LiDARTouch: Monocular metric depth estimation with a few-beam LiDAR

Florent Bartoccioni[a,b], Éloi Zablocki[a], Patrick Pérez[a], Matthieu Cord[a], Karteek Alahari[b]

[a] *Valeo.ai, 100 rue de Courcelle, 75017 Paris, France*
[b] *Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France.*

ABSTRACT

Vision-based depth estimation is a key feature in autonomous systems, which often relies on a single camera or several independent ones. In such a monocular setup, dense depth is obtained with either additional input from one or several expensive LiDARs, e.g., with 64 beams, or camera-only methods, which suffer from scale-ambiguity and infinite-depth problems. In this paper, we propose a new alternative of densely estimating *metric* depth by combining a monocular camera with a light-weight LiDAR, e.g., with 4 beams, typical of today's automotive-grade mass-produced laser scanners. Inspired by recent self-supervised methods, we introduce a novel framework, called *LiDARTouch*, to estimate dense depth maps from monocular images with the help of "touches" of LiDAR, i.e., without the need for dense ground-truth depth. In our setup, the minimal LiDAR input contributes on three different levels: as an additional model's input, in a self-supervised LiDAR reconstruction objective function, and to estimate changes of pose (a key component of self-supervised depth estimation architectures). Our LiDARTouch framework achieves new state of the art in self-supervised depth estimation on the KITTI dataset, thus supporting our choices of integrating the very sparse LiDAR signal with other visual features. Moreover, we show that the use of a few-beam LiDAR alleviates scale ambiguity and infinite-depth issues that camera-only methods suffer from. We also demonstrate that methods from the fully-supervised depth-completion literature can be adapted to a self-supervised regime with a minimal LiDAR signal.

1. Introduction

Accurately estimating depth in scenes is a prerequisite for a wide range of computer vision tasks, from computing semantic occupancy grid (Ng et al., 2020; Lee and Medioni, 2016) to object detection without labels (Koestler et al., 2020; Deng et al., 2017) and multi-modal unsupervised domain adaptation (Jaritz et al., 2020). In particular, autonomous systems require an acute spatial understanding of their surroundings to plan and act safely, and the capacity to estimate depth is central to achieving this (Zeng et al., 2019; Srikanth et al., 2019; Philion and Fidler, 2020). For such applications, two lines of approach exist to infer depth in a scene, depending on the available data: LiDAR-based completion and camera-only estimation methods. LiDAR-based depth *completion* methods produce depth maps from one or multiple *dense* LiDARs (e.g., 32 or 64 beams) (Xu et al., 2019; Tang et al., 2020; Jaritz et al., 2018; Park et al., 2020) and essentially interpolate the scene structure from the input signal. However, these approaches are so far unfit for automotive-grade settings, as they rely on expensive sensors — often costing more than a car alone — and require a rich supervisory signal for training, composed of 64-beam LiDAR point clouds densely accumulated over time at a very high acquisition cost. An alternative is explored by camera-only methods that predict dense depth maps with either stereo (Chang and Chen, 2018; Kendall et al., 2017) or monocular (Godard et al., 2017, 2019; Guizilini et al., 2020a; Casser et al., 2019a; Mahjourian et al., 2018; Wang et al., 2018; Zhou et al., 2017; Kuznietsov et al., 2017; Yin and Shi, 2018; Guizilini et al., 2020b) setups. These models address the task of depth *estimation* and, contrary to the depth completion setup, do not leverage LiDAR point clouds. While such methods are appealing, as they rely on much cheaper and versatile sensors, monocular approaches suffer from ambiguity in the map scale they produce: most of them can only generate *relative* depth maps, i.e., up to an unknown global scaling factor, which makes them unusable in a real-world setting.
Moreover, their predictions can be catastrophic for objects with no relative motion with respect to the ego-camera, e.g., vehicles in front, which are likely estimated at infinite depth (Zhou et al., 2017; Godard et al., 2019; Mahjourian et al., 2018; Wang et al., 2018; Yin and Shi, 2018; Guizilini et al., 2020a; Casser et al., 2019a). Lastly, they are critically impeded by low-light conditions (at night or indoors) and adverse weather (in heavy rain, dense fog or snow storm) (Gruber et al., 2019).

In this paper, we propose the LiDARTouch framework, where dense *metric* depth is estimated by combining a monocular camera with a *minimal* sparse LiDAR input (e.g., 4 beams). Our motivations to use a sparse LiDAR input are diverse. First,

Table 1: **High-level positioning of LiDARTouch vs depth estimation and depth completion methods.** Our LiDARTouch framework addresses critical weaknesses of self-supervision depth estimation approaches, while being cheaper and far more scalable than fully-supervised depth completion methods.

		Supervision		
Approach	Input	Depth regression	Photo. reconst.	Strengths (S) and Weaknesses (W)
Depth estimation	Image	No	Yes	S: scales well (self-supervised, very cheap sensor) W: relative depth, catastrophic estimations (moving objects)
Depth completion	Image and dense LiDAR	w.r.t. dense GT depth	No	S: metric depth, very good performance W: scales poorly (expensive sensors and GT annotations)
LiDARTouch (ours)	Image and few-beam LiDAR	w.r.t. few-beam LiDAR	Yes	S: scales well (self-supervised, cheap sensors) S: metric depth, good performance

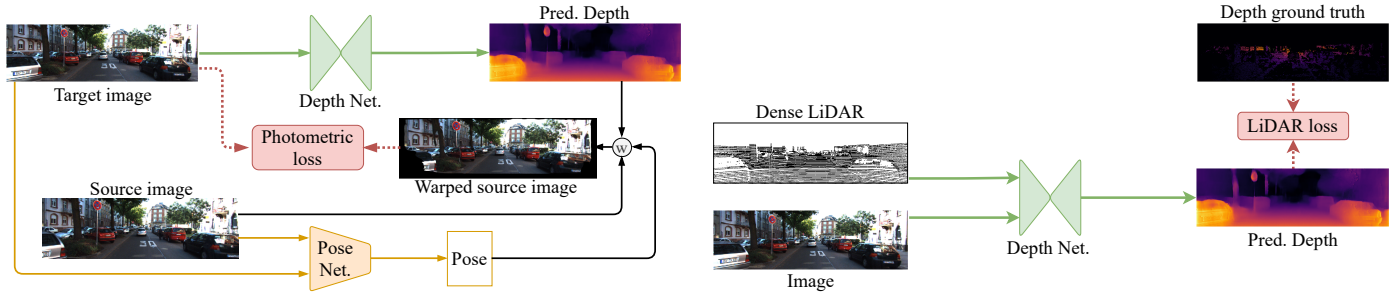
from a practical perspective, 4-beam laser scanners are currently embedded in consumer-grade vehicles and they are a hundred times less expensive than their dense (64-beam) counterparts. Second, we expect that such a LiDAR signal, although being extremely sparse, can provide valuable cues for monocular depth estimation, thus alleviating scale-ambiguity and infinite-depth problems. Third, we hypothesize that a light LiDAR touch will result in the overall model correctly estimating the depth of moving objects, notably cars, alleviating the infinite-depth issue. Finally, from a security perspective, such an approach makes it difficult to attack the camera signal alone (Yamanaka et al., 2020), due to a form of data redundancy between the camera and LiDAR.

Leveraging recent advances in monocular depth estimation (Zhou et al., 2017; Godard et al., 2019; Guizilini et al., 2020a; Watson et al., 2019), our approach is *self-supervised*. This setting is significantly less data-hungry than the fully-supervised alternative, which requires densified and stereo-filtered depth maps as ground truth (Fu et al., 2018; Xu et al., 2019; Tang et al., 2020; Jaritz et al., 2018; Park et al., 2020). We emphasize that this self-supervised learning setting, combined with the fact that it only involves widely available and low-priced sensors, makes the overall approach particularly scalable. Indeed, it becomes possible to estimate dense and metric depth maps on datasets and domains lacking depth ground truth (Chang et al., 2019; Sun et al., 2020; Caesar et al., 2019). Moreover, from an industrial perspective, the LiDARTouch framework naturally scales with the data acquired by a vehicle fleet without the need for any annotation. Under this new regime, we propose the adaptation of recent methods from the two aforementioned streams of approaches for inferring depth. On the one hand, we adapt fully-supervised depth completion methods, namely ACMNet (Zhao et al., 2021) and NLSPN (Park et al., 2020), to a much sparser LiDAR using our self-supervised setup. On the other hand, we strengthen the very embodiment of self-supervised monocular camera-only methods, namely Monodepth2 (Godard et al., 2019), to integrate the new complementary LiDAR information. We then perform an extensive study on the contribution brought by the sparse LiDAR signal at different levels as: (1) an additional input, (2) a new information source to estimate better poses, and (3) a form of self-supervision. A high-level positioning of LiDARTouch with respect to depth estimation and completion approaches is summarized in Table 1.

To evaluate the adapted models and validate our hypotheses, we propose a novel training and evaluation protocol on the KITTI dataset (Geiger et al., 2012) which includes the degradation of the raw 64-beam LiDAR data to obtain 4 beams. We also propose a new metric to quantitatively measure the infinite-depth problem. This allows us to verify one of our core hypotheses that the use of very limited LiDAR information corrects infinite-depth degeneracies of camera-only methods. In comparison to depth completion methods, our LiDARTouch framework overcomes the need for depth ground truth and leads to highly improved results with respect to approaches that are naïvely adapted to the self-supervised setting. In addition, we show that it is possible to successfully adapt architectures from the depth completion literature, as well as camera-based depth estimation methods, into a unified framework which alleviates problems from which these two lines of approaches suffer.

We make the following contributions:

1. We propose LiDARTouch, a new *self-supervised* depth estimation framework, where a *minimal* LiDAR and a monocular camera are available without access to any ground-truth depth annotations. This configuration is close to *in situ* conditions of today's vehicles, which is seldom addressed in other works.
2. We demonstrate that models trained within our LiDARTouch framework close the performance gap between self-supervised monocular depth estimation and fully-supervised depth completion learning schemes, proving that the need for ground-truth acquisition and costly sensors can be alleviated.
3. We show that models trained within our LiDARTouch framework do not suffer from critical scale-ambiguity and infinite-depth issues, in contrast to camera-only models. We evaluate this a novel metric to quantitatively measure the infinite-depth issue for the first time in the literature.
4. We demonstrate that LiDARTouch is a versatile learning framework by successfully applying it to a range of network architectures: Networks from the depth-completion literature are revamped to work with very sparse LiDAR instead of dense ones and camera-only models are adapted to integrate LiDAR data.
5. We study the influence of LiDAR inputs at each stage of our framework extensively. Our experiments show that integrating sparse LiDAR in a self-supervised scheme is not trivial. We provide key insights for the community on how the fusion scheme, the pose method and the supervisions interact.



(a) Self-supervised image-only depth estimation.

(b) Fully-supervised depth completion.

Fig. 1: **Illustration of the two paradigms for depth estimation.** (a) The left figure shows the classical learning system from self-supervised image-only depth estimation literature, e.g., SfMLearner (Zhou et al., 2017) or Monodepth2 (Godard et al., 2019). The model is trained to resynthesize the target image given (i) neighboring source images with different viewpoints, (ii) the estimated depth of the target image, and (iii) the relative change of pose between the target and source views. (w) denotes image warping given pose change and target depth map. (b) This figure summarizes the depth completion pipeline, e.g., models ACMNet (Zhao et al., 2021) or NLSPN (Park et al., 2020), which employs a multi-modal depth prediction network that is learned by regressing a provided ground-truth depth.

2. Background and related work

In the remainder of this paper we refer to a LiDAR as *dense* if it has more than 32 beams, and call it *sparse* or *minimal* otherwise. Depth ground truth, required by fully-supervised methods, is obtained from a dense LiDAR signal, accumulated over several sweeps. A camera stereo setup is then used to remove trail artifacts from moving objects. We will refer to such densified point-cloud data as *accumulated* LiDAR. These three density levels are illustrated in Figure 2. We now detail the two lines of approaches related to our work: camera-only monocular self-supervised methods and LiDAR-based fully-supervised depth completion systems.

Monocular self-supervised methods. In a fully- or semi-supervised setting, several models estimate depth in a camera-only monocular setup (Fu et al., 2018; Kuznietsov et al., 2017; Amiri et al., 2019), but acquiring depth ground truth for outdoor environments at scale is challenging and expensive. To overcome this issue, a few camera-based works (Godard et al., 2017; Zhou et al., 2017; Casser et al., 2019a) propose a *self-supervised* alternative to the use of ground-truth depth. Leveraging a set of consecutive frames, this paradigm predicts the depth for one of them and the relative changes in pose across nearby views. The model is trained by minimizing a photometric reconstruction error defined over these views (Figure 1a). Two important issues with such approaches hinder their widespread usage: the scale ambiguity of the produced depth maps and the infinite-depth problem.

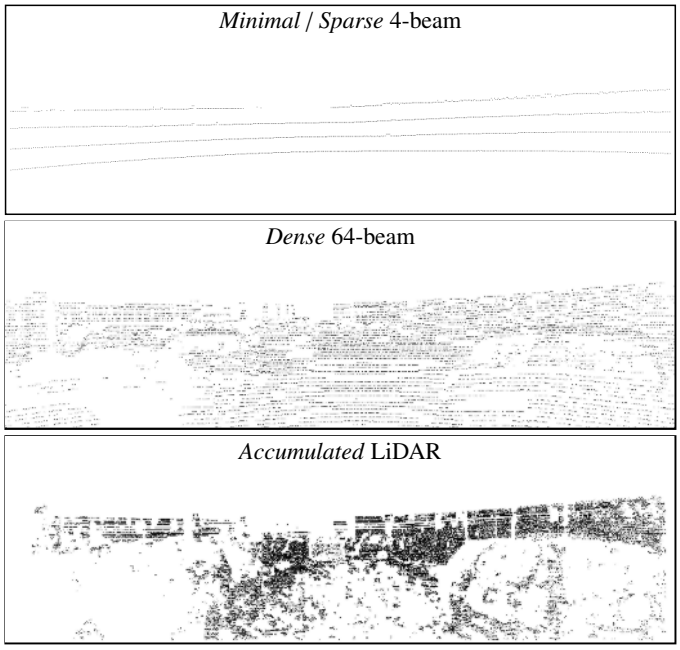


Fig. 2: **Different LiDAR densities.** *Dense* 64-beam point clouds are typically used as the input of depth completion approaches, which are supervised with *accumulated* LiDAR seen as ground truth (GT). These point clouds are far denser than the *minimal* LiDAR we use. Note that LiDAR data is often not available in the upper part of the scenes.

The *scale-ambiguity* problem stems from the view synthesis formulation being ill-posed. The formulation is scale ambiguous, as the target view can be correctly reconstructed regardless of the scale of the prediction. As a consequence, estimated depth maps are *relative* — up to an unknown global scaling factor — and models thus need additional supervision to accurately estimate a *metric* depth. Several self-supervised approaches rely on ground-truth LiDAR signal to scale their depth estimation at test time (Zhou et al., 2017; Godard et al., 2019; Casser et al., 2019a; Yin and Shi, 2018; Mahjourian et al., 2018; Wang et al., 2018). Alternatively, the recent PackNet model (Guizilini et al., 2020a) proposes to automatically scale estimations with additional constraints imposed by the instantaneous velocity of the ego-vehicle. Some works have also moved to a stereo setup to disambiguate the scale factor, using additional information, at train time only (Godard et al., 2017; Groenendijk et al., 2020) or also at run time (Chang and Chen, 2018; Kendall et al., 2017; Cheng et al., 2019), thus abandoning the monocular setup.

The second issue of infinite depth arises when objects move at the same speed as the camera. In this common situation, a trivial solution for the model is to predict that these objects are infinitely far and big, as they do not change in appearance through time (Zhou et al., 2017; Godard et al., 2019; Guizilini et al., 2020a). Recent proposals to address this problem exploit semantic segmentation of classes known to be often dynamic (e.g., cars, trucks) (Casser et al., 2019a,b), or automatically

prune the dataset by removing these objects (Guizilini et al., 2020b). The robustness of both these approaches to novel test scenarios, however, remains unclear.

In our work, we build on camera-only methods to additionally integrate LiDAR information and show that: (i) very few direct depth measures suffice to have a metrically-scaled dense depth estimation, and (ii) the infinite-depth issue can be partially or completely solved with the use of LiDAR input, depending on its resolution and position, without any additional assumptions.

Depth completion methods typically estimate a dense depth map from raw LiDAR measurements. Current deep-learning based methods for depth completion (Xu et al., 2019; Tang et al., 2020; Jaritz et al., 2018; Park et al., 2020; Ma and Karaman, 2018; Kumar et al., 2018; Zhao et al., 2021) usually learn to regress ground-truth depth maps in a fully-supervised setup (Figure 1b). Such approaches generally operate over RGB and LiDAR inputs.

A popular approach is to use one encoder per modality and fuse them at each resolution scale (Tang et al., 2020; Guizilini et al., 2021) or at the feature bottleneck only (Jaritz et al., 2018). An other option is early fusion, where both modalities are concatenated at the very begining of the architecture (Xu et al., 2019; Park et al., 2020; Ma et al., 2019) Some fusion module, as the one of GuideNet Tang et al. (2020), only considers the image as a guiding signal for the LiDAR features. This assumes that the LiDAR input is sufficient, i.e., high-resolution, for estimating depth, and thus unsuitable for our case. This limits the approach Tang et al. (2020) to estimate depth from high-resolution 64-beam LiDAR both at train and run time, making it incomparable to ours as we do not have access to such data. On the contrary, the SAN architecture (Guizilini et al., 2021), can handle various levels of LiDAR sparsity with sparse convolutions. Alternatively, networks like ACMNet (Zhao et al., 2021) and NLSPN (Park et al., 2020) propagate sparse LiDAR features into image features where depth measurements are not available. ACMNet (Zhao et al., 2021) uses a multi-scale co-attention-guided graph propagation strategy for depth completion. It propagates the sparse and irregularly distributed LiDAR measurements through a nearest-neighbor encoding. In addition, it uses a symmetric gated fusion strategy to fuse multi-modal contextual information throughout the decoder. The NLPSN architecture (Park et al., 2020) jointly estimates an initial depth map, a pixel-wise confidence and non-local affinity kernels. This initial depth map is iteratively refined with the input LiDAR features using the predicted confidence map and affinity kernels.

All the aforementioned depth completion methods employ a 64-beam input LiDAR and are trained with accumulated LiDAR as supervision. Here, most of the scene structure is available and the task amounts to color-guided depth interpolation. This design prevents these works from being easily adapted to new domains. Indeed, acquisition of ground-truth data is expensive and not scalable, as it is obtained from high-resolution LiDARs and stereo cameras. In contrast, our work specifically focuses on minimal 4-beam LiDAR directly, with no densely accumulated LiDAR data as supervision. We emphasize that in this very sparse 4-beam regime, almost no structural information can be directly extracted for the input signal. The task we propose is then more akin to depth estimation than depth completion.

A closely related work to ours is the model of Ma et al. (2019), which also uses LiDAR as a *supervisory* signal in a monocular self-supervised setting. LiDAR and camera signals are merged through an early fusion and the change of pose is estimated by solving a Perspective-n-Point problem. However, their setup is different to ours. Their study focus on the dense depth completion regime, i.e., with a 64-beam LiDAR, while we work on depth estimation with a minimal 4-beam LiDAR. Moreover, they do not compare against other existing architectures in the self-supervised setting. In contrast, we perform thorough evaluations with existing work by adapting camera-only and depth completion methods to our extremely scarce LiDAR regime. Additionally, we propose a different supervision scheme and the use of multiple views in photometric reconstruction. These choices lead to a substantial improvement on the KITTI dataset. Finally, we provide in-depth analyses on the impact brought by the LiDAR signal at different levels.

3. LiDARTouch framework

This section is organized as three parts, each corresponding to a different and complementary use of the light LiDAR signal. In Section 3.1, we present the architecture of the depth network, shown in green in Figure 3, which estimates depth by fusing the monocular image with the sparse LiDAR point-cloud. In Section 3.2, we detail the self-supervision objectives involving a photometric reconstruction along with a LiDAR self-supervision, as illustrated in red in Figure 3. Lastly, Section 3.3 introduces methods to estimate the relative change of pose between the source and target views, depicted by the orange part of Figure 3.

3.1. Depth network

The core of our depth estimation system is a neural network taking the target image I_t coupled with H_t, the LiDAR data projected in the image plane, as input, and predicting a depth map $\hat{D}_t$. Given the multi-modal nature of the input, our depth network employs a fusion strategy, that can be either early or multi-scale. In this paper, we consider four different architectures that are illustrated in Figure 4. Three of them are from the recent depth-completion literature, namely NLSPN (Park et al., 2020), S2D (Ma et al., 2019) and ACMNet (Zhao et al., 2021). The fourth one, we refer to as Monodepth2-L, is an extension of the camera-only model Monodepth2 (Godard et al., 2019) to operate over the additional LiDAR input (we provide details of this extension in Appendix A).

The two architectures NLSPN (Park et al., 2020) and S2D (Ma et al., 2019), illustrated in Figures 4b and 4a respectively, employ an early-fusion strategy, combining image and LiDAR features from the start, through concatenation. Early fusion directly mixes features from both modalities, thus potentially enabling richer interactions across them. The NLSPN architecture additionally re-injects the LiDAR signal at the end

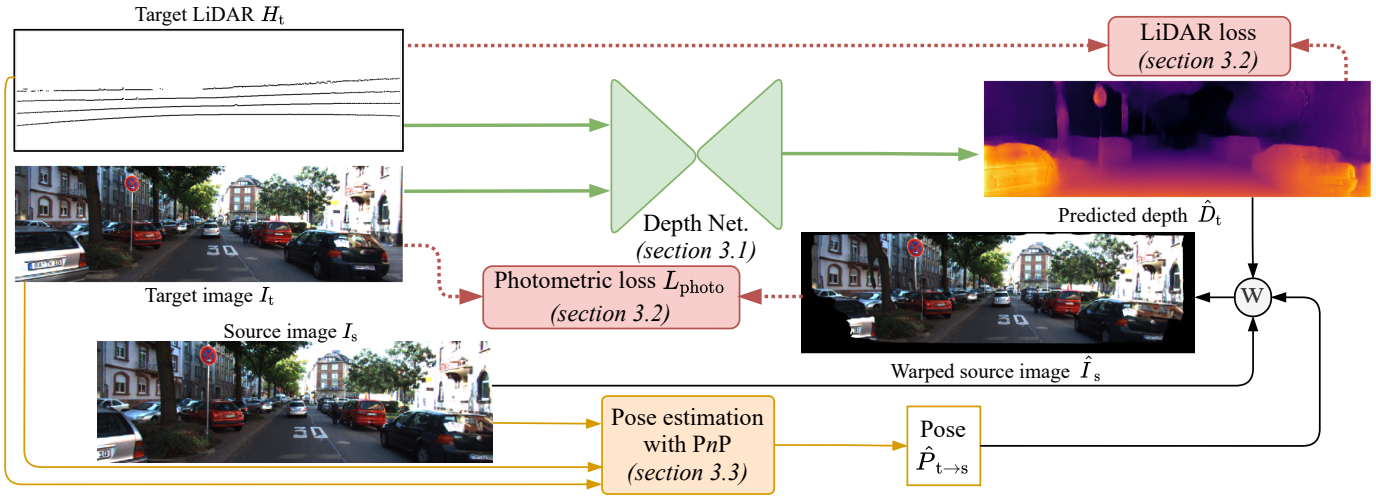


Fig. 3: **Overview of our LiDARTouch learning framework.** The proposed framework leverages ideas from both the camera-only depth estimation approach (illustrated in Figure 1a) and fully-supervised depth completion methods (illustrated in Figure 1b). In LiDARTouch, the light touch of LiDAR is integrated at three different stages: as an input of the depth network, as a self-supervision signal, and to estimate a scaled pose.

of the processing, as a late refinement strategy to mitigate signal degradation due to normalization layers.

In contrast, Monodepth2-L and ACMNet architectures, represented in Figure 4c and 4d respectively, use a multi-scale fusion. They both encode LiDAR and visual data separately so that these modalities are processed differently and their learned features are progressively integrated together. This design merges modalities more carefully than the early-fusion strategy, which is desirable as visual and LiDAR inputs carry complementary semantics. The two encoders, based on ResNet-18 (He et al., 2016), are independent and modality-specific features are fused with a series of concatenations. ACMNet, on the other hand, employs a more sophisticated co-attention strategy to mutually guide the features in the encoders and mix the features in the decoders to finally fuse them into one prediction.

3.2. *Self-supervision objectives*

Our challenging setting, where depth ground truth is unavailable for training the model, prevents the depth network architecture to be supervised directly. We address this by training the network under the supervision of two combined objectives. The first one, photometric reconstruction L_{photo}, is inspired by recent advances in self-supervised camera-only monocular depth estimation (Zhou et al., 2017; Godard et al., 2017, 2019). However, as discussed in Section 2, training with this objective alone leads to scale and infinite-depth issues. Consequently, we leverage a LiDAR self-reconstruction objective, which uses sparse yet complementary LiDAR information to mitigate these issues.

Self-supervised photometric reconstruction L_{photo}. We recall that the photometric reconstruction problem is a surrogate task aimed at resynthesizing a target image, given neighboring source images with different viewpoints (Zhou et al., 2017; Godard et al., 2019; Ma et al., 2019). Solutions to this task build on optimization approaches for disparity, motion and depth estimation without learning, based on photo-consistency. The central idea is to combine pose and depth predictions to project a neighboring source image into the target view. The underlying intuition is that to accurately resynthesize the target view from the source one, both the depth and pose estimation must be accurate.

Formally, the *target* image I_t is considered with a set S of *source* images I_s in its temporal vicinity. First, the depth network predicts the dense depth map $\hat{D}_t$ for the target image I_t. Second, the relative changes of pose $\hat{P}_{t\to s}$ between the target and source views are estimated — we detail this in Section 3.1. One pose transformation $\hat{P}_{t\to s} = \left(\begin{smallmatrix}\hat{R} & \hat{r}\\ 0 & 1\end{smallmatrix}\right) \in SE(3)$ is estimated for each source image $I_s \in S$, where $\hat{R}$ is a rotation matrix and $\hat{r}$ the translation component. Given the estimates of depth and pose, and the camera intrinsics K, a source image I_s can be warped via a differentiable geometric transformation into synthetic image $\hat{I}_s$ in the target view. More precisely, for homogeneous coordinates p_t of a pixel in the target image, the projected coordinates p_s in the source image are computed with:

$$p_s \simeq K\hat{P}_{t\to s}\hat{D}_t(p_t)K^{-1}p_t\,. \tag{1}$$

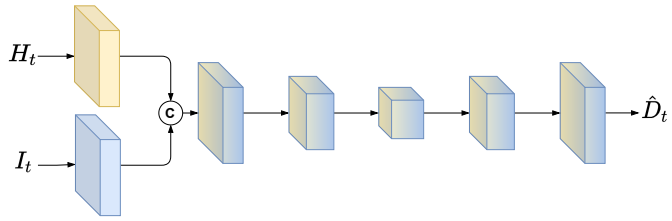
For a pair (I_s, I_t) of source-target images, the reconstructed image $\hat{I}_s$ is enforced to match the target image I_t by a pixel-wise image reconstruction error based on both an L_1 intensity loss and a structural similarity (SSIM) loss (Loza et al., 2006). Note that this formulation assumes Lambertian surfaces.

More formally, at a given pixel location p, this loss reads:

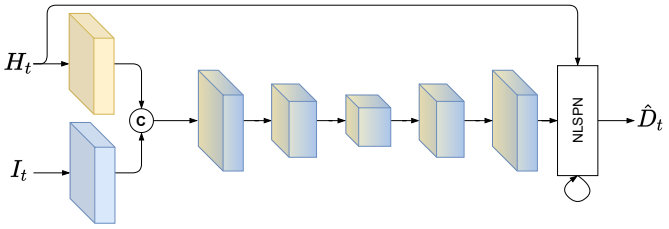
$$L_{photo}(p) = \min_{I_s\in S}\Big\{\frac{\alpha}{2}(1-\mathrm{SSIM}(I_t,\hat{I}_s)(p)) + (1-\alpha)\big|I_t(p)-\hat{I}_s(p)\big|\Big\}, \tag{2}$$

where α is a hyper-parameter balancing the contributions of the two terms. Moreover, taking the minimum value over all source images $I_s \in S$ limits the impact of errors resulting from occlusions and disocclusions in the scene due to motion of the ego-car and/or of the other scene elements (Godard et al., 2019). To take into account objects with no motion with respect to the ego-car, this loss is only applied to pixels whose appearance between frames varies (Godard et al., 2019).

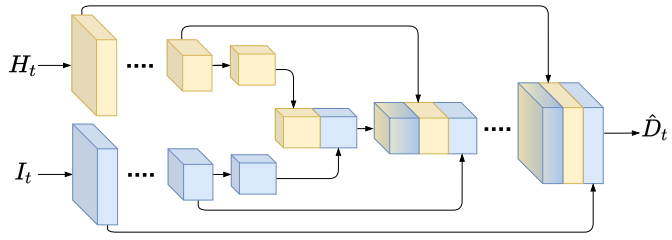
LiDAR self-supervision. As detailed in Section 2, a model solely trained with the photometric reconstruction loss L_{photo}



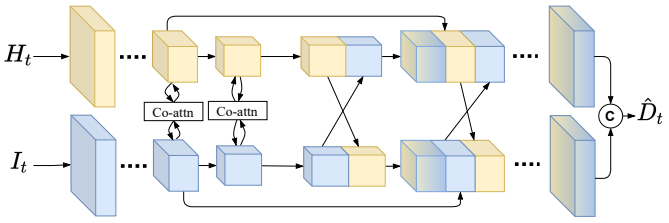
(a) S2D (Ma et al., 2019)



(b) NLSPN (Park et al., 2020)



(c) Monodepth2-L, extension of (Godard et al., 2019)



(d) ACMNet (Zhao et al., 2021)

Fig. 4: **Depth networks with different image-LiDAR fusion strategies.** We depict early (a), hybrid (early and late for b) as well as multiscale (c and d) fusion-based architectures. Volumes in yellow indicate LiDAR feature tensors and blue ones are image feature tensors. We indicate the mixing of modalities with a color grading of the two colors on the volumes. The architecture (c) is our extension of (Godard et al., 2019) to make it operate over *minimal* LiDAR input. We denote the concatenation operator by Ⓒ .

suffers from a scale-ambiguity issue and may be affected by the infinite-depth problem. In the following, we describe the new role of the low-density input LiDAR as a supervisory signal to mitigate this problem. We assume that this complementary information source can provide minimal-yet-crucial cues to disambiguate the estimated depth, at a global scale level and especially for moving objects. Furthermore, a sparse depth signal can refine the photometric supervision for small objects, thus improving overall performances (Watson et al., 2019). Inspired by the depth completion and the stereo depth estimation literature, we consider three different ways of using LiDAR as a supervisory signal: a straightforward L_1 regression along with two refinements that either control the interference with the photometric reconstruction or take into account the inherent noise of the LiDAR signal.

First, we consider a naïve self-supervision scheme, an L_1 loss for all pixels having a LiDAR measurement, in addition to the photometric loss L_{photo}:

$$L_{\text{naïve}}(p) = \begin{cases} |\hat{D}_t(p) - H_t(p)| + L_{\text{photo}}(p) & \text{if } H_t(p) > 0, \\ L_{\text{photo}}(p) & \text{otherwise,} \end{cases} \quad (3)$$

where p is an index over the pixels, $\hat{D}_t$ the estimated depth and H_t the input LiDAR projected in the target image plane. The latter being sparse, not all pixels have LiDAR data available; we use the encoding $H_t(p) = 0$ for such pixels.

Second, we consider the *masked* self-supervised objective proposed in Ma et al. (2019). It makes the LiDAR regression and the photometric loss exclusive by masking-out the photometric loss L_{photo} on pixels with a LiDAR measurement. Denoting L_{masked} as this loss, it is given by:

$$L_{\text{masked}}(p) = \begin{cases} |\hat{D}_t(p) - H_t(p)| & \text{if } H_t(p) > 0, \\ L_{\text{photo}}(p) & \text{otherwise.} \end{cases} \quad (4)$$

This loss is similar to $L_{\text{naïve}}$ but avoids potential conflicts between the photometric and LiDAR reconstructions.

Lastly, inspired by Watson et al. (2019), we also introduce the *hinted* self supervision, L_{hinted}, that takes into account the inherent noise of the LiDAR signal. Despite being a direct depth measurement, raw LiDAR signal is noisy for a number of reasons, including potentially imprecise calibration, approximated projection, and the fact that the camera and LiDAR are not exactly positioned at the same place, which results in objects observable by one but hidden to the other. Therefore, the loss L_{hinted} integrates the LiDAR self-supervision only where image reconstruction is more precise by using the LiDAR signal instead of the estimated depth. More precisely, two versions of the photometric contribution of the pixel are computed: the regular pixel-wise photometric loss L_{photo}, using the estimated depth map $\hat{D}_t$ in Eq. (1), and L^H_{photo} using the input projected LiDAR H_t instead of $\hat{D}_t$ in Eq. (1). Then we only supervise with the LiDAR reconstruction when $L^H_{\text{photo}} < L_{\text{photo}}$. The objective is thus:

$$L_{\text{hinted}}(p) = \begin{cases} |\hat{D}_t(p) - H_t(p)| + L_{\text{photo}}(p) & \text{if } L^H_{\text{photo}}(p) < L_{\text{photo}}(p) \\ L_{\text{photo}}(p) & \text{otherwise.} \end{cases} \quad (5)$$

3.3. Pose estimation

The formulation of the photometric reconstruction involves the change of pose $\hat{P}_{t\to s}$ between the target image I_t and source view I_s for the source image warping. A first possibility, which is widely used in monocular self-supervised depth estimation (Zhou et al., 2017; Godard et al., 2019; Guizilini et al., 2020a; Casser et al., 2019a), uses a so-called *pose network* jointly trained with the depth network. However, due to the monocular ambiguity, this approach can only estimate a relative pose and thus relative depth maps, which then must be rescaled

by an unknown factor. Instead, we explore another way to estimates a metric pose, by leveraging the LiDAR information and solving a Perspective-n-Point problem (Lepetit et al., 2009; Gao et al., 2003). As such, depth estimation should also align to a real-world scaling.

Perspective-n-Point (PnP). The PnP problem originally seeks the absolute pose of a camera given a set of 3D points and their corresponding 2D image projections. In our case, we use the PnP formulation to estimate the change of pose between the target and source views, i.e., given the target image I_t and LiDAR measurements, as well as the source image I_s.

First, pairs of pixels (p_t, p_s) matching in both views I_t and I_s are found using the SIFT descriptor (Lowe, 2004) based on a DoG keypoint detector. Then, the sole pairs for which p_t has a LiDAR measurement are considered. This gives us the pairs of 3D-2D points, where points p_t are complemented with depth measurements and match the 2D points p_s of the source image I_s. Given these pairs, we can precisely estimate the *metric*-scaled 6D rigid transformation between the target and source poses by minimizing the cumulative projection error.

In challenging real-life situations, and especially when dealing with a 4-beam LiDAR, finding matching pixels that have LiDAR measurements can be arduous, making this method prone to errors. Hence, we follow Ma et al. (2019) to remove outliers in the set of point correspondences by using RANSAC in conjunction with the PnP solving algorithm. When this filtering step is insufficient for the algorithm solving the PnP problem to converge, we discard the training sample.

4. Experimental protocol

The first component of our protocol is the dataset used for the experiments, namely KITTI (Geiger et al., 2012), and our preprocessing to reduce the raw 64-beam LiDAR to a 4-beam one (Section 4.1). We then introduce baselines in Section 4.2. Additional details are given in the appendix.

4.1. Dataset and evaluation metrics

To train models in our LiDARTouch framework, we need a dataset that provides a camera stream with aligned sparse LiDAR data for training. We also require this dataset to have ground-truth depth data with an associated benchmark to assess and compare our test performances. We are aware of only one dataset matching both of these requirements, namely KITTI. It contains 1.5 hours of recorded driving sessions in urban environment from a video stream synchronized with LiDAR data. Depth ground truth is available: it is derived from dense LiDAR signals accumulated over five sweeps and stereo filtered. Overall, we use this dataset to train and evaluate the quality of the predictions of our framework, and to compare against baselines and variants. On the KITTI dataset (Geiger et al., 2012), we use the so-called Eigen split (Eigen et al., 2014) for train, val and test with a minor modification for the val and test. The ground-truth LiDAR of Uhrig et al. (2017) is not available for some of the frames of the Eigen splits (fewer than 10). Following common practice (Godard et al., 2019; Guizilini et al., 2020a), we removed them from the val and test splits. Thus, the total number of examples are 22537, 873 and 652 respectively for the train, val and test sets.

The LiDAR data provided in KITTI is obtained with high-end 64-beam sensors, appropriate for *evaluating* our self-supervised models, but much denser than what is expected to *train* our LiDARTouch framework. Consequently, we perform a filtering step to extract 4 beams out of the raw 64-beam LiDAR data. To conform with prior works (Ma et al., 2019; Jaritz et al., 2018; Guizilini et al., 2019) and better compare with them, we sample LiDAR beams uniformly: 1 beam is kept every 16. Note that with such a sampling, while 4 beams are extracted, only three beams effectively project onto the image plane as one beam falls out of the considered visual region.

Evaluation metrics. Evaluation is conducted against accumulated ground-truth LiDAR obtained following Uhrig et al. (2017), with the metrics defined in Eigen et al. (2014). This includes the absolute (Abs Rel) and square (Sq Rel) relative errors, the root mean square error (RMSE), and its log version ($\text{RMSE}_{\log}$), as well as precision-under-threshold metrics measuring the percentage of depth predictions $\hat{D}$ close enough to the ground-truth depth D, in the sense of the value $\delta := \max(\frac{\hat{D}}{D}, \frac{D}{\hat{D}})$ being under a user-defined threshold. Following Eigen et al. (2014), we consider three thresholds: $\delta < 1.25$, $\delta < 1.25^2$ and $\delta < 1.25^3$.

4.2. Notations, ablations and external baselines

Notations. To refer to the network architecture, independently of the rest of the learning framework, we use Monodepth2, Monodepth2-L, NLSPN, ACMNet and S2D. When we refer to whole models, i.e., architectures trained under the LiDARTouch framework, we append the 'LiDARTouch' prefix. For example, we note 'LiDARTouch-ACMNet' when we adapt the ACMNet architecture into the LiDARTouch framework.

For clarity, the inputs and the supervision schemes that are employed by the models are recalled in the tables of the experiments section. The input of each depth prediction model includes an image (noted '$\mathcal{I}$') and, optionally, a sparse 4-beam LiDAR point cloud ('$\mathcal{L}^4$'). We considered the following supervisions strategies: self-supervised photometric reconstruction ('P') associated to loss Eq. (2), supervised LiDAR ground-truth regression with L_1 loss ('L_{gt}'), or LiDAR self-supervision ('L_4') with one of the three options in Eqs. (3), (4), or (5).

Ablation: Pose estimation with a pose network. In Section 3.3, we presented the PnP algorithm, which estimates metric pose changes from source to target views. To highlight the gains enabled by the use of the extra LiDAR information for computing the pose, we experiment by training a *pose network* instead, a widely used component of monocular depth estimation models (Zhou et al., 2017; Godard et al., 2019; Guizilini et al., 2020a; Casser et al., 2019a). For each target-source image pair, the pose network outputs the 6D rigid transformation between views. It is differentiable and trained jointly with the depth network. When only trained with the photometric error (Eq. (2)), the 6D transformation is estimated up to a scale factor due to the monocular ambiguity. This results in a relative depth estimation requiring to be rescaled by the LiDAR depth ground-truth median value (not available in our case).

A solution is to use data from the IMU/GNSS to supervise the pose estimation scale. In the context of depth estimation, such an approach has been explored by Guizilini et al. (2020a). Formally, we first obtain the approximate change in pose between the source and target views ($P_{t\to s}$) from integrated inertial measurements. Then, we extract its translation component r and make the predicted pose translation component $\hat{r}$ regress its magnitude:

$$L_{\text{imu}} = \Big| \|r\|_2 - \|\hat{r}\|_2 \Big|. \tag{6}$$

As for a given pose there is a unique depth minimizing Eq. (2), constraining the pose's magnitude to a metric scale forces the depth estimation to be metric as well.

Baselines: Monocular methods. We compare against state-of-the-art monocular self-supervised approaches such as SfMLearner (Zhou et al., 2017), Vid2Depth (Mahjourian et al., 2018), GeoNet (Yin and Shi, 2018), DDVO (Wang et al., 2018), Monodepth2 (Godard et al., 2019) and PackNet-SfM (Guizilini et al., 2020a). Note that these methods can only produce relative depth maps, as they use an unsupervised pose network, so they have to be rescaled using the ground-truth LiDAR. Comparisons with these methods is thus unfair, in their favor.

Additionally, we compare with methods that directly produce metric depth by leveraging additional supervision. This includes (1) DORN (Fu et al., 2018), a camera-only method fully-supervised by a dense LiDAR signal, (2) Kuznietsov et al. (2017), a semi-supervised method using stereo reconstruction and dense LiDAR supervision, and (3) PackNet-SfM (Guizilini et al., 2020a) model supervised with IMU prior.

Baselines: Depth completion methods. We also compare against supervised depth completion methods, namely ACMNet (Zhao et al., 2021), NLSPN (Park et al., 2020) and S2D (Ma et al., 2019). However, their original versions are not trained and evaluated on the same splits as monocular methods. We re-train and evaluate them on the Eigen split, in their fully-supervised setting but with only a 4-beam LiDAR input. Additionally, we also train and evaluate these depth completion methods when the depth ground truth is simply replaced by the 4-beam LiDAR input for supervision signal. We refer to this setting as 'Naïve self-sup.'.

5. Influence of a touch of LiDAR

In this section, we validate setups where the depth network converges to a metric scale. In particular, in Section 5.1, we disentangle the contributions brought by LiDAR with an ablation study on the three levels of integration presented in Section 3: as a self-supervision signal, as a depth network's input, and as additional information for pose estimation. We also investigate various combinations of LiDAR self-supervision schemes and depth networks in Section 5.2.

5.1. Ablation of LiDAR

We begin with an ablation study to assess the contribution brought by sparse LiDAR at three different levels: supervision, input and pose. We define our LiDARTouch framework as using a P*n*P for pose estimation, LiDAR self-supervision (L_4) with the *masked* loss variant, and a bi-modal depth network (i.e., taking RGB and LiDAR as input). Models that belong to this framework are highlighted as light blue cells in Table 2. For the sake of clarity, in this section we focus on the leftmost three columns for direct comparison with LiDARTouch. Other learning setups are discussed in detail in Appendix C.

LiDAR as an input. First, we study the contribution brought by LiDAR when it is used as an input to the depth network in addition to the image signal. Results in the first column of Table 2 show that the Monodepth2 architecture, which does not use LiDAR as input, is consistently outperformed by all the other bi-modal architectures leveraging LiDAR input. These architectures achieve a relative improvement of 11-13% compared to Monodepth2. This validates the positive influence of integrating few-beam LiDAR as an input.

Self-supervision with the sparse LiDAR. Next, we study the impact of using a 4-beam LiDAR as a self-supervisory signal by removing it from the LiDARTouch framework, which leaves only the photometric loss ('P'). This corresponds to the second column in Table 2. Overall, the results support our claim that the use of LiDAR self-supervision improves or is similar in performance with respect to the photometric-only supervision schemes.

Although ACMNet, NLSPN and S2D architectures show slightly better performance when trained with P*n*P and the photometric-only loss, i.e., without any LiDAR self-supervision, they are severely affected by the infinite-depth issue (see Section 7).

Moreover, when using the photometric loss alone ('P' in the table), Monodepth2 and Monodepth2-L are hard to train. Indeed, while P*n*P pose is metric by construction, the depth network is initialized randomly and has to converge to a metric scale with the photometric reconstruction as the sole learning signal. Without any precaution, we observe large numerical differences in scale at initialization between the pose and depth, which provoke unstable training for the depth network. To address this instability, we divide the translation component of the P*n*P pose by a factor α during training and multiply the depth prediction consequently at inference (details in Appendix F). This procedure is inspired by the baseline scaling introduced in Monodepth2 for the stereo setting (Godard et al., 2019). We indicate models that need to be trained using this strategy with '$*$' in Table 2. On the other hand, under the LiDARTouch framework, all depth networks train well without requiring training tricks.

Pose estimation with a sparse LiDAR. We now show that a precise computation of the change of pose is critical to estimate depth maps that are correctly scaled, and that a touch of LiDAR is beneficial for this purpose. To demonstrate this, we experiment by replacing P*n*P in our LiDARTouch setup with a pose network that does not use any LiDAR information, as detailed in Section 4.2. This ablation of LiDARTouch corresponds to the third the column, 'P+L_4' under 'w/ pose network', in Table 2.

The main difference between these two setups is that P*n*P methods produce *metric* poses by construction, which leads to the collapse of depth solutions to metric depths. In opposition, the use of a pose network requires a joint alignment and con-

Table 2: **Pose estimation ablation**. We report precision (%) under threshold ($\delta < 1.25$) on the KITTI test split; higher is better. As we are interested in *metric* depth estimations, contrary to common practice (Godard et al., 2019; Guizilini et al., 2020a), estimations are **not** rescaled with LiDAR GT. Light blue cells indicate configurations corresponding to our LiDARTouch framework. Some models are more difficult to train and indicated in light grey cells. In particular, '$*$' implies that a rescaling of the pose was used for a stable training, and '†' indicates that the LiDAR signal had to be dilated to avoid overfitting to the LiDAR input (more details in Appendix D). When using a pose network with photometric supervision only (dark gray cells), the estimation can only be *relative* and the scores are all below 1%. More details are provided in Section 5.1.

		w/ P*n*P		w/ pose network			
Depth network		P+L_4	P	P+L_4	P	P+imu	P+L_4 +imu
Monodepth2	LiDARTouch	86.1	86.2*	83.9	-	86.2	86.5
Monodepth2-L	LiDARTouch	96.9	96.2*	94.9	-	96.4	96.5
ACMNet	LiDARTouch	97.4	97.5	91.2 †	-	27.0	95.3
NLSPN	LiDARTouch	95.9	96.8	94.2 †	-	38.5	94.1 †
S2D	LiDARTouch	96.2	96.4	93.9 †	-	28.7	94.0 †

vergence to a metric scale between the depth and pose networks as they are both randomly initialized. While Monodepth2-L achieves this, it can be observed that the use of a pose network instead of P*n*P degrades performance up to 6% when compared to LiDARTouch. Above all, we observe a tendency for ACMNet, NLSPN and S2D to overfit the LiDAR signal (see Figure B.10b for an example).

We find that the multi-scale prediction and supervision during training of Monodepth2 and Monodepth2-L are key for the models not to overfit the sparse 4-beam LiDAR data. Indeed, supervision at the lowest scale (1:8) increases the number of pixels getting supervision from LiDAR as pixels with associated LiDAR signal are expanded due to the difference in scale.

Building on this observation, we propose a procedure to simulate this behavior in order to avoid LiDAR overfitting for mono-scale networks without changing their architectures. To simulate a LiDAR self-supervision at a lower scale, we apply a *dilation* morphological operation on the 4-beam LiDAR at the supervision level. This artificially increases the number of pixels receiving LiDAR supervision, albeit in a noisy manner, and enables the mono-scale depth networks ACMNet, NLSPN and S2D to produce globally coherent metric depth estimations. We report results of models trained with this procedure (indicated by '†') in Table 2 and provide technical details as well as qualitative examples in Appendix D.

On the other hand, training under our LiDARTouch framework eliminates the need for such tricks. Indeed, results demonstrate that our LiDARTouch framework, using LiDAR as self-supervision, in input and in pose computation yields competitive performances for all the five architectures, a more stable training compared to any other configuration, and alleviates the infinite-depth problem as we will show in Section 7.

5.2. *LiDAR self-supervision variants*

We compare in Table 3 the variants for the LiDAR loss defined in Section 3.2, namely the *naïve* compound loss Eq. (3), the *masked* one Eq. (4), which prevents interferences with the photometric error, and the *hinted* loss Eq. (5), which handles the noise of the LiDAR signal. These experiments are conducted for the four different depth networks considered in Section 3.1. Overall, averaged over all architectures, the *masked* version of the LiDAR loss achieves the best results, demonstrating the need to reduce interferences between the LiDAR and photometric supervisions. On the other hand, we observe that the *hinted* loss yields the worst results. We expected the *naïve* loss to have the worst performance as it does not consider the noise in LiDAR, but it appears that the control the *hinted* loss imposes is too strong and discards too many of the already scarce LiDAR measurements. Hence, it confirms that the *masked* LiDAR self-supervision is the most effective.

Table 3: **Variants comparison of the LiDAR self-supervision.** RMSE metric (lower is better) on the Eigen test split of KITTI. Models are trained with photometric self-supervision (P) in conjunction with one of the three considered variants of minimal-LiDAR self-supervision (L_4). All models are trained with P*n*P for pose estimation.

Self-supervision	Monodepth2	Monodepth2-L	ACMNet	NLSPN	S2D
P + L_4(naïve)	4.504	2.796	2.490	3.084	2.839
P + L_4(hinted)	4.794	2.813	2.563	3.271	2.982
P + L_4(masked)	4.517	2.696	2.504	3.014	2.776

6. Comparison against related works

In Table 4, we report evaluations of the four architectures presented in Section 3.1, trained within our LiDARTouch framework against camera-only baselines.

Self-supervised camera-only methods. First, we show that training under our framework outperforms self-supervised monocular depth estimation methods (Zhou et al., 2017; Mahjourian et al., 2018; Yin and Shi, 2018; Wang et al., 2018; Godard et al., 2019; Guizilini et al., 2020a). We note that contrary to other methods, ours uses few-beam LiDAR as input. Furthermore, self-supervised monocular depth estimation approaches only estimate relative depth and thus are rescaled with ground truth before evaluation. With our approach, this unrealistic and impractical rescaling step is no longer needed.

Supervised camera-only methods. We also obtain better results than monocular depth estimation models trained with ground truth and optional stereo (Fu et al., 2018; Kuznietsov et al., 2017), while not requiring either of those. While the latter does not use few-beam LiDAR as input, not requiring ground truth at train time makes our method trainable at scale on any domain.

Overall, we showed that by integrating few-beam LiDAR in the pipeline, we substantially increase performances on all metrics over other methods not using few-beam LiDAR.

We compare our LiDARTouch framework against two supervision schemes from the depth completion literature: full-supervision with ground truth (L_{gt}) and self-supervision (L_4-

Table 4: **Comparison against monocular depth estimation methods.** Results are reported on the KITTI Eigen split (Eigen et al., 2014) with improved ground truth (Uhrig et al., 2017). A few self-supervised methods produce relative-depth maps and their prediction must be rescaled using ground-truth information; this is identified by '*gt rescaled*' in the table. Some of the methods also benefit from an extra pre-training, on ImageNet (Deng et al., 2009) or Cityscapes (Cordts et al., 2016), denoted with ∘ or ⋆ superscripts, respectively. The model *Monodepth2* in italic indicates our re-implementation of (Godard et al., 2019) without pre-training and post-processing. Input includes the image only ('$\mathcal{I}$'), or combined with the few-beam LiDAR point cloud ('$\mathcal{L}^4$'). Supervision includes photometric loss ('P'), IMU prior ('imu'), stereo reconstruction ('ste') and LiDAR supervision with either dense ground truth ('L_{gt}') or sparse 4-beam LiDAR ('L_4').

	Method	Input	Superv.	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$RMSE_{log}$ ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Sup.	DORN° (Fu et al., 2018)	$\mathcal{I}$	L_{gt}	0.072	0.307	2.727	0.120	0.932	0.984	0.995
	Kuznietsov et al.° (Kuznietsov et al., 2017)	$\mathcal{I}$	L_{gt}+ste	0.089	0.478	3.610	0.138	0.906	0.980	0.995
Self-supervised *gt rescaled*	SfMLearner⋆ (Zhou et al., 2017)	$\mathcal{I}$	P	0.176	1.532	6.129	0.244	0.758	0.921	0.971
	Vid2Depth⋆ (Mahjourian et al., 2018)	$\mathcal{I}$	P	0.134	0.983	5.501	0.203	0.827	0.944	0.981
	GeoNet⋆ (Yin and Shi, 2018)	$\mathcal{I}$	P	0.132	0.994	5.240	0.193	0.883	0.953	0.985
	DDVO (Wang et al., 2018)	$\mathcal{I}$	P	0.126	0.866	4.932	0.185	0.851	0.958	0.986
	Monodepth2 (our reimplem.)	$\mathcal{I}$	P	0.099	0.591	4.030	0.149	0.897	0.976	0.993
	Monodepth2° (Godard et al., 2019)	$\mathcal{I}$	P	0.090	0.545	3.942	0.137	0.914	0.983	0.995
	PackNet-SfM⋆ (Guizilini et al., 2020a)	$\mathcal{I}$	P	0.071	0.359	3.153	0.109	0.944	0.990	0.997
	Monodepth2 w/ IMU supervision	$\mathcal{I}$	P+imu	0.110	0.729	4.565	0.172	0.862	0.965	0.989
	PackNet-SfM⋆ (Guizilini et al., 2020a)	$\mathcal{I}$	P+imu	0.075	0.384	3.293	0.114	0.938	0.984	0.995
	LiDARTouch-SAN	$\mathcal{I}$+$\mathcal{L}^4$	P+L_4	0.063	0.396	3.318	0.118	0.946	0.982	0.993
	LiDARTouch-NLSPN	$\mathcal{I}$+$\mathcal{L}^4$	P+L_4	0.053	0.336	3.013	0.106	0.959	0.987	0.994
	LiDARTouch-S2D	$\mathcal{I}$+$\mathcal{L}^4$	P+L_4	0.059	0.285	2.776	0.102	0.962	0.988	0.995
	LiDARTouch-Monodepth2-L	$\mathcal{I}$+$\mathcal{L}^4$	P+L_4	0.047	0.267	2.696	0.090	0.969	0.991	0.996
	LiDARTouch-ACMNet	$\mathcal{I}$+$\mathcal{L}^4$	P+L_4	0.044	0.242	2.504	0.086	0.974	0.991	0.996

Table 5: **Comparison against supervised and naively self-supervised depth completion schemes.** Input includes the image and the 4-beam LiDAR ($\mathcal{I}$+$\mathcal{L}^4$)

	Network	Superv.	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$\delta < 1.25$ ↑
GT sup.	ACMNet	L_{gt}	0.030	0.143	2.112	0.983
	NLSPN	L_{gt}	0.044	0.214	2.617	0.971
	S2D	L_{gt}	0.035	0.152	2.271	0.979
	SAN	L_{gt}	0.037	0.172	2.491	0.976
Naive self-sup.	ACMNet	L_4	0.714	9.751	15.88	0.057
	NLSPN	L_4	4.133	268.4	51.96	0.010
	S2D	L_4	0.849	12.84	17.53	0.077
	SAN	L_4	0.426	6.226	14.148	0.243
LiDARTouch	ACMNet	P+L_4	0.044	0.242	2.504	0.974
	NLSPN	P+L_4	0.053	0.336	3.013	0.959
	S2D	P+L_4	0.059	0.285	2.776	0.962
	SAN	P+L_4	0.063	0.396	3.318	0.946

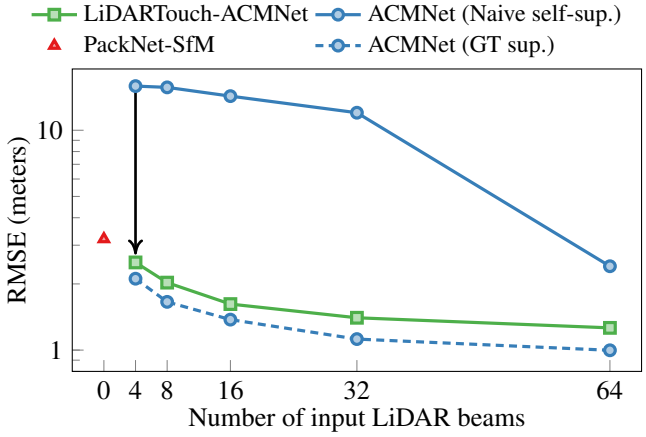


Fig. 5: **Comparison of different supervision schemes for the ACMNet architecture.** In the depth-completion setting, results are highly degraded when ground-truth depth information is no longer available for supervision (blue plots, 'GT sup.' *vs.* 'Naive self-sup.'). By combining ideas from self-supervised monocular depth estimation along with a careful integration of the LiDAR signal, we show that our self-supervised LiDARTouch framework can reach performance very close to the one offered by fully-supervised depth completion, as illustrated by the black arrow. Note that the *y*-axis is log-scaled.

naïve). These results are reported for the three architectures in Table 5.

Supervised depth completion methods. Unsurprisingly, supervising the training of any of the architectures with the privileged ground-truth depth yields better results than our LiDARTouch framework. However, LiDARTouch remain very competitive, e.g., 2.504 vs. 2.112 in RMSE for ACMNet. We also investigate the impact of the density of the input LiDAR on these scores in Figure 5. We observe that LiDARTouch is consistently close to the fully-supervised depth completion alternative when the number of layers varies.

Self-supervised depth completion method. The results in Table 5 show that the models trained with naïve 4-beam LiDAR self-supervision are unable to converge to decent results. Architectures cannot generalize from such a sparse LiDAR input as the supervisory signal is not sufficient. Moreover, in Figure 5, we remark that the naïve self-supervision scheme makes performance plummet when the LiDAR data becomes sparser. Furthermore, for the sake of completeness, we also experiment with SAN (Guizilini et al., 2021), a recent depth completion method with similar fusion scheme to the Monodepth2-L we propose in Section 3.1. Overall the results of SAN in Table 4 and Table 5 fall within the expected range, i.e., better than camera-only methods.

7. Alleviating the infinite-depth problem

We now study the infinite-depth problem affecting traditional pipelines and how well does the LiDARTouch framework solve it. First, we introduce a new metric to assess the degree and the frequency to which a model dramatically overestimates the distance to cars ahead (Section 7.1). This metric is employed for a quantitative evaluation of the problem in Section 7.2. Besides, we also provide a qualitative analysis of the problem and the significant improvements offered by LiDARTouch (Section 7.3).

7.1. Catastrophic Distance Rate (CDR) metric

Monocular image-only depth estimation methods suffer from the infinite-depth problem: vehicles with a motion close to that of the ego vehicle (in other words, with almost no relative motion) can be estimated as being infinitely far away. In the context of autonomous vehicles, such anomalies can lead to potentially dangerous outcomes. This critical weakness of image-only methods is not well reflected in the commonly-used evaluation metrics, as errors associated with these local flaws are overwhelmed by global scores aggregated at a dataset level.

This problem was qualitatively evaluated in some recent work (Zhou et al., 2017; Godard et al., 2019; Casser et al., 2019a,b; Guizilini et al., 2020a) but no precise measurement of its severity has yet been proposed. To address this issue, we define a novel quantitative metric, called the catastrophic distance rate (CDR), to assess the degree to which a model tends to make such disastrous predictions.

CDR measures the percentage of cars whose estimated distance to the ego-car is catastrophically poor in the test set. To this end, we use instance segmentation masks for all the vehicles of every image of the test set. With these vehicle instances, CDR is computed in a two-step process:

1. Instance mask filtering to keep the ones potentially concerned by the infinite-depth problem;
2. Computation of the depth error measured on these instance masks.

Instance mask filtering. For the first step of our CDR metric, we filter out irrelevant masks to only focus on vehicles typically concerned by the infinite-depth problem, i.e., first vehicle in front, unoccluded and not too far. As we use a centered frontal camera, we begin by discarding vehicles that are not in the center of the scene. We also remove cars whose instance masks are too small, considered too far from the ego vehicle. Then, to assess whether a car is occluded or not, we assume that a heavily occluded vehicle generally has a non-convex shape (e.g., incised by the front vehicle) and that, on the contrary, the mask of a non-occluded car is approximately convex. The overall process is illustrated in Figure 6 and further details are provided in the appendix.

CDR computation. CDR estimates the percentage of instances for which the relative depth error is above a manually-defined "catastrophic" threshold τ.

Within each segmentation mask M_k, indexed by $k \in \mathcal{K}$, we define the set $\mathcal{V}_k$ of pixels that possess a ground-truth LiDAR depth measurement: $\mathcal{V}_k = \{p \mid M_k(p) > 0 \wedge D_k(p) > 0\}$. Note that, as with H_t, $D(p) = 0$ if and only if there is no LiDAR point projecting at p. In the KITTI test set, the average size of $\mathcal{V}_k$ is 543. The error R_k made by the model on the instance mask M_k is measured by the average signed depth error over $\mathcal{V}_k$:

$$R_k = \frac{1}{|\mathcal{V}_k|} \sum_{p \in \mathcal{V}_k} \frac{\hat{D}_k(p) - D_k(p)}{D_k(p)}, \tag{7}$$

where $|\mathcal{V}_k|$ is the cardinality of $\mathcal{V}_k$. Please note that no absolute value is involved in the design of R_k as we focus only on the infinite-depth problem, i.e., $\hat{D}(p) > D(p)$, when a car is predicted catastrophically further away than its true position.

By thresholding the error R_k and aggregating it over instances, we define the "Catastrophic Distance Rate" as:

$$\text{CDR}(\tau) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} [\![R_k > \tau]\!], \tag{8}$$

with $[\![\cdot]\!]$ the Iverson bracket, $|\mathcal{K}|$ the number of instance masks and τ a user-defined threshold. For example, $\text{CDR}(\tau = 0.5) = 20\%$ indicates that the distance to front vehicles is over-estimated by more than 50% of the true distance in 20% of the cases.

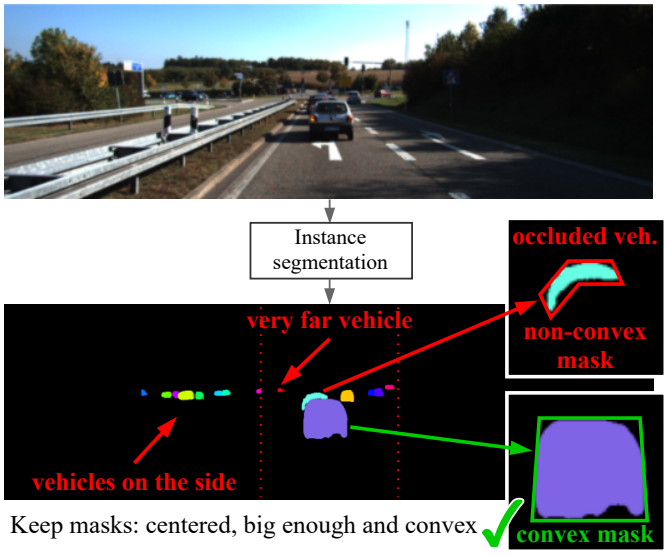


Fig. 6: **Selecting vehicles to compute the CDR metric.** The aim is to extract the individual mask of the first vehicles in front of the ego-car. These are indeed vehicles affected by infinite-depth error due to a small relative motion, leading to potentially catastrophic consequences. The proposed CDR metric computes the rate of such failures over the test set.

7.2. Quantitative analysis

To verify our intuition that LiDAR self-supervision is a suitable means to mitigate the infinite-depth problem, we study three models:

- A model that does not use the LiDAR signal at all, noted 'Monodepth w/ IMU supervision', which heavily suffers from the infinite-depth issue;

- A model with LiDAR as input and for the P*n*P-estimated pose, but supervised solely with the photometric loss, noted 'ACMNet$^{P}_{PnP}$';
- A model trained within the LiDARTouch framework, using LiDAR for the depth network, pose estimation and self-supervision, noted 'LiDARTouch-ACMNet'.

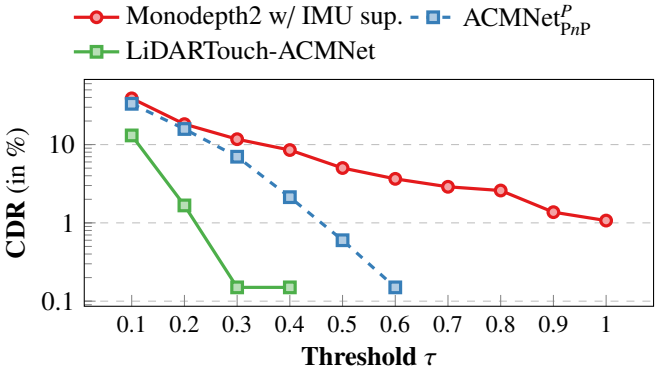


Fig. 7: **Plot of the CDR metric for various thresholds τ.** y-axis is log-scaled.

We plot the distribution of the CDR metric against the chosen threshold τ in Figure 7. We observe that the more LiDAR information is integrated, the fewer catastrophic estimations occur.

Indeed, ACMNet$^{P}_{PnP}$, which uses LiDAR both in input and pose, improves over Monodepth2 but is still affected by the infinite-depth issue. We also see a clear improvement of our LiDARTouch-ACMNet over the two other models. For example, for $\tau = 0.5$, i.e., the distance of a car is overestimated by at least half, Monodepth2 has a metric score of 5.02% while ACMNet$^{P}_{PnP}$ has 0.6% and LiDARTouch-ACMNet 0.0%. Such results show that Monodepth2 predictions cannot be trusted for downstream tasks such as car detection or free space estimation that are both required by functions like automatic emergency braking, keep-lane assist or adaptive cruise control. While ACMNet$^{P}_{PnP}$ reduces the likelihood of catastrophic estimation by 8 folds for $\tau = 0.5$, 0.6% is still too high to implement in a critical system intended for wide commercial use.

Overall, a network trained with our pipeline is significantly less impacted by the infinite-depth problem and we validate our hypothesis that, during training, the LiDAR self-supervision disambiguates cars estimated too far from their real distance. Hence, our models can accurately and safely handle moving objects with no relative motion, typical of cars in fluid traffic.

7.3. Qualitative analysis

The three examples in Figure 8 illustrate the improvement of our framework over the classic self-supervised camera-only pipeline. On the leftmost column, we observe a typical 'hole' in the depth map where Monodepth2 with IMU supervision estimates a vehicle three times more distant than in reality. in contrast to our model without such holes.

In addition to Figure 8, we provide some qualitative analyses where we show the depth maps obtained for different frameworks in Figure 9. First, we observe better overall depth maps with LiDARTouch-ACMNet than with Monodepth2. For example, we better estimate the two moving cyclists in Figure 9a as well as the fine tree trunks in Figure 9c.

As expected, the fully-supervised method ACMNet (GT-sup.) delivers the best-qualitative depth maps, as it leverages privileged ground-truth LiDAR depth during training. However, we observe that self-supervised approaches (Monodepth2 and LiDARTouch-ACMNet) better estimate areas near the top of the scene. This can be explained as LiDAR points are absent from regions above the road, which hinders ACMNet (GT-sup.) prediction in these regions due to the lack of supervisory signal it uses (last row in Figure 9).

Despite the successful integration of LiDAR in LiDARTouch, we note that some local depth estimation artifacts still occur, similar to the maps obtained from self-supervised depth estimation methods. Typically, this concerns distorted, reflective and color-saturated regions because the photometric reconstruction loss assumes Lambertian surfaces (cars in Figure 9c). Our model may also produce blurry depth predictions for small or thin objects, such as traffic signs (Figures 9a and 9b).

8. Conclusion

In this paper, we introduce LiDARTouch, a novel self-supervised framework for depth estimation with few-beam LiDAR. While being extremely sparse, we show that the LiDAR signal can be leveraged at three complementary levels of a self-supervised learning scheme. Across four different architectures, the LiDARTouch framework can reach competitive performances with respect to fully-supervised depth completion methods while being significantly cheaper and more annotation friendly. Moreover, we show that the sparse LiDAR signal provides valuable cues to disambiguate monocular depth estimation at a global level as well as for moving objects. Our method can be trained on any domain with no modification, and it can thus bring accurate and metric depth estimation at a fleet scale.

With our novel LiDARTouch framework, the new CDR metric to measure the infinite-depth problem, and the associate source code of our code, we hope to enable further research on the task of monocular depth prediction with minimal LiDAR input, typical of real-world assisted/automated driving systems.

Acknowledgements. Karteek Alahari was supported in part by the ANR grant AVENUE (ANR-18-CE23-0011). This work was granted access to the HPC resources of IDRIS under the allocation 2021-101766 made by GENCI.

Authors' biography

Florent Bartoccioni is a CIFRE PhD candidate in valeo.ai and Inria Grenoble Rhône-Alpes. He received the M.Sc. degree in computer science from the École Normale Supérieure de Rennes, France, in 2019. His research interests include dynamic scene forecasting and 3D perception for autonomous robots.

Éloi Zablocki is a research scientist at valeo.ai. He obtained his Ph.D. at Sorbonne University in 2019, on multimodal machine learning with language and vision. His research interests

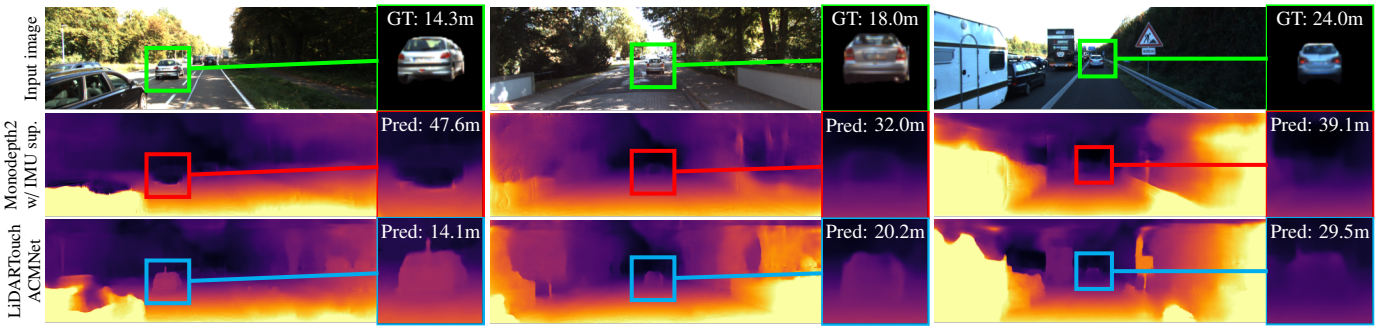


Fig. 8: **Mitigation of the infinite-depth problem.** Self-supervised image-only approaches tend to predict objects with no relative-motion at an infinite depth, as indicated by the hole in the depth close-up (red). In contrast, our LiDARTouch framework estimates the depth of these vehicles, as shown in the green close-up. Note that for the example in the middle, we verified that no LiDAR measurement falls on the car. This shows that our training framework can generalize well to cases where no LIDAR is available on critical moving vehicles.

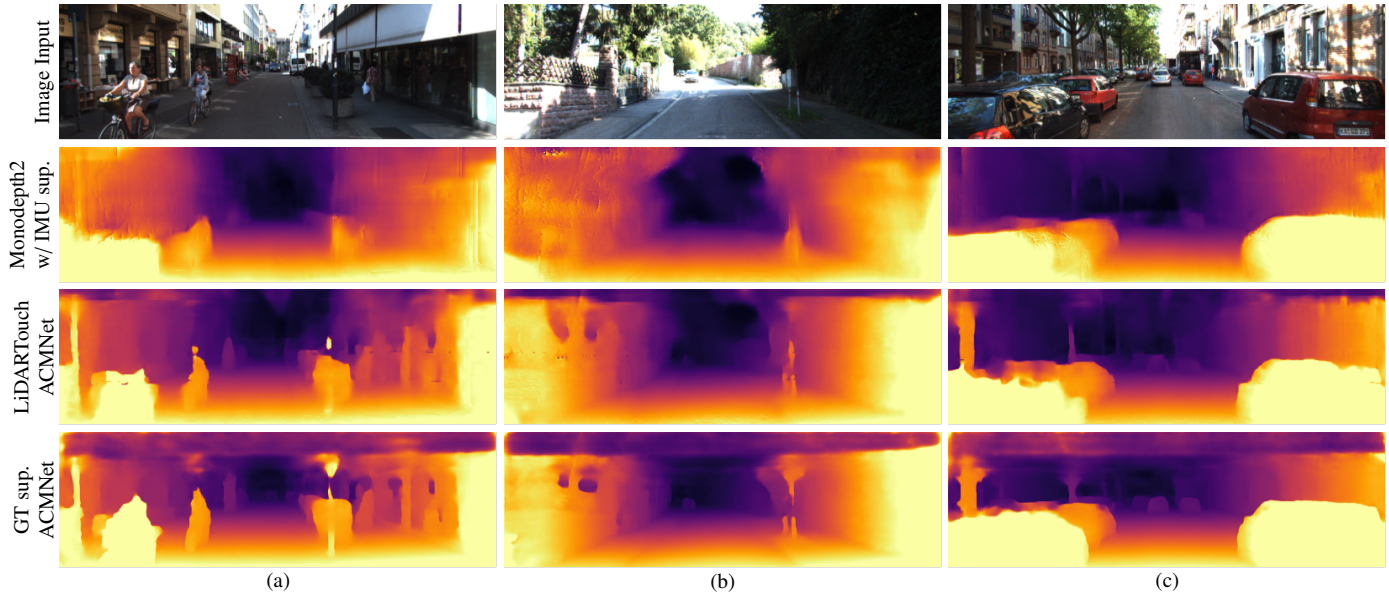


Fig. 9: **Qualitative comparison of LiDARTouch with other existing frameworks**. Monodepth2 is trained with IMU supervision. The model trained with GT supervision gives sharper depth estimates, but struggles in regions where GT signal is not available (e.g., top of the scene).

include computer vision for scene understanding, motion forecasting, and explainability of machine learning models.

Patrick Pérez is Scientific Director of valeo.ai. Before joining Valeo, Patrick Pérez has been a researcher at Technicolor (2009-2018), Inria (1993-2000, 2004-2009) and Microsoft Research Cambridge (2000-2004). His research interests include multimodal scene understanding and computational imaging.

Matthieu Cord is a full professor at Sorbonne University. He is also a part-time principal scientist at valeo.ai. His research expertise includes computer vision, machine learning, and artificial intelligence. He is the author of more than 150 publications on image classification, segmentation, deep learning, and multimodal vision and language understanding. He is an honorary member of the Institut Universitaire de France and served from 2015 to 2018 as an AI expert at CNRS and at the French National Research Agency.

Karteek Alahari is a senior researcher at Inria in the Grenoble - Rhône-Alpes center. He was previously a postdoctoral fellow in the Inria WILLOW team at the Department of Computer Science in École Normale Supérieure, after completing his PhD in 2010 in the UK. His current research focuses on learning robust and effective visual representations, when only partially-supervised data is available.

References

Amiri, A.J., Loo, S.Y., Zhang, H., 2019. Semi-supervised monocular depth estimation with left-right consistency using deep neural network, in: IEEE ROBIO.

Bradski, G., 2000. The opencv library. Dr. Dobb's Journal of Software Tools .

Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O., 2019. nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 .

Casser, V., Pirk, S., Mahjourian, R., Angelova, A., 2019a. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos, in: AAAI.

Casser, V., Pirk, S., Mahjourian, R., Angelova, A., 2019b. Unsupervised monocular depth and ego-motion learning with structure and semantics, in: CVPR Workshop.

Chang, J.R., Chen, Y.S., 2018. Pyramid stereo matching network, in: CVPR.

Chang, M., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., Hays, J., 2019. Argoverse: 3D tracking and forecasting with rich maps, in: CVPR.

Cheng, X., Zhong, Y., Dai, Y., Ji, P., Li, H., 2019. Noise-aware unsupervised deep lidar-stereo fusion, in: CVPR.
Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The Cityscapes dataset for semantic urban scene understanding, in: CVPR.
Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F., 2009. ImageNet: A large-scale hierarchical image database, in: CVPR.
Deng, Z., Todorovic, S., Jan Latecki, L., 2017. Unsupervised object region proposals for rgb-d indoor scenes. CVIU 154, 127–136.
Douglas, D.H., Peucker, T.K., 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: Intl. J. Geographic Information and Geovisualization 10, 112–122.
Eigen, D., Puhrsch, C., Fergus, R., 2014. Depth map prediction from a single image using a multi-scale deep network, in: NeurIPS.
Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D., 2018. Deep ordinal regression network for monocular depth estimation, in: CVPR.
Gao, X., Hou, X., Tang, J., Cheng, H., 2003. Complete solution classification for the perspective-three-point problem. IEEE TPAMI 25, 930–943.
Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for autonomous driving? The KITTI vision benchmark suite, in: CVPR.
Godard, C., Aodha, O.M., Firman, M., Brostow, G.J., 2019. Digging into self-supervised monocular depth estimation, in: ICCV.
Godard, C., Mac Aodha, O., Brostow, G.J., 2017. Unsupervised monocular depth estimation with left-right consistency, in: CVPR.
Groenendijk, R., Karaoglu, S., Gevers, T., Mensink, T., 2020. On the benefit of adversarial training for monocular depth estimation. CVIU 190, 102848.
Gruber, T., Bijelic, M., Heide, F., Ritter, W., Dietmayer, K., 2019. Pixel-accurate depth evaluation in realistic driving scenarios, in: 3DV.
Guizilini, V., Ambrus, R., Burgard, W., Gaidon, A., 2021. Sparse auxiliary networks for unified monocular depth prediction and completion, in: CVPR.
Guizilini, V., Ambrus, R., Pillai, S., Raventos, A., Gaidon, A., 2020a. 3D packing for self-supervised monocular depth estimation, in: CVPR.
Guizilini, V., Hou, R., Li, J., Ambrus, R., Gaidon, A., 2020b. Semantically-guided representation learning for self-supervised monocular depth, in: ICLR.
Guizilini, V., Li, J., Ambrus, R., Pillai, S., Gaidon, A., 2019. Robust semi-supervised monocular depth estimation with reprojected distances, in: CoRL.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: CVPR.
Jaritz, M., de Charette, R., Wirbel, É., Perrotton, X., Nashashibi, F., 2018. Sparse and dense data with CNNs: Depth completion and semantic segmentation, in: 3DV.
Jaritz, M., Vu, T.H., de Charette, R., Wirbel, E., Pérez, P., 2020. xMUDA: Cross-modal unsupervised domain adaptation for 3D semantic segmentation, in: CVPR.
Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., 2017. End-to-end learning of geometry and context for deep stereo regression, in: ICCV.
Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: ICLR.
Koestler, L., Yang, N., Wang, R., Cremers, D., 2020. Learning monocular 3D vehicle detection without 3D bounding box labels, in: GCPR.
Kumar, V.R., Milz, S., Witt, C., Simon, M., Amende, K., Petzold, J., Yogamani, S.K., Pech, T., 2018. Monocular fisheye camera depth estimation using sparse lidar supervision, in: IEEE ITSC.
Kuznietsov, Y., Stückler, J., Leibe, B., 2017. Semi-supervised deep learning for monocular depth map prediction, in: CVPR.
Lee, Y.H., Medioni, G., 2016. Rgb-d camera based wearable navigation system for the visually impaired. CVIU 149, 3–20.
Lepetit, V., Moreno-Noguer, F., Fua, P., 2009. Epnp: An accurate o(n) solution to the pnp problem. IJCV 81, 155–166.
Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. IJCV 60, 91–110.
Loza, A., Mihaylova, L., Canagarajah, N., Bull, D.R., 2006. Structural similarity-based object tracking in video sequences, in: IEEE FUSION.
Ma, F., Cavalheiro, G.V., Karaman, S., 2019. Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera, in: ICRA.
Ma, F., Karaman, S., 2018. Sparse-to-dense: Depth prediction from sparse depth samples and a single image, in: ICRA.
Mahjourian, R., Wicke, M., Angelova, A., 2018. Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints, in: CVPR.
Mohan, R., Valada, A., 2021. EfficientPS: Efficient panoptic segmentation. IJCV 129, 1551 – 1579.
Ng, M., Radia, K., Chen, J., Wang, D., Gog, I., Gonzalez, J., 2020. BEV-Seg: Bird's eye view semantic segmentation using geometry and semantic point cloud, in: CVPR.
Park, J., Joo, K., Hu, Z., Liu, C.K., Kweon, I.S., 2020. Non-local spatial propagation network for depth completion, in: ECCV.
Philion, J., Fidler, S., 2020. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D, in: ECCV.
Srikanth, S., Ansari, J.A., R., K.R., Sharma, S., Murthy, J.K., Krishna, K.M., 2019. INFER: INtermediate representations for FuturE pRediction, in: IROS.
Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al., 2020. Scalability in perception for autonomous driving: Waymo open dataset, in: CVPR.
Tang, J., Tian, F.P., Feng, W., Li, J., Tan, P., 2020. Learning guided convolutional network for depth completion. IEEE TIP .
Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A., 2017. Sparsity invariant CNNs, in: 3DV.
Wang, C., Buenaposada, J.M., Zhu, R., Lucey, S., 2018. Learning depth from monocular videos using direct methods, in: CVPR.
Watson, J., Firman, M., Brostow, G.J., Turmukhambetov, D., 2019. Self-supervised monocular depth hints, in: ICCV.
Xu, Y., Zhu, X., Shi, J., Zhang, G., Bao, H., Li, H., 2019. Depth completion from sparse LiDAR data with depth-normal constraints, in: ICCV.
Yamanaka, K., Matsumoto, R., Takahashi, K., Fujii, T., 2020. Adversarial patch attacks on monocular depth estimation networks. IEEE Access 8.
Yin, Z., Shi, J., 2018. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose, in: CVPR.
Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R., 2019. End-to-end interpretable neural motion planner, in: CVPR.
Zhao, S., Gong, M., Fu, H., Tao, D., 2021. Adaptive context-aware multi-modal network for depth completion. IEEE TIP .
Zhou, T., Brown, M., Snavely, N., Lowe, D.G., 2017. Unsupervised learning of depth and ego-motion from video, in: CVPR.

Appendix A. Implementation details

Training. We train all our models for 30 epochs using the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is set to 10^{-4} and divided by two halfway through training.

In all training pipelines, following common practice (Godard et al., 2017; Guizilini et al., 2020a; Godard et al., 2019), we add an edge-aware smoothing regularization loss to encourage the predicted depth map $\hat{D}_t$ to be locally smooth while taking into account sharp boundaries:

$$L_{\text{smooth}} = |\partial_x \hat{D}_t| e^{-|\partial_x I_t|} + |\partial_y \hat{D}_t| e^{-|\partial_y I_t|}, \tag{A.1}$$

with the index p over pixels omitted for clarity.

Monodepth2 extension. Our Monodepth2-L architecture is similar to Monodepth2 at the difference that we use a second ResNet-18 encoder specifically for the LiDAR modality. We only remove the first batch-normalization layer of the LiDAR ResNet, as using it would imply the computation of ineffective statistics given that the LiDAR input mostly contains zeros (encoding measurement absence).

Pose estimation. To solve the P*n*P problem, we use an open-source implementation of P*n*P methods with RANSAC from the OpenCV library (Bradski, 2000). We use 100 iterations and a reprojection error threshold of 2. Even after RANSAC, the remaining outliers are numerous enough to hinder training. Therefore, we remove the relative pose estimates for which the

translation magnitude $\|\hat{r}\|$ is too large. In effect, we first compute the median value of translation magnitude for each relative pose of the train set. Then, we remove all examples that are too far-off the median. When using a pose network, we follow (Godard et al., 2019) and use a ResNet-18 taking two images in input and outputting the parameters of $\hat{P}_{t\to s}$, the rigid transformation between the two views.

Evaluation after rescaling. Baselines and models from prior works that only provide relative-depth maps have their predictions rescaled so that they have the same mean compared to the ground truth against which they are evaluated. This is mentioned as '*gt rescaled*' in Table 4. For methods that directly produce metric depth maps, like ours, we do not apply this post-processing procedure and depth maps are kept at the originally-predicted scale.

CDR Metric. To compute results with our CDR metric, we first extract instance masks with EfficientPS (Mohan and Valada, 2021). Among these masks, we want to focus only on those of close-by, non-occluded vehicles, i.e., first vehicles in front of the ego-car. These vehicles are particularly prone to infinite-depth mistakes, with safety-critical consequences when it happens. To do this selection, vehicles that are not in front of the ego-car are discarded, as measured by not belonging to the central band of the scene (size is 20% of the image width) captured by the front camera. Vehicle having instance masks calculated with fewer than 20 pixels are considered too far from the ego vehicle. Then, to assess whether a car is occluded or not, we assume that a heavily occluded vehicle generally has a non-convex shape (e.g., incised by the front vehicle) and that, on the contrary, the mask of a non-occluded car is approximately convex. Accordingly, we first smooth segmentation masks and fill noisy areas where the intensity changes rapidly (e.g., edges, small holes from the wheels) by applying a morphological *dilation* operator. We use a square kernel of size 10 and 4 iterations for this operation. The masks now being smoothed, we then approach their shape by a polygon from which we can tell if they are convex or not. To approximate each pixel blob by a polygon, we use the Douglas–Peucker algorithm (Douglas and Peucker, 1973). The algorithm ensures the fit of the approximated polygon with an accuracy parameter dependent of the pixel blob size. After this first filtering step, 657 valid masks remain out of the 4460 vehicle masks of the KITTI test split.

Extracting 4 beams from 64-beam point clouds. In the KITTI dataset, the LiDAR data in a frame is provided as a unique point cloud, that is, a set of (x, y, z) coordinates, without the beam indexes, i.e., which of the 64 lasers has been used for each measurement. We needed to recover this information for our experiments. Fortunately, in KITTI the points are recorded in an orderly manner. The points of one beam follow the points of another in the direction of laser rotation (counter-clockwise). This means that, inside the data stream of a same frame, each rotation completion indicates a change of beam. More precisely, the coordinate basis of the LiDAR is oriented with x: positive forward and y: positive to the left of the car. Then we can compute the horizontal angle in radian of each point with:

$$\phi = \arctan2(y, x). \tag{A.2}$$

We use the 2-argument arctangent instead of classic arctangent, $\arctan(y/x)$, as the latter cannot distinguish between diametrically opposite directions. Then, by computing the horizontal angle (azimuth) of each point, we can separate data for each beam by detecting when ϕ changes from 360° to 0° in the stream of points. This way, we have access to the ring index for each LiDAR point and can, thus, freely sparsify the LiDAR data.

Code release. To enable comparison with our work in the future, all the processing steps described above will be included in the source code we plan to release. We will also release pre-trained models with our code for training and evaluating them.

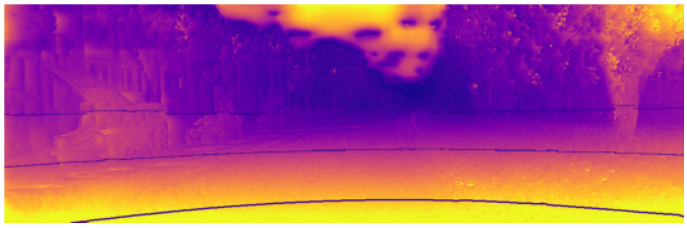
Appendix B. Overfitting to input LiDAR

In this section, we provide qualitative examples as well as elements of analysis for the convergence behavior observed on ACMNet, NLSPN and S2D that we call '*overfitted* to LiDAR input'. To this end, we compare S2D (*overfitted*) to Monodepth2-L (*metric*) trained with a pose network and ('P+L_4') supervision for 30 epochs. In essence, we refer to models as *overfitted* when most of the depth prediction is consistent but only *relative*, while depth prediction is only metric on pixels with LiDAR data. On Figure B.10b, we can clearly observe the difference in scale between areas with and without LiDAR data. Likewise, we can quantitatively observe the existence of two distinct scales within predictions of S2D. In the middle plot of Figure B.11, the median value of the inverse depth prediction (disparity) on pixels with LiDAR are roughly the same for S2D and Monodepth2-L, they are both scaled metrically. On the other hand, in the top plot of Figure B.11 showing the median value of disparity on pixels without LiDAR, there is a clear difference between Monodepth2-L, that is properly scaled, and S2D that converged to a random scale.

From a supervisory perspective, the depth network is stuck within a local minima where the photometric loss is mostly minimized apart on pixels with LiDAR data where it is clear the pixels are projected at different scale (see Figure B.10a). The amount of pixels with LiDAR data being very small, the erroneous photometric loss is on these areas is strongly dampened by the average over the whole image. So strongly dampened that that photometric loss between S2D and Monodepth2-L, respectively an *overfitted* and a *metric* model, almost perfectly match (see photometric loss Figure B.12). At the same time the LiDAR loss has already reached a minimum and the smoothness loss is not powerful enough to regularize this convergence behavior.

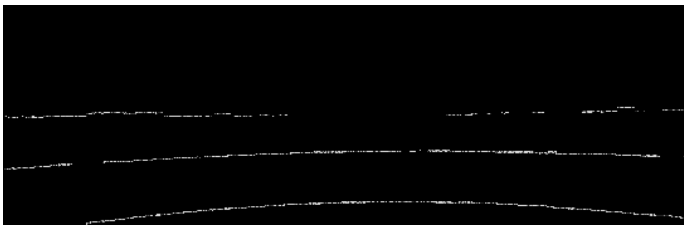
This convergence profile is expected because there are an infinite number of depth prediction scales for which the photometric loss is minimized over areas with no LiDAR data. Hence, there is an infinite number of local minima leading to this *overfitted* behavior. On the contrary, when using LiDAR self-supervision, only one depth prediction scale exists, the *metric* one, to obtain a globally coherent reconstruction. We propose a solution to this problem for S2D as well as ACMNet and NLSPN in Appendix D.

(a) Reconstructed view from source image, depth and pose predictions



(b) Deth estimation

(c) Mask on pixels without LiDAR data



(d) Mask on pixels with LiDAR data

Fig. B.10: **Predictions of a model overfitting to LiDAR input.** We show (a) a reconstructed view from source image, depth and pose prediction (b) a depth estimation considered as *overfitted* to the LiDAR input (c) a binary mask where the value is 1 for pixels without LiDAR data and 0 otherwise (d) a binary mask where the value is 1 for pixels with LiDAR data and 0 otherwise.

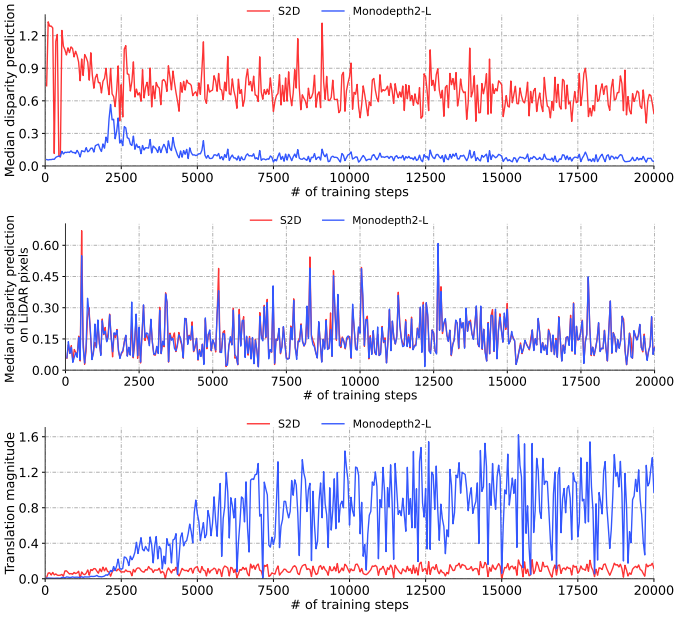


Fig. B.11: **Statistics for the depth and pose outputs over a training run for S2D (*overfitted*) and Monodepth2-L (*metric*) with a pose network and ('P+L$_4$') supervision.** Respectively from top to bottom, we provide the median value of the disparity predicted by the depth network on pixels without LiDAR data (see Figure B.10c), then on pixels with LiDAR data (see Figure B.10d), and the magnitude of the relative pose's translation component (i.e., by how much the pose network estimates the car moved between two views).

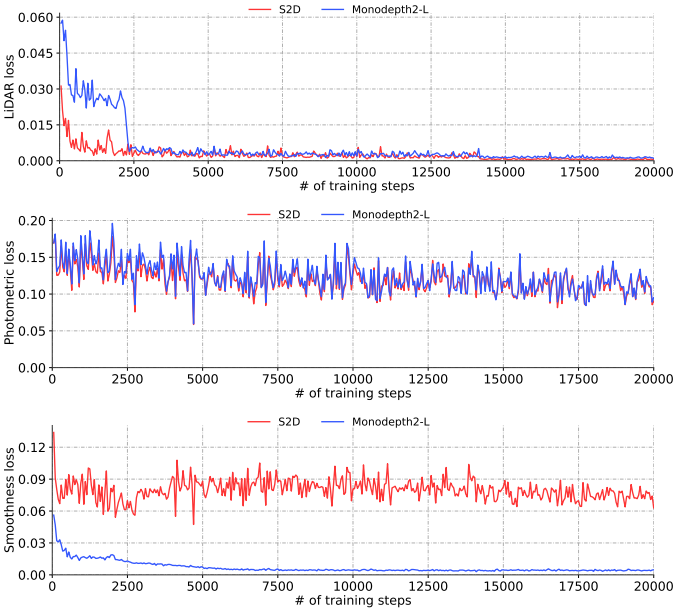


Fig. B.12: **Loss values over a training run for S2D (*overfitted*) and Monodepth2-L (*metric*) with a pose network and ('P+L$_4$') supervision.** Respectively from top to bottom, we provide the values over a training run for the self-supervised LiDAR loss, the photometric loss as well as the smoothness loss.

Appendix C. Ablation of LiDAR: further analysis

In this section, we analyse results for learning setups not described in Section 5.1. In particular, we continue to study the use of a pose network instead of P*n*P, with 'P', 'P+IMU' or 'P+L_4+IMU' supervision.

Overall, we observe very poor performances with the use of the pose network. First, we note that the use of photometric reconstruction only ('P' in Table 2) leads to to **relative** depth for all networks (dark gray cells in the Table 2). Indeed, this setup is well known as being an ill-posed problem (Zhou et al., 2017; Godard et al., 2019; Wang et al., 2018; Guizilini et al., 2020a); the pose provided by the monocular pose network can only be relative without additional information, and the depth estimation is thus unscaled as well.

To enforce a metric scale, we train the pose network with additional supervision in the form of an IMU prior ('P+IMU'), as explained in Section 4.2. While this helps Monodepth2 and Monodepth2-L to correctly train, ACMNet, NLSPN and S2D architectures cannot reach good performances when a joint alignment between a pose and depth network is required (see Appendix F for more details).

With further supervision from the input LiDAR ('P+L_4+IMU'), we can slightly increase results for Monodepth2 and Monodepth2-L as well as significantly boosting results for ACMNet compared to the ('P+IMU') setup (253% increase). However, similar to ACMNet, NLSPN and S2D in the ('P+L_4') setup (see Section 5.1), NLSPN and S2D tends to overfit the input LiDAR. Hence, we use the same *dilation* procedure, as detailed in Appendix D, for these models to avoid overfitting the LiDAR input.

Appendix D. Dilated LiDAR

Contrarily to Monodepth2 and Monodepth2-L, when trained with a pose network and LiDAR self-supervision, the networks ACMNet, NLSPN and S2D tend to overfit the LiDAR. Most of the depth prediction is consistent but only relative, while depth prediction on pixels with LiDAR data is metric (see Appendix B for an example). The main difference between these architectures is that Monodepth2 and Monodepth2-L are supervised at multiple scales (1:1, 1:2, 1:4 and 1:8) while ACMNet, NSLPN and S2D are only supervised at the final resolution (1:1). Supervision at the lowest scale (1:8) artificially increases the number of pixels getting supervision from LiDAR as a LiDAR point spans multiple pixels when projected at low resolutions.

We hypothesize that the mono-scale training is the cause of overfitting to LiDAR input when training with LiDAR self-supervision. This is confirmed by the fact that, when Monodepth2-L is only supervised at the scale 1:1, the model collapses into the *overfitted* regime which highlights the importance of multi-scale training.

As modifying the mono-scale networks is non-trivial, we propose to self-supervise with a dilated LiDAR to compensate for the lack of multi-scale supervision and to avoid overfitting the LiDAR input. More precisely, we apply two iterations of a *dilation* morphological operator with a kernel of 10×10 on the 4-beam LiDAR at the supervision level only (i.o.w., we do not apply *dilation* on the LiDAR input). The aim is to increase the number of pixels receiving LiDAR supervision, albeit in a noisy manner, (Figure D.13). This simple procedure, while remaining a trick, enables mono-scale architectures to avoid overfitting the input LiDAR and to converge to metric depth estimation. On the other hand, none of the architectures need such special care when trained under our LiDARTouch framework. We report results of models trained with this procedure with the superscript † inTable 2.

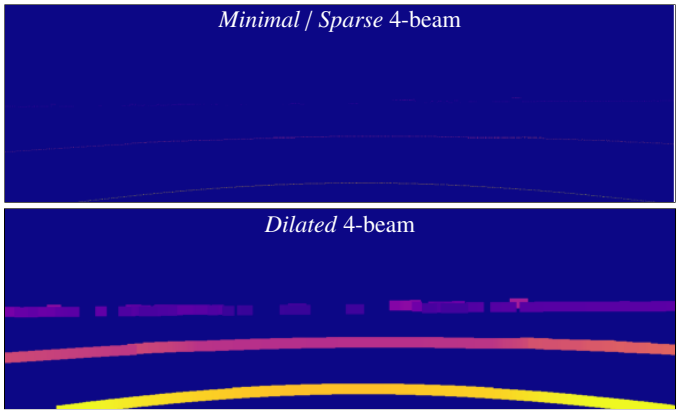


Fig. D.13: **Visual difference between vanilla and dilated LiDAR.**

In addition to this strategy, we explored various experimental setups and combination of hyper-parameters when training with (P+L4) and (P+L4+IMU) for mono-scale networks:

- Dividing the sparse LiDAR depth values (used as input and/or ground-truth) by a factor α at train time and multiply depth prediction consequently at validation. The network still overfits to LiDAR data with $\alpha \in \{10, 100, 1000\}$.
- Decreasing the contribution of the depth loss in the global objective to mitigate the overfitting behavior to LiDAR points. With $\lambda \in \{1, 1e-1, 1e-2, 1e-3\}$, the model still overfits the LiDAR. With $\lambda \in \{1e-4, 1e-5\}$ the network stops overfitting the LiDAR data but the depth estimation becomes only relative instead of being metric.
- Increasing the contribution of the smoothness loss in the global objective. By doing so, we hoped to uniformize the scale of the depth prediction on pixels without LiDAR that are neighbors to pixels with LiDAR. The network still overfits to LiDAR data with $\lambda \in \{1e-1, 1e-2, 1e-3\}$.
- Varying learning rate from $1e-3$ to $1e-5$. The network still overfits to LiDAR data.

Appendix E. Pose scaling is critical when using a P*n*P pose estimation with photometric loss only

Most of the depth network's learning signal comes from the reconstruction of the target image from the source image. For a given scale, a correct photometric reconstruction corresponds to a unique pair of depth and pose. Hence, for one to be metrically scaled, both the depth and the pose have to be metric. However, the networks are initialized randomly and thus need to jointly align and converge to a metric scale.

On the other hand, when using P*n*P, the estimated pose is metric thanks to LiDAR data (see Section 3.3), thus, only the depth network has to converge to the correct scale. However, this may produce a large difference in scale at initialization between the pose and depth, provoking unstable training for the depth network. Thus, one strategy we adopt to stabilize training is to divide the translation component of the P*n*P pose by 10 and multiply the depth prediction by 10 at inference time. Models trained with this strategy are indicated with the superscript $*$ in Table 2.

To circumvent these difficult training behaviors, we can use the P*n*P method to produce metric poses, and further enforce the collapse of the depth solutions to a metric scale with additional LiDAR self-supervision. This is consistently verified with the use of photometric and LiDAR supervisions (P+L_4) for each of the five architectures considered and leads to the best results compared to any other configuration (see Table 2). These results demonstrate that the use of LiDAR both as self-supervision and in pose computation yields performance on-par or better than camera-only setups.

Appendix F. Poor performances for ACMNet, NLSPN and S2D when trained with P+IMU

Unfortunately, we cannot make these models converge to metric depth estimations. We describe below the combination of hyper-parameters we experimented with:

- Dividing the pose GT (translation magnitude) by 10, 100, 1000 and multiplying depth predictions consequently.
- Varying the contribution of the smoothness loss with $\lambda \in \{1e-1, 1e-2, 1e-3\}$.
- Varying learning rate from $1e-3$ to $1e-5$.

In all these cases, **the networks still converge to bad quality depth estimations**.

We also investigate Monodepth2-L only supervised at the biggest scale to evaluate the influence of multi-scale training in the 'P+IMU' setup. We found that performances slightly decreased, but the network still converges to metric depth estimations. Hence, in this setup, multi-scale training does not seem to be crucial.