Unsupervised confidence for LiDAR depth maps and applications

This paper has been accepted for publication at the
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, 2022. ©IEEE

Unsupervised confidence for LiDAR depth maps and applications

Andrea Conti, Matteo Poggi, Filippo Aleotti and Stefano Mattoccia
University of Bologna

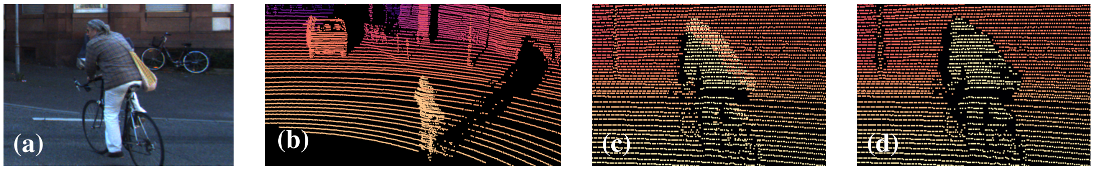

Figure 1. **Outliers filtering in LiDAR depth maps**. Given an image (a) and a LiDAR pointcloud (b), the projection of the latter over the image plane does not properly handle occlusions between the two points of view, assigning wrong depth values to the foreground (c). Our method (d) learns to remove these outliers reliably and without supervision.

Abstract— **Depth perception is pivotal in many fields, such as robotics and autonomous driving, to name a few. Consequently, depth sensors such as LiDARs rapidly spread in many applications. The 3D point clouds generated by these sensors must often be coupled with an RGB camera to understand the framed scene semantically. Usually, the former is projected over the camera image plane, leading to a sparse depth map. Unfortunately, this process, coupled with the intrinsic issues affecting all the depth sensors, yields noise and gross outliers in the final output. Purposely, in this paper, we propose an effective unsupervised framework aimed at explicitly addressing this issue by learning to estimate the confidence of the LiDAR sparse depth map and thus allowing for filtering out the outliers. Experimental results on the KITTI dataset highlight that our framework excels for this purpose. Moreover, we demonstrate how this achievement can improve a wide range of tasks.**

I. Introduction

Depth perception plays a crucial role in computer vision, enabling it to tackle tasks such as autonomous driving, object manipulation, and more. The 3D structure of a sensed environment can be inferred through passive and active sensing technologies. The former has been deployed for decades using stereo [41], [37], structure-from-motion [42] or multi-view-stereo [43]. Each of these methods has its flaws and constraints. For instance, stereo depth perception requires two calibrated cameras and struggles where the scene lacks texture. On the other hand, active sensing relies on specialized sensors, and in the case of LiDARs (Light Detection And Ranging), flooding the scene with a laser beam and computing the distance of each point by measuring the traveling time of the ray. Despite being accurate, LiDARs struggle, for instance, when sensing not Lambertian surfaces due to multi-path interference or subsurface scattering. Moreover, the resulting point cloud is sparse and not coupled with any visual information. Thus, it is common to jointly use it with a standard camera and project the LiDAR point cloud over the camera image plane, resulting in a sparse depth map. However, this procedure raises a fundamental issue due to the different points of view of the two devices and the intrinsic sparsity of the LiDAR's output. Specifically, it leads to wrong depth values in the final RGB-D image, as shown in Figure 1. Therefore, *LiDAR depth map filtering* is a serious problem to be tackled.

To date, LiDAR sensors are massively used to source ground-truth data in primary scientific datasets, such as KITTI [13], DrivingStereo [51], and many others [4], [47], [12], powering state-of-the-art deep learning techniques in computer vision. However, when projecting the depth map over the camera image plane, the issue mentioned above is usually tackled by enforcing consistency between the depth map and the values obtained through a stereo algorithm [17], [45] or deep stereo network [51]. Nonetheless, these approaches have some flaws as well: i) they require stereo cameras during acquisition with the LiDAR and ii) they do not filter out only the LiDAR errors, but also the stereo algorithm errors, thus affecting the cleaned data with the intrinsic limitations of the stereo setup (e.g., filtering out depth measures in textureless areas). Despite these limitations, these approaches are viable when pre-processing a dataset beforehand is feasible. However, they might not be applicable in real applications where a stereo setup is unavailable or when the depth labels for training are needed at runtime, for instance, to adapt stereo networks online [36].

To address all of these limitations, we propose a fast deep neural network framework, trained in an unsupervised manner, capable of predicting accurately the *uncertainty* of the projected LiDAR sparse depth map using a simple RGB-D setup. Such uncertainty, or complementary *confidence*, can be then deployed to filter out the errors, for instance, by enforcing a percentile to be removed or by using an absolute threshold. Therefore, the main novelties introduced by our work can be summarized as follows:

- We propose the first deep learning framework designed to compute the confidence of LiDAR depth maps
- We provide a peculiar supervision scheme enabling an unsupervised training of the model, thus not requiring any expensive ground-truth depth annotation

Moreover, experimenting over two splits of KITTI [13] i) we uphold our claims assessing the effectiveness of our method versus existing alternatives [53], [10], constantly outperforming even supervised techniques [10] and ii) we illustrate how filtering LiDAR depth maps with our approach yields consistent improvements in applications such as depth completion [27], [26], guided stereo [35] and sensor-guided optical flow [31] frameworks.

II. Related work

LiDAR sensors in computer vision. The massive diffusion of LiDAR sensors in computer vision has begun with the release of the KITTI dataset [13], an extensive collection of several thousand images and pointclouds acquired by a moving car equipped with a Velodyne LiDAR and stereo cameras. Eventually, more datasets followed this seminal work, such as DrivingStereo [51], Argoverse [4], Apolloscape [47] and DSEC [12]. In between, over the KITTI dataset many computer vision tasks have been tackled, such as LiDAR SLAM [46], [30], 3D object detection [9], [54], semantic segmentation [2], [5], object tracking [50], [23], 3D scene flow [40], [15], fusion of LiDAR measurement with stereo [35], [48], [6] or optical flow [31] and depth completion [45], [27], [26], [10], [18]. The latter is one of the iconic problems in this field and processes LiDAR pointclouds projected into depth maps over the image plane as input. The standard benchmark for this task is hosted by KITTI [45] and has been obtained in a semi-automatic manner from Velodyne raw data by accumulating multiple point clouds and filtering outliers/moving objects employing consistency with a stereo algorithm [17]. As a result, ground-truth maps of the completion benchmark miss several labels where stereo algorithm struggle.

Confidence estimation in computer vision. Estimating the confidence (or, complementary, the uncertainty) has been object of study in classical computer vision problems, such as optical flow [44] or stereo [41]. For the former task, we can distinguish between *model-inherent* methods [3], [25], [49], that are part of the flow estimation model itself, and *ad-hoc* approaches [28], [24] that process already estimated optical flow maps. For the latter task, a similar distinction can be found in the literature, with methods processing the cost volume [19] or already estimated disparity maps [38], [33]. In deep learning, uncertainty estimation has been studied as well [21] and applied to specific tasks such as optical flow [20], single image depth estimation [32], visual odometry [8], semantic segmentation of LiDAR pointclouds [7] and multi-task learning [22] as well. However, prior works, in general, require some supervision from additional data, such as ground-truth labels for the specific task over which uncertainty is modeled [20] or additional images in case of self-supervision [32]. In contrast, in our case, we source supervision from the raw input data alone.

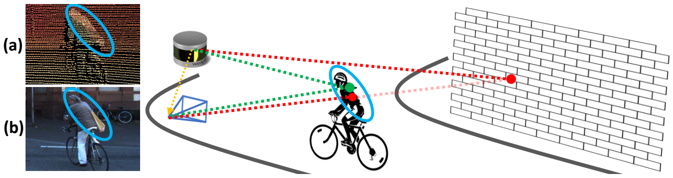

Fig. 2. **Outliers formation process due to occlusions.** When a LiDAR and an RGB camera acquire from different viewpoints, projecting the point cloud into a depth map (a) on the image (b) introduces outliers (blue oval), e.g. points visible by the LiDAR occluded to the camera (red), yet projected near foreground points visible to both (green).

LiDAR confidence estimation. Few works allow computing the confidence on the LiDAR sparse depth map itself. Indeed, most attention to confidence estimation was given to estimating the uncertainty of the depth predicted by the overall framework using the LiDAR data, as in the case of depth completion [11], [10]. To the best of our knowledge, only the following works estimate the confidence on the input LiDAR sparse depth map. Eldesokey et al. [10] were the first explicitly modeling such confidence, but this happens as a side effect of making their depth completion framework more robust versus input noise. Thus it requires a massive amount of labeled data (i.e., the whole KITTI dataset with accumulated ground-truth depth maps) to train the model for its primary task: performing depth completion. Zhao et al. [53] propose instead a depth completion method that does not rely on learning, by using the local surface geometry of depth points, and enhance their system by employing a binary outliers detection algorithm. Our solution differs from this latter since i) it is a learned method and ii) it generates confidence in place of a binary score, which allows for a finer outliers filtering mechanism, as we will see in our experiments. Finally, concerning LiDAR filtering through a stereo setup, we mention the work by Cheng et al. [6], filtering the outliers in LiDAR depth maps while performing a noise-aware fusion with stereo data by checking consistency with the output of the fusion itself.

III. Proposed approach

At first, we introduce the reasons which give rise to outliers in LiDAR depth maps, and then we describe our framework specifically designed to filter them out.

A. Outliers in LiDAR depth maps

There exist two leading causes of errors in LiDAR depth maps: i) erroneous measurements consequence of the LiDAR technology, for instance, originated by reflective or dark surfaces – over which the behavior of the emitted beams become unpredictable – or by other technological limitations (for instance, the mechanical rotation performed by the Velodyne HDL-64E used in KITTI [14]) and ii) incorrect projection of depth values near object boundaries, due to occlusions originated by the different viewpoints of the LiDAR sensor and the RGB camera.

Figure 2 provides an intuitive overview about the second issue: concerning an urban scene acquired by a LiDAR and a camera, with a cyclist in the foreground and a wall far in the background (example available in the KITTI dataset). The different position of the two sensors causes some background regions to be visible to one of the two while occluded to the

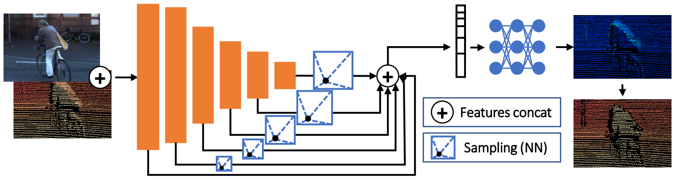

Fig. 3. **Proposed architecture.** A convolutional encoder (orange) extracts features at different resolutions. We query features for each pixel with a valid LiDAR value and concatenate them (+) in a vector, fed to an MLP (blue) to estimate confidence.

other. For instance, the red point on the wall is perceived by the LiDAR, but the cyclist occludes it in the image acquired by the camera. On the other hand, regions in the foreground are visible to both sensors, as the green point on the cyclist. When projecting LiDAR points into a depth map, specifically by mapping them over the camera image plane, depth values from occluded points in the background are projected into 2D pixel coordinates of foreground regions. The sparse nature of LiDAR points makes them visible in the resulting depth map shown in Figure 2 (a), labeling the RGB image acquired by the camera (b) with wrong depth values in regions occluding the background points sensed by the LiDAR.

This bleeding effect, agnostic to the sensor accuracy, occurs in all LiDAR-Camera setups, including the depth maps made available by the KITTI completion dataset [45], thus affecting the methods competing over the completion benchmark itself. Therefore, we introduce a carefully designed deep learning framework to deal with this issue, only using the RGB image coupled with the LiDAR depth.

B. Architecture

Figure 3 depicts the architecture designed to estimate confidence; it is composed of a multi-scale encoder and a prediction MLP (Multi Layer Perceptron) head. We feed the encoder with the concatenation of the RGB image and the sparse LiDAR depth map.

Features extraction. The encoder of the network consists of three 3×3 conv2D layers with 32, 64 and 64 output channels, followed by five encoding blocks made by a 2×2 MaxPool operator with stride 2 and two 3×3 conv2D layers having the same number of output channels, respectively 128, 256, 512, 512 and 512 for the five blocks. The encoder extracts features at full resolution from the first three conv2D layers, and at five more resolutions from the five aforementioned blocks, respectively at $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ and $\frac{1}{32}$. Extracting features at multiple scales allows increasing the receptive field and considering complex features from large areas in the image. This strategy, for instance, allows evaluating the shape of a large complex object such as a car to tell which depth measurements are outliers. Furthermore, the multi-scale extraction plays a crucial role since we apply an MLP head that does not consider the local information around each feature vector due to its inherent nature. Leaky ReLUs follow each convolutional/fully connected layer.

Confidence estimation. Once multi-scale features have been extracted, a sampling process occurs at each scale (using nearest-neighbor interpolation to sample at smaller scales) to compose a feature vector of size $64+128+256+512+512+512$ for each depth measurement contained in the LiDAR sparse depth map. The MLP head infers confidence by processing the high-level information regarding the image context and the original sparse depth distribution. Such an estimation comes in the form of variance, similarly to predictive uncertainty strategies [21] (the lower, the more confident). It is worth noting that a plain convolutional decoder could be used in place of an MLP. However, the specific task we are tackling does not require generating a dense output. Thus a simple MLP can estimate the confidence only for the meaningful pixels in the input depth map. We will show in Sec. IV how this approach steadily improves accuracy.

C. Unsupervised learning procedure

The following section describes our peculiar unsupervised training procedure, not relying on any ground-truth data.

Proxy labels generation. To train our model, we argue that nearby pixels should share similar depth values [34] except for points near discontinuities. Therefore, we take into account for each LiDAR depth point the other valid depth points inside a patch $P_N(x)$ of size $N \times N$ and compute a *proxy label* representing a plausibly correct depth for each original LiDAR depth value available:

$$d^*_x = f(\{d : d \in P_N(x),\ d > 0\}) \tag{1}$$

To speed up the training procedure and obtain a faster convergence, we use a fixed f function. Precisely, we extract the minimum depth among the valid depths contained in the patch. Another approach might be to use the average of the valid depths in the patch. However, in the presence of occlusions, this strategy would cause both background and foreground depths to be detected as outliers since both are far from the average depth occurring between the two. In contrast, using the minimum depth value correctly selects the foreground points as reliable in the presence of occlusions. As a drawback, it may lead to indiscriminately detecting as outliers most of the pixels in the background, even if not occluded. However, in practice, we will show that the network learns to ameliorate this issue and that the patch size affects the performance to a lesser extent. To support the effectiveness of our proposal, in Sec. IV we compare the behaviour of the network using the minimum, the average, and the KITTI ground-truth depth itself as proxy labels.

Loss function. We model the confidence of LiDAR depth d assuming a Gaussian distribution centred in the proxy label d^* with variance σ^2, the latter encoding the depth uncertainty. Thus, during training, the network learns to regress σ by minimizing the negative log-likelihood of the distribution.

$$\mathcal{L}_G = -\ln\left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(d-d^*)^2}{2\sigma^2}}\right) \tag{2}$$

Windows size	CV split [45]	142 split [29]	Average
5×5	0.1517	0.2291	0.1904
7×7	0.1318	**0.1975**	0.1647
9×9	**0.1292**	0.1985	**0.1639**
11×11	0.1316	0.1999	0.1658
13×13	0.1372	0.2055	0.1714

(a)

Sampled features	CV split [45]	142 split [29]
$\frac{1}{32}$	0.2436	0.3220
$\frac{1}{32}+\frac{1}{16}$	0.1941	0.2597
$\frac{1}{32}+...+\frac{1}{8}$	0.1680	0.2302
$\frac{1}{32}+...+\frac{1}{4}$	0.1486	0.2109
$\frac{1}{32}+...+\frac{1}{2}$	0.1362	0.2010
All	**0.1292**	**0.1985**

(b)

TABLE I. **Experimental results – ablation study.** We measure the impact of window size used to extract proxy labels d^* and multi-resolution sampling. We report AUC values on the KITTI CV [45] and 142 [29] splits. In each sub-table, best results are **bold** and second best underlined.

	Proxy	Head	Loss	CV split [45]	142 split [29]	Average
	d^*_{avg}	Decoder	$\mathcal{L}_L$	0.5286	0.8796	0.7041
	d^*_{avg}	Decoder	$\mathcal{L}_{L^*}$	0.2023	0.3730	0.2877
	d^*_{avg}	Decoder	$\mathcal{L}_G$	0.3692	0.5064	0.4378
	d^*_{avg}	Decoder	$\mathcal{L}_{G^*}$	0.1805	0.2403	0.2104
‡	d^*_{avg}	MLP	$\mathcal{L}_L$	0.5890	0.9624	0.7757
	d^*_{avg}	MLP	$\mathcal{L}_{L^*}$	0.1565	0.2649	0.2107
‡	d^*_{avg}	MLP	$\mathcal{L}_G$	0.2430	0.2811	0.2621
	d^*_{avg}	MLP	$\mathcal{L}_{G^*}$	0.1546	0.2197	0.1872

	Proxy	Head	Loss	CV split [45]	142 split [29]	Average
‡	d^*_{min}	Decoder	$\mathcal{L}_L$	0.5569	0.7641	0.6605
	d^*_{min}	Decoder	$\mathcal{L}_{L^*}$	0.1558	0.3693	0.2626
‡	d^*_{min}	Decoder	$\mathcal{L}_G$	0.8715	1.2760	1.0738
	d^*_{min}	Decoder	$\mathcal{L}_{G^*}$	0.1382	0.2548	0.1965
‡	d^*_{min}	MLP	$\mathcal{L}_L$	0.6457	1.1370	0.8914
	d^*_{min}	MLP	$\mathcal{L}_{L^*}$	**0.1267**	0.2446	0.1857
	d^*_{min}	MLP	$\mathcal{L}_G$	0.4801	0.6744	0.5773
	d^*_{min}	MLP	$\mathcal{L}_{G^*}$	0.1292	**0.1985**	**0.1639**

TABLE II. **Experimental results – ablation study.** We measure the impact of the three main design strategies in our unsupervised framework. We report AUC values on the KITTI CV [45] and 142 [29] splits. In each sub-table, we report the best result in **bold** and the second-best underlined. ‡ means 10^{-6} learning rate to avoid divergence.

We can rewrite (2) as follows:

$$\mathcal{L}_G \approx \ln(\sigma) + \frac{(d-d^*)^2}{2\sigma^2} \tag{3}$$

According to our experiments, (3) becomes unstable when $\sigma \ll 1$ since it leads to enormous loss values hampering the learning procedure. Therefore, we constrain the network output to be $\sigma \geq 1$, obtaining the following additional advantages. The regularization term $\ln(\sigma)$ is 0 when σ reaches its minimum value ($\sigma = 1$). Besides, small σ values no longer magnify $(d-d^*)^2$, an unwelcome event since the network aims to minimise this term as much as possible.

Nonetheless, by taking into account the derivative of (3)

$$\frac{d}{d\sigma}\mathcal{L}_G = \frac{1}{\sigma} - \frac{(d-d^*)^2}{\sigma^3} \tag{4}$$

and solving for the minimum, we obtain $\sigma_{min} = |d - d^*|$. Hence, to constraint the minimum of the loss function in the chosen domain (i.e. $\sigma \geq 1$), we also need to enforce $(d-d^*)^2 \geq 1$ in (3). Consequently, our final loss becomes:

$$\mathcal{L}_{G*} = \ln(\sigma) + \frac{(|d-d^*|+1)^2}{2\sigma^2}, \quad \sigma \geq 1 \tag{5}$$

In the next section we compare the performance of (3), with $\sigma \geq 1$, and (5) to measure the impact of this strategy.

Finally, it is worth observing that we might model uncertainty with other distributions such as the Laplacian, for which the previous observations still hold. Additional details are available in the **supplementary material**.

IV. Experimental results

We now assess the effectiveness of our framework in comparison with state-of-the-art. Source code is available at https://github.com/andreaconti/lidar-confidence.

A. Evaluation dataset and training protocol

We evaluate our framework on the KITTI dataset [14], a standard benchmark in the field providing both images and raw LiDAR depth maps obtained from 151 video sequences, as well as accurate ground-truth labels. Such annotation, based on semi-automatic procedures, is highly time-consuming and requires stereo images. For instance, the KITTI completion dataset [45] provides ground-truth maps obtained by accumulating 11 consecutive LiDAR pointclouds. Then, outliers (due to noise or moving objects) are removed by looking at inconsistency with respect to the output of the Semi-Global Matching (SGM) stereo algorithm [17]. This labeling strategy allows generating massive data (about 44.5K samples, 93K if considering stereo pairs) with the side-effect of losing several labels where LiDAR and SGM are not consistent. An even more accurate and laborious strategy consists of manually refining the labeling process, as done for the KITTI 2015 stereo dataset [29]. In this case, 3D CAD models have been used to obtain an accurate annotation for cars at the cost of much more effort (indeed, only 200 annotated samples are available).

In our experiments, we select two evaluation splits:

- **CV split**: composed of 1K images from the KITTI Completion Validation set [45]
- **142 split**: a subset of 142 images from KITTI 2015 overlapping with KITTI completion, thus providing both raw LiDAR depth maps and manually annotated ground-truth

We train models using the 113 video sequences that do not overlap with any of the two splits. Moreover, since our framework quickly converges, we need just a few samples to achieve state-of-the-art results; thus, we use a subset of about 6K (one every five frames) samples. Nonetheless, for a fair comparison, we retrain our supervised competitor [10] over the whole available training set, yet avoiding overlapping with the 142 split (over which the weights released by the authors have been trained on).

Our framework is trained for 3 epochs only, using the ADAM optimizer with a learning rate of 10^{-5}, with batches of 2 samples made of 320×1216 crops on a single NVIDIA RTX 3090. To train the models by Eldesokey et al. [10] we use the authors' code following the recommended settings.

B. Outliers detection

Evaluation metrics. We start by evaluating the performance of our method and existing approaches [10], [53] at detecting outliers in LiDAR depth maps. Purposely, we compute the Area Under the sparsification Curve (AUC), a standard metric for this task [20], [33], [32]. Namely, for each depth map in the dataset, pixels with both LiDAR and ground-truth depth available are sorted in increasing order of confidence score and gradually removed (2% each time). The

Method	CV split [45]	142 split [29]
Surface [53]	0.7014	1.4446
$\lvert d - d^*_{avg} \rvert$	0.2161	0.2641
$\lvert d - d^*_{min} \rvert$	0.2053	0.2665
Ours	**0.1292**	**0.1985**
Optimal	0.0271	0.0393

(a) unsupervised

Method	CV split [45]	142 split [29]
NCNN-Conf-L1 [10]	0.1530	0.3093
NCNN-Conf-L2 [10]	0.4624	0.8329
pNCNN-Exp [10]	0.8131	1.5430
Ours †	**0.1172**	**0.2094**
Optimal	0.0271	0.0393

(b) supervised

TABLE III. **Experimental results – outliers removal.** We report AUC values on the KITTI CV [45] and 142 [29], comparing with unsupervised (a) and supervised (b) methods. Best method per category in **bold**, second best underlined, absolute best in **red**. † means $d^* =$ ground-truth depth.

Root Mean Squared Error (RMSE) over remaining pixels is computed each time and a curve is drawn. The area under the curve quantitatively assesses the effectiveness at removing outliers (the lower, the better). Optimal AUC is obtained by removing pixels in decreasing order of depth error.

Ablation study. We first assess the impact of several factors in our framework. In Table I, we show the effect of the window size used to compute proxy labels d^* and multi-resolution features sampling on our final model. For these experiments, we assume (5) as loss function. Table I (a) shows that a 9×9 patch allows us to train our framework at its best, while models trained on proxy labels computed on smaller or larger windows gradually achieve worse results. Intuitively, tiny windows lead the network toward over-fitting on high confidence values (i.e., more LiDAR values are likely to be close to d^*). While using larger windows leads to the opposite behavior (i.e., most LiDAR values will have a high difference compared to d^* and drive, for instance, the network to predict low confidence in the presence of any depth discontinuity). Finally, Table I (b) reports that, not surprisingly, the best results are obtained by sampling features from any resolutions, from full to $\frac{1}{32}$.

Then, we measure the impact of the different design choices used to implement our model. Specifically: i) different proxy label generation functions (d^*_{min} and d^*_{avg} for respectively the minimum and the average among the valid depths in the patch), ii) the prediction head (MLP or a five layers decoder with skip connections, where each layer has two 3×3 convolutional blocks followed by 2×2 nearest-neighbor upsampling) and iii) the distribution function underlying the loss term between Gaussian $\mathcal{L}_G$, Laplacian $\mathcal{L}_L$ [21] and the modified version of both $\mathcal{L}_{G^*}$ and $\mathcal{L}_{L^*}$ as described in Sec. III-C. Table II collects results by several variants of our framework on both CV and 142 splits, using a 9×9 window and sampling features at all resolutions following the outcomes from Table I. The scores are generally lower on the CV split because of the many missing labels from ground-truth maps obtained semi-automatically, resulting in several outliers being missing in the AUC evaluation. We can also notice that $\mathcal{L}_{L^*}$ and $\mathcal{L}_{G^*}$ always outperform their original counterparts, assessing the quality of our formulation. Moreover, the synergy between the minimum proxy label strategy and the MLP yields the best results overall. Finally, even if both $\mathcal{L}_{L^*}$ and $\mathcal{L}_{G^*}$ are competitive, we choose $\mathcal{L}_{G^*}$ since it leads to the best overall results.

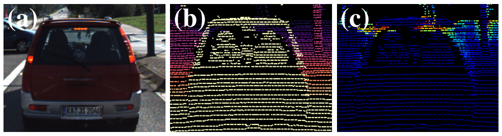

Fig. 4. **Qualitative results (142 split).** We show RGB images (a), raw LiDAR depth (b) and estimated confidence maps (c).

Filtering Method	RMSE (28.85% filtering)	% filtered (∼0.68 RMSE)
SGM [17]	0.6886	28.85
Ours	**0.2085**	**3.90**

Filtering Method	RMSE (20.08% filtering)	% filtered (∼0.92 RMSE)
Reversing [1]	0.9171	20.08
Ours	**0.2518**	**2.39**

Filtering Method	RMSE (1.47% filtering)	% filtered (∼1.42 RMSE)
Surface [53]	1.4189	1.47
Ours	**1.1841**	**0.97**

Filtering Method	RMSE (12.99% filtering)	% filtered (∼0.72 RMSE)
LiDARStereoNet [6]	0.7176	12.99
Ours	**0.3261**	**3.75**

TABLE IV. **Experimental results – semi-automatic annotation.** We evaluate the annotation performance of our method on the 142 split, either by fixing the % of filtered points or the final RMSE achieved by the competitors.

Comparison with state-of-the-art. Table III reports a comparison with existing approaches, namely Surface [53] and NCNN variants [10] on both splits, grouping unsupervised and non-learned methods on left (a) and supervised ones on right (b). Since our framework can be trained on ground-truth labels as well, we report this additional experiment, marked with †, to compare it with supervised methods directly. While this slightly improves the performance on the CV split, which is labeled with the same semi-automatic procedure of the training set, it leads to worse results on the accurate ground-truth maps of the 142 split. This outcome is not surprising since several outliers do not have a corresponding ground-truth value on KITTI CV and are never observed during supervised training over it. In contrast, as reported in the table, our unsupervised strategy is intrinsically unaffected by this bias.

Overall, our framework turns out the best approach, both when trained with and without ground-truth supervision. Indeed, even when trained in an unsupervised manner, it already outperforms supervised methods [10]. Moreover, we can notice that the absolute difference between LiDAR values and proxy labels is already a good cue to remove outliers, easily outperforming Surface [53] and often being better than supervised approaches [10]. Our framework learns to leverage such proxy labels and steadily exceeds their limitations, leading to even better results. Focusing on the former Table III (a), we can notice how [53] performs poorly at sparsification. Indeed, Surface performs a binary classification of inliers and outliers, removing only a small set of pixels (respectively 1.60% and 1.47% of the pixels with available ground-truth on CV split and 142 split), yet leaving many outliers on stage. Nonetheless, since a binary method is penalized by AUC evaluation, we will provide more fair comparisons in Sec. IV-C.

Concerning supervised methods in Table III (b), we interestingly notice that among NCNN variants, the one performing better at modeling uncertainty after completion

Model	LiDAR filtering	Removed points (%)	RMSE (mm)	MAE (mm)	iRMSE (1/km)	iMAE (1/km)
Self-Sparse-to-Dense [26]	None	None	1102.062	303.007	4.316	1.670
Self-Sparse-to-Dense [26]	Surface [53]	3.74	974.443	295.587	4.313	1.665
Self-Sparse-to-Dense [26]	Ours	1.20	**959.100**	**288.457**	**4.187**	**1.640**

(a)

Model	LiDAR filtering	Removed points (%)	RMSE (mm)	MAE (mm)	iRMSE (1/km)	iMAE (1/km)
Sparse-to-Dense [27]	None	None	676.061	274.109	3.097	1.705
Sparse-to-Dense [27]	Surface [53]	3.74	703.597	276.701	3.087	**1.689**
Sparse-to-Dense [27]	Ours	0.70	**647.473**	**270.554**	**3.060**	1.695

(b)

Model	LiDAR filtering	Removed points (%)	RMSE (mm)	MAE (mm)	iRMSE (1/km)	iMAE (1/km)
PENet [18]	None	None	593.196	178.869	2.242	0.940
PENet [18]	Surface [53]	3.74	616.753	182.139	2.285	0.943
PENet [18]	Ours	0.70	**569.449**	**177.057**	**2.223**	**0.936**

(c)

TABLE V. **Experimental results – Depth Completion.** Results on CV split by different completion models processing LiDAR filtered according to different strategies. Range: 50m.

according to [10], i.e. pNCNN-Exp, is the worst at detecting outliers in the input. On the contrary, NCNN-Conf-L1 is the best variant on raw LiDAR – although always outperformed by our approach, either supervised or unsupervised.

The superior accuracy achieved by our model comes at the cost of slightly higher complexity. Our network counts 16M weights versus the 300K of NCNN variants [10], leading to higher runtime on both 3090 and Jetson TX2 GPUs – respectively 0.02 and 1.02 seconds by our model versus 0.01 and 0.31 required by NCNN variants [10]. However, our model achieves better results and does not require any ground-truth depth label for training. For completeness, we also report the runtime required by Surface [53]. Although implemented on CPU, thus not directly comparable with our method and NCNN, [53] takes, respectively, 0.43 and 2.63 seconds on the same desktop PC equipped with a 3090 GPU – and an i9-10900X – and the Jetson TX2 CPU.

Figure 4 shows qualitative examples of confidence maps estimated by our unsupervised framework. More results are available in the **supplementary material**.

C. Applications

Finally, we evaluate how our unsupervised model impacts some relevant applications making use of LiDAR data.

Semi-automatic annotation. The first direct application of our strategy consists of filtering LiDAR depth maps to obtain accurate, per-pixel depth annotations. Semi-automatic processes [45] usually rely on an external stereo setup and check for consistency between LiDAR values and disparity maps. We compare our unsupervised model to this approach, either using a hand-crafted algorithm [17] or state-of-the-art self-supervised stereo networks [1], Surface [53] and a LiDAR-stereo fusion framework [48]. The comparison is performed on the 142 split since it provides manually annotated and refined ground-truth, in contrast to the CV split obtained semi-automatically. We limit to single depth map filtering and do not accumulate pointclouds over time to avoid issues with moving objects. When filtering using stereo methods, we convert LiDAR depth into disparity and filter pixels having a difference with the stereo disparity > 1.

In Table IV, we report a sub-table for each competitor, measuring the percentage of pixels with both available LiDAR and ground-truth values that are discarded, as well as the filtered RMSE. The RMSE without filtering is 2.5698 meters. Since the four competitors rely on a binary criterion to remove outliers, we both commit to i) remove the same amount of pixels they do and prove that our framework better reduces the error, ii) reduce the RMSE to the same value as our competitors and prove that our framework can achieve such an error by removing fewer points. Thus, in each comparison, we respectively i) remove the same percentage of the competitor and measure our final RMSE (first column), ii) filter pixels as long as we get the same RMSE of the competitor and measure the % of pixels we removed to obtain it (second column). Our method consistently achieves a much lower error when filtering the same percentage of pixels as the competitors. Moreover, it can reach the same final RMSE by removing a fraction of points, i.e. about 7-8 times less compared to stereo methods [17], [1] yet not requiring two cameras as they do. Moreover, by committing to a single fixed threshold as one would do in a real application – e.g., by constantly removing only 5% pixels – our model outperforms all the competitors, with 0.5850 RMSE. Finally, we can notice how leveraging stereo matching generally removes a high percentage of points (20-30%) because of the several regions where stereo methods struggle, such as occlusions or untextured regions. In contrast, Surface [53] removes very few points but yields a significantly higher RMSE.

Self-supervised/supervised depth completion. We now show how filtering outliers improves the performance of networks for depth completion, the most iconic task performed starting from LiDAR depth maps, without specifically re-training neither our framework nor the depth completion network. Following [53], Table V shows results achieved by the Sparse-to-Dense framework – using the weights released by the authors trained either without (a) [26] or with (b) [27] supervision – when processing inputs that have been filtered through unsupervised techniques like ours and Surface [53]. We compute standard depth completion metrics, such as RMSE and Mean Absolute Error (MAE), inverse RMSE and inverse MAE on points up to 50m, to focus on the foreground objects (mostly affected by the outliers). Concerning the self-supervised variant (a), we can notice how filtering with Surface [53] improves all metrics by removing nearly 4% of the total pixels with available LiDAR values. Concerning our method, we can achieve a larger improvement by limiting this percentage to 1.20%, hinting that more precise filtering of the outliers, yet limited to fewer pixels, is more effective for the depth completion task. This is confirmed by experiments on the supervised variant (b): in this case, using Surface [53] only improves inverse metrics, while our method always improves all metrics by removing 0.70% pixels only, resulting slightly worse only in iMAE compared to Surface. Finally, we report (c) experiments with PENet [18], a state-of-the-art framework for supervised completion, which confirm the previous findings.

Guided Stereo Matching. We also measure the boost in performance of a sensor fusion framework combining passive stereo with LiDAR sensors filtering raw data with our method. Purposely, we choose the guided stereo framework

Model	LiDAR filtering	Removed points (%)	> 2 (%)	> 3 (%)	> 4 (%)	> 5 (%)	MAE (px)
PSMNet-ft-gd-tr	None	None	5.14	3.39	2.69	2.29	1.08
PSMNet-ft-gd-tr	Surface[53]	4.37	4.75	3.01	2.34	1.95	1.01
PSMNet-ft-gd-tr	Ours	25.00	**4.60**	**2.80**	**2.13**	**1.75**	**0.94**

TABLE VI. **Experimental results – Guided Stereo.** Results on 142 split, with PSMNet weights provided by [35] and guided with LiDAR filtered according to different strategies.

Guide Source	LiDAR filtering	Removed points (%)	EPE (px)	Fl (%)	Density (%)
Ego +RIC +MaskRCNN [16]	None	None	0.80	2.35	3.16
Ego +RIC +MaskRCNN [16]	Surface [53]	4.37	0.74	2.35	3.06
Ego +RIC +MaskRCNN [16]	Ours	7.00	**0.73**	**2.17**	2.93

(a)

Model	LiDAR filtering	Removed points (%)	EPE (px)	Fl (%)
guided-QRAFT [31]	None	None	2.08	5.97
guided-QRAFT [31]	Surface [53]	4.37	2.07	5.98
guided-QRAFT [31]	Ours	7.00	**2.05**	**5.82**

(b)

TABLE VII. **Experimental results – Sensor-Guided Optical Flow.** Results on 142 split. (a) Accuracy of flow hints, obtained from LiDAR filtered according to different strategies, (b) accuracy of guided QRAFT [31] (CTK).

[35] (since it does not explicitly take into account noise, differently from [6]) and collect in Table VI the accuracy yielded by PSMNet-ft-gd-tr – the model provided by the authors – on the 142 split and report the percentages of pixels with error larger than 2, 3, 4 and 5, together with MAE as in [35]. While Surface [53] can slightly improve all metrics by removing less than 5% of the total pixels with available LiDAR value, by filtering a more significant amount of pixels with our method, up to 25%, we can further improve and achieve the best accuracy. Interestingly, the guided stereo framework has an opposite behavior with respect to depth completion, as it benefits more from strict filtering.

Sensor-Guided Optical Flow. In the final evaluation, we leverage our framework to improve the performance of the Sensor-Guided Optical Flow pipeline [31]. It combines flow hints sourced using a LiDAR sensor with a deep optical flow network by filtering LiDAR points before hints computation. Table VII collects both the accuracy of flow hints (a) and the final results achieved by QRAFT weights trained on Chairs, Things and KITTI (CTK), as provided by the authors of [31]. On top, we can notice how Surface [53] can reduce the flow end-point error (EPE) of the computed hints, yet it cannot reduce the number of outliers with error larger than 3 pixels or 5% (Fl). In contrast, by removing 7% least confident pixels, our proposal can effectively improve all metrics. At the bottom, we report the results achieved by guided QRAFT by using filtered hints. The impact of filtering is lower compared to other depth-related tasks. Indeed, in this task, LiDAR points are only one of several sources of errors, among camera pose estimation, flow estimation for dynamic objects and semantic segmentation. However, our method can consistently reduce both EPE and Fl metrics, whereas Surface cannot [53].

Current limitations. Despite its effectiveness, we can see two main limitations in our framework. First, it adapts to the specific RGB + LiDAR setup used for training. Nonetheless, this appears to be a minor limitation since the training procedure is fast and completely unsupervised. The second concerns the need to choose a confidence threshold for outliers removal, which is however a concern shared with most approaches in this field [20], [39], [32], [33], [52].

V. Conclusion

In this paper, we tackle the nowadays common problem of detecting outliers in LiDAR depth maps obtained by projecting the pointclouds over the image plane of an RGB camera. To deal with it, we have proposed an unsupervised framework trained solely on an RGB image plus the raw LiDAR depth map to estimate the LiDAR sensor confidence. Compared to existing methodologies [53], [10], it yields state-of-the-art performance. Moreover, it constantly improves relevant tasks relying on LiDAR depth maps, such as semi-automatic annotation, depth completion [27], [26], guided stereo [35] and sensor-guided optical flow [31].

Acknowledgment. This work was partially funded by University of Bologna and Ministero dello Sviluppo Economico (MISE) within the Proof of Concept 2020 program.

References

[1] Filippo Aleotti, Fabio Tosi, Li Zhang, Matteo Poggi, and Stefano Mattoccia. Reversing the cycle: self-supervised deep stereo through enhanced monocular distillation. In *European Conference on Computer Vision*, pages 614–632. Springer, 2020.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[3] Andrés Bruhn and Joachim Weickert. A confidence measure for variational optic flow methods. In *Geometric Properties for Incomplete Data*, pages 283–298. Springer, 2006.

[4] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] Ran Cheng, Ryan Razani, Yuan Ren, and Liu Bingbing. S3net: 3d lidar sparse semantic segmentation network. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[6] Xuelian Cheng, Yiran Zhong, Yuchao Dai, Pan Ji, and Hongdong Li. Noise-aware unsupervised deep lidar-stereo fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6339–6348, 2019.

[7] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In *International Symposium on Visual Computing*, pages 207–222. Springer, 2020.

[8] Gabriele Costante and Michele Mancini. Uncertainty estimation for data-driven visual odometry. *IEEE Transactions on Robotics*, 36(6):1738–1757, 2020.

[9] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[10] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Confidence propagation through cnns for guided sparse depth regression. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2423–2436, 2019.

[12] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 2021.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[15] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly Supervised Learning of Rigid 3D Scene Flow, 2021.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[17] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[18] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. In *ICRA*, 2021.

[19] Xiaoyan Hu and Philippos Mordohai. A quantitative evaluation of confidence measures for stereo vision. *PAMI*, 34(11):2121–2133, 2012.

[20] Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018.

[21] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[22] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weight losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[23] Aleksandr Kim, Aljoša Ošep, and Laura Leal-Taix'e. Eagermot: 3d multi-object tracking via sensor fusion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[24] Claudia Kondermann, Rudolf Mester, and Christoph Garbe. A statistical confidence measure for optical flows. In *European Conference on Computer Vision*, pages 290–301. Springer, 2008.

[25] Jan Kybic and Claudia Nieuwenhuis. Bootstrap optical flow confidence and uncertainty measure. *Computer Vision and Image Understanding*, 115(10):1449–1462, 2011.

[26] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3288–3295. IEEE, 2019.

[27] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4796–4803. IEEE, 2018.

[28] Oisin Mac Aodha, Ahmad Humayun, Marc Pollefeys, and Gabriel J Brostow. Learning a confidence measure for optical flow. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1107–1120, 2012.

[29] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.

[30] Yue Pan, Pengchuan Xiao, Yujie He, Zhenlei Shao, and Zesong Li. Mulls: Versatile lidar slam via multi-metric linear least square. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[31] Matteo Poggi, Filippo Aleotti, and Stefano Mattoccia. Sensor-guided optical flow. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[32] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[33] Matteo Poggi, Seungryong Kim, Fabio Tosi, Sunok Kim, Filippo Aleotti, Dongbo Min, Kwanghoon Sohn, and Stefano Mattoccia. On the confidence of stereo matching in a deep-learning era: a quantitative evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[34] Matteo Poggi and Stefano Mattoccia. Learning a general-purpose confidence measure based on o(1) features and a smarter aggregation strategy for semi global matching. In *3DV*, pages 509–518. IEEE, 2016.

[35] Matteo Poggi, Davide Pallotti, Fabio Tosi, and Stefano Mattoccia. Guided stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 979–988, 2019.

[36] Matteo Poggi, Alessio Tonioni, Fabio Tosi, Stefano Mattoccia, and Luigi Di Stefano. Continual adaptation for deep stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[37] Matteo Poggi, Fabio Tosi, Konstantinos Batsos, Philippos Mordohai, and Stefano Mattoccia. On the synergies between machine learning and binocular stereo for depth estimation from images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[38] Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Quantitative evaluation of confidence measures in a machine learning world. In *ICCV*, pages 5228–5237, 2017.

[39] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[40] Rishav Rishav, Ramy Battrawy, René Schuster, Oliver Wasenmüller, and Didier Stricker. Deeplidarflow: A deep learning architecture for scene flow estimation using monocular camera and sparse lidar. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10460–10467. IEEE, 2020.

[41] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[42] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[43] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006.

[44] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2432–2439. IEEE, 2010.

[45] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 international conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017.

[46] Han Wang, Chen Wang, and Lihua Xie. Intensity scan context: Coding intensity and geometry relations for loop closure detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2095–2101. IEEE, 2020.

[47] Peng Wang, Xinyu Huang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[48] Tsun-Hsuan Wang, Hou-Ning Hu, Chieh Hubert Lin, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. 3d lidar and stereo fusion using stereo matching network with conditional cost volume normalization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[49] Anne S Wannenwetsch, Margret Keuper, and Stefan Roth. Probflow: Joint optical flow and uncertainty estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1173–1182, 2017.

[50] Hai Wu, Qing Li, Chenglu Wen, Xin Li, Xiaoliang Fan, and Cheng Wang. Tracklet proposal network for multi-object tracking on point clouds. In *IJCAI*, 2021.

[51] Guorun Yang, Xiao Song, Chaoqin Huang, Zhidong Deng, Jianping Shi, and Bolei Zhou. Drivingstereo: A large-scale dataset for stereo matching in autonomous driving scenarios. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[52] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1281–1292, 2020.

[53] Yiming Zhao, Lin Bai, Ziming Zhang, and Xinming Huang. A surface geometry model for lidar depth completion. *IEEE Robotics and Automation Letters*, 6(3):4457–4464, 2021.

[54] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.