HALS: A Height-Aware Lidar Super-Resolution Framework for Autonomous Driving

George Eskandar[1] Sanjeev Sudarsan[1] Karim Guirguis[2,3] Janaranjani Palaniswamy[1] Bharath Somashekar[1] Bin Yang[1]
University of Stuttgart[1] Robert Bosch GmbH[1] Karlsruhe Institute of Technology[2]

Abstract

Lidar sensors are costly yet critical for understanding the 3D environment in autonomous driving. High-resolution sensors provide more details about the surroundings because they contain more vertical beams, but they come at a much higher cost, limiting their inclusion in autonomous vehicles. Upsampling lidar pointclouds is a promising approach to gain the benefits of high resolution while maintaining an affordable cost. Although there exist many pointcloud upsampling frameworks, a consistent comparison of these works against each other on the same dataset using unified metrics is still missing. In the first part of this paper, we propose to benchmark existing methods on the KITTI dataset. In the second part, we introduce a novel lidar upsampling model, HALS: Height-Aware Lidar Super-resolution. HALS exploits the observation that lidar scans exhibit a height-aware range distribution and adopts a generator architecture with multiple upsampling branches of different receptive fields. HALS regresses polar coordinates instead of spherical coordinates and uses a surface-normal loss. Extensive experiments show that HALS achieves state-of-the-art performance on 3 real-world lidar datasets.

1. Introduction

Light detection and ranging (lidar) pointclouds are vital for the geometrical understanding of a surrounding environment for autonomous vehicles. Numerous tasks are dependent on 3D information from lidar, such as 3D object detection [12, 16, 24, 33, 37], 3D semantic segmentation [19], localization [4], mapping [29] and path planning [18]. Mounted on the vehicle, lidar emits pulses of infrared light waves to retrieve accurate 3D position information. Although the horizontal resolution of lidar is high, the vertical resolution is usually low and depends on the number of channels present in the sensor (16, 32, 64 and 128 typically). Denser pointclouds in the vertical direction are desirable because they provide more cues about the environment, thereby improving the performance of many computer vision tasks [21]. However, high-resolution lidar sensors come at a significantly higher cost [23], which would impede their commercial use in autonomous driving. It is often the case that high-resolution sensors are used in test drives to validate a perception pipeline in an autonomous vehicle, but are then replaced by cheaper low-resolution sensors for commercial purposes.

Lidar upsampling (or super-resolution) is a promising approach toward achieving a trade-off between cost and performance. Lidar super-resolution is more challenging than its image counterpart because pointclouds are not structured in a grid-like image. Also, 3D neural networks are computationally expensive and might not scale to a large number of points. Although many pointcloud super-resolution algorithms have achieved remarkable performance, a consistent benchmark comparing the different upsampling algorithms is missing. For instance, [14, 31, 34, 36] were tested on datasets containing synthetic objects with fewer points ($\sim 10k$) than a typical lidar scan ($\sim 100k$). [11, 23] were designed and trained on lidar pointclouds extracted from a driving simulator like Carla [6], while others have been evaluated on real datasets [10, 22, 23, 27]. Moreover, different evaluation metrics have been used in various works. We argue that benchmarking the different approaches on the same dataset with unified metrics would help identify their strengths and would pave the way toward developing a better lidar upsampling model. Thus, this work attempts to close the gap on two questions: *how to benchmark different lidar super-resolution methods?* And *how to upsample real lidar scans while preserving their 3D geometry?* Most importantly, we only focus on real-world lidar data because methods that work well on synthetic data might not generalize well to real data [32].

Contribution. We present 3 main contributions.

1) We present a consistent comparative evaluation of several representative frameworks on the KITTI Raw dataset [8]. We classify the methodologies into two groups: point-based and grid-based methods. The former use PointNet operations directly on the pointcloud, while the latter project the pointcloud on image coordinates and deploy 2D convolutional networks. Our key insight from the benchmark is that grid-based methods are superior to point-based

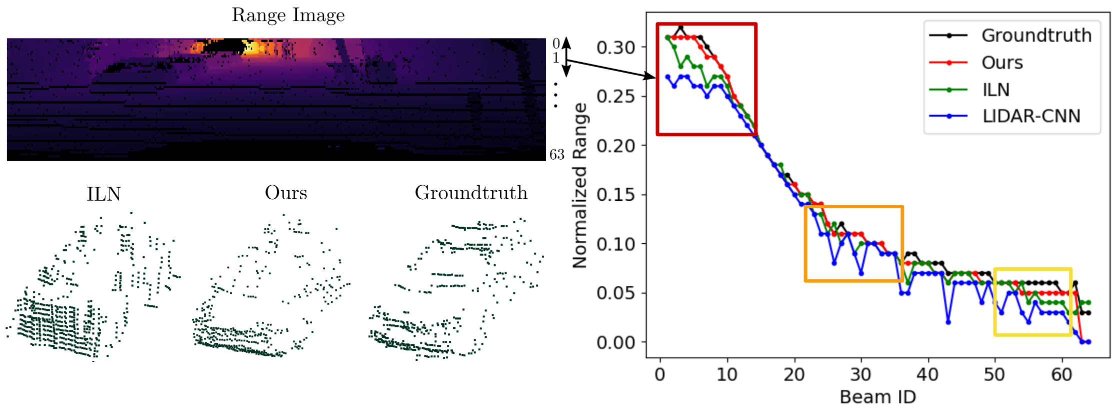

Figure 1. When projected on a 2D spherical range image, a lidar scan exhibits a height-dependent range distribution. We find that far-away objects are usually represented in the upper part of the range image (Beam ID 0 is the highest row in the range image). Upsampled lidar scans should also follow this distribution. We record the average range per beam ID of generated pointclouds from our height-aware generator and 2 state-of-the-art models on the KITTI Object Dataset and show that we can better follow the ground truth distribution. Extracted cars from upsampled lidar scans demonstrate that the overall geometry and shape of foreground objects are better preserved.

ones because of their vertically large receptive field, which spans multiple lines.

2) We take a closer look at the geometry of lidar scans and find that real-world lidar range image exhibits a height-dependent range distribution. Specifically, the beams corresponding to high elevation angles (in the upper part of the range image) have a higher average range and standard deviation. This insight is often overlooked in previous works because they use synthetic lidar pointclouds, which have a uniform range distribution and thus lead to inaccurate shapes or noisy points when applied to real pointclouds.

3) We introduce HALS, a Height-Aware Lidar Super-resolution framework that achieves state-of-the-art results on three real-world lidar datasets: KITTI Raw [8], KITTI Object [8] and nuScenes [3]. The proposed approach is designed to match best the range distribution of the high-resolution ground truth (Fig. 1). We design a generator with two upsampling branches of different receptive fields and fuse the two outputs using confidence maps that model the uncertainty of each branch in its prediction. The multi-scale receptive fields allow the network to adapt to the height-dependent range distribution by knowing *where to gather information*. Moreover, instead of regressing spherical coordinates, we regress the polar range and height of the pointcloud, achieving a more accurate synthesis by reducing the vertical quantization error in the 2D-to-3D projection. A surface normal loss is added to improve the lidar scan.

2. Related Works

Image Super-Resolution. Image super-resolution has thrived in recent years thanks to advancements in deep learning. A pioneering work, SRCNN [5], developed a CNN-based architecture to upsample images. Then, a series of enhanced frameworks have been proposed [13, 15, 30]. For instance, SRResNet [13] is a framework which deploys residual blocks and a transposed convolution layer to improve visual perception. A state-of-the-art method SWIN-IR replaces the SRResNet backbone with the SWIN transformer [17].

Pointcloud Upsampling or super-resolution refers to the task of adding more points to a pointcloud while preserving its shape. Recent pointcloud super-resolution algorithms [14, 31, 34, 36] are based on deep learning. The learning occurs in a self-supervised way by downsampling the pointclouds using either a uniform or non-uniform point dropout scheme, then upsampling them with deep learning using a reconstruction loss. Most works in this task focused on upsampling pointclouds of single objects instead of whole scenes. Note that single-object pointclouds are spatially uniform (the point density is consistent across the object), which is not the case in lidar. The question of whether these methods can extend to lidar is yet to be answered. Architectures of these works [14, 34, 36] are based on PointNet-operations [20], or graph convolution layers (AR-GCN [31]). Since these methods operate on the raw points in 3D, they are commonly referred to as *point-based methods*.

Lidar Upsampling refers to the task of upsampling a low-resolution lidar pointcloud, which typically originates from a lidar sensor with a few number of vertical channels. It is a sub-task of the pointcloud upsampling task. Lidar pointclouds depict a whole scene with many objects of different classes, and contain a considerably larger number of points than single objects pointclouds ($100k$ vs. $10k$).

Grid-based methods like [10, 11, 22, 23, 26, 27] have explicitly addressed this task. They project the pointcloud on a 2D grid map and employ 2D convolutional networks to upsample the image before reprojecting it into the 3D space.

	Point-based methods	Grid-based methods
Pointcloud Upsampling	[14, 31, 34, 36]	NA
Lidar Upsampling	[14, 31, 34, 36] [22, 26]	[10, 11, 23, 27]

Table 1. Overview of the related works. Only grid-based methods using a spherical projection are applicable to lidar upsampling.

Point-based methods, which were developed for generic pointcloud upsampling [14, 31, 34, 36], can also be applied to lidar pointclouds, although this has not yet been studied. Additionally, two point-based methods have been recently introduced to explicitly address lidar upsampling: SWD [22] employs a sliced Wasserstein distance loss, while SGA [26] proposes an upsampling algorithm without training and shows comparable performance to LIDAR-SR [23] on real outdoor scenes. We summarize this section in Tab. 1.

3. Overview of upsampling algorithms

In this section, we explore how point-based methods and grid-based methods perform in the lidar upsampling task by conducting a consistent comparison on the KITTI Raw dataset [8] with unified metrics.

Problem Formulation. Formally, let $\mathcal{P}$ be the input lidar pointcloud with N points, where $\mathcal{P} = \{(x_i, y_i, z_i) \ : i = 1, .., N\}$, and z is the vertical height. $\mathcal{P}$ can be projected onto 2D image coordinates (u, v) resulting in a range image representation $\mathcal{Q}$. We define (H, W) to be the height and width of $\mathcal{Q}$, $f = f_{up} + f_{down}$ the vertical field-of-view of the lidar sensor, $r = \sqrt{x^2 + y^2 + z^2}$ the range of the point and $d = \sqrt{x^2 + y^2}$ the radial distance. The 2D spherical projection can be expressed in Eq. (1) as follows:

$$\begin{aligned} u &= \frac{1}{2}[1 - (\arctan2(\frac{y}{x})\pi^{-1})]W, u \in [0, W] \\ v &= [1 - (\arcsin(zr^{-1}) + f_{up})f^{-1}]H, v \in [0, H] \end{aligned} \tag{1}$$

$\mathcal{Q}$ is commonly represented in spherical coordinates ($\mathcal{Q}_{u,v} = r$) [16, 19, 23, 27, 28]. u represents the azimuth while v represents the elevation. In Fig. 1, we illustrate a range image: bright values represent far away objects, while darker colors represent a small range value (near objects). Note that during the projection, multiple points can fall in the same bin (u, v). In this case, we take the point with the smallest range, while other points are considered as occluded. On the other hand, there exists some bins with no point, these are empty bins for which $\mathcal{Q}_{u,v} = 0$. The number of empty bins is small but not negligible. For example, in the KITTI Object dataset, we find 15% to 25% of the total number of bins to be empty. Eq. (1) can be inverted to project the range image back into 3D space.

We seek to transform a low-resolution pointcloud, $\mathcal{P}_{LR}$ to a high-resolution one, $\hat{\mathcal{P}}_{HR}$, with a larger number of points in the vertical direction. Note that both $\mathcal{Q}_{LR}$ and $\mathcal{Q}_{HR}$ have the same vertical field-of-view f, but different number of lines. This means that the upsampled range image, $\hat{\mathcal{Q}}_{HR}$, should have a bigger height H and the same W as $\mathcal{Q}_{LR}$. In practice, we are given a dataset consisting of high-resolution lidar scans $\mathcal{P}_{HR}$. We downsample $\mathcal{P}_{HR}$ to a lower resolution by skipping a number of lines, and we train different models to reconstruct $\mathcal{P}_{HR}$. We use the range image representation for the downsampling operation since each row in $\mathcal{Q}$ represents a lidar beam. Moreover, it is an invertible transformation, meaning $\mathcal{P}_{LR}$ can be obtained by uniformly downsampling the rows of $\mathcal{Q}_{HR}$ and projecting $\mathcal{Q}_{LR}$ back to 3D.

Experimental Setup. We train all models on a $\times 4$-upsampling factor in a self-supervised way. To downsample the high-resolution ground truth by a factor of 4, every 4th row in $\mathcal{Q}_{HR}$ is sampled, following the convention in lidar upsampling works [11, 23, 27]. While grid-based methods [11, 23, 27] upsample $\mathcal{Q}_{LR}$ into $\hat{\mathcal{Q}}_{HR}$ then compute $\hat{\mathcal{P}}_{HR}$ using the inverse of Eq. (1), point-based [14, 31, 34, 36] methods directly upsample $\mathcal{P}_{LR}$ into $\hat{\mathcal{P}}_{HR}$.

Baselines We train and evaluate point-based methods [11, 23, 27] and grid-based methods [11, 23, 27]. All models are trained with their original hyperparameters using their officially published codes. Note that the models [10, 22, 26] are not included as the code was not available to run the comparison. In our benchmark, we also include SWIN-IR [15], a state-of-the-art transformer-based image super-resolution model. SWIN-IR is trained on range images similar to the other grid-based models.

Dataset We choose the KITTI Raw dataset [8] as the benchmark. The lidar sensor in KITTI is the Velodyne HDL 64-E, which has a 64-lines resolution and around 2048 points approximately per line. However, processing this resolution with point-based methods is computationally hefty and a model with batchsize of 1 does not even fit into the used GeForce RTX 2080 Ti GPU. To ensure a fair comparison between all algorithms, we limit the size of the pointcloud to $10k$ points (40×256) using the preprocessing of [2]. We use the train/validation/test ($40k/80/700$) split used by [2].

Metrics While grid-based methods for lidar upsampling have used 2D-metrics like mean absolute error (MAE) or root mean square error (RMSE) on the range image to evaluate the performance, point-based methods have mainly used the earth-moving distance (EMD) and chamfer distance (CD) [7], because they measure the distance between subsets in $\mathcal{R}^3$. Moreover, it has been shown in [1] that EMD strongly correlates with perceptual quality. For this reason, we choose EMD and CD as the unified metrics to benchmark all methods on the KITTI Raw dataset. We present the results in Tab. 2.

Results of the comparative study. From Tab. 2, we can observe that point-based methods underperform on lidar upsampling, especially when trained from scratch on lidar. All evaluated point-based models, except PUNet, do not con-

Framework		Pretrained[1]	EMD ↓	CD ↓
Point-based	3PU [34]	✓	1265	6.58
	3PU [34]	✗	924	6.37
	PUGAN [14]	✗	866	77.82
	ARGCN [31]	✗	829	29.07
	PUGAN [14]	✓	385	0.86
	PUNet [36]	✓	371	1.70
	ARGCN [31]	✓	265	0.73
	PUNet [36]	✗	242	0.67
Grid-based	LIDAR-CNN [27]	✗	**100**	**0.05**
	LIDAR-SR [23]	✗	101	**0.05**
	ILN [11]	✗	104	0.06
	SWIN [15]	✗	101	**0.05**

[1] Pretrained on a synthetic dataset

Table 2. Quantitative comparisons on KITTI Raw Benchmark.

verge during training. For further investigation, we evaluate point-based models pretrained on a dataset of synthetic single object pointclouds proposed by PUNet [36]. Surprisingly, PUGAN and ARGCN perform better when pretrained on this dataset than when trained on the pointclouds directly. We highlight that during pretraining no lidar scan was seen, as the pretraining dataset only contains single object pointclouds. By looking at the upsampled pointclouds (Supplementary Material), we can see that, except for PUNet, no point-based method was able to replicate the lidar scan pattern. The upsampled pointclouds exhibit higher density around each line, but no new lines are added in between. We hypothesize that these networks are not able to reproduce the scan pattern by architectural design, and hence do not converge during training.

In order to understand why point-based methods (except PUNet) cannot replicate the scan pattern, we inspect and compare their architectures. In general, each network can be modeled with three parts: (i) a feature extraction backbone, (ii) a feature expansion module, and (iii) pointset generation layer. Although these parts are different across all models, it is the feature extractor of PUNet that is distinct from other algorithms due to its hierarchical design. In PUNet, point features are computed at different scales, followed by a multi-level feature aggregation resulting in a wide receptive field that captures both local and global information in the pointcloud. Subsequent architectures like PUGAN, 3PU, and ARGCN have smaller receptive fields, an intentional design choice that outperforms PUNet on single object datasets with a uniform spatial point distribution. However, a local receptive field that does not span different vertical lines in lidar scans will lead to the generation of new points located close to the input points. Consequently, the lidar scan pattern cannot be replicated (see Supplementary Material). Note that there is no straightforward way to enlarge the receptive field of these architectures. For instance, increasing the number of nearest neighbors in the layers of 3PU [34] and ARGCN [31] would entail a very high-computational cost, with only a marginal increase in receptive field. Also, there is no guarantee that increasing the number of nearest neighbors for each point will include points from other vertical channels.

On the other hand, grid-based methods [11, 23, 27] benefit from structuring the pointcloud into a range image. This representation has several desirable properties: it is invertible, it provides a 2D dense spatial grid structure to the 3D sparse and unstructured pointclouds, thereby intrinsically modeling the lidar scan pattern, and it can be efficiently processed using the widespread hardware-accelerated convolution operations. Their good performance can be attributed to the fact that even convolutional filters with small receptive fields span different lidar beams vertically. Results in Tab. 2 show little difference between all four models.

This analysis motivates us to adopt a grid-based approach as well. However, grid-based models have weaknesses: they suffer from the smoothing effects of convolutional layers, which blur edges and sharp object boundaries [23]. These effects are mitigated to some extent in the previous works either by leveraging pretrained point segmentation networks [27] (which incurs extra label costs), by filtering uncertain points [10, 23] or by using an interpolation approach [11]. We take a different approach by first analyzing the lidar scan geometry and point distribution.

4. Proposed Methodology

In this section, we first take a closer look at the range image representation, and we introduce our framework.

4.1. The Range Distribution of Different Beams

It is well-established in image super-resolution [13], that minimizing the $\mathcal{L}_1$-loss with a CNN architecture leads to blurring artifacts and neglecting high-frequencies from edges and corners. A close visual inspection of a range image reveals even more challenges compared to camera images. First, range images contain black lines or streaks, which correspond to no-return from the laser sensors. The juxtaposition of empty regions and foreground objects creates high frequencies in the 2D image space. CNNs might find it hard to process the empty regions, leading to the generation of noisy points. Second, far-away objects appear to be constrained or squished in the top rows, suggesting that the upper and lower parts of the range image have different range distributions. To confirm this second observation, we measure the average range value and standard deviation of non-zero points in each row across all lidar scans in 3 datasets and report these statistics in Fig. 2. We notice that the range distribution shows a similar tendency across the 3 datasets. The upper rows of the image have on average, a higher mean range and standard deviation than the middle and bottom rows.

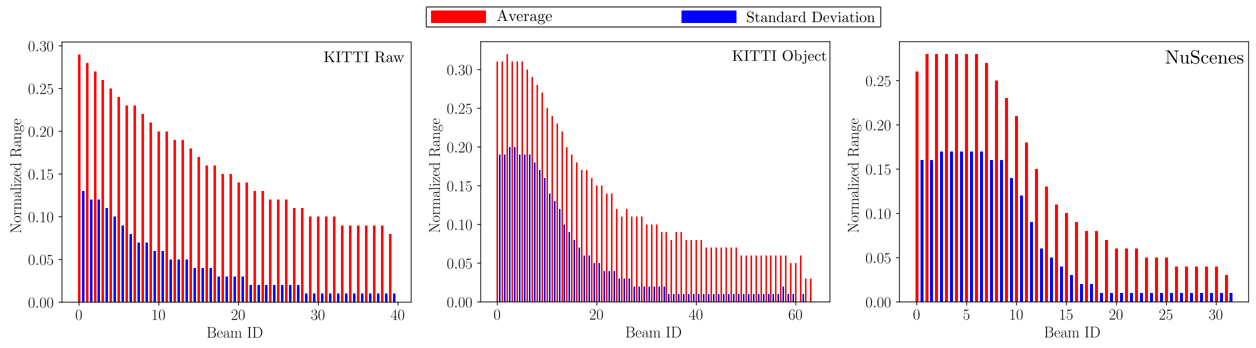

Figure 2. We record the average range and standard deviation per beam (represented as a row in the range image.) Beam 0 is the highest one from the ground. The range distribution exhibits a height-dependent behavior: far away objects are mostly represented in the upper part of the range image. Also, the standard deviation of the range is bigger in the upper rows, suggesting higher spatial frequencies.

4.2. Generator Architecture

Our goal is to design a generator architecture that is sensitive to the height of the range image. For this, we rethink the architecture of super-resolution networks and base our design on four principles to accommodate the lidar range distribution.

1. Upper and lower parts should be upsampled differently.
2. The network should be able to adaptively identify the upper and lower parts of the range image without the need for data-specific hyperparameters.
3. In the upper part of the range image, an upsampling layer should only observe a small neighborhood around each pixel, while it should be allowed to observe larger neighborhoods in the lower part. The reason for this is that far-away objects are compressed in the upper part and occupy a smaller scale, while objects in the lower parts are typically closer and bigger. The neighborhood observed by the network can be adjusted by controlling the receptive field.
4. The generator should not contain any downsampling layers (pooling, strided convolutions). The reason for this is that the height of the range image is very small compared to RGB images (e.g., 16 pixels), and downsampling layers would abstract away details in the vertical directions.

Our *key idea* is to upsample the full-sized range image multiple times with different receptive fields and fuse the outputs together using confidence maps from each upsampling branch. We find that 2 different receptive fields (a small one and a large one) are enough to cover the range distribution (Sec. 5.2). We obtain the small receptive field by cascading a small number of layers (shallow backbone), while larger receptive fields result from cascading a larger number of layers (deep backbone). To this end, we present our architecture in Fig. 3. The dimensions of the input range image $\mathcal{Q}_{LR}$ is $[B, C, H, W]$, where B is the batch size, C is the number of features of the range image (defined later) and H, W are the input resolution. The proposed generator has a point encoding layer, a feature extractor which consists of 16 blocks, two upsampling layers and two final regression layers. The point encoding layer is a shared multilayer perceptron (MLP) which transforms the input features to the high dimensional feature space (64 channels). The MLP is implemented by a 1×1 convolutional layer. Both the shallow and deep backbones are connected. The shallow backbone has 4 dilated residual blocks (DRBs) and the deep backbone has 12 DRBs. The upsampling layers are placed after each backbone, each followed by a regression layer which outputs $C+1$ channels: C channels for the range image features and 1 channel which features the mask logits. The two masks are then concatenated and passed to a softmax activation function to normalize them, such that the summation of the masks is equal to 1. We call the range image and mask from the shallow backbone $\hat{\mathcal{Q}}_{shallow}$ and $\mathbf{m}_{shallow}$. The output of the other branch consists of $\hat{\mathcal{Q}}_{deep}$ and $\mathbf{m}_{deep}$. The final range image is a weighted average of the two branches (Eq. (2)). Note that the masks are not binary: they have continuous values between 0 and 1 to weigh the predictions of each branch.

$$\begin{aligned}&\hat{\mathcal{Q}} = \hat{\mathcal{Q}}_{shallow} \cdot \mathbf{m}_{shallow} + \hat{\mathcal{Q}}_{deep} \cdot \mathbf{m}_{deep} \\ &\mathbf{m}_{shallow} + \mathbf{m}_{deep} = 1\end{aligned} \tag{2}$$

One might think that having two generators (one for the upper part and one for the lower part) would lead to the same result. However, having 2 generators would increase the number of parameters which is undesired in real-time applications. Moreover, we show in Sec. 5.2 that 2 generators are suboptimal compared to the proposed method. Our height-aware generator namely has two advantages. First, the shallow layers of the backbone learn faster because of the gradient flow from the shallow upsampling branch. This is similar to having an auxiliary loss [25]. Second, by allowing the upsampling branches to predict the whole range image as opposed to only one part of it, we allow the branches to correct each other. The masks can thus be interpreted as the upsampling layer's confidence map: low value indicate high epistemic uncertainty and would weight down the

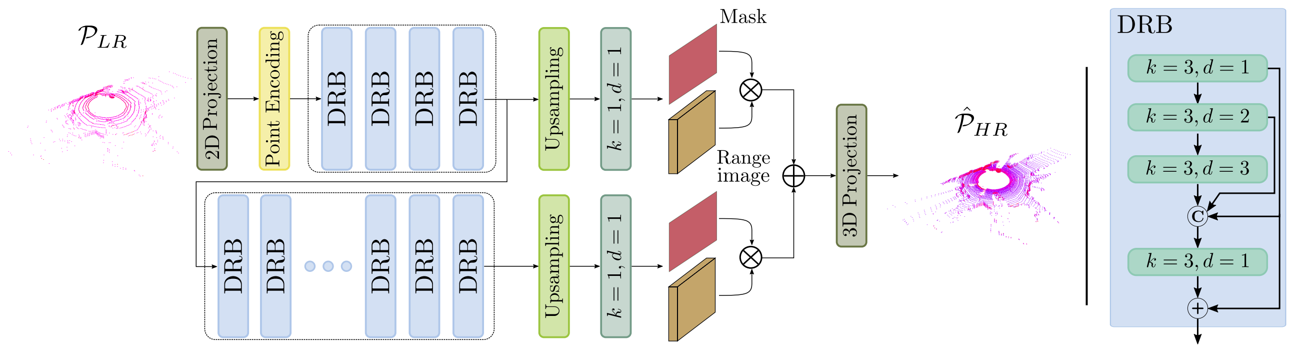

Figure 3. *Left*: The proposed generators architecture. We upsample the pointcloud at two locations in the backbone, intentionally chosen to have a local and a global receptive field. Both outputs are fused using confidence maps from each branch. *Right*: The architecture of the DRB used in the backbone to provide flexible receptive fields.

branch's prediction. This would not be the case if we have 2 separate generators, as each generator will only learn on some part of the range image, leading to a weaker training signal. Finally, to allow for more flexible receptive fields, we choose to replace normal residual blocks inside the backbones by the DRBs, inspired by [16]. The dilated convolutions allow to capture coarse and fine details at various receptive fields. We include the block design in Fig. 3.

4.3. Range image with polar coordinates

An assumption often made in the spherical projection (Eq. (1)) is that all lidar beams have equal angular spacing in the vertical direction. This assumption does not always hold, as it depends on the sensor type. The sensor's vertical resolution may vary with the absolute value of the elevation angle. Some sensors exhibit a dense vertical angular resolution in the middle and a sparse resolution in the upper and lower part of the range image. However, the inverse of the spherical projection leads to quantization errors when computing the absolute height of the points, $z = rsin(\theta)$, because it assumes that all elevation angles θ are regularly spaced. We argue that it is more beneficial for the network to directly observe and learn the polar or cartesian coordinates, similar to the point-based operations, in order to reduce this vertical quantization error. To this end, we represent the lidar scene as a range image with polar coordinates ($\mathcal{Q}(u, v) = (d, z)$), as they empirically show a lower generalization error than cartesian coordinates. This representation is used for the input and output range image features ($C = 2$). Note that when projecting the range image back to 3D space, the inverse of Eq. (1) is only applied for (x, y) using the polar range d and assuming equally spaced azimuth angles from u, while z value is taken from the upsampling branches and is not estimated from v. By directly estimating the z value for each point in the final layer, the generator can better match the real point distribution in the vertical direction.

4.4. Surface Normal Loss

Previous works have mostly used the $\mathcal{L}_1$ loss with respect to the ground truth to train the network. However, the $\mathcal{L}_1$ loss is not sensitive to the high-frequency details in the range image and might lead to noisy object boundaries. To regularize the training, we draw inspiration from monocular depth estimation (MDE), which is the task of estimating a depth map from a camera image. Although MDE is very different from lidar upsampling, we draw a connection between the two fields: both estimate depth values on a 2D plane. In MDE, it has been shown [9] that the $\mathcal{L}_1$ loss does not sufficiently penalize the shift in the estimated edges. To compensate for these shortcomings, some MDE works [9, 35] estimate the surface normal of the estimated depth map and penalize its deviations from the surface normal of the ground truth. We adopt a robust variant of the surface normal loss, called the virtual normal loss ($\mathcal{L}_{VNL}$) [35] in addition to the used $\mathcal{L}_1$ loss. Specifically, K groups of 3 non-colinear points are sampled from each pointcloud and the normal vector to the plane formed by triplet is computed. The final loss is a combination of $\mathcal{L}_1$ and $\mathcal{L}_{VNL}$. Let n_k be the normal vector estimated from a group of points in the ground truth, and $\hat{n}_k$ be the normal vector estimated from $\hat{\mathcal{Q}}$ from the same group of points. Note that we use the $\mathcal{L}_1$ loss on the polar coordinates (d, z). The final loss used for training can be expressed as:

$$\begin{aligned}\mathcal{L} = &\frac{1}{HWC} \sum_{i,j,c}^{H,W,C} || \hat{\mathcal{Q}}_{HR}(u, v, c) - \mathcal{Q}_{HR}(u, v, c) ||_1 \\ &+ \frac{1}{K} \sum_{k=1}^{K} || \hat{n}_k - n_k ||_1\end{aligned} \tag{3}$$

5. Experiments

In this section, we demonstrate that the proposed approach achieves improved generation quality on 3 real-world autonomous driving datasets.

5.1. Experimental Setup

We conduct experiments on 3 datasets: KITTI Raw which was previously introduced, KITTI Object [8] and nuScenes [3]. The lidar sensor used in KITTI Object and KITTI Raw dataset is the Velodyne HDL-64E which has 64 beams and a vertical field of view of $26.8°$, while nuScenes uses Velodyne HDL-32E with 32 beams and a vertical field of view of $40°$. AS previously mentioned, the number of points is reduced in KITTI Raw so that the range image size is 40×256. Larger resolutions are used in KITTI Object and nuScenes, where the ground truth resolution is 64×700 for the first and 32×1024 for the second. We choose these resolutions to be able to calculate the EMD metric, as larger pointcloud sizes do not fit into the used GPU (GeForce RTX 2080 Ti). We use a train/validation/test of $(3612/100/3769)$ on KITTI Object, $(40k/80/700)$ on KITTI Raw and $(27k/1k/6k)$ on nuScenes. A batchsize of 32 is used for KITTI Raw, while a batchsize of 24 is used for KITTI Object and nuScenes. We use the ADAM optimizer with an initial learning rate of 0.0001 that is decayed by a factor of 0.5 every 40 epochs on nuScenes and KITTI Raw and every 80 epochs on KITTI Object.

To evaluate the upsampling quality, we use the EMD and CD. Moreover, we add 6 metrics that were used in previous works. We calculate the MAE and RMSE between the generated and ground truth range images. MAE and RMSE were used in previous works on grid-based lidar upsampling [10, 11, 23, 27]. Similar to ILN [11], generated and ground truth pointclouds are voxelized (using a voxel size of $0.1m \times 0.1m \times 0.1m$). If one or more points fall inside a voxel, it is assigned a value of 1, else it is assigned 0. Then, we calculate the Intersection-over-Union (IoU), the Precision, Recall and F1-score with respect to the ground truth. These 4 metrics measure the 3D alignment of the generated pointcloud with the ground truth in a *coarser* way than EMD and CD (which measure the difference between point distributions). They indicate how much the structure of the pointcloud is similar to the high-resolution real pointcloud.

5.2. Ablation studies

To showcase the importance of our contributions, we ablate different parts of our model. We make an ablation study on the generator design on the nuScenes dataset ($2\times$ upsampling rate) in Tab. 3. In the supplementary material, we include an incremental component analysis on KITTI Raw dataset ($4\times$ upsampling rate) and show a visualization of the generated masks ($\mathbf{m}_{shallow}$ and $\mathbf{m}_{deep}$).

Ablation on the Generator Design. In Tab. 3, we study several design choices in the generator design on the nuScenes Dataset ($2\times$ upsampling rate). Note that the nuScenes dataset is harder than KITTI Raw and KITTI Object, as it has fewer lines, more sparse regions in the range image and a wider vertical field of view. We start with our baseline, SRResNet [13] with $\mathcal{L}_1$-loss only in Configuration 0. Then, we add polar coordinates, DRBs, and $\mathcal{L}_{VNL}$ and we refer to this model as generator (A). In Configuration AA, we replace the baseline in configuration 0 with the HALS generator and keep the spherical coordinates and the $\mathcal{L}_1$-loss only. Both these models show clear improvements relative to the Baseline. In configuration B, we use 2 generators (A), one on the upper part of the range image (from row 0 to 3) and one on the lower part only (from row 4 to 15). Both generators have 16 DRBs. We notice an improvement in most metrics, confirming our hypothesis that different parts of the range image need different upsampling models as they exhibit different properties. In configuration C, we reduce the receptive field of the generator for the upper part (4 DRBs only), but we notice a slight decrease in 3D metrics compared to configuration B. This could be attributed to the low model capacity of the upper generator. Then in configurations D, E and F, we try different settings for our height-aware generator. Namely, in configuration D, we place 3 upsampling branches instead of 2, after blocks number 4, 8, and 16. In configuration E, we place 2 branches at blocks 12 and 16, while in configuration F, they are placed at blocks 8 and 16. Finally, we show the proposed model, which features 2 branches at blocks 4 and 16 in configuration G. Specifically, HALS has a superior performance compared to 2 generators in configurations B and C, while having fewer parameters. Moreover, placing the first upsampling branch after a small number of blocks shows superior performance than placing it after a larger number of blocks (configurations E and F), highlighting the importance of the receptive field as a design parameter.

5.3. Main Results

We compare the proposed approach with the state-of-the-art grid-based models in Tab. 4 and Tab. 5 on 3 datasets, with different resolutions and upsampling rates. We include all grid-based baselines and we add to them the SWIN-IR [15], as it is a state-of-the-art image super-resolution model. HALS outperforms the baselines on the majority of metrics, sometimes by a significant margin (especially in EMD, CD and IoU). Note that 2D metrics are less important than 3D, since the pointcloud lies in the 3D space. For instance, SWIN-IR has the lowest RMSE and MAE on KITTI Object and KITTI Raw but this does not translate to the best performance in 3D metrics. On nuScenes, ILN has a good performance on IoU, Precision, Recall and F1-score but bad performance on EMD and CD. On the other hand, SWIN-IR is better on EMD and CD but lacks behind in IoU and F1-score. The proposed model achieves strong results on all 6 3D metrics simultaneously. Object detection and qualitative results are added in the supplementary material.

Config.	Model	EMD ↓	CD ↓	MAE ↓	RMSE ↓	IoU ↑	Precision ↑	Recall ↑	F1-score ↑
0	Baseline [13]	388	0.231	0.82	4.97	0.317	0.467	0.493	0.48
A	Baseline [13] w/ + polar + $\mathcal{L}_{VNL}$	365	0.20	0.74	4.68	0.457	0.614	0.639	0.626
AA	HALS generator + spherical w/o $\mathcal{L}_{VNL}$	357	0.192	0.71	4.77	0.415	0.576	0.595	0.585
B	2 generators (A) same receptive field	349	0.179	0.71	4.81	0.491	0.648	0.666	0.657
C	2 generators (A) diff. receptive field	368	0.189	**0.69**	**4.67**	0.480	0.639	0.665	0.652
D	(A) w/ 3 upsampling layers at (4,8,16)	344	0.180	**0.69**	4.70	0.487	0.644	0.664	0.654
E	(A) w/ 2 upsampling layers at (12, 16)	351	0.186	0.71	4.73	0.471	0.626	0.654	0.640
F	(A) w/ 2 upsampling layers at (8, 16)	343	0.178	**0.69**	4.70	0.493	0.653	0.668	0.661
G	(A) w/ 2 upsampling layers at (4, 16) - HALS	**338**	**0.171**	**0.69**	4.72	**0.505**	**0.664**	**0.676**	**0.670**

Table 3. Ablation study on the generator design performed on the nuScenes Dataset, with x2 upsampling rate.

Model	EMD ↓	CD ↓	MAE ↓	RMSE ↓	IOU ↑	Precision ↑	Recall ↑	F1-score ↑	EMD ↓	CD ↓	MAE ↓	RMSE ↓	IOU ↑	Precision ↑	Recall ↑	F1-score ↑
	KITTI Object Dataset 2x Output Resolution: 64×700								KITTI Object Dataset 4x Output Resolution: 64×700							
Bilinear	596	0.22	0.81	2.80	0.164	0.274	0.284	0.279	834	0.27	1.3	3.94	0.11	0.181	0.203	0.191
LIDAR-CNN [27]	193	0.043	0.283	2.68	0.394	0.567	0.562	0.564	390	0.105	0.48	2.98	0.256	0.4	0.41	0.411
LIDAR-SR [23]	279	0.042	0.628	4.47	0.253	0.399	0.407	0.403	757	0.11	1.11	5.59	0.277	0.447	0.419	0.432
SWIN-IR [15]	212	0.057	0.303	**2.47**	0.221	0.365	0.358	0.362	391	0.105	**0.44**	**2.81**	0.376	0.537	0.554	0.545
ILN [11]	233	**0.038**	0.347	2.73	0.567	0.712	0.73	0.723	629	0.101	0.49	2.97	0.336	0.501	0.504	0.502
Ours	**189**	**0.038**	**0.273**	2.67	**0.581**	**0.738**	**0.735**	**0.734**	**369**	**0.09**	**0.44**	3.01	**0.402**	**0.567**	**0.573**	**0.57**
	nuScenes Dataset 2x Output Resolution: 32×1024								nuScenes Dataset 4x Output Resolution: 64×700							
Bilinear	595	0.89	1.53	5.31	0.106	0.181	0.201	0.19	793	1.21	2.81	7.62	0.056	0.099	0.113	0.105
LIDAR-CNN [27]	388	0.231	0.82	4.97	0.317	0.467	0.493	0.48	572	0.66	1.34	**6.30**	0.142	0.236	0.258	0.247
LIDAR-SR [23]	514	0.209	1.39	6.77	0.2	0.34	0.325	0.332	871	0.63	2.59	9.47	0.133	0.251	0.22	0.234
SWIN-IR [15]	373	0.21	0.75	4.9	0.332	0.489	0.506	0.498	565	0.67	1.38	6.44	0.176	0.285	0.315	0.30
ILN [11]	383	0.198	0.73	4.82	0.501	0.656	**0.678**	0.667	585	0.61	1.45	6.49	0.251	0.385	0.417	0.4
Ours	**338**	**0.171**	**0.69**	**4.72**	**0.505**	**0.664**	0.676	**0.670**	**510**	**0.58**	**1.30**	**6.30**	**0.274**	**0.421**	**0.435**	**0.428**

Table 4. Quantitative comparison against state-of-the art grid-based lidar super-resolution methods.

Model	EMD ↓	CD ↓	MAE ↓	RMSE ↓	IOU ↑	Precision ↑	Recall ↑	F1-score ↑
	KITTI Raw Dataset 4x Output Resolution: 40×256							
Bilinear	173	0.110	0.62	1.30	0.097	0.177	0.174	0.176
LIDAR-CNN	101	0.052	0.19	0.86	0.393	0.564	0.564	0.564
LIDAR-SR	130	0.162	0.39	2.03	0.342	0.515	0.506	0.51
SWIN-IR	101	0.051	0.19	**0.85**	0.451	0.621	0.621	0.621
ILN	104	0.061	0.23	0.93	0.392	0.588	0.54	0.563
Ours	**82**	**0.015**	**0.17**	0.89	**0.510**	**0.672**	**0.671**	**0.671**

Table 5. Quantitative comparison against state-of-the art grid-based lidar super-resolution methods.

5.4. Discussion and Limits

The results of the benchmark that we presented in Sec. 3 are comprehensive, but they are not conclusive. Although current grid-based methods outperform point-based approaches, future works can focus on improving point-based methods. However, the biggest challenge would be how to make point-based methods suitable for real-time deployment as they require more memory and computation time. Scaling to a large number of points is still a challenge.

Lidar upsampling can be used to increase the performance of many downstream computer vision applications (object detection, point segmentation...) all while using low-resolution sensors. In this work, we do not explore these applications, as it is a topic that deserves to be addressed on its own. Therefore, we leave it to future works.

Another limitation of the proposed approach is that it learns a deterministic mapping: the generator can synthesize only one high-resolution pointcloud from the low-resolution input. Future works can focus on how to apply generative models like generative adversarial networks, flows or diffusion models to lidar upsampling in order to learn a conditional probability distribution.

6. Conclusion

In this work, we have benchmarked the performance of different point-based and grid-based methods in the lidar upsampling tasks on the KITTI Raw dataset. Our analysis revealed the superiority of grid-based methods due to their vertical receptive field which spans different beams. We have also shed light on a peculiar characteristic of the range images; namely, the range distribution varies with the beam height, starting with a high mean and standard deviation at the top, which gradually decrease towards the bottom. We have proposed a generator architecture to match this height-dependent range distribution. By assigning varying receptive fields to different vertical parts of the range image, the generator collects the necessary spatial information to upsample the scene while preserving its shape. We have also changed the network's input and output representation to polar coordinates to explicitly generate the points height information. Finally, we have adopted a surface normal loss to preserve the 3D structure. With the proposed contributions, HALS sets a new standard for lidar upsampling on 3 real-world challenging datasets. Extensive ablation studies were conducted to validate the soundness of our design choices.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 3

[2] Lucas Caccia, Herke van Hoof, Aaron C. Courville, and Joelle Pineau. Deep generative modeling of lidar data. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5034–5040, 2019. 3

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2, 7

[4] Xieyuanli Chen, Ignacio Vizzo, Thomas Läbe, Jens Behley, and C. Stachniss. Range image-based lidar localization for autonomous vehicles. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5802–5808, 2021. 1

[5] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:295–307, 2016. 2

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1

[7] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2463–2471, 2017. 3

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 2, 3, 7

[9] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1043–1051. IEEE, 2019. 6

[10] Younghwa Jung, Seung-Woo Seo, and Seong-Woo Kim. Fast point clouds upsampling with uncertainty quantification for autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7776–7782. IEEE, 2022. 1, 2, 3, 4, 7

[11] Youngsun Kwon, Minhyuk Sung, and Sung-Eui Yoon. Implicit lidar network: Lidar super-resolution via interpolation weight prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8424–8430. IEEE, 2022. 1, 2, 3, 4, 7, 8

[12] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1

[13] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017. 2, 4, 7, 8

[14] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: A point cloud upsampling adversarial network. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7202–7211, 2019. 1, 2, 3, 4

[15] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021. 2, 3, 4, 7, 8

[16] Zhidong Liang, Ming Zhang, Zehan Zhang, Xian Zhao, and Shiliang Pu. Rangercnn: Towards fast and accurate 3d object detection with range image representation. *ArXiv*, abs/2009.00206, 2020. 1, 3, 6

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2

[18] Wenjie Luo, Binh Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 1

[19] Andres Milioto, Ignacio Vizzo, Jens Behley, and C. Stachniss. Rangenet ++: Fast and accurate lidar semantic segmentation. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220, 2019. 1, 3

[20] C. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 2

[21] Jasmine Richter, Florian Faion, Di Feng, Paul Benedikt Becker, Piotr Sielecki, and Claudius Glaeser. Understanding the domain gap in lidar object detection networks. *arXiv preprint arXiv:2204.10024*, 2022. 1

[22] Artem Savkin, Yida Wang, Sebastian Wirkert, Nassir Navab, and Federico Tombari. Lidar upsampling with sliced wasserstein distance. *IEEE Robotics and Automation Letters*, 2022. 1, 2, 3

[23] Tixiao Shan, Jinkun Wang, Fanfei Chen, Paul Szenher, and Brendan Englot. Simulation-based lidar super-resolution for ground vehicles. *Robotics and Autonomous Systems*, 134:103647, 12 2020. 1, 2, 3, 4, 7, 8

[24] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 5

[26] Di Tian, Dangjun Zhao, Dongyang Cheng, and Junchao Zhang. Lidar super-resolution based on segmentation and geometric analysis. *IEEE Transactions on Instrumentation and Measurement*, 71:1–17, 2022. 2, 3

[27] Larissa T. Triess, David Peter, Christoph B. Rist, Markus Enzweiler, and J. Marius Zöllner. Cnn-based synthesis of realistic high-resolution lidar data. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1512–1519, 2019. 1, 2, 3, 4, 7, 8

[28] Larissa T. Triess, David Peter, Christoph B. Rist, and Johann Marius Zöllner. Scan-based semantic segmentation of lidar point clouds: An experimental study. *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1116–1121, 2020. 3

[29] Ignacio Vizzo, Xieyuanli Chen, Nived Chebrolu, Jens Behley, and C. Stachniss. Poisson surface reconstruction for lidar odometry and mapping. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5624–5630, 2021. 1

[30] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018. 2

[31] Huikai Wu and Kaiqi Huang. Point cloud super resolution with adversarial residual graph networks. In *31st British Machine Vision Conference 2020, BMVC*. BMVA Press, 2020. 1, 2, 3, 4

[32] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Transfer learning from synthetic to real lidar point cloud for semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2795–2803, 2022. 1

[33] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors (Basel, Switzerland)*, 18, 2018. 1

[34] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3d point set upsampling. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5951–5960, 2019. 1, 2, 3, 4

[35] Wei Yin, Yifan Liu, and Chunhua Shen. Virtual normal: Enforcing geometric constraints for accurate and robust depth prediction. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021. 6

[36] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2018. 1, 2, 3, 4

[37] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14489–14498, 2021. 1