Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving

Yurong You*, Yan Wang*, Wei-Lun Chao*, Divyansh Garg, Geoff Pleiss,
Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger

Cornell University, Ithaca, NY
{yy785, yw763, wc635, dg595, gp346, bh497, mc288, kqw4}@cornell.edu

Abstract

Detecting objects such as cars and pedestrians in 3D plays an indispensable role in autonomous driving. Existing approaches largely rely on expensive LiDAR sensors for accurate depth information. While recently pseudo-LiDAR has been introduced as a promising alternative, at a much lower cost based solely on stereo images, there is still a notable performance gap. In this paper we provide substantial advances to the pseudo-LiDAR framework through improvements in stereo depth estimation. Concretely, we adapt the stereo network architecture and loss function to be more aligned with accurate depth estimation of far away objects (currently the primary weakness of pseudo-LiDAR). Further, we explore the idea to leverage cheaper but extremely sparse LiDAR sensors, which alone provide insufficient information for 3D detection, to de-bias our depth estimation. We propose a depth-propagation algorithm, guided by the initial depth estimates, to diffuse these few exact measurements across the entire depth map. We show on the KITTI object detection benchmark that our combined approach yields substantial improvements in depth estimation and stereo-based 3D object detection — outperforming the previous state-of-the-art detection accuracy for far-away objects by 40%. Our code will be publicly available at https://github.com/mileyan/Pseudo_Lidar_V2.

1. Introduction

Safe driving in autonomous cars requires the detection and accurate 3D localization of cars, pedestrians and other objects. This in turn requires accurate depth information, which can be obtained from LiDAR (Light Detection And Ranging) sensors. Although highly precise and reliable, LiDAR sensors are notoriously expensive: a 64-beam model can cost around $75,000 (USD).[1] The alternative is to obtain depth information through inexpensive commodity cameras. However, in spite of recent dramatic progress in stereo-based 3D object detection brought by pseudo-LiDAR [36], a significant performance gap remains especially for far away objects (which we want to detect early to allow time for reaction). The trade-off between affordability and safety creates an ethical dilemma.

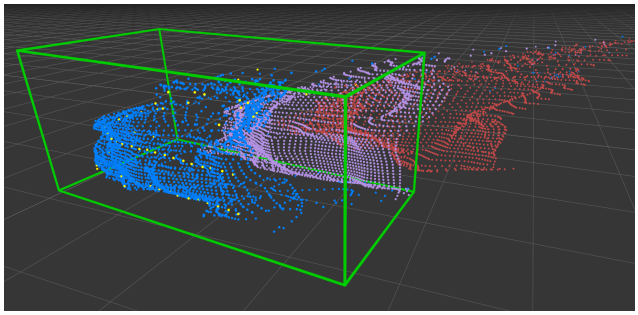


Figure 1: **An illustration of our proposed depth estimation and correction method.** The green box is the ground truth location of the car in the KITTI data set. The red points are obtained with a stereo disparity network. Purple points, obtained with our stereo depth network (SDN), are much closer to the truth. After depth propagation (blue points) with a few (yellow) LiDAR measurements the car is squarely inside the green box. (One floor square is 1m×1m.)

In this paper we propose a possible solution to this remaining challenge that combines insights from both perspectives. We observe that the higher 3D object localization error of stereo-based systems stems entirely from the higher error in depth estimation (after the 3D point cloud is obtained the two approaches are identical [36]). Importantly, this error is not

* Equal contributions

[1] The information is obtained from the automotive LiDAR market report: http://www.woodsidecap.com/wp-content/uploads/2018/04/Yole_WCP-LiDAR-Report_April-2018-FINAL.pdf

random but *systematic*: we observe that stereo methods do indeed *detect* objects with high reliability, yet they estimate the depth of the *entire* object as either too far or too close. See Figure 1 for an illustration: the red stereo points capture the car but are shifted by about 2m completely outside the ground-truth location (green box). If we can de-bias these depth estimates it should be possible to obtain accurate 3D localization even for distant objects without exorbitant costs.

We start by revisiting the depth estimation routine embedded at the heart of state-of-the-art stereo-based 3D detection approaches [36]. A major contributor to the systematic depth bias comes from the fact that depth is typically not computed directly. Instead, one first estimates the *disparity* — the horizontal shift of a pixel between the left and right images — and then *inverts* it to obtain pixel-wise depth. While the use of deep neural networks has largely improved disparity estimation [2, 7, 24, 37], designing and learning the networks to optimize the accuracy of *disparity estimation* simply over-emphasizes nearby objects due to the reciprocal transformation. For instance, a unit disparity error (in pixels) for a 5-meter-away object means a 10cm error in depth: the length of a side mirror. The same disparity error for a 50-meter-away object, however, becomes a 5.8m error in depth: the length of an entire car. Penalizing both errors equally means that the network spends more time correcting subtle errors on nearby objects than gross errors on far away objects, resulting in degraded depth estimates and ultimately poor detection and localization for far away objects. We thus propose to adapt the stereo network architecture and loss function for direct depth estimation. Concretely, the cost volume that fuses the left-right images and the subsequent 3D convolutions are the key components in stereo networks. Taking the central assumption of convolutions — all neighborhood can be operated in an identical manner — we propose to construct the cost volume on the grid of depth rather than disparity, enabling 3D convolutions and the loss function to perform exactly on the right scale for depth estimation. We refer to our network as stereo depth network (SDN). See Figure 1 for a comparison of 3D points obtained with SDN (purple) and disparity estimation (red).

Although our SDN improves the depth estimates significantly, stereo images are still inherently 2D and it is unclear if they can ever match the accuracy and reliability of a true 3D LiDAR sensor. Although LiDAR sensors with 32 or 64 beams are expensive, LiDAR sensors with only 4 beams are two orders of magnitude cheaper[2] and thus easily affordable. The 4 laser beams are very sparse and ill-suited to capture 3D object shapes by themselves, but if paired with stereo images they become the ideal tool to de-bias our dense stereo depth estimates: a single high-precision laser beam may inform us how to correct the depth of an entire car or pedestrian in its path. To this end, we present a novel depth-propagation algorithm, inspired by graph-based manifold learning [38, 33, 41]. In a nutshell, we connect our estimated 3D stereo point cloud locally by a nearest neighbor graph, such that points corresponding to the same object will share many local paths with each other. We match the few but exact LiDAR measurements first with pixels (independent of depth) and then with their corresponding 3D points to obtain accurate depth estimates for several nodes in the graph. Finally, we propagate this exact depth information along the graph using a label diffusion mechanism — resulting in a *dense and accurate depth map* at *negligible cost*. In Figure 1 we see that the few (yellow) LiDAR measurements are sufficient to position almost all final (blue) points of the entire car within the green ground truth box.

We conduct extensive empirical studies of our approaches on the KITTI object detection benchmark [12, 11] and achieve remarkable results. With solely stereo images, we outperform the previous sate-of-the-art [36] by 10%. Further adding a cheap 4-beam LiDAR brings another 27% relative improvement — on some metrics, our approach is nearly on par with those based on a 64-beam LiDAR but can potentially save 95% in cost.

2. Background

3D object detection. Most work on 3D object detection operates on 3D point clouds from LiDAR as input [18, 20, 26, 46, 8, 35, 9, 45, 17]. Frustum PointNet [29] applies PointNet [30, 31] to the points directly, while Voxelnet [48] quantizes them into 3D grids. For street scenes, several work finds that processing points from the bird's-eye view can already capture object contours and locations [6, 47, 16]. Images have also been used, but mainly to supplement LiDAR [25, 43, 22, 6, 16]. Early work based solely on images — mostly built on the 2D frontal-view detection pipeline [32, 14, 23] — fell far behind in localizing objects in 3D [19, 39, 40, 1, 27, 4, 42, 3, 28, 5].

Pseudo-LiDAR. This gap has been reduced significantly recently with the introduction of the pseudo-LiDAR framework proposed in [36]. This framework applies a drastically different approach from previous image-based 3D object detectors. Instead of directly detecting the 3D bounding boxes from the frontal view of a scene, pseudo-LiDAR begins with image-based depth estimation, predicting the depth $Z(u, v)$ of each image pixel (u, v). The resulting depth map Z is then back-projected into a 3D point cloud: a pixel (u, v) will

[2]The Ibeo Wide Angle Scanning (ScaLa) sensor with 4 beams cost $600 (USD). In this paper we simulate the 4-beam LiDAR response on KITTI benchmark [12, 11] by sparsifying the original 64-beam signal.

be transformed to (x, y, z) in 3D by

$$\begin{aligned} &\text{depth: } z = Z(u, v), \\ &\text{width: } x = \frac{(u - c_U) \times z}{f_U}, \\ &\text{height: } y = \frac{(v - c_V) \times z}{f_V}, \end{aligned} \tag{1}$$

where (c_U, c_V) is the camera center and f_U and f_V are the horizontal and vertical focal length. The 3D point cloud is then treated exactly as LiDAR signal — any LiDAR-based 3D detector can be applied seamlessly. By taking the state-of-the-art algorithms from both ends [2, 16, 29], pseudo-LiDAR obtains the highest image-based performance on the KITTI object detection benchmark [12, 11]. Our work builds upon this framework.

Stereo disparity estimation. Pseudo-LiDAR relies heavily on the quality of depth estimation. Essentially, if the estimated pixel depths match those provided by LiDAR, pseudo-LiDAR with any LiDAR-based detector should be able to achieve the same performance as that obtained by applying the same detector to the LiDAR signal. According to [36], depth estimation from stereo pairs of images [24, 44, 2] are more accurate than that from monocular (*i.e.*, single) images [10, 13] for 3D object detection. We therefore focus on stereo depth estimation, which is routinely obtained from estimating disparity between images.

A disparity estimation algorithm takes a pair of left-right images I_l and I_r as input, captured from a pair of cameras with a horizontal offset (i.e., baseline) b. Without loss of generality, we assume that the algorithm treats the left image, I_l, as reference and outputs a disparity map D recording the horizontal disparity to I_r for each pixel (u, v). Ideally, $I_l(u, v)$ and $I_r(u, v + D(u, v))$ will picture the same 3D location. We can therefore derive the depth map Z via the following transform (f_U: horizontal focal length),

$$Z(u, v) = \frac{f_U \times b}{D(u, v)}. \tag{2}$$

A common pipeline of disparity estimation is to first construct a 4D disparity cost volume C_{disp}, in which $C_{\text{disp}}(u, v, d, :)$ is a feature vector that captures the pixel difference between $I_l(u, v)$ and $I_r(u, v + d)$. It then estimates the disparity $D(u, v)$ for each pixel (u, v) according to the cost volume C_{disp}. One basic algorithm is to build a 3D cost volume with $C_{\text{disp}}(u, v, d) = \|I_l(u, v) - I_r(u, v + d)\|_2$ and determine $D(u, v)$ by $\arg\min_d C_{\text{disp}}(u, v, d)$. Advanced algorithms exploit more robust features in constructing C_{disp} and perform structured prediction for D. In what follows, we give a concise introduction on PSMNet [2], a state-of-the-art algorithm used in [36].

PSMNet begins with extracting deep feature maps h_l and h_r from I_l and I_r, respectively. It then constructs

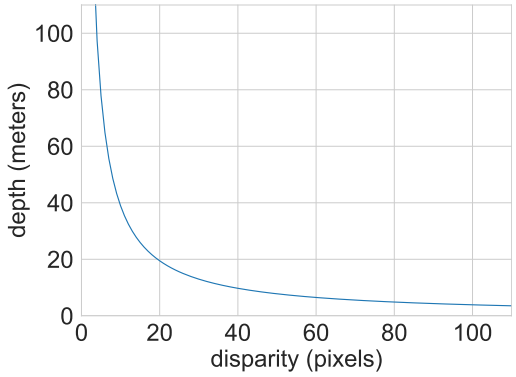


Figure 2: **The disparity-to-depth transform.** We set $f_U = 721$ (in pixels) and $b = 0.54$ (in meters) in Equation 2, which are the typical values used in the KITTI object detection benchmark.

$C_{\text{disp}}(u, v, d, :)$ by concatenating features of $h_l(u, v)$ and $h_r(u, v + d)$, followed by layers of 3D convolutions. The resulting 3D tensor S_{disp}, with the feature channel size ending up being one, is then used to derive the pixel disparity via the following weighted combination,

$$D(u, v) = \sum_d \text{softmax}(S_{\text{disp}}(u, v, d)) \times d, \tag{3}$$

where softmax is performed along the 3rd dimension of S_{disp}. PSMNet can be learned end-to-end, including the image feature extractor and 3D convolution kernels, to minimize the disparity error

$$\sum_{(u,v)\in\mathcal{A}} \ell(D(u, v) - D^\star(u, v)), \tag{4}$$

where ℓ is the smooth L1 loss, $D^\star$ is the ground truth map, and $\mathcal{A}$ contains pixels with ground truth.

3. Stereo Depth Network (SDN)

A stereo network designed and learned to minimize the disparity error (cf. Equation 4) may over-emphasize nearby objects with smaller depths and therefore perform poorly in estimating depths for far away objects. To see this, note that Equation 2 implies that for a given error in disparity δD, the error in depth, δZ, increases *quadratically* with depth:

$$Z \propto \frac{1}{D} \Rightarrow \delta Z \propto \frac{1}{D^2}\delta D \Rightarrow \delta Z \propto Z^2 \delta D, \tag{5}$$

and the middle term is obtained by differentiating $Z(D)$ w.r.t. D. In particular, using the settings on the KITTI dataset, a single pixel error in disparity implies only a 0.1m error in depth at a depth of 5 meters, but a 5.8m error at a depth of 50 meters. See Figure 2 for a mapping from disparity to depth.

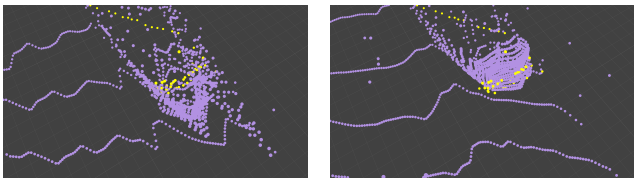


Figure 3: **Disparity cost volume (left) vs. depth cost volume (right).** The figure shows the 3D points obtained from LiDAR (yellow) and stereo (pruple) corresponding to a car in the KITTI dataset, seen from the bird's-eye view (BEV). Points from the disparity cost volume are stretched out and noisy; while points from the depth cost volume capture the car contour faithfully.

Depth Loss. We propose two essential changes to adapt a stereo network for direct depth estimation. First, we learn the stereo network to directly optimize the depth loss

$$\sum_{(u,v)\in\mathcal{A}} \ell(Z(u,v) - Z^{\star}(u,v)). \tag{6}$$

Z and $Z^{\star}$ can be obtained from D and $D^{\star}$ respectively using Equation 2. The change from the disparity to the depth loss corrects the disproportionally strong emphasis on tiny depth errors of nearby objects — a necessary but still insufficient change to overcome the problems of disparity estimation.

Depth Cost Volume. To facilitate accurate depth learning (rather than disparity) we need to address the internals of the depth estimation pipeline. A crucial source of error are the 3D convolutions within the 4D disparity cost volume, where the same convolutional kernels are applied for the entire cost volume. This is highly problematic as it implicitly assumes that the effect of a convolution is homogeneous throughout — which is clearly violated by the reciprocal depth to disparity relation (Figure 2). For example, it may be completely appropriate to locally smooth two neighboring pixels with disparity 85.7 and 86.3 (changing the depth by a few cm to smooth out a surface), whereas applying the same kernel for two pixels with disparity 3.7 and 4.3 could easily move the 3D points by 10m or more.

Taking this insight and the central assumption of convolutions — all neighborhoods can be operated upon in an identical manner — into account, we propose to instead construct depth cost volume C_{depth}, in which $C_{\text{depth}}(u,v,z,:)$ will encode features describing how likely the depth $Z(u,v)$ of pixel (u,v) is z. The subsequent 3D convolutions will then operate on the grid of depth, rather than disparity, affecting neighboring depths identically, independent of their location. The resulting 3D tensor S_{depth} is then used to predict the pixel depth similar to Equation 3

$$Z(u,v) = \sum_{z} \text{softmax}(S_{\text{depth}}(u,v,z)) \times z.$$

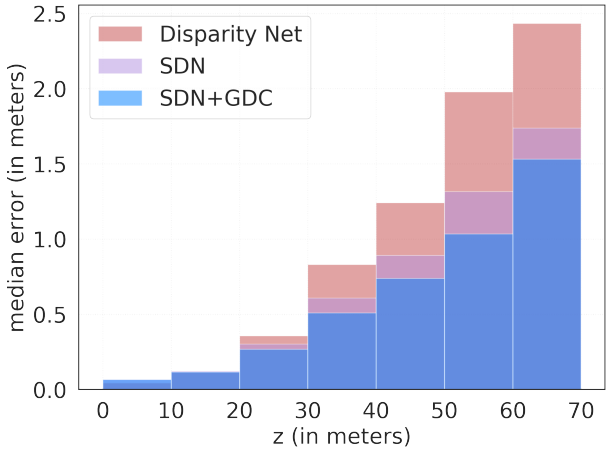


Figure 4: **Depth estimation errors.** We compare depth estimation error on 3,769 validation images of the KITTI data set, taking LiDAR depths as ground truths. We separate pixels according to their truth depths (z).

We construct the new depth volume, C_{depth}, based on the intuition that $C_{\text{depth}}(u,v,z,:)$ and $C_{\text{disp}}\left(u,v,\frac{f_U \times b}{z},:\right)$ should lead to equivalent "cost". To this end, we apply a bilinear interpolation to construct C_{depth} from C_{disp} using the depth-to-disparity transform in Equation 2. Figure 5 (top) depicts our stereo depth network (SDN) pipeline. Crucially, all convolution operations are operated on C_{depth} exclusively. Figure 4 compares the median values of absolute depth estimation errors using disparity cost volume (disparity net: *i.e.*, PSMNet) and the depth cost volume (SDN). As expected, for far away depth, SDN leads to drastically smaller errors with only marginal increases in the very near range (which disparity based methods over-optimize).

4. Depth Correction

Our SDN significantly improves depth estimation and more precisely renders the object contours. However, there is a fundamental limitation in stereo because of the discrete nature of pixels: the disparity, being the difference in the horizontal coordinate between corresponding pixels, has to be *quantized* at the level of individual pixels while the depth is *continuous*. Although the quantization error can be alleviated with higher resolution images, the computational depth prediction cost scales *cubic* with pixel width and height — pushing the limits of GPUs in autonomous vehicles.

We therefore explore a hybrid approach by leveraging a cheap LiDAR with extremely sparse (*e.g.*, 4 beams) but accurate depth measurements to *correct* this bias. We note that such sensors are too *sparse* to capture object shapes and cannot be used alone for detection. However, by projecting the LiDAR points into the image plane we obtain exact

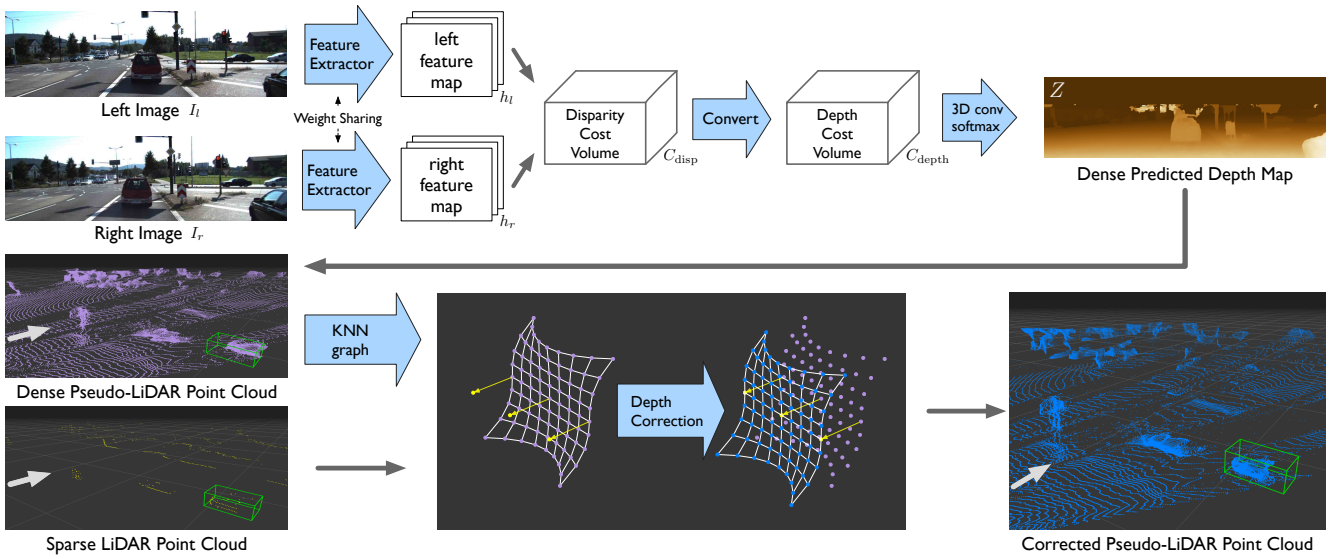


Figure 5: **The whole pipeline of improved stereo depth estimation:** (top) the stereo depth network (SDN) constructs a depth cost volume from left-right images and is optimized for direct depth estimation; (bottom) the graph-based depth correction algorithm (GDC) refines the depth map by leveraging sparser LiDAR signal. The gray arrows indicates the observer's view point. We superimpose the (green) ground-truth 3D box of a car, the same one in Figure 1. The corrected points (blue; bottom right) are perfectly located inside the ground truth box.

depths on a small portion of "landmark" pixels.

We present a graph-based depth correction (GDC) algorithm that effectively combines the *dense* stereo depth that has rendered object shapes and the *sparse* accurate LiDAR measurements. Conceptually, we expect the corrected depth map to have the following properties: globally, landmark pixels associated with LiDAR points should possess the exact depths; locally, object shapes captured by neighboring 3D points, back-projected from the input depth map (cf. Equation 1), should be preserved. Figure 5 (bottom) illustrates the algorithm.

Input Matching. We take as input the two point clouds from LiDAR and Pseudo-LiDAR (PL) by stereo depth estimation. Latter is obtained by converting pixels (u, v) with depth z to 3D points (x_u, y_v, z). First, we characterize the local shapes by the directed K-nearest-neighbor (KNN) graph in the PL point cloud that connects each point to its KNN neighbors with appropriate weights (using accelerated KD-Trees [34]). Similarly, we can project the 3D LiDAR points onto pixel locations (u, v) and match them to corresponding 3D stereo points. W.l.o.g. assume that we are given "ground truth" LiDAR depth for the first n points and no ground truth for the remaining m points. We refer to the 3D stereo depth estimates as $Z \in \mathbb{R}^{n+m}$ and the LiDAR depths as $G \in \mathbb{R}^n$.

Edge weights. To construct the KNN graph in 3D we ignore the LiDAR information on the first n points and only use their predicted stereo depth in Z. Let $\mathcal{N}_i$ denote the set of neighbors of the i^{th} point. Further, let $W \in \mathbb{R}^{(n+m)\times(n+m)}$ denote the weight matrix, where W_{ij} denotes the edge-weight between points i and j. Inspired by prior work in manifold learning [33, 38] we choose the weights to be the coefficients that reconstructs the depth of any point from the depths of its neighbors in $\mathcal{N}_i$. We can solve for these weights with the following constrained quadratic optimization problem:

$$W = \arg\min_W \|Z - WZ\|_2^2,$$
$$\text{s.t. } W\mathbf{1} = \mathbf{1} \text{ and } W_{ij} = 0 \text{ if } j \notin \mathcal{N}_i. \quad (7)$$

Here $\mathbf{1} \in \mathbb{R}^{n+m}$ denotes the all-ones vector. As long as we pick $k > 3$ and the points are in general positions there are infinitely many solutions that satisfy $Z = WZ$, and we pick the minimum L_2 norm solution (obtained with slight L_2 regularization) for robustness reasons.

Depth Correction. Let us denote the corrected depth values as $Z' \in \mathbb{R}^{n+m}$, with $Z' = [Z'_L; Z'_{PL}]$ and $Z'_L \in \mathbb{R}^n$ and $Z'_{PL} \in \mathbb{R}^m$. For the n points with LiDAR measurements we update the depth to the (ground truth) values $Z'_L = G$. We then solve for Z'_{PL} given G and the weighted KNN graph encoded in W. Concretely, we update the remaining depths Z'_{PL} such that the depth of any point i can still be be reconstructed with high fidelity as a weighted sum of its KNN neighbors' depths using the learned weights W; *i.e.* if point $i : 1 \le i \le n$ is moved to its new depth G_i, then its neighbors in $\mathcal{N}_i$ must also be corrected such that $G_i \approx \sum_{j\in\mathcal{N}_i} W_{ij} Z'_j$. Further, the neighbors' neighbors must be corrected and the depth of the few n points propagates across

the entire graph. We can solve for the final Z' directly with another quadratic optimization,

$$Z' = \arg\min_{Z'} \|Z'W - Z'\|^2, \text{ s.t. } Z'_{1:n} = G. \quad (8)$$

To illustrate the correction process, imagine the simplest case where the depth of only a single point ($n = 1$) is updated to $G_1 = Z_1 + \delta$. A new optimal depth for Equation 8 is to move all the remaining points similarly, i.e. $Z' = Z + \mathbf{1}\delta$: as $Z = WZ$ and $W\mathbf{1} = \mathbf{1}$ we must have $W(Z+\mathbf{1}\delta) = Z+\mathbf{1}\delta$. In the setting with $n > 1$, the least-squares loss ensures a soft diffusion between the different LiDAR depth estimates. Both optimization problems in Equation 7 and Equation 8 can be solved exactly and efficiently with sparse matrix solvers. We summarize the procedure as an algorithm in the supplemental material. From the view of graph-based manifold learning, our GDC algorithm is reminiscent of the locally linear embedding [33] with landmarks to guide the final solution [38]. Figure 1 illustrates beautifully how the initial 3D point cloud from SDN (purple) of a car in the KITTI data set is corrected with a few sparse LiDAR measurements (yellow). The resulting points (blue) are right inside the ground-truth box and clearly show the contour of the car. Figure 4 shows the additional improvement from the GDC (blue) over the pure SDN depth estimates. The error is corrected over the entire image where many regions have no LiDAR measurements. For objects such as cars the improvements through GDC are far more pronounced, as these typically are touched by the four LiDAR beams and can be corrected effectively.

5. Experiments

5.1. Setup

We refer to our combined method (SDN and GDC) for 3D object detection as PSEUDO-LIDAR++ (PL++ in short). To analyze the contribution of each component, we evaluate SDN and GDC independently and jointly across several settings. For GDC we set $k = 10$ and consider adding signal from a (simulated) 4-beam LiDAR, unless stated otherwise.

Dataset, Metrics, and Baselines. We evaluate on the KITTI dataset [11, 12], which contains 7,481 and 7,518 images for training and testing. We follow [4] to separate the 7,481 images into 3,712 for training and 3,769 validation. For each (left) image, KITTI provides the corresponding right image, the 64-beam Velodyne LiDAR point cloud, and the camera calibration matrices. We focus on 3D object detection and bird's-eye-view (BEV) localization and report results on the *validation set*. Specifically, we focus on the "car" category, following [6, 43]. We report average precision (AP) with IoU (Intersection over Union) thresholds at 0.5 and 0.7. We denote AP for the 3D and BEV tasks by AP_{3D} and AP_{BEV}. KITTI divides each category into easy, moderate, and hard cases, according to the 2D box height and occlusion/truncation level. We compare to four stereo-based detectors: PSEUDO-LIDAR (PL in short) [36], 3DOP [4], S-RCNN [21], and MLF-STEREO [42].

Stereo depth network (SDN). We use PSMNET [2] as the backbone for our stereo depth estimation network (SDN). We follow [36] to pre-train SDN on the synthetic Scene Flow dataset [24] and fine-tune it on the 3,712 training images of KITTI. We obtain depth ground truth of these images by projecting the corresponding LiDAR points onto images. We also train a PSMNET in the same way for comparison, which minimizes disparity error.

3D object detection. We apply three algorithms for 3D object detection: AVOD [16], PIXOR [47], and P-RCNN [35]. All can utilize information from LiDAR and/or monocular images. We use the released implementations of AVOD (more specifically, AVOD-FPN) and P-RCNN. We implement PIXOR ourselves with a slight modification to include visual information (denoted as PIXOR*). We train all models on the 3,712 training data from scratch by replacing the LiDAR points with pseudo-LiDAR data generated from stereo depth estimation (see the supplemental material for details.)

Sparser LiDAR. We simulate sparser LiDAR signal with fewer beams by first projecting the 64-beam LiDAR points onto a 2D plane of horizontal and vertical angles. We quantize the vertical angles into 64 levels with an interval of $0.4°$, which is close to the SPEC of the 64-beam LiDAR. We keep points fallen into a subset of beams to mimic the sparser signal (see the supplemental material for details.)

5.2. Experimental results

We summarize the main results on KITTI object detection in Table 1. Several important trends can be observed: **1)** Our PL++ with enhanced depth estimations by SDN and GDC yields consistent improvement over PL across all settings; **2)** PL++ with GDC refinement of 4 laser beams (INPUT: L# + S) performs significantly better than PL++ with only stereo inputs (INPUT: S); **3)** PL experiences a substantial drop in accuracy from IoU 0.5 to 0.7 for *hard* objects. This indicates that PL does indeed manage to detect objects that are far away, but systematically places them at the wrong depth. Once an overlap of 0.7 is required (IoU = 0.7), the object is too far out of place and is no longer registered as detect. Interestingly, here is where we experience the largest gain — from PL: P-RCNN ($AP_{BEV} = 52.7$) to PL++: P-RCNN ($AP_{BEV} = 73.4$) with input as L# + S. Note that the majority of the gain originates from GDC, as PL++ with solely stereo input only improves the score to 57.3 AP_{BEV}. **4)** Compared to LiDAR, PL++ is only outperformed by at most 13% AP_{BEV}, even at the hard case under IoU at 0.7. **5)** For IoU at 0.5, with the aid of only 4 LiDAR beams, PL++ is boosted to a level comparable to models with 64-beam LiDAR signals.

Table 1: **3D object detection results on KITTI validation.** We report AP_{BEV} / AP_{3D} (in %) of the **car** category, corresponding to average precision of the bird's-eye view and 3D object detection. We arrange methods according to the input signals: M for monocular images, S for stereo images, L for 64-beam LiDAR, and L# for *sparse 4-beam* LiDAR. PL stands for PSEUDO-LIDAR. *Our* PSEUDO-LIDAR ++ *(PL++) with enhanced depth estimation* — SDN *and* GDC— *are in blue.* Methods with 64-beam LiDAR are in gray. Best viewed in color.

Detection algorithm	Input	IoU = 0.5			IoU = 0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard
3DOP [4]	S	55.0 / 46.0	41.3 / 34.6	34.6 / 30.1	12.6 / 6.6	9.5 / 5.1	7.6 / 4.1
MLF-STEREO [42]	S	-	53.7 / 47.4	-	-	19.5 / 9.8	-
S-RCNN [21]	S	87.1 / 85.8	74.1 / 66.3	58.9 / 57.2	68.5 / 54.1	48.3 / 36.7	41.5 / 31.1
PL: AVOD [36]	S	89.0 / 88.5	77.5 / 76.4	68.7 / 61.2	74.9 / 61.9	56.8 / 45.3	49.0 / 39.0
PL: PIXOR*	S	89.0 / -	75.2 / -	67.3 / -	73.9 / -	54.0 / -	46.9 / -
PL: P-RCNN	S	88.4 / 88.0	76.6 / 73.7	69.0 / 67.8	73.4 / 62.3	56.0 / 44.9	52.7 / 41.6
PL++: AVOD	S	89.4 / 89.0	79.0 / 77.8	70.1 / 69.1	77.0 / 63.2	63.7 / 46.8	56.0 / 39.8
PL++: PIXOR*	S	89.9 / -	78.4 / -	74.7 / -	79.7 / -	61.1 / -	54.5 / -
PL++: P-RCNN	S	89.8 / 89.7	83.8 / 78.6	77.5 / 75.1	82.0 / 67.9	64.0 / 50.1	57.3 / 45.3
PL++: AVOD	L# + S	90.2 / 90.1	87.7 / 86.9	79.8 / 79.2	86.8 / 70.7	76.6 / 56.2	68.7 / 53.4
PL++: PIXOR*	L# + S	95.1 / -	85.1 / -	78.3 / -	84.0 / -	71.0 / -	65.2 / -
PL++: P-RCNN	L# + S	90.3 / 90.3	87.7 / 86.9	84.6 / 84.2	88.2 / 75.1	76.9 / 63.8	73.4 / 57.4
AVOD [16]	L + M	90.5 / 90.5	89.4 / 89.2	88.5 / 88.2	89.4 / 82.8	86.5 / 73.5	79.3 / 67.1
PIXOR* [47, 22]	L + M	94.2 / -	86.7 / -	86.1 / -	85.2 / -	81.2 / -	76.1 / -
P-RCNN [35]	L	96.3 / 96.1	88.6 / 88.5	88.6 / 88.5	87.8 / 81.7	86.0 / 74.4	85.8 / 74.5

Table 2: 3D object detection results on the **car** category on the *test* set. We compare our methods (in blue) and 64-beam LiDAR (in gray), using P-RCNN as the object detector. We report AP_{BEV} / AP_{3D} at IoU = 0.7. †: Results from the KITTI leaderboard.

Input signal	Easy	Moderate	Hard
PL++ (SDN)	75.5 / 60.4	57.2 / 44.6	53.4 / 38.5
PL++ (SDN +GDC)	83.8 / 68.5	73.5 / 54.7	66.5 / 51.2
†LiDAR	89.5 / 85.9	85.7 / 75.8	79.1 / 68.3

Table 3: **Ablation study on stereo depth estimation.** We report AP_{BEV} / AP_{3D} (in %) of the **car** category at IoU= 0.7 on KITTI validation. DL: depth loss. The best result of each column is in bold font.

Stereo depth	P-RCNN		
	Easy	Moderate	Hard
PSMNET	73.3 / 62.3	55.9 / 44.8	52.6 / 41.4
PSMNET + DL	80.1 / 65.5	61.9 / 46.8	56.0 / 43.0
SDN	**82.0 / 67.9**	**64.0 / 50.1**	**57.3 / 45.3**

Results on the secret KITTI test set. Table 2 summarizes our test results for the car category on the KITTI test set server. We see a similar gap between our methods and LiDAR as on the validation set, suggesting that our approach does not simply over-fit to the validation data. There is no category for 4-beam LiDAR, but at the time of submission, our approach without LiDAR refinement (pure SDN) is placed at the top position among all the image-based algorithms on the KITTI leaderboard.

In the following sections, we conduct a series of experiments to analyze the performance gain by our approaches and discuss several key observations. *We will mainly experiment with* P-RCNN*: we find that the results with* AVOD *and* PIXOR* *follow similar trends and include them in the supplemental material.*

Depth loss and depth cost volume. To turn a disparity network (*e.g.*, PSMNET) into SDN, there are two subsequent changes: **1)** change the disparity loss into the depth loss; **2)** change the disparity cost volume into the depth cost volume. In Table 3, we uncover the effect of these two changes separately. Regarding AP_{BEV}/AP_{3D} (moderate), the metric used in the KITTI leaderboard, the depth loss gains us $6\%/2\%$ improvement, while the depth cost volume brings another $2\%/3\%$. Essentially, we demonstrate that the two components are complementary to improve depth estimation.

Impact of sparse LiDAR beams. In PSEUDO-LIDAR ++, we leverage 4-beam LiDAR by GDC to correct stereo depth. We then ask the following question: is it possible that the gain solely comes from adding 4-beam LiDAR points? In Table 4, we study this question by comparing the detection result against that of models using **1)** sole 4-beam LiDAR point clouds and **2)** pseudo-LiDAR point clouds with corresponding parts replaced by 4-beam LiDAR. It can be seen that 4-beam LiDAR itself performs fairly well on locating

Table 4: **Ablation study on leveraging sparse LiDAR.** We report AP_{BEV} / AP_{3D} (in %) of the **car** category at IoU= 0.7 on KITTI validation. L#: 4-beam LiDAR signal alone. SDN + L#: pseudo-LiDAR with corresponding parts replaced by 4-beam LiDAR. The best result of each column is in bold font. See text for details.

Stereo depth	P-RCNN		
	Easy	Moderate	Hard
SDN	82.0 / 67.9	64.0 / 50.1	57.3 / 45.3
L#	73.2 / 56.1	71.3 / 53.1	70.5 / 51.5
SDN + L#	86.3 / 72.0	73.0 / 56.1	67.4 / 54.1
SDN + GDC	**88.2 / 75.1**	**76.9 / 63.8**	**73.4 / 57.4**

far away objects but cannot capture close objects precisely, while simply replacing pseudo-LiDAR with LiDAR prevents the model from detecting far away object accurately. In contrast, our proposed GDC method effectively combines the merits of the two signals, achieving superior performance than using them alone.

6. Conclusion

In this paper we made two contributions to improve the 3D object detection in autonomous vehicles without expensive LiDAR. First, we identify the disparity estimation as a main source of error for stereo based systems and propose a novel approach to learn depth directly end-to-end instead of through disparity estimates. Second, we advocate that one should not use expensive LiDAR sensors to learn the local structure and depth of objects. Instead one can use commodity stereo cameras for the former and a cheap sparse LiDAR to correct the systematic bias in the resulting depth-estimates. We provide a novel graph propagation algorithm that integrates the two data modalities and propagates the initial depth estimates with two sparse matrix solvers. The resulting system, Pseudo-LiDAR++, performs almost on par with 64-beams LiDAR systems for $75,000 but only requires 4 beams and two commodity cameras, which could be obtained with a total cost of less than $800.

Acknowledgments

This research is supported in part by grants from the National Science Foundation (III-1618134, III-1526012, IIS-1149882, IIS-1724282, and TRIPODS-1740822), the Office of Naval Research DOD (N00014-17-1-2175), and the Bill and Melinda Gates Foundation. We are thankful for generous support by Zillow and SAP America Inc. We thank Gao Huang for helpful discussion.

References

[1] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, 2017. 2

[2] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *CVPR*, 2018. 2, 3, 6

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

[4] 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 *NIPS*, 2015. 2, 6, 7

[5] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals using stereo imagery for accurate object class detection. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1259–1272, 2018. 2

[6] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 2, 6

[7] X. Cheng, P. Wang, and R. Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *ECCV*, 2018. 2

[8] X. Du, M. H. Ang Jr, S. Karaman, and D. Rus. A general pipeline for 3d detection of vehicles. In *ICRA*, 2018. 2

[9] 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 *ICRA*, 2017. 2

[10] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, pages 2002–2011, 2018. 3

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2, 3, 6

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 3, 6

[13] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 3

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017. 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 10

[16] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 2018. 2, 3, 6, 7

[17] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2

[18] B. Li. 3d fully convolutional network for vehicle detection in point cloud. In *IROS*, 2017. 2

[19] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *CVPR*, 2019. 2

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

[21] P. Li, X. Chen, and S. Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *CVPR*, 2019. 6, 7

[22] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018. 2, 7, 10

[23] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. 2

[24] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 2, 3, 6

[25] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez. Sensor fusion for joint 3d object detection and semantic segmentation. *arXiv preprint arXiv:1904.11466*, 2019. 2

[26] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *CVPR*, 2019. 2

[27] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 2

[28] 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*, 53:110–122, 2017. 2

[29] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018. 2, 3

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 2

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[33] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 2000. 2, 5, 6

[34] M. Shevtsov, A. Soupikov, and A. Kapustin. Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. In *Computer Graphics Forum*, volume 26, pages 395–404. Wiley Online Library, 2007. 5

[35] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 2, 6, 7

[36] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019. 1, 2, 3, 6, 7, 10

[37] Y. Wang, Z. Lai, G. Huang, B. H. Wang, L. van der Maaten, M. Campbell, and K. Q. Weinberger. Anytime stereo image depth estimation on mobile devices. *arXiv preprint arXiv:1810.11408*, 2018. 2

[38] K. Q. Weinberger, B. Packer, and L. K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *AISTATS*, 2005. 2, 5, 6

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

[40] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *WACV*, 2017. 2

[41] Z. Xiaojin and G. Zoubin. Learning from labeled and unlabeled data with label propagation. *Tech. Rep., Technical Report CMU-CALD-02–107, Carnegie Mellon University*, 2002. 2

[42] B. Xu and Z. Chen. Multi-level fusion based 3d object detection from monocular images. In *CVPR*, 2018. 2, 6, 7

[43] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *CVPR*, 2018. 2, 6

[44] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*, 2014. 3

[45] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2

[46] B. Yang, M. Liang, and R. Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155, 2018. 2

[47] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 2, 6, 7, 10

[48] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 2

Supplementary Material

We provide details omitted in the main text.

- Appendix A: detailed implementation of the GDC algorithm. (section 4 of the main paper).
- Appendix B: additional details of experimental setups (subsection 5.1 of the main paper).
- Appendix C: additional experimental results (subsection 5.2 of the main paper).

A. Graph-based Depth Correction (GDC) Algorithm

Here we present the GDC algorithm in detail (see algorithm 1). The two steps described in the main paper can be easily turned into two (sparse) linear systems and then solved by using Lagrange multipliers. For the first step, we solve a problem that is slightly modified from that described in the main paper (for more accurate reconstruction). For the second step, we use the *Generalized Minimal Residual Method* (GMRES) to iteratively solve the sparse linear system.

B. Experimental Setup

B.1. Sparse LiDAR generation

In this section, we explain how we generate sparser LiDAR with fewer beams from a 64-beam LiDAR point cloud from KITTI dataset in detail. For every point $(x_i, y_i, z_i) \in \mathbb{R}^3$ of the point cloud in one scene (in LiDAR coordinate system (x: front, y: left, z: up, and $(0, 0, 0)$ is the location of the LiDAR sensor)), we compute the elevation angle θ_i to the LiDAR sensor as

$$\theta_i = \arg\cos\left(\frac{\sqrt{x_i^2 + y_i^2}}{\sqrt{x_i^2 + y_i^2 + z_i^2}}\right).$$

We order the points by their elevation angles and slice them into separate lines by step $0.4°$, starting from $-23.6°$ (close to the Velodyne 64-beam LiDAR SPEC). We select LiDAR points whose elevation angles fall within $[-2.4°, -2.0°) \cup [-0.8°, -0.4°)$ to be the 2-beam LiDAR signal, and similarly $[-2.4°, -2.0°) \cup [-1.6°, -1.2°) \cup [-0.8°, -0.4°) \cup [0.0°, 0.4°)$ to be the 4-beam LiDAR signal. We choose them in such a way that consecutive lines has a $0.8°$ interval, following the SPEC of the "cheap" 4-beam LiDAR ScaLa. We visualize these sparsed LiDAR point clouds from the bird's-eye view on one example scene in Figure 6.

B.2. 3D object detection algorithms

In this section, we provide more details about the way we train 3D object detection models on pseudo-LiDAR point clouds. For AVOD, we use the same model as in [36]. For P-RCNN, we use the implementation provided by the authors. Since the P-RCNN model exploits the sparse nature of LiDAR point clouds, when training it with pseudo-LiDAR input, we will first sparsify the point clouds into 64 beams using the method described in subsection B.1. For PIXOR*, we implement the same base model structure and data augmentation specified in [47], but without the "decode fine-tune" step and focal loss. Inspired by the trick in [22], we add another image feature (ResNet-18 [15]) branch along the LiDAR branch, and concatenate the corresponding image features onto the LiDAR branch at each stage. We train PIXOR* using RMSProp with momentum 0.9, learning rate 10^{-5} (decay by 10 after 50 and 80 epochs) for 90 epochs. The BEV evaluation results are similar to the reported results, see Table 1.

C. Additional Results

C.1. Ablation study

In Table 5 and Table 6 we provide more experimental results aligned with experiments in subsection 5.2 of the main paper. We conduct the same experiments on two other models, AVOD and PIXOR*, and observe similar trends of improvements brought by learning with the depth loss (from PSMNET to PSMNET +DL), constructing the depth cost volume (from PSMNET +DL to SDN), and applying GDC to correct the bias in stereo depth estimation (comparing SDN +GDC with SDN).

C.2. Using fewer LiDAR beams

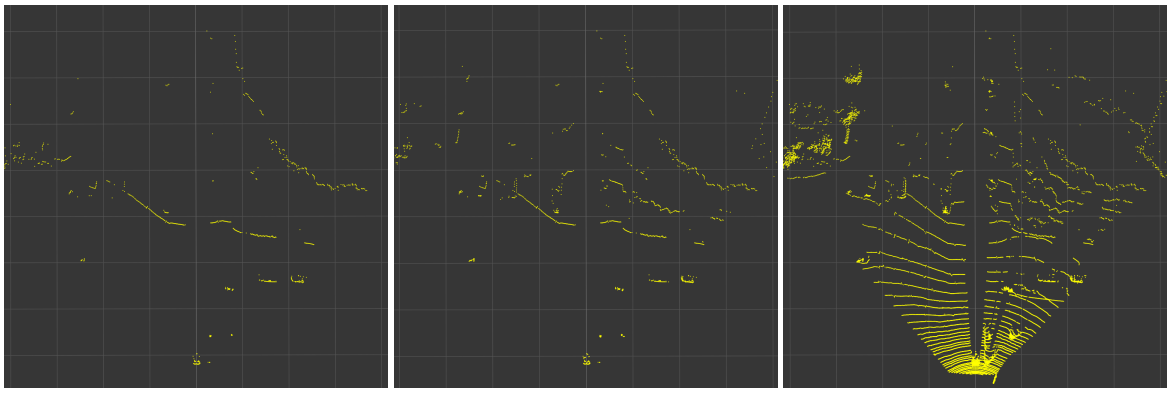
In PL++ (*i.e.*, SDN + GDC), we use 4-beam LiDAR to correct the predicted point cloud. In Table 7, we investigate using fewer (and also potentially cheaper) LiDAR beams for depth correction. We observe that even with 2 beams, GDC can already manage to combine the two signals and yield a better performance than using 2-beam LiDAR or pseudo-LiDAR alone.

C.3. Qualitative results

In Figure 7, we show detection results using P-RCNN with different input signals on a randomly chosen scene in the KITTI object validation set. Specifically, we show the results from the frontal-view images and the bird's-eye view (BEV) point clouds. In the BEV map, the observer is on the left-hand side looking to the right. For nearby objects (*i.e.*, bounding boxes close to the left in the BEV map), we see that P-RCNN with any point cloud performs fairly well in localization. However, for far away objects (*i.e.*, bounding boxes close to the right), PSEUDO-LIDAR with depth estimated from PSMNET predicts objects (green boxes) deviated from the ground truths (red boxes). Moreover, the noisy PSMNET points also leads to several false positives. In contrast, the detected boxes by our PSEUDO-LIDAR ++,

Algorithm 1: Graph-based depth correction (GDC). ";" stands for column-wise concatenation.

Input: Stereo depth map $Z \in \mathbb{R}^{(n+m)\times 1}$, the corresponding pseudo-LiDAR (PL) point cloud $P \in \mathbb{R}^{(n+m)\times 3}$, and LiDAR depths $G \in \mathbb{R}^{n\times 1}$ on the first the n pixels.
Output: Corrected depth map $Z' \in \mathbb{R}^{(n+m)\times 1}$
function GDC(Z, P, G, K)
  Solve: $W = \arg\min_{W\in\mathbb{R}^{(n+m)\times(n+m)}} \|W\|^2$
    s.t. $Z - W \cdot Z = 0$,
      $W_{ij} = 0$ if $j \notin \mathcal{N}_i$ according to P,
      $\sum_j W_{ij} = 1$ for $\forall i = 1, \ldots, n+m$.
  Solve: $Z'_{PL} = \arg\min_{Z'_{PL}\in\mathbb{R}^{m\times 1}} \|[G; Z'_{PL}] - W[G; Z'_{PL}]\|^2$
  return $[G; Z'_{PL}]$
end



(a) 2-beam (b) 4-beam (c) 64-beam (full)

Figure 6: Bird's-eye views of sparsed LiDAR point clouds on an example scene. The observer is on the bottom side looking up. We filter out points not visible from the left-image. (One floor square is 10m $\times$ 10m.)

Table 5: **Ablation study on stereo depth estimation.** We report AP_{BEV} / AP_{3D} (in %) of the **car** category at IoU$=0.7$ on the KITTI validation set. DL stands for depth loss.

Depth Estimation	PIXOR⋆			AVOD		
	Easy	Moderate	Hard	Easy	Moderate	Hard
PSMNET	73.9 / -	54.0 / -	46.9 / -	74.9 / 61.9	56.8 / 45.3	49.0 / 39.0
PSMNET + DL	75.8 / -	56.2 / -	51.9 / -	75.7 / 60.5	57.1 / 44.8	49.2 / 38.4
SDN	79.7 / -	61.1 / -	54.5 / -	77.0 / 63.2	63.7 / 46.8	56.0 / 39.8

Table 6: **Ablation study on leveraging sparse LiDAR.** We report AP_{BEV} / AP_{3D} (in %) of the **car** category at IoU$=0.7$ on the KITTI validation set. L# stands for 4-beam LiDAR signal. SDN +L# means we simply replace the depth of a portion of pseudo-LiDAR points by L#.

Depth Estimation	PIXOR⋆			AVOD		
	Easy	Moderate	Hard	Easy	Moderate	Hard
SDN	79.7 / -	61.1 / -	54.5 / -	77.0 / 63.2	63.7 / 46.8	56.0 / 39.8
L#	72.0 / -	64.7 / -	63.6 / -	77.0 / 62.1	68.8 / 54.7	67.1 / 53.0
SDN + L#	75.6 / -	59.4 / -	53.2 / -	84.1 / 66.0	67.0 / 53.1	58.8 / 46.4
SDN + GDC	84.0 / -	71.0 / -	65.2 / -	86.8 / 70.7	76.6 / 56.2	68.7 / 53.4

Table 7: **Ablation study on the sparsity of LiDAR.** We report AP_{BEV} / AP_{3D} (in %) of the **car** category at IoU$= 0.7$ on the KITTI validation set. L# stands for using sparse LiDAR signal alone. The number in brackets indicates the number of beams in use.

Depth Estimation	P-RCNN			PIXOR*		
	Easy	Moderate	Hard	Easy	Moderate	Hard
L# (2)	69.2 / 46.3	62.8 / 41.9	61.3 / 40.0	66.8 / -	55.5 / -	53.3 / -
L# (4)	73.2 / 56.1	71.3 / 53.1	70.5 / 51.5	72.0 / -	64.7 / -	63.6 / -
SDN + GDC (2)	87.2 / 73.3	72.0 / 56.6	67.1 / 54.1	82.0 / -	65.3 / -	61.7 / -
SDN + GDC (4)	88.2 / 75.1	76.9 / 63.8	73.4 / 57.4	84.0 / -	71.0 / -	65.2 / -

either with SDN alone or with SDN +GDC, aligns pretty well with the ground truth boxes, justifying our targeted improvement in estimating far away depths.

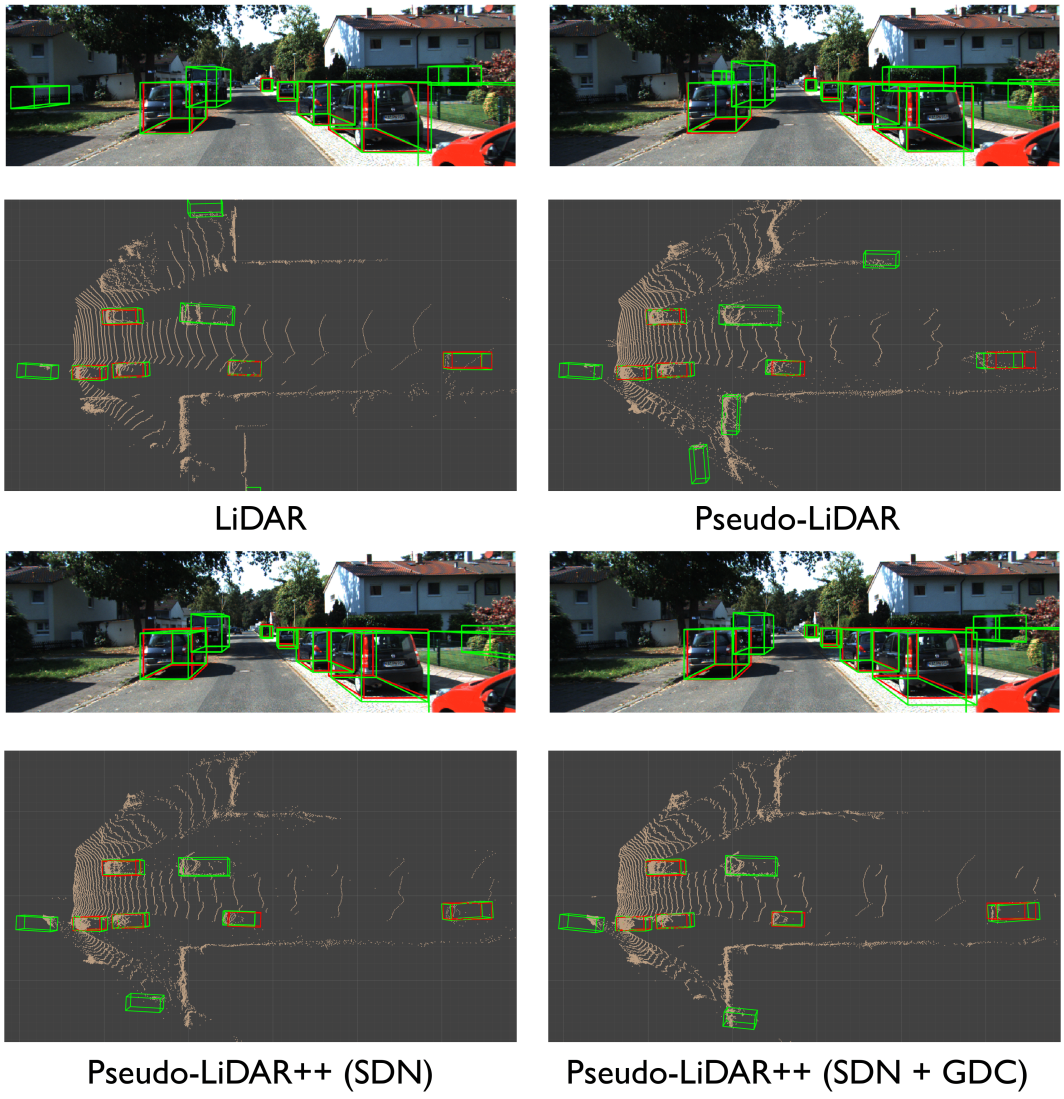


Figure 7: **Qualitative Comparison.** We show the detection results on a KITTI validation scene by P-RCNN with different input point clouds. We visualize them from both frontal-view images and bird's-eye view (BEV) point maps. Ground-truth boxes are in red and predicted bounding boxes are in green. The observer is at the left-hand side of the BEV map looking to the right. In other words, ground truth boxes on the right are more far away (*i.e.* deeper) from the observer, and hence hard to localize. Best viewed in color.