampling weights from the VGG-16 network pretrained on ImageNet. Despite our network has three branches, the number of parameters is about 75% of the VGG-16 network. The inference time of the network for one image is around 0.36s on a Titan X GPU.

Input Representation. In the case of KITTI, which provides only annotations for objects in the front view (around 90° field of view), we use point cloud in the range of [0, 70.4] $\times$ [-40, 40] meters. We also remove points that are out of the image boundaries when projected to the image plane. For bird's eye view, the discretization resolution is set to 0.1m, therefore the bird's eye view input has size of 704×800. Since KITTI uses a 64-beam Velodyne laser scanner, we can obtain a 64×512 map for the front view points. The RGB image is up-scaled so that the shortest size is 500.

Training. The network is trained in an end-to-end fashion. For each mini-batch we use 1 image and sample 128 ROIs, roughly keeping 25% of the ROIs as positive. We train the network using SGD with a learning rate of 0.001 for 100K iterations. Then we reduce the learning rate to 0.0001 and train another 20K iterations.

4. Experiments

We evaluate our MV3D network on the challenging KITTI object detection benchmark [8]. The dataset provides 7,481 images for training and 7,518 images for testing. As the test server only evaluates 2D detection, we follow [4] to split the training data into *training* set and *validation* set, each containing roughly half of the whole training data. We conduct 3D box evaluation on the validation set. We focus our experiments on the car category as KITTI provides enough car instances for our deep network based approach. Following the KITTI setting, we do evaluation on three difficulty regimes: *easy*, *moderate* and *hard*.

Metrics. We evaluate 3D object proposals using *3D box recall* as the metric. Different from 2D box recall [13], we compute the IoU overlap of two *cuboids*. Note that the cuboids are not necessary to align with the axes, i.e., they could be oriented 3D boxes. In our evaluation, we set the 3D IoU threshold to 0.25 and 0.5, respectively. For the final 3D detection results, we use two metrics to measure the accuracy of 3D localization and 3D bounding box detection. For 3D localization, we project the 3D boxes to the ground plane (i.e., bird's eye view) to obtain oriented bird's eye

Data	AP_{3D} (IoU=0.5)			AP_{loc} (IoU=0.5)			AP_{2D} (IoU=0.7)		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Early Fusion	93.92	87.60	87.23	94.31	88.15	87.61	87.29	85.76	78.77
Late Fusion	93.53	87.70	86.88	93.84	88.12	87.20	87.47	85.36	78.66
Deep Fusion w/o aux. loss	94.21	88.29	87.21	94.57	88.75	88.02	88.64	85.74	79.06
Deep Fusion w/ aux. loss	**96.02**	**89.05**	**88.38**	**96.34**	**89.39**	**88.67**	**95.01**	**87.59**	**79.90**

Table 3: **Comparison of different fusion approaches:** Peformance are evaluated on KITTI *validation* set.

Data	AP_{3D} (IoU=0.5)			AP_{loc} (IoU=0.5)			AP_{2D} (IoU=0.7)		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
FV	67.6	56.30	49.98	74.02	62.18	57.61	75.61	61.60	54.29
RGB	73.68	68.86	61.94	77.30	71.68	64.58	83.80	76.45	73.42
BV	92.30	85.50	78.94	92.90	86.98	86.14	85.00	76.21	74.80
FV+RGB	77.41	71.63	64.30	82.57	75.19	66.96	86.34	77.47	74.59
FV+BV	95.19	87.65	80.11	95.74	88.57	88.13	88.41	78.97	78.16
BV+RGB	**96.09**	88.70	80.52	**96.45**	89.19	80.69	89.61	**87.76**	79.76
BV+FV+RGB	96.02	**89.05**	**88.38**	96.34	**89.39**	**88.67**	**95.01**	87.59	**79.90**

Table 4: **An ablation study of multi-view features**: Peformance are evaluated on KITTI *validation* set.

view boxes. We compute Average Precision (AP_{loc}) for the bird's eye view boxes. For 3D bounding box detection, we also use the Average Precision (AP_{3D}) metric to evaluate the full 3D bounding boxes. Note that both the bird's eye view boxes and the 3D boxes are oriented, thus object orientations are implicitly considered in these two metrics. We also evaluate the performance of 2D detection by projecting the 3D boxes to the image plane. Average Preicision (AP_{2D}) is also used as the metric. Following the KITTI convention, IoU threshold is set to 0.7 for 2D boxes.

Baslines. As this work aims at 3D object detection, we mainly compare our approach to LIDAR-based methods VeloFCN [16], Vote3Deep [6] and Vote3D [25], as well as image-based methods 3DOP [4] and Mono3D [3]. For fair comparison, we focus on two variants of our approach, i.e., the purely LIDAR-based variant which uses bird's eye view and front view as input (**BV+FV**), and the multimodal variant which combines LIDAR and RGB data (**BV+FV+RGB**). For 3D box evaluation, we compare with VeloFCN, 3DOP and Mono3D since they provide results on the validation set. For Vote3Deep and Vote3D, which have no results publicly available, we only do comparison on 2D detection on the test set.

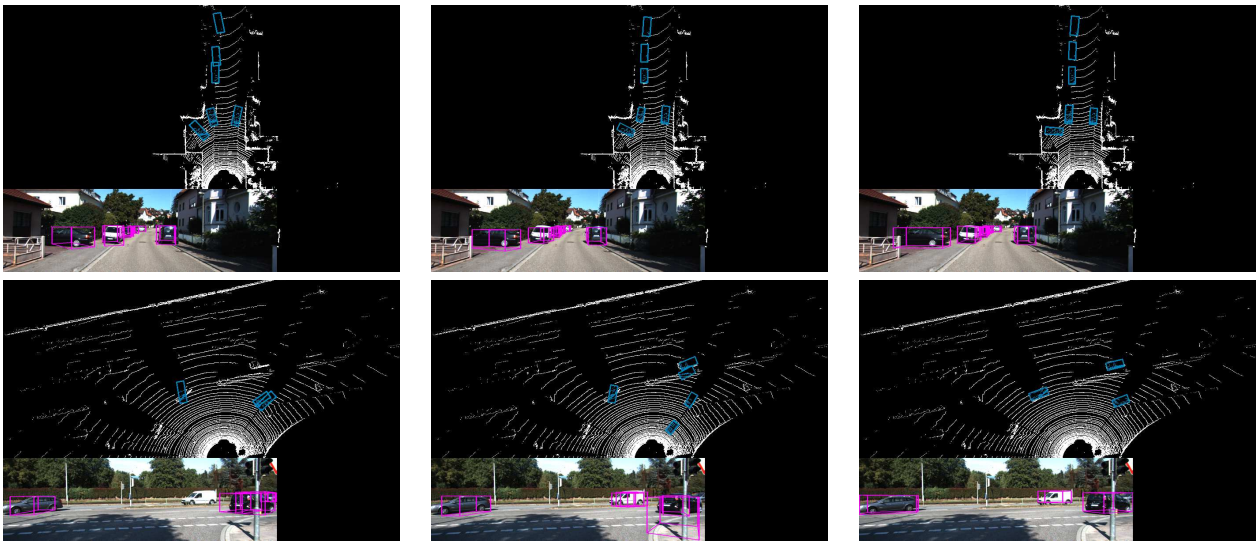
3D Proposal Recall. 3D box recall are shown in Fig. 5. We plot recall as a function of IoU threshold using 300 proposals. Our approach significantly outperforms 3DOP [4] and Mono3D [3] across all the IoU thresholds. Fig. 5 also shows 3D recall as a function of the proposal numbers under IoU threshold of 0.25 and 0.5, respectively. Using only 300 proposals, our approach obtains **99.1%** recall at IoU threshold of 0.25 and **91%** recall at IoU of 0.5. In contrast, when using IoU of 0.5, the maximum recall that 3DOP can achieve is only 73.9%. The large margin suggests the advantage of our LIDAR-based approach over image-based methods.

Method	Data	Easy	Mod.	Hard
Faster R-CNN [18]	Mono	86.71	81.84	71.12
3DOP [4]	Stereo	93.04	88.64	79.10
Mono3D [3]	Mono	92.33	88.66	78.96
SDP+RPN [29, 18]	Mono	90.14	88.85	78.38
MS-CNN [1]	Mono	90.03	89.02	76.11
SubCNN [28]	Mono	90.81	89.04	79.27
Vote3D [25]	LIDAR	56.80	47.99	42.57
VeloFCN [16]	LIDAR	71.06	53.59	46.92
Vote3Deep [6]	LIDAR	76.79	68.24	63.23
Ours (BV+FV)	LIDAR	87.00	79.24	78.16
Ours (BV+FV+RGB)	LIDAR+Mono	**89.11**	**87.67**	**79.54**

Table 5: **2D detection performance:** Average Precision (AP_{2D}) (in %) for car category on KITTI *test* set. Methods in the first group optimize 2D boxes directly while the second group optimize 3D boxes.

3D Localization. We use IoU threshold of 0.5 and 0.7 for 3D localization evaluation. Table 1 shows AP_{loc} on KITTI *validation* set. As expected, all LIDAR-based approaches performs better than stereo-based method 3DOP [4] and monocular method Mono3D [3]. Among LIDAR-based approaches, our method (BV+FV) outperforms VeloFCN [16] by $\sim$**25%** AP_{loc} under IoU threshold of 0.5. When using IoU=0.7 as the criteria, our improvement is even larger, achieving $\sim$**45%** higher AP_{loc} across easy, moderate and hard regimes. By combining with RGB images, our approach is further improved. We visualize the localization results of some examples in Fig. 6.

3D Object Detection. For the 3D overlap criteria, we focus on 3D IoU of 0.5 and 0.7 for LIDAR-based methods. As these IoU thresholds are rather strict for image-based methods, we also use IoU of 0.25 for evaluation. As shown in Table 2, our "BV+FV" method obtains $\sim$**30%** higher AP_{3D} over VeloFCN when using IoU of 0.5, achieving 87.65%



3DOP [4] VeloFCN [16] Ours

Figure 6: **Qualitative comparisons of 3D detection results:** 3D Boxes are projected to the bird's eye view and the images.

AP_{3D} in the *moderate* setting. With criteria of IoU=0.7, our multimodal approach still achieves 71.29% AP_{3D} on *easy* data. In the *moderate* setting, the best AP_{3D} that can be achieved by 3DOP using IoU=0.25 is 68.82%, while our approach achieves 89.05% AP_{3D} using IoU=0.5. Some 3D detectioin results are visualized in Fig. 6.

Ablation Studies. We first compare our deep fusion network with early/late fusion approaches. As commonly used in literature, the join operation is instantiated with concatenation in the early/late fusion schemes. As shown in Table 3, early and late fusion approaches have very similar performance. Without using auxiliary loss, the deep fusion method achieves ∼0.5% improvement over early and late fusion approaches. Adding auxiliary loss further improves deep fusion network by around 1%.

To study the contributions of the features from different views, we experiment with different combination of the bird's eye view (BV), the front view (FV), and the RGB image (RGB). The 3D proposal network is the same for all the variants. Detailed comparisons are shown in Table 4. If using only a single view as input, the bird's eye view feature performs the best while the front view feature the worst. Combining any of the two views can always improve over individual views. This justifies our assumption that features from different views are complementary. The best overal performance can be achieved when fusing features of all three views.

2D Object Detection. We finally evaluate 2D detection performance on KITTI *test* set. Results are shown in Table 5. Among the LIDAR-based methods, our "BV+FV" approach outperforms the recently proposed Vote3Deep [6] method by **14.93%** AP_{2D} in the *hard* setting. In overall, image-based methods usually perform better than LIDAR-based methods in terms of 2D detection. This is due to the fact that image-based methods directly optimize 2D boxes while LIDAR-based methods optimize 3D boxes. Note that despite our method optimizes 3D boxes, it also obtains competitive results compared with the state-of-the-art 2D detection methods.

Qualitative Results. As shown in Fig. 6, our approach obtains much more accurate 3D locations, sizes and orientation of objects compared with stereo-based method 3DOP [4] and LIDAR-based method VeloFCN [16]. We refer readers to the supplementary materials for many additional results.

5. Conclusion

We have proposed a multi-view sensory-fusion model for 3D object detection in the road scene. Our model takes advantage of both LIDAR point cloud and images. We align different modalities by generating 3D proposals and projecting them to multiple views for feature extraction. A region-based fusion network is presented to deeply fuse multi-view information and do oriented 3D box regression. Our approach significantly outperforms existing LIDAR-based and image-based methods on tasks of 3D localization and 3D detection on KITTI benchmark [8]. Our 2D box results obtained from 3D detections also show competitive performance compared with the state-of-the-art 2D detection methods.

Acknowledgements. The work was supported by National Key Basic Research Program of China (No. 2016YFB0100900) and NSFC 61171113.

References

[1] Z. Cai, Q. Fan, R. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, 2016. 4, 5, 7

[2] J. Carreira and C. Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *PAMI*, 34(7):1312–1328, 2012. 3

[3] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016. 1, 2, 3, 6, 7

[4] 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. 1, 2, 3, 6, 7, 8

[5] V. Dhiman, Q. H. Tran, J. J. Corso, and M. Chandraker. A continuous occlusion model for road scene understanding. In *CVPR*, pages 4331–4339, 2016. 2

[6] M. Engelcke, D. Rao, D. Zeng Wang, C. Hay Tong, and I. Posner. Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks. *arXiv:1609.06666*, 2016. 1, 2, 3, 7, 8

[7] M. Enzweiler and D. M. Gavrila. A multilevel mixture-of-experts framework for pedestrian classification. *IEEE Transactions on Image Processing*, 20(10):2967–2979, 2011. 1, 2

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 6, 8

[9] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2, 4

[10] A. Gonzalez, D. Vazquez, A. Lopez, and J. Amores. On-board object detection: Multicue, multimodal, and multiview random forest of local experts. In *IEEE Transactions on Cybernetics*, 2016. 1, 2

[11] V. Hegde and R. Zadeh. Fusionnet: 3d object classification using multiple data representations. *CoRR*, abs/1607.05695, 2016. 2

[12] J. Hoffman, S. Gupta, and T. Darrell. Learning with side information through modality hallucination. In *CVPR*, 2016. 5

[13] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *PAMI*, 2015. 6

[14] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv:1605.07648*, 2016. 1, 2, 5

[15] B. Li. 3d fully convolutional network for vehicle detection in point cloud. *arXiv:1611.08069*, 2016. 2

[16] B. Li, T. Zhang, and T. Xia. Vehicle detection from 3d lidar using fully convolutional network. In *Robotics: Science and Systems*, 2016. 1, 2, 3, 6, 7, 8

[17] C. R. Qi, M. N. H. Su, A. Dai, M. Yan, and L.Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 2016. 2

[18] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3, 4, 7

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv:1409.1556*, 2014. 6

[20] S. Song and M. Chandraker. Joint sfm and detection cues for monocular 3d localization in road scenes. In *Computer Vision and Pattern Recognition*, pages 3734–3742, 2015. 2

[21] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*. 2014. 2

[22] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *CVPR*, 2016. 2, 3, 4, 5

[23] H. Su, S.Maji, E.Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015. 2

[24] K. Van de Sande, J. Uijlings, T. Gevers, and A. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011. 3

[25] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Proceedings of Robotics: Science and Systems*, 2015. 1, 2, 3, 7

[26] J. Wang, Z. Wei, T. Zhang, and W. Zeng. Deeply-fused nets. *arXiv:1605.07716*, 2016. 2, 5

[27] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *CVPR*, 2015. 2

[28] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *arXiv:1604.04693*. 2016. 7

[29] 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. 7

[30] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3d representations for object recognition and modeling. *PAMI*, 2013. 2

[31] M. Z. Zia, M. Stark, and K. Schindler. Are cars just 3d boxes? jointly estimating the 3d shape of multiple objects. In *CVPR*, pages 3678–3685, 2014. 2

[32] L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*. 2014. 3