LiDARTouch: Monocular metric depth estimation with a few-beam LiDAR
Florent Bartoccioni, Éloi Zablocki, Patrick Pérez, Matthieu Cord, Karteek Alahari

To cite this version:
Florent Bartoccioni, Éloi Zablocki, Patrick Pérez, Matthieu Cord, Karteek Alahari. LiDARTouch: Monocular metric depth estimation with a few-beam LiDAR. 2021. hal-03508099

HAL Id: hal-03508099
https://hal.archives-ouvertes.fr/hal-03508099
Preprint submitted on 3 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
LiDARTouch: Monocular metric depth estimation with a few-beam LiDAR

Florent Bartoccioni\textsuperscript{a,b}, Éloi Zablocki\textsuperscript{a}, Patrick Pérez\textsuperscript{a}, Matthieu Cord\textsuperscript{a}, Karteek Alahari\textsuperscript{b}

\textsuperscript{a}Valeo.ai, 100 rue de Courcelle, 75017 Paris, France
\textsuperscript{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 \textit{metric} depth is estimated by combining a monocular camera with a \textit{minimal} sparse LiDAR input (e.g., 4 beams). Our motivations to use a sparse LiDAR input are diverse. First, 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 problems. Third, we hypothesize that a light LiDAR touch will result in the overall model correctly estimating the depth of moving objects, notably cars. 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 \textit{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 ground depth 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 naively 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 \textit{self-supervised} depth estimation framework, where a \textit{minimal} LiDAR and a monocular camera are available while ground-truth depth annotations are not accessible. This configuration is close to \textit{in situ} conditions of today’s vehicles, which is seldom addressed in other works.

2. To present our LiDARTouch regime, we revamp four recent architectures: three of them are adapted models from the depth-completion literature and one is an extension of a self-supervised monocular depth estimation model that integrates LiDAR information at different levels. With each of these models, we reduce the performance gap significantly between self-supervised monocular depth estimation and fully-supervised depth completion on the KITTI dataset.

3. We study extensively the influence of LiDAR at each stage of the framework. We show that the use of a few-beam LiDAR alleviates scale-ambiguity and infinite-depth issues from which camera-only methods suffer. To do this, we define a novel metric to quantitatively measure the infinite-depth issue for the first time in the literature.

2. Background and related work

For the rest of this paper, we refer to a LiDAR as \textit{dense} if it has more than 32 beams, and call it \textit{sparse} or \textit{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.

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 par-
tially or completely solved with the use of LiDAR input, depending on its resolution and position, without any additional assumptions.

**Depth completion methods.** They 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 visual and LiDAR inputs.

In Tang et al. (2020) and Jaritz et al. (2018), one encoder per modality is proposed with a multi-scale and a late fusion respectively. Alternatively, Xu et al. (2019); Park et al. (2020); Ma et al. (2019) use an early fusion. The fusion module of 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 in 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.

Alternatively, models 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. 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. 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.

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.

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 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 $L_{\text{photo}}$, 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 $L_{\text{photo}}$. 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 $\hat{P}_{t\rightarrow s}$ between the target and source views are estimated — we detail this in Section 3.1. One pose transformation $\hat{P}_{t\rightarrow s} = \begin{pmatrix} R & \hat{t} \\ 0 & 1 \end{pmatrix}$ is estimated for each source image $I_s \in S$, where $R$ is a rotation matrix and $\hat{t}$ 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 $p_s$ of a pixel in the target image, the projected coordinates $p_t$ in the source image are computed with:

$$p_t = K\hat{P}_{t\rightarrow s}\hat{D}_t(p_s)K^{-1}p_s.$$  \hfill (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:

$$L_{\text{photo}}(p) = \min_{\hat{I}_s \in S} \left[ \alpha \left( 1 - \text{SSIM}(I_t, \hat{I}_s(p)) \right) + (1 - \alpha) \left| I_t(p) - \hat{I}_s(p) \right| \right].$$  \hfill (2)

where $\alpha$ 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 $L_{\text{photo}}$ 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 $L_{\text{photo}}$:

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

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 $L_{\text{photo}}$ on pixels with a LiDAR measurement. Denoting $L_{\text{masked}}$ as this loss, it is given by:

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

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

Lastly, inspired by Watson et al. (2019), we also introduce the hinted self-supervision, $L_{\text{hinted}}$, 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 $L_{\text{hinted}}$ 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 $L_{\text{photo}}$, using the estimated depth map $\hat{D}_t$ in Eq. (1), and $L_{\text{photo}}^H$ using the input projected LiDAR $H_t$ instead of $\hat{D}_t$ in Eq. (1). Then we only supervise with the LiDAR reconstruction when $L_{\text{photo}}^H < L_{\text{photo}}$. The objective is thus:

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

3.3. Pose estimation

The formulation of the photometric reconstruction involves the change of pose $\hat{P}_{t\leftarrow s}$ 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 estimate 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 $I_s$. 

---

**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 $\circ$.
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, and in Section 4.3 propose a new metric to assess the occurrence of catastrophically-erroneous depth predictions on key objects.

Additional details are given in the supplementary material.

4.1. Dataset and common 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.

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 ‘\(I\)’) and, optionally, a sparse 4-beam LiDAR point cloud (‘\(\mathcal{L}_4\)’). We considered the following supervision-strategies: self-supervised photometric reconstruction (‘\(P\)’) associated to loss Eq. (2), supervised LiDAR ground-truth regression with \(L_1\) loss (‘\(L_1\)’), 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 \((P_{\text{trans}})\) from integrated inertial measurements. Then, we extract its translation component \(r\) and make the predicted pose translation component \(\hat{r}\) regress its magnitude:

\[
L_{\text{imu}} = \|\|r\|_2 - \|\hat{r}\|_2\|.
\]
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 SfM-Learner (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 ACM-Net (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 a small relative motion, leading to potentially catastrophic consequences. The proposed CDR metric computes the rate of such failures over the test set.

### 4.3. 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 masks, CDR is computed in a two-step process:

1. **Instance mask filtering** to keep the ones potentially concerned by the infinite-depth problem;

```latex
\begin{align*}
\text{keep masks: centered, big enough and convex} & \quad \text{convex mask} \\
\text{occluded vehicle mask} & \quad \text{non-convex mask}
\end{align*}
```

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 5 and further details are provided in the supplementary material.

**CDR computation.** CDR estimates the percentage of instances for which the relative depth error is above a manually-defined “catastrophic” threshold $\tau$.

Within each segmentation mask $M_k$, indexed by $k \in \mathcal{K}$, we define the set $V_k$ of pixels that possess a ground-truth LiDAR depth measurement: $V_k = \{ p \mid M_k(p) > 0 \land D_k(p) > 0 \}$. Note that, as with $H$, $D(p) = 0$ if and only if there is no LiDAR point projecting at $p$. In the KITTI test set, the average size of $V_k$ 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 $V_k$:

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

where $|V_k|$ is the cardinality of $V_k$. 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:

$$CDR(\tau) = \frac{1}{|\mathcal{X}|} \sum_{k \in \mathcal{X}}[R_k > \tau],$$

(8)

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

5. Experiments and discussions

We validate the effectiveness of our LIDARTouch framework by 1) evaluating and disentangling the contributions brought by the multiple LiDAR integrations, 2) obtaining state-of-the-art results on self-supervised depth estimation, 3) successfully adapting architectures from the depth completion literature to a self-supervised scheme, 4) studying how LiDAR integration alleviates the infinite-depth problem, and 5) analysing qualitatively our models.

5.1. Contributions brought by a touch of LiDAR

We begin with an ablation study to assess the contribution brought by the sparse LiDAR at three different levels: as a self-supervision signal, as an input of the depth network, and as additional information for pose estimation.

Self-supervision with the sparse LiDAR. To study the impact of having 4-beam LiDAR as a self-supervisory signal, in addition to the photometric reconstruction, we experiment with the configurations presented in Section 3. We also consider various combinations of LiDAR self-supervision schemes and depth networks. The results of these experiments are reported in Table 2.

These results support that the use of a LiDAR self-supervision improves performance or stays on-par with photometric-only supervision scheme. This is expected as we include additional information from the LiDAR in the supervisory signal. Indeed, LiDAR self-supervision appears to be especially necessary for Monodepth2 and Monodepth2-L, the photometric self-supervision being not sufficient to learn a meaningful model. Concerning ACMNet, NLSPN and S2D architectures, they show slightly better performances when trained with the photometric loss only. However, without any LiDAR self-supervision, they remain badly affected by the infinite-depth phenomenon as we will discuss in Section 5.4.

We now go further in studying the design of LiDAR self-supervision. In Table 2, we compare the loss variants for the LiDAR reconstruction defined in Section 3.2, namely the naive 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 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. The control that it imposes is too strong and discards too many of the already scarce LiDAR measurements. Hence, masked LiDAR self-supervision, being the most effective, is employed in all further experiments.

LiDAR as an input. Next, we study the contribution brought by LiDAR when used as an additional input to the model, and discuss the modeling choices to merge LiDAR and image signals in this case. In Table 2, we compare the Monodepth2 architecture, whose depth network does not use LiDAR as input, against all other bi-modal architectures leveraging LiDAR input. We observe that Monodepth2 is consistently outperformed by these models when they are trained with LiDAR self-supervision, with a relative improvement of at least 13%. This validates the positive influence of integrating the few-beam LiDAR as an input.

When no LiDAR self-supervision is involved, we observe a clear superiority of the early-fusion designs over multi-scale fusion. Indeed, ACMNet, NLSPN and S2D start leveraging the LiDAR signal at the encoder level, which gives metric-scale depth predictions that are more accurate. In contrast, Monodepth2 and Monodepth2-L, which are either not using LiDAR at all or integrating it later in the architecture, fail to align the scale of their depth predictions with the metric pose coming from the PnP solver. In this case the network converges to a degenerate case where depth is uniform. When LiDAR self-supervision is employed (P+L4), these fusion differences tend to have much less impact on the learning process. This suggests that the architecture choice is directly linked to the supervision scheme and that LiDAR self-supervision helps in avoiding degenerated local minima.

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 the use of a touch of LiDAR is beneficial for this purpose. To demonstrate this, we experiment by replacing the PnP depth estimation with a pose network that does not use any LiDAR information, as detailed in Section 5.2. Results of these experiments, conducted with various depth networks and supervision schemes, are reported in Table 3.

Overall, we observe very poor performances with the use of the pose network. First, we note that the use of photometric reconstruction only (P) leads to unsatisfactory results for all net-
works. Indeed, the pose provided by the pose network is only relative, and the depth estimation is thus unscaled as well. To enforce a metric scale, we train the pose network with an additional supervision, either the IMU prior explained in Section 5.2 (P+imu), or the LiDAR reconstruction self-supervision (P+L4). While this helps Monodepth2 and Monodepth2-L to correctly train, the ACMNet, NLSPN and S2D architectures either reach a degenerated local minima or overfit to the LiDAR input. This signifies that the pose network cannot be a guide consistently to produce accurate depth maps with the supervision schemes we consider.

An alternative to using a pose network is to integrate the LiDAR signal in the pose estimation step. This enables the use of PnP methods that produce scaled poses, and thus, the collapse of the depth solutions to the metric depth. This is consistently verified with the use of the photometric and LiDAR supervisions (P+L4) for each of the five architectures considered, and leads to the best results compared to any other configuration.

5.2. Comparison against camera-only methods

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

Self-supervised camera-only methods. First, we show that training under our framework outperforms both supervised (Fu et al., 2018; Kuznietsov et al., 2017) and 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.

5.3. Comparison against depth completion methods

We compare our LiDARTouch framework against two supervision schemes from the depth completion literature: full-supervision with ground truth (Lgt) and self-supervision (L4-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 ACMet. We also investigate the impact of the density of the input LiDAR on these scores in Figure 6. We observe that LiDARTouch is consistently close to the fully-supervised depth completion alternative when the number of layers varies.

5.4. Alleviating the infinite-depth problem

Next, we quantitatively investigate the infinite-depth problem, i.e., the degree and the frequency to which a model dramatically overestimates the distance to cars in front, using the novel
A few self-supervised methods produce relative-depth maps and their prediction must be rescaled using ground-truth information; this is identified by \( \text{gt rescaled} \) 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 \( \circ \) or \( \star \) 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 (\( 'I' \)), or combined with the few-beam LiDAR point cloud (\( 'L' \)). 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 (\( 'L_{4b} \)).

### Table 4: Comparison against monocular depth estimation methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Input Superv.</th>
<th>Abs Rel ( \downarrow )</th>
<th>Sq Rel ( \downarrow )</th>
<th>RMSE ( \downarrow )</th>
<th>RMSE ( \log ) ( \downarrow )</th>
<th>( \delta &lt; 0.25 ) ( \uparrow )</th>
<th>( \delta &lt; 1.25 ) ( \uparrow )</th>
<th>( \delta &lt; 1.25^2 ) ( \uparrow )</th>
<th>( \delta &lt; 1.25^3 ) ( \uparrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DORN(^{+}) (Fu et al., 2018)</td>
<td>( L_{gt} )</td>
<td>0.072</td>
<td>0.307</td>
<td>2.727</td>
<td>0.120</td>
<td>0.932</td>
<td>0.984</td>
<td>0.995</td>
<td></td>
</tr>
<tr>
<td>Kuznietsov et al.(^{+}) (Kuznietsov et al., 2017)</td>
<td>( L_{gt} )</td>
<td>0.089</td>
<td>0.478</td>
<td>3.610</td>
<td>0.138</td>
<td>0.906</td>
<td>0.980</td>
<td>0.995</td>
<td></td>
</tr>
<tr>
<td>SfMLearner(^{+}) (Zhou et al., 2017)</td>
<td>( P )</td>
<td>0.176</td>
<td>1.532</td>
<td>6.129</td>
<td>0.244</td>
<td>0.758</td>
<td>0.921</td>
<td>0.971</td>
<td></td>
</tr>
<tr>
<td>Monodepth(^{+}) (Godard et al., 2019)</td>
<td>( P )</td>
<td>0.134</td>
<td>0.983</td>
<td>5.501</td>
<td>0.203</td>
<td>0.827</td>
<td>0.944</td>
<td>0.981</td>
<td></td>
</tr>
<tr>
<td>GeoNet(^{+}) (Yin and Shi, 2018)</td>
<td>( P )</td>
<td>0.132</td>
<td>0.994</td>
<td>5.240</td>
<td>0.193</td>
<td>0.883</td>
<td>0.953</td>
<td>0.985</td>
<td></td>
</tr>
<tr>
<td>DDVO (Wang et al., 2018)</td>
<td>( P )</td>
<td>0.126</td>
<td>0.866</td>
<td>4.932</td>
<td>0.185</td>
<td>0.851</td>
<td>0.958</td>
<td>0.986</td>
<td></td>
</tr>
<tr>
<td>Monodepth(^{+}) (Godard et al., 2019)</td>
<td>( P )</td>
<td>0.099</td>
<td>0.591</td>
<td>4.030</td>
<td>0.149</td>
<td>0.897</td>
<td>0.976</td>
<td>0.993</td>
<td></td>
</tr>
<tr>
<td>PackNet-SIM(^{+}) (Guizilini et al., 2020a)</td>
<td>( P )</td>
<td>0.071</td>
<td>0.359</td>
<td>3.153</td>
<td>0.109</td>
<td>0.944</td>
<td>0.990</td>
<td>0.997</td>
<td></td>
</tr>
<tr>
<td>Monodepth(^{+}) w/ IMU supervision</td>
<td>( P+imu )</td>
<td>0.110</td>
<td>0.729</td>
<td>4.565</td>
<td>0.172</td>
<td>0.862</td>
<td>0.965</td>
<td>0.989</td>
<td></td>
</tr>
<tr>
<td>PackNet-SIM(^{+}) (Guizilini et al., 2020a)</td>
<td>( P+imu )</td>
<td>0.075</td>
<td>0.384</td>
<td>3.293</td>
<td>0.114</td>
<td>0.938</td>
<td>0.984</td>
<td>0.995</td>
<td></td>
</tr>
<tr>
<td>LiDARTouch-NLSPN</td>
<td>( +L_{4b} ) ( +P+L_4 )</td>
<td>0.053</td>
<td>0.336</td>
<td>3.013</td>
<td>0.106</td>
<td>0.959</td>
<td>0.987</td>
<td>0.994</td>
<td></td>
</tr>
<tr>
<td>LiDARTouch-S2D</td>
<td>( +L_{4b} ) ( +P+L_4 )</td>
<td>0.059</td>
<td>0.285</td>
<td>2.776</td>
<td>0.102</td>
<td>0.962</td>
<td>0.988</td>
<td>0.995</td>
<td></td>
</tr>
<tr>
<td>LiDARTouch-Monodepth2-L</td>
<td>( +L_{4b} ) ( +P+L_4 )</td>
<td>0.047</td>
<td>0.267</td>
<td>2.696</td>
<td>0.090</td>
<td>0.969</td>
<td>0.991</td>
<td>0.996</td>
<td></td>
</tr>
<tr>
<td>LiDARTouch-ACMNet</td>
<td>( +L_{4b} ) ( +P+L_4 )</td>
<td>0.044</td>
<td>0.242</td>
<td>2.504</td>
<td>0.086</td>
<td>0.974</td>
<td>0.991</td>
<td>0.996</td>
<td></td>
</tr>
</tbody>
</table>

### Table 5: Comparison against supervised and naively self-supervised depth completion schemes

<table>
<thead>
<tr>
<th>Network</th>
<th>Superv.</th>
<th>Abs Rel ( \downarrow )</th>
<th>Sq Rel ( \downarrow )</th>
<th>RMSE ( \downarrow )</th>
<th>( \delta &lt; 1.25 ) ( \uparrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT-sup.</td>
<td>ACMNet</td>
<td>0.030</td>
<td>0.143</td>
<td>2.112</td>
<td>0.983</td>
</tr>
<tr>
<td></td>
<td>NLSNP</td>
<td>0.044</td>
<td>0.214</td>
<td>2.617</td>
<td>0.971</td>
</tr>
<tr>
<td></td>
<td>S2D</td>
<td>0.035</td>
<td>0.152</td>
<td>2.271</td>
<td>0.979</td>
</tr>
<tr>
<td>Nave self-sup.</td>
<td>ACMNet</td>
<td>0.714</td>
<td>9.751</td>
<td>15.88</td>
<td>0.057</td>
</tr>
<tr>
<td></td>
<td>NLSNP</td>
<td>4.133</td>
<td>268.4</td>
<td>51.96</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>S2D</td>
<td>0.849</td>
<td>12.84</td>
<td>17.53</td>
<td>0.077</td>
</tr>
<tr>
<td>LiDARTouch</td>
<td>ACMNet</td>
<td>0.044</td>
<td>0.242</td>
<td>2.504</td>
<td>0.974</td>
</tr>
<tr>
<td></td>
<td>NLSNP</td>
<td>0.053</td>
<td>0.336</td>
<td>3.013</td>
<td>0.959</td>
</tr>
<tr>
<td></td>
<td>S2D</td>
<td>0.059</td>
<td>0.285</td>
<td>2.776</td>
<td>0.962</td>
</tr>
</tbody>
</table>

metric introduced in Section 4.3. Our intuition is that LiDAR self-supervision is a suitable means to mitigate this problem.

To verify this hypothesis, 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 PnP-estimated pose, but supervised solely with the photometric loss, noted \( \text{ACMNet}^p \);\(^{p} \);
- A model trained within the LiDARTouch framework, using LiDAR for the depth network, pose estimation and self-supervision, noted ‘LiDARTouch-ACMNet’.

We plot the distribution of the CDR metric against the chosen threshold \( \tau \) in Figure 8. We observe that the more LiDAR information is integrated, the fewer catastrophic estimations occur.

Indeed, \( \text{ACMNet}^p \), 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 \( \text{ACMNet}^p \) 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 \( \text{ACMNet}^p \) 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.

In Figure 7, we present three examples along with selected close-ups highlighting the infinite-depth problem. For example, on the leftmost column, we observe a typical ‘hole’ in the depth map where Monodepth2 with IMU supervision estimates a vehicle three times as far as its true distance.

5.5. Qualitative analysis

In addition to Figure 7, 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).

6. 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 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.

Karteeck 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


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


Godard, C., Mac Aodha, O., Brostow, G.J., 2017. Unsupervised monocular depth estimation with left-right consistency, in: CVPR.


Guizilini, V., Li, J., Ambrus, R., Pillai, S., Gaidon, A., 2019. Robust semi-supervised monocular depth estimation with reprojected distances, in: CoRL.


for the visually impaired. CVIU 149, 3–20.
Ma, F., Karaman, S., 2018. Sparse-to-dense: Depth prediction from sparse depth samples and a single image, in: ICRA.

Appendix A. Implementation details

Training. All our models are trained 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 $D_t$ to be locally smooth while taking into account sharp boundaries:

$$I_{\text{smooth}} = |\partial_x \hat{D}_t e^{-|\partial_y \hat{D}_t|} | + |\partial_y \hat{D}_t e^{-|\partial_x \hat{D}_t|} |,$$

(A.1)

with the index $p$ over pixels omitted for clarity.

Monodpth2 extension. Our Monodpth2-L architecture is similar to Monodpth2 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 PnP problem, we use an open-source implementation of PnP 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 $||\hat{P}||$ 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 $\hat{P}_{t-1}$, 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 extracts 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 on 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 dilation morphological 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 on the pixel blob size. After this first filtering step, 657 valid masks are produced metric depth maps, like ours, we do not apply this post-processing procedure and depth maps are kept at the originally-predicted scale.

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:

$$\phi = \arctan2(y, x).$$ (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 $\phi$ changes from $360^\circ$ to $0^\circ$ 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.