PointPainting: Sequential Fusion for 3D Object Detection

Sourabh Vora Alex H. Lang Bassam Helou Oscar Beijbom
nuTonomy: an Aptiv Company
{sourabh, alex, bassam, oscar}@nutonomy.com

Abstract

Camera and lidar are important sensor modalities for robotics in general and self-driving cars in particular. The sensors provide complementary information offering an opportunity for tight sensor-fusion. Surprisingly, lidar-only methods outperform fusion methods on the main benchmark datasets, suggesting a gap in the literature. In this work, we propose PointPainting: a sequential fusion method to fill this gap. PointPainting works by projecting lidar points into the output of an image-only semantic segmentation network and appending the class scores to each point. The appended (painted) point cloud can then be fed to any lidar-only method. Experiments show large improvements on three different state-of-the art methods, Point-RCNN, VoxelNet and PointPillars on the KITTI and nuScenes datasets. The painted version of PointRCNN represents a new state of the art on the KITTI leaderboard for the bird's-eye view detection task. In ablation, we study how the effects of Painting depends on the quality and format of the semantic segmentation output, and demonstrate how latency can be minimized through pipelining.

1. Introduction

Driven partially by the interest in self-driving vehicles, significant research effort has been devoted to 3D object detection. In this work we consider the problem of fusing a lidar point cloud with an RGB image. The point cloud provides a very accurate range view, but with low resolution and texture information. The image, on the other hand, has an inherent depth ambiguity but offers fine-grained texture and color information. This offers the compelling research opportunity of how to design a detector which utilizes the best of two worlds.

Early work on KITTI [6] such as MV3D [3] and AVOD [10] proposed multi-view fusion pipelines to exploit these synergies. However recent detectors such as PointPillars [11], VoxelNet [34, 29] and STD [32] use only lidar and still significantly outperform these methods. Indeed, despite recent fusion research [18, 27, 13, 33], the top methods on the popular KITTI leaderboard [6] are lidar only. Does this mean lidar makes vision redundant for 3D object detection?

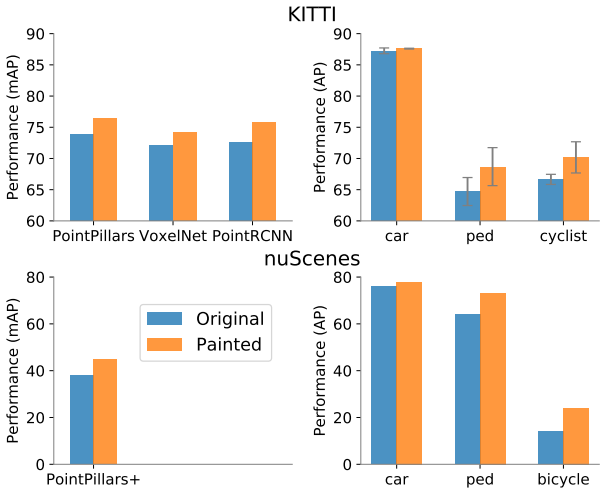

Figure 1. PointPainting is a general fusion method that be can used with any lidar detection network. Top left: PointPillars[11], VoxelNet [34, 29], and PointRCNN [21] on the KITTI [6] bird's-eye view *val* set (Table 1). The painted version of PointRCNN is state of the art on the KITTI *test* set outperforming all published fusion and lidar-only methods (Table 2). Top right: improvements are larger for the harder pedestrian (ped.) and cyclist classes. Error bars indicate std. across methods. Bottom left: PointPillars+ evaluated on the nuScenes [1] *test* set. The painted version of PointPillars+ improves all 10 classes for a total boost of 6.3 mAP (Table 3). Bottom right: selected class improvements for Painted PointPillars+ show the challenging bicycle class has the largest gains.

The answer, surely, must be no. Consider the example in Fig. 3, where the pedestrian and signpost are clearly visible in the image, yet look more or less identical in the lidar modality. Surely vision based semantic information should be useful to improve detection of such objects. Also, by first principle, adding more information should at the minimum yield the *same* result, not *worse*. So why has it been so difficult? One reason is due to viewpoint misalignment.

While both sensors are natively captured in the range-view, most state of the art methods such as PointPillars [11] or STD [32] use convolutions in the bird's-eye view. This

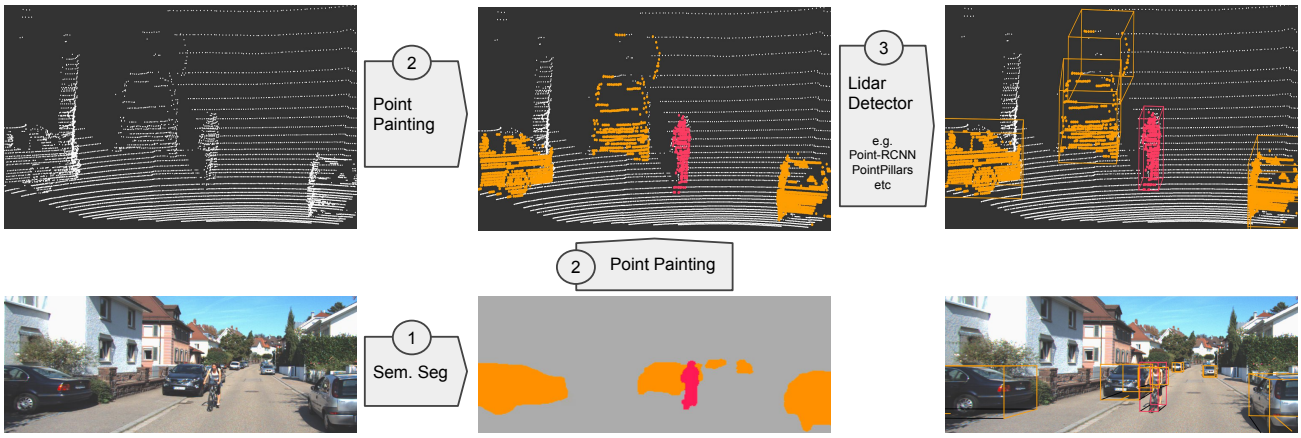

Figure 2. PointPainting overview. The PointPainting architecture consists of three main stages: (1) image based semantics network, (2) fusion (painting), and (3) lidar based detector. In the first step, the images are passed through a semantic segmentation network obtaining pixelwise segmentation scores. In the second stage, the lidar points are projected into the segmentation mask and decorated with the scores obtained in the earlier step. Finally, a lidar based object detector can be used on this decorated (painted) point cloud to obtain 3D detections.

view has several advantages including lack of scale ambiguity and minimal occlusions. It also does not suffer from the depth-blurring effect which occurs with applying 2D convolutions to the range view [25]. As a result, bird's-eye view methods outperform top range-view methods, such as LaserNet [17, 16], on the KITTI leaderboard. However, while a lidar point cloud can trivially be converted to bird's-eye view, it is much more difficult to do so with an image.

Hence, a core challenge of sensor fusion network design lies in consolidating the lidar bird's-eye view with the camera view. Previous methods can be grouped into four categories: object-centric fusion, continuous feature fusion, explicit transform and detection seeding.

Object-centric fusion, pioneered by MV3D [3] and AVOD [10], is the most obvious choice for a two-stage architecture. Here, the modalities have different backbones, one in each view, and fusion happens at the object proposal level by applying roi-pooling in each modality from a shared set of 3D proposals. This allows for end-to-end optimization but tends to be slow and cumbersome.

A second family of methods applies "continuous feature fusion" to allow feature information to be shared across all strides of the image and lidar backbones [14, 27]. These methods can be used with single-state detection designs but require a mapping to be calculated, *a priori*, for each sample, from the point-cloud to the image. One subtle but important draw-back of this family of methods is "feature-blurring". This occurs since each feature vector from the bird's-eye view corresponds to multiple pixels in the image-view, and vice versa. ContFuse [14] proposes a sophisticated method based on kNN, bilinear interpolation and a learned MLP to remedy this, but the core problem persists.

A third family of methods attempts to explicitly transform the image to a bird's-eye view representation [20] and do the fusion there [25, 28, 33]. Some of the most promising *image-only* methods use this idea of first creating an artificial point cloud from the image and then proceeding in the bird's-eye view [25, 28]. Subsequent work attempts fusion based on this idea, but the performance falls short of state of the art [33], and requires several expensive steps of processing to build the pseudo-point cloud.

A fourth family of methods uses detection seeding. There, semantics are extracted from an image *a priori* and used to seed detection in the point cloud. Frustrum PointNet [18] and ConvNet [26] use the 2D detections to limit the search space inside the frustum while IPOD [31] uses semantic segmentation outputs to seed the 3D proposal. This improves precision, but imposes an upper bound on recall.

The recent work of Liang et. al [13] tried combining several of these concepts. Results are not disclosed for all classes, but the method is outperformed on the car class by the top lidar-only method STD [32] (Table 2).

In this work we propose PointPainting: a simple yet effective sequential fusion method. Each lidar point is projected into the output of an image semantic segmentation network and the channel-wise activations are concatenated to the intensity measurement of each lidar point. The concatenated (painted) lidar points can then be used in any lidar detection method, whether bird's-eye view [11, 29, 21, 34, 30, 32] or front-view [12, 17]. PointPainting addresses the shortcomings of the previous fusion concepts: it does not add any restrictions on the 3D detection architecture; it does not suffer from feature or depth blurring; it does not require a pseudo-point cloud to be computed, and it does not limit the maximum recall.

Note that for lidar detection methods that operate directly on the raw point cloud [11, 34, 17, 21], PointPainting requires minimal network adaptations such as changing the

Method	mAP	Car			Pedestrian			Cyclist		
	Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
PointPillars [11]	73.78	90.09	87.57	86.03	71.97	67.84	62.41	85.74	65.92	62.40
Painted PointPillars	76.27	90.01	87.65	85.56	77.25	72.41	67.53	81.72	68.76	63.99
Delta	+2.50	-0.08	0.08	-0.47	**+5.28**	**+4.57**	**+5.12**	-4.02	+2.84	+1.59
VoxelNet [34, 29]	71.83	89.87	87.29	86.30	70.08	62.44	55.02	85.48	65.77	58.97
Painted VoxelNet	73.55	90.05	87.51	86.66	73.16	65.05	57.33	87.46	68.08	65.59
Delta	+1.71	+0.18	+0.22	+0.36	+3.08	+2.61	+2.31	**+1.98**	+2.31	+6.62
PointRCNN [21]	72.42	89.78	86.19	85.02	68.37	63.49	57.89	84.65	67.59	63.06
Painted PointRCNN	75.80	90.19	87.64	86.71	72.65	66.06	61.24	86.33	73.69	70.17
Delta	**+3.37**	**+0.41**	**+1.45**	**+1.69**	+4.28	+2.57	+3.35	+1.68	**+6.10**	**+7.11**

Table 1. PointPainting applied to state of the art lidar based object detectors. All lidar methods show an improvement in bird's-eye view (BEV) mean average precision (mAP) of car, pedestrian, and cyclist on KITTI *val* set, moderate split. The corresponding 3D results are included in Table 7 in the Supplementary Material where we observe a similar improvement.

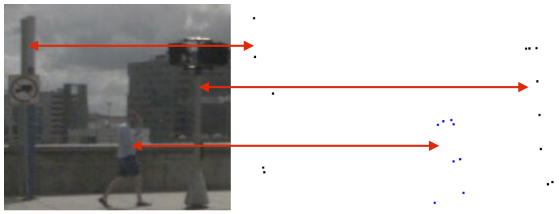

Figure 3. Example scene from the nuScenes [1] dataset. The pedestrian and pole are 25 meters away from the ego vehicle. At this distance the two objects appears very similar in the point cloud. The proposed PointPainting method would add semantics from the image making the lidar detection task easier.

number of channels dedicated to reading the point cloud. For methods using hand-coded features [30, 22], some extra work is required to modify the feature encoder.

PointPainting is sequential by design which means that it is not always possible to optimize, end-to-end, for the final task of 3D detection. In theory, this implies sub-optimality in terms of performance. Empirically, however, PointPainting is more effective than all other proposed fusion methods. Further, a sequential approach has other advantages: (1) semantic segmentation of an image is often a useful stand-alone intermediate product, and (2) in a real-time 3D detection system, latency can be reduced by pipelining the image and lidar networks such that the lidar points are decorated with the semantics from the previous image. We show in ablation that such pipelining does not affect performance.

We implement PointPainting with three state of the art lidar-only methods that have public code: PointPillars [11], VoxelNet (SECOND) [34, 29], and PointRCNN [21]. PointPainting consistently improved results (Figure 1) and indeed, the painted version of PointRCNN achieves state of the art on the KITTI leaderboard (Table 2). We also show a significant improvement of 6.3 mAP (Table 4) for Painted PointPillars+ on nuScenes [1].

Contributions. Our main contribution is a novel fusion method, PointPainting, that augments the point cloud with image semantics. Through extensive experimentation we show that PointPainting is:

- **general** – achieving significant improvements when used with 3 top lidar-only methods on the KITTI and nuScenes benchmarks;
- **accurate** – the painted version of PointRCNN achieves state of the art on the KITTI benchmark;
- **robust** – the painted versions of PointRCNN and PointPillars improved performance on *all classes* on the KITTI and nuScenes *test* sets, respectively.
- **fast** – low latency fusion can be achieved by pipelining the image and lidar processing steps.

2. PointPainting Architecture

The PointPainting architecture accepts point clouds and images as input and estimates oriented 3D boxes. It consists of three main stages (Fig. 2). (1) Semantic Segmentation: an image based sem. seg. network which computes the pixel wise segmentation scores. (2) Fusion: lidar points are painted with sem. seg. scores. (3) 3D Object Detection: a lidar based 3D detection network.

2.1. Image Based Semantics Network

The image sem. seg. network takes in an input image and outputs per pixel class scores. These scores serve as compact summarized features of the image. There are several key advantages of using sem. seg. in a fusion pipeline. First, sem. seg. is an easier task than 3D object detection since segmentation only requires local, per pixel classification, while object detection requires 3D localization and classification. Networks that perform sem. seg. are easier to train and are also amenable to perform fast inference. Second, rapid advances are being made in sem. seg. [4, 36], which allows PointPainting to benefit from advances in both segmentation and 3D object detection. Finally, in a robotics

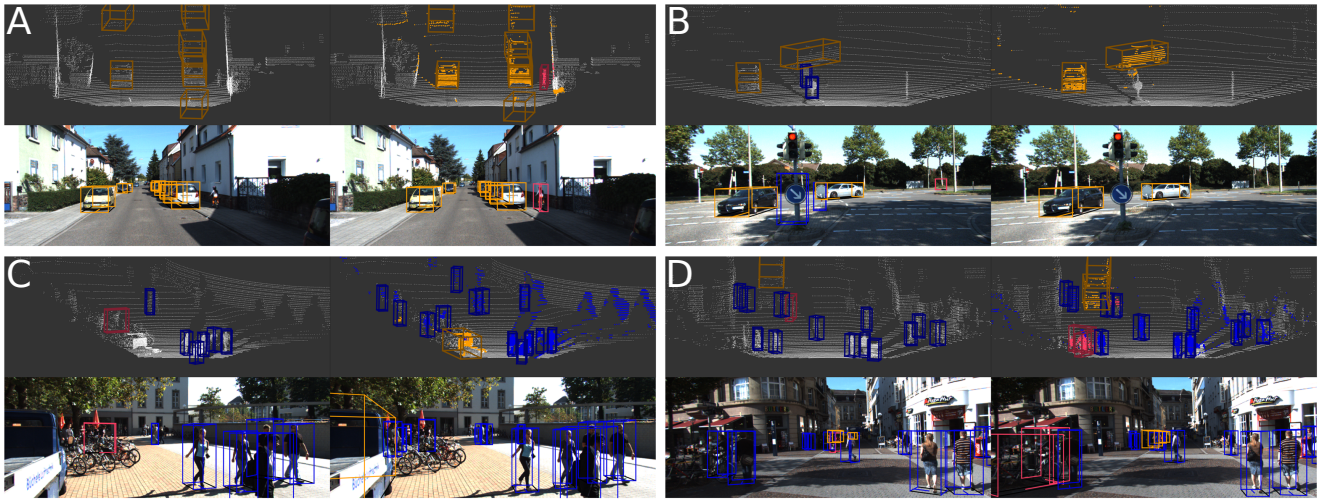

Figure 4. Qualitative analysis of KITTI results. We created four different comparison figures. For each comparison, the upper left is the original point cloud, while the upper right is the painted point cloud with the segmentation outputs used to color car (orange), cyclist (red) and pedestrian (blue) points. PointPillars / Painted PointPillars predicted 3D bounding boxes are displayed on the both the input point cloud (upper left / right) and projected into the image (lower left / right). The orientation of boxes is shown by a line connecting the bottom center to the front of the box.

or autonomous vehicle system, sem. seg. outputs are useful independent outputs for tasks like free-space estimation.

In this paper, the segmentation scores for our KITTI experiments are generated from DeepLabv3+ [2, 36, 19], while for nuScenes experiments we trained a custom, lighter, network. However, we note that PointPainting is agnostic to the image segmentation network design.

2.2. PointPainting

Algorithm 1 PointPainting(L, S, T, M)

Inputs:
Lidar point cloud $L \in \mathbb{R}^{N,D}$ with N points and $D \geq 3$.
Segmentation scores $S \in \mathbb{R}^{W,H,C}$ with C classes.
Homogenous transformation matrix $T \in \mathbb{R}^{4,4}$.
Camera matrix $M \in \mathbb{R}^{3,4}$.

Output:
Painted lidar points $P \in \mathbb{R}^{N,D+C}$

for $\vec{l} \in L$ **do**
  $\vec{l}_{\text{image}} = \text{PROJECT}(M, T, \vec{l}_{xyz})$ ▷ $\vec{l}_{\text{image}} \in \mathbb{R}^2$
  $\vec{s} = S[\vec{l}_{\text{image}}[0], \vec{l}_{\text{image}}[1], :]$ ▷ $\vec{s} \in \mathbb{R}^C$
  $\vec{p} = \text{Concatenate}(\vec{l}, \vec{s})$ ▷ $\vec{p} \in \mathbb{R}^{D+C}$
end for

Here we provide details on the painting algorithm. Each point in the lidar point cloud is (x, y, z, r) or (x, y, z, r, t) for KITTI and nuScenes respectively, where x, y, z are the spatial location of each lidar point, r is the reflectance, and t is the relative timestamp of the lidar point (applicable when using multiple lidar sweeps [1]). The lidar points are transformed by a homogenous transformation followed by a projection into the image. For KITTI this transformation is given by $T_{\text{camera}\leftarrow\text{lidar}}$. The nuScenes transformation requires extra care since the lidar and cameras operate at different frequencies. The complete transformation is:

$$T = T_{(\text{camera}\leftarrow\text{ego})} T_{(\text{ego}_{t_c}\leftarrow\text{ego}_{t_l})} T_{(\text{ego}\leftarrow\text{lidar})}$$

with transforms: lidar frame to the ego-vehicle frame; ego frame at time of lidar capture, t_l, to ego frame at the image capture time, t_c; and ego frame to camera frame. Finally, the camera matrix, M, projects the points into the image.

The output of the segmentation network is C class scores, where for KITTI $C = 4$ (car, pedestrian, cyclist, background) and for nuScenes $C = 11$ (10 detection classes plus background). Once the lidar points are projected into the image, the segmentation scores for the relevant pixel, (h, w), are appended to the lidar point to create the painted lidar point. Note, if the field of view of two cameras overlap, there will be some points that will project on two images simultaneously and we randomly choose the segmentation score vector from one of the two images. Another strategy can be to choose the more discriminative score vector by comparing their entropies or the margin between the top two scores. However, we leave that for future studies.

2.3. Lidar Detection

The decorated point clouds can be consumed by any lidar network that learns an encoder, since PointPainting just changes the input dimension of the lidar points. PointPainting can also be utilized by lidar networks with hand-

Method	Modality	mAP	Car			Pedestrian			Cyclist		
		Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D[3]	L & I	N/A	86.62	78.93	69.80	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN[10]	L & I	64.07	90.99	84.82	79.62	58.49	**50.32**	**46.98**	69.39	57.12	51.09
IPOD[31]	L & I	64.60	89.64	84.62	79.96	**60.88**	49.79	45.43	78.19	59.40	51.38
F-PointNet[18]	L & I	65.20	91.17	84.67	74.77	57.13	49.57	45.48	77.26	61.37	53.78
F-ConvNet[26]	L & I	67.89	91.51	85.84	76.11	57.04	48.96	44.33	**84.16**	68.88	60.05
MMF[13]	L, I & M	N/A	93.67	88.21	81.99	N/A	N/A	N/A	N/A	N/A	N/A
LaserNet[17]	L	N/A	79.19.	74.52	68.45	N/A	N/A	N/A	N/A	N/A	N/A
SECOND[29]	L	61.61	89.39	83.77	78.59	55.99	45.02	40.93	76.5	56.05	49.45
PointPillars[11]	L	65.98	90.07	86.56	82.81	57.60	48.64	45.78	79.90	62.73	55.58
STD[32]	L	68.38	**94.74**	**89.19**	**86.42**	60.02	48.72	44.55	81.36	67.23	59.35
PointRCNN[21]	L	66.92	92.13	87.39	82.72	54.77	46.13	42.84	82.56	67.24	60.28
Painted PointRCNN	L & I	**69.86**	92.45	88.11	83.36	58.70	49.93	46.29	83.91	**71.54**	**62.97**
Delta	Δ I	**+2.94**	**+0.32**	**+0.72**	**+0.64**	**+3.93**	**+3.80**	**+3.45**	**+1.35**	**+4.30**	**+2.69**

Table 2. Results on the KITTI test BEV detection benchmark. We see that Painted PointRCNN sets a new state of the art (**69.86 mAP**) in BEV detection performance. The modalities are lidar (L), images (I), and maps (M). The delta is the difference due to Painting, ie Painted PointRCNN minus PointRCNN. The corresponding 3D results are included in Table 8 in the Supplementary Material.

engineered encoder [30, 22], but requires specialized feature engineering for each method. In this paper, we demonstrate that PointPainting works with three different lidar detectors: PointPillars [11], VoxelNet [34, 29], and PointRCNN [21]. These are all state of the art lidar detectors with distinct network architectures: single stage (PointPillars, VoxelNet) vs two stage (PointRCNN), and pillars (PointPillars) vs voxels (VoxelNet) vs point-wise features (PointRCNN). Despite these different design choices, all lidar networks benefit from PointPainting (Table 1). Note that we were as inclusive as possible in this selection, and to the best of our knowledge, these represent all of the top KITTI detection leaderboard methods that have public code.

3. Experimental setup

In this section we present details of each dataset and the experimental settings of PointPainting.

3.1. Datasets

We evaluate our method on the KITTI and nuScenes datasets.

KITTI. The KITTI dataset [6] provides synced lidar point clouds and front-view camera images. It is relatively small with 7481 samples for training and 7518 samples for testing. For our test submission, we created a minival set of 784 samples from the training set and trained on the remaining 6733 samples. The KITTI object detection benchmark requires detection of cars, pedestrians, and cyclists. Ground truth objects were only annotated if they are visible in the image, so we follow the standard practice [3, 34] of only using lidar points that project into the image.

nuScenes. The nuScenes dataset [1] is larger than the KITTI dataset (7x annotations, 100x images). It it annotated with 3D bounding boxes for 1000 20-second scenes at 2Hz resulting in 28130 samples for training, 6019 samples for validation and 6008 samples for testing. nuScenes comprises the full autonomous vehicle data suite: synced lidar, cameras and radars with complete 360 coverage; in this work, we use the lidar point clouds and RGB images from all 6 cameras. The 3D object detection challenge evaluates the performance on 10 classes: cars, trucks, buses, trailers, construction vehicles, pedestrians, motorcycles, bicycles, traffic cones and barriers. Further, the dataset has an imbalance challenge with cars and pedestrians most frequent, and construction vehicles and bicycles least frequent.

3.2. Semantics Network Details

Here we provide more details on the semantics networks.

KITTI. For experiments on KITTI [6], we used the DeepLabv3+ network[1]. The network was first pretrained on Mapillary [7], then finetuned on Cityscapes [5], and finally finetuned again on KITTI pixelwise sem. seg. [6]. Note that the class definition of cyclist differs between KITTI sem. seg. and object detection: in detection a cyclist is defined as rider + bike, while in sem. seg. a cyclist is defined as only the rider with bike a separate class. There was therefore a need to map bikes which had a rider to the cyclist class, while supressing parked bikes to background. We did this after painting by mapping all points painted with the bike class within a $1m$ radius of a rider to the cyclist class; the rest to background.

nuScenes. There was no public semantic segmentation method available on nuScenes so we trained a custom network using the nuImages dataset.[2] nuImages consists of

[1] https://github.com/NVIDIA/semantic-segmentation

[2] We used an early access version; https://www.nuscenes.org/images.

Methods	mAP	Car	Truck	Bus	Trailer	Ctr. Vhl.	Ped.	Motorcycle	Bicycle	Tr. Cone	Barrier
PointPillars [11, 1]	30.5	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
PointPillars+	40.1	76.0	31.0	32.1	36.6	11.3	64.0	34.2	14.0	45.6	56.4
Painted PointPillars+	46.4	77.9	35.8	36.1	37.3	15.8	73.3	41.5	24.1	62.4	60.2
Delta	**+6.3**	**+1.9**	**+4.8**	**+3.9**	**+0.7**	**+4.5**	**+9.3**	**+7.3**	**+10.1**	**+16.8**	**+3.8**

Table 3. Per class nuScenes performance. Evaluation of detections as measured by average precision (AP) or mean AP (mAP) on nuScenes test set. Abbreviations: construction vehicle (Ctr. Vhl.), pedestrian (Ped.), and traffic cone (Tr. Cone).

100k images annotated with 2D bounding boxes and segmentation labels for all nuScenes classes. The segmentation network uses a ResNet [9] backbone to generate features at strides 8 to 64 for a FCN [15] segmentation head that predicts the nuScenes segmentation scores.

3.3. Lidar Network Details

We perform experiments using three different lidar networks: PointPillars [11], VoxelNet [34, 29], and PointRCNN [21]. The fusion versions of each network that use PointPainting will be referred to as being painted (e.g. Painted PointPillars).

KITTI. We used the publicly released code for PointPillars[3], VoxelNet[4] and PointRCNN[5] and decorate the point cloud with the sem. seg. scores for 4 classes. This changes the original decorated point cloud dimensions from $9 \rightarrow 13$, $7 \rightarrow 11$, and $4 \rightarrow 8$ for PointPillars, VoxelNet, and PointRCNN respectively. For PointPillars, the new encoder has $(13, 64)$ channels, while for VoxelNet it has $(11, 32), (64, 128)$ channels. The 8 dimensional painted point cloud for PointRCNN is given as input to both the encoder and the region pooling layer. No other changes were made to the public experimental configurations.

nuScenes. We use PointPillars for all nuScenes experiments. This requires changing the decorated point cloud from $7 \rightarrow 18$, and the encoder has $(18, 64)$ channels now.

In order to make sure the effect of painting is measured on a state of the art method, we made several improvements to the previously published PointPillars setup [1] boosting the mAP by 10% on the nuScenes bechmark (Table 4). We refer to this improved baseline as PointPillars+. The changes are inspired by [35] and comprise modifying pillar resolution, network architecture, attribute estimation, sample weighting, and data augmentation. First, the pillar resolution was reduced from 0.25 m to 0.2 m to allow for better localization of small objects. Second, the network architecture was changed to include more layers earlier in the network. Third, neither PointPillars nor PointPillars+ predict attributes, instead the attribute estimation heuristic was improved. Rather than using the most common attribute for each class, the predicted velocities and heights of each box are used to better estimate each attribute. Fourth, to reduce the class imbalance during training, a sample based weighting method was used where each sample was weighted according to the number of annotations in the sample. Fifth, the global yaw augmentation was changed from π to $\pi/6$.

[3]https://github.com/nutonomy/second.pytorch
[4]https://github.com/traveller59/second.pytorch
[5]https://github.com/sshaoshuai/PointRCNN

4. Results

In this section, we present PointPainting results on the KITTI and nuScenes datasets and compare to the literature.

4.1. Quantitative Analysis

4.1.1 KITTI

All detection results are measured using the official KITTI evaluation detection for bird's-eye view (BEV) and 3D. The BEV results are presented here while the 3D results are included in the Supplementary Material. The KITTI dataset is stratified into easy, moderate, and hard difficulties, and the official KITTI leaderboard is ranked by performance on moderate average precision (AP).

Validation Set First, we investigate the effect of PointPainting on three leading lidar detectors. Fig. 1 and Table 1 demonstrate that PointPainting improves the detection performance for PointPillars [11], VoxelNet [34, 29], and PointRCNN [21]. The PointPainting semantic information led to a widespread improvement in detection: 24 of 27 comparisons (3 experiments $\times$ 3 classes $\times$ 3 strata) were improved by PointPainting. While the greatest changes were for the more challenging scenarios of pedestrian and cyclist detection, most networks even saw an improvement on cars. This demonstrates that the utility of PointPainting is independent of the underlying lidar network.

Test Set Here we compare PointPainting with state of the art KITTI test results. The KITTI leaderboard only allows one submission per paper, so we could not submit all Painted methods from Table 1. While Painted PointPillars performed better than Painted PointRCNN on the val set, of the two only PointPillars has public code for nuScenes. Therefore, to establish the generality of PointPainting, we chose to submit Painted PointPillars results to nuScenes test, and use our KITTI submission on Painted PointRCNN.

Method	Modality	NDS	mAP
MonoDis [23]	Images	38.4	30.4
PointPillars [11, 1]	Lidar	45.3	30.5
PointPillars+	Lidar	55.0	40.1
Painted PointPillars+	Lidar & Images	58.1	46.4
MEGVII [35]	Lidar	63.3	52.8

Table 4. nuScenes test results. Detection performance is measured by nuScenes detection score (NDS) [1] and mean average precision (mAP).

As shown in Table 2, PointPainting leads to a robust improvement on the test set for PointRCNN: the average precision increases for every single class across all strata. Painted PointRCNN establishes new state of the art performance on mAP and cyclist AP.

Based on the consistency of Painted PointRCNN improvements between val and test ($+2.73$ and $+2.94$ respectively), and the generality of PointPainting (Table 1), it is reasonable to believe that other methods in Table 2 would decidedly improve with PointPainting. The strength, generality, robustness, and flexibility of PointPainting suggests that it is the leading method for image-lidar fusion.

4.2. nuScenes

To establish the versatility of PointPainting, we examine Painted PointPillars results on nuScenes. As a first step, we strengthened the lidar network baseline to PointPillars+. Even with this stronger baseline, PointPainting increases mean average precision (mAP) by $+6.3$ on the test set (Table 4). Painted PointPillars+ is only beat by MEGVII's lidar only method on nuScenes. However, MEGVII's network [35] is impractical for a realtime system since it is an extremely large two stage network that requires high resolution inputs and uses multi-scale inputs and ensembles for test evaluation. Therefore, Painted PointPillars+ is the leading realtime method on nuScenes.

The detection performance generalized well across classes with every class receiving a boost in AP from PointPainting (Table 3). In general, the worst performing detection classes in PointPillars+ benefited the most from painting, but there were exceptions. First, traffic cones received the largest increase in AP ($+16.8$) despite already having robust PointPillars+ detections. This is likely because traffic cones often have very few lidar points on them, so the additional information provided by semantic segmentation is extremely valuable. Second, trailer and construction vehicles had lower detection gains, despite starting from a smaller baseline. This was a consequence of the segmentation network having its worst recall on these classes (overall recall of 72%, but only 39% on trailers and 40% on construction vehicles; see Supplementary Material for details). Finally, despite a baseline of 76 AP, cars still received a $+1.9$ AP boost, signaling the value of semantic information even for classes well detected by lidar only.

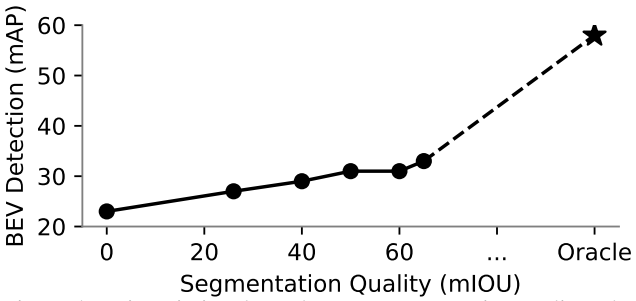

Figure 5. PointPainting dependency on segmentation quality. The Painted PointPillars detection performance, as measured by mean average precision (mAP) on the val split, is compared with respect to the quality of semantic segmentation network used in the painting step, as measured by mean intersection over union (mIoU). The oracle uses the 3D bounding boxes as semantic segmentation.

4.3. Qualitative Analysis

Here we give context to the evaluation metrics with some qualitative comparisons in Fig. 4 using Painted PointPillars, the best performing network on KITTI val set. In Fig. 4 A, original PointPillars correctly detects the cars, but misses a cyclist. The painted point cloud resolves this and the cyclist is detected. It also yields better orientation estimates for the vehicles. A common failure mode of lidar based methods is confusion between pedestrians and poles (Fig. 3). As expected, PointPainting can help resolve this (Fig. 4 B). Fig. 4 C suggests that the lidar detection step can correct incorrect painting. The loose segmentation masks in the image correctly paint nearby pedestrians, but extra paint also gets splattered onto the wall behind them. Despite this incorrect semantic information, the network does not predict false positive pedestrians. This leaves unanswered the precise characteristics of sem. seg. (e.g. precision vs recall) to optimize for PointPainting. In Fig. 4 D, Painted PointPillars predicts two false positive cyclists on the left because of two compounding mistakes. First, the sem. seg. network incorrectly predicts pedestrians as riders as they are so close to the parked bikes. Next, the heuristic that we used to resolve the discrepancy in the cyclist definition between detection and segmentation annotations (See Section 3.2) exacerbated the problem by painting all bikes with the cyclist class. However, throughout the rest of the crowded scene, the painted points lead to better oriented pedestrians, fewer false positives, and better detections of far away cars.

5. Ablation Studies

Here we perform ablation studies on the nuScenes dataset. All studies used the Painted PointPillars architecture and were trained for a quarter of the training time as compared to the test submissions. Using the one-cycle optimizer [24], we achieved 33.9 mAP and 46.3 NDS on the

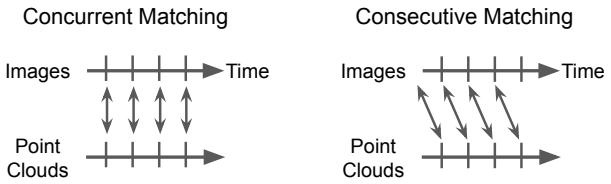

Figure 6. Reducing latency by pipelining. A Painted lidar network requires both point clouds and images. Using the most recent image (Concurrent Matching) adds latency since the lidar network must wait for the image segmentation results. This latency can be minimized using pipelining if the Painted network uses the segmentation mask of previous images (Consecutive Matching). Using consecutive matching, we found that Painted PointPillars only adds a latency of 0.75 ms over the original PointPillars architecture. See Supplementary Material for further details.

nuScenes val set as opposed to 44.85 mAP and 57.34 NDS for full training of Painted PointPillars+.

5.1. Dependency on Semantics

Quality. In PointPainting, the lidar points are fused with the semantic segmentation of the image. We investigate the impact of the semantic segmentation quality on the final detection performance. Using nuScenes, we generate a series of sem. seg. networks with varying segmentation quality by using multiple intermediate checkpoints from training. As shown in Fig. 5, improved sem. seg. (as measured by mean IOU), leads to improved 3D object detection.

For an upper bound, we include an "oracle" which uses the ground truth 3D boxes to paint the lidar points. This significantly improves the detection performance (+27 mAP), which demonstrates that advances in semantic segmentation would radically boost 3D object detection.

Using the oracle doesn't guarantee a perfect mAP because of several limitations. First, the ground truth bounding box can contain irrelevant points (e.g. from the ground). Second, nuScenes annotates all objects that contain a single lidar point. Turning one lidar point into an accurate, oriented 3D bounding box is difficult. Third, we trained it for the same total time as the other ablation studies, but it would probably benefit from longer training. Finally, PointPillars' stochastic sampling of the point cloud could significantly filter, or eliminate, the points that contain semantic information if the ground truth object contains only a few points.

Scores vs Labels. We investigate the effect of the segmentation prediction format on detection performance. To do so we convert the segmentation scores to a one hot encoding, effectively labelling each pixel as the class with the highest score. When using the labels instead of scores, the NDS was unchanged and the mAP was, surprisingly, +0.4 higher. However, the gains are marginal and within the noise of training. We also hypothesize that for future studies, a combination of calibrated [8] segmentation scores and a larger PointPillars encoder would perform better.

Comparing these results with the segmentation quality ablation suggests that future research focus more on improving segmentation quality and less on representation.

Method	Matching	NDS	mAP
Painted PointPillars	Concurrent	46.3	33.9
Painted PointPillars	Consecutive	46.4	33.9

Table 5. Time delay analysis. Painted PointPillars results on nuScenes when using concurrent matching (which incurs latency), or consecutive matching (which allows real-time pipelining) as shown in Figure 6. The use of the previous image minimizes latency without any drop in detection performance.

5.2. Sensitivity to Timing

We investigate the sensitivity of the lidar network to delays in semantic information. In the simplest scenario, which we used in all previous results, each point cloud is matched to the most recent image (Concurrent Matching - Fig. 6). However, this will introduce a latency in a real time system as the fusion step will have to wait for the image based sem. seg. scores. To eliminate the latency, the sem. seg. scores of the previous image can be pipelined into the lidar network (Consecutive Matching - Fig. 6). This involves an ego-motion compensation step where the lidar pointcloud is first transformed to the coordinate system of the ego-vehicle in the last frame followed by a projection into the image to get the segmentation scores. Our experiments suggest that using the previous images does not degrade detection performance (Table 5). Further, we measure that PointPainting only introduces an additional latency of 0.75 ms for the Painted PointPillars architecture (see Supplementary Material for details). This demonstrates that PointPainting can achieve high detection performance in a realtime system with minimal added latency.

6. Conclusion

In this paper, we present PointPainting, a novel sequential fusion method that paints lidar point clouds with image based semantics. PointPainting produces state of the art results on the KITTI and nuScenes challenges with multiple different lidar networks. The PointPainting framework is flexible and can combine the outputs of any segmentation network with any lidar network. The strength of these results and the general applicability demonstrate that PointPainting is the leading architecture when fusing image and lidar information for 3D object detection.

7. Acknowledgements

We thank Donghyeon Won, Varun Bankiti and Venice Liong for help with the semantic segmentation model for nuScenes, and Holger Caesar for access to nuImages.

References

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1, 3, 4, 5, 6, 7

[2] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 4

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 1, 2, 5, 11

[4] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen. Panoptic-deeplab. *ICCV Workshop*, 2019. 3

[5] 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. In *CVPR*, 2016. 5

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 1, 5

[7] N. Gerhard, T. Ollmann, S. R. Bulo, and P. Kontschieder. The Mapillary Vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 5

[8] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 8

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[10] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 2018. 1, 2, 5, 11

[11] 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. 1, 2, 3, 5, 6, 7, 11

[12] B. Li, T. Zhang, and T. Xia. Vehicle detection from 3d lidar using fully convolutional network. In *RSS*, 2016. 2

[13] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019. 1, 2, 5, 11

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

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 6

[16] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez. Sensor fusion for joint 3d object detection and semantic segmentation. In *CVPR Workshop*, 2019. 2

[17] 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, 5, 11

[18] 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. 1, 2, 5, 11

[19] F. A. Reda, G. Liu, K. J. Shih, R. Kirby, J. Barker, D. Tarjan, A. Tao, and B. Catanzaro. Sdc-net: Video prediction using spatially-displaced convolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 718–733, 2018. 4

[20] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3d object detection. In *BMVC*, 2019. 2

[21] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 2, 3, 5, 6, 11

[22] M. Simon, S. Milz, K. Amende, and H.-M. Gross. Complex-yolo: Real-time 3d object detection on point clouds. *CoRR*, abs/1803.06199, 2018. 3, 5

[23] A. Simonelli, S. Rota Bulo, L. Porzi, M. Lopez-Antequera, and P. Kontschieder. Disentangling monocular 3d object detection. *ICCV*, 2019. 7

[24] L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018. 7

[25] 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. 2

[26] Z. Wang and K. Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *IROS*, 2019. 2, 5, 11

[27] Z. Wang, W. Zhan, and M. Tomizuka. Fusing bird's eye view lidar point cloud and front view camera image for 3d object detection. In *IVS*, 2018. 1, 2

[28] X. Weng and K. Kitani. Monocular 3d object detection with pseudo-lidar point cloud. *ICCV Workshop*, 2019. 2

[29] Y. Yan, Y. Mao, and B. Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018. 1, 2, 3, 5, 6, 11

[30] B. Yang, W. Luo, and R. Urtasun. PIXOR: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 2, 3, 5

[31] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. Ipod: Intensive point-based object detector for point cloud. *arXiv preprint arXiv:1812.05276*, 2018. 2, 5, 11

[32] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *CVPR*, pages 1951–1960, 2019. 1, 2, 5, 11

[33] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. *ICLR*, 2020. 1, 2

[34] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1, 2, 3, 5, 6, 11

[35] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019. 6, 7

[36] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *CVPR*, 2019. 3, 4

PointPainting: Sequential Fusion for 3D Object Detection

Supplementary Material

A. PointPainting: 3D results

In this section, we present 3D results of PointPainting on the KITTI validation and test sets.

Validation Set Similar to bird's-eye view results (Table 1), we see that PointPainting substantially improves 3D detection performance on the validation set. As seen in Table 7, 23 out of 27 comparisons (3 experiments x 3 classes x 3 strata) were improved by PointPainting.

Test Set In the test set (Table 8), we observe that PointPainting consistently improves 3D detection results of PointRCNN on the pedestrians and cyclists classes across all difficulty strata (easy, medium and hard). However, we see that the 3D results on the car class drops substantially. We think this could be because of overfitting on our minival set which was very small (see Section 3.1).

B. PointPainting Latency

In Section 5.2, we concluded that Consecutive matching (see Figure 6) can minimize the latency introduced by PointPainting without any drop in detection performance. Here we provide a detailed breakdown of the latency introduced by PointPainting in the case of Consecutive matching.

Projection This step involves transforming the pointcloud to the coordinate system of the ego-vehicle in the previous frame followed by a projection into the camera images to get the segmentation scores. This operation only adds a latency of 0.15 ms.

Encoding The Painted PointPillars encoder operates on an 18 dimensional decorated pointcloud as opposed to the 7 dimensional pointcloud in the original PointPillars architecture. We measure the runtimes for both the encoders in TensorRT and find that PointPainting adds an additional latency of 0.6 ms in the encoding stage.

Thus, Painted PointPillars only introduces an additional latency of 0.75 ms over PointPillars when Consecutive matching is used. This makes Painted PointPillars a strong candidate for realtime camera-lidar fusion.

Class	Recall (%)	Precision (%)
Car	94	89
Bus	71	92
Construction Vehicle	40	58
Trailer	39	79
Truck	69	76
Motorcycle	89	87
Bicycle	58	84
Pedestrian	80	86
Barrier	81	80
Traffic Cone	78	84

Table 6. Class-wise Precision and Recall of the semantic segmentation network trained on the nuImages dataset.

C. nuImages Semantic Segmentation

Here we present some stats on the semantic segmentation network that we trained on the nuImages dataset. The mean intersection over union (mIoU) on the validation set was 0.65. The class-wise precision and recall on the validation set is shown in Table 6. Our model performs the best on the car class and worst on the construction vehicle and trailer classes.

Method	mAP	Car			Pedestrian			Cyclist		
	Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
PointPillars [11]	66.96	87.22	76.95	73.52	65.37	60.66	56.51	82.29	63.26	59.82
Painted PointPillars	69.03	86.26	76.77	70.25	71.50	66.15	61.03	79.12	64.18	60.79
Delta	+2.07	-0.96	-0.18	-3.27	**+6.13**	**+5.49**	**+4.52**	-3.17	+0.92	+0.97
VoxelNet [34, 29]	67.12	86.85	76.64	74.41	67.79	59.84	52.38	84.92	64.89	58.59
Painted VoxelNet	68.01	87.15	76.66	74.75	68.57	60.93	54.01	85.61	66.44	64.15
Delta	+0.89	+0.3	+0.02	+0.34	+0.78	+1.09	+1.63	+0.69	+1.55	**+5.56**
PointRCNN [21]	67.01	86.75	76.05	74.30	63.29	58.32	51.59	83.68	66.67	61.92
Painted PointRCNN	70.34	88.38	77.74	76.76	69.38	61.67	54.58	85.21	71.62	66.98
Delta	**+3.33**	**+1.63**	**+1.69**	**+2.46**	+6.09	+3.35	+2.29	**+1.53**	**+4.95**	+5.06

Table 7. PointPainting applied to state of the art lidar based object detectors. All lidar methods show an improvement in 3D mean average precision (mAP) of car, pedestrian, and cyclist on KITTI *validation* set, moderate split.

Method	Modality	mAP	Car			Pedestrian			Cyclist		
		Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D[3]	L & I	N/A	74.97	63.63	54.00	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN[10]	L & I	54.86	83.07	71.76	65.73	50.46	42.27	39.04	63.76	50.55	44.93
F-PointNet[18]	L & I	56.02	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.12	49.01
F-ConvNet[26]	L & I	**61.61**	87.36	76.39	66.69	52.16	43.38	38.80	**81.98**	**65.07**	**56.54**
MMF[13]	L, I & M	N/A	**88.40**	77.43	70.22	N/A	N/A	N/A	N/A	N/A	N/A
PointPillars[11]	L	58.29	82.58	74.31	68.99	51.45	41.92	**38.89**	77.10	58.65	51.92
STD[32]	L	61.25	87.95	**79.71**	75.09	**53.29**	**42.47**	38.35	78.69	61.59	55.30
PointRCNN[21]	L	57.94	86.96	75.64	**70.70**	47.98	39.37	36.01	74.96	58.82	52.53
Painted PointRCNN	L & I	58.82	82.11	71.70	67.08	50.32	40.97	37.87	77.63	63.78	55.89
Delta	Δ I	**+0.88**	-4.85	-3.94	-3.62	**+2.34**	**+1.6**	**+1.86**	**+2.67**	**+4.96**	**+3.36**

Table 8. Results on the KITTI test 3D detection benchmark. The modalities are lidar (L), images (I), and maps (M). The delta is the difference due to Painting, ie Painted PointRCNN minus PointRCNN. We don't include a few entries from Table 2 because LaserNet[17] did not publish 3D results and SECOND[29], IPOD[31] no longer have their entries on the public leaderboard since KITTI changed to a 40 point interpolated AP metric instead of 11.