2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

A General Pipeline for 3D Detection of Vehicles

Xinxin Du[1], Marcelo H. Ang Jr.[2], Sertac Karaman[3] and Daniela Rus[3]

Abstract— **Autonomous driving requires 3D perception of vehicles and other objects in the in environment. Much of the current methods support 2D vehicle detection. This paper proposes a flexible pipeline to adopt any 2D detection network and fuse it with a 3D point cloud to generate 3D information with minimum changes of the 2D detection networks. To identify the 3D box, an effective model fitting algorithm is developed based on generalised car models and score maps. A two-stage convolutional neural network (CNN) is proposed to refine the detected 3D box. This pipeline is tested on the KITTI dataset using two different 2D detection networks. The 3D detection results based on these two networks are similar, demonstrating the flexibility of the proposed pipeline. The results rank second among the 3D detection algorithms, indicating its competencies in 3D detection.**

I. INTRODUCTION

Vision-based car detection has been well developed and widely implemented using deep learning technologies. The KITTI [1] benchmark site reports that the state of the art algorithms are able to achieve $\sim 90\%$ average precision (AP).

However, for autonomous vehicles, car detection in 2D images is not sufficient to provide enough information for the vehicle to perform planning and decision making due to the lack of depth data. For a robust and comprehensive perception system in autonomous vehicle, 3D car detection, including car dimensions, locations and orientations in the 3D world, is essential. However the state of the art for 3D car detection algorithms only achieve 62% AP. Gaps still exist as compared to the 2D detection performance and the problem remains as challenging.

According to the types of input sources, the current algorithms for 3D vehicle detection can be categorised into four major groups, including (1) mono image based, (2) stereo image, (3) LiDAR (Light Detection and Ranging), and (4) fusion between mono image and Lidar.

Mono images lack the depth information to recover the 3D location of detected obstacles, therefore assumptions and approximations have to be made. Stereo image based approaches normally involve the construction of depth maps from stereo correspondence matching. The performance of this type of approach depends heavily on the depth map reconstruction and the accuracy drops as distance from the vehicle increases.

LiDAR, despite its high cost, is able to provide the most direct measurement of object location. But it lacks color information and it is always sparse which poses difficulties in classification. In order to make use of the full capabilities of LiDAR and camera, fusion approaches have been proposed in the literature. To make use of the deep CNN architecture, the point cloud needs to be transformed into other formats. In the process of transformation, information is lost.

The prior approaches for 3D vehicle detection are not as effective as those for 2D detection. Little attention has been paid to how to transfer the advantages and lessons learnt from 2D detection approaches to 3D detection approaches. Moreover, the field is lacking effective 3D detection approaches that enable the existing 2D approaches to provide 3D information. The state of the art 2D approaches can not be applied to autonomous vehicles which require 3D information.

In this paper, we propose a flexible 3D vehicle detection pipeline which can make use of any 2D detection network and provide accurate 3D detection results by fusing the 2D network with a 3D point cloud. The general framework structure is illustrated in Fig. 1. The raw image is passed to a 2D detection network which provides 2D boxes around the vehicles in the image plane. Subsequently, a set of 3D points which fall into the 2D bounding box after projection is selected. With this set, a model fitting algorithm detects the 3D location and 3D bounding box of the vehicle. In the next step, a new CNN network, which takes the points that fit into the 3D bounding box as input, carries out the final 3D box regression and classification. It requires minimum effort to modify the existing 2D networks to fit into the pipeline, adding just one additional regression term at the output layer to estimate the vehicle dimensions. The main contributions of the paper are:

1) A general pipeline that enables any 2D detection network to provide accurate 3D detection information.
2) Three generalised car models with score maps, which achieve a more efficient model fitting process.
3) A two-stage CNN that can further improve the detection accuracy.

This pipeline has been tested using two outstanding 2D networks, PC-CNN [2] and MS-CNN [3]. The 3D detection performances based on both networks were evaluated using the KITTI dataset [1]. We significantly lead the majority of the algorithms in both bird eye detection and 3D detection tasks. We also achieved comparable results to the current state of the art algorithm MV3D [4] in both tasks.

[1]Xinxin Du is with the Singapore-MIT Alliance for Research and Technology (SMART) Centre, Singapore xinxin@smart.mit.edu

[2]Marcelo H. Ang Jr. is with the National University of Singapore, Singapore mpeangh@nus.edu.sg

[3]Sertac Karaman and Daniela Rus are with the Massachusetts Institute of Technology, Cambridge, MA, USA sertac@mit.edu rus@csail.mit.edu

978-1-5386-3081-5/18/$31.00 ©2018 IEEE

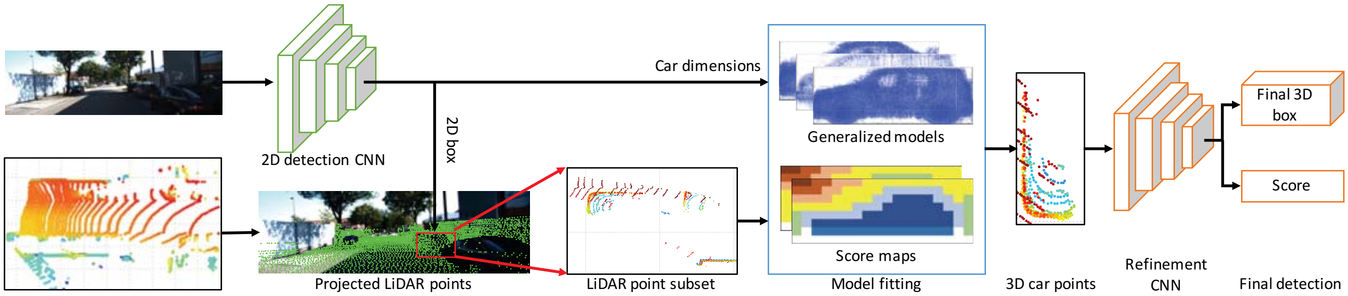

Fig. 1: General fusion pipeline. All of the point clouds shown are in 3D, but viewed from the top (bird's eye view). The height is encoded by color, with red being the ground. A subset of points is selected based on the 2D detection. Then, a model fitting algorithm based on the generalised car models and score maps is applied to find the car points in the subset and a two-stage refinement CNN is designed to fine tune the detected 3D box and re-assign an objectiveness score to it.

II. Related work

This section reviews the works that are related to the proposed pipeline in details. It also highlights the differences between our proposal and the prior works.

Mono Image Approaches: In [5], a new network was designed to estimate the car dimensions, orientations and probabilities given a detected 2D box from an existing network. Using the criterion that the perspective projection of a 3D box should fit tightly with the 2D box in the image, a 3D box was recovered by using the estimated information. Similarly, in Mono3D [6], 3D box proposals were generated based on flat ground assumption. In DeepMANTA [7], the vehicle orientation and size were estimated from a deep CNN. Additionally, the network also estimated 36 locations of key points on the car in the image coordinates. A 2D/3D shape matching algorithm [8] was applied to estimate vehicle 3D poses based on these 36 2D part locations.

Another set of algorithms, e.g. [9], [10], [11] and [12], defined 3D car models with occlusion patterns, carried out detection of the patterns in the 2D image and recovered the 3D occluded structure by reasoning through a MAP (maximum a posteriori) framework.

These approaches are sensitive to the made assumptions and the parameter estimation accuracy. As shown in the result section, our method outperforms them significantly.

Stereo Image Approaches: The depth map from stereo correspondence is normally appended to the RGB image as the fourth channel. The RGB-D image is passed to one or more CNNs in order to carry out the detection. In [13], Pham *et al*. proposed a two-stream CNN where the depth channel and RGB channel went through two separate CNN branches and were fused before the fully connected layers. These approaches do not require additional depth sensors and can generate dense depth channel. But the accuracy of the estimated depth degrades with the real distance.

Lidar Approaches: The common framework involves three steps: pre-processing (e.g. voxelization), segmentation and classification. A detailed review of LiDAR approaches can be found in [14]. Wang *et al*. [15] proposed a different approach where the point cloud was converted into 3D feature grids and a 3D detection window was slid through the feature grids to identify vehicles. In [16], the point cloud was converted into a 2D point map and a CNN was designed to identify the vehicle bounding boxes in the 2D point map. In [17], the authors extended the approach of [16] and applied 3D deep CNN directly on the point cloud. However, this approach is very time consuming and memory intensive due to the 3D convolutions involved. To improve, [18] proposed a voting mechanism able to perform sparse 3D convolution.

Fusion Approaches: In [19], the sparse point cloud is converted to a dense depth image, which is similar to a stereo one. The RGB-D image was passed through a CNN for detection. In [20], the point cloud was converted into a three-channel map HHA which contains horizontal disparity, height above ground and angle in each channel. The resulting six-channel image RGB-HHA was processed by a CNN for detection of vehicles. However these two methods will not be able to output the 3D information directly from the network.

In order to address this, MV3D (multi-view 3D) detection network proposed by Chen *et al*. [4] included one more type of input generated from the point cloud, the bird's eye view feature input. This input has no projective loss as compared to the depth map, thus 3D proposal boxes can be generated directly. This approach has achieved the current state of the art in 3D vehicle detection. It generates 2D boxes from 3D boxes while ours generate 3D boxes from 2D boxes. And MV3D explores the entire point cloud while ours only focus on a few subsets of the point cloud, which is more efficient and saves computation power.

2D Detection: The proposed pipeline is flexible in regards to the choice of 2D detection networks. To enable the 2D network to estimate the dimensions of the cars, only a slight change is required on the last fully connected layer. Both [5] and [7] proposed ways to encode the car dimensions to the network. For better accuracy, the 2D detection networks proposed in [21], [7] and [22] can be incorporated since they are the leading networks for 2D detection. For faster computation, the approaches presented in [23] and [24] can be implemented. In this paper, we implement PC-CNN [2] and MS-CNN [3] to demonstrate the flexibility of the pipeline.

Model Fitting: In [9], Xiang *et al*. proposed 3D voxel patterns (3DVPs) as the 3D car model. 3DVPs encode the occlusion, self-occlusion and truncation information. A boosting detector was designed to identify the 3DVPs in the image, while [10] implemented a sub-category awareness CNN for 3DVP detection.

Deformable part-based models (DPM) can be found in [25], [26], [27] and [28]. Different classifiers were trained to detect DPM. Fidler *et al*. extended the DPM to a 3D cuboid model in [29] in order to allow reasoning in 3D. In [12], [30], [11] and [7], 3D wireframe models were used. Similarly, each wire vertex is encoded with its visibility.

Due to the various vehicle types, sizes, and occlusion patterns, these prior approaches require a substantial number of models in order to cover all possible cases. In our approach, only three models are used and the occlusion pattern is assigned online when doing model fitting.

III. Technical Approach

The proposed pipeline takes a color image as one of its inputs. The first step is to generate 2D bounding boxes for the candidate vehicles. Secondly, these bounding boxes are used to select subsets of the point clouds, using the transformation between the camera and LiDAR. Due to the perspective nature of the camera, the 3D point subset may spread across a much larger area than the vehicle itself as shown in Fig.1. This subset also contains a substantial number of non-vehicle points and points on neighbouring vehicles. All these artefacts add challenges to the 3D box detection.

A. Car dimension estimation

One additional regression layer is needed at the end of the given 2D detection network. This regression method was inspired by [7] and [5]. First the average dimensions for all the cars and vans in KITTI dataset is obtained. Let $[\bar{h},\ \bar{l},\ \bar{w}]$ denote height, length and width of the vehicle. The ground truth regression vector $\Delta_i^* = (\delta_h,\ \delta_l,\ \delta_w)$ is defined as:

$$\delta_h = log(h^*/\bar{h}) \quad \delta_l = log(l^*/\bar{l}) \quad \delta_w = log(w^*/\bar{w}) \tag{1}$$

The dimension regression loss is shown as:

$$L_d(i) = \lambda_d C_i R(\Delta_i - \Delta_i^*) \tag{2}$$

where λ_d is the weighting factor to balance the losses defined in the original network, e.g. classification loss, 2D box regression loss; C_i is 1 if the 2D box is a car and 0 otherwise; R is the smooth L_1 loss function defined in [31] and Δ_i is the regression vector from the network.

To train the modified network, we can reuse the pre-trained weights from the original network for initialisation. Only a small part of the network needs to be re-trained while the rest can be kept as fixed during training. For example, in MS-CNN, we only re-trained the convolution layer and the fully connected layers after ROI pooling in the detection sub-network and in PC-CNN, we re-trained the GoogleNet layer, convolution layer and the fully connected layers after the Deconvolution layer in the detection sub-network.

B. Vehicle model fitting

We first generate a set of 3D box proposals. For each proposal, the points within the 3D box are compared to the three generalised car models. The proposal with the highest score is selected for the two-stage CNN refinement.

The 3D box proposals are generated following the principle of RANSAC algorithm (random sample consensus). In each iteration, one point is selected randomly. A second point is randomly selected from points within the cube centred at the first point and with the side length of $1.5l$, where l is the car length from the 2D CNN dimension estimation and 1.5 (empirically set) compensates the estimation error. A vertical plane is derived from these two points. Any points with a distance to the plane less than a threshold are considered as inliers to the plane. A maximum 20 points are then randomly selected from the inliers. At each point, a second vertical plane, passing through that point and perpendicular to the first vertical plane, is derived.

Along the intersection line between these two vertical planes, eight 3D boxes can be generated based on the estimated car width and length. Since the first vertical plane is visible, based on the view direction, four boxes are eliminated. At each of the remaining box locations, a new range is defined by expanding the box by 1.5 times along both w and l directions. The lowest point within the new range can be found and it determines the ground of the 3D box while the roof of the 3D box is set based on the height estimation. In summary, at each iteration, maximum 80 3D box proposals can be generated.

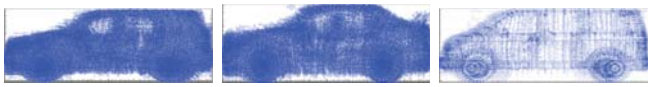
(a) SUV (b) Sedan (c) Van

Fig. 2: Generalised car models

Only three generalised car models are used for model fitting. They represent three categories of cars, SUVs, Sedans and Vans. Hatchback cars are considered to be SUVs. We observe that the relative distances/positions between different parts of a car do not vary significantly for different cars from the same category even with different sizes. This invariance indicates that if the cars under the same category are normalised to the same dimension $[h,\ l,\ w]$, their shapes and contours will be similar. We verified this and generalised the car models by normalising the cars in the 3D CAD dataset used in [7], [29] and [32]. Figure 2 illustrates the side view of the point cloud plots for the three categories. Each plot is an aggregation of the points that are generated from the 3D CAD models, aligned to the same direction and normalised to the same dimension. The SUV/hatchback plot consists of points from 58 CAD models, the sedan plot consists of 65 point sets, and the van plot consists of points from 10 models.

Each aggregation is then voxelized to a $8 \times 18 \times 10$ matrix along the $[h\ l\ w]$ direction. Each element in the matrix is assigned different scores based on its position. The elements representing the car shell/surface are assigned a score of 1, indicating that 3D points in the model fitting process fall on the car surface will be counted towards the overall score. The elements inside or outside the car shell are assigned negative scores, and the further away they get from the car shell (either inwards or outwards), the smaller the assigned values. This indicates that no points should be detected from outside or inside the car by LiDAR and the overall score will be penalised for such detections. The elements at the bottom layer of the matrix are assigned a score of 0. Points detected at the bottom layer could be the ground or car's tires, which are difficult to distinguish from each other. They will not be penalised or counted.

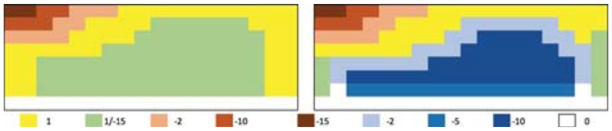

Fig. 3: Score maps for car surface (left) and center slice (right). Scores are indicated at bottom.

Self-occlusion can be easily determined from the view direction. This is encoded online when doing the model fitting since view direction changes for different 3D box proposals. Negative scores are assigned to the car surface elements if they are self-occluded. Furthermore, for simplicity, only the four vertical facets are considered for self-occlusion analysis while car roof and bottom are not considered.

Two slices of the score assignment from the SUV category are shown in Fig. 3, with the left image depicting the side facet and the right image illustrating the center slice. The car exterior and interior are indicated by orange and blue while the bottom is indicated white. Yellow and green refer to the shell/surface of the car, while green further indicates that those areas might be self-occluded.

Points within the 3D box proposals will be voxelised into $8 \times 18 \times 10$ grids and compared to the three potential vehicle models. Due to the orientation ambiguity, the grids are rotated around their vertical center axis by 180 degree and are then compared to the three models. Out of all the bounding box proposals, the one with the highest score is selected for the next step.

C. Two-stage refinement CNN

To further align the detected 3D box to the point cloud, we designed a two-stage CNN. In the literature, 3D CNNs are commonly used to process 3D point clouds, e.g. [16]. However, these CNNs are extremely slow and memory intensive. In this paper, we found that 2D CNNs are sufficient.

With the points in a given 3D box, the first CNN outputs a new 3D box. A new set of points can be found within the new 3D box. The second CNN outputs a probability score based on the new set of points to indicate how likely these points represent an actual car.

However, point sets cannot be input to the CNN directly. We apply normalization and voxelization strategies in order to formalize the points in matrix form in order to fit to the CNN. Furthermore, consistent with 2D image detection cases [33], [3], a bounding box context is able to provide additional information to improve the detection accuracy. We also include the context of the 3D bounding box as input to the CNN.

Given a 3D box from the model fitting process, our pipeline expands it along its h, l, w directions by ~ 1.5 times respectively to include its context. The points inside this expanded box are normalised and voxelised into a $24 \times 54 \times 32$ matrix. The matrix is sparse with $\sim 0.6\%$ occupied elements on average. As compared to the generalised model, we doubled the resolution of the voxelisation in order to reserve more spatial details and patterns of the distribution of the points. Note that the normalisation is anisometric, it has different scale ratios along different directions.

The backbones of the CNNs in both stages are based on the VGG-net with configuration D as described in [34]. After each convolution, an ELU (exponential linear units) layer [35], instead of Re(ctified)LU layer, is adopted for a more stable training process. The first stage CNN has two parallel outputs, one for 3D box regression and the other for classification, while the second stage CNN only has one output, classification.

$$\begin{aligned}
&\delta^*_{x_c} = (X^*_c - X_c)/L \quad \delta^*_{y_c} = (Y^*_c - Y_c)/H \quad \delta^*_{z_c} = (Z^*_c - Z_c)/W \\
&\delta^*_{x_l} = (X^*_l - X_l)/L \quad \delta^*_{y_l} = (Y^*_l - Y_l)/H \quad \delta^*_{z_l} = (Z^*_l - Z_l)/W \\
&\delta^*_{w} = log(W^*/W) \qquad (3)
\end{aligned}$$

The classification loss for both CNNs is *SoftMax* loss and the 3D box regression loss is *SmoothL*1 loss. The ground truth regression vector Δ^*_{3d} defined in (3) has seven elements, three for the center of the box, three for the left bottom corner and one for the width. It is just sufficient to recover the 3D bounding box from these seven elements. Due to the anisometric normalisation, a quartic polynomial needs to be solved. Note that across all the inputs, $X_{c/l}$, $Y_{c/l}$, $Z_{c/l}$, L, H, W are all constant as all the 3D boxes are aligned and normalised to the same size.

Classification has two classes, car and background. A 3D box is classified as positive when the IOU (intersection over union) between its bird's eye view box and the ground truth bird's eye view box is greater than a specific threshold. This threshold is 0.5 for the first stage CNN and 0.7 for the second. 0.7 is consistent with the criteria set by KITTI benchmark. The reason to set a lower threshold for the first stage is to train the network so that it is able to refine the boxes with IoU between 0.5 to 0.7 to a better position where the IoU may be greater than 0.7; otherwise the network will take those boxes as negative and will not be trained to refine them.

The training of the two networks is carried out independently as they do not share layers. The training batch size is 128, with 50% being positive. Both CNNs are trained for 10K iterations with a constant learning rate of 0.0005.

TABLE I: Average Precision benchmark for bird's eye view and 3D box based on KITTI validation set.

Algorithm	Bird's Eye View						3D Box					
	IoU = 0.5			IoU = 0.7			IoU = 0.5			IoU = 0.7		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
*Mono3D [6]	30.50	22.39	19.16	5.22	5.19	4.13	25.19	18.20	15.52	2.53	2.31	2.31
*3DOP [9]	55.04	41.25	34.55	12.63	9.49	7.59	46.04	34.63	30.09	6.55	5.07	4.10
**Deep3DBox [5]	29.96	24.91	19.46	9.01	7.94	6.57	24.76	21.95	16.87	5.40	5.66	3.97
*VeloFCN [16]	79.68	63.82	62.80	40.14	32.08	30.47	67.92	57.57	52.56	15.20	13.66	15.98
*MV3D [4]	96.34	89.39	88.67	86.55	78.10	76.67	96.02	89.05	88.38	71.29	62.68	56.56
Ours (MS-CNN [3])	90.36	88.46	84.75	82.17	77.15	74.42	87.16	87.38	79.40	55.82	55.26	51.89
Ours (PC-CNN [2])	88.31	83.74	79.62	83.61	77.36	69.61	87.69	79.92	78.65	57.63	51.74	51.39

* sources from [4]

** sources from [5], which uses different validation set, so its APs are calculated from the 1848 common images with our validation set.

IV. EXPERIMENT RESULTS AND DISCUSSION

To verify the flexibility of our approach, the pipeline is tested using PC-CNN [2] and MS-CNN [3]. The performance based on both networks is evaluated using the challenging KITTI dataset [1], which contains 7481 images for training/validation and 7518 images for testing. The training/validation set has annotated ground truth for 2D bounding box in the image plane and 3D bounding box in real world. Following [36], we split the training/validation set into training and validation sub-sets. The training sub-set is purely used to train the car dimension regression and two-stage CNN while the validation sub-set is for evaluation only. KITTI divides the cars into easy, moderate and hard groups based on their visibilities. We follow this same convention for our evaluation. To further verify the performance of the proposed pipeline, we also tested it using our own autonomous vehicles.

Metrics: The primary focus of this paper is on 3D detection, we do not evaluate the performance of the pipeline for 2D detection tasks. Following the evaluation metrics proposed in [4], we evaluate our proposal based on the Average Precession (AP) for bird's eye view boxes and for 3D boxes. The bird's eye view boxes are generated by projecting the 3D boxes on the same ground plane. The AP is calculated based on the IoU between the output boxes and the ground truth boxes, while in [9] and [7], the distance between two boxes are used. We feel that IoU is a more comprehensive index than distance, as it implicitly accounts for not only distance but also alignment and size.

Bird's Eye View & 3D Box AP: We compare the outputs from our pipeline with other algorithms which output 3D box information, including Mono3D [6], 3DOP [9] and Deep3DBox [5] using image data only, VeloFCN [16] using LiDAR data only and MV3D [4] using fusion.

The IoU threshold for true positive detection is set at 0.5 and 0.7. The left part of TABLE I shows the results from bird's eye view. In general, the point cloud based approaches all significantly lead the image-based approaches for both IoUs. Within the point cloud based approaches, our pipeline outperforms VeloFCN significantly but underperforms MV3D marginally. When $IoU = 0.5$, our performance with PC-CNN is about 7% worse on average than MV3D and 5% worse for MS-CNN. When $IoU = 0.7$, the performances with PC-CNN and MS-CNN are both very close to MV3D except for the performance with PC-CNN for the hard group (7% worse than MV3D).

The 3D box detection comparisons are listed in the right part of TABLE I. Similarly for both IoU thresholds, our method significantly outperforms all the approaches with the single exception of MV3D. On average, the overall performance is about 10% worse than MV3D for both $IoU =$ 0.5 and 0.7 except that the performance with MS-CNN for moderate group at $IoU = 0.5$ is only 1.6% less than MV3D.

We only use point clouds to generate the 3D box and do not take any color information from the image into account. Comparing it to VeloFCN, which also only takes point clouds as inputs, shows the effectiveness of our approach, processing the point cloud as subsets instead of as a whole. Comparing to MV3D, image color information is necessary to further boost the performance of our pipeline. One possible solution is to extract the feature layer which is right before the *ROI pooling* layer in the 2D detection CNN. Based on the 3D box from the model fitting process, we could find its corresponding 2D box in the image plane and carry out *ROI pooling* on the extracted feature layer in order to extract the feature vector. Then fuse the feature vector with the one from the refinement CNN to output the final 3D box and its probability.

Flexibility Analysis: The comparison between the two approaches using the proposed pipeline verifies the pipeline's flexibility. Although both PC-CNN and MS-CNN have different network structures, both achieve comparable AP for the two tasks and IoU thresholds. Furthermore, the two-stage refinement CNN was trained based on the pipeline with PC-CNN and re-used in the pipeline with MS-CNN without any further tuning on the network. This further confirms the flexibility and adaptability of our proposed pipeline.

Car Dimension Regression Impact: We show the impact from the car dimension regression on the original 2D detection CNN in TABLE II. Similarly, AP is populated for the 2D detection task in image plane. Following KITTI, the IoU threshold is set at 0.7. The left part shows the performance of the original 2D detection CNN while the right part indicates the results after appending the car dimension regression term. The impact is not very significant for both networks, and it even improves the performance marginally for some groups.

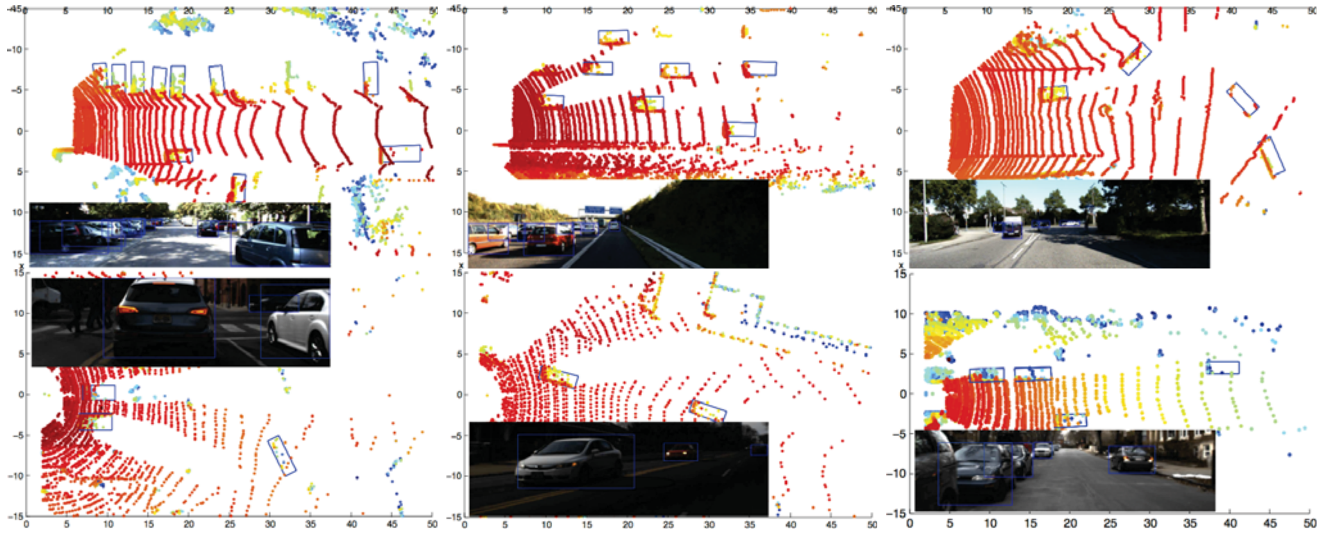

Fig. 4: Qualitative result illustration on KITTI data (top row) and Boston data (bottom row). Blue boxes are the 3D detection results.

TABLE II: Impact on the original 2D detection CNN from appending the car dimension regression term.

2D Detection	Original			With Dimension Regression		
	Easy	Moderate	Hard	Easy	Moderate	Hard
MS-CNN	91.64	89.95	79.55	93.98	89.92	79.69
PC-CNN	94.62	89.60	79.97	90.22	89.03	81.64

Ablation Study: To analyse the effectiveness of the steps involved in the 3D box generation, the AP is calculated after each step (model fitting, first stage CNN and second stage CNN) for both bird's eye view and 3D box tasks. For this study, the IoU threshold is set to 0.5. Since the results based on MS-CNN and PC-CNN are quite comparable, only PC-CNN results are presented in TABLE III.

The results from the model fitting are not as good as the final ones, but they are better than all the image based algorithms in TABLE I and comparable to VeloFCN. This indicates that the model fitting algorithm can work properly.

TABLE III: Ablation study based on KITTI validation set. Numbers indicate AP with IoU threshold at 0.5.

Step	Bird's Eye View			3D Box		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Model fitting	77.71	73.27	70.06	56.32	51.33	47.40
First CNN	88.16	83.60	79.65	87.51	79.76	78.81
Second CNN	88.31	83.74	79.62	87.69	79.92	78.65

With the first CNN, the detection performance is improved significantly in both bird's eye view and 3D box tasks. The improvement is $\sim 10\%$ and $\sim 30\%$ respectively. This shows that although only 2D convolution is used and the input 3D matrix is very sparse, the network is still very powerful and effective to locate the 3D box. The improvement from the second CNN is insignificant since it is not designed to regress the 3D box. It is designed to reshuffle the probability of the 3D box from the first CNN.

Qualitative Results: The first row in Fig. 4 shows some of the 3D detection results by applying our pipeline with PC-CNN on KITTI validation dataset. We also tested it using our own dataset collected at Boston USA. The setup of the data collection vehicle is similar to KITTI, with differences in the relative positions between the LiDAR, camera and the car. We applied the pipeline, which is trained based on KITTI training dataset, directly on the Boston data without any fine-tuning of the network weights. The system still works as shown in the second row of Fig. 4. It shows the generalisation capability of the proposed pipeline and indicates its potentials in executing 3D vehicle detection in real situations beyond a pre-designed dataset. Interested readers may refer to the link for video illustrations (`https://www.dropbox.com/s/5hzjvw911xa5mye/kitti_3d.avi?dl=0`).

V. CONCLUSIONS

In this paper we propose a flexible 3D vehicle detection pipeline which is able to adopt the advantages of any 2D detection networks in order to provide 3D information. The effort to adapt the 2D networks to the pipeline is minimal. One additional regression term is needed at the network output to estimate vehicle dimensions. The pipeline also takes advantage of point clouds in 3D measurements. An effective model fitting algorithm based on generalised car models and score maps is proposed to fit the 3D bounding boxes from the point cloud. Finally a two-stage CNN is developed to fine tune the 3D box. The outstanding results based on two different 2D networks indicate the flexibility of the pipeline and its capability in 3D vehicle detection.

ACKNOWLEDGMENT

This research was supported by the National Research Foundation, Prime Minister's Office, Singapore, under its CREATE programme, Singapore-MIT Alliance for Research and Technology (SMART) Future Urban Mobility (FM) IRG. We also gratefully acknowledge the support from NVIDIA

Corporation. Toyota Research Institute ("TRI") also provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We are grateful for their support.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] X. Du, M. H. A. Jr., and D. Rus, "Car detection for autonomous vehicle: Lidar and vision fusion approach through deep learning framework," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017.

[3] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision*, pp. 354–370, Springer, 2016.

[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] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," *arXiv preprint arXiv:1612.00496*, 2016.

[6] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2156, 2016.

[7] 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.

[8] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[9] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1903–1911, 2015.

[10] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pp. 924–933, IEEE, 2017.

[11] M. Zeeshan Zia, M. Stark, and K. Schindler, "Are cars just 3d boxes?-jointly estimating the 3d shape of multiple objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3678–3685, 2014.

[12] M. Zeeshan Zia, M. Stark, and K. Schindler, "Explicit occlusion modeling for 3d object class representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3326–3333, 2013.

[13] C. C. Pham and J. W. Jeon, "Robust object proposals re-ranking for object detection in autonomous driving using convolutional neural networks," *Signal Processing: Image Communication*, pp. –, 2017.

[14] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[15] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection.," in *Robotics: Science and Systems*, 2015.

[16] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

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

[18] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 1355–1361, IEEE, 2017.

[19] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 681–687, IEEE, 2015.

[20] J. Schlosser, C. K. Chow, and Z. Kira, "Fusing lidar and images for pedestrian detection using convolutional neural networks," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 2198–2205, IEEE, 2016.

[21] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," in *CVPR*, 2017.

[22] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2129–2137, 2016.

[23] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 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 *European Conference on Computer Vision*, pp. 21–37, Springer, 2016.

[25] B. Pepik, M. Stark, P. Gehler, and B. Schiele, "Teaching 3d geometry to deformable part models," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3362–3369, IEEE, 2012.

[26] D. Forsyth, "Object detection with discriminatively trained part-based models," *Computer*, vol. 47, no. 2, pp. 6–7, 2014.

[27] J. J. Yebes, L. M. Bergasa, R. Arroyo, and A. Lázaro, "Supervised learning and evaluation of kitti's cars detector with dpm," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 768–773, IEEE, 2014.

[28] B. Pepik, M. Stark, P. Gehler, and B. Schiele, "Multi-view and 3d deformable part models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 11, pp. 2232–2245, 2015.

[29] S. Fidler, S. Dickinson, and R. Urtasun, "3d object detection and viewpoint estimation with a deformable 3d cuboid model," in *Advances in neural information processing systems*, pp. 611–619, 2012.

[30] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler, "Detailed 3d representations for object recognition and modeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2608–2623, 2013.

[31] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[32] L.-C. Chen, S. Fidler, A. L. Yuille, and R. Urtasun, "Beat the mturkers: Automatic image labeling from weak 3d supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3198–3205, 2014.

[33] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2874–2883, 2016.

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

[35] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[36] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*, pp. 424–432, 2015.