Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

Akhil Gurram\textsuperscript{1,2}, Ahmet Faruk Tuna\textsuperscript{2}, Fengyi Shen\textsuperscript{2,3}, Onay Urfalioglu\textsuperscript{2}, and Antonio M. López\textsuperscript{1,4}

Abstract—Depth information is essential for on-board perception in autonomous driving and driver assistance. Monocular depth estimation (MDE) is very appealing since it allows for appearance and depth being on direct pixelwise correspondence without further calibration. Best MDE models are based on Convolutional Neural Networks (CNNs) trained in a supervised manner, i.e., assuming pixelwise ground truth (GT). Usually, this GT is acquired at training time through a calibrated multi-modal suite of sensors. However, also using only a monocular system at training time is cheaper and more scalable. This is possible by relying on structure-from-motion (SfM) principles to generate self-supervision. Nevertheless, problems of camouflage, object, visibility changes, static-camera intervals, textureless areas, and scale ambiguity, diminish the usefulness of such self-supervision.

In this paper, we perform monocular depth estimation by virtual-world supervision (MonoDEVS) and real-world SfM self-supervision. We compensate the SfM self-supervision limitations by leveraging virtual-world images with accurate semantic and depth supervision, and addressing the virtual-to-real domain gap. Our MonoDEVSNet outperforms previous MDE CNNs trained on monocular and even stereo sequences.

Index Terms—Self-supervised monocular depth estimation, on-board vision, domain adaptation, ADAS, autonomous driving.

I. INTRODUCTION

Understanding the semantic content of images is enough to solve many vision-based applications. However, augmenting semantic information with depth is essential for many other applications too, as for on-board perception in autonomous driving and driver assistance. Due to cost and maintenance considerations, we wish to predict depth from the same single camera used to predict semantics, so having a direct pixelwise correspondence without further calibration. Therefore, in this paper, we focus on monocular depth estimation (MDE) on-board vehicles, thus, facing outdoor traffic environments. Recent advances on MDE rely on Convolutional Neural Networks (CNNs). Let $\Psi$ be a CNN architecture for MDE with weights $\theta$, which takes a single image $x$ as input, and estimates its pixelwise depth map $d$ as output, i.e., $\Psi(\theta; x) \to d$. The $\Psi$’s can be trained in a supervised manner, i.e., finding the values of $\theta$ by assuming access to a set of images with pixelwise depth ground truth (GT). Usually, such a GT is acquired at training time through a multi-modal suite of sensors, at least consisting of a camera calibrated with a LiDAR or some type of 3D laser scanner variant [1–10]. Alternatively, we can use self-supervision based on either a calibrated stereo rig [11–14], or a monocular system and structure-from-motion (SfM) principles [15–18], or on a combination of both [19]. Combining stereo self-supervision and LiDAR supervision has been also analyzed [20–22]. In any case, the cheaper and simpler the suite of sensors used at training time, the better in terms of scalability and general access to the technology; however, the more challenging training a $\Psi$. Currently, supervised methods tend to outperform self-supervised ones [23], thus, improving the latter is an open challenge worth to pursue.

This paper focuses on the most challenging setting, namely, when at training time we only have a single on-board camera allowing for SfM based self-supervision. Using only such a self-supervision may give rise to depth estimation inaccuracies due to camouflage (objects moving as the camera may not be distinguished from background), visibility changes (occlusion changes, non-Lambertian surfaces), static-camera cases (i.e., stopped ego-vehicle), and textureless areas, as well as to scale ambiguity (depth could only be estimated up to an unknown scale factor). In fact, an interesting approach to compensate for these problems could be leveraging virtual-world images (RGB) with accurate pixelwise depth (D) supervision. Using virtual worlds [24–30], we can acquire as many RGBD virtual-world samples as needed. However, these virtual-world samples can only be useful provided we address the virtual-to-real domain gap [31–35], which links MDE with visual domain adaptation (DA), a realm of research in itself [36–38].

We propose to perform monocular depth estimation by virtual-world supervision (MonoDEVS) and real-world SfM self-supervision, estimating depth in absolute scale. By relying on standard benchmarks, we show that our MonoDEVSNet outperforms previous ones trained on monocular and even stereo sequences. We think our released code and models\textsuperscript{1} will help researchers and practitioners to address applications requiring on-board depth estimation, also establishing a strong baseline to be challenged in the future.

In the following, Section II summarizes previous works related to ours. Section III details our proposal. Section IV describes the experimental setting and discusses the obtained results. Finally, Section V summarizes the presented work and conclusions, and draws the work we target for the near future.

\textsuperscript{1}https://github.com/HMRC-AEL/MonoDEVSNet
II. RELATED WORK

MDE was first addressed by combining hand-crafted features and shallow machine learning [11, 39–41]. However, nowadays, best performing models are based on CNNs [23]. Therefore, we review the CNN-based approaches to MDE which are most related to ours.

A. Supervised MDE

Relying on depth GT, Eigen et al. [1] developed a Ψ architecture for coarse-to-fine depth estimation with a scale-invariant loss function. This pioneering work inspired new CNN-based architectures to MDE [2–6, 8, 9], which also assume depth GT supervision. MDE has been also tackle as a task on a multi-task learning framework, typically together with semantic segmentation as both tasks aim at producing pixelwise information and, eventually, may help each other to improve their predictions at object boundaries. For instance, this is the case of some Ψ’s for indoor scenarios [42–44].

These proposals assume that pixelwise depth and class GT are simultaneously available at training time. However, this is expensive, being scarcely available for outdoor scenarios. In order to address this problem, Gurram et al. [7] proposed a training framework which does not require depth and class GT to be available for the same images. Guizilini et al. [45] used an out-of-the-box CNN for semantic segmentation to train semantically-guided depth features while training Ψ.

The drawback of these supervised approaches is that the depth GT usually comes from expensive LiDARs, which must be calibrated and synchronized with the cameras; i.e., even if the objective is to use only cameras for the functionality under development. Moreover, LiDAR depth is sparse compared to available image resolutions. Besides, surfaces like vehicle glasses or dark vehicles may be problematic for LiDAR sensing. Consequently, depth self-supervision and alternative sources of supervision are receiving increasing interest.

B. Self-supervised MDE

Using a calibrated stereo rig to provide self-supervision for MDE is a much cheaper alternative to camera-LiDAR suites. Garg et al. [12] pioneered this approach by training Ψ with a warping loss involving pairs of stereo images. Godard et al. [13] introduced epipolar geometry constraints with additional terms for smoothing and enforcing consistency between left-right image pairs. Chen et al. [46] improved MDE results by enforcing semantic consistency between stereo pairs, via a joint training of Ψ and semantic segmentation. Pillai et al. [14] implemented sub-pixel convolutional layers for depth super-resolution, as well as a novel differentiable layer to improve depth prediction on image boundaries, a known limitation of stereo self-supervision. Other authors [20, 21] still complement stereo self-supervision with sparse LiDAR supervision.

SfM principles [47] can be also followed to provide self-supervision for MDE. In fact, in this setting we can assume a monocular on-board system at training time. Briefly, the underlying idea is that obtaining a frame, x_t, from consecutive ones, x_{t±1}, can be decomposed into jointly estimating the scene depth for x_t and the camera pose at time t relative to its pose at time t ± 1; i.e., the camera ego-motion. Thus, we can train a CNN to estimate (synthesize) x_t from x_{t±1}, where, basically, the photo-metric error between x_t and ̂x_t acts as training loss, being ̂x_t the output of this CNN (i.e., the synthesized view). After the training process, part of the CNN can perform MDE up to a scale factor (relative depth).

Zhou et al. [15] followed this idea, adding an explainability mask to compensate for violations of SfM assumptions (due to frame-to-frame changes on the visibility of frame’s content, textureless surfaces, etc.). This mask is estimated by a CNN jointly trained with Ψ to output a pixelwise belief on the synthesized views. Later, Yin et al. [16] proposed GeoNet, which aims at improving MDE by also predicting optical flow to explicitly consider the motion introduced by dynamic objects (e.g., vehicles, pedestrians), i.e. a motion that violates SfM assumptions. However, this was effective on predicting occlusions, but not in significantly improving MDE accuracy.

Godard et al. [19] followed the idea of having a mask to indicate stationary pixels, which should not be taken into account by the loss driving the training. Such pixels typically appear on vehicles moving at the same speed as the camera, or can even correspond to full frames in case the ego-vehicle stops and, thus, the camera becomes stationary for a while. Pixels of similar appearance in consecutive frames are considered as stationary. A simple definition which can work because, instead of using a training loss based on absolute photometric errors (i.e., on minimizing pairwise pixel differences), it is used the structure similarity index measurement (SSIM) [48]. Moreover, within the so-called MonoDepth2 framework, Godard et al. [19] combine SfM and stereo self-supervision to establish state-of-the-art results. Alternatively, Guizilini et al. [45] addressed the presence of dynamic objects by a two-stage MDE training process. The first stage ignores the presence of such objects, returning a Ψ trained with a loss based on SSIM. Then, before running the second stage, the training sequences are processed to filter out frames that may contain erroneous depth estimations due to moving objects. Such frames are identified by applying Ψ, a RANSAC algorithm to estimate the ground plane from their estimated depth, and determining if there is a significant number of pixels that would be projected far below the ground plane. Finally, in the second stage, Ψ is retrained from scratch without the filtered frames.

Zhu et al. [17] focused on avoiding scale inconsistencies among different frames as produced under SfM self-supervision, specially when they are from sequences whose underlying depth range is too different. In this case, both depth and optical flow estimation CNNs are trained, but not a pose estimation one. Instead, the optical flow between two frames is used to find robust pixel correspondences between them, which are used to compute their relative camera pose, computing the fundamental matrix by the 8-point algorithm, and then performing triangulation between the corresponding pixels of these frames. Overall, a sparse depth pseudo-GT is estimated and used as supervision to train Ψ. However, even robustifying scale consistency among frames, this method still outputs relative depth rather than absolute one. To avoid this problem, Guizilini et al. [22] used sparse LiDAR supervision with...
SfM self-supervision, relying on depth and pose estimation networks. More recently, Guizilini et al. [18] relied on camera velocity (i.e., the available vehicle velocity) to solve scale ambiguity in a pure SfM self-supervision setting. In particular, a velocity supervision loss trains the pose estimation CNN to learn scale-aware camera translation which, in turn, enables scale-aware depth estimation.

Overall, this literature shows the relevance of achieving MDE via SfM self-supervision and strategies to account for violation of SfM assumptions, as well as to obtain absolute depth values. Among these strategies, complementing SfM self-supervision with supervision (depth GT) coming from additional sensors such as a LiDAR and/or a stereo rig seems to be the most robust approach to address all the problems at once. However, then, a single camera would not be enough at training time. In this paper, we also complement SfM self-supervision with accurate depth supervision. However, instead of relying on additional sensors, we use virtual-world data.

C. Virtual-world data for MDE

Training $\Psi$ on virtual-world images to later perform on real-world ones, requires to address the virtual-to-real domain gap. Many approaches perform a virtual-to-real image-to-image translation coupled to the training of $\Psi$. This translation usually relies on generative adversarial networks (GANs) [49, 50], since to train them only unpaired and unlabeled sets of real- and virtual-world images are required.

Zheng et al. [31] proposed T$^2$Net. In this case, a GAN and $\Psi$ are jointly trained, where the GAN aims at performing virtual-to-real translation while acting as an auto-encoder for real-world images. The translated images are the input for $\Psi$ since they have depth supervision. Additionally, a GAN operating on the encoder weights (features) of $\Psi$ was incorporated during training to force similar depth feature distributions between translated and real-world images. However, this feature-level GAN worsen MDE results in outdoor scenarios. Kundu et al. [32] proposed AdaDepth, which trains a common feature space for real- and virtual-world images, i.e., a space where it is not possible to distinguish the domain of the input images. Then, depth estimation is trained from this feature space. To achieve this, adversarial losses are used at the feature space level as well as at the estimated depth level.

Cheng et al. [35] proposed $S^3$Net, which extends T$^2$Net with SfM self-supervision. In this case, GAN training involves semantic and photo-metric consistency. Semantic consistency between the virtual-world images and their GAN-translated counterparts is required, which is measured via semantic segmentation (which involves also to jointly train a CNN for this task). Photo-metric consistency is required for consecutive GAN-translated images, which is measured via optical flow. Note that semantic segmentation and optical flow GT is available for virtual-world images. $\Psi$ uses the GAN-translated images as input and is trained end-to-end with the GAN. Then, a further fine-tuning step of $\Psi$ is performed using only the real-world sequence and SfM self-supervision, i.e., involving the training of a pose estimation CNN while fine-tuning. During this process, a masking mechanism inspired in [19] is also used to compensate for SfM-adverse scenarios. Contrary to AdaDepth and T$^2$Net, $S^3$Net just outputs relative depth.

Zhao et al. [33] proposed GASDA, which leverages real-world stereo and virtual-world data. In this case, the CycleGAN idea [51] is used to perform DA, which actually involves two GANs, one for virtual-to-real image translation and another for real-to-virtual. Two $\Psi$’s are trained coupled to CycleGAN, one intended to process images with real-world appearance (actual real-world images or GAN-translated from the virtual domain), the other to process images with synthetic appearance (actual virtual-world images or GAN-translated from the real domain). In fact, at testing time, the most accurate depth results are obtained by averaging the output of these two $\Psi$’s, which also involves to translate the real-world images to the virtual domain by the corresponding GAN. Thanks to the stereo data, left-right depth and geometry consistency losses are also included during training aiming at obtaining a more accurate $\Psi$. PNVR et al. [34] proposed SharinGAN for training a DA GAN coupled to a specific task. One of the selected tasks is MDE with stereo self-supervision, as in [33]. In this case, real- and virtual-world images are transformed to a new image domain where their appearance discrepancies are minimized to perform MDE from them, i.e, the GAN and the $\Psi$ are jointly trained end-to-end. SharinGAN outperformed GASDA. However, at testing time, before performing the MDE, the real-world images must be translated by the GAN to the new image domain. Both GASDA and SharinGAN produce absolute scale depth.

D. Relationship of MonoDEVSNet with previous literature

In term of operational training conditions, the most similar paper to ours is $S^3$Net [35]. However, contrary to $S^3$Net, our MonoDEVSNet can estimate depth in absolute scale. On the other hand, methods based on pure SfM self-supervision such as [15], [16], [19] (only SfM setting), and [45], just report relative depth. In order to compare MonoDEVSNet with them, we have estimated relative depth too. We will see how we outperform these methods, proving the usefulness of leveraging depth supervision from virtual worlds. In fact, regarding relative depth, we also outperform $S^3$Net. Methods leveraging virtual-world data such as GASDA [33] and SharinGAN [34], rely on real-world stereo data at training time, while we only require monocular sequences. On the other hand, our training framework can be extended to accommodate stereo data if available, although it is not our current focus. $S^3$Net, GASDA, SharinGAN, T$^2$Net [31], and AdaDepth [32], leverage ideas from GAN-based DA to reduce the virtual-to-real domain gap, either in image space ($S^3$Net, GASDA, SharinGAN, T$^2$Net) or in feature space (AdaDepth). We have analyzed both, image and feature based DA, finding that the later outperforms the former. In particular, by using the Gradient-Reversal-Layer (GRL) DA strategy [52, 53], up to the best of our knowledge, not previously applied to MDE. Currently, we outperform the SfM self-supervision framework in [18] thanks to the virtual-world supervision and our GRL DA strategy. However, using vehicle velocity to obtain absolute depth as in [18], is a complementary strategy that could be also incorporated in our framework, although it is not the focus on this paper.
**III. METHODS**

In this section, we introduce MonoDEVSNet, which aims at leveraging virtual-world supervision to improve real-world SfM self-supervision. Since we train from both real- and virtual-world data jointly, we describe our supervision and self-supervision losses, the loss for addressing the virtual-to-real domain gap, and the strategy to obtain depth in absolute scale. Our proposal is visually summarized in Fig. 1.

**A. Training data**

For training MonoDEVSNet, we assume two sources of data. On the one hand, we have image sequences acquired by a monocular system on-board a vehicle while driving in real-world traffic. We denote as $x^r_t$ one of such frames acquired at time $t$. We denote these data as $X^r = \{x^r_t\}_{t=1}^{N^r}$, where $N^r$ is the number of frames from the real-world sequences. These frames do not have associated GT. On the other hand, we have analogous sequences but acquired on a virtual world, i.e., on-board a vehicle immersed in a traffic simulation. We denote as $x^c_t$ one of such virtual-world frames acquired at time $t$. We refer to these data as $X^c = \{x^c_t\}_{t=1}^{N^c}$, where $N^c$ is the number of frames from the virtual-world sequences. The images in $X^c$ do have associated GT, since it can be automatically generated. In particular, as it is commonly available in today’s simulators, we assume pixelwise depth and semantic class GT. We define $Y^s = \{<d^1_t, c^1_t>\}_{t=1}^{N^s}$ to be this GT; i.e., given $x^r_t$, $d^1_t$ is its depth GT, and $c^1_t$ its semantic class GT.

**B. MonoDEVSNet architecture: $\Psi(\theta; x)$**

MonoDEVSNet, i.e., our $\Psi(\theta; x)$, is composed of three main blocks: a encoding block of weights $\theta^{enc}$, a multi-scale pyramidal block, $\theta^{pyr}$, and a decoding block inspired in [19], $\theta^{dec}$. Therefore, the total set of weights is $\theta = \{\theta^{enc}, \theta^{pyr}, \theta^{dec}\}$. Here, $\theta^{enc}$ acts as a backbone of features. Moreover, since we aim at evaluating several encoders, the role of the multi-scale pyramid block is to adapt the bottleneck of the chosen encoder to the decoder. At testing time $\Psi(\theta; x)$ will process any real-world image $x$ acquired on-board the ego-vehicle, while at training time either $x \in X^r$ or $x \in X^c$.

**C. Problem formulation**

Training $\Psi(\theta; x)$ consists in finding the optimum weight values, $\theta^*$, by solving the problem:

$$\theta^* = \min_{\theta} \mathcal{L}(\theta; X^r, X^c, Y^s)$$

(1)

where $\mathcal{L}$ is a loss function, and $X^r, Y^s$ indicates the use of the virtual-world frames with their GT. As we are going to detail, $\mathcal{L}$ relies on three different losses, namely, $\mathcal{L}^{sd}, \mathcal{L}^{op}$ and $\mathcal{L}^{om}$. The loss $\mathcal{L}^{sd}$ focuses on training $\theta$ based on SfM self-supervision, thus, only relying on real-world data sequences. The SfM self-supervision is achieved with the support of a camera pose estimation task performed by a CNN, $T$, of weights $\theta^{sd}$. Thus, we have $\mathcal{L}^{sd}(\theta, \theta^{sd}, X^r)$. The loss $\mathcal{L}^{op}$ focuses on training $\theta$ with virtual-world supervision, in particular, using depth and semantic GT from virtual-world sequences. Therefore, we have $\mathcal{L}^{op}(\theta; X^c, Y^s)$. Finally, $\mathcal{L}^{om}$ focuses on creating domain-invariant features $\theta^{enc}$ as part of $\theta$. In particular, we rely on a binary real/virtual domain-classifier CNN, $D$, of weights $\{\theta^{enc}, \theta^{dom}\}$. Thus, we have $\mathcal{L}^{om}(\theta^{enc}, \theta^{dom}; X^r, X^c)$.

**D. SfM Self-supervised loss: $\mathcal{L}^{sd}(\theta, \theta^{sd}; X^r)$**

Since we focus on improving MDE by the additional use of virtual-world data, for the SfM self-supervision we leverage from the state-of-the-art proposal in [19], which we briefly summarize here for the sake of completeness as:

$$\mathcal{L}^{sd}(\theta, \theta^{sd}; X^r) = \sum_{t=2}^{N^r-1} P_{\mathcal{L}}(\theta, \theta^{sd}) + \lambda S_{\mathcal{L}}^c(\theta)$$

(2)

As previously introduced in [13], the term $\lambda S_{\mathcal{L}}^c(\theta)$ is a constant weighted loss to force local smoothness on $\Psi(\theta; x^c_t)$, taking into account the edges of $x^c_t$. The term $P_{\mathcal{L}}(\theta, \theta^{sd})$ is the actual SfM-inspired loss. It involves the joint training of the depth estimation weights, $\theta$, and the relative camera pose estimation weights, $\theta^{sd}$. Figure 1 illustrates the CNN, $T$, associated to these weights, which takes as input two consecutive frames, e.g., $(x^c_t, x^c_{t+1})$, and outputs the pose transform (rotation and translation). $\hat{T}^{c}_{t-1:t+1} = T(\theta^{sd}, x^c_t, x^c_{t+1})$, between them. Then, as can be seen in Fig. 1, a projection module takes $\hat{T}^{c}_{t-1:t+1}, x^c_{t+1}$, and the depth estimation $\Psi(\theta; x^c_t)$, to generate the synthesized frame $\hat{x}^c_{t+1\rightarrow t+1}(\theta, \theta^{sd})$ which, ideally, should match $x^c_t$. In fact, both frames adjacent to $x^c_t$ are considered for robustness. Accordingly, the SFM-inspired component of $\mathcal{L}^{sd}$ can be defined as:

$$P_{\mathcal{L}}(\theta, \theta^{sd}) = cpe(x^c_t, \hat{x}^c_{t\rightarrow t+1}(\theta, \theta^{sd}))$$

(3)

where $cpe()$ is a pixelwise conditioned photo-metric error and $cpe()$ its average over the pixels. Obtaining $cpe()$ starts by computing two pixelwise photo-metric error measurements, $pe(x^c_t, x^c_{t+1})$ and $pe(\hat{x}^c_{t\rightarrow t+1}(\theta, \theta^{sd}), \hat{x}^c_{t\rightarrow t+1}(\theta, \theta^{sd}))$. The loss $\mathcal{L}^{op}$ focuses on training $\theta$ with virtual-world supervision, in particular, using depth and semantic GT from virtual-world sequences. Therefore, we have $\mathcal{L}^{op}(\theta; X^c, Y^s)$. Finally, $\mathcal{L}^{om}$ focuses on creating domain-invariant features $\theta^{enc}$ as part of $\theta$. In particular, we rely on a binary real/virtual domain-classifier CNN, $D$, of weights $\{\theta^{enc}, \theta^{dom}\}$. Thus, we have $\mathcal{L}^{om}(\theta^{enc}, \theta^{dom}; X^r, X^c)$.
where \( p_e(x'_{t-1}, x'_t, x'_{t+1}) = \min(p_e(x'_0, x'_{t-1}), p_e(x'_0, x'_{t+1})) \), and \( p_e(x'_{tA}, x'_{tB}) \) is the pixelwise photo-metric error between \( x'_{tA} \) and \( x'_{tB} \) proposed in [13], i.e., based on local structural similarity (SSIM) and pixelwise photo-metric absolute differences between \( x'_{tA} \) and \( x'_{tB} \). Thus, \( \min() \) applies pixelwise. Then, a pixelwise binary mask, called auto-mask in [19], is computed as:

\[
\omega^e (x'_t, x'_{t+1}) = \begin{cases} 
1 & \text{if } p_e(x'_{t-1}, x'_t, x'_{t+1}) < p_e(x'_0, x'_t, x'_t) \\
0 & \text{otherwise}
\end{cases},
\]

where \( [\cdot] \) denotes the Iverson bracket applied pixelwise. Finally, \( cpe() \) is computed as:

\[
cpe(x'_t, x'_{t+1}) = \omega^e (x'_t, x'_{t+1}) \circ \omega^e (x'_{t-1}, x'_t, x'_{t+1})
\]

where \( \circ \) stands for pixelwise multiplication. The auto-mask \( \omega^e (x'_t, x'_{t+1}) \) conditions which pixels of \( p_e() \) are considered during the gradient computation of \( L^d \), i.e., \( \omega^e (x'_t, x'_{t+1}) \) is computed in the forward pass of CNN training, but it is considered as a constant during back-propagation. As explained in [19], the aim of \( \omega^e (x'_t, x'_{t+1}) \) is to remove, during training, the influence of pixels which remain the same between adjacent frames because they are assumed to often indicate SFM violations such as a static camera, objects moving as the camera, or low texture regions. Finally, we remark that the support of \( \vartheta^d \) is needed at training time, but not at testing time.

### E. Supervised loss: \( L^s(\theta; X^s, Y^s) \)

In this case, since we address an estimation problem and we have accurate GT, we base \( L^s \) on the L1 metric. On the other hand, MDE is specially interesting to determine conditions which pixels of \( p_e() \) are considered during back-propagation. As explained in [19], the aim of \( \omega^e (x'_t, x'_{t+1}) \) is to remove, during training, the influence of pixels which remain the same between adjacent frames because they are assumed to often indicate SFM violations such as a static camera, objects moving as the camera, or low texture regions. Finally, we remark that the support of \( \vartheta^d \) is needed at training time, but not at testing time.

### F. Domain adaptation loss: \( L^\text{DA}(\theta^\text{enc}, \theta^\text{dec}; X^r, X^s) \)

As can be seen in Fig. 1, we aim at learning depth features, \( \theta^\text{enc} \), so that it cannot be distinguished whether they were generated from a real-world input frame (target domain) or a virtual-world one (source domain); in other words, learning a domain invariant \( \theta^\text{enc} \). Taking into account that we do not have accurate depth GT in the target domain, while we do have it for the source domain, we need to apply an unsupervised DA technique to train \( \theta^\text{enc} \). In addition, as part of \( \theta \), the training of \( \theta^\text{enc} \) must result on an accurate \( \Psi(\theta; x) \). Achieving this accuracy and domain invariance are adversarial goals. Accordingly, we propose to use the Gradient-Reversal-Layer (GRL) idea introduced in [52], which, up to the best of our knowledge, has not been applied before for DA in the context of MDE. In this approach, the domain invariance of \( \theta^\text{enc} \) is measured by a binary target/source domain-classifier CNN, \( D \), of weights \{\( \theta^\text{enc} \), \( \theta^\text{dec} \)\}. In [52], a logistic loss is proposed to train the domain classifier. In our case, this is set as:

\[
L^\text{DA}(\theta^\text{enc}, \theta^\text{dec}; X^r, X^s) = \sum_{t=1}^{N^r} \log(D(\theta^\text{enc}, \theta^\text{dec}; x'_t)) + \sum_{t=1}^{N^s} \log(1 - D(\theta^\text{enc}, \theta^\text{dec}; x'_t))
\]

where we assume that \( D(\theta^\text{enc}, \theta^\text{dec}; x) \) outputs 1 if \( x \in X^r \) and 0 if \( x \in X^s \). The GRL has no parameters and connects \( \theta^\text{enc} \) with \( \theta^\text{dec} \) (see Fig. 1). Its behavior is exactly as explained in [52]. This means that during forward passes of training, it acts as an identity function, while, during back-propagation, it reverses the gradient vector passing through it. Both the GRL and \( \theta^\text{dec} \) are required at training time, but not at testing time.

### G. Overall training procedure

Algorithm 1 summarizes the steps to compute the needed gradient vectors for mini-batch optimization. In particular, we need the gradients related to MonoDEVSNet weights, \( \theta = \{\theta^\text{enc}, \theta^\text{pyr}, \theta^\text{dec}\} \), and the weights of the auxiliary tasks, i.e., \( \vartheta^d \) for SFM self-supervision, and \( \vartheta^\text{DA} \) for DA. Regarding gradient computation, we do not need to distinguish \( \vartheta^\text{DA} \) from \( \vartheta^\text{dec} \), so we define \( \vartheta^\text{DA} = \{\vartheta^\text{DA}, \vartheta^\text{dec}\} \). In Alg. 1, we introduce an equalizing factor between supervised and self-supervised losses, \( \lambda^\text{DA} \in \mathbb{R} \), which aims at avoiding one loss dominating the other along the training. A priori, we could set a constant factor. However, in practice, we have found that having an adaptive value is more useful. Therefore, inspired by the GradNorm idea [54], we use the ratio between the supervised and self-supervised losses. Algorithm 1 also introduces the scaling factor \( \vartheta^\text{DA} \in \mathbb{R} \), which, following [52], controls the trade-off between optimizing \( \theta^\text{enc} \) to obtain an accurate \( \Psi(\theta; x) \) model versus being domain invariant. Finally, \( L^\text{DA}(\theta^\text{enc}, \theta^\text{dec}; X^r, X^s) \) and \( L^\text{DA}(\theta^\text{enc}, \theta^\text{dec}; X^r, \emptyset) \) indicate whether this loss must be computed only using virtual- or real-world data, respectively.

### H. Absolute depth computation

The virtual-world supervised data trains \( \Psi(\theta; x) \) on absolute depth values, while the real-world SFM self-supervised data trains \( \Psi(\theta; x) \) on relative depth values. Thanks to the unsupervised DA, the depth features \( \theta^\text{enc} \) are trained to be domain invariant. However, according to our experiments, this is not sufficient for \( \Psi(\theta; x) \) producing accurate absolute depth values.
at testing time. Fortunately, thanks to the use of virtual-world data, we can still compute a global scaling factor, $\psi \in \mathbb{R}$, so that $\psi \Psi(\theta; x)$ is accurate in absolute depth terms. For that, we assume that the sequences in $X^s$ are acquired with a camera analogous to the one used to acquire the sequences in $X^r$. Here analogous refers to using the same number of pixels, field of view, frame rate, and mounted on-board in similar heading directions. Note that simulators are flexible enough for setting these camera parameters as needed. Accordingly, we train a $\Psi(\theta; x)$ model using only data from $X^s$ and SFM self-supervision, i.e., as if we would not have supervision for $X^s$. Then, we find the median depth value produced by this model on the virtual-world data, $d^v_m \in \mathbb{R}$. Finally, we set $\psi = d^v_m / d^m_m$, where $d^m_m \in \mathbb{R}$ is the median depth value of the GT. This pre-processing step is performed once and the model discarded afterwards. Other works follow a similar idea [15, 17, 19, 35, 45] to compute absolute depth but relying on LiDAR data as GT reference, while we only rely on virtual-world data.

Algorithm 1: Computing the gradients $\Delta_{\text{gen}c}$, $\Delta_{\text{osp}}$, $\Delta_{\text{gen}a}$ for a mini-batch $X^c_B \subset X^c$, $X^a_B \subset X^a \in \Psi_{\text{pre-de}}$. $\nabla_{\xi} F(\xi_1, \xi_2)$ refers to back-propagation on $F(\xi_1, \xi_2)$ with respect to weights $\xi_1 \in \xi_1 \cup \xi_2$, $\emptyset$ is the empty set.

**Forward Passes with $\{X^c_B, Y^c_B\}$**

$$F^c(\theta) \leftarrow L^c(\theta; X^c_B, Y^c_B)$$
$$L^a,s(F^c \rightarrow \omega^a L^a(F_{\text{gen}c}, \theta^a; X^c_B, Y^c_B)$$

**Back-propagation for Supervision & DA**

$$\Delta_{\text{osp}} \leftarrow \nabla_{\text{osp}} F^c(\theta)$$
$$\Delta_{\text{gen}a} \leftarrow \nabla_{\text{gen}a} L^a(F^c \rightarrow \omega^a L^a(F_{\text{gen}a}, \theta^a; X^c_B, Y^c_B)$$

**Forward Passes with $X^a_B$**

$$L^a(\theta, \theta^d) \leftarrow L^a(\theta, \theta^d; X^a_B)$$
$$L^a,s,F^a(\theta^a, \theta^d) \leftarrow \omega^a L^a(\theta^a, \theta^d; X^a_B, \emptyset)$$

**Back-propagation for Self-supervision & DA**

$$\Delta_{\text{osp}} \leftarrow \nabla_{\text{osp}} L^a(\theta, \theta^d)$$
$$\Delta_{\text{gen}a} \leftarrow \nabla_{\text{gen}a} L^a(\theta^a, \theta^d) \rightarrow F^a \rightarrow \omega^a L^a(F_{\text{gen}a}, \theta^a)$$

**Setting the final gradient vectors**

$$\Delta_{\text{osp}} \leftarrow \nabla_{\text{osp}} L^a(\theta, \theta^d)$$
$$\Delta_{\text{gen}a} \leftarrow \Delta_{\text{gen}a} + \Delta_{\text{gen}a}$$
$$\omega^s \leftarrow \frac{L^c(\theta) / \Psi(\theta; x)}{L^a(\theta)}$$
$$\Delta_{\text{osp}} \leftarrow \Delta_{\text{osp}} + \omega^s \Delta_{\text{osp}}$$
$$\Delta_{\text{gen}a} \leftarrow \Delta_{\text{gen}a} + \Delta_{\text{gen}a}$$

IV. EXPERIMENTAL RESULTS

In this section, we start by defining the datasets and evaluation metrics used in our experiments. After, we provide relevant implementation and training details of MonoDEVSNet. Finally, we present and discuss our quantitative and qualitative results, comparing them with those from previous literature as well as performing an ablative analysis focused on the main components of MonoDEVSNet.

A. Datasets and evaluation metrics

We use publicly available datasets and metrics which are *de facto* standards in MDE research. In particular, we use KITTI Raw (KR) [55] and Virtual KITTI (VK) [25] as real- and virtual-world sequences, respectively. We follow Zhou et al. [15] training-testing split. From the training split we select 12K monocular triplets, i.e., samples of the form $(x_{t-1}^c, x_t^c, x_{t+1}^c)$. The testing split consists of 697 isolated images with LiDAR-based GT, actually introduced by Eigen et al. [1]. In addition, for considering the semantic content of the images in the analysis of results, we also use KITTI Stereo 2015 (KS) [56] for testing. This dataset consists of 200 isolated images with enhanced depth maps and semantic labels. VK is used only for training, we also use 12K monocular triplets (non-rotated camera subset) with associated depth GT. In this case, the triplets are used to calibrate the global scaling factor $\psi$ (see Sect. III-H), while for actual training supervision only 12K isolated frames are used. As the depth GT of VK ranges up to $\sim 655m$, to match the range of KR’s LiDAR-based GT, we clip it to 80m ($d^m_m$). VK includes similar weather conditions as KR/KS, and adds situations with fog, overcast, and rain, as well as sunrise and sunset illumination.

Finally, as is common since [13], we use Make3D dataset [57] for assessing generalization since it is based on photographs at urban and natural areas. Therefore, Make3D shows views and content pretty much different from those on-board a vehicle as KR, KS, and VK. The images come with depth GT acquired by a 3D scanner. There are 534 images with depth GT, organized in a standard split of 400 for training and 134 for testing. We use the latter, since we rely on Make3D only for testing our proposal.

In order to assess quantitative MDE results, we use the standard metrics introduced by Eigen et al. [1], i.e., the average absolute relative error (abs-rel), the average squared relative error (sq-rel), the root mean square error (rms), and the rms log error (rms-log). For these metrics, the lower the better. In addition, the accuracy (termed as $\delta$) under a threshold $\tau \in \{1.25, 1.25^2, 1.25^3\}$ is also used as metric. In this case, the higher the better. The abs-rel error and the $\delta < \tau$ are percentage measurements, sq-rel and rms are reported in meters, and rms-log is similar (reported in meters) to rms but applied to logarithm depth values.

These metrics are applied to absolute depth values for MDE models trained with depth supervision coming from either LiDAR [1–10,22], stereo [11–14,19], real-world stereo and virtual-world depth [33,34], or stereo and LiDAR [20,21]. However, MDE models trained on pure SFM self-supervision can only estimate depth in relative terms, i.e., up to scale.
Moreover, the scale factor varies from image to image, a problem known as scale inconsistency. In this case, before computing the above metrics, it is applied a per-image correction factor computed at testing time [15–17, 19, 35, 45]. In particular, given a test image \( x \) with GT and estimated depth \( \hat{d}(x) \), respectively, the common practice consists of computing a scale \( \psi(x) \in \mathbb{R} \) as the ratio median\( (d(x))/\)median\( (\hat{d}(x)) \), and then compare \( \psi(x) d(x) \) with \( d(x) \). On the other hand, SFM self-supervision with the help of additional information can train models able to produce absolute scale in testing time. For instance, [18] uses the ego-vehicle speed and, in fact, virtual-world supervision can help too [31, 32]. The latter approach is the one followed in this paper, especially thanks to the procedure presented in Sect. III-H. Therefore, \( \Psi(\theta; x) \) will be evaluated in relative scale terms, and \( \psi \Psi(\theta; x) \) in absolute terms. Please, note that our \( \psi \in \mathbb{R} \) scaling factor is constant for all the evaluated images and computed at training time. In the following, when presenting quantitative results, we will make clear if they are in relative or absolute terms.

### B. Implementation details

We start by selecting the actual CNN layers to implement \( \Psi(\theta; x) \). Since we leverage the SFM self-supervision idea from [19], a straightforward implementation would be to use its ResNet-based architecture as it is. However, the High-Resolution Network (HRNet) architecture [58], exhibits better accuracy in visual tasks such as semantic segmentation and object detection, suggesting that it can be a better backbone than ResNet. Thus, we decided to start our experiments by comparing ResNet and HRNet backbones using the SFM self-supervision framework provided in [19]. In particular, we assess different ResNet/HRNet architectures for \( \theta^{enc} \), while using the proposal in [19] for \( \theta^{dec} \). Then, when using ResNet

We start by selecting the actual CNN layers to implement \( \Psi(\theta; x) \). Since we leverage the SFM self-supervision idea from [19], a straightforward implementation would be to use its ResNet-based architecture as it is. However, the High-Resolution Network (HRNet) architecture [58], exhibits better accuracy in visual tasks such as semantic segmentation and object detection, suggesting that it can be a better backbone than ResNet. Thus, we decided to start our experiments by comparing ResNet and HRNet backbones using the SFM self-supervision framework provided in [19]. In particular, we assess different ResNet/HRNet architectures for \( \theta^{enc} \), while using the proposal in [19] for \( \theta^{dec} \). Then, when using ResNet

| \( \theta^{enc} \) Backb. | MW | abs-rel | sq-rel | rms | rms-log | 1.25\(^{n}\) | 1.25\(^{s}\) |
|------------------------|----|--------|-------|-----|---------|----------|----------|
| ResNet-18              | 11.6 | 0.115 | 0.882 | 4.701 | 0.190 | 0.879 | 0.961 | 0.982 |
| ResNet-50              | 25.5 | 0.110 | 0.831 | 4.642 | 0.187 | 0.883 | 0.962 | 0.982 |
| ResNet-101             | 44.5 | 0.110 | 0.809 | 4.712 | 0.187 | 0.878 | 0.960 | 0.982 |
| ResNet-152             | 60.2 | 0.107 | 0.800 | 4.629 | 0.184 | 0.885 | 0.962 | 0.982 |
| HRNet-W18              | 9.5  | 0.107 | 0.846 | 4.671 | 0.184 | 0.887 | 0.962 | 0.982 |
| HRNet-W32              | 29.3 | 0.107 | 0.881 | 4.794 | 0.187 | 0.886 | 0.961 | 0.981 |
| HRNet-W48              | 65.3 | 0.105 | 0.791 | 4.590 | 0.182 | 0.888 | 0.963 | 0.982 |

Interpolation is performed from the \( \sim 1242 \times 375 \) original resolution. As optimizer, we use ADAM with learning rate set as \( lr = 10^{-4} \), and the rest of its hyper-parameters remain with default values. The weights \( \theta^{enc} \) are initialized from available ImageNet pre-training, \( \theta^{pyr} \), \( \theta^{dec} \), and \( \theta^{ins} \) are randomly initialized with Kaiming weights, while the ResNet-50 part of \( \theta^{sf} \) is also initialized with ImageNet and the rest (convolutional layers to output the pose vector) following Kaiming. The mini-batch size is of 16 images, 50%/50% from real/virtual domains. To minimize over-fitting, we apply standard data augmentation such as horizontal flip, a 50% chance of random brightness, contrast, saturation, and hue jitter with ranges of \( \pm 0.2, \pm 0.2, \pm 0.2, \) and \( \pm 0.1 \), respectively. Remaining hyper-parameters were set as \( \lambda = 0.001 \) in Eq. (2), \( \omega^{ins} = 10 \) in Alg. 1, and in Eq. (6) our mask \( \varpi \) is set to have values of \( 1.0 \) for traffic participants (vehicles, pedestrians, etc.), \( 0.5 \) for static infrastructure (buildings, road, vegetation, etc.), and \( 0.0 \) for the sky and pixels with depth over \( d^{max} \) (here \( 80 \)m).

### D. Results and discussion

#### 1) Depth evaluation: We start by assessing MDE in relative terms. Table II presents MonoDEVSNet results (Ours) and those from previous works based on SFM self-supervision. From this table we can draw several observations. Regarding DA, MonoDEVSNet (VK-v1) outperforms S\( ^{3}\)Net (VK-v1) in all metrics. The new version of VK (VK-v2) allows us to obtain even better results. MonoDEVSNet with virtual-world supervision outperforms the version with only SFM self-supervision (best result in Table I) in all metrics,
no matter the VK version we use. Overall, MonoDEVSNet outperforms most previous methods, being on pair with [45].

2) Absolute depth assessment: While assessing depth in relative terms is a reasonable option to compare methods purely based on SFM self-supervision, the most relevant evaluation is in terms of absolute depth. These are presented in Table III. The first (top) block of this table shows results based on depth supervision from LiDAR, thus, a priori they can be thought of as upper-bounds for methods based on self-supervision. The second block shows methods that only use virtual-world supervision. The second and fourth (bottom) blocks show results based on stereo and SFM self-supervision, respectively. Methods in gray use DA supported by VK. We can draw several observations from this table. MonoDEVSNet (Ours) is the best performing among those leveraging supervision from VK v1 and, consistently with the results on relative depth, by using VK v2 we improve MonoDEVSNet results. In fact, MonoDEVSNet based on VK v2 outperforms all self-supervised methods, including those using stereo rigs instead of monocular systems. We are not yet able to reach the performance of the best methods supervised with LiDAR data. However, it is clear that our proposal is able to successfully combine real-world SFM self-supervision and virtual-world supervision. Thus, we think it is worth to keep this line of research until reaching the LiDAR-based upper-bounds.

3) Ablative analysis of MonoDEVSNet: It is also worth to analyze the contribution of the main components of our proposal. In rows 1-6 of Table IV, we add one component at a time showing performance for absolute depth. The 1st row corresponds to using the real-world data with SFM self-supervision and the virtual-world images with only depth supervision, i.e., without using neither semantic supervision (ς₁), nor gradient equalization (ς₃), nor domain adaptation (ς₄), nor mixed mini-batches (50/50), nor the global scaling factor (ψ). By comparing 1st and 2nd rows (i.e., w/o ψ and w/ ψ, resp.), we can see how relevant is obtaining a good global scaling factor to output absolute depth. In fact, adding ψ to the virtual-world depth supervision shows the higher improvement among all the components of our proposal. Then, using mixed mini-batches of real- and virtual-world data improves the performance over alternating mini-batches of only either real- or virtual-world data. This can be seen by comparing 2nd and 3rd rows (i.e., w/o 50/50 and w/ 50/50, resp.). If we alternate the domains, the optimization of a mini-batch is dominated by self-supervision (real-world data), and the optimization of the next mini-batch is dominated by supervision (virtual-world data). Thus, there is not an actual joint optimization of SFM self-supervised and supervised losses, which turns to be relevant. Yet, as can be seen in 4th row, when we add the DA component (ς₄) we improve further the depth estimation results. As can bee seen in 5th row, adding the equalization (ς₃) between gradients coming from supervision and self-supervision also improves the depth estimation results. Finally, adding the virtual-world mask (ς₀) leads to the best performance in 6th row. Overall, this analysis shows how all the considered components are relevant in our proposal. We also remark that these components are needed only to train θ, but only ψ and θ are required at testing time. Additionally, we have assessed the effect of simplifying the SFM self-supervised loss that we leverage from [19], here summarized in Sect. III-D. In particular, we neither use the auto-mask (ς₀(ς₄)), nor the multi-scale depth loss, and we replaced the minimum re-projection loss by the usual average re-projection loss (i.e., we re-define pe(xₘ₊₁, x₀, xₘ₋₁) in Sect. III-D). Results are shown in the 7th row. The metrics show worse values than in 6th row (All), but still outperforming or being on pair with PackNet-SfM and the stereo self-supervised methods of Table III.

In addition to this ablative analysis, we did additional experiments changing the DA mechanism. In particular, instead of taking direct real- and virtual-world images as input to train Ψ(θ; x), a GAN-based CNN, G, processes these images...
to create a common image space in which (hopefully) it is not possible to distinguish the domain. In short, we train an overall CNN \( \Psi(\theta; G(\vartheta^G; x)) \), where \( x \) can come from either the real or the virtual domain, and \( \vartheta^G \) are the weights of \( G \). These weights are jointly trained with all the rest (\( \theta, \vartheta^d, \vartheta^s \)) to optimize depth estimation and minimize the possibility of discriminating the original domain of a sample \( x^G = G(\vartheta^G; x) \). Table IV shows results using such a GAN when removing \( \vartheta^A \) (8th row) and when keeping it (9th row). As we can see, this approach does not improve performance. Moreover, it turns out that we alternate mini-batches of pure real- or virtual-world ones; not using \( \vartheta^A \) (rows 1-2) means that we alternate mini-batches of pure real- or virtual-world samples. Row 7 corresponds to a simplification of the SiM self-supervised loss. \( \vartheta^S \) (rows 8-9) refers to a GAN-based DA approach. LB (lower bound, row 10) indicates the use of only virtual-world data. UB (upper bound, row 12) indicates the use of KITTI LiDAR-based supervision instead of virtual-world data. Rows 11 and 13 show the difference of our best model (All) with respect to LB and UB, respectively. \( \uparrow \) means that All is LB units better, while \( \downarrow \) means that it is UB units worse. All/W18 (row 14) and All/W32 (row 15) refer to using the All configuration by relying on HRNet-W18 and HRNet-W32, respectively.

| Configuration | abs-rel | sq-rel | rms | \( \text{ms-\log} \) | 1.25% | 1.25% |
|---------------|---------|--------|-----|-----------------|------|------|
| 1. \( \{X^*, Y^* \} \) | 0.368 | 2.601 | 8.025 | 0.514 | 0.080 | 0.478 | 0.883 |
| 2. +\( \psi \) | 0.140 | 0.876 | 4.915 | 0.217 | 0.280 | 0.397 | 0.380 |
| 3. +\( \psi, \psi^d \) | 0.128 | 0.880 | 4.618 | 0.198 | 0.344 | 0.597 | 0.982 |
| 4. +\( \psi^A \) | 0.110 | 0.724 | 4.450 | 0.187 | 0.873 | 0.960 | 0.983 |
| 5. +\( \psi, \psi^d \) | 0.106 | 0.716 | 4.441 | 0.188 | 0.876 | 0.962 | 0.982 |
| 6. +\( \vartheta^A (\text{All}) \) | 0.104 | 0.721 | 4.396 | 0.185 | 0.880 | 0.962 | 0.983 |
| 7. Simplified \( \vartheta^A \) | 0.105 | 0.736 | 4.471 | 0.190 | 0.875 | 0.960 | 0.981 |
| 8. All+\( \vartheta^A \); \( \vartheta^s \) | 0.119 | 0.809 | 4.654 | 0.196 | 0.857 | 0.958 | 0.982 |
| 9. All+\( \vartheta^A \) | 0.106 | 0.748 | 4.503 | 0.191 | 0.873 | 0.959 | 0.981 |
| 10. LB | 0.165 | 1.280 | 5.628 | 0.248 | 0.777 | 0.916 | 0.965 |
| 11. Tall vs. \( \text{LB} \) | 0.094 | 0.559 | 4.722 | 0.193 | 0.778 | 0.997 | 0.997 |
| 12. UB | 0.088 | 0.583 | 4.978 | 0.164 | 0.906 | 0.970 | 0.970 |
| 13. Tall vs. \( \text{UB} \) | 0.086 | 0.581 | 4.718 | 0.164 | 0.906 | 0.970 | 0.970 |
| 14. All/W18 | 0.109 | 0.773 | 4.524 | 0.190 | 0.871 | 0.960 | 0.982 |
| 15. All/W32 | 0.107 | 0.754 | 4.510 | 0.188 | 0.875 | 0.960 | 0.982 |

Finally, we have done experiments using HRNet-W18 and HRNet-W32. The results are shown in 14th and 15th rows of Table IV, respectively. Indeed, as it happens with the results on relative depth (Table I), HRNet-W48 outperforms these more lightweight versions of HRNet. However, by using HRNet-W18 and HRNet-W32 we still outperform or are on pair with the state-of-the-art self-supervised methods shown in Table III, i.e., those based on stereo self-supervision and PackNet-SfM.

4) Qualitative results: Figure 3 presents qualitatively results relying on the depth color map commonly used in the MDE literature. We show results for representative methods in Table III, namely, DORN (LiDAR supervision), SharinGAN (stereo self-supervision and virtual-world supervision), PackNet-SfM (SiM self-supervision and ego-vehicle speed supervision), and MonoDEVSNet (Ours) using VK_v1 and VK_v2 (SiM self-supervision and virtual-world supervision). We also show the corresponding LiDAR-based GT. This GT shows that for LiDAR configurations such as the one used to acquire KITTI dataset, detecting some close vehicles may be problematic since only a few LiDAR points capture their presence. Despite being trained on LiDAR supervision, DORN provides more accurate depth information in these corner cases than the raw LiDAR, which is an example of the relevance of MDE in general. However, DORN shows worse results in these corner cases than the rest (SharinGAN/PackNet-SfM/Ours), even being more accurate in terms of MDE metrics, which focus on global assessment. SharinGAN has more difficulties than PackNet-SfM and our proposal for providing sharp borders in vertical objects/infra-structure (e.g., vehicles, pedestrians, traffic signs, trees). An interesting point to highlight is also the qualitative difference that we observe on our results depending on the use of VK version. In VK_v1 data, vehicle windows appear as transparent to depth, like in many cases happens with LiDAR data, while in VK_v2 they appear as solid. This is translated to the MDE results as we can observe comparing the two bottom rows of Fig. 3. Technically, we think the qualitative results of VK_v2 make more sense since the windows are there at the given depth. However, what we would like to highlight is that we can select one option or another thanks to the use of virtual-world data.

5) Additional insights: In terms of qualitative results we think the best performing and most similar approaches are PackNet-SfM and MonoDEVSNet, both relying only on real-world monocular systems. Thus, we perform a deeper comparison of them. First, following the analysis introduced in the PackNet-SfM article [18], Fig. 4 plots the abs-rel metric of both methods as a function of depth. Results are similar up to 20m, then our proposal clearly outperforms PackNet-SfM up to 70m, where both methods start to perform similarly and in the last part of the range, up to 80m, PackNet-SfM outperforms our proposal. How these differences translate to the abs-rel global metric depends on the number of pixels falling in each distance range, which we show as an histogram in the same plot. We see how for the KR testing set most of the pixels fall in the 5–20m depth range, where both methods perform more similarly. Second, we provide further comparative insights by using KS data since it has associated per-class semantic GT, which we are going to use for evaluation purposes. Note that,
Fig. 3: Qualitative results on the (KR) Eigen et al. testing split [1]. From top to bottom: input images, their LiDAR-based depth GT (interpolated for visualization using [61]), DORN, SharinGAN, PackNet-SfM, ours VK_v1, and ours VK_v2.

Fig. 4: Abs-rel error as a function of depth, computed on (KR) Eigen et al. testing split [1]. The height of each bar indicates the percentage of pixels whose depth GT falls within the depth range covered by the bar. We indicate five values as reference. We compare PackNet-SfM and MonoDEVSNet (Ours).

Fig. 5: Qualitative results on KS data. From left to right: input images, PackNet-SfM, MonoDEVSNet (Ours).

although KS is a different data split than the one used in the experiments shown so far (KR), still is KITTI data; thus, we are not yet facing experiments about generalization. Figure 5 compares qualitative results of PackNet-SfM vs. MonoDEVSNet. We can see how PackNet-SfM misses some vehicles that our proposal does not. We believe that these vehicles may be moving at a similar speed w.r.t the ego-vehicle, which may be problematic for pure SfM-based approaches and we hypothesize that virtual-world supervision can help to avoid this problem. Figure 6 shows the corresponding abs-rel metric per-class, focusing on the most relevant classes for driving. Note how the main differences between PackNet-SfM and

Fig. 6: Per-class abs-rel results of PackNet-SfM and MonoDEVSNet (Ours) on KS data. This per-class score is the average of the abs-rel metric only considering the pixels of each class at a time. We also indicate the percentage of pixels of each class over all the pixels from the images in KS.

Fig. 7: Point cloud representation on KR Eigen test split [1] and KS data from left to right. From top to bottom: input images, MonoDEVSNet textured point cloud.
TABLE V: Absolute depth results on Make3D testing set. All the shown methods use Make3D only for testing (generalization), except (†) which fine-tunes on Make3D training set too.

| Method                  | abs-rel | sq-rel | rms   |
|-------------------------|---------|--------|-------|
| [31] T²Net (VK\_v1)     | 0.508   | 6.589  | 8.935 |
| [32] AdaDepth-S\_² (VK\_v1) | 0.452   | 3.71   | 9.559 |
| [33] GASDA (VK\_v1)     | 0.403   | 6.709  | 10.424|
| [34] SharGAN (VK\_v1)   | 0.377   | 4.900  | 8.388 |
| Ours (VK\_v1)           | 0.385   | 3.992  | 7.949 |
| Ours (VK\_v2)           | 0.377   | 3.782  | 8.011 |

MonoDEVSNet are observed on vehicles, especially on cars. Additional qualitative results are added in Fig. 7, where we can see how original images from KR and KS can be rendered as a textured point cloud. In particular, the viewpoint of these renders can change with respect to the original images thanks to the absolute depth values obtained with MonoDEVSNet.

6) Generalization: As done in the previous literature using VK to support MDE [31–34], we assess generalization on Make3D dataset. As in this literature, we follow the standard data conditioning (cropping and resizing) for models trained on KR, as well as the standard protocol introduced in [13] to compute MDE evaluation metrics (e.g. only depth below 70m is considered). Table V presents the quantitative results usually reported for Make3D, and ours. Note how, in generalization terms, our method also outperforms the rest. Moreover, Fig. 8 shows how our proposal captures the depth structure even better than the depth GT, which is build from 55 x 305 depth maps acquired by a 3D scanner.

V. CONCLUSION

For on-board perception, we have addressed monocular depth estimation by virtual-world supervision (MonoDEVS) and real-world SFM-inspired self-supervision; the former compensating for the inherent limitations of the latter. This challenging setting allows to rely on a monocular system not only at testing time, but also at training time; a cheap and scalable approach. We have designed a CNN, MonoDEVSNet, which seamlessly trains on real- and virtual-world data, exploiting semantic and depth supervision from the virtual-world data, and addressing the virtual-to-real domain gap by a relatively simple approach which does not add computational complexity in testing time. We have performed a comprehensive set of experiments assessing quantitative results in terms of relative and absolute depth, generalization, and we show the relevance of the components involved on MonoDEVSNet training. Our proposal yields state-of-the-art results within the SFM-based setting, even outperforming stereo-based self-supervised approaches. Qualitative results also confirm that MonoDEVSNet properly captures the depth structure of the images. As a result, we show the usefulness of leveraging virtual-world supervision to ultimately reach the upper-bound performance of methods based on LiDAR supervision. Therefore, our next steps will focus on analyzing the detailed differences between LiDAR-based supervision methods and MonoDEVSNet to find better ways to benefit from virtual-world supervision.

REFERENCES

[1] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in Neural Information Processing Systems (NeurIPS), 2014.
[2] F. Liu, C. Shen, G. Lin, and Y. Reid, “Learning depth from single monocular images using deep convolutional neural fields,” IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 38, no. 10, pp. 2024–2039, 2016.
[3] A. Roy and S. Todorovic, “Monocular depth estimation using neural regression forest,” in Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.
[4] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” in Int. Conf. on 3D Vision (3DV), 2016.
[5] Y. Cao, Z. Wu, and C. Shen, “Estimating depth from monocular images as classification using deep fully convolutional residual networks,” IEEE Trans. on Circuits and Systems for Video Technology, 2017.
[6] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, “Deep ordinal regression network for monocular depth estimation,” in Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2018.
[7] A. Gurram, O. Urfalioglu, I. Halfaoui, F. Bouzaraa, and A. López, “Monocular depth estimation by learning from heterogeneous datasets,” in Intelligent Vehicles Symposium (IV), 2018.
[8] L. He, G. Wang, and Z. Hu, “Learning depth from single images with deep neural network embedding focal length,” IEEE Trans. on Image Processing, vol. 27, no. 9, pp. 4676–4689, 2018.
[9] D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, and E. Ricci, “Structured attention guided convolutional neural fields for monocular depth estimation,” in Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2018.
[10] W. Yin, Y. Liu, C. Shen, and Y. Yan, “Enforcing geometric constraints of virtual normal for depth prediction,” in Int. Conf. on Computer Vision (ICCV), 2019.
[11] A. Saxena, J. Schulte, and A. Ng, “Depth estimation using monocular and stereo cues,” in Int. Joint Conf. on Artificial Intelligence, 2007.
[12] R. Garg, V. Kumar, G. Carneiro, and I. Reid, “Unsupervised cnn for single view depth estimation: Geometry to the rescue,” in European Conference on Computer Vision (ECCV), 2016.
[13] C. Godard, A. Aodha, and G. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2017.
[14] S. Pillai, R. Ambruş, and A. Gaidon, “SuperDepth: Self-supervised, super-resolved monocular depth estimation,” in Int. Conf. on Robotics and Automation (ICRA), 2019.
[15] T. Zhou, M. Brown, N. Snavely, and D. Lowe, “Unsupervised learning of depth and ego-motion from video,” in Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2017.
[16] Z. Yin and J. Shi, “GeoNet: Unsupervised learning of dense depth, optical flow and camera pose,” in Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2018.
[17] W. Zhao, S. Liu, Y. Shu, and Y.-J. Liu, “Towards better generalization: Joint depth-pose learning without PoseNet,” in Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2020.
