Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud

Xinshuo Weng
Carnegie Mellon University
xinshuow@cs.cmu.edu

Kris Kitani
Carnegie Mellon University
kkitani@cs.cmu.edu

Abstract

Monocular 3D scene understanding tasks, such as object size estimation, heading angle estimation and 3D localization, is challenging. Successful modern day methods for 3D scene understanding require the use of a 3D sensor such as a depth camera, a stereo camera or LiDAR. On the other hand, single image based methods have significantly worse performance, but rightly so, as there is little explicit depth information in a 2D image. In this work, we aim at bridging the performance gap between 3D sensing and 2D sensing for 3D object detection by enhancing LiDAR-based algorithms to work with single image input. Specifically, we perform monocular depth estimation and lift the input image to a point cloud representation, which we call pseudo-LiDAR point cloud. Then we can train a LiDAR-based 3D detection network with our pseudo-LiDAR end-to-end. Following the pipeline of two-stage 3D detection algorithms, we detect 2D object proposals in the input image and extract a point cloud frustum from the pseudo-LiDAR for each proposal. Then an oriented 3D bounding box is detected for each frustum. To handle the large amount of noise in the pseudo-LiDAR, we propose two innovations: (1) use a 2D-3D bounding box consistency constraint, adjusting the predicted 3D bounding box to have a high overlap with its corresponding 2D proposal after projecting onto the image; (2) use the instance mask instead of the bounding box as the representation of 2D proposals, in order to reduce the number of points not belonging to the object in the point cloud frustum. Through our evaluation on the KITTI benchmark, we achieve the top-ranked performance on both bird's eye view and 3D object detection among all monocular methods, effectively quadrupling the performance over previous state-of-the-art.

1. Introduction

3D object detection from a single image (monocular vision) is an indispensable part of future autonomous driving and robot vision because a single cheap onboard camera is readily available in most modern cars. Successful modern day methods for 3D object detection heavily rely on 3D sensors, such as a depth camera, a stereo camera or a laser scanner (*i.e.*, LiDAR), which can provide explicit 3D information about the entire scene. The major disadvantages of this category of methods are: (1) the limited working range of the depth camera (often less than 10 meters); (2) the calibration and synchronization process of the stereo camera, causing it hard to scale on most modern cars; (3) the high cost of the LiDAR, especially when a high-resolution LiDAR is needed for detecting faraway objects accurately.

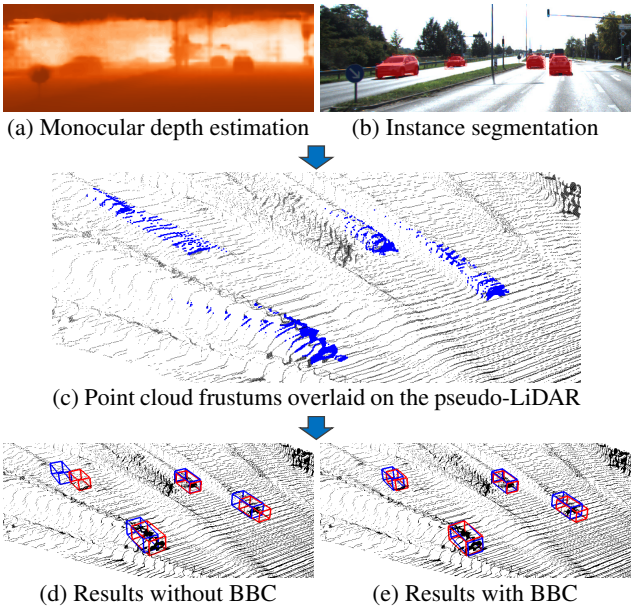


Figure 1: (a) Monocular depth estimation and (b) Instance segmentation from a single input image; (c) Extracted point cloud frustums **(blue)** overlaid on the pseudo-LiDAR **(black)**; 3D bounding box detection (**blue**) results (d) without bounding box consistency (BBC) and (e) with BBC. Ground truth shown in **red**.

On the other hand, a single camera, although cannot provide explicit depth information, is several orders of magnitude cheaper than the LiDAR and can capture the scene clearly up to approximately 100 meters. Although people have explored the possibility of monocular 3D object detection for a decade [75, 6, 73, 32, 74, 42, 12, 58, 30, 31, 21], state-of-the-art monocular methods can only yield drasti-

cally low performance in contrast to the high performance achieved by the LiDAR-based methods (*e.g.*, 13.6% average precision (AP) [58] vs. 86.5% AP [20] on the moderate set of cars of KITTI [14] dataset).

In this paper, we aim at bridging this performance gap between 3D sensing and 2D sensing for 3D object detection by extending LiDAR-based algorithms to work with single image input, without using the stereo camera, the depth camera, or the LiDAR. We introduce an intermediate 3D point cloud representation of the data, referred to as *"pseudo-LiDAR"*. Intuitively, we first perform monocular depth estimation and generate the pseudo-LiDAR for the entire scene by lifting every pixel within the image into its 3D coordinate given the estimated depth. Then we can train any LiDAR-based 3D detection network with the pseudo-LiDAR. Specifically, we extend a popular two-stage LiDAR-based 3D detection algorithm, Frustum PointNets [33]. Following the same pipeline, we detect 2D object proposals in the input image and extract a point cloud frustum from the pseudo-LiDAR for each 2D proposal. Then an oriented 3D bounding box is detected for each frustum.

In addition, we observe that there is a large amount of noise in the pseudo-LiDAR compared to the precise LiDAR point cloud due to the inaccurate monocular depth estimation. This noise often reflects in two ways: (1) The extracted point cloud frustum might be largely off and there is a *local misalignment* with respect to the LiDAR point cloud .This may result in a poor estimate of the object center location, especially for the faraway objects with more severe misalignment; (2) The extracted point cloud frustum always has a *long tail* – depth artifacts around the periphery of an object stretching back into the 3D space to form a tail shape – because the estimated depth is not accurate around the boundaries of the object. Therefore, predicting the object's size in 3D becomes challenging.

We propose two innovations to handle the above issues: (1) To alleviate the local misalignment, we use a 2D-3D bounding box consistency constraint, adjusting the predicted 3D bounding box to have a high overlap with its corresponding 2D detected proposals after projecting onto the image. During training, we formulate this constraint as a *bounding box consistency loss* (BBCL) to supervise the learning. During testing, a *bounding box consistency optimization* (BBCO) is solved subject to this constraint using a global optimization method to further improve the prediction results. (2) To cut off the long tail and reduce the number of points not belonging to the object in the point cloud frustum, we use the *instance mask* as the representation of the 2D proposals as opposed to using the bounding box in [33]. We argue that, in this way, the extracted point cloud frustum is much cleaner, and thus making it easier to predict the object's size.

Our pipeline is shown in Figure 2. To date, we achieve the top-ranked performance on bird's eye view and 3D object detection among all monocular methods, raising the accuracy by up to **22.0%** absolute AP, **quadrupling** the performance over previous state-of-the-art. We emphasize that we also achieve an improvement by up to **6.0%** absolute AP (relatively **14.2%**) over all concurrent works [50, 28, 36].

Our contributions are summarized as follows: (1) We propose a pipeline of monocular 3D object detection, enhancing the LiDAR-based methods to work with single image input; (2) We show empirically that the bottleneck of the proposed pipeline is the noise in the pseudo-LiDAR due to inaccurate monocular depth estimation; (3) We propose to use a bounding box consistency loss during training and a consistency optimization during testing to adjust the 3D bounding box prediction; (4) We demonstrate the benefit of using instance mask as the representation of the 2D detected proposals; (5) We achieve the state-of-the-art performance and show an unprecedented improvement over all monocular methods on standard 3D object detection benchmark.

2. Related Work

LiDAR-Based 3D Object Detection. Existing works have explored three ways of processing the LiDAR data for 3D object detection: (1) As the convolutional neural networks (CNNs) can naturally process images, many works focus on projecting the LiDAR point cloud into the bird's eye view (BEV) images as a pre-processing step and then regressing the 3D bounding box based on the features extracted from the BEV images [2, 54, 55, 24, 20, 62, 57, 61]; (2) On the other hand, one can divide the LiDAR point cloud into equally spaced 3D voxels and then apply 3D CNNs for 3D bounding box prediction [25, 60, 71]; (3) The most popular approach so far is to directly process the LiDAR point cloud through the neural network without pre-processing [22, 10, 44, 63, 59, 39, 40, 43, 11, 69, 16, 52, 33, 23]. To this end, novel neural networks that can directly consume the point cloud are developed [7, 34, 46, 67, 18, 51, 15]. Although LiDAR-based methods can achieve remarkable performance, they require that the high-resolution and precise LiDAR point cloud is available.

Monocular 3D Object Detection. Unlike LiDAR-based methods requiring the precise LiDAR point cloud, monocular methods only require a single image, posing the task of 3D object detection more challenging. [6] proposes to sample candidate bounding boxes in 3D and score their 2D projection based on the alignment with multiple semantic priors: shape, instance segmentation, context, and location. [28] introduces a differentiable ROI lifting layer to predict the 3D bounding box on top of the features extracted from the input image and depth estimate. On the other hand, instead of estimating the pixel-wise depth for the entire scene, [36] proposes a novel instance depth estimation module to

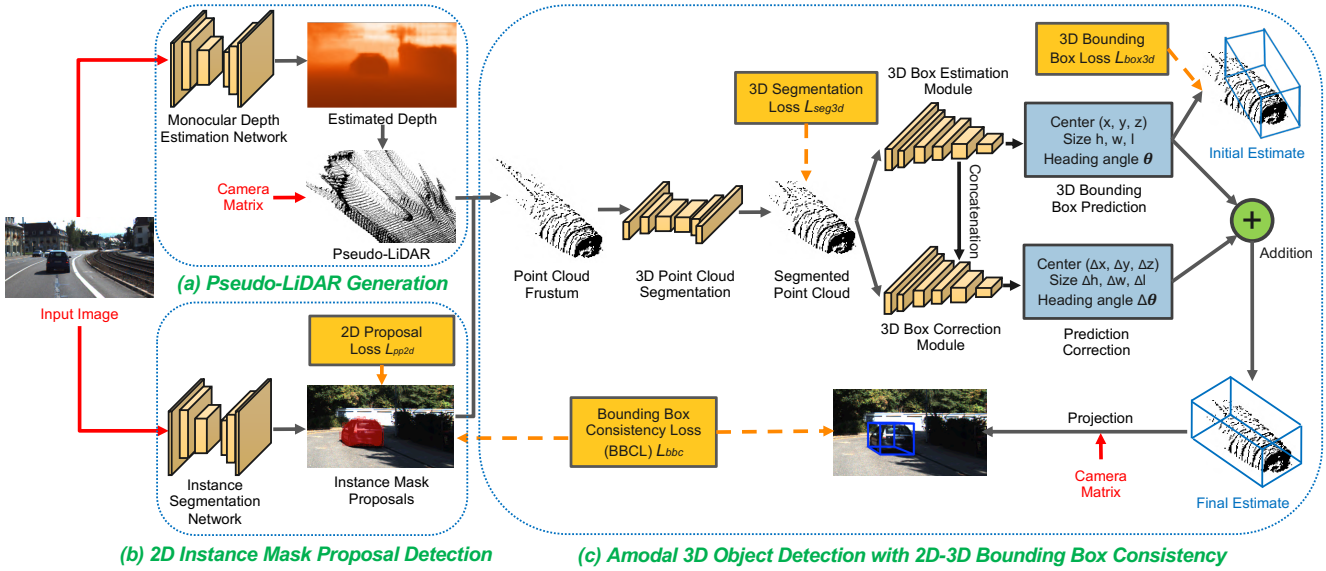


Figure 2: **Proposed Pipeline.** (a) Lift every pixel of input image to 3D coordinates given estimated depth to generate pseudo-LiDAR; (b) Instance mask proposals detected for extracting point cloud frustum; (c) 3D bounding box estimated **(blue)** for each point cloud frustum made to be consistent with corresponding 2D proposal. All inputs and losses indicated in **red** and **orange** respectively.

predict the depth of the targeting 3D bounding box's center. In order to avoid using a coarse approximation (*i.e.*, 3D bounding box) to the true 3D extent of objects, previous works [75, 12, 31, 73, 3, 56, 74, 21] have built fine-grained part-based models or leverage the existing CAD model collections [4] in order to exploit rich 3D shape priors and reason about occlusion in 3D. [32] enhances monocular 3D object detection algorithm to work with the image captured by 360° panoramic cameras.

Models leveraging the 2D-3D bounding box consistency constraint are also related to our work. [30] proposes to train a 2D CNN to estimate a subset of 3D bounding box parameters (*i.e.*, the object's size and orientation). During testing, they combine these estimates with the constraint to compute the remaining of parameters, namely the object center location. As a result, the prediction of the object center location highly relies on the accuracy of the orientation and object size estimates. In contrast, we train a successful PointNet-based 3D detection network and learn to predict the complete set of parameters. Also, we formulate the bounding box consistency constraint as a *differentiable loss* during training and a *constrained optimization* during testing to adjust 3D bounding box prediction. More importantly, we achieve an absolute AP improvement by up to 26.1% over [30] (from 5.6% to an unprecedented 31.7%) – a surprising 5× improvement in performance.

The work of [50] and [58] both estimate the depth and generate a pseudo-LiDAR point cloud from the single image input for 3D detection. We go one step beyond them by observing the local misalignment and long tail issues in the noisy pseudo-LiDAR and propose to use bounding box consistency constraint as a supervision signal and instance mask as the representation of the 2D proposals to mitigate the issues. We also show an absolute AP improvement by up to 21.2% and 6.0% over [58] and [50] respectively.

Supervision via Consistency. Formulating a well-known geometry constraint to a differentiable loss for training not only provides a supervision signal for free but also makes the outputs of the model geometrically consistent with each other. [9] proposes a registration loss to train a facial landmark detector, forcing the outputs are consistent across adjacent frames. [27, 64, 26, 65, 35] jointly predict the depth and surface normal with a consistency loss forcing two outputs are compatible with each other. The multi-view supervision loss is proposed in [47, 38, 48, 66, 68, 19], making the prediction consistent across viewpoints. In addition, [72, 41, 53, 5, 1, 70] propose the cycle consistency loss, in the sense that if we translate our prediction into other domain and translate back, we should arrive back to the original input. In terms of consistency across dimensions, [49, 21, 29] propose an inverse-graphics framework, which makes the prediction in 3D and ensures its 2D projection consistent with the 2D input. Similarly, our proposed BBCL forces the projection of the predicted 3D bounding box to be consistent with its 2D detected proposal.

3. Approach

Our goal is to estimate the oriented 3D bounding box of objects from only a single RGB image. During both train-

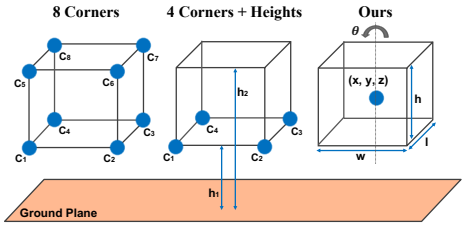


Figure 3: Comparison of the 3D bounding box parameterization between 8 corners [57], 4 corners with heights [20] and ours. Our compact parameterization requires minimal number of parameters for an oriented 3D bounding box.

ing and testing, we do not require any data from the LiDAR, stereo and depth camera. The only assumption is that the camera matrix is known. Following [33], we parameterize our 3D bounding box output as a set of seven parameters, including the 3D coordinate of the object center (x, y, z), object's size h, w, l and its heading angle θ. Visualization of our parameterization compared to others is illustrated in Figure 3. We argue that our compact parameterization requires the minimal number of parameters for an oriented 3D bounding box.

In Figure 2, our pipeline consists of: (1) pseudo-LiDAR generation, (2) 2D instance mask proposal detection and (3) amodal 3D object detection with 2D-3D bounding box consistency. Based on the pseudo-LiDAR and instance mask proposals, point cloud frustums can be extracted, which are passed to train the amodal 3D detection network. The bounding box consistency loss and bounding box consistency optimization are used to adjust the 3D box estimate.

3.1. Pseudo-LiDAR Generation

Monocular Depth Estimation. To lift the input image to the pseudo-LiDAR point cloud, a depth estimate is needed. Thanks to the successful work called *DORN* [13], we directly adopt it as a sub-network in our pipeline and initialize it using pre-trained weights. For convenience, we do not update the weights of the depth estimation network during training, and it can be regarded as an off-line module to provide the depth estimate. As our pipeline is agnostic to the choice of monocular depth estimation network, we can replace it with other networks if necessary.

Pseudo-LiDAR Generation. Our proposed pipeline can enhance the LiDAR-based 3D detection network to work with single image input, without the need for 3D sensors. To this end, generating a point cloud from the input image that can mimic the LiDAR data is the essential step. Given the depth estimate and camera matrix, deriving the 3D location (X_c, Y_c, Z_c) in the camera coordinate for each pixel (u, v) is simply as:

$$X_c = \frac{(u - c_x)Z_c}{f_x} \quad (1)$$

$$Y_c = \frac{(v - c_y)Z_c}{f_y} \quad (2)$$

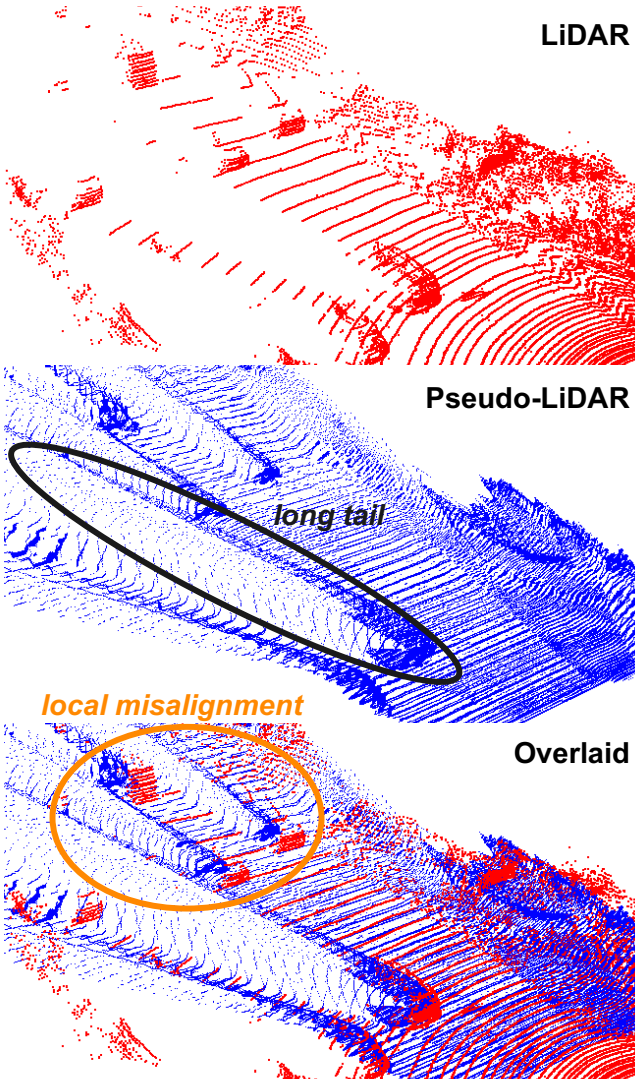


Figure 4: Comparison between the **LiDAR** (top), **pseudo-LiDAR** (middle) and an overlaid version (bottom). Two types of noise discussed in Section 3.1 are indicated in **orange** (local misalignment) and **black** (long tail) eclipses.

where Z_c is the estimated depth of the pixel in the camera coordinate and (c_x, c_y) is the pixel location of the camera center. f_x and f_y are the focal length of the camera along x and y axes. Given the camera extrinsic matrix $C = [R\ t]$, one can also obtain the 3D location of the pixel in the world coordinate (X, Y, Z) by computing $C^{-1}[X_c, Y_c, Z_c]^T$ and dividing by the last element. We refer to this generated 3D point cloud as *pseudo-LiDAR*.

Pseudo-LiDAR vs. LiDAR Point Cloud. To make sure the pseudo-LiDAR is compatible with the LiDAR-based algorithms, it is natural to compare the pseudo-LiDAR with the LiDAR point cloud via visualization. An example is shown in Figure 4. We observe that, although the generated pseudo-LiDAR aligns well with the precise LiDAR

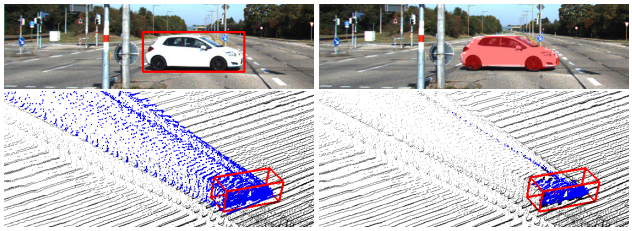
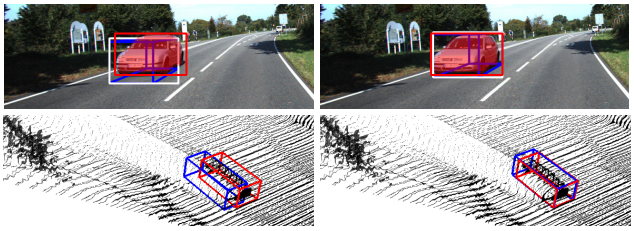


Figure 5: **Effectiveness of Instance Mask Proposal.** Top left: 2D box proposal. Top right: Instance mask proposal. Bottom left: Point cloud frustum lifted from 2D box proposal with noisy long tail. Bottom right: Point cloud frustum lifted from instance mask proposal with no tail. Ground truth box corresponding to the frustum shown in **red**.



(a) Results without BBC (b) Results with BBC

Figure 6: **Effect of Bounding Box Consistency (BBC).** Top Row: Minimum bounding rectangle (MBR) of 3D box estimate (white), instance mask (**red**). Bottom left: Poor 3D box estimate without BBC. Bottom right: Improved 3D box estimate with BBC. Ground truth shown in **red**.

point cloud in terms of the *global* structure, there is a large amount of *local noise* in the pseudo-LiDAR due to inaccurate monocular depth estimation. This noise often reflects in two ways: (1) The extracted point cloud frustum might be largely off and there is a *local misalignment* with respect to the LiDAR point cloud. This may result in a poor estimate of the object center location, especially for the faraway objects with more severe misalignment. For example, in the **orange** eclipse of Figure 4, the point cloud frustums fall behind their LiDAR counterpart; (2) The point cloud frustum extracted from the pseudo-LiDAR often has a *long tail* because the estimated depth is not accurate around the boundaries of the object. Therefore, predicting the size of the objects becomes challenging. An example of point cloud frustum with the long tail is shown in the **black** eclipse of Figure 4.

In addition, a distinction of the pseudo-LiDAR from the LiDAR point cloud is the density of the point cloud. Although a high-cost LiDAR can provide high-resolution point cloud, the number of LiDAR points is still at least one order of magnitude less than the pseudo-LiDAR point cloud. We will show how the density of the point cloud affects the performance in the experiment section.

3.2. 2D Instance Mask Proposal Detection

In order to generate a point cloud frustum for each object, we first detect an object proposal in 2D. Unlike previous works using the bounding box as the representation of the 2D proposals [52, 33, 50, 58], we claim that it is better to use the instance mask, especially when the point cloud frustum is extracted from the noisy pseudo-LiDAR and thus has a large number of redundant points. We compare the generated point cloud frustum corresponding to the bounding box and instance mask proposal in Figure 5. In the left column, we demonstrate that, when we lift all the pixels within the 2D bounding box proposal into 3D, the generated point cloud frustum has the *long tail* issue as discussed in Section 3.1. On the other hand, in the right column of Figure 5, lifting only the pixels within the instance mask proposal significantly removes the points not being enclosed by the ground truth box, resulting in a point cloud frustum with no tail. Specifically, we consider the Mask R-CNN [17] as our instance segmentation network.

3.3. Amodal 3D Object Detection

Based on the generated pseudo-LiDAR and 2D instance mask proposals, we can extract a set of point cloud frustums, which are then passed to train a two-stage LiDAR-based 3D detection algorithm for 3D bounding box prediction. In this paper, we experiment with Frustum PointNets [33]. In brief, we segment the point cloud frustum in 3D to further remove the points not belonging to the objects. Then we sample a fixed number of points from the segmented point cloud for 3D bounding box estimation, including estimating the center (x, y, z), size h, w, l and heading angle θ. For more details of this part, please refer to the Frustum PointNets [33].

3.4. 2D-3D Bounding Box Consistency (BBC)

To alleviate the local misalignment issue, we use the geometry constraint of the bounding box consistency to refine our 3D bounding box estimate. Given an inaccurate 3D bounding box estimate, it is highly possible that its 2D projection also does not match well with the corresponding 2D proposal. An example is shown in Figure 6a. By adjusting the 3D bounding box estimate in 3D space so that its 2D projection can have a higher 2D Intersection of Union (IoU) with the corresponding 2D proposal, we demonstrate that the 3D IoU of 3D bounding box estimate with its ground truth can be also increased, shown in Figure 6b.

Formally, we first convert the 3D bounding box estimate $(x,\ y,\ z,\ h,\ w,\ l,\ \theta)$ to the 8 corner representation $\left\{(p_x^n, p_y^n, p_z^n)\right\}_{n=1}^{8}$. Then its 2D projection $\{(u^n, v^n)\}_{n=1}^{8}$ can be computed given the camera projection matrix. Then we can compute the minimum bounding rectangle (MBR), which is a tuple $t^e = (t_x^e, t_y^e, t_w^e, t_h^e)$, representing the smallest axis-aligned 2D bounding box that can enclose the 2D point set $\{(u^n, v^n)\}_{n=1}^{8}$. Similarly, we can obtain the MBR

of the 2D mask proposal $t^p = (t^p_x, t^p_y, t^p_w, t^p_h)$. The goal of the optimization is to increase the 2D IoU between the 2D bounding box t^e and t^p.

Bounding Box Consistency Loss (BBCL). During training, we propose a PointNet-based 3D box correction module[1] for bounding box refinement. The 3D box correction module takes the segmented point cloud and features extracted from the 3D box estimation module as the input, and outputs a correction of the 3D bounding box parameters (*i.e.*, a residual). Then our final estimate E_f is the summation over the initial estimate E_i and the residual. The loss can be formulated as follows:

$$L_{bbc} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t^e_i - t^p_i) \tag{3}$$

Where t^e and t^p can be computed deterministically from the final estimate E_f and 2D mask proposal respectively as described in Section 3.4. As the gradients can be back-propagated through the entire network, we can thus train our 3D detection network with BBCL end-to-end.

Bounding Box Consistency Optimization (BBCO). During testing, we further refine the final estimate with the BBC constraint as a post-processing step. For each pair of the 3D bounding box estimate and its 2D proposal, we solve the same optimization problem and minimize the L_{bbc} in Equation 3 using a global search optimization method.

4. Experiments

4.1. Settings

Dataset. We evaluate on the KITTI bird's eye view and 3D object detection benchmark [14], containing 7481 training and 7518 testing images as well as the corresponding LiDAR point clouds, stereo images, and full camera matrix. We use the same training and validation split as [33]. We emphasize again, during training and testing, our approach does not use any LiDAR point cloud or stereo image data.

Evaluation Metric. We use the evaluation toolkit provided by KITTI, which computes the precision-recall curves and average precision (AP) with the IoU thresholds at 0.5 and 0.7. We denote the AP for the bird's eye view (BEV) and 3D object detection as AP_{BEV} and AP_{3D} respectively.

Baselines. We compare our method with previous state-of-the-art: Mono3D [6], Deep3DBox [30] and MLF-MONO [58]. To show the superiority of our method, we also compare with three recent concurrent works: ROI-10D [28], MonoGRNet [36] and PL-MONO [50].

4.2. Implementation Details

2D Instance Mask Proposal Detection. As only 200 training images with pixel-wise annotation are provided by KITTI instance segmentation benchmark, it is not enough for training an instance segmentation network from scratch. Therefore, we first train our instance segmentation network[2] on Cityscapes dataset [8] with 3475 training images and then fine-tune on the KITTI dataset.

Amodal 3D Object Detection. To analyze the full potential of the Frustum PointNets [33] for 3D object detection with pseudo-LiDAR, we experiment with its different variants in our ablation study: (1) Removing the intermediate supervision from the 3D segmentation loss L_{seg3d} so that network can only implicitly learn to segment point cloud via minimizing the 3D bounding box loss L_{box3d}; (2) Removing the TNet proposed in [33] for object center regression and learning to predict the object center location using the 3D box estimation module; (3) Varying number of points sampled from the segmented point cloud to show the effect of point cloud density.

Bounding Box Consistency Optimization (BBCO). We use the differential evolution [45] as our global search optimization method to refine our 3D bounding box estimate during testing. The final estimate from the network is used as the initialization of the optimization method. The bounds of the 3D bounding box parameters are linearly increasing based on the object's depth, *i.e.*, the further the objects are, the more their 3D bounding box can be adjusted.

4.3. Experimental Results

Comparison with State-of-the-Art Methods. We summarize the bird's eye view and 3D object detection results (AP_{BEV} and AP_{3D}) on KITTI val set in Table 1. Our method consistently outperforms all monocular methods by a large margin on all levels of difficulty with different evaluation metrics. We highlight that, at IoU = 0.7 (moderate) – the metric used to rank algorithms on the KITTI leader board – we nearly **quadruple** the AP_{3D} performance over previous state-of-the-art (from 5.7 by MLF-MONO [58] to 21.0 by ours). We emphasize that we also achieve an improvement by up to **6.0%** (from 42.3% by PL-MONO [50] to 48.3% by ours) absolute AP_{3D} (relatively **14.2%**) over the best performed concurrent work on the moderate set at IoU = 0.5. Examples of our 3D bounding box estimate on KITTI val set are visualized in Figure 7.

Results on Pedestrian and Cyclist. We report AP_{BEV} and AP_{3D} results on KITTI val set for pedestrians and cyclists at IoU = 0.5 in Table 2. We emphasize that the bird's eye view and 3D object detection from a single image for pedestrians and cyclists are much more challenging than cars due to the small sizes of the objects. Therefore, none[3] of prior monoc-

[1] Details of the specific architectures are described in the supplementary

[2] Details about the performance of our instance segmentation network are in the supplementary material.

[3] To avoid confusion, we note that [50] is the first to present results on pedestrians and cyclists from stereo input instead of monocular input.

Table 1: Quantitative comparison on KITTI **val** set. We report the average precision (in %) of car category on bird's eye view and 3D object detection as AP_{BEV} and AP_{3D}. Methods in top three rows are previous state-of-the-art and methods in middle three rows colored in green are concurrent works. Our method consistently outperforms all monocular methods.

Method	AP_{BEV} / AP_{3D} (in %), **IoU = 0.5**			AP_{BEV} / AP_{3D} (in %), **IoU = 0.7**		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [6]	30.5 / 25.2	22.4 / 18.2	19.2 / 15.5	5.2 / 2.5	5.2 / 2.3	4.1 / 2.3
Deep3DBox [30]	30.0 / 27.0	23.8 / 20.6	18.8 / 15.9	10.0 / 5.6	7.7 / 4.1	5.3 / 3.8
MLF-MONO [58]	55.0 / 47.9	36.7 / 29.5	31.3 / 26.4	22.0 / 10.5	13.6 / 5.7	11.6 / 5.4
ROI-10D [28]	46.9 / 37.6	34.1 / 25.1	30.5 / 21.8	14.5 / 9.6	9.9 / 6.6	8.7 / 6.3
MonoGRNet [36]	- / 50.5	- / 37.0	- / 30.8	- / 13.9	- / 10.2	- / 7.6
PL-MONO [50]	70.8 / 66.3	49.4 / 42.3	42.7 / 38.5	40.6 / 28.2	26.3 / 18.5	22.9 / 16.4
Ours	**72.1 / 68.4**	**53.1 / 48.3**	**44.6 / 43.0**	**41.9 / 31.5**	**28.3 / 21.0**	**24.5 / 17.5**

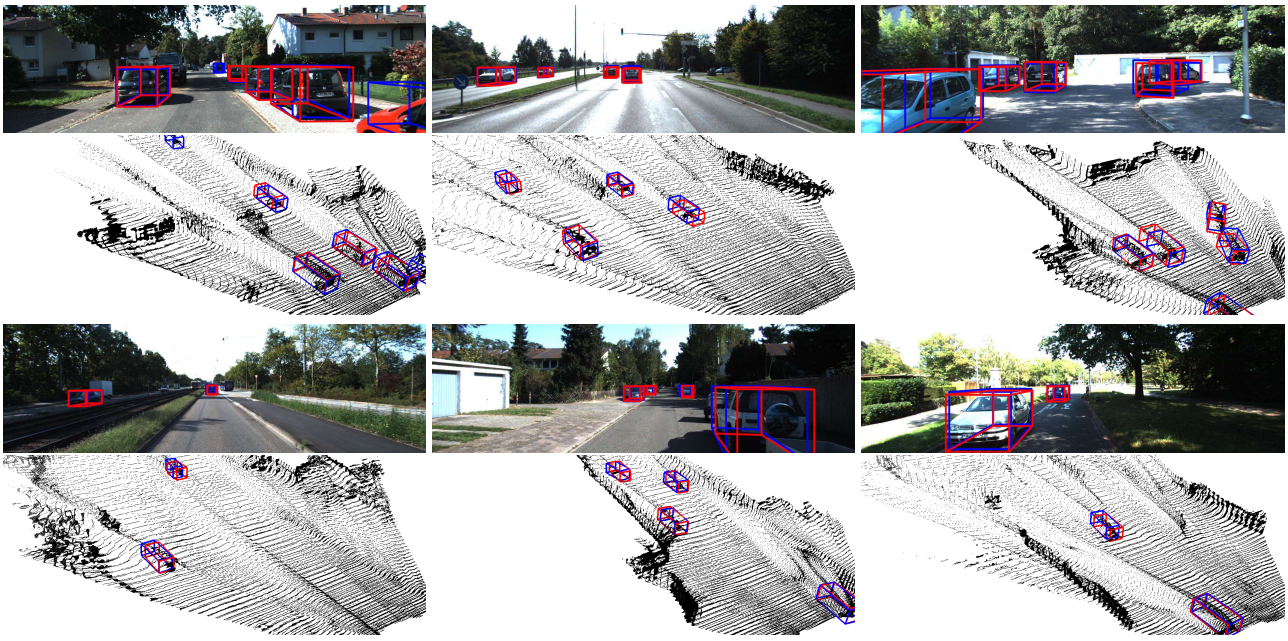


Figure 7: Qualitative results of our proposed method on KITTI **val** set. We visualize our 3D bounding box estimate (in blue) and ground truth (in red) on the frontal images (1st and 3rd rows) and pseudo-LiDAR point cloud (2nd and 4th rows).

Table 2: AP_{BEV} / AP_{3D} performance on KITTI **val** set for pedestrians and cyclists at IoU = 0.5.

Category	Easy	Moderate	Hard
Pedestrian	14.4 / 11.6	13.8 / 11.2	12.0 / 10.9
Cyclist	11.0 / 8.5	7.7 / 6.5	6.8 / 6.5

ular works has ever reported the results for pedestrians and cyclists. Although our reported AP_{BEV} and AP_{3D} performance for pedestrians and cyclists are significantly worse than for cars, we argue that this is a good starting point for future monocular work.

4.4. Ablation Study

Unless otherwise mentioned, we conduct all the ablative analysis by comparing different variants against our baseline, which does not use the instance mask as the representation of the 2D proposal and bounding box consistency to refine the 3D bounding box estimate. Specifically, for our baseline, we use 2D bounding boxes detected by the Faster R-CNN [37] as the 2D proposals, use the DORN [13] to estimate the depth in order to generate the pseudo-LiDAR, and follow the original Frustum PointNet [33] for 3D bounding box estimation. We train our baseline from scratch by random initializing its weights and sample 512 points from the segmented point cloud for 3D bounding box estimation. All **positive** ablative analysis is summarized in Table 3. The best-performed model, also illustrated in Figure 2, is the combination of using instance mask proposals, training with BBCL, testing with BBCO and removing the TNet from the Frustum PointNets.

Instance Mask vs. Bounding Box Proposal. We replace the bounding box proposals in the baseline with our pro-

Table 3: Summarized **positive** ablative analysis on KITTI **val** set. We show individual and combined effects of using instance mask proposal (+Mask), training with bounding box consistency loss (+BBCL), testing with bounding box consistency optimization (+BBCO) and removing the TNet from the amodal 3D detection network (-TNet).

Method	AP_{BEV} / AP_{3D} (in %), **IoU = 0.5**			AP_{BEV} / AP_{3D} (in %), **IoU = 0.7**		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Ours (baseline)	71.4 / 66.2	49.8 / 42.5	42.8 / 38.6	40.4 / 28.9	26.5 / 18.2	22.9 / 16.2
Ours + Mask	70.8 / 64.7	51.4 / 44.5	44.4 / 40.4	41.2 / 29.4	27.8 / 19.8	24.2 / 17.5
Ours + BBCO	71.9 / 68.2	50.4 / 46.6	43.3 / 40.9	**42.0** / **31.7**	27.4 / 20.8	23.3 / 17.1
Ours + BBCL	71.7 / **68.5**	50.3 / 46.5	43.2 / 40.5	41.6 / 31.3	27.0 / 20.8	23.1 / 17.1
Ours - TNet	70.4 / 66.0	49.8 / 42.6	42.7 / 38.6	41.7 / 29.4	26.4 / 18.5	23.0 / 16.4
Ours + Mask + BBCO	71.1 / 67.7	52.1 / 48.2	**44.8** / 42.3	40.7 / 28.9	27.4 / 20.0	24.0 / 17.1
Ours + Mask + BBCO - TNet	71.1 / 68.1	52.3 / **48.3**	**44.8** / 42.2	41.5 / 28.5	28.3 / 20.3	24.1 / 17.2
Ours + Mask + BBCO - TNet + BBCL	**72.1** / 68.4	**53.1** / **48.3**	44.6 / **43.0**	41.9 / 31.5	**28.3** / **21.0**	**24.5** / **17.5**

Table 4: Effect of 3D segmentation loss L_{seg3d}. AP_{BEV} and AP_{3D} results on KITTI **val** set for car category at IoU = 0.7.

loss L_{seg3d}	Easy	Moderate	Hard
w/ (baseline)	40.4 / 28.9	26.5 / 18.2	22.9 / 16.2
w/o	32.9 / 21.8	22.4 / 15.5	20.4 / 14.8

Table 5: Effect of point cloud density. AP_{BEV} and AP_{3D} results on KITTI **val** set for car category at IoU = 0.7.

Num. of Points	Easy	Moderate	Hard
4096	41.1 / 29.0	26.9 / 18.4	23.1 / 16.4
2048	41.1 / 28.9	26.3 / 18.2	22.9 / 16.2
1024	40.7 / 29.2	26.0 / 18.2	22.9 / 16.1
512 (baseline)	40.4 / 28.9	26.5 / 18.2	22.9 / 16.2
256	41.8 / 29.1	26.5 / 18.3	23.0 / 16.2

posed instance mask proposals. In Table 3, we observe Ours+Mask consistently outperforms Ours (baseline) about 1-2% AP on all subsets except for the easy set at IoU = 0.5.

Effect of Bounding Box Consistency. In Table 3, we compare Ours (baseline) with Ours+BBCL (training the baseline model with bounding box consistency loss) and Ours+BBCO (applying bounding box consistency optimization during testing). We show that either BBCL or BBCO improves the performance significantly, *e.g.*, AP_{3D} from 42.5% to 46.6% in the moderate set at IoU = 0.5.

Removing the TNet. We observe a mild improvement when comparing Ours-TNet with Ours (baseline) at IoU = 0.7 in Table 3. On the other hand, removing the TNet does not make any obvious difference on all sets at IoU = 0.5.

Effect of 3D Segmentation Loss. In Table 4, we also compare our baseline with the variant trained without the 3D segmentation loss L_{seg3d}. We observe a significant performance drop, meaning that it is difficult to learn the point cloud segmentation network without direct supervision.

Effect of Point Cloud Density. In Table 5, we compare models trained with the different number of points sampled from the segmented point cloud before feeding into the 3D box estimation module. Surprisingly, it turns out increasing the point cloud density (*e.g.*, from 512 to 4096 points) does not improve the performance.

Table 6: Fine-tuning vs. training from scratch. AP_{BEV} and AP_{3D} results on KITTI **val** set for car category at IoU = 0.7.

Initialization	Easy	Moderate	Hard
random (baseline)	40.4 / 28.9	26.5 / 18.2	22.9 / 16.2
pre-trained	40.6 / 27.1	26.1 / 18.1	22.6 / 16.0

Fine-Tuning vs. Training from Scratch. In Table 6, we compare our baseline (*i.e.*, training with randomly initialized weights) with its variant, which initializes the weights from the pre-trained model of Frustum PointNets. Surprisingly, training with the pre-trained weights slightly drops the performance. We argue that it is because the pre-trained model provided by Frustum PointNets might have overfitted on the LiDAR point cloud data and cannot be easily adapted to consume our pseudo-LiDAR input.

5. Conclusion

In this paper, we propose a novel monocular 3D object detection pipeline that can enhance LiDAR-based algorithms to work with single image input, without the need of 3D sensors (*e.g.*, the stereo camera, the depth camera or the LiDAR). The essential step of the proposed pipeline is to lift the 2D input image to a 3D point cloud, which we call *pseudo-LiDAR* point cloud. To handle the *local misalignment* and *long tail* issues caused by the noise in the pseudo-LiDAR, we propose to (1) use a 2D-3D bounding box consistency constraint to refine our 3D box estimate; (2) use the instance mask proposal to generate the point cloud frustum. Importantly, our method achieves the top-ranked performance on KITTI bird's eye view and 3D object detection benchmark among all monocular methods, quadrupling the performance over previous state-of-the-art. Although our focus is monocular 3D object detection, our method can be easily extended to work with stereo image input.

References

[1] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh. Recycle-GAN: Unsupervised Video Retargeting. *ECCV*, 2018. 3

[2] J. Beltran, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. de la Escalera. BirdNet: A 3D Object Detection Framework from LiDAR information. *ITSC*, 2018. 2

[3] F. Chabot, M. Chaouch, and J. Rabarisoa. Deep MANTA: a Coarse-to-Fine Many-Task Network for Joint 2D and 3D Vehicle Analysis from Monocular Image. *CVPR*, 2017. 3

[4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv*, 2015. 3

[5] H. Chang, J. Lu, A. Research, F. Yu, and A. Finkelstein. PairedCycleGAN: Asymmetric Style Transfer for Applying and Removing Makeup. *CVPR*, 2018. 3

[6] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3D Object Detection for Autonomous Driving. *CVPR*, 2016. 1, 2, 6, 7

[7] I. Cherabier, C. Hane, M. R. Oswald, and M. Pollefeys. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, 2017. 2

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *CVPR*, 2016. 6

[9] X. Dong, S.-i. Yu, X. Weng, S.-e. Wei, Y. Yang, and Y. Sheikh. Supervision-by-Registration: An Unsupervised Approach to Improve the Precision of Facial Landmark Detectors. *CVPR*, 2018. 3

[10] X. Du, M. H. Ang, S. Karaman, and D. Rus. A General Pipeline for 3D Detection of Vehicles. *ICRA*, 2018. 2

[11] 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. *ICRA*, 2017. 2

[12] S. Fidler, S. Dickinson, and R. Urtasun. 3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model. *NIPS*, 2012. 1, 3

[13] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep Ordinal Regression Network for Monocular Depth Estimation. *CVPR*, 2018. 4, 7

[14] A. Geiger, P. Lenz, and R. Urtasun. Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite. *CVPR*, 2012. 2, 6

[15] B. Graham, M. Engelcke, and L. van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. *CVPR*, 2018. 2

[16] F. Gustafsson and E. Linder-Norén. Automotive 3D Object Detection Without Target Domain Annotations. *Technical Report*, 2018. 2

[17] K. He, G. Gkioxari, P. Doll, and R. Girshick. Mask R-CNN. *ICCV*, 2017. 5

[18] B.-S. Hua, M.-K. Tran, and S.-K. Yeung. Pointwise Convolutional Neural Networks. *CVPR*, 2018. 2

[19] Y. Jafarian, Y. Yao, and H. S. Park. MONET: Multiview Semi-Supervised Keypoint via Epipolar Divergence. *arXiv*, 2018. 3

[20] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. *IROS*, 2018. 2, 4

[21] A. Kundu, Y. Li, and J. M. Rehg. 3D-RCNN: Instance-level 3D Object Reconstruction via Render-and-Compare. *CVPR*, 2018. 1, 3

[22] J. Lahoud and B. Ghanem. 2D-Driven 3D Object Detection in RGB-D Images. *ICCV*, 2017. 2

[23] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. *arXiv*, 2018. 2

[24] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep Continuous Fusion for Multi-Sensor 3D Object Detection. *ECCV*, 2018. 2

[25] W. Luo, B. Yang, and R. Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. *CVPR*, 2018. 2

[26] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised Learning of Depth and Ego-Motion from Monocular Video Using 3D Geometric Constraints. *CVPR*, 2018. 3

[27] Y. Man, X. Weng, and K. Kitani. GroundNet: Segmentation-Aware Monocular Ground Plane Estimation with Geometric Consistency. *arXiv:1811.07222*, 2018. 3

[28] F. Manhardt, W. Kehl, and A. Gaidon. ROI-10D: Monocular Lifting of 2D Detection to 6D Pose and Metric Shape. *arXiv*, 2019. 2, 6, 7

[29] P. Moreno, C. K. Williams, C. Nash, and P. Kohli. Overcoming Occlusion with Inverse Graphics. *ECCV*, 2016. 3

[30] A. Mousavian, D. Anguelov, J. Košecká, and J. Flynn. 3D Bounding Box Estimation Using Deep Learning and Geometry. *CVPR*, 2017. 1, 3, 6, 7

[31] M. Oberweger, M. Rad, and V. Lepetit. Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. *ECCV*, 2018. 1, 3

[32] G. Payen de La Garanderie, A. Atapour Abarghouei, and T. P. Breckon. Eliminating the Blind Spot: Adapting 3D Object Detection and Monocular Depth Estimation to 360 Panoramic Imagery. *ECCV*, 2018. 1, 3

[33] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. *CVPR*, 2018. 2, 4, 5, 6, 7

[34] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NIPS*, 2017. 2

[35] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia. GeoNet: Geometric Neural Network for Joint Depth and Surface Normal Estimation. *CVPR*, 2018. 3

[36] Z. Qin, J. Wang, and Y. Lu. MonoGRNet: A Geometric Reasoning Network for Monocular 3D Object Localization. *arXiv*, 2018. 2, 6, 7

[37] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *NIPS*, 2015. 7

[38] P. Sermanet, C. Lynch, J. Hsu, and S. Levine. Time-Contrastive Networks: Self-Supervised Learning from Multi-view Observation. *CVPRW*, 2017. 3

[39] S. Shi, X. Wang, and H. Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. *arXiv*, 2019. 2

[40] K. Shin, Y. P. Kwon, and M. Tomizuka. RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement. *arXiv*, 2018. 2

[41] J. Song, K. Pang, Y.-Z. Song, T. Xiang, and T. Hospedales. Learning to Sketch with Shortcut Cycle Consistency. *CVPR*, 2018. 3

[42] S. Song and M. Chandraker. Joint SFM and Detection Cues for Monocular 3D Localization in Road Scenes. *CVPR*, 2015. 1

[43] S. Song and J. Xiao. Sliding Shapes for 3D Object Detection in Depth Images. *ECCV*, 2014. 2

[44] S. Song and J. Xiao. Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images. *CVPR*, 2016. 2

[45] R. Storn and K. Price. Differential Evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 1997. 6

[46] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. *CVPR*, 2018. 2

[47] S. Tulsiani, A. A. Efros, and J. Malik. Multi-View Consistency as Supervisory Signal for Learning Shape and Pose Prediction. *CVPR*, 2018. 3

[48] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-View Supervision for Single-View Reconstruction via Differentiable Ray Consistency. *CVPR*, 2017. 3

[49] H. Y. F. Tung, A. W. Harley, W. Seto, and K. Fragkiadaki. Adversarial Inverse Graphics Networks: Learning 2D-to-3D Lifting and Image-to-Image Translation from Unpaired Supervision. *ICCV*, 2017. 3

[50] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger. Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. *arXiv*, 2019. To appear in CVPR 2019. 2, 3, 5, 6, 7

[51] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *arXiv*, 2018. 2

[52] Z. Wang and K. Jia. Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection. *arXiv*, 2019. 2, 5

[53] X. Weng. CyLKs: Unsupervised Cycle Lucas-Kanade Network for Landmark Tracking. *arXiv:1811.11325*, 2018. 3

[54] S. Wirges, T. Fischer, C. Stiller, and J. B. Frias. Object Detection and Classification in Occupancy Grid Maps Using Deep Convolutional Networks. *ITSC*, 2018. 2

[55] S. Wirges, M. Reith-Braun, M. Lauer, and C. Stiller. Capturing Object Detection Uncertainty in Multi-Layer Grid Maps. *arXiv*, 2019. 2

[56] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-Driven 3D Voxel Patterns for Object Category Recognition. *CVPR*, 2015. 3

[57] C. Xiaozhi, M. Huimin, W. Ji, L. Bo, and X. Tian. Multi-View 3D Object Detection Network for Autonomous Driving. *CVPR*, 2017. 2, 4

[58] B. Xu and Z. Chen. Multi-Level Fusion based 3D Object Detection from Monocular Images. *CVPR*, 2018. 1, 2, 3, 5, 6, 7

[59] D. Xu, D. Anguelov, and A. Jain. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. *CVPR*, 2018. 2

[60] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 2

[61] B. Yang, M. Liang, and R. Urtasu. HDNET: Exploiting HD Maps for 3D Object Detection. *CoRL*, 2018. 2

[62] B. Yang, W. Luo, and R. Urtasun. PIXOR: Real-time 3D Object Detection from Point Clouds. *CVPR*, 2018. 2

[63] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. IPOD: Intensive Point-based Object Detector for Point Cloud. *arXiv*, 2018. 2

[64] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia. LEGO: Learning Edge with Geometry all at Once by Watching Videos. *CVPR*, 2018. 3

[65] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised Learning of Geometry with Edge-aware Depth-Normal Consistency. *AAAI*, 2018. 3

[66] Y. Yao and H. S. Park. Multiview Cross-Supervision for Semantic Segmentation. *arXiv*, 2018. 3

[67] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. PU-Net: Point Cloud Upsampling Network. *CVPR*, 2018. 2

[68] Y. Zhang and H. S. Park. Multiview Supervision By Registration. *arXiv*, 2018. 3

[69] X. Zhao, Z. Liu, R. Hu, and K. Huang. 3D Object Detection Using Scale Invariant and Feature Reweighting Networks. *arXiv*, 2019. 2

[70] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros. Learning Dense Correspondence via 3D-guided Cycle Consistency. *CVPR*, 2016. 3

[71] Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *CVPR*, 2018. 2

[72] J.-y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *ICCV*, 2017. 3

[73] M. Z. Zia, M. Stark, and K. Schindler. Explicit Occlusion Modeling for 3D Object Class Representations. *CVPR*, 2013. 1, 3

[74] M. Z. Zia, M. Stark, and K. Schindler. Are Cars just 3D Boxes? Jointly Estimating the 3D Shape of Multiple Objects. *CVPR*, 2014. 1, 3

[75] M. Z. Zia, M. Stark, and K. Schindler. Towards Scene Understanding with Detailed 3D Object Representations. *IJCV*, 2015. 1, 3

Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud Supplementary Material

Xinshuo Weng
Carnegie Mellon University
xinshuow@cs.cmu.edu

Kris Kitani
Carnegie Mellon University
kkitani@cs.cmu.edu

0. Overview

This document provides additional technical details, extra experiments, more visualization and justification of our idea. Each section in this document corresponds to the subsection of the approach section in the main paper.

1. Pseudo-LiDAR Generation

Additional Visualization of LiDAR vs. Pseudo-LiDAR

We provide the additional visual comparison between the LiDAR and pseudo-LiDAR point cloud in Figure 2, demonstrating again the *local misalignment* and *long tail* issues we have observed in the pseudo-LiDAR point cloud.

2. 2D Instance Mask Proposal Detection

Justification of Using Instance Mask Proposal for 3D Point Cloud Segmentation and 3D Box Estimation

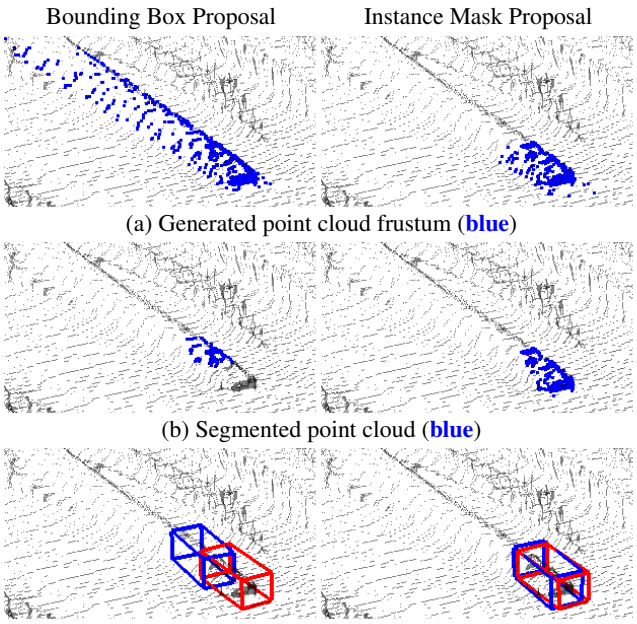
In the main paper, we justify the effectiveness of using instance mask proposal to generate the point cloud frustum with no tail. We provide further details here about how the generated point cloud frustum with no tail can improve the results in the subsequent 3D point cloud segmentation and 3D bounding box estimation module.

An example of visualization is shown in Figure 1. In the left column, the point cloud frustum is generated from the bounding box proposal and has a long tail, making the 3D point cloud segmentation task difficult (*e.g.*, in the middle left of the figure, the segmented point cloud misses lots of points belonging to the object and still contains background points). This further causes a poor 3D box estimation, especially a poor object center estimate. On the other hand, the point cloud frustum generated from the instance mask proposal with no tail, shown in the right column, can reduce a large number of background points so that the subsequent point cloud segmentation and 3D box estimation can be more accurate.

Quantitative Comparison of the 2D Instance Mask and Bounding Box Proposal

To compare our instance mask proposals with the bounding box proposals used in the baseline, we compute the minimum bounding rectangle (MBR) of our 2D mask proposals. We report the average precision (in %) of car category on val set of KITTI [1] 2D object detection benchmark as AP_{2D} in Table 1. IoU thresholds of 0.5 and 0.7 are used.

Unsurprisingly, we find that the MBR of our mask pro-



(a) Generated point cloud frustum (**blue**)

(b) Segmented point cloud (**blue**)

(c) 3D bounding box estimation (**blue**) and ground truth (**red**)

Figure 1: **Justification of Using Instance Mask Proposal.** We visualize the generated point cloud frustum (top row), segmented point cloud (middle row) and the 3D box prediction (bottom row) from the bounding box and instance mask proposal respectively. We show that the frustum from the instance mask with no tail makes the 3D point cloud segmentation easier and results in a better 3D box estimate.

Table 1: 2D proposal evaluation. AP_{2D} performance on KITTI **val** set for car category at IoU = 0.5 / 0.7.

Proposal Type	AP_{2D} (in %), **IoU = 0.5 / 0.7**		
	Easy	Moderate	Hard
Bounding Box	97.2 / 96.5	97.3 / 90.3	90.0 / 87.6
Instance Mask	96.0 / 87.6	89.6 / 75.7	80.3 / 59.4

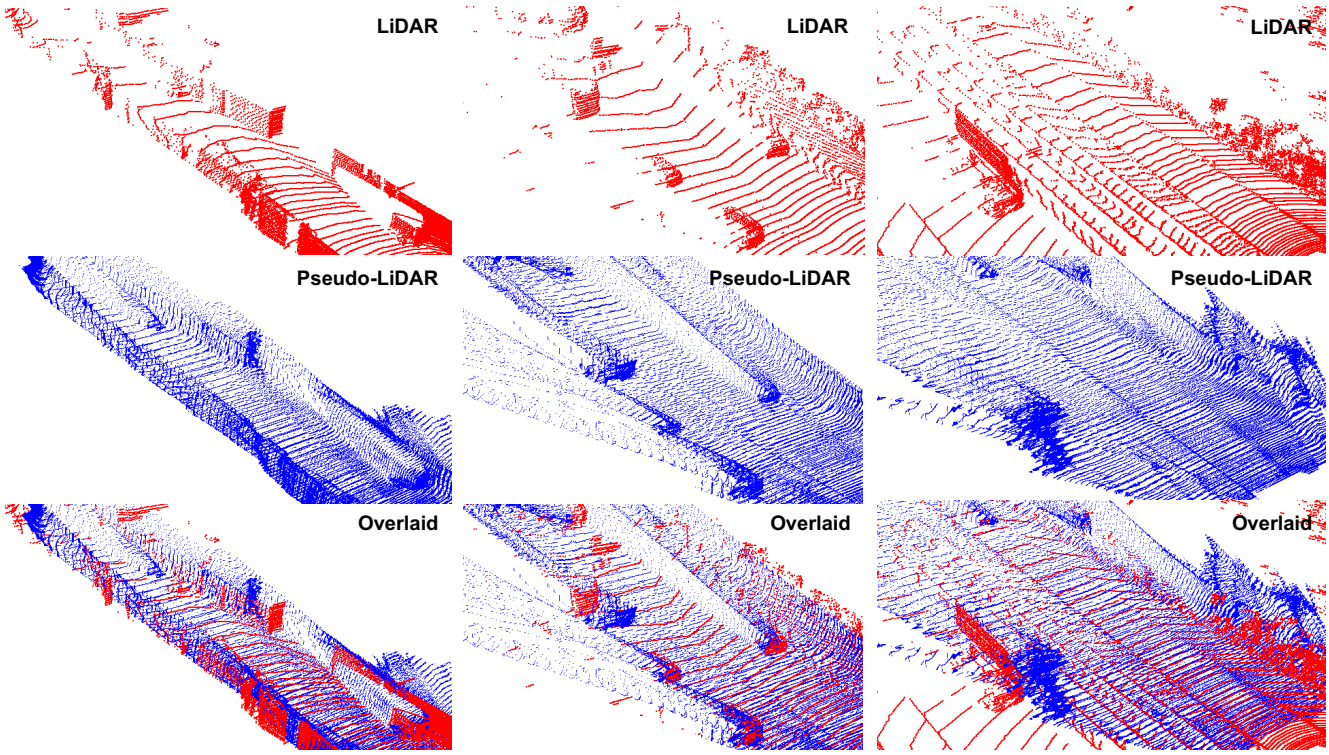


Figure 2: Additional visual comparison between the **LiDAR** (top), **pseudo-LiDAR** (middle) and an overlaid version (bottom).

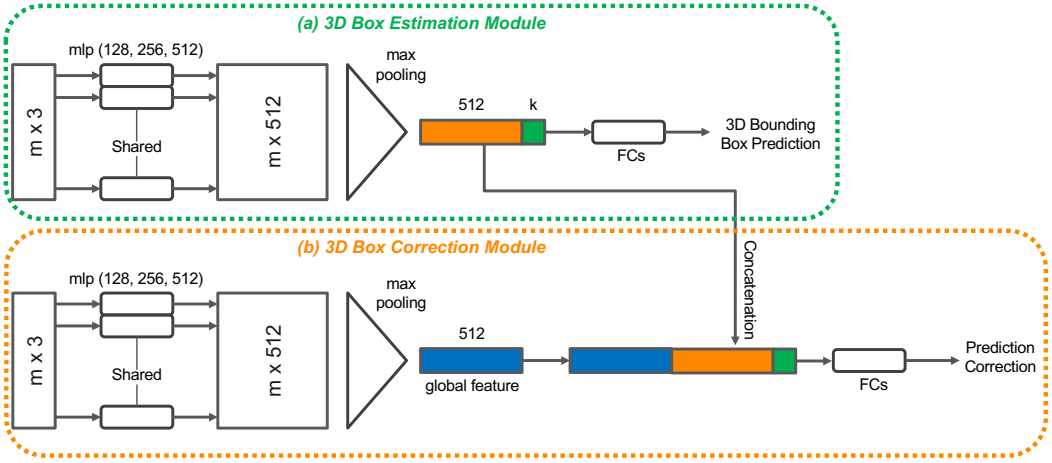


Figure 3: **Network Architecture of the 3D Box Estimation and 3D Box Correction Module.** Both modules take the point cloud as the input. The 3D box estimation module outputs the full 3D box parameters and the 3D box correction module outputs the residual of the parameters. k is the number of class (*e.g.*, 3 in KITTI). The length-k vector (in **green**) is a one-hot vector denotes which class the input point cloud belongs to. mlp denotes the multi-layer perceptron.

posal performs worse than the 2D bounding box proposal due to the lack of the pixel-level instance segmentation annotation on KITTI (only 200 images annotated with instance masks compared to 7500 images annotated with bounding boxes). However, the performance of the bird eye view and 3D object detection when using these 2D mask proposals is surprisingly higher than when using 2D bounding box proposals, which is shown in Ours (baseline) and Ours+Mask of Table 3 in the main paper. This further strengthens the effectiveness of using the instance mask proposal for 3D box estimation, *i.e.* detecting the 3D bounding box from the frustum with no tail is much easier.

3. Amodal 3D Object Detection

Network Architecture of the 3D Box Correction Module

We show the network architecture in Figure 3. Similar to the 3D box estimation module proposed in [2], we also use a PointNet-based network for our 3D box correction module.

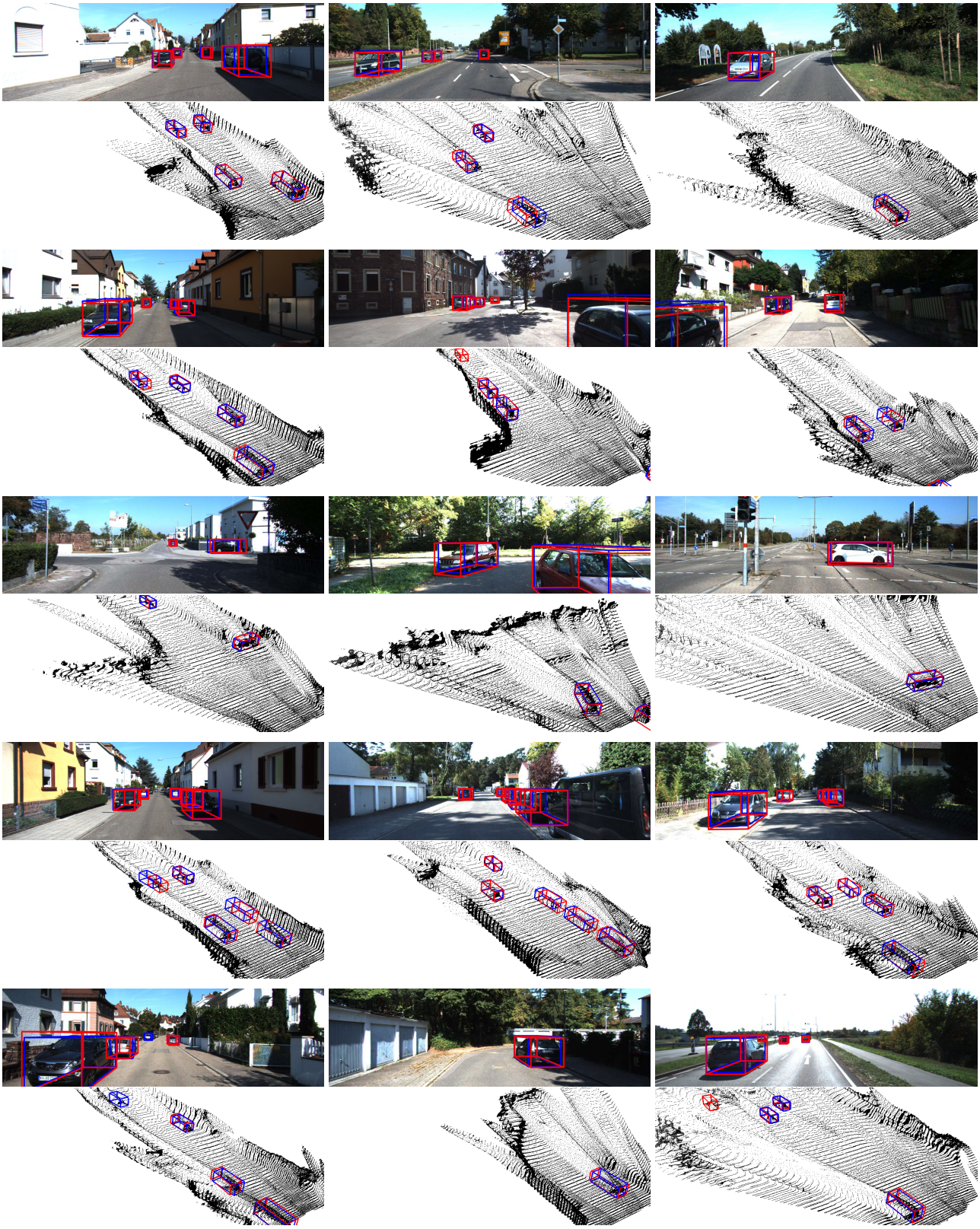


Figure 4: Additional qualitative results of our method on KITTI **val** set. We visualize our 3D bounding box estimate (in **blue**) and ground truth (in **red**) on the frontal images (1st and 3rd rows) and pseudo-LiDAR point cloud (2nd and 4th rows).

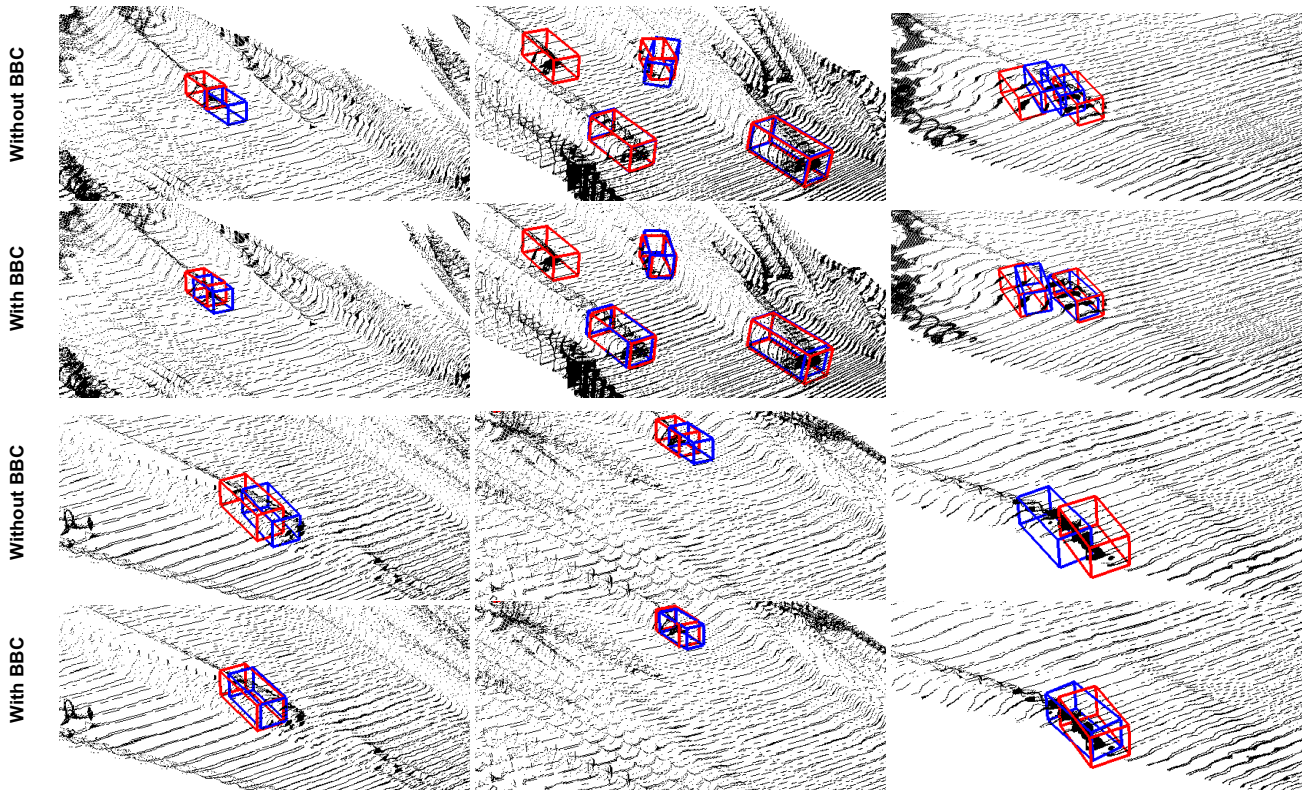


Figure 5: **Additional Visualization about the Effect of Bounding Box Consistency (BBC)**. We visualize our 3D bounding box estimate (**blue**) without BBC (in 1st and 3rd rows) and with BBC (in 2nd and 4th rows). Ground truth is shown in **red**. We show that using the bounding box consistency improves the 3D IoU between the 3D box estimate and the ground truth.

The major difference is that the 3D box estimation module predicts the 3D box parameters while our 3D box correction module outputs the correction to the prediction (*i.e.*, residual of the parameters). In addition, we concatenate the features extracted from the 3D box estimation module with the global feature extracted from the 3D box correction module for predicting the residual of the parameters.

4. 2D-3D Bounding Box Consistency (BBC)

Additional Visualization about the Effect of Bounding Box Consistency

We provide the extra visual comparison between the 3D bounding box estimate with and without using the BBC in Figure 5. The 3D bounding box results shown in the 2nd and 4th rows, which are estimated from the model trained with bounding box consistency loss and post-processed with bounding box consistency optimization, clearly improve the 3D IoU over the 3D bounding box results without using the BBC, shown in the 1st and 3rd rows.

5. Experiments

Additional Visualization of 3D Object Detection Results

We provide additional qualitative results in Figure 4. We show that, from only a single RGB image, the 3D bounding box detection for the car category can be very accurate, even for the challenging faraway objects (*e.g.*, in the 6th row 1st column and 8th row 2nd column of the Figure 4).

References

[1] A. Geiger, P. Lenz, and R. Urtasun. Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite. *CVPR*, 2012. 1

[2] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. *CVPR*, 2018. 2