

# Robust Monocular Depth Estimation under Challenging Conditions

Stefano Gasperini<sup>\*,1,2</sup> Nils Morbitzer<sup>\*,1</sup> HyunJun Jung<sup>1</sup>  
Nassir Navab<sup>1</sup> Federico Tombari<sup>1,3</sup>

<sup>1</sup> Technical University of Munich <sup>2</sup> VisualAIs <sup>3</sup> Google

## Abstract

While state-of-the-art monocular depth estimation approaches achieve impressive results in ideal settings, they are highly unreliable under challenging illumination and weather conditions, such as at nighttime or in the presence of rain. In this paper, we uncover these safety-critical issues and tackle them with *md4all*: a simple and effective solution that works reliably under both adverse and ideal conditions, as well as for different types of learning supervision. We achieve this by exploiting the efficacy of existing methods under perfect settings. Therefore, we provide valid training signals independently of what is in the input. First, we generate a set of complex samples corresponding to the normal training ones. Then, we train the model by guiding its self- or full-supervision by feeding the generated samples and computing the standard losses on the corresponding original images. Doing so enables a single model to recover information across diverse conditions without modifications at inference time. Extensive experiments on two challenging public datasets, namely *nuScenes* and *Oxford RobotCar*, demonstrate the effectiveness of our techniques, outperforming prior works by a large margin in both standard and challenging conditions. Source code and data are available at: <https://md4all.github.io>.

## 1. Introduction

Estimating the depth of a scene is a fundamental task for autonomous driving and robotics navigation. While supervised monocular depth estimation approaches have achieved remarkable results, they rely on ground truth data which is expensive and time-consuming to produce [20, 13]. This requires costly 3D sensors (e.g., LiDAR) and significant additional data processing [20, 13].

To circumvent these issues, geometrical constraints on stereo pairs or monocular videos have been widely explored

\* The authors contributed equally.

Contact author: Stefano Gasperini ([stefano.gasperini@tum.de](mailto:stefano.gasperini@tum.de)).

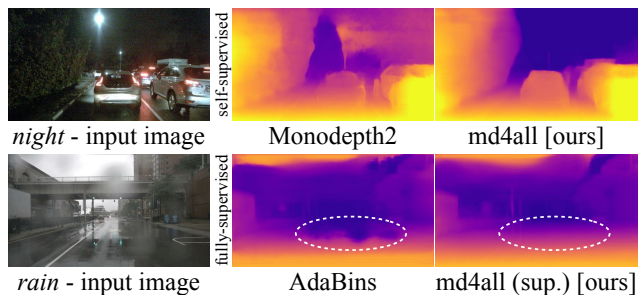


Figure 1. Predictions in challenging settings [4] for self-supervised [12] and supervised [1] methods. Standard approaches fail due to training assumptions or sensor artifacts. Under both supervisions, our *md4all* makes the same models robust in all conditions.

to learn depth estimation in a self-supervised manner [12, 26, 35, 8, 11]. Monocular training solutions are the most inexpensive and rely on the smallest amount of assumptions on the sensor setup, as they require only image sequences captured by a single camera.

Self-supervised methods rely on photometric assumptions and pixel correspondences [12, 35]. State-of-the-art approaches [12, 42, 33] deliver sharp and accurate estimates in standard conditions (i.e., sunny and cloudy), but suffer from a variety of inherent issues, such as scale ambiguity and difficulties with dynamic objects. While prior works have already proposed robust methods to address these problems [13, 9], there is still a major issue preventing the wide applicability of self-supervised depth estimators in safety-critical settings, such as autonomous driving. Darkness and adverse weather conditions (e.g., night, rain, snow, and fog) introduce noise in the pixel correspondences. As displayed in Figure 1, this is detrimental to the effectiveness of such methods, thereby requiring ad hoc solutions.

As shown in Figure 2, this problem is particularly severe at nighttime due to reflections (e.g., caused by streetlights and vehicle headlights), noise, and the general inability of the embedded cameras to capture details in dark areas. This leads to wrong depth estimates, which can be dangerous in safety-critical settings. A few pioneering works have already explored this problem, albeit with highly-

complex pipelines and significant architecture changes affecting inference as well [39, 38, 23, 34, 37], such as illumination-specific branches. Additionally, prior methods that can operate both at night- and daytime introduce a significant trade-off concerning the standard daytime performance [23, 38], highlighting the need for a new solution.

In adverse weather conditions such as rain, monocular models are similarly fooled by reflections and decreased visibility. However, rain introduces another problem. While radars are robust in such conditions, LiDARs become unreliable, as they introduce multi-path and the so-called blooming effects (Figure 2). In autonomous driving, since supervised depth estimation approaches learn from LiDAR data, this causes them to learn also such erroneous measurements, rendering them unreliable in rainy settings (Figure 1). Analogous issues occur with snow and fog. These problems are relatively unexplored, demanding new solutions.

Alarmingly, no general solution currently allows an image-based depth estimator to work reliably under all conditions. Since LiDAR can constitute a misleading training signal in adverse weather, and pixel correspondences are problematic too (e.g., at night), neither existing supervised [25] nor self-supervised [12, 35] techniques work well in such challenging settings. A straightforward solution for the supervised case would be using synthetic data [41, 31], as by simply not modeling the sensor issues, a simulator could produce perfect ground truth in adverse weather. However, this is not only unexplored, but it would introduce a series of problems, such as a substantial syn2real gap due to the difficulty of modeling challenging conditions realistically (requiring, e.g., domain adaptation).

In this paper, we address these open issues with a simple and effective solution that works reliably in a variety of conditions and for multiple types of supervision. We approach this challenging problem by considering the success of existing methods in standard illumination and weather settings [12, 14, 13, 9]. This motivated us to find a way for them to work also under challenging scenarios, exploiting what makes them learn depth effectively in ideal conditions. Our core idea is based on training the model by providing always valid training signals as if it was sunny or cloudy, even when samples with adverse conditions are given. We apply this general principle to both supervised and self-supervised depth estimation via a set of techniques to improve the model robustness and reduce the performance gap between standard and hard conditions. The main contributions of this paper can be summarized as follows:

- We show how estimating depth in adverse conditions (e.g., night and rain) is problematic for both self- and fully-supervised approaches, requiring new solutions.
- We propose md4all: a simple and effective technique to make standard models robust in diverse conditions.

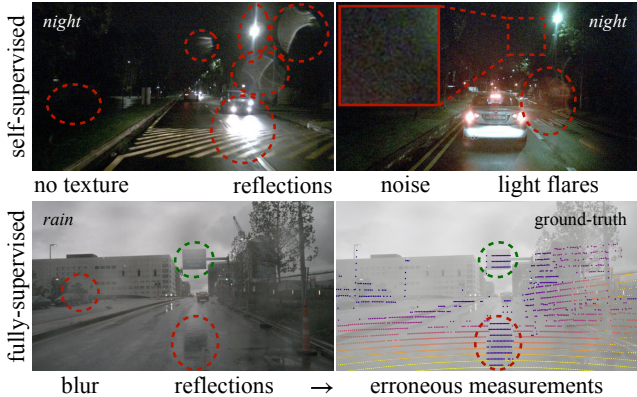


Figure 2. Detrimental factors to monocular depth estimation in difficult settings from nuScenes [4]. Self-supervised works have issues with textureless areas, reflections, and noise. Supervised ones learn artifacts from the ground truth sensor (LiDAR is shown).

- We apply our generic method to both fully- and self-supervised monocular settings.
- We generate and share open-source images in adverse conditions corresponding to the sunny and cloudy samples of nuScenes [4] and Oxford Robotcar [24].

With md4all, we substantially outperform prior solutions delivering robust estimates in a variety of conditions.

## 2. Related Work

### 2.1. Supervised Monocular Depth Estimation

The problem of estimating depth from a single color image is challenging due to the countless 3D scenarios that can produce the same 2D projection, making it an ill-posed problem. Nevertheless, significant progress has been made, thanks to the introduction of CNN-based architectures by Eigen et al. [6] and fully-convolutional networks with residual connections by Laina et al. [21] to estimate dense depth maps from monocular inputs. While many supervised methods have focused on directly regressing to depth measurements from LiDAR sensors (as in KITTI [10]) or RGB-D cameras (as in NYU-Depth v2 [32]), DORN [7] tackles the task in an ordinal manner. AdaBins [1] extended DORN via a linear combination of predictions across adaptive bins. Moreover, BTS uses a multi-stage local planar guidance [22] and P3Depth exploits coplanar pixels [25]. Others investigated the benefit of depth estimation while tackling other tasks, such as 3D object detection [18].

**Issues** While the supervision signal from 3D sensors is reliable in ideal conditions (e.g., sunny, cloudy), it severely degrades in photometrically challenging scenarios [20]. Outdoor, LiDAR sensors deliver erroneous measurements in adverse weather conditions, such as rain, snow and fog. As Jung et al. demonstrated indoor [20], training on an inexact ground truth leads depth models to learn the sensor ar-

tifacts and deliver wrong outputs. This problem is relatively unexplored outdoors, e.g., with rain. A few works investigated depth completion in simulated settings with LiDAR and radar in input [41] or event cameras and RGB [31]. In this paper, we explore this issue on AdaBins [1] and provide a simple solution to estimate depth reliably in diverse conditions, regardless of the sensor artifacts.

## 2.2. Self-Supervised Monocular Depth Estimation

To bypass the need for expensive LiDAR data, self-supervised methods employ view reconstruction constraints through stereo pairs [8, 11] or monocular videos [12, 47]. The latter utilizes motion parallax from a moving camera in a static environment [36] and requires simultaneous depth and camera pose transformation prediction. Significant advancements have been made since Zhou et al.’s pioneering video-based approach [47], including novel loss terms [12], network architectures that preserve details [13], the use of cross-task dependencies [19, 14], pseudo labels [26], vision transformers [45], uncertainty estimation [27], and 360 degrees depth predictions [16].

### 2.2.1 Solutions to Inherent Issues

**Scale ambiguity** Video-based methods predict depth up to scale, requiring median-scaling with ground truth data at test time [12]. Guizilini et al. [13] used the readily available odometry information to achieve scale awareness via weak velocity supervision on the pose transformation.

**Dynamic scenes** Due to the moving camera in a static world assumption [36], video-based methods have issues with dynamic objects, e.g., cars. To address this, Monodepth2 [12] uses an auto-masking loss on the static pixels, R4Dyn [9] adds weak radar supervision on the objects, and DRAFT [15] combines optical and scene flows.

**Darkness** Low visibility is detrimental to the losses used to learn depth because noise and lack of details prevent establishing pixel correspondences across the frames. DeFeat-Net [34] was among the first to mitigate this, with a cross-domain dense feature representation. ADFA [37] uses a generative adversarial network (GAN) to adapt nighttime features to daytime ones. R4Dyn [9] shows that radar is beneficial not only for dynamic objects but also at nighttime as a byproduct. RNW [39] reduces the irregularities at nighttime via, e.g., image enhancement and a GAN-based regularizer. ADIDS [23] uses separate networks for day and night images, partially sharing weights. ITDFA [44] is similar to ADFA, doing feature adaptation from night to day, with images generated with a GAN. WSGD [38] combines denoising with a lighting change decoder to predict per-pixel changes. While these works made significant steps towards solving the problem, they either have complex pipelines with dedicated branches for day and

night [23, 44], use additional sensors [9], suffer from a significant trade-off on the daytime performance [38], or are not meant to operate on multiple conditions, such as both day and night [37, 39, 44]. Therefore, an effective solution without inference complications is yet to be found.

**Adverse weather** As at nighttime, in adverse weather such as rain, fog, and snow, the limited visibility prevents establishing correct correspondences. Even fully-supervised approaches have issues in these settings [20]. So far, only a handful of works have explored depth estimation with adverse weather. ITDFA [44] requires an encoder for each condition and was not shown to work in both standard and adverse settings. R4Dyn [9] and MonoViT [45] are robust methods that delivered improvements also in adverse conditions as a side effect. Thus, this problem is largely unexplored, demanding a general solution.

Unlike prior works, in this paper, we propose a simple and effective solution enabling a standard monocular model to estimate depth in diverse conditions (e.g., day, night, and rain) without any difference at inference time compared to a common encoder-decoder pipeline [12]. Additionally, ours does not degrade the output quality in standard settings.

## 3. Method

In this paper, we enable a model to estimate depth reliably in diverse conditions (e.g., day, night, and rain). Displayed in Figures 3 and 4, our techniques exploit the effectiveness of existing approaches in standard conditions (e.g., daytime in good weather) to increase their robustness in adverse settings. Towards this end, we perform day-to-adverse image translation, train on the generated adverse samples, and learn only from valid training signals from the original day inputs. This simple idea is suitable to both self-supervised (Section 3.1) and supervised (Section 3.2) frameworks and is general to operate under various weather and illumination settings (including fog and snow).

### 3.1. md4all - Self-Supervised

We build upon a scale-aware video-based monocular method (Section 3.1.1). As described in Section 2.2.1, night and bad weather cause issues to self-supervised approaches. We address this with md4all by computing the losses only on the ideal samples corresponding to the hard ones given as input (Section 3.1.2). We then take this concept even further by distilling knowledge from a frozen self-supervised model trained only on the ideal samples (Section 3.1.3).

#### 3.1.1 Self-Supervised Baseline

We build on a standard video-based monocular depth baseline equivalent to the framework shown in Figure 4 when considering  $x = 0$  (i.e., no translation). We predict both the depth  $\hat{D}_t$  of a target frame and the pose transformations

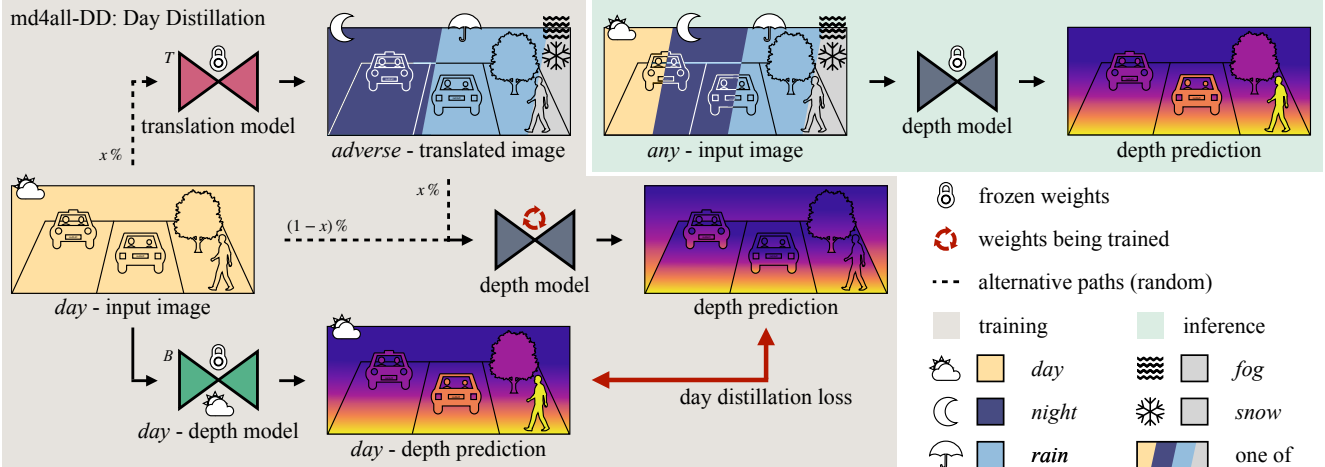


Figure 3. Our md4all-DD framework. The frozen *day* - depth model estimates on easy samples and provides guidance to another model fed with a mix of easy and translated inputs. Inference is done with a simple single model for both fully- and self-supervised md4all.

between the target  $I_t$  and source frames  $I_{s \in \{t-1, t+1\}}$ , with which we warp the source into a reconstructed target view. As in [12, 13], a loss is computed on the appearance shift between  $I_t$  and the reconstruction [47], alongside the structural similarity [40]. Following [12], we account for partial occlusions via the minimum reprojection error  $\mathcal{L}_p$ , and we ignore static pixels. Another loss  $\mathcal{L}_s$  promotes smoothness and preserves edges [11].  $\mathcal{L}_p$  and  $\mathcal{L}_s$  are calculated at all decoder scales, upsampled to the input size [12].

So far, this is equivalent to Monodepth2 [12]. Then, we add the weak velocity supervision  $\mathcal{L}_v$  to achieve scale-awareness [13] and allow consistent predictions, beneficial when distilling knowledge between different models.

**Architecture** Unlike previous works having specialized branches [23, 44], we leave the architecture unchanged (e.g., [12]). Instead, we act on the training process. Our approach is general and not bound to a specific architecture.

### 3.1.2 md4all-AD: Always Daytime, No Bad Weather

Our md4all-AD configuration is shown in Figure 4. The core idea is learning from easy samples, even when given challenging ones (e.g., night) as if it was always daytime with good visibility (i.e., sunny or cloudy). This allows using the same established losses described in Section 3.1.1, which would otherwise fail with difficult inputs.

**Day-to-adverse translation** To achieve the above, we need easy samples corresponding to the challenging ones. This means having paired images  $(e_i, h_i^c)$ , with  $e_i \in E$  and  $E$  being the set of easy samples (i.e., sunny or cloudy),  $h_i^c \in H$  with  $H$  the set of the difficult samples from the conditions of interest  $c \in C$  (e.g., snow). While an image translation method could convert the training  $H$  into easy ones, removing information is easier than adding it. Therefore, we generate  $H$  from  $E$  (e.g., turning  $e_i$  into nighttime).

Specifically, for each  $e_i$  and each condition  $c$  we aim to improve (e.g., night and rain), we obtain  $h_i^c = T^c(e_i)$ . We do this with  $c$  image translation models  $T^c$  trained at an earlier stage, increasing the training set size by  $C \times E$ .

**Training scheme** We then train depth and pose models as shown in Figure 4. During training, we feed to the depth model  $m_i$ , which is either  $h_i^c$  (for  $x\%$  of the inputs, as a random mix of  $c$ ) or  $e_i$  from the pre-existing training data. Additionally, we normalize the inputs depending on the recording time (i.e., day/night) to learn robust features agnostic of the input condition. The Appendix shows how performing this step only during training delivers similar results. Then, in the case of particularly noisy night samples (e.g., nuScenes [4]), we augment the inputs with heavy noise. The pose model always takes the sequence  $[e_{i-1}, e_i, e_{i+1}]$ , corresponding to  $m_i$ . If fed  $h_i^c$ , the pose network would have issues assessing the pixel correspondences.

**Learning in all conditions** Computing the losses  $\mathcal{L}_p$  and  $\mathcal{L}_s$  on  $h_i^c$  would lead to issues because of the difficulty of establishing correspondences in adverse conditions (Section 2.2.1). For this reason, training on  $E$  and deploying on  $H$  is more effective than training on both (Section 4.2), proving the limitations of standard methods. Our solution to this challenging problem is relatively simple: as shown in the figure, we provide a reliable training signal by always calculating the losses on  $E$ . Specifically, they are always computed on  $e_i$ , even when the depth model is fed with  $h_i^c$  ( $x\%$ ). This constructed setting constitutes the ideal condition in which the losses  $\mathcal{L}_p$  and  $\mathcal{L}_s$  are already proven successful [12], eliminating the source of the issues. This leads the depth model to learn to extract robust features, regardless of whether the input belongs to  $E$  or  $H$ .

**Inference** After training depth and pose models, the latter is discarded, while our depth model is a simple encoder-decoder capable of estimating depth in multiple conditions.



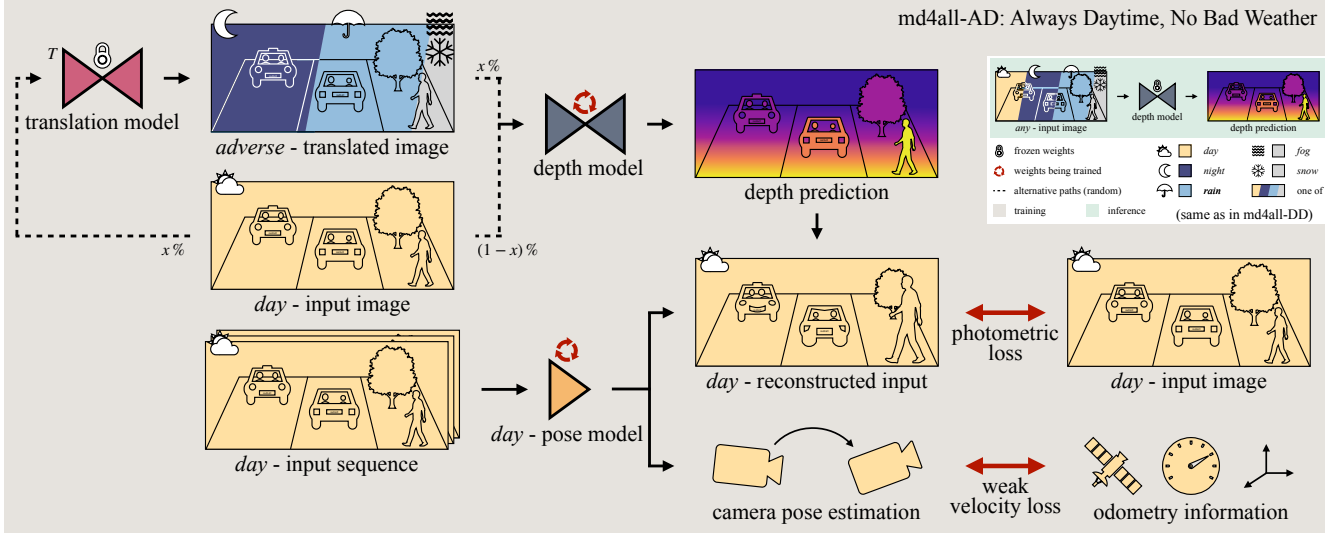


Figure 4. Our self-supervised md4all-AD framework. With  $x = 0$ , it is equivalent to the *day* - depth model in Figure 3 and the baseline. The depth model is trained with a mix of easy and translated samples, while the training signal is always from the easy ones.

As shown at the top of Figure 3, since we do not apply any architectural modification, at inference time, we predict depth with the same model through the same model parameters, regardless of the input condition. While dedicated models or branches may lead to better performance, switching between them is not always trivial, e.g., at dusk or with light rain. Therefore, we opted for a single monocular model, which does not penalize inference time compared to the same model trained only on  $E$ .

### 3.1.3 md4all-DD: Day Distillation

We take md4all-AD (Section 3.1.2) to the next level by simplifying the training scheme with md4all-DD. The core idea of md4all-DD is the same as for md4all-AD: we aim to learn depth only from  $E$ , pretending that the conditions  $C$  detrimental for the losses never occur.

Our md4all-DD framework mimics model estimates in ideal settings  $E$ , regardless of the difficulty of the input. As shown in Figure 3, we achieve this via knowledge distillation from a depth network  $B$  (baseline) trained at an earlier stage on  $E$  to a new depth model  $DD$  for both easy and adverse scenarios (i.e.,  $E$  and  $H$ ). The latter is fed  $m_i$ , i.e., the same mix of  $e_i$  and  $h_i^c$  as in md4all-AD (Section 3.1.2), while the former is given only  $e_i$ .  $DD$  is optimized solely through the following objective:

$$\mathcal{L}_d = \frac{1}{N} \sum_{j=1}^N \frac{|DD(m_i)_j - B(e_i)_j|}{DD(m_i)_j} \quad (1)$$

where  $N$  is the number of pixels,  $DD(m_i)$  is  $DD$ 's depth prediction on  $m_i$  (i.e., an easy or hard sample), and  $B(e_i)$  is  $B$ 's estimation on  $e_i$  (i.e., an easy sample).  $DD$  learns to

follow  $B$  at the output level, even when fed the problematic  $h_i^c$ , without being affected by the detrimental factors occurring in adverse settings. Inference is unchanged.

## 3.2. md4all - Supervised

Learning depth from a 3D sensor in adverse conditions exposes issues inherent to the sensor and the way it measures depth [20]. With bad weather (e.g., rain), LiDARs provide erroneous measurements (Figure 2), so learning from their signal means copying their artifacts as well (Figure 1). This has been ignored so far for monocular depth.

Regardless of the input, we address the sensor issues by learning from  $E$ . Analogously to the self-supervised setting, we use image pairs  $(e_i, h_i^c)$  and specify our method as md4all-AD following the self-supervised definition (Section 3.1.2), except for the supervision signal. Thus, we train the depth model with  $m_i$  and learn from the LiDAR signal of  $e_i$ . Thus, the supervision is from artifact-free data in ideal conditions  $E$ , such that the models never experiences the sensor issues. As in the self-supervised setup, the inference is unchanged. While md4all-DD (Section 3.1.3) also applies to the supervised case, using AD is more reasonable since reliable ground truth data from  $e_i$  is available.

## 4. Experiments and Results

### 4.1. Experimental Setup

**Datasets and metrics** We used two public driving datasets containing various illumination and weather conditions: nuScenes [4] and Oxford RobotCar [24]. **nuScenes** is a challenging large-scale dataset with 15h of driving in Boston and Singapore, diverse scenes, and difficult conditions. We distinguished good visibility (i.e., *day-clear*),

Method	sup.	tr.data	<i>day-clear</i> – nuScenes			<i>night</i> – nuScenes			<i>day-rain</i> – nuScenes		
			absRel	RMSE	$\delta_1$	absRel	RMSE	$\delta_1$	absRel	RMSE	$\delta_1$
Monodepth2 [12]	M*	<i>a: dnr</i>	0.1477	6.771	85.25	2.3332	32.940	10.54	0.4114	9.442	60.58
Monodepth2 [12]	M*	<i>d</i>	0.1374	6.692	85.00	0.2828	9.729	51.83	0.1727	7.743	77.57
PackNet-SfM [13]	Mv	<i>d</i>	0.1567	7.230	82.64	0.2617	11.063	56.64	0.1645	8.288	77.07
R4Dyn w/o r in [9]	Mvr	<i>d</i>	0.1296	6.536	85.76	0.2731	12.430	52.85	0.1465	7.533	80.59
R4Dyn [9] (radar)	Mvr	<i>d</i>	<b>0.1259</b>	<b>6.434</b>	<b>86.97</b>	0.2194	10.542	62.28	<b>0.1337</b>	<b>7.131</b>	<b>83.91</b>
RNW [39]	M*	<i>dn</i>	0.2872	9.185	56.21	0.3333	10.098	43.72	0.2952	9.341	57.21
[ours] baseline	Mv	<i>d</i>	0.1333	6.459	85.88	0.2419	10.922	58.17	0.1572	7.453	79.49
[ours] md4all-AD	Mv	<i>dT(nr)</i>	0.1523	6.853	83.11	0.2187	9.003	68.84	0.1601	7.832	78.97
[ours] <b>md4all-DD</b>	Mv	<i>dT(nr)</i>	0.1366	6.452	84.61	<b>0.1921</b>	<b>8.507</b>	<b>71.07</b>	0.1414	7.228	80.98
AdaBins [1]	GT	<i>a: dnr</i>	0.1384	5.582	81.31	0.2296	7.344	63.95	0.1726	6.267	76.01
[ours] <b>md4all-AD</b>	GT	<i>dnT(r)</i>	<b>0.1206</b>	<b>4.806</b>	<b>88.03</b>	<b>0.1821</b>	<b>6.372</b>	<b>75.33</b>	<b>0.1562</b>	<b>5.903</b>	<b>82.82</b>

Table 1. Evaluation of self- and GT-supervised methods on the nuScenes [4] validation set. Supervisions (sup.): M: via monocular videos, \*: test-time median-scaling via LiDAR, v: weak velocity, r: weak radar, GT: via LiDAR data. Training data (tr.data): *d*: *day-clear*, *T*: translated in, *n*: *night* (incl. *night-rain*), *r*: *day-rain*, *a*: *all*. Visual support: 1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup> best. More conditions and metrics in the Appendix.

*night* (including *night-rain*), and *day-rain*. We used the official split following R4Dyn [9], with 15129 training images (with synced sensors), and 6019 validation ones (of which 4449 *day-clear*, 602 *night*, and 1088 *rain*). **RobotCar** was collected in Oxford, UK, by traversing the same route multiple times in a year. It features a mix of *day* and *night* scenes. We followed the split and setup of WSGD [38], with 16563 *day* training samples and 1411 test ones (with synced sensors, of which 709 *night*). While we focused on night, rain, sun, and overcast, the Appendix shows preliminary results with fog and snow from the **DENSE** dataset [2]. We report on the standard metrics and errors up to 50m for RobotCar as in [38], and 80m for nuScenes as in [9]. More results can be found in the Appendix.

**Implementation details** Our self-supervised models use a ResNet-18 backbone [17] and learn from an image triplet sized 576x320 for nuScenes and 544x320 for RobotCar. The supervised model and md4all-DD are given only one keyframe. At inference time, all models take a single RGB input. We set  $x = |C|/(|C|+1)$  %, with  $|C|$  being the number of the adverse conditions of interest  $C$ , e.g.,  $x = 66%$  for a model to work with *rain*, *night* and *day*, and within  $x%$  we used equally distributed data among  $C$ . So, our models see an equal amount of inputs for each condition. We used the same hyperparameters as Monodepth2 [12] and AdaBins [1] for self- and fully-supervised models, respectively. All models were trained on a single 24GB GPU.

**Image translation** We translated each  $e_i$  image to  $h_i^c$ . Diffusion models [28, 29] are not suitable due to the lack of already paired images. Datasets with multiple drives on the same roads [5, 30, 24] do not solve this issue due to the lack of synchronization and environmental changes. So we opted for GANs. For each condition  $c$ , we used a ForkGAN model [46]  $T^c$  to translate all *day-clear* training samples  $E$  of nuScenes, with  $c \in C = \{\textit{night}, \textit{rain}\}$ . We trained Fork-

GAN on BDD100K [43] and fine-tuned it on the nuScenes training set. For RobotCar, we used  $T^c$  to translate all *day* samples  $E$  into *night* ones. RobotCar contains more *night* samples than nuScenes, so we trained  $T^c$  directly on its training set. We share publicly all generated  $h_i^c$  images.

**Prior works and baselines** We compared ours with a variety of works [9, 12, 13, 39, 38, 23, 34, 1]. We applied ours on the self-supervised Monodepth2-based baseline of Section 3.1.1 and the fully-supervised AdaBins [1].

## 4.2. Quantitative Results

**Night – nuScenes** In Table 1, we report results for nuScenes [4] across various settings. Night samples present strong noise levels and reflections that are detrimental for self-supervised models (Figure 2), causing the absRel errors of most methods to double from ideal conditions (i.e., *day-clear*) to *night*. The difficulty of learning from night inputs is evident comparing Monodepth2 [12] trained only on *day-clear* ( $d$ ) against *all* conditions ( $a$ ), with the latter severely underperforming. PackNet [13] improved at *night* and *rain*, albeit doing worse in standard settings, possibly due to its large model and the relatively small dataset. PackNet’s velocity supervision also helped over Monodepth2 (md2) with our baseline. Thanks to the extra radar signal, R4Dyn [9] delivered significant improvements, although at *night*, only adding radar in input was beneficial over md2. md2 trained only on *day-clear* data outperformed RNW’s complex pipeline [39]. We retrained RNW on the official split (the authors reported an absRel of 0.3150 at *night* on their split [39]). Remarkably, despite being based on the same model as md2, at *night*, our simple techniques reduced absRel by 32% and relatively increased  $\delta_1$  by 37% (DD). Our md4all also outperformed the radar-based R4Dyn at *night*. This is thanks to the ability of our method to extract robust features from monocular data even in the dark.

Method	source	sup.	tr.data	<i>day</i> – RobotCar				<i>night</i> – RobotCar			
				absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
Monodepth2 [12]	[ours]	M*	<i>d</i>	0.1196	0.670	<b>3.164</b>	86.38	0.3029	1.724	5.038	45.88
DeFeatNet [34]	[38]	M*	<i>a: dn</i>	0.2470	2.980	7.884	65.00	0.3340	4.589	8.606	58.60
ADIDS [23]	[38]	M*	<i>a: dn</i>	0.2390	2.089	6.743	61.40	0.2870	2.569	7.985	49.00
RNW [39]	[38]	M*	<i>a: dn</i>	0.2970	2.608	7.996	43.10	0.1850	1.710	6.549	73.30
WSGD [38]	[38]	M*	<i>a: dn</i>	0.1760	1.603	6.036	75.00	0.1740	1.637	6.302	75.40
[ours] baseline	[ours]	Mv	<i>d</i>	0.1209	0.723	3.335	86.61	0.3909	3.547	8.227	22.51
[ours] <b>md4all-DD</b>	[ours]	Mv	<i>dT(n)</i>	<b>0.1128</b>	<b>0.648</b>	<u>3.206</u>	<b>87.13</b>	<b>0.1219</b>	<b>0.784</b>	<b>3.604</b>	<b>84.86</b>

Table 2. Evaluation of self-supervised works on the RobotCar [24] test set. Trailing 0 added to the values from [38]. Notation from Table 1.

**Night – RobotCar** In Table 2, we report results for RobotCar [24]. Here we compare with various approaches that also target depth estimation in challenging conditions [34, 23, 39, 38]. They all focus on *night* issues, tested here. Our md4all outperforms them all across the board, with substantially better estimates at *night* than theirs during the *day*: the previous best WSGD [38]’s *day* absRel error is 45% higher than ours at *night*. This is thanks to the simplicity of our approach, which does not rely on complex architectures, but makes existing models robust in adverse conditions by changing their input and training signals.

**Rain – nuScenes** Rain is less problematic than darkness due to the lack of cues in the latter. Results are shown in Table 1, with all methods performing better with *rain* than at *night*. Our self-supervised monocular md4all-DD significantly improved over Monodepth2 and the baseline, performing close to the radar-based R4Dyn [9].

**Fully-supervised** Table 1 reports also results in supervised settings. LiDAR data is reliable in the dark, so *night* scenes are less of an issue. Instead, *rain* inputs are particularly interesting for supervised works due to the reflection issues shown in Figures 1 and 2. For supervised settings, we applied our method on AdaBins [1]. It is to be considered that LiDAR artifacts may have an impact on the *rain* values, such that perfect estimates would not score perfectly because the ground truth is wrong (Figure 2). So, while we can assess the improvements of md4all at handling the blur caused by raindrops, we cannot correctly quantify its impact on eliminating the artifacts. Therefore, these comparisons are more meaningful when considered alongside qualitative outputs (Figure 5). Our supervised md4all performed better than AdaBins both quantitatively and qualitatively, eliminating the dependency on the sensor artifacts. Additionally, thanks to the strong regularization introduced by the translated samples, our model generalizes significantly better than the standard AdaBins, leading to vast improvements across the board, also at *night*. Training on the sparse LiDAR signal of nuScenes [4] (Figures 2 and 5) can lead to overfitting. Ours is a beneficial data augmentation technique, adding diversity to the training, as the model is shown  $|C| + 1$  variations of each *day-clear* input.

**Day(-clear)** While we do not include any modification addressing standard conditions, we still see improvements over the baselines across both datasets and supervision types (Tables 1 and 2). This is due to the training mix of easy and translated samples acting as a strong data augmentation and regularization technique. Since the same weights are optimized on all conditions, they learn to extract robust features which are beneficial also with good visibility. Instead, RNW [39] is meant for operating only at night.

**All conditions** Remarkably, across all tested conditions, md4all significantly improves over Monodepth2 and AdaBins on which we applied it, without the need for specialized branches (Tables 1 and 2). There is no trade-off introduced when training our unique md4all model for multiple conditions, as the scores and errors remain equivalent or even improve compared to training only in ideal settings. This proves the effectiveness and generality of our simple ideas. The Appendix includes preliminary results with *snow* and *fog* on the challenging DENSE dataset [2].

**AD and DD** Our md4all delivers improvements both as DD and AD (Tables 1 and 2). While the two are applicable under both supervisions, available and reliable ground truth alongside the *day-clear* data makes AD more suitable for supervised setups. DD works better than AD in self-supervised settings thanks to the simplified training scheme and the guidance of our strong baseline.

**Robustness against translations** In Table 3, we assess the impact of the quality of image translation on our method. While the selected ForkGAN [46] translates better than CycleGAN [48], it does not give perfect outputs either (Appendix). Since we use the translations to learn robust features, their imperfections even help our model’s robustness.

Method	<i>avg/all</i>	<i>day</i>	<i>night</i>
[ours] w/ CycleGAN [48]	0.1244	0.1159	0.1328
[ours] w/ ForkGAN [46]	<b>0.1174</b>	<b>0.1128</b>	<b>0.1219</b>
[ours] w/ degraded ForkGAN	0.1213	0.1159	0.1266

Table 3. Robustness of md4all-DD against translations from different GANs. Evaluation of absRel on the RobotCar [24] test set.



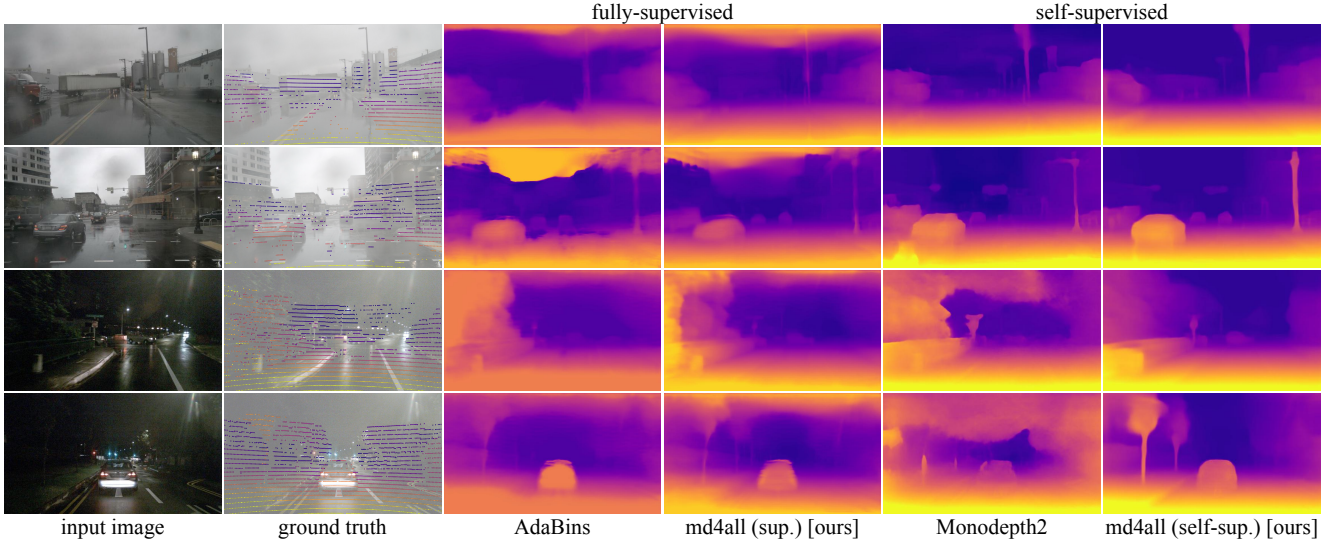


Figure 5. Comparison on nuScenes [4] between fully-sup. AdaBins [1] w/o and w/ ours, and self-sup. Monodepth2 [12] w/o and w/ ours.

business by making it harder to recover information for the depth task, as the translations act as data augmentation and regularization. The table confirms the robustness of md4all, performing similarly regardless of which GAN is used, even when degrading 10% of the inputs via random erasing.

### 4.3. Qualitative Results

Qualitative comparisons in Figures 5 and 6 confirm the quantitative findings, with our md4all delivering improved estimates in both adverse and standard conditions. On nuScenes [4] (Figure 5), unlike the baselines, both our models correctly identified the truck in the first rainy sample. As shown in Figure 2, rain leads to artifacts in the LiDAR ground truth, which cause the standard fully-supervised AdaBins [1] to learn them and estimate the road wrongly. Our supervised md4all exhibits no such artifacts as it was not trained with the problematic rainy samples but rather on our translated ones, which have reliable ground truth. Instead, self-supervised methods have issues at *night*. While Monodepth2 [12] could identify critical elements of the scenes (e.g., car and sign), its difficulties in extracting information in the dark are evident. Monodepth2 had fewer issues

with brighter *night* samples, as shown in the Appendix. Our self-supervised md4all delivered sharp estimates, identifying even the two trees on the left side of the bottom input, which are particularly hard to see. For RobotCar [24] (Figure 6), we compared on the same samples displayed by WSGD in their paper [38]. As in Table 2, our md4all delivered better and sharper estimates in both conditions, correctly estimating the people’s distance.

**Limitations** md4all improves in all tested conditions, but DD may propagate errors from the baseline. Thus, a stronger baseline would help. Despite the robustness against translations (Table 3), GANs [46] could be problematic. Better translations would help eliminate the domain gap, as seen with RobotCar (Table 2). GANs require many adverse images for training. Hard-to-distinguish data distributions (e.g., light snow vs. overcast) may create problems. md4all is applicable to stereo-based models too, but only given consistent translations for the stereo images. Future work may focus on eliminating the dependency on the GAN. Furthermore, md4all does not address the issue of dynamic objects, so flow [15] or weak radar supervision [9] may be beneficial, albeit adding complexity. The core ideas of this work can be extended to other tasks.

The **Appendix** includes a variety of extra results, e.g., experiments with *snow* and *fog*, and sample translations.

## 5. Conclusion

We presented the simple and effective md4all, enabling a single monocular model to estimate depth robustly in both standard and challenging conditions (e.g., night, rain). We showed md4all delivering significant improvements under both fully- or self-supervised settings, overcoming the detrimental factors that make adverse conditions problematic.

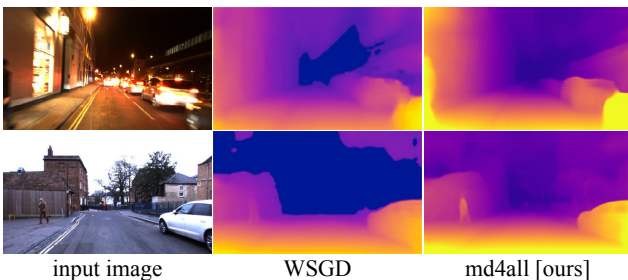


Figure 6. Comparison on RobotCar [24] samples between our self-supervised and WSGD [38]. Outputs of WSGD are from [38].



## A. Supplementary Material

The source code of our method, the main trained models reported in the experiments, and the generated translated images are publicly available at <https://md4all.github.io>.

This appendix includes additional details and results. Sections A.1 and A.2 include additional information on the method and the experimental setup, while Sections A.3 and A.4 introduce more results, quantitatively and qualitatively, respectively.

In particular, this appendix is organized as follows:

- Section A.1.1 includes details about our supervised framework.
- Section A.1.2 adds details about our self-supervised baseline.
- Section A.1.3 describes the noise and time-dependent normalization used for our self-supervised models.
- Section A.2.1 further reports details about the experimental setup for image translation.
- Section A.2.2 includes details on the setup used for the nuScenes dataset.
- Section A.2.3 includes details on the setup used for the RobotCar dataset.
- Section A.2.4 adds details about the experimental setup for prior works.
- Section A.3.1 reports a detailed ablation study of our method on both nuScenes and RobotCar.
- Section A.3.2 adds preliminary results with snow and fog on the DENSE dataset.
- Section A.3.3 analyzes the effect of different data distributions among the conditions during training on RobotCar.
- Section A.3.4 compares different configurations of our supervised framework on nuScenes.
- Section A.3.5 looks into quantitative results with rain at nighttime and averages across the various conditions on nuScenes.
- Section A.3.6 compares methods on the test set of nuScenes.
- Section A.3.7 analyzes the performance of the methods at varying distances from the ego vehicle, both on nuScenes and RobotCar.
- Section A.4.1 reports qualitative results of our self-supervised method on nuScenes.

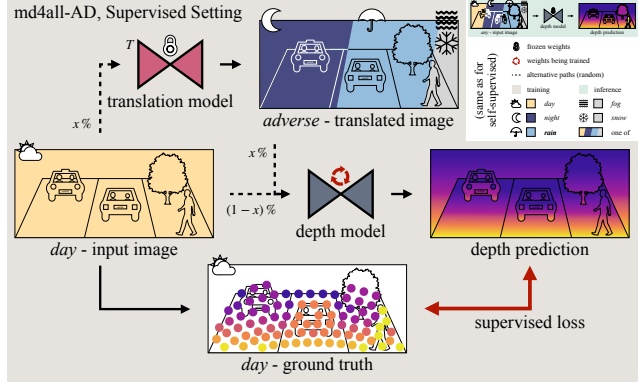


Figure 7. Apart from the type of supervision, our md4all-AD supervised framework works similarly to the md4all-AD self-supervised framework described in Section 3.1.2 and Figure 4, with Always Daytime, No Bad Weather. The depth model is trained with a mix of easy and translated samples, while its supervision is obtained from the ground truth data corresponding only to the easy samples. As in the self-supervised case, inference time is unchanged and performed with a single depth model without specialized branches (top right).

- Section A.4.2 adds qualitative results of our fully-supervised method on nuScenes.
- Section A.4.3 discusses qualitative results of our self-supervised method on RobotCar.
- Section A.4.4 looks into failure cases of our self- and fully-supervised methods, exemplified on nuScenes.
- Section A.4.5 analyzes images generated via image translations for both nuScenes and RobotCar.
- Section A.5 lists attempted and alternative approaches that did not work.

### A.1. Additional Details on the Method

#### A.1.1 Supervised md4all

In the main paper, we mostly focused on the more complex self-supervised setting (Sections 3.1.2 and 3.1.3), and we extended our method to the supervised setup (Section 3.2), making it the first depth estimation work to explore and address bad weather in supervised monocular settings.

Supervised models learn directly from the ground truth (e.g., LiDAR data). However, in adverse conditions (e.g., rain), the ground truth becomes unreliable (Figure 2). Our supervised md4all aims to eliminate the sources of unreliability in the ground truth by providing a reliable signal in all conditions. As shown in Figure 7, we achieve this with the same principles described for the self-supervised settings: having a single depth model learn robust features agnostic of the condition in input by feeding a mix of easy and hard samples, with the ground truth always corresponding to the

easy samples. Therefore, we use the same image translation model to generate adverse images corresponding to the easy ones in the training data. Then, we train the depth model with a mix of original easy and generated adverse inputs. Unlike the self-supervised settings where the training signal came from a pre-trained baseline model (md4all-DD) or the photometric losses (md4all-AD), the training signal is obtained directly from the ground truth data for supervised methods. When translating an image  $e_i$  to adverse conditions  $h_i^c$ , we use as ground truth for  $h_i^c$  the LiDAR data corresponding to  $e_i$ .

We associate this supervised method with our AD configuration since no distillation from a pre-trained model occurs (unlike for DD). Moreover, the depth model is trained in the same way as its baseline, i.e., via the ground truth, in an Always Daytime, no Bad Weather fashion, similarly to our self-supervised AD model (Section 3.1.2). Furthermore, the translated images should be used only for those conditions that render the ground truth unreliable. Therefore, for the experiments, we translated the inputs from *day-clear* to *day-rain*, since the ground truth is unreliable with rain (Figure 2). Still, we used the original *night* inputs since the ground truth is reliable at night (Figure 5).

### A.1.2 Self-Supervised Baseline

In this section, we further describe the loss functions used for the baseline of Section 3.1.1. Such baseline is equivalent to Monodepth2 [12] made scale-aware through weak velocity supervision from Guizilini et al. [13].

The photometric loss is the combination of  $\mathcal{L}_1$ -loss and SSIM [40], as done in [11]:

$$\begin{aligned} \mathcal{L}_1(I_t, \hat{I}_t) &= \left\| I_t - \hat{I}_t \right\|_1 \\ \mathcal{L}_{\text{SSIM}} &= 1 - \text{SSIM} \left( I_t, \hat{I}_t \right) \\ pe \left( I_t, \hat{I}_t \right) &= (1 - \alpha) \mathcal{L}_1(I_t, \hat{I}_t) + \frac{\alpha}{2} \mathcal{L}_{\text{SSIM}}(I_t, \hat{I}_t) \end{aligned} \quad (2)$$

where  $\alpha = 0.85$  is a weight to balance between the two terms. Furthermore, similarly to [12], we account for partial occlusions by only considering the minimum reprojection error:

$$\mathcal{L}_p \left( I_t, \hat{I}_{s \rightarrow t} \right) = \min_s pe \left( I_t, \hat{I}_{s \rightarrow t} \right). \quad (3)$$

Moreover, following the so-called auto-mask from Monodepth2 [12], we automatically mask out the pixels that do not change appearance across different frames:

$$M_a = \min_s \mathcal{L}_p \left( I_t, I_s \right) > \min_s \mathcal{L}_p \left( I_t, \hat{I}_{s \rightarrow t} \right). \quad (4)$$

Therefore, the photometric loss is only computed in the areas where  $M_a = 1$ . Additionally, to encourage local

smoothness and preserve sharp edges, we use the following term from [11]:

$$\mathcal{L}_s(I_t, d_t^*) = \frac{1}{N} \sum_{p \in N} \sum_{i \in x, y} |\partial_i d_t^*(p)| e^{-|\partial_i I_t|} \quad (5)$$

where  $|\cdot|$  is the absolute value computed element-wise,  $\partial_x$  and  $\partial_y$  are the gradients in x and y directions, and  $d_t^* = d_t^*/\bar{d}_t^*$  is the inverse of the depth prediction normalized by the mean.

As already described in Section 3.1.1, we follow [13] by using a weak velocity supervision  $\mathcal{L}_v$  to achieve scale-awareness. This is defined as:

$$\mathcal{L}_v(\hat{\mathbf{t}}_{t \rightarrow s}, \mathbf{t}_{t \rightarrow s}) = \left| \|\hat{\mathbf{t}}_{t \rightarrow s}\|_2 - \|\mathbf{t}_{t \rightarrow s}\|_2 \right| \quad (6)$$

where  $\hat{\mathbf{t}}_{t \rightarrow s}$  and  $\mathbf{t}_{t \rightarrow s}$  are the predicted and ground truth pose translations, respectively, which can be easily obtained from the available odometry information, through the ego vehicle speed and the time interval across frames.

### A.1.3 Noise and Normalization

We did not apply either of these two techniques in the supervised setting, i.e., where we applied our method on Ad-aBins [1], since the LiDAR ground truth provides a strong signal which already enables such robust feature extraction.

For the self-supervised models, we normalize the inputs at training time depending on the time of the day (i.e., day and night). Towards this end, we precompute the mean and variance of the pixel values across the two conditions throughout the dataset and normalize the inputs accordingly. In Tables 4 and 5, we show how this time-dependent normalization has a positive impact at training time, as it aligns the features in a condition-agnostic manner. Additionally, we show that this normalization can be avoided at inference time for similar results. Avoiding it ensures that the operations executed across all conditions are identical at inference time. Thus, after deployment, our method does not require any knowledge about the current weather and illumination settings, which may be hard to define and may intersect with other conditions (e.g., wet ground without rain or dusk). Nevertheless, due to the relatively small difference during inference, we used time-dependent normalization for self-supervised models unless otherwise noted.

Furthermore, in the case of camera sensors delivering significant noise levels (e.g., nuScenes [4]), we augment the inputs of self-supervised models with heavy noise. The noise is randomly applied to 50% of the inputs, regardless of their condition. This helps to learn more robust features. When the noise is used, we compute the losses on the samples without noise. Specifically, we generated the noise by adding to the image a random pattern following the uniform distribution [0.005, 0.05], then clamped the pixel values to

[0, 1], thereby ensuring that the input remains within a valid range.

## A.2. Additional Details on the Experimental Setup

### A.2.1 Day-to-adverse Translation

For the experiments, we focused on two adverse conditions in nuScenes [4] (i.e., *rain* and *night*) and one in RobotCar [24] (i.e., *night*), alongside the standard conditions *day-clear / day*. Towards this end, we trained two different ForkGAN [46] models for nuScenes, one for each condition, and one for RobotCar, to enable translations from *day-clear* to each challenging condition, tailored to each dataset. For the RobotCar dataset, we trained the GAN using the 34128 daytime samples from the scene 2014-12-09-13-21-02 and the 32585 nighttime samples from 2014-12-16-18-44-24. The dataset offered enough samples to train the image translation model thanks to the high frame rate. Instead, the nuScenes dataset only provides 6951 samples for *day-rain* and 4706 for *night*, which are insufficient for the GAN to learn such day-to-adverse translation. Therefore, to learn the transition from *day-clear* to *day-rain*, we additionally used all *day-rain* samples from the nuImages dataset [4] resulting in a total number of 19857 *day-rain* frames. We balanced this with the 19685 *day-clear* images of the nuScenes training set. Since the nuImages dataset does not provide any metadata about the weather condition, we manually labeled all its samples with their respective weather condition. Nevertheless, *night* samples are insufficient in nuScenes and nuImages (14302) to train a GAN. For this reason, we first trained the *day-clear* to *night* translation model on BDD100K [43], which includes 36728 *day* and 27971 *night* images in its training set. Then, we fine-tuned it on the available nuScenes *night* samples from the training set.

### A.2.2 nuScenes

For the depth experiments on nuScenes [4], we followed the setup of R4Dyn [9], using the official data splits and evaluating up to 80 meters comparing the predictions with a single LiDAR scan. As in R4Dyn, we discarded static frames (i.e., where the ego vehicle is stationary) for self-supervised models. While a single scan is highly sparse compared to the dense depth prediction, it limits the artifacts introduced by accumulating multiple scans over time for denser ground truth (e.g., due to moving objects and changing perspectives). We augmented the inputs with heavy noise for self-supervised models to mimic that in the night samples. For the supervised setting, learning from such a sparse signal means reducing the workload needed for producing the ground truth, albeit rendering it more challenging. As it is standard for supervised setups, the models do not learn the depth of the unreachable areas for the ground truth sensor

(e.g., the sky for LiDAR). All qualitative images and quantitative results reported in the main paper and this supplementary material are from the validation set unless otherwise noted (e.g., test set in Table 14).

### A.2.3 RobotCar

For the experiments on RobotCar [24], we followed the setup of WSGD [38] using the six sequences in the 2014-12-09-13-21-02 traversal as daytime samples, and the six sequences in the 2014-12-16-18-44-24 traversal as nighttime ones. Since the peculiarity of RobotCar is that it was recorded by driving over the same route multiple times over a year, a training-test split with non-overlapping drives is required to properly assess the models' generalization capabilities. Therefore, we used the split provided by WSGD.

As in [39], we used the left images of the front stereo-camera (Bumblebee XB3), of which we removed the bottom 20% (i.e., ego vehicle bonnet), and the ground truth data from the LMS front LiDAR sensor. We used the official toolbox to accumulate multiple LiDAR scans and project them to the input images. Towards this end, we used visual odometry, as recommended by the official documentation of the dataset, and a time margin of  $\pm 4e6$  from the origin timestamp, as in [39]. As commonly done for self-supervised methods, we discard static frames thresholding the translation provided by the visual odometry. We did not apply heavy noise for RobotCar as the night samples did not exhibit it. Furthermore, since the RobotCar camera occasionally suffers from inconsistent illumination across neighboring frames, we discarded these too. Specifically, we removed all triplets where the keyframe's mean RGB value is  $\geq 0.9$ , or the RGB mean value difference between two consecutive frames is  $> 0.05$ . In addition, only the images with a corresponding LiDAR ground truth could be evaluated.

For the experiment with degraded translations via random erasing (Table 3), we applied it randomly to 10% of the inputs, with a patch sized randomly between 5% and 10% of the input dimensions, placed randomly within the image, with an aspect ratio between 0.3 and 3.3. When applying random erasing, the performance of md4all marginally improved in terms of  $\delta_1$  by 0.32% on *all* (absRel slightly decreased as shown in Table 3), thanks to the augmentation and regularization effect introduced by the patches. Throughout the main paper and this supplementary material, all qualitative images and quantitative results are from the test set defined by WSGD.

### A.2.4 Prior Works

For prior works on RobotCar, we reported the values computed by Vankadari et al. [38], who retrained RNW [39] on a non-overlapping split (which inherently reduced

ID	Method	tr.data	<i>day-clear</i> – nuScenes			<i>night</i> – nuScenes			<i>day-rain</i> – nuScenes		
			absRel	RMSE	$\delta_1$	absRel	RMSE	$\delta_1$	absRel	RMSE	$\delta_1$
A0	md2 [12], <i>all</i>	<i>a: dnr</i>	0.1477	6.771	85.25	2.3332	32.940	10.54	0.4114	9.442	60.58
A1	md2, <i>n</i> real	<i>dn</i>	0.1345	6.575	85.47	2.4536	34.295	11.71	0.1753	7.701	77.13
A2	md2, <i>n</i> transl. <sub>15%</sub>	<i>dT(n)</i>	0.1390	6.670	85.36	0.2655	9.892	54.44	0.1861	7.800	76.28
A3	md2, <i>day-c</i> only	<i>d</i>	0.1374	6.692	85.00	0.2828	9.729	51.83	0.1727	7.743	77.57
A4	+ v.-sup = b.line	<i>d</i>	0.1333	6.459	<b>85.88</b>	0.2419	10.922	58.17	0.1572	7.453	79.49
A5	+ noise, $\mathcal{L}$ clean	<i>d</i>	0.1428	6.609	84.43	0.2256	9.672	63.50	0.1592	7.619	78.95
A6	+ all <i>n</i> transl.	<i>dT(n)</i>	0.1624	7.042	80.50	0.2214	9.092	67.01	0.1752	8.272	76.41
A7	– pose transl.	<i>dT(n)</i>	0.1597	7.143	81.37	0.2184	8.754	66.90	0.1689	8.210	77.23
ADn	+ <i>day</i> loss only	<i>dT(n)</i>	0.1433	6.954	83.27	0.2230	9.001	68.63	0.1545	7.915	78.36
A9	– time norm.	<i>dT(n)</i>	0.1554	6.949	81.66	0.2121	<u>8.502</u>	67.43	0.1627	7.797	77.62
ADa	ADn + <i>r</i> transl.	<i>dT(nr)</i>	0.1523	6.853	83.11	0.2187	9.003	68.84	0.1601	7.832	78.97
A11	+ <i>day</i> distill.	<i>dT(nr)</i>	0.1387	6.621	84.11	0.1960	8.595	70.08	0.1444	7.355	80.20
DDn	ADn + distill.	<i>dT(n)</i>	<b>0.1302</b>	<b>6.373</b>	85.02	<u>0.1959</u>	<b>8.471</b>	<u>70.14</u>	0.1429	7.312	79.60
DDr	<i>r</i> distill.	<i>dT(r)</i>	<u>0.1323</u>	<u>6.437</u>	85.18	0.2502	11.847	57.02	<b>0.1364</b>	<b>7.100</b>	<b>81.37</b>
<b>DDa</b>	ADa + distill.	<i>dT(nr)</i>	0.1366	6.452	84.61	<u>0.1921</u>	<u>8.507</u>	<b>71.07</b>	<u>0.1414</u>	<u>7.228</u>	<u>80.98</u>
A15	– test time norm.	<i>dT(nr)</i>	0.1367	<u>6.449</u>	84.56	<b>0.1881</b>	8.524	<u>70.65</u>	<u>0.1412</u>	<u>7.234</u>	<u>80.99</u>

Table 4. Ablation study on our self-supervised method on the nuScenes [4] validation set. md2: Monodepth2 [12]. DDa is our self-supervised approach reported throughout this work for nuScenes: md4all-DD trained on  $dT(nr)$ . Instead, ADa is our self-supervised md4all-AD model. Notation reused from Table 1.

the scores), and also re-evaluated DeFeatNet [34] and ADIDS [23] on the same test split (again reducing the scores). Among works focusing on depth estimation in the dark, only RNW reported its results on the more challenging nuScenes dataset. However, since the authors reported their scores on a different, custom split, we retrained their model on the official split. For this reason, the results of RNW differ from those reported directly by Wang et al. in their paper [39]. Nevertheless, the difference is relatively small as RNW reported at *night*: absRel of 0.3150 (0.3333 from our experiment with RNW), sqRel of 3.793 (4.006), RMSE of 9.6408 (10.098), and  $\delta_1$  of 50.81 (43.72). While this performance gap should be attributed to the different data splits used, it does not affect the comparisons since our models performed significantly better than what Wang et al. reported in their paper [39], both quantitatively and qualitatively. Furthermore, on nuScenes, we also report the values of R4Dyn [9] and PackNet-SfM [13], as provided to us by the authors of [9]. Additional related works tackling adverse conditions exist (Section 2.2.1). Still, their lack of open-source code or their use of unconventional and unclear experimental setups prevented us from directly comparing with their methods.

### A.3. Additional Quantitative Results

#### A.3.1 Ablation Study

In Table 4, we report an ablation study over the main components of our method.

We started from a Monodepth2 [12] trained on the en-

tire training set of nuScenes [4] (A0), meaning all available conditions. A0 performed poorly under adverse conditions due to the difficulty of establishing pixel correspondences across consecutive *night* and *rain* frames. A0 delivered scores and errors similar to those reported for Monodepth2 by prior works in their papers, such as RNW [39] and WSGD [38]. Furthermore, the outputs of A0 exhibit the same issues shown by RNW and WSGD in their qualitative comparisons (e.g., holes in the ground), which are not present and much improved when training Monodepth2 only on *day-clear*, as reported throughout this work (A3). In particular, with A0-A3, we show how the standard Monodepth2 performs substantially better than the complex RNW overall and significantly better than WSGD in the daytime (Table 5).

A1 is a Monodepth2 model trained on *day-clear* and *night* (i.e., everything excluding *day-rain*). A1 performed similarly to A0 at *night*, but significantly better with *rain*. Additionally, it can be seen how training on *day-rain* samples negatively affected the *day-clear* performance (A0) while excluding such rainy samples improved in standard conditions (A1). Then, A2 is another Monodepth2 model, trained on *day-clear* and the translated *night* samples we generated with the GAN. A2 was fed a mix of *day-clear* and generated *night* ones with  $x = 15\%$ , to resemble the day-night distribution of the training set (used by A1). The comparison of A1 with A2 shows mainly two aspects about the translated images (Figure 17): they are not entirely realistic, and, unlike the real ones, they do not prevent establishing the pixel correspondences. If the generated samples



Method	tr.data	day – RobotCar				night – RobotCar			
		absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
Monodepth2 [12]	$d$	0.1196	0.670	<b>3.164</b>	86.38	0.3029	1.724	5.038	45.88
WSGD [38]	$a: dn$	0.1760	1.603	6.036	75.00	0.1740	1.637	6.302	75.40
[ours] baseline	$d$	0.1209	0.723	3.335	86.61	0.3909	3.547	8.227	22.51
[ours] md4all-AD	$dT(n)$	<b>0.1113</b>	0.707	3.248	<b>88.02</b>	0.1223	0.851	3.723	<b>85.77</b>
[ours] <b>md4all-DD</b>	$dT(n)$	0.1128	0.648	3.206	87.13	<b>0.1219</b>	<b>0.784</b>	<b>3.604</b>	84.86
[ours] md4all-DD w/o test time norm.	$dT(n)$	0.1129	<b>0.640</b>	3.190	87.02	0.1256	0.824	3.703	83.87
[ours] md4all-AD w/ LiDAR scaling	$dT(n)$	0.1192	0.747	3.184	86.81	0.1275	0.834	3.641	86.15
[ours] md4all-DD w/ LiDAR scaling	$dT(n)$	<b>0.1133</b>	<b>0.642</b>	<b>3.052</b>	<b>87.45</b>	<b>0.1230</b>	<b>0.739</b>	<b>3.439</b>	<b>86.41</b>

Table 5. Evaluation of self-supervised works on the RobotCar [24] test set up to 50 meters. Different configurations of our method are compared with Monodepth2 and WSGD. LiDAR scaling indicates the use of LiDAR data at test-time to scale the predictions (as done by Monodepth2 and WSGD), which is equivalent to \* in the supervision notation of Table 1 and 2. The method highlighted in bold is reported throughout this work for RobotCar as md4all-DD trained on  $dT(n)$ . This Table extends Table 2.

were completely realistic (i.e., like the real *night* ones from nuScenes), there would have been a much smaller difference between A1 and A2. In particular, the generated images do not fully resemble the real *night* ones (Figure 17), especially for the noise, which is more consistent throughout the generated frames compared to the real ones, and the darkness levels, with images that are not as black as the real *night* ones of nuScenes. This lack of realism in the generated images is the reason for the performance improvement of A2 at *night* compared to A1.

Similarly, WSGD [38] showed the importance of denoising *night* images, with noise detrimental to the models. Since the translated images do not exhibit the same kind of noise and reflections as the real ones and are particularly unrealistic when translating a sunny sample (Figure 17), A2 was able to establish pixel correspondences across the translated samples to a certain extent. Additionally, as  $x$  randomizes the condition of each input independently, with A2, the translations also introduce a regularization effect as data augmentation. A3 was also a Monodepth2 model but trained only on the *day-clear* samples. As shown in Table 1, this improves significantly compared to training on all conditions (A0) due to the impossibility of establishing correspondences at *night* for A0.

The weak velocity supervision [13] (A4) improved significantly over A3, thanks to better pose estimates. Compared to A0-A3, which need ground truth median scaling at test time, A4 is scale-aware and does not use it. With A5, we added heavy noise (consistently throughout the triplets) but computed the losses on the clean samples (i.e., without noise). This made it worse for *day-clear* and *day-rain*, but improved for *night* compared to A4. The motivation for A5 develops from the intense noise present in the *night* samples of nuScenes (Figure 2), which may confuse the models. We did not apply this under supervised settings (i.e., our method on AdaBins). The improvement seen with

adding noise while computing the loss on the inputs without noise paved the way for the concept of our AD model. With A6, we added the translated images generated with the GAN from day to night ( $x = 50\%$ ) to the training data. This made it worse than A0 for *day-clear*, but similarly to A2, it improved for *night* due to the lack of realism of the generated samples, which allowed to establish pixel correspondences. For A6 (and A2), perfectly realistic generated samples would have been detrimental to learning.

With A7, we did not feed the translated images to the pose model but only to the depth one. This guarantees reasonable pose estimates, which improve the task at hand under all three conditions. Then, with ADn, we computed the losses only on the *day-clear*  $e_i$  samples, corresponding to the translated ones given as input. This significantly improved the model performance on *day-clear*, reaching a level similar to A3 (i.e., only a marginal degradation on the standard conditions). It should be noted that if the translated images perfectly mimicked the real *night* ones, A2, A6, and A7 would have performed relatively poorly, i.e., similarly to A0 and A1 at *night*. In the case of perfect day-to-night translations, always computing the loss only on the *day-clear*  $e_i$  samples (as in ADn, instead of calculating it on the translated ones, as in A2, A6, and A7) would have had a significantly positive impact at *night*.

With A9, we removed the time-dependent normalization from ADn, which was used from A6 to ADn. This shows that this technique benefits both *day-clear* and *day-rain*, as it helps construct a unified representation for all conditions. With ADa, we incorporated day-to-rain translations to ADn alongside the day-to-night images ( $x = 66\%$ , i.e., one-third for each condition). This delivered a similar performance to ADn (e.g., improved for *night*, and improved the RMSE, with a worse absRel for *day-clear*). As for Table 1, the LiDAR ground truth is not fully reliable for *day-rain* and also significantly sparser than for *day-clear* (Figure 2). With

Method	<i>day-clear</i>		<i>fog</i>		<i>snow</i>	
	absRel	$\delta_1$	absRel	$\delta_1$	absRel	$\delta_1$
md2 [12]	0.1642	82.35	0.1698	81.97	0.1798	76.68
[ours]	<b>0.1520</b>	<b>83.54</b>	<b>0.1524</b>	<b>83.36</b>	<b>0.1788</b>	<b>77.93</b>

Table 6. These are only preliminary results (details in Section A.3.2). Evaluations with *snow* and *fog* on the DENSE dataset [2]. AbsRel and  $\delta_1$  are reported for each condition. Monodepth2 [12] (md2) trained on *day-clear* is compared with our md4all-DD trained on *day-clear* plus translated images to *fog* and *snow* ( $x = 66\%$ ).

A11, we added the day distillation loss (Equation 1) on the translated inputs while keeping the standard losses for the *day-clear* inputs. This combination improved across the board. Then, with DD, we simplified the training process by using only the day distillation loss for all inputs (including *day-clear*). Thus, DDn does this for *day-clear* and *night*, DDr does it for *day-clear* and *day-rain*. Our day distillation provides a dense and reliable signal (from A4 inferring only on *day-clear* samples), improving the errors and metrics across the board.

Finally, with A15, we show the impact of avoiding the time-dependent normalization at test time. Compared to DDa, A15 does not apply such time-dependent normalization at inference time but only at training time. A15 obtains comparable results throughout the various settings. Instead, as shown with A9, the time-dependent normalization is helpful at training time. After training, our model has learned robust features agnostic to the condition, allowing it to perform similarly regardless of the image normalization applied at test time. This demonstrates how our method does not need any condition-specific setups at inference time to deliver robust predictions, clearly separating our md4all from previous works requiring custom branches for each condition. In the supervised settings (i.e., our method applied on AdaBins), we did not perform any time-dependent normalization since the strong LiDAR supervision is enough to learn depth estimation at *night*.

In Table 5, we report various configurations of our method on the RobotCar [24] dataset. As for nuScenes (A15 in Table 4), we show that not applying the time-dependent normalization at test-time (i.e., executing the same operations with the same setup across the different conditions) does not negatively affect the predictions, achieving comparable results. Furthermore, we show how the results change when applying the median scaling via LiDAR data at test time. This technique is used by Monodepth2 [12], WSGD [38], and most other methods compared in this work. Our model does not need such scaling via ground truth data, thanks to its scale awareness learned via the weak velocity supervision introduced by PackNet-SfM [13].

Method	<i>avg/all</i>		<i>day</i>		<i>night</i>	
	absRel	$\delta_1$	absRel	$\delta_1$	absRel	$\delta_1$
md2 [12]	0.2122	65.92	0.1196	86.38	0.3029	45.88
70d - 30n	0.1189	<b>86.39</b>	0.1138	<b>87.80</b>	0.1239	<b>85.01</b>
<b>50d - 50n</b>	<b>0.1174</b>	85.99	<b>0.1128</b>	87.13	<b>0.1219</b>	84.86
30d - 70n	0.1221	85.86	0.1168	87.16	0.1273	84.59

Table 7. Impact of different training data distributions between *day* ( $d$ ) and *night* ( $n$ ) samples by varying the parameter  $x$ . AbsRel and  $\delta_1$  are reported on the test set of the RobotCar dataset [24]. Different distributions are shown for md4all-DD,  $a$ . In the rest of this work, the balanced 50d-50n configuration (i.e.,  $x = 50\%$ ) was used.

Nevertheless, accurate scaling can further improve the results, especially at *night*. Compared to nuScenes, the RobotCar dataset provides less precise odometry information, causing difficulties for the baseline and our models to learn the correct scaling. This can be seen by the improved scores at *night* when applying the median scaling via LiDAR data. With reliable scaling via the ground truth data, md4all-DD outperforms md4all-AD.

**AD and DD** While the benefit of our DD configuration over AD is evident for nuScenes, the gap is not as significant for RobotCar, with the two delivering comparable results (Table 5). This difference can be attributed to various reasons. First of all, nuScenes is more challenging, as demonstrated by the lower scores obtained by the models across all conditions, especially at *night*. Thus, the improvements of DD over AD might be reduced for RobotCar since AD already achieves solid results. The higher amount of images available on RobotCar to learn the translation task led to more realistic translations than nuScenes (Section A.4.5). Then, the less precise odometry information of RobotCar impacted the performance of the baseline through weak velocity supervision. Therefore, the baseline possibly learned wrong poses. This is not the case on nuScenes (Table 4), where the baseline (A4) improved significantly over Monodepth2 (A3). This did not happen for RobotCar. Since our md4all-DD learns to mimic the baseline via knowledge distillation, our model is directly affected by the weaker baseline in RobotCar, delivering similar results to AD. Instead, in nuScenes the gap between AD and DD is substantial throughout the conditions.

As shown with Monodepth2 [12] and AdaBins [1], our method is widely flexible and applicable to different architectures and types of supervision. While being out of the scope of this work, our approach can be seamlessly applied to other self-supervised or supervised frameworks, such as PackNet-SfM [13], since we do not alter the model architecture, but only its training scheme. In particular, to apply the proposed md4all to an existing depth estimation method, no structural changes are needed, as it is sufficient to feed

Method	tr.data	<i>day-clear</i> – nuScenes			<i>night</i> – nuScenes			<i>day-rain</i> – nuScenes		
		absRel	RMSE	$\delta_1$	absRel	RMSE	$\delta_1$	absRel	RMSE	$\delta_1$
AdaBins [1]	<i>a: dnr</i>	0.1384	5.582	81.31	0.2296	7.344	63.95	0.1726	6.267	76.01
AdaBins [1]	<i>d</i>	0.1138	4.805	87.98	0.3336	14.002	45.77	0.1540	6.119	81.20
[ours] md4all-AD, <i>rain</i>	<i>dT(r)</i>	<b>0.1052</b>	<b>4.621</b>	<b>89.58</b>	0.2644	10.749	55.51	<b>0.1380</b>	6.030	<b>83.32</b>
[ours] <b>md4all-AD, all</b>	<i>dnT(r)</i>	<u>0.1206</u>	<u>4.806</u>	<u>88.03</u>	<b>0.1821</b>	<b>6.372</b>	<b>75.33</b>	<u>0.1562</u>	<b>5.903</b>	<u>82.82</u>

Table 8. Additional evaluation of LiDAR-supervised methods on the nuScenes [4] validation set. This table adds the second and third lines compared to Table 1.

Method	<i>avg/all</i> – nuScenes				<i>night-rain</i> – nuScenes				<i>d-clear</i>	<i>night</i>	<i>d-rain</i>
	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$	sqRel	sqRel	sqRel
md2 [12], <i>d</i>	0.1576	2.002	7.164	80.49	0.3148	3.001	9.523	46.72	1.820	2.879	2.296
R4Dyn [9], <i>d</i> (radar)	<b>0.1365</b>	1.830	6.957	<b>84.01</b>	0.2431	2.945	10.055	56.95	<b>1.661</b>	2.889	1.938
RNW [39], <i>dn</i>	0.2931	3.557	9.304	55.13	0.3400	4.783	10.189	44.68	3.433	4.066	3.796
baseline, <i>d</i>	0.1480	2.032	7.065	82.08	0.2684	3.368	10.664	53.54	1.738	2.776	2.273
md4all-AD, <i>dT(nr)</i>	0.1602	2.245	7.226	81.02	0.2470	3.442	9.153	65.17	2.141	2.991	2.259
<b>md4all-DD, dT(nr)</b>	0.1429	<b>1.828</b>	<b>6.782</b>	82.67	<b>0.2143</b>	<b>2.628</b>	<b>8.376</b>	<b>68.03</b>	1.752	<b>2.386</b>	<b>1.829</b>
AdaBins [1], <i>a</i>	0.1604	1.103	5.868	78.72	0.2343	1.704	7.088	61.62	0.980	1.773	1.249
<b>md4all-AD, dnT(r)</b>	<b>0.1328</b>	<b>0.952</b>	<b>5.139</b>	<b>85.92</b>	<b>0.1967</b>	<b>1.632</b>	<b>6.423</b>	<b>71.67</b>	<b>0.821</b>	<b>1.525</b>	<b>1.199</b>

Table 9. Evaluation of fully-supervised (based on AdaBins [1]) and self-supervised methods (based on md2: Monodepth2 [12]) on the nuScenes [4] validation set. The models are the same as in Table 1. This table complements Table 1 with an evaluation on *all* conditions combined, as well as the most challenging *night-rain*.

to the model the translated images  $x\%$  of the time during training.

### A.3.2 DENSE Dataset: Snow and Fog

Disclaimer: First, please consider that these are only preliminary experiments and that we have not yet explored these conditions and models to the same extent as *night* and *rain* in the rest of this work. Nevertheless, we report them here as they provide interesting insights.

In Table 6, we show a first attempt to tackle the problem of monocular depth estimation in the presence of snow or fog with the DENSE dataset [2]. While our md4all performed better than the standard Monodepth2 [12] across the board, the improvement is relatively small compared to the other datasets and conditions explored (e.g., Tables 1 and 2). There are multiple reasons for this, explained below.

An impactful aspect to be considered is related to the available data. The condition boundaries are somewhat blurry. Overcast *day-clear* samples can be similar to light *fog* or light *snow*. This is problematic for the GAN used for image translation, which cannot distinguish the distributions and adequately learn the translation task.

Furthermore, the term snow is generic and includes various scenarios, such as light snow, heavy snow, blizzard, partly covered ground, fully covered ground, piles of snow, or wet ground with light snow falling. These settings differ substantially, but all belong to the same *snow* condition.

This high variability is problematic for the translation task. While this issue can occur similarly with *night* and *rain* too, it is not as severe, and the diversity is more limited.

Another significant issue is the amount of usable image data for these conditions, which is insufficient to properly learn the translation task with ForkGAN [46]. As we did for *night* for nuScenes (Section 4.1), also for DENSE, we had to supplement with extra snow images taken from another dataset: Boreas [3]. We trained the snow ForkGAN with 17591 *day-clear* and 8443 *snow* samples from DENSE, plus 25036 *day-clear* and 26437 *snow* samples from Boreas for the pre-training. While supplementing with data from Boreas helped, the number of images from DENSE was relatively low compared to nuScenes and RobotCar, preventing effective translations.

Training data for the depth models was 7947 *day-clear* keyframes for DENSE. These keyframes were relatively few (15129 were used for nuScenes and 17790 for RobotCar), and they were extracted from short sequences, so they did not exhibit high variability. This reduced the depth estimation performance of the models. The validation set was also small with only 289 for *day-clear*, 1281 for *snow*, and 543 for *fog*. DENSE contains more images, but those were not usable due to various reasons, e.g., they were captured by different sensors.

Furthermore, as with *rain* (Figure 2), the LiDAR is not reliable in the presence of *snow* or *fog*, as it often captures snowflakes, fog particles, or is even obstructed by the snow

Method	40m – <i>day-clear</i> – nuScenes				40m – <i>night</i> – nuScenes				40m – <i>day-rain</i> – nuScenes			
	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
md2	<b>0.1095</b>	<b>0.796</b>	<b>3.535</b>	88.89	0.2401	1.640	5.842	60.40	0.1405	1.083	4.259	82.64
b.line	0.1131	0.932	3.624	<b>89.46</b>	0.2118	1.816	6.476	63.47	0.1333	1.200	4.397	83.66
AD	0.1306	1.074	3.866	86.95	0.1907	1.670	5.414	73.24	0.1329	1.083	4.332	83.54
<b>DD</b>	0.1173	0.877	3.592	88.22	<b>0.1672</b>	<b>1.322</b>	<b>5.025</b>	<b>75.50</b>	<b>0.1190</b>	<b>0.927</b>	<b>4.036</b>	<b>85.37</b>

Table 10. Evaluation up to 40 meters of self-supervised approaches on the validation set of nuScenes [4]. md2: Monodepth2 [12] trained on *d*. b.line: baseline trained on *d*. AD: md4all-AD trained on  $dT(nr)$ . DD: md4all-DD trained on  $dT(nr)$ . The models are the same as in Table 1.

Method	40m – <i>avg/all</i> – nuScenes				40m – <i>night-rain</i> – nuScenes			
	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
md2	0.1276	<b>0.926</b>	3.882	85.04	0.2768	2.020	6.604	53.26
b.line	0.1262	1.063	4.034	85.93	0.2466	2.301	7.440	57.13
AD	0.1369	1.135	4.096	85.03	0.2184	2.064	6.066	68.96
<b>DD</b>	<b>0.1225</b>	0.930	<b>3.807</b>	<b>86.49</b>	<b>0.1925</b>	<b>1.653</b>	<b>5.661</b>	<b>71.38</b>

Table 11. Evaluation up to 40 meters of self-supervised approaches on the validation set of nuScenes [4]. md2: Monodepth2 [12] trained on *d*. b.line: baseline trained on *d*. AD: md4all-AD trained on  $dT(nr)$ . DD: md4all-DD trained on  $dT(nr)$ . The models are the same as in Table 1. This table complements Table 10.

accumulated on the sensor itself. We mitigated this problem by filtering the erroneous LiDAR points via clustering, but we could not eliminate all problematic measurements. As seen for *rain* on nuScenes, in adverse conditions the LiDAR sensor is unable to collect measurements at further distances (e.g., Figure 5 *rain* vs. *night* ground truth depth). Therefore, we could only evaluate a limited set of points at a closer distance. We used a single LiDAR scan as ground truth. Additionally, among the snow data, many samples were recorded in remote areas with relatively flat surroundings. Considering the limited distance and the flat surroundings, a model overfitting on flat ground may seem erroneously adequate by obtaining good quantitative results.

Additionally, for the weak velocity supervision of our baseline, we exploited the information from the CAN bus, as provided by the authors of DENSE. We used the vehicle speed and the frame rate to compute the camera translation between the frames. However, since the vehicle speed is provided as single value for each short sequence, the camera poses could only be coarsely approximated. This likely affected the performance of the baseline, hence that of our md4all-DD too. Furthermore, we used the CAN data to discard static inputs (i.e., stationary ego vehicle) and those where the ego vehicle is turning. We filtered the latter when the steering wheel angle exceeded  $20^\circ$ . This filtering led to the numbers indicated above.

All these points should be considered when evaluating these preliminary results on DENSE.

First of all, regarding Table 6, it can be seen how the *day-clear* results are not as good as those seen for nuScenes (Table 1) or RobotCar (Table 2). This could be attributed to

DENSE containing more challenging data. More likely, it is due to the inability of the models to properly generalize on DENSE due to the relatively low diversity in the training data and the limited amount of training samples. Therefore, our md4all-DD learned from a weak baseline which could not correctly estimate depth in standard conditions. Nevertheless, our md4all-DD outperformed Monodepth2 in standard settings, thanks to the regularization effect of our translations.

In the table, we report light-fog for *fog* and full-coverage or currently snowing for *snow*. We opted for light-fog since dense-fog exhibited too few LiDAR points for the evaluation, all at relatively close distances (easier). Instead, light-fog allowed for a more thorough assessment at further distances. For reference, all results were better with dense-fog than light-fog. For *snow*, we selected those with full-coverage or weather metadata snow. This is because, among the annotated conditions, they had the most precise boundaries with other conditions.

Moreover, both models perform similarly with *fog* as in ideal settings (i.e., *day-clear*). While this hints that fog is not as challenging as rain or night (Tables 1 and 2), the values are also affected by the limited distance of the ground truth used for the evaluation. Therefore, the performance may degrade significantly at further distances due to the fog preventing seeing the details, but that cannot be evaluated. Nevertheless, already at the available ground truth distances, our model outperformed Monodepth2, on which ours is based.

Despite the limited distance of the ground truth, *snow* appears more challenging than *fog*, causing a significant



Method	60m – <i>day-clear</i> – nuScenes				60m – <i>night</i> – nuScenes				60m – <i>day-rain</i> – nuScenes			
	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
md2	0.1283	<b>1.387</b>	5.447	86.15	0.2739	2.469	8.444	53.40	0.1623	1.779	6.312	79.23
b.line	<b>0.1279</b>	1.522	5.422	<b>86.91</b>	0.2348	2.779	9.502	59.28	0.1506	1.833	6.267	80.73
AD	0.1461	1.745	5.744	84.23	0.2113	2.526	7.789	69.92	0.1519	1.774	6.421	80.28
<b>DD</b>	0.1310	1.419	<b>5.364</b>	85.65	<b>0.1859</b>	<b>2.029</b>	<b>7.377</b>	<b>72.10</b>	<b>0.1347</b>	<b>1.463</b>	<b>5.938</b>	<b>82.24</b>

Table 12. Evaluation up to 60 meters of self-supervised approaches on the validation set of nuScenes [4]. md2: Monodepth2 [12] trained on *d*. b.line: baseline trained on *d*. AD: md4all-AD trained on  $dT(nr)$ . DD: md4all-DD trained on  $dT(nr)$ . The models are the same as in Table 1.

Method	60m – <i>avg/all</i> – nuScenes				60m – <i>night-rain</i> – nuScenes			
	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
md2	0.1483	1.558	5.886	81.76	0.3086	2.782	8.790	47.76
b.line	0.1423	1.698	5.966	83.15	0.2648	3.130	9.858	54.09
AD	0.1536	1.827	6.057	82.16	0.2402	2.983	8.233	65.79
<b>DD</b>	<b>0.1371</b>	<b>1.487</b>	<b>5.658</b>	<b>83.74</b>	<b>0.2101</b>	<b>2.368</b>	<b>7.661</b>	<b>68.58</b>

Table 13. Evaluation up to 60 meters of self-supervised approaches on the validation set of nuScenes [4]. md2: Monodepth2 [12] trained on *d*. b.line: baseline trained on *d*. AD: md4all-AD trained on  $dT(nr)$ . DD: md4all-DD trained on  $dT(nr)$ . The models are the same as in Table 1. This table complements Table 12.

drop in performance compared to the ideal settings (i.e., *day-clear*). With snow, the limited data available to learn proper translations substantially impacted our method’s performance, which obtained only slightly better scores than Monodepth2.

Stronger condition boundaries (e.g., more precise annotations) and more training data would significantly improve the translations and our method’s outcomes. Furthermore, depth ground truth reaching further distances without any artifacts would allow us to assess the actual performance of the models. While these factors would contribute to a more considerable gap between the proposed md4all and Monodepth2, the issues with the translations also highlight the limitations of our approach: the difficulty in collecting adverse data that would lead to solid results (e.g., Table 2 with RobotCar [24]).

Due to the substantial limitations encountered with this data, the DENSE dataset is unsuitable for depth estimation. However, we used it to provide these preliminary results with snow and fog. New real data with artifact-free long-distance ground truth is needed to properly explore monocular depth estimation in these conditions.

### A.3.3 Different Distributions of Conditions

In Table 7, we explore the effect of different data distributions among the conditions during training. We vary this via the parameter  $x$ . In the rest of this work,  $x$  was selected to equally distribute the inputs among the conditions. So for RobotCar 50% for half for *day* and half for *night* (i.e., 50*d* - 50*n* in the table); for nuScenes 66% corresponding to one third for each of *day-clear*, *night*, and *rain*; one third each

also for the DENSE dataset.

While intuitively increasing the amount of *day* images could improve the performance on *day*, this is not the case by randomizing via  $x$  at each training sample independently. This is because with enough epochs, our model sees all images in all conditions, so the training data remains unchanged, causing only minor differences as the model might be fed more or fewer translations (Table 7). For *day*, beyond the observed regularization effect (e.g., Table 2), there is little room for gains as long as the baseline model  $B$  to distill from remains the same. Instead, if  $x$  affected which portion of the training data is translated, it would have a more significant impact than shown in Table 7. In that case, seeing too many or too few translated images may impair the performance as the model does not experience enough of the ideal settings or not enough adverse conditions to tackle them properly.

### A.3.4 Supervised Configuration Comparisons

Table 8 reports a comparison of different supervised configurations of AdaBins [1] and our md4all-AD applied on AdaBins. Specifically, AdaBins trained only on *day-clear* and *day-rain* compared to the AdaBins trained in all conditions (i.e., *a*). Analogously, our model trained on *day-clear* and translated *day-rain* samples performed better than ours trained on all but substantially worse at *night*. As seen in the self-supervised case, our model trained in all conditions outperformed the baseline across the board (i.e., AdaBins trained on all), thereby not introducing any trade-off while improving in adverse conditions over the model it is based on.

Method	test – 40m – nuScenes				test – 60m – nuScenes				test – 80m – nuScenes			
	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
md2	0.1162	0.811	3.701	87.59	0.1376	1.364	5.650	84.08	0.1465	1.755	6.941	82.69
RNW	0.2500	2.237	6.114	62.63	0.2781	3.420	9.222	57.50	0.2900	4.169	11.289	55.56
b.line	0.1126	0.840	3.827	87.17	0.1275	1.364	5.747	84.33	0.1332	1.679	6.938	83.22
AD	0.1214	0.881	3.851	86.99	0.1353	1.387	5.731	84.14	0.1409	1.691	6.915	83.00
<b>DD</b>	<b>0.1090</b>	<b>0.757</b>	<b>3.606</b>	<b>88.29</b>	<b>0.1221</b>	<b>1.204</b>	<b>5.418</b>	<b>85.57</b>	<b>0.1277</b>	<b>1.503</b>	<b>6.607</b>	<b>84.46</b>
AdaBins	0.1434	<b>0.617</b>	<b>3.233</b>	83.09	0.1494	0.852	4.689	80.67	0.1532	1.055	5.849	79.62
<b>AD sup.</b>	<b>0.1182</b>	0.641	3.279	<b>89.40</b>	<b>0.1221</b>	<b>0.785</b>	<b>4.293</b>	<b>87.98</b>	<b>0.1240</b>	<b>0.887</b>	<b>5.021</b>	<b>87.33</b>

Table 14. Evaluation up to 40, up to 60, and up to 80 meters of self-supervised approaches on the test set of nuScenes [4]. All conditions are evaluated here (*avg/all*). AD sup.: our md4all-AD trained on  $dnT(r)$  applied on AdaBins [1]. md2: Monodepth2 [12] trained on  $d$ . b.line: baseline trained on  $d$ . AD: md4all-AD trained on  $dT(nr)$ . DD: md4all-DD trained on  $dT(nr)$ . Time-dependent normalization was not applied to obtain these results as nuScenes provides no condition annotations for the test set. The models are the same as in Table 1.

### A.3.5 nuScenes *Night-rain* and Average

In Table 9, we report results on more conditions of nuScenes [4], such as the most difficult *night-rain* and an average over *all*, alongside the sqRel errors not fitting in Table 1 (due to the limited space available). *All* is not computed as an average on the various conditions but rather as an average of the performance on each sample (i.e., *night* counts marginally, accounting for only 10% of the images). Our model outperforms the baseline AdaBins across the board for the supervised case. Similarly, in the self-supervised setting, our md4all improved significantly over the baseline and Monodepth2 [12], second only in ideal conditions (*day-clear*) to the radar-based R4Dyn [9].

### A.3.6 nuScenes Test Set

Table 14 reports errors and metrics on the official test set of nuScenes [4]. Since no metadata about the weather and illumination conditions is available for the test set, the values are to be considered for *all* conditions combined. As for the validation set, results were computed with the available ground truth data (i.e., LiDAR). The test set of nuScenes is more accessible than its validation set, with the models achieving better performance in the former across all three distance ranges. For these reasons, we focused on the validation set, and we report the test set results here for completeness. Nevertheless, our supervised and self-supervised models performed better than the models they are based on (i.e., AdaBins and Monodepth2, respectively) across the various conditions and depths.

### A.3.7 Evaluation over Different Distances

For completeness, we also include the results for self-supervised models computed up to different distance ranges, namely up to 40, 60, and 80 meters. This complements the values reported in the main paper (up to 80 meters for nuScenes, and up to 50 for RobotCar) and should ease

comparisons with future works. These results are reported across the following tables:

- Tables 10, 11, and 14 for nuScenes up to 40 meters.
- Tables 12, 13, and 14 for nuScenes up to 60 meters.
- Tables 1, 8, 9, and 14 for nuScenes up to 80 meters.
- Table 15 for RobotCar up to 40 meters.
- Table 2 for RobotCar up to 50 meters.
- Table 16 for RobotCar up to 60 meters.
- Table 17 for RobotCar up to 80 meters.

The scores improve at lower distances as the problem becomes more effortless. Changes are limited due to the sparsity of the LiDAR at further distances, especially for RobotCar. While evaluating using more LiDAR frames would have allowed us to evaluate at greater distances, we avoided it as it would have introduced artifacts in the ground truth for dynamic objects. Nevertheless, our models consistently improved across the various distances and conditions tested. As described in Section 4.1, the results we reported are to be considered up to 80 meters for nuScenes, and up to 50 meters for RobotCar, unless otherwise noted.

## A.4. Additional Qualitative Results

For all qualitative results reported in this work (also Figures 1, 5, and 6 in the main paper), the predictions of our self-supervised model were all performed by our md4all-DD trained on all conditions (i.e.,  $dT(nr)$  for nuScenes and  $dT(n)$  for RobotCar) and based on Monodepth2, while the supervised ones were all performed by our md4all-AD trained on  $dnT(r)$  and based on AdaBins. All qualitative images of the standard Monodepth2 were produced by the model trained only on *day-clear* (*day* for RobotCar). In contrast, the ones for the standard AdaBins were made by

Method	tr.data	40m – <i>day</i> – RobotCar				40m – <i>night</i> – RobotCar			
		absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
Monodepth2 [12]	<i>d</i>	0.1181	<b>0.614</b>	<b>3.034</b>	86.51	0.3022	1.702	4.984	45.97
[ours] baseline	<i>d</i>	0.1198	0.678	3.229	86.69	0.3908	3.541	8.206	22.52
[ours] md4all-AD	<i>dT(n)</i>	<b>0.1099</b>	0.650	3.130	<b>88.10</b>	<b>0.1203</b>	0.762	3.531	<b>85.87</b>
[ours] <b>md4all-DD</b>	<i>dT(n)</i>	0.1120	0.618	3.125	87.18	0.1206	<b>0.723</b>	<b>3.479</b>	84.92

Table 15. Evaluation of self-supervised works on the RobotCar [24] test set up to 40 meters. The models are the same as in Table 2.

Method	tr.data	60m – <i>day</i> – RobotCar				60m – <i>night</i> – RobotCar			
		absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
Monodepth2 [12]	<i>d</i>	0.1201	0.698	<b>3.215</b>	86.38	0.3029	1.728	5.045	45.88
[ours] baseline	<i>d</i>	0.1213	0.746	3.382	86.61	0.3909	3.548	8.228	22.51
[ours] md4all-AD	<i>dT(n)</i>	<b>0.1116</b>	0.731	3.291	<b>88.02</b>	0.1231	0.903	3.812	<b>85.76</b>
[ours] <b>md4all-DD</b>	<i>dT(n)</i>	0.1130	<b>0.661</b>	3.234	87.13	<b>0.1225</b>	<b>0.824</b>	<b>3.664</b>	84.86

Table 16. Evaluation of self-supervised works on the RobotCar [24] test set up to 60 meters. The models are the same as in Table 2.

Method	tr.data	80m – <i>day</i> – RobotCar				80m – <i>night</i> – RobotCar			
		absRel	sqRel	RMSE	$\delta_1$	absRel	sqRel	RMSE	$\delta_1$
Monodepth2 [12]	<i>d</i>	0.1203	0.718	3.245	86.38	0.3030	1.729	5.046	45.88
[ours] baseline	<i>d</i>	0.1214	0.759	3.404	86.61	0.3909	3.548	8.228	22.51
[ours] md4all-AD	<i>dT(n)</i>	<b>0.1118</b>	0.742	3.308	<b>88.02</b>	0.1236	0.952	3.880	<b>85.76</b>
[ours] <b>md4all-DD</b>	<i>dT(n)</i>	0.1131	<b>0.666</b>	<b>3.243</b>	87.13	<b>0.1229</b>	<b>0.865</b>	<b>3.713</b>	84.86

Table 17. Evaluation of self-supervised works on the RobotCar [24] test set up to 80 meters. The models are the same as in Table 2.

training it on all conditions, meaning that the best scoring models of each type produced them.

The following sections introduce new qualitative results on the various settings and conditions, such as nuScenes with self-supervised models (Section A.4.1), nuScenes with supervised methods (Section A.4.2), RobotCar with self-supervised approaches (Section A.4.3), failure cases (Section A.4.4), and translated images (Section A.4.5).

#### A.4.1 nuScenes – Self-Supervised

**Night – nuScenes** In Figure 8, we compare our md4all-DD to Monodepth2 [12] on particularly challenging *night* samples due to the extreme darkness levels, as well as the high amounts of reflections (also due to the wet ground, in *night-rain* conditions). Instead, in Figure 9, we compare the same models on brighter, i.e., easier samples. It can be seen that the standard Monodepth2 trained on *day-clear* had marked difficulties with darker scenes (Figure 8), while it delivered satisfactory results in brighter settings (Figure 9). This significant difference can be attributed to the fact that Monodepth2 was trained only on *day-clear* (the model trained on *all* conditions performed significantly worse, especially at *night*, as shown in Table 1) and that the details are easier to grasp in brighter *night* inputs compared to very dark ones. In brighter images, the road, vehicles, curb, grass, and

trees on the sides can be seen relatively well thanks to the light emitted by the streetlights, allowing the standard Monodepth2 to estimate reasonable depth maps. Conversely, with darker scenes, the depth cues are highly challenging to extract due to the noise and general blackness. Nevertheless, Monodepth2 tended to estimate the ground well also in darker settings. This can be due to the model being biased toward scenes with flat and regular roads in front of the ego vehicle and the road being the most visible part of the images. Instead, Monodepth2 had severe issues with the sky, which is not evaluated since the ground truth is unavailable. Nevertheless, the proposed md4all delivered good and sharp estimates in dark and bright environments.

**Rain – nuScenes** In Figure 10, we compare our md4all-DD to Monodepth2 [12] on *rainy* samples of nuScenes. While the standard Monodepth2 trained on *day-clear* delivered reasonable estimates for the most part, it was severely affected by the reflections on the wet ground (e.g., first row). Also, the blurriness due to water drops affected Monodepth2, triggering overly smooth estimates. The figure shows also Monodepth2 having issues estimating the depth of the sky (second and fifth row). Since none of these problems occurs without rain (i.e., *day-clear*, shown in Figure 11), they can be attributed to the challenging weather conditions. Instead, our md4all delivered reasonable depth estimates regardless of the adverse settings, because its su-



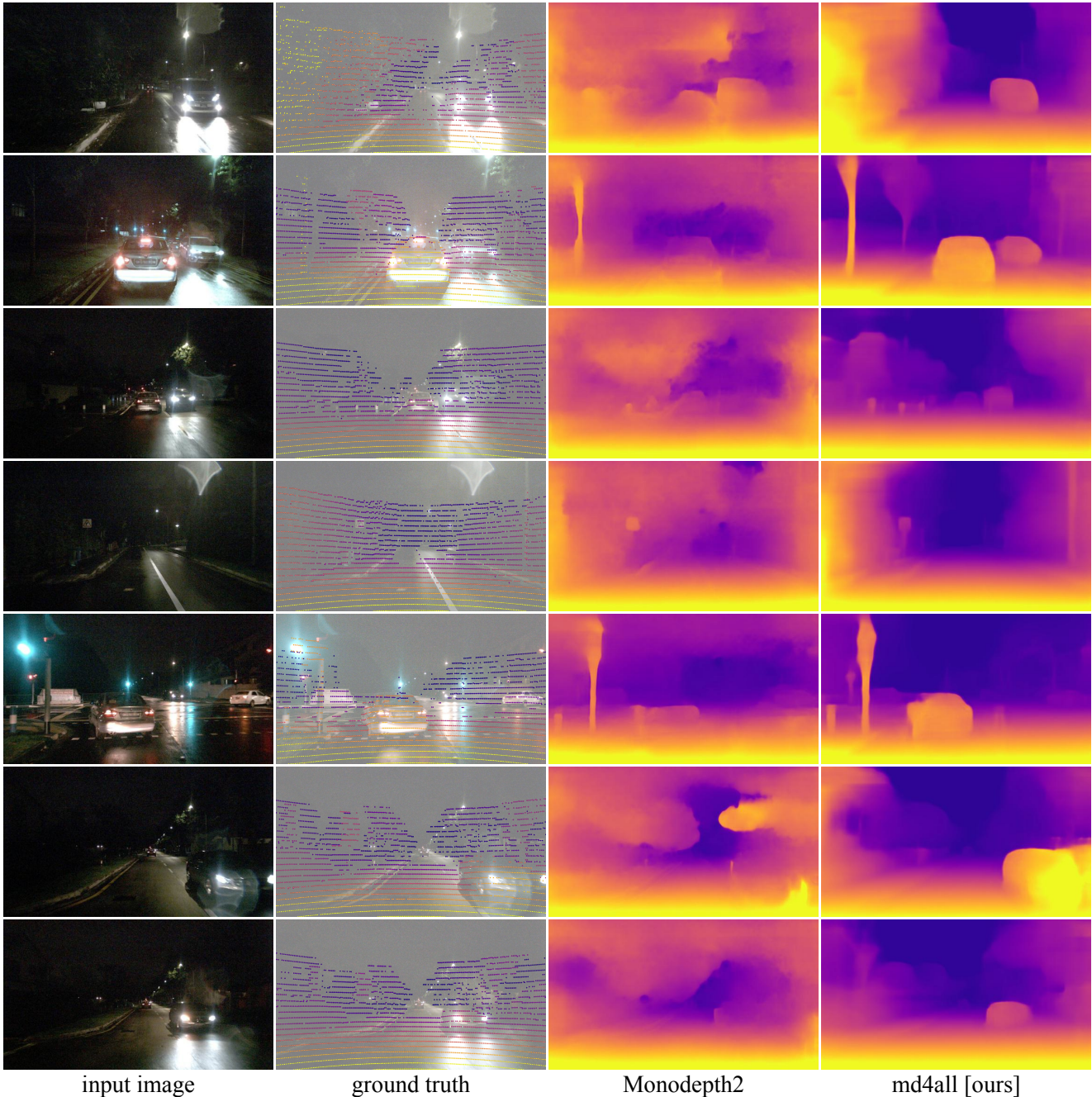


Figure 8. Comparison of self-supervised models on nuScenes [4] *night* samples. The standard Monodepth2 [12] is compared to our md4all-DD applied to Monodepth2. This set of samples is particularly challenging for the standard Monodepth2 due to the overall darkness and reflections.

pervision signal was always from ideal settings.

**Day-clear – nuScenes** In the lower half of Figure 11, we compare the performance at daytime. The *day-clear* scores of Monodepth2 ( $d$ ) and our md4all-DD ( $a$ ) are relatively close to one another (Table 1). Based on the same model (i.e., Monodepth2), the depth maps of the two are similar, with ours delivering smoother surfaces (e.g., vehi-

cles), with fewer artifacts around the objects, such as the black car on the left of the bottom input. Our extra smoothness is due to the day distillation scheme, propagating dense estimates from the baseline. Since our same model predicts significantly better depth maps in adverse conditions than Monodepth2, the figure confirms that our md4all does not degrade in ideal conditions. This proves the effectiveness of



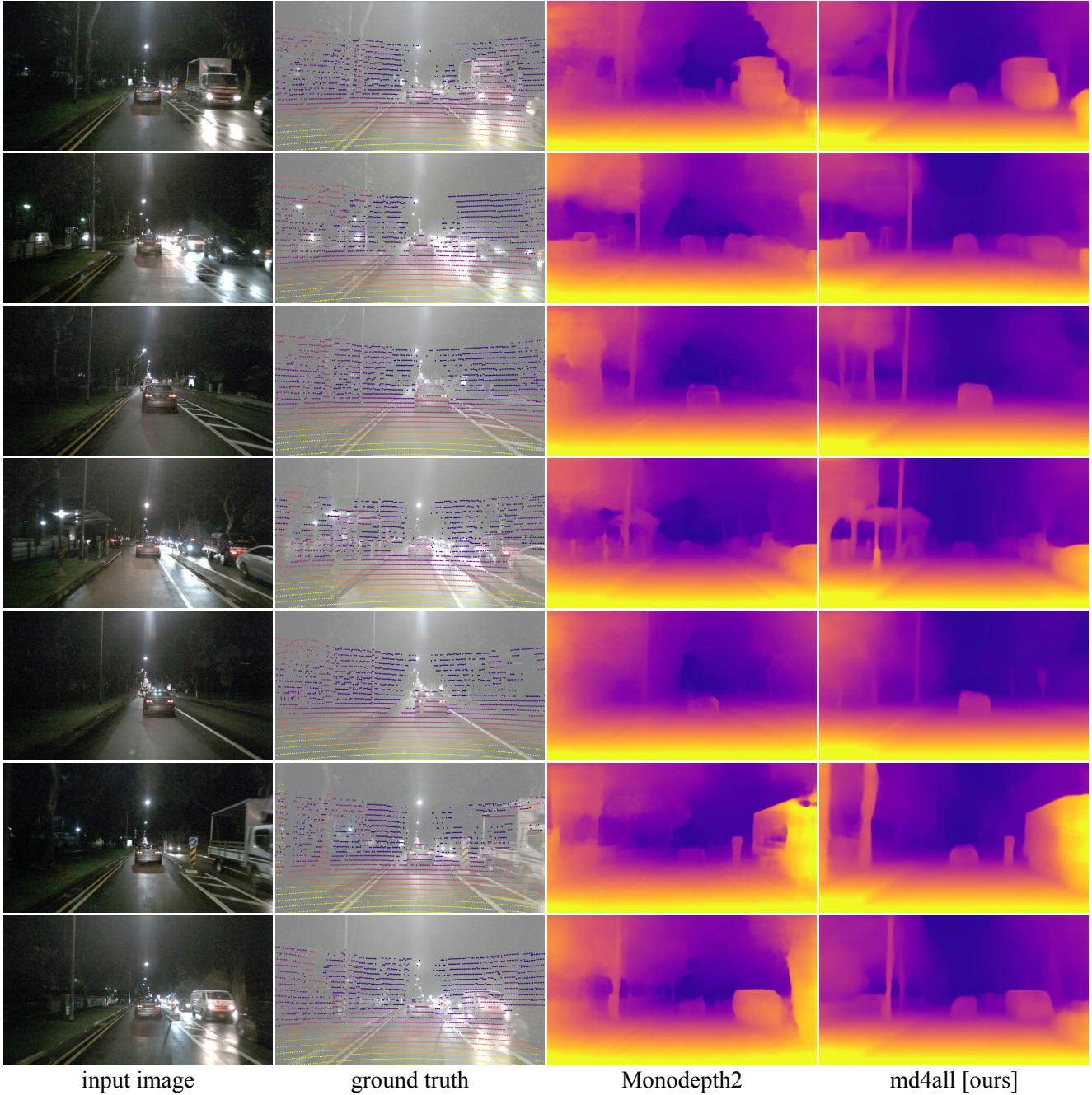


Figure 9. Comparison of self-supervised models on relatively bright nuScenes [4] *night* samples. The standard Monodepth2 [12] is compared to our md4all-DD applied to Monodepth2. This set of samples could be handled reasonably by the standard Monodepth2, thanks to the high brightness of the scenes.

our techniques, as they do not introduce any performance trade-off in standard settings while enabling significantly better outcomes in challenging conditions.

#### A.4.2 nuScenes – Supervised

**Night – nuScenes** In Figure 12, we compare our md4all-AD to AdaBins [1] on particularly challenging *night* samples due to the extreme darkness levels and the high amount of reflections (same samples as in Figure 8). Our md4all-AD delivered overall sharper and more accurate estimates, as can be seen for the thin structures (e.g., poles and tree

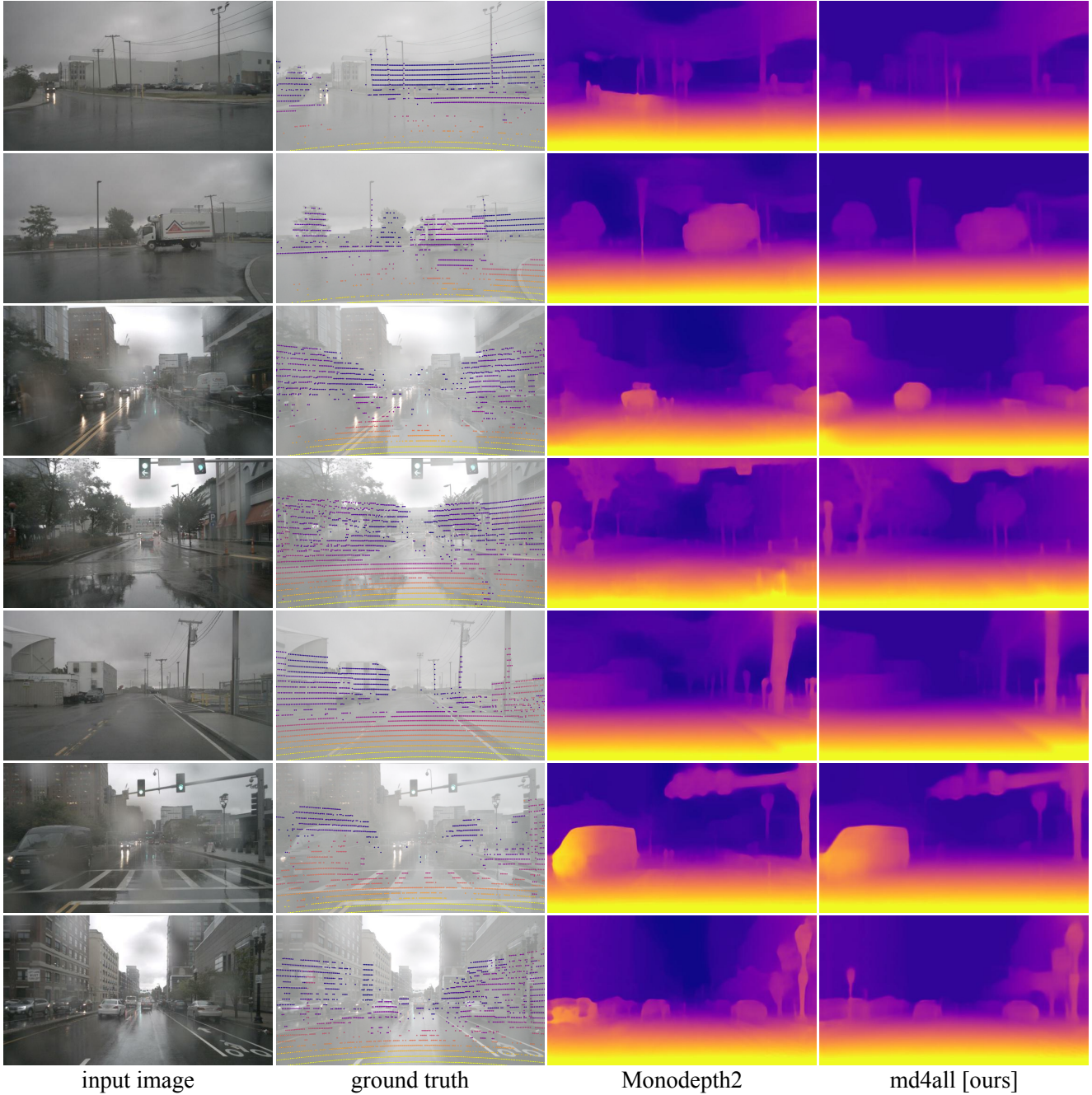


Figure 10. Comparison of self-supervised models on nuScenes [4] *rain* samples. The standard Monodepth2 [12] is compared to our md4all-DD applied to Monodepth2.

trunks), as well as the boundaries of the vehicles in the bottom two rows. Although all depth maps are displayed with the same colormap, the standard AdaBins never estimated depths that triggered the use of the bright yellow color (i.e., closest distance). This problem is less pronounced during daytime (Figure 11). This can be attributed to the standard AdaBins suffering from overfitting on the sparse ground truth data of nuScenes. Towards this end, we had to ap-

ply early stopping to prevent severe horizontal artifacts from appearing on the outputs following the LiDAR detections. Applying our md4all on AdaBins introduced challenging augmentations during training, reducing overfitting and allowing it to train longer. This led to a model that estimated depth throughout a more extensive range without the horizontal artifacts. The qualitative results align with the scores gap in Table 1. Moreover, some artifacts are noticeable for



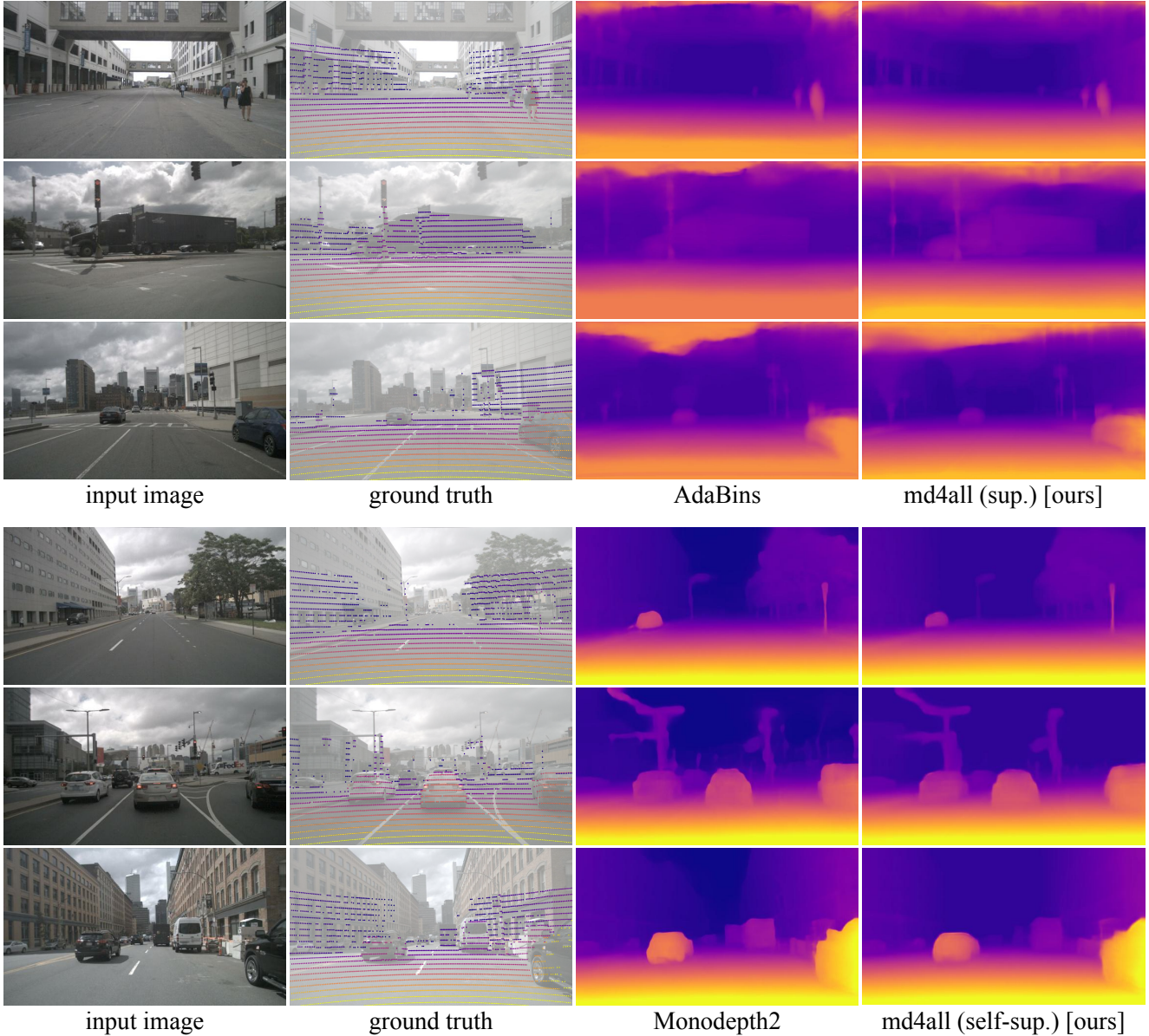


Figure 11. Comparison of models on nuScenes [4] *day-clear* samples. In the upper half, the standard AdaBins [1] is compared to our md4all-AD applied to AdaBins. In the lower half, the standard Monodepth2 [12] is compared to our md4all-DD applied to Monodepth2.

both models around the roof of the vehicles. These are due to the ground truth LiDAR data reporting further estimates in those areas (e.g., fifth row). This is caused by the relative position of the LiDAR sensor to the camera, with the former being at a higher location, thereby seeing beyond objects compared to the camera’s perspective.

**Rain – nuScenes** In Figure 13, we compare our md4all-AD to AdaBins [1] on *rain* samples of nuScenes. As the standard AdaBins learned from the erroneous measurements of the LiDAR ground truth (e.g., Figure 2), it systematically estimated artifacts on the ground, resulting in holes in its depth maps. The erroneous LiDAR measure-

ments can also be seen from the ground truth reported for each image. Such wrong measurements are caused systematically by each highly reflective object (e.g., traffic sign) reflected on the wet ground. Therefore, training on such a wrong signal causes the standard AdaBins to replicate the artifacts in its output (e.g., in front of the stop sign in the fourth row). Nevertheless, thanks to the reliable training signal from ideal conditions, our md4all-AD delivered good estimates without such artifacts.

**Day-clear – nuScenes** In the upper half of Figure 11, we compare the performance at daytime. The scores between AdaBins and our md4all-AD trained on all conditions favor

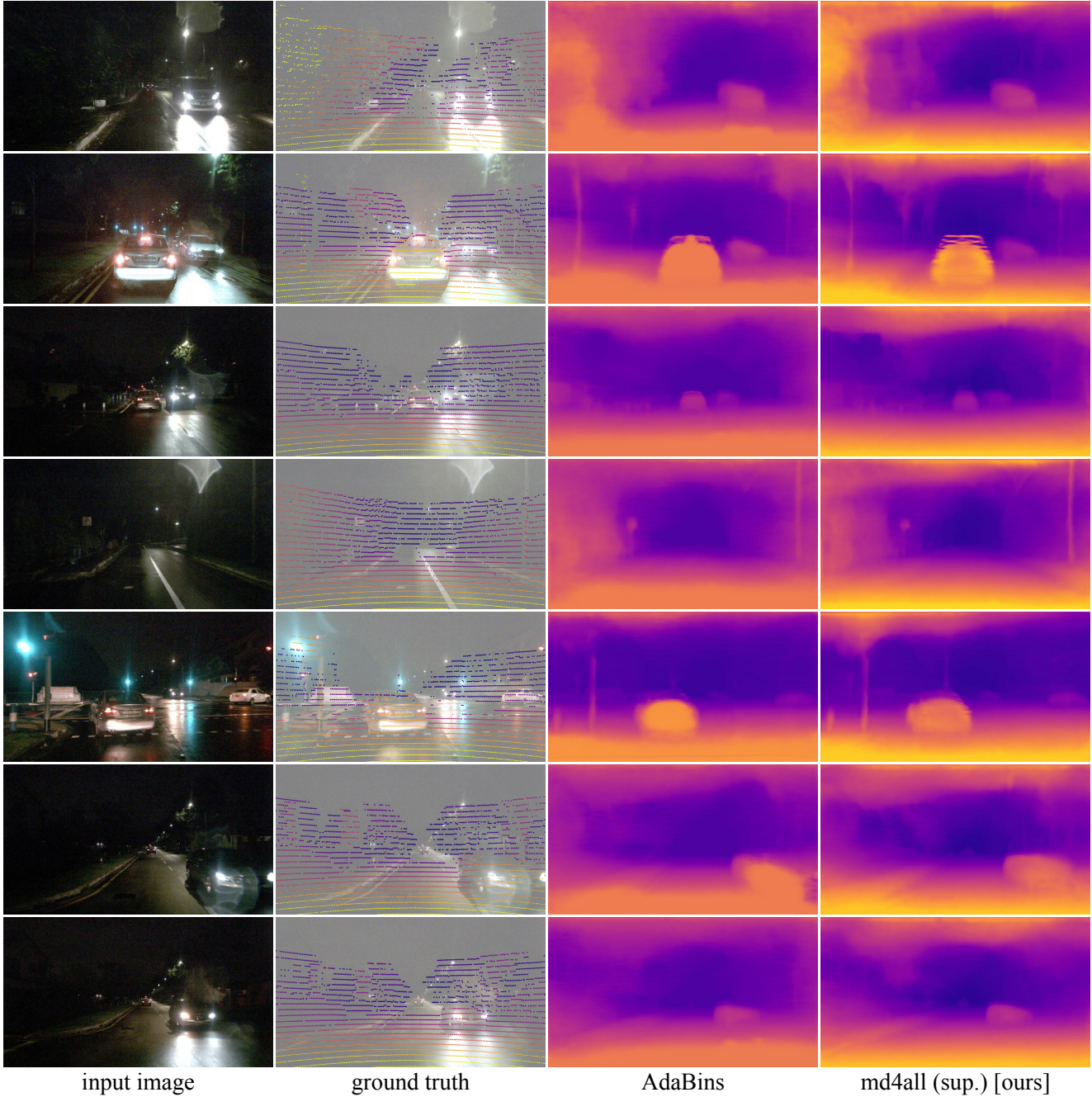


Figure 12. Comparison of supervised models on nuScenes [4] *night* samples. The standard AdaBins [1] is compared to our md4all-AD applied to AdaBins. This set of samples is particularly challenging due to the overall darkness and reflections.

ours (Table 1). This can be seen from the depth maps, with ours delivering smoother estimates for the road and sharper details throughout. Ours separated the truck in the second frame from the background. As discussed above, this can be attributed to the strong tendency of the standard AdaBins to overfit. Instead, by feeding a mix of original and translated images, our method acts as data augmentation, mitigating the problem. Nevertheless, the limitations of such

a sparse ground truth signal are evident, with smooth edges due to its sparsity and see-through effects caused by the displacement between the LiDAR and the camera. For these reasons, self-supervised outputs look better overall, as the models are also less prone to overfitting.



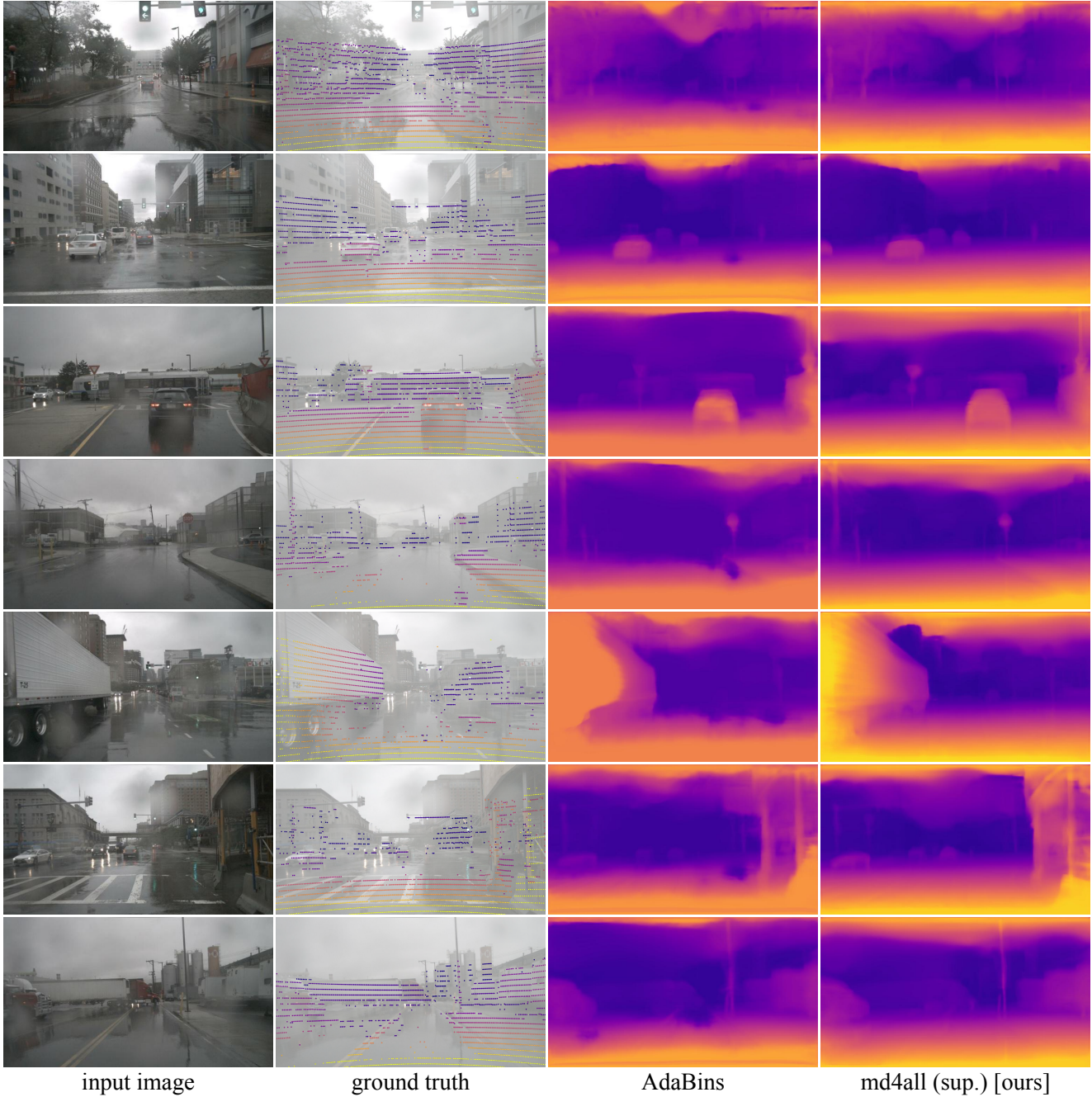


Figure 13. Comparison of supervised models on nuScenes [4] *rain* samples. The standard AdaBins [1] is compared to our md4all-AD applied to AdaBins.

### A.4.3 RobotCar – Self-Supervised

**Comparison with other *night* methods** In Figure 14, we compare our md4all-DD with various other methods targeting depth estimation in *night* conditions, namely DeFeatNet [34], ADIDS [23], and WSGD [38], as well as the standard Monodepth2 [12] designed for ideal conditions. The samples shown are exactly the ones displayed by WSGD [38] in their paper, from which we took the out-

puts of [34, 23, 38]. However, unlike theirs [38], for ours and Monodepth2, we do not manually threshold the maximum depth, showing the entire depth estimation, including further distances. This is possibly the reason for theirs being artificially dark in the background. Remarkably, as seen already in Table 2 and Figure 6, our md4all-DD delivered more accurate and sharper estimates in both conditions, and especially at *night*, thanks to its robust feature extraction

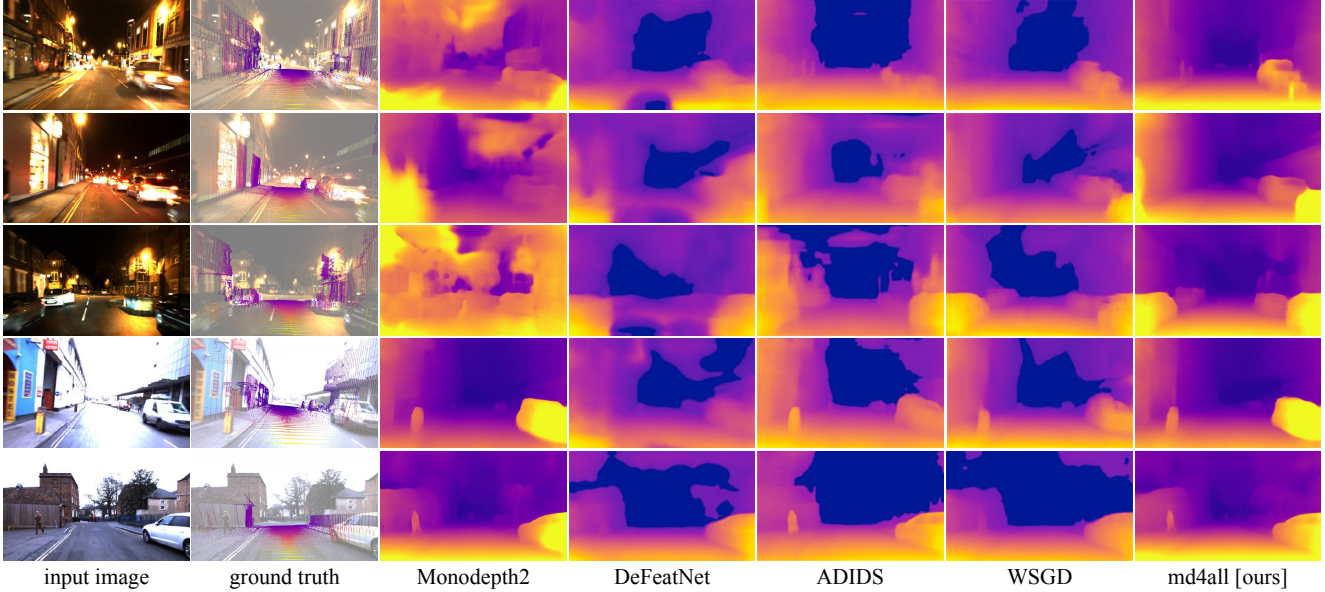


Figure 14. Comparison of self-supervised models on RobotCar [24] *night* and *day* samples. The samples are exactly the ones reported by Vankadari et al. in WSGD [38], from which we took directly the predictions of DeFeatNet [34], ADIDS [23], and WSGD [38]. We compare these with Monodepth2 [12] and our md4all-DD.

suitable for both *day* and *night* inputs.

**Night – RobotCar** Figure 15 shows additional outputs of the standard Monodepth2 [12] compared to our md4all-DD applied on Monodepth2. Although the *night* scenes are not as dark and not as noisy as those from nuScenes [4] (e.g., Figure 8), Monodepth2 had major issues estimating the depth of the image. Darkness and streetlights were detrimental for the standard Monodepth2 (e.g., in the second row). Moreover, compared to nuScenes, the images from RobotCar are often blurry at night, especially those from turns (first row). Nevertheless, the proposed md4all estimated reasonable depth maps regardless of these issues in the input. Due to the textureless pure white sky in the *day* samples of RobotCar (last two rows), the models had issues capturing its depth during training. The same occurred for the top of buildings, which are often too bright and indistinguishable from the sky in the images (e.g., fourth input in Figure 14). This problem is inherent to the data itself and causes the sky to always be predicted to be relatively close (brighter color). This did not happen for nuScenes.

**Day – RobotCar** Figure 15 also shows additional predictions during daytime (bottom rows). As seen in Table 2, while performing significantly better at *night*, the performance of the proposed method does not degrade during the *day* compared to the standard Monodepth2 [12]. Due to the knowledge distillation from the baseline to our md4all-DD model, surfaces result smoother, but edges remain sharp. The bottom input is particularly challenging as it features a turn. Both models correctly estimated the scene’s depth, with Monodepth2 delivering more details on the tree in the

foreground but less on the tree in the background (left). As discussed for the *night* scenes, the textureless white sky of RobotCar prevents the models from learning its depth correctly, which causes erroneous estimations at test time. Instead, nuScenes [4] includes different sky conditions (e.g., sunny and cloudy), allowing the models to learn its depth.

#### A.4.4 Failure Cases

While our techniques bring significant improvements across various conditions, there is still room for improvement. Figure 16 reports failure cases of our models. Inherited from Monodepth2 [12], our self-supervised model has issues with dynamic objects, especially oncoming traffic, whose distance is wrongly estimated due to the violations of the moving camera in a static world assumption [9]. This occurs in all three conditions. Furthermore, the model is occasionally fooled by reflections on the ground or misleading shadows which look like objects (first and sixth row). Additionally, highly dark scenes are also challenging (third row) due to the lack of information across nearly black pixels. These issues could be mitigated by integrating the data from the cost-effective radar, as in R4Dyn [9], which is robust against adverse weather and challenging illumination conditions. Our method is not bound to a specific architecture or pipeline, so it could be applied to R4Dyn directly. Another drawback our md4all inherited from Monodepth2, which we did not address, is the lack of temporal consistency in the depth predictions. Instead, for the supervised setting, we focused on eliminating the artifacts due to the erroneous ground truth measurements. While such issues





Figure 15. Comparison of self-supervised models on RobotCar [24] *night* and *day* samples. The standard Monodepth2 [12] is compared to our md4all-DD applied to Monodepth2.

are appropriately addressed, other problems persist, such as with particular reflections and the blur caused by water drops (third row). Radar may help here too.

#### A.4.5 Day-to-Adverse Translation

This section shows samples of the translated images we generated with ForkGAN [46]. We publicly share all translated images corresponding to the ideal settings from the training sets of nuScenes [4] and RobotCar [24].

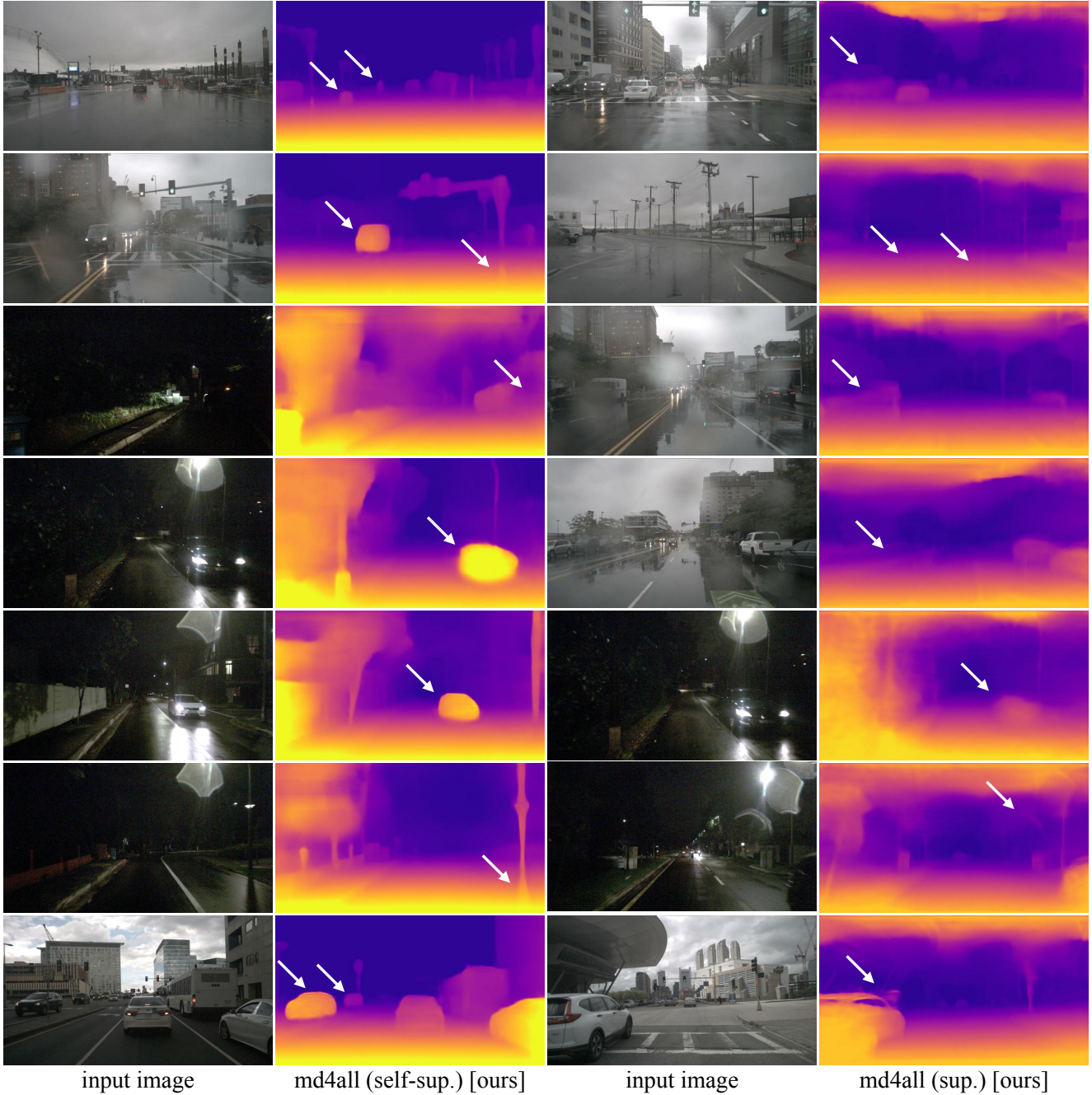


Figure 16. Failure cases of our self-supervised (md4all-DD, Monodepth2-based) and supervised (md4all-AD, AdaBins-based) models on samples from nuScenes [4]. White arrows mark issues in the predictions.

**nuScenes** In Figure 17, we show examples of the translated images used to train our models on nuScenes. The GAN [46] added plausible reflections and lights simulating wet ground or streetlights. While these additions are not always realistic, they bring challenging variations to the training set, resembling the adverse conditions in the first and last columns. The standard *day-clear* training set includes a mix of sunny and cloudy scenes. For cloudy ones, shadows

are limited, with less contrast overall, making it easier for the translation model. Instead, the inputs with sunny conditions are particularly challenging for the GAN, resulting in less convincing outputs (e.g., in the last row). The GAN also learned to add water drops blurring certain areas for *rain*, and intense noise for *night*. The results of our models would directly benefit from improvements in the translations towards greater realism. With perfect translations,





Figure 17. Example of day-to-adverse image translations on nuScenes [4]. Training samples are translated from *day-clear* to both *rain* and *night*. For reference, in the first and last columns, we included real *rain* and *night* samples from the validation set. We share publicly the translated *night* and *rain* images for the entire *day-clear* training set.

our md4all would further reduce the gap between ideal and challenging conditions.

**RobotCar** In Figure 18, we report examples of the translated images used on RobotCar. In this case, the GAN was trained entirely on RobotCar, thanks to the high amount of *night* samples available. Since the weather in the *day* samples tends to be always cloudy, with a consistently white sky, the GAN did not exhibit the issues seen for nuScenes with sunny inputs (Figure 17). Instead, the GAN delivered highly plausible samples with a higher degree of realism than for nuScenes, enabling our model to narrow the margin between daytime and nighttime performances. As seen for nuScenes, the GAN added streetlights resembling the ones seen throughout the dataset. Furthermore, it can be seen clearly that the GAN learned the headlights of the ego vehicle, which it added to illuminate the road ahead.

For both datasets, while being somewhat plausible, the translated samples are not perfect. For nuScenes, for example, noise patterns in the *night* translations are repeated similarly over the lower left corner of the images. The noise is more unstructured for the real *night* images. Additionally, the GAN did not learn to turn on the lights of other cars, which are often a source of issues for the models.

### A.5. Attempted Approaches That Did Not Work

We explored several alternative solutions for this challenging problem, and we intend to mention them in this section to help future researchers who target the same issues. As written in Section 4.1, we experimented with diffusion models to learn our day-to-adverse image translation. However, the lack of paired images for *day* and *night* made it unfeasible. Therefore, we opted for GANs, which do not need



Figure 18. Example of day-to-adverse image translations on RobotCar [24]. Training samples are translated from *day* to *night*. For reference, we included real *night* samples from the test set in the central column. We share publicly the translated *night* images for the entire *day* training set.

paired inputs. Furthermore, we experimented with the following methods on the *night* samples of nuScenes, seeking improvements over the standard Monodepth2. Spatial and temporal attention could have allowed the model to focus on valuable information. Still, it did not bring an improvement, possibly due to the large amount of noise varying across different *night* images, which could have prevented to attend on the helpful information. Additionally, we experimented with incorporating geometrical priors (e.g., from edge detection) into the losses, but they were similarly not beneficial. The simple solution presented in this work was the most effective to tackle this complex problem.

## References

- [1] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. AdaBins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021. 1, 2, 3, 6, 7, 8, 10, 14, 15, 17, 18, 21, 23, 24, 25
- [2] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11682–11692, 2020. 6, 7, 14, 15
- [3] Keenan Burnett, David J Yoon, Yuchen Wu, Andrew Z Li, Haowei Zhang, Shichen Lu, Jingxing Qian, Wei-Kang Tseng, Andrew Lambert, Keith YK Leung, Angela P Schoellig, and Timothy D Barfoot. Boreas: A multi-season autonomous driving dataset. *The International Journal of Robotics Research*, 42(1-2):33–42, 2023. 15
- [4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
- [5] Carlos A Diaz-Ruiz, Youya Xia, Yurong You, Jose Nino, Junan Chen, Josephine Monica, Xiangyu Chen, Katie Luo, Yan Wang, Marc Emond, et al. Ithaca365: Dataset and driving perception under repeated and challenging weather conditions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21383–21392, 2022. 6
- [6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*, 27, 2014. 2
- [7] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018. 2
- [8] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proceedings of the European Conference on Computer Vision*, pages 740–756. Springer, 2016. 1, 3



- [9] Stefano Gasperini, Patrick Koch, Vinzenz Dallabetta, Nassir Navab, Benjamin Busam, and Federico Tombari. R4Dyn: Exploring radar for self-supervised monocular depth estimation of dynamic scenes. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 751–760. IEEE, 2021. [1](#), [2](#), [3](#), [6](#), [7](#), [8](#), [11](#), [12](#), [15](#), [18](#), [26](#)
- [10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. [2](#)
- [11] Clément Godard, Oisín Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017. [1](#), [3](#), [4](#), [10](#)
- [12] Clément Godard, Oisín Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. [1](#), [2](#), [3](#), [4](#), [6](#), [7](#), [8](#), [10](#), [12](#), [13](#), [14](#), [15](#), [16](#), [17](#), [18](#), [19](#), [20](#), [21](#), [22](#), [23](#), [25](#), [26](#), [27](#)
- [13] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Rantotas, and Adrien Gaidon. 3D packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2485–2494, 2020. [1](#), [2](#), [3](#), [4](#), [6](#), [10](#), [12](#), [13](#), [14](#)
- [14] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. In *International Conference on Learning Representations*, 2020. [2](#), [3](#)
- [15] Vitor Guizilini, Kuan-Hui Lee, Rareş Ambruş, and Adrien Gaidon. Learning optical flow, depth, and scene flow without real-world labels. *IEEE Robotics and Automation Letters*, 7(2):3491–3498, 2022. [3](#), [8](#)
- [16] Vitor Guizilini, Igor Vasiljevic, Rares Ambrus, Greg Shakhnarovich, and Adrien Gaidon. Full surround monodepth from multiple cameras. *IEEE Robotics and Automation Letters*, 7(2):5397–5404, 2022. [3](#)
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. [6](#)
- [18] Kuan-Chih Huang, Tsung-Han Wu, Hung-Ting Su, and Winston H Hsu. MonoDTR: Monocular 3d object detection with depth-aware transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4012–4021, 2022. [2](#)
- [19] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proceedings of the European Conference on Computer Vision*, pages 53–69, 2018. [3](#)
- [20] HyunJun Jung, Patrick Ruhkamp, Guangyao Zhai, Nikolas Brasch, Yitong Li, Yannick Verdie, Jifei Song, Yiren Zhou, Anil Armagan, Slobodan Ilic, Aleš Leonardis, Nassir Navab, and Benjamin Busam. On the importance of accurate geometry data for dense 3D vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 780–791, 2023. [1](#), [2](#), [3](#), [5](#)
- [21] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 239–248. IEEE, 2016. [2](#)
- [22] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019. [2](#)
- [23] Lina Liu, Xibin Song, Mengmeng Wang, Yong Liu, and Liangjun Zhang. Self-supervised monocular depth estimation for all day images using domain separation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12737–12746, 2021. [2](#), [3](#), [4](#), [6](#), [7](#), [12](#), [25](#), [26](#)
- [24] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017. [2](#), [5](#), [6](#), [7](#), [8](#), [11](#), [13](#), [14](#), [17](#), [19](#), [26](#), [27](#), [30](#)
- [25] Vaishakh Patil, Christos Sakaridis, Alexander Liniger, and Luc Van Gool. P3Depth: Monocular depth estimation with a piecewise planarity prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1610–1621, 2022. [2](#)
- [26] Andra Petrovai and Sergiu Nedevschi. Exploiting pseudo labels in a self-supervised learning framework for improved monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1578–1588, 2022. [1](#), [3](#)
- [27] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3227–3237, 2020. [3](#)
- [28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. [6](#)
- [29] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *Proceedings of the ACM SIGGRAPH Conference*, pages 1–10, 2022. [6](#)
- [30] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10765–10775, 2021. [6](#)
- [31] Peilun Shi, Jiachuan Peng, Jianing Qiu, Xinwei Ju, Frank Po Wen Lo, and Benny Lo. EVEN: An event-based framework for monocular depth estimation at adverse night conditions. *arXiv preprint arXiv:2302.03860*, 2023. [2](#), [3](#)
- [32] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from

- RGBD images. *Proceedings of the European Conference on Computer Vision*, 7576:746–760, 2012. [2](#)
- [33] Xibin Song, Wei Li, Dingfu Zhou, Yuchao Dai, Jin Fang, Hongdong Li, and Liangjun Zhang. MLDA-Net: Multi-level dual attention-based network for self-supervised monocular depth estimation. *IEEE Transactions on Image Processing*, 30:4691–4705, 2021. [1](#)
- [34] Jaime Spencer, Richard Bowden, and Simon Hadfield. DeFeat-Net: General monocular depth via simultaneous unsupervised representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14402–14413, 2020. [2](#), [3](#), [6](#), [7](#), [12](#), [25](#), [26](#)
- [35] Fabio Tosi, Filippo Aleotti, Matteo Poggi, and Stefano Mattoccia. Learning monocular depth estimation infusing traditional stereo knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9799–9809, 2019. [1](#), [2](#)
- [36] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979. [3](#)
- [37] Madhu Vankadari, Sourav Garg, Anima Majumder, Swagat Kumar, and Ardhendu Behera. Unsupervised monocular depth estimation for night-time images using adversarial domain feature adaptation. In *Proceedings of the European Conference on Computer Vision*, pages 443–459. Springer, 2020. [2](#), [3](#)
- [38] Madhu Vankadari, Stuart Golodetz, Sourav Garg, Sangyun Shin, Andrew Markham, and Niki Trigoni. When the Sun Goes Down: Repairing photometric losses for all-day depth estimation. In *Proceedings of the Conference on Robot Learning*, pages 1992–2003. PMLR, 2023. [2](#), [3](#), [6](#), [7](#), [8](#), [11](#), [12](#), [13](#), [14](#), [25](#), [26](#)
- [39] Kun Wang, Zhenyu Zhang, Zhiqiang Yan, Xiang Li, Baobei Xu, Jun Li, and Jian Yang. Regularizing Nighttime Weirdness: Efficient self-supervised monocular depth estimation in the dark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16055–16064, 2021. [2](#), [3](#), [6](#), [7](#), [11](#), [12](#), [15](#)
- [40] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. [4](#), [10](#)
- [41] Mengchen Xiong, Xiao Xu, Dong Yang, and Eckehard Steinbach. Robust depth estimation in foggy environments combining RGB images and mmWave radar. In *International Symposium on Multimedia*, pages 34–41. IEEE, 2022. [2](#), [3](#)
- [42] Jiaxing Yan, Hong Zhao, Penghui Bu, and YuSheng Jin. Channel-wise attention-based network for self-supervised monocular depth estimation. In *International Conference on 3D vision (3DV)*, pages 464–473. IEEE, 2021. [1](#)
- [43] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020. [6](#), [11](#)
- [44] Chaoqiang Zhao, Yang Tang, and Qiyu Sun. Unsupervised monocular depth estimation in highly complex environments. *Transactions on Emerging Topics in Computational Intelligence*, 6(5):1237–1246, 2022. [3](#), [4](#)
- [45] Chaoqiang Zhao, Youmin Zhang, Matteo Poggi, Fabio Tosi, Xianda Guo, Zheng Zhu, Guan Huang, Yang Tang, and Stefano Mattoccia. MonoViT: Self-supervised monocular depth estimation with a vision transformer. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 668–678. IEEE, 2022. [3](#)
- [46] Ziqiang Zheng, Yang Wu, Xinran Han, and Jianbo Shi. ForkGAN: Seeing into the rainy night. In *Proceedings of the European Conference on Computer Vision*, pages 155–170. Springer, 2020. [6](#), [7](#), [8](#), [11](#), [15](#), [27](#), [28](#)
- [47] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017. [3](#), [4](#)
- [48] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017. [7](#)