3D Packing for Self-Supervised Monocular Depth Estimation

Vitor Guizilini Rareș Ambruș Sudeep Pillai Allan Raventos Adrien Gaidon
Toyota Research Institute (TRI)
`first.lastname@tri.global`

Abstract

Although cameras are ubiquitous, robotic platforms typically rely on active sensors like LiDAR for direct 3D perception. In this work, we propose a novel self-supervised monocular depth estimation method combining geometry with a new deep network, PackNet, learned only from unlabeled monocular videos. Our architecture leverages novel symmetrical packing and unpacking blocks to jointly learn to compress and decompress detail-preserving representations using 3D convolutions. Although self-supervised, our method outperforms other self, semi, and fully supervised methods on the KITTI benchmark. The 3D inductive bias in PackNet enables it to scale with input resolution and number of parameters without overfitting, generalizing better on out-of-domain data such as the NuScenes dataset. Furthermore, it does not require large-scale supervised pretraining on ImageNet and can run in real-time. Finally, we release DDAD (Dense Depth for Automated Driving), a new urban driving dataset with more challenging and accurate depth evaluation, thanks to longer-range and denser ground-truth depth generated from high-density LiDARs mounted on a fleet of self-driving cars operating world-wide.[†]

1. Introduction

Accurate depth estimation is a key prerequisite in many robotics tasks, including perception, navigation, and planning. Depth from monocular camera configurations can provide useful cues for a wide array of tasks [23, 30, 34, 36], producing dense depth maps that could complement or eventually replace expensive range sensors. However, learning monocular depth via direct supervision requires ground-truth information from additional sensors and precise cross-calibration. Self-supervised methods do not suffer from these limitations, as they use geometrical constraints on image sequences as the sole source of supervision. In this work, we address the problem of jointly estimating scene structure and camera motion across RGB image sequences using a self-supervised deep network.

While recent works in self-supervised monocular depth estimation have mostly focused on engineering the loss function [5, 33, 47, 53], we show that performance critically depends on the model architecture, in line with the observations of [27] for other self-supervised tasks. Going beyond image classification models like *ResNet* [20], our **main contribution** is a new convolutional network architecture, called *PackNet*, for high-resolution self-supervised monocular depth estimation. We propose new packing and unpacking blocks that jointly leverage 3D convolutions to learn representations that maximally propagate dense appearance and geometric information while still being able to run in real time. Our **second contribution** is a novel loss that can optionally leverage the camera's velocity when available (e.g., from cars, robots, mobile phones) to solve the inherent scale ambiguity in monocular vision. Our **third contribution** is a new dataset: *Dense Depth for Automated Driving (DDAD)*. It leverages diverse logs from a fleet of well-calibrated self-driving cars equipped with cameras and high-accuracy long-range LiDARs. Compared to existing benchmarks, DDAD enables much more accurate depth evaluation at range, which is key for high resolution monocular depth estimation methods (cf. Figure 1).

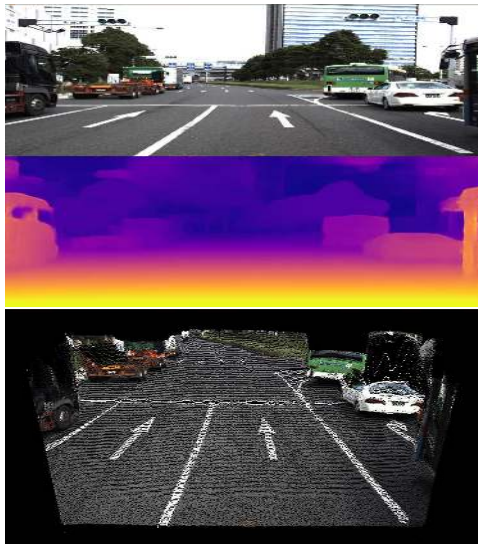


Figure 1: **Example metrically accurate PackNet prediction** (map and textured point cloud) on our DDAD dataset.

[†]Video: https://www.youtube.com/watch?v=b62iDkLgGSI
[†]Dataset: https://github.com/TRI-ML/DDAD
[†]Code: https://github.com/TRI-ML/packnet-sfm

Our experiments on the standard KITTI benchmark [16], the recent NuScenes dataset [4], and our new proposed DDAD benchmark show that our self-supervised monocular approach *i)* improves on the state of the art, especially at longer ranges; *ii)* is competitive with fully supervised methods; *iii)* generalizes better on unseen data; *iv)* scales better with number of parameters, input resolution, and more unlabeled training data; *v)* can run in real time at high resolution; and *vi)* does not require supervised pretraining on ImageNet to achieve state-of-the-art results; or test-time ground-truth scaling if velocity information is available at training time.

2. Related Work

Depth estimation from a single image poses several challenges due to its ill-posed and ambiguous nature. However, modern convolutional networks have shown that it is possible to successfully leverage appearance-based patterns in large scale datasets in order to make accurate predictions.

Depth Network Architectures Eigen et al. [13] proposed one of the earliest works in convolutional-based depth estimation using a multi-scale deep network trained on RGB-D sensor data to regress the depth directly from single images. Subsequent works extended these network architectures to perform two-view stereo disparity estimation [35] using techniques developed in the flow estimation literature [12]. Following [12, 35], Umenhofer et al. [42] applied these concepts to simultaneously train a depth and pose network to predict depth and camera ego-motion between successive unconstrained image pairs.

Independently, dense pixel-prediction networks [2, 31, 48] have made significant progress towards improving the flow of information between encoding and decoding layers. Fractional pooling [19] was introduced to amortize the rapid spatial reduction during downsampling. Lee et al. [29] generalized the pooling function to allow the learning of more complex patterns, including linear combinations and learnable pooling operations. Shi et al. [39] used sub-pixel convolutions to perform Single-Image-Super-Resolution, synthesizing and super-resolving images beyond their input resolutions, while still operating at lower resolutions. Recent works [38, 51] in self-supervised monocular depth estimation use this concept to super-resolve estimates and further improve performance. Here, we go one step further and introduce new operations relying on 3D convolutions for learning to preserve and process spatial information in the features of encoding and decoding layers.

Self-Supervised Monocular Depth and Pose As supervised techniques for depth estimation advanced rapidly, the availability of target depth labels became challenging, especially for outdoor applications. To this end, [15, 17] provided an alternative strategy involving training a monocular depth network with stereo cameras, without requiring ground-truth depth labels. By leveraging Spatial Transformer Networks [22], Godard et al [17] use stereo imagery to geometrically transform the right image plus a predicted depth of the left image into a synthesized left image. The loss between the resulting synthesized and original left images is then defined in a fully-differentiable manner, using a Structural Similarity [44] term and additional depth regularization terms, thus allowing the depth network to be self-supervised in an end-to-end fashion.

Following [17] and [42], Zhou et al. [52] generalize this to self-supervised training in the *purely* monocular setting, where a depth and pose network are simultaneously learned from unlabeled monocular videos. Several methods [5, 26, 33, 43, 46, 47, 51, 53] have advanced this line terms,of work by incorporatingthese methods, ad,ditional loss and constraints. All, however, take advantage of constraints in monocular Structure-from-Motion (SfM) training that only allow the estimation of depth and pose up to an unknown scale factor, and rely on the ground-truth LiDAR measu,rements to scale their depth estimates appropriately for evaluation purposes [52]. Instead, in this work we show that, by simply using the instantaneous velocity of the camera during training, we are able to learn a *scale-aware* depth and pose model, alleviating the impractical need to use LiDAR ground-truth depth measurements at test-time.

3. Self-Supervised Scale-Aware SfM

In self-supervised monocular SfM training (Fig. 2), we aim to learn: (i) a monocular depth model $f_D : I \to D$, that predicts the scale-ambiguous depth $\hat{D} = f_D(I(p))$ for every pixel p in the target image I; and (ii) a monocular ego-motion estimator $f_{\mathbf{x}} : (I_t, I_S) \to \mathbf{x}_{t\to S}$, that predicts the set of 6-DoF rigid transformations for all $s \in S$ given by $\mathbf{x}_{t\to s} = \left(\begin{smallmatrix}\mathbf{R} & \mathbf{t}\\ \mathbf{0} & \mathbf{1}\end{smallmatrix}\right) \in \text{SE(3)}$, between the target image I_t and the set of source images $I_s \in I_S$ considered as part of the temporal context. In practice, we use the frames I_{t-1} and I_{t+1} as source images, although using a larger context is possible. Note that in the case of monocular SfM both depth and pose are estimated up to an unknown scale factor, due to the inherent ambiguity of the photometric loss.

3.1. Self-Supervised Objective

Following the work of Zhou et al. [52], we train the depth and pose network simultaneously in a *self-supervised* manner. In this work, however, we learn to recover the inverse-depth $f_d : I \to f_D^{-1}(I)$ instead, along with the ego-motion estimator $f_{\mathbf{x}}$. Similar to [52], the overall self-supervised objective consists of an appearance matching loss term $\mathcal{L}_p$ that is imposed between the synthesized target image $\hat{I}_t$ and the target image I_t, and a depth regularization term $\mathcal{L}_s$ that ensures edge-aware smoothing in the depth estimates $\hat{D}_t$. The objective takes the following form:

$$\mathcal{L}(I_t, \hat{I}_t) = \mathcal{L}_p(I_t, I_S) \odot \mathcal{M}_p \odot \mathcal{M}_t + \lambda_1 \, \mathcal{L}_s(\hat{D}_t) \quad (1)$$

where $\mathcal{M}_t$ is a binary mask that avoids computing the photometric loss on the pixels that do not have a valid mapping, and $\odot$ denotes element-wise multiplication. Additionally, λ_1 enforces a weighted depth regularization on the objective. The overall loss in Equation 1 is averaged per-pixel, pyramid-scale and image batch during training. Fig. 2 shows a high-level overview of our training pipeline.

Appearance Matching Loss. Following [17, 52] the pixel-level similarity between the target image I_t and the synthesized target image $\hat{I}_t$ is estimated using the Structural Similarity (SSIM) [44] term combined with an L1 pixel-wise loss term, inducing an overall photometric loss given by Equation 2 below.

$$\mathcal{L}_p(I_t, \hat{I}_t) = \alpha \, \frac{1 - \text{SSIM}(I_t, \hat{I}_t)}{2} + (1 - \alpha) \, \| I_t - \hat{I}_t \| \quad (2)$$

While multi-view projective geometry provides strong cues for self-supervision, errors due to parallax in the scene have an undesirable effect incurred on the photometric loss. We mitigate these undesirable effects by calculating the minimum photometric loss per pixel for each source image in the context I_S, as shown in [18], so that:

$$\mathcal{L}_p(I_t, I_S) = \min_{I_S} \mathcal{L}_p(I_t, \hat{I}_t) \quad (3)$$

The intuition is that the same pixel will not be occluded or out-of-bounds in all context images, and that the association with minimal photometric loss should be the correct one. Furthermore, we also mask out static pixels by removing those which have a *warped* photometric loss $\mathcal{L}_p(I_t, \hat{I}_t)$ higher than their corresponding *unwarped* photometric loss $\mathcal{L}_p(I_t, I_s)$, calculated using the original source image without view synthesis. Introduced in [18], this auto-mask removes pixels whose appearance does not change between frames, which includes static scenes and dynamic objects with no relative motion, since these will have a smaller photometric loss when assuming no ego-motion.

$$\mathcal{M}_p = \min_{I_S} \mathcal{L}_p(I_t, I_s) > \min_{I_S} \mathcal{L}_p(I_t, \hat{I}_t) \quad (4)$$

Depth Smoothness Loss. In order to regularize the depth in texture-less low-image gradient regions, we incorporate an edge-aware term (Equation 5), similar to [17]. The loss is weighted for each of the pyramid-levels, and is decayed by a factor of 2 on down-sampling, starting with a weight of 1 for the 0^{th} pyramid level.

$$\mathcal{L}_s(\hat{D}_t) = |\delta_x \hat{D}_t| e^{-|\delta_x I_t|} + |\delta_y \hat{D}_t| e^{-|\delta_y I_t|} \quad (5)$$

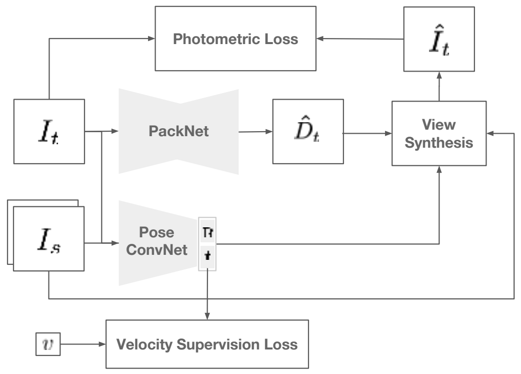


Figure 2: **PackNet-SfM**: Our proposed scale-aware self-supervised monocular structure-from-motion architecture. We introduce *PackNet* as a novel depth network, and optionally include weak velocity supervision at training time to produce *scale-aware* depth and pose models.

3.2. Scale-Aware SfM

As previously mentioned, both the monocular depth and ego-motion estimators f_d and $f_{\mathbf{x}}$ predict *scale-ambiguous* values, due to the limitations of the monocular SfM training objective. In other words, the scene depth and the camera ego-motion can only be estimated up to an unknown and ambiguous scale factor. This is also reflected in the overall learning objective, where the photometric loss is agnostic to the *metric* depth of the scene. Furthermore, we note that all previous approaches which operate in the self-supervised monocular regime [5, 15, 17, 33] suffer from this limitation, and resort to artificially incorporating this scale factor at test-time, using LiDAR measurements.

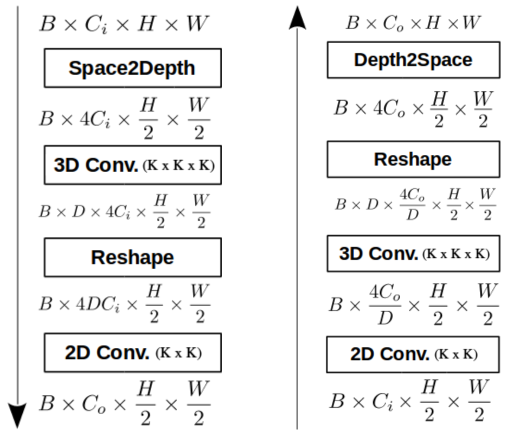
Velocity Supervision Loss. Since instantaneous velocity measurements are ubiquitous in most mobile systems today, we show that they can be directly incorporated in our self-supervised objective to learn a metrically accurate and *scale-aware* monocular depth estimator. During training, we impose an additional loss $\mathcal{L}_v$ between the magnitude of the pose-translation component of the pose network prediction $\hat{\mathbf{t}}$ and the measured instantaneous velocity scalar v multiplied by the time difference between target and source frames $\Delta T_{t \to s}$, as shown below:

$$\mathcal{L}_v(\hat{\mathbf{t}}_{t \to s}, v) = \Big| \|\hat{\mathbf{t}}_{t \to s}\| - |v| \Delta T_{t \to s} \Big| \quad (6)$$

Our final scale-aware self-supervised objective loss $\mathcal{L}_{\text{scale}}$ from Equation 1 becomes:

$$\mathcal{L}_{\text{scale}}(I_t, \hat{I}_t, v) = \mathcal{L}(I_t, \hat{I}_t) + \lambda_2 \, \mathcal{L}_v(\hat{\mathbf{t}}_{t \to s}, v) \quad (7)$$

where λ_2 is a weight used to balance the different loss terms. This additional velocity loss allows the pose network to make metrically accurate predictions, subsequently resulting in the depth network also learning metrically accurate estimates to maintain consistency (cf. Section 5.4).


$B \times C_i \times H \times W$
Space2Depth
$B \times 4C_i \times \frac{H}{2} \times \frac{W}{2}$
3D Conv. (K x K x K)
$B \times D \times 4C_i \times \frac{H}{2} \times \frac{W}{2}$
Reshape
$B \times 4DC_i \times \frac{H}{2} \times \frac{W}{2}$
2D Conv. (K x K)
$B \times C_o \times \frac{H}{2} \times \frac{W}{2}$
$B \times C_o \times H \times W$
Depth2Space
$B \times 4C_o \times \frac{H}{2} \times \frac{W}{2}$
Reshape
$B \times D \times \frac{4C_o}{D} \times \frac{H}{2} \times \frac{W}{2}$
3D Conv. (K x K x K)
$B \times \frac{4C_o}{D} \times \frac{H}{2} \times \frac{W}{2}$
2D Conv. (K x K)
$B \times C_i \times \frac{H}{2} \times \frac{W}{2}$

(a) Packing (b) Unpacking

Figure 3: **Proposed 3D packing and unpacking blocks.** *Packing* replaces striding and pooling, while *unpacking* is its symmetrical feature upsampling mechanism.

4. PackNet: 3D Packing for Depth Estimation

Standard convolutional architectures use aggressive striding and pooling to increase their receptive field size. However, this potentially decreases model performance for tasks requiring fine-grained representations [19, 49]. Similarly, traditional upsampling strategies [6, 11] fail to propagate and preserve sufficient details at the decoder layers to recover accurate depth predictions. In contrast, we propose a novel encoder-decoder architecture, called *PackNet*, that introduces new 3D *packing* and *unpacking* blocks to *learn* to *jointly* preserve and recover important *spatial* information for depth estimation. This is in alignment with recent observations that information loss is not a necessary condition to learn representations capable of generalizing to different scenarios [21]. In fact, progressive expansion and contraction in a fully invertible manner, without discarding "uninformative" input variability, has been shown to increase performance in a wide variety of tasks [3, 10, 25]. We first describe the different blocks of our proposed architecture, and then proceed to show how they are integrated together in a single model for monocular depth estimation.

4.1. Packing Block

The *packing* block (Fig. 3a) starts by folding the spatial dimensions of convolutional feature maps into extra feature channels via a `Space2Depth` operation [39]. The resulting tensor is at a reduced resolution, but in contrast to striding or pooling, this transformation is invertible and comes at no loss. Next, we *learn to compress* this concatenated feature space in order to reduce its dimensionality to a desired number of output channels. As we show in our experiments (cf. Section 5.6), 2D convolutions are not designed to directly leverage the tiled structure of this feature space. Instead, we propose to first *learn to expand* this structured representation via a 3D convolutional layer. The resulting higher dimensional feature space is then flattened (by simple reshaping) before a final 2D convolutional contraction layer. This structured feature expansion-contraction, inspired by invertible networks [3, 21] although we do not ensure invertibility, allows our architecture to dedicate more parameters to learn how to compress key spatial details that need to be preserved for high resolution depth decoding.

	Layer Description	K	Output Tensor Dim.
#0	Input RGB image		3×H×W
	Encoding Layers		
#1	Conv2d	5	64×H×W
#2	Conv2d → Packing	7	64×H/2×W/2
#3	ResidualBlock (x2) → Packing	3	64×H/4×W/4
#4	ResidualBlock (x2) → Packing	3	128×H/8×W/8
#5	ResidualBlock (x3) → Packing	3	256×H/16×W/16
#6	ResidualBlock (x3) → Packing	3	512×H/32×W/32
	Decoding Layers		
#7	Unpacking (#6) → Conv2d (⊕ #5)	3	512×H/16×W/16
#8	Unpacking (#7) → Conv2d (⊕ #4)	3	256×H/8×W/8
#9	InvDepth (#8)	3	1×H/8×W/8
#10	Unpacking (#8) → Conv2d (⊕ #3 ⊕ Upsample(#9))	3	128×H/4×W/4
#11	InvDepth (#10)	3	1×H/4×W/4
#12	Unpacking (#10) → Conv2d (⊕ #2 ⊕ Upsample(#11))	3	64×H/2×W/2
#13	InvDepth (#12)	3	1×H/2×W/2
#14	Unpacking (#12) → Conv2d (⊕ #1 ⊕ Upsample(#13))	3	64×H×W
#15	InvDepth (#14)	3	1×H×W

Table 1: **Summary of our *PackNet* architecture** for self-supervised monocular depth estimation. The *Packing* and *Unpacking* blocks are described in Fig. 3, with kernel size $K = 3$ and $D = 8$. *Conv2d* blocks include *GroupNorm* [45] with $G = 16$ and ELU non-linearities [7]. *InvDepth* blocks include a 2D convolutional layer with $K = 3$ and sigmoid non-linearities. Each *ResidualBlock* is a sequence of 3 2D convolutional layers with $K = 3/3/1$ and ELU non-linearities, followed by *GroupNorm* with $G = 16$ and *Dropout* [40] of 0.5 in the final layer. *Upsample* is a nearest-neighbor resizing operation. Numbers in parentheses indicate input layers, with ⊕ as channel concatenation. Bold numbers indicate the four inverse depth output scales.

4.2. Unpacking Block

Symmetrically, the *unpacking* block (Fig. 3b) *learns to decompress and unfold* packed convolutional feature channels back into higher resolution spatial dimensions during the decoding process. The unpacking block replaces convolutional feature upsampling, typically performed via nearest-neighbor or with learnable transposed convolutional weights. It is inspired by sub-pixel convolutions [39], but adapted to reverse the 3D packing process that the features went through in the encoder. First, we use a 2D convolutional layer to produce the required number of feature channels for a following 3D convolutional layer. Second, this 3D convolution learns to expand back the compressed spatial features. Third, these unpacked features are converted back to spatial details via a reshape and `Depth2Space` operation [39] to obtain a tensor with the desired number of output channels and target higher resolution.

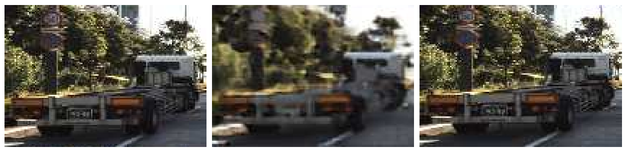

(a) Input Image (b) Max Pooling + Bilinear Upsample (c) Pack + Unpack

Figure 4: **Image reconstruction** using different encoder-decoders: (b) standard max pooling and bilinear upsampling, each followed by 2D convolutions; (c) one packing-unpacking combination (cf. Fig. 3) with $D = 2$. All kernel sizes are $K = 3$ and $C = 4$ for intermediate channels.

4.3. Detail-Preserving Properties

In Fig. 4, we illustrate the detail-preserving properties of our packing / unpacking combination, showing we can get a near-lossless encoder-decoder for single image reconstruction by minimizing the L1 loss. We train a simple network composed of one packing layer followed by a symmetrical unpacking one and show it is able to almost exactly reconstruct the input image (final loss of 0.0079), including sharp edges and finer details. In contrast, a comparable baseline replacing packing / unpacking with max pooling / bilinear upsampling (and keeping the 2D convolutions) is only able to learn a blurry reconstruction (final loss of 0.063). This highlights how *PackNet* is able to learn more complex features by preserving spatial and appearance information end-to-end throughout the network.

4.4. Model Architecture

Our *PackNet* architecture for self-supervised monocular depth estimation is detailed in Table 1. Our symmetrical encoder-decoder architecture incorporates several packing and unpacking blocks, and is supplemented with skip connections [35] to facilitate the flow of information and gradients throughout the network. The decoder produces intermediate inverse depth maps that are upsampled before being concatenated with their corresponding skip connections and unpacked feature maps. These intermediate inverse depth maps are also used at training time in the loss calculation, after being upsampled to to the full output resolution using nearest neighbors interpolation.

5. Experiments

5.1. Datasets

KITTI [16]. The KITTI benchmark is the de facto standard for depth evaluation. More specifically, we adopt the training protocol used in Eigen et al. [13], with Zhou et al.'s [52] pre-processing to remove static frames. This results in 39810 images for training, 4424 for validation and 697 for evaluation. We also consider the improved ground-truth depth maps from [41] for evaluation, which uses 5 consecutive frames to accumulate LiDAR points and stereo information to handle moving objects, resulting in 652 high-quality depth maps.

DDAD (Dense Depth for Automated Driving). As one of our contributions, we release a diverse dataset of urban, highway, and residential scenes curated from a global fleet of self-driving cars. It contains 17,050 training and 4,150 evaluation frames with ground-truth depth maps generated from dense LiDAR measurements using the Luminar-H2 sensor. This new dataset is a more realistic and challenging benchmark for depth estimation, as it is diverse and captures precise structure across images ($30k$ points per frame) at longer ranges (up to $200m$ vs $80m$ for previous datasets). See supplementary material for more details.

NuScenes [4]. To assess the generalization capability of our approach w.r.t. previous ones, we evaluate KITTI models (without fine-tuning) on the official NuScenes validation dataset of 6019 front-facing images with ground-truth depth maps generated by LiDAR reprojection.

CityScapes [8]. We also experiment with pretraining our monocular networks on the CityScapes dataset, before fine-tuning on the KITTI dataset. This also allows us to explore the scalability and generalization performance of different models, as they are trained with increasing amounts of unlabeled data. A total of 88250 images were considered as the training split for the CityScapes dataset, using the same training parameters as KITTI for 20 epochs.

5.2. Implementation Details

We use PyTorch [37] with all models trained across 8 Titan V100 GPUs. We use the Adam optimizer [24], with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The monocular depth and pose networks are trained for 100 epochs, with a batch size of 4 and initial depth and pose learning rates of $2 \cdot 10^{-4}$ and $5 \cdot 10^{-4}$ respectively. Training sequences are generated using a stride of 1, meaning that the previous $t-1$, current t, and posterior $t+1$ images are used in the loss calculation. As training proceeds, the learning rate is decayed every 40 epochs by a factor of 2. We set the SSIM weight to $\alpha = 0.85$, the depth regularization weight to $\lambda_1 = 0.001$ and, where applicable, the velocity-scaling weight to $\lambda_2 = 0.05$.

Depth Network. Unless noted otherwise, we use our *PackNet* architecture as specified in Table 1. During training, all four inverse depth output scales are used in the loss calculation, and at test-time only the final output scale is used, after being resized to the full ground-truth depth map resolution using nearest neighbor interpolation.

Pose Network. We use the architecture proposed by [52] *without* the explainability mask, which we found not to improve results. The pose network consists of 7 convolutional layers followed by a final 1×1 convolutional layer. The input to the network consists of the target view I_t and the context views I_S, and the output is the set of 6 DOF transformations between I_t and I_s, for $s \in S$.

5.3. Depth Estimation Performance

First, we report the performance of our proposed monocular depth estimation method when considering longer distances, which is now possible due to the introduction of our new DDAD dataset. Depth estimation results using this dataset for training and evaluation, considering cumulative distances up to 200m, can be found in Fig. 5 and Table 2. Additionally, in Fig. 6 we present results for different depth intervals calculated independently. From these results we can see that our *PackNet-SfM* approach significantly outperforms the state-of-the-art [18], based on the *ResNet* family, the performance gap consistently increasing when larger distances are considered.

Second, we evaluate depth predictions on KITTI using the metrics described in Eigen et al. [13]. We summarize our results in Table 3, for the original depth maps from [13] and the accumulated depth maps from [41], and illustrate their performance qualitatively in Fig. 7. In contrast to previous methods [5, 18] that predominantly focus on modifying the training objective, we show that our proposed *PackNet* architecture can by itself bolster performance and establish a new state of the art for the task of monocular depth estimation, trained in the self-supervised monocular setting.

Furthermore, we show that by simply introducing an additional source of unlabeled videos, such as the publicly available CityScapes dataset (CS+K) [8], we are able to further improve monocular depth estimation performance. As indicated by Pillai et al. [38], we also observe an improvement in performance at higher image resolutions, which we attribute to the proposed network's ability to properly preserve and process spatial information *end-to-end*. Our best results are achieved when injecting both more unlabeled data at training time and processing higher resolution input images, achieving performance comparable to semi-supervised [28] and fully supervised [14] methods.

5.4. Scale-Aware Depth Estimation Performance

Due to their inherent scale ambiguity, self-supervised monocular methods [18, 33, 52] evaluate depth by scaling their estimates to the median ground-truth as measured via LiDAR. In Section 3.2 we propose to also recover the metric scale of the scene from a single image by imposing a loss on the magnitude of the translation for the pose network output. Table 3 shows that introducing this weak velocity supervision at training time allows the generation of *scale-aware* depth models with similar performance as their unscaled counterparts, with the added benefit of not requiring ground-truth depth scaling (or even velocity information) at test-time. Another benefit of scale-awareness is that we can compose metrically accurate trajectories directly from the output of the pose network. Due to space constraints, we report pose estimation results in supplementary material.

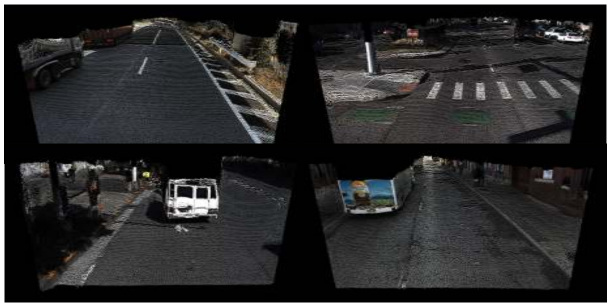


Figure 5: **PackNet pointcloud** reconstructions on DDAD.

Method	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta_{1.25}$
Monodepth2 (R18)	0.381	8.387	21.277	0.371	0.587
Monodepth2‡ (R18)	0.213	4.975	18.051	0.340	0.761
Monodepth2 (R50)	0.324	7.348	20.538	0.344	0.615
Monodepth2‡ (R50)	0.198	4.504	16.641	0.318	0.781
PackNet-SfM	**0.162**	**3.917**	**13.452**	**0.269**	**0.823**

Table 2: **Depth Evaluation on DDAD**, for 640 x 384 resolution and distances up to 200m. While the *ResNet* family heavily relies on large-scale supervised ImageNet [9] pretraining (denoted by ‡), *PackNet* achieves significantly better results despite being trained from scratch.

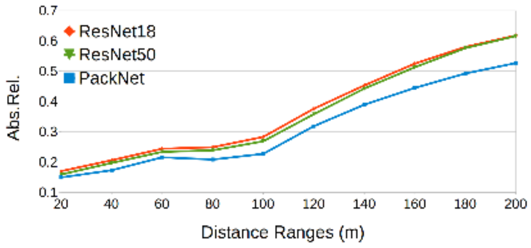


Figure 6: **Depth Evaluation on DDAD binned at different intervals**, calculated independently by only considering ground-truth depth pixels in that range (0-20m, 20-40m, ...).

5.5. Network Complexity

The introduction of packing and unpacking as alternatives to standard downsampling and upsampling operations increases the complexity of the network, due to the number of added parameters. To ensure that the gain in performance shown in our experiments is not only due to an increase in model capacity, we compare different variations of our *PackNet* architecture (obtained by modifying the number of layers and feature channels) against available *ResNet* architectures. These results are depicted in Fig. 8 and show that, while the *ResNet* family stabilizes with diminishing returns as the number of parameters increase, the *PackNet* family matches its performance at around 70M parameters and further improves as more complexity is added. Finally, the proposed architecture (Table 1) reaches around 128M parameters with an inference time of 60ms on a Titan V100 GPU, which can be further improved to < 30ms using TensorRT [1], making it suitable for real-time applications.

	Method	Supervision	Resolution	Dataset	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original [13]	SfMLearner [52]	M	416 x 128	CS + K	0.198	1.836	6.565	0.275	0.718	0.901	0.960
	Vid2Depth [33]	M	416 x 128	CS + K	0.159	1.231	5.912	0.243	0.784	0.923	0.970
	DF-Net [53]	M	576 x 160	CS + K	0.146	1.182	5.215	0.213	0.818	0.943	0.978
	Struct2Depth [5]	M	416 x 128	K	0.141	1.026	5.291	0.215	0.816	0.945	0.979
	Zhou et al.‡ [50]	M	1248 x 384	K	0.121	0.837	4.945	0.197	0.853	0.955	0.982
	Monodepth2‡ [18]	M	640 x 192	K	0.115	0.903	4.863	0.193	0.877	0.959	0.981
	Monodepth2‡ [18]	M	1024 x 320	K	0.115	0.882	4.701	0.190	0.879	0.961	0.982
	PackNet-SfM	M	640 x 192	K	0.111	0.785	4.601	0.189	0.878	0.960	0.982
	PackNet-SfM	M+v	640 x 192	K	0.111	0.829	4.788	0.199	0.864	0.954	0.980
	PackNet-SfM	M	640 x 192	CS + K	0.108	**0.727**	4.426	0.184	0.885	0.963	**0.983**
	PackNet-SfM	M+v	640 x 192	CS + K	0.108	0.803	4.642	0.195	0.875	0.958	0.980
	PackNet-SfM	M	1280 x 384	K	0.107	0.802	4.538	0.186	0.889	0.962	0.981
	PackNet-SfM	M+v	1280 x 384	K	0.107	0.803	4.566	0.197	0.876	0.957	0.979
	PackNet-SfM	M	1280 x 384	CS + K	0.104	0.758	**4.386**	**0.182**	**0.895**	**0.964**	0.982
	PackNet-SfM	M+v	1280 x 384	CS + K	**0.103**	0.796	4.404	0.189	0.881	0.959	0.980
Improved [41]	SfMLeaner [52]	M	416 x 128	CS + K	0.176	1.532	6.129	0.244	0.758	0.921	0.971
	Vid2Depth [33]	M	416 x 128	CS + K	0.134	0.983	5.501	0.203	0.827	0.944	0.981
	GeoNet [47]	M	416 x 128	CS + K	0.132	0.994	5.240	0.193	0.883	0.953	0.985
	DDVO [43]	M	416 x 128	CS + K	0.126	0.866	4.932	0.185	0.851	0.958	0.986
	EPC++ [32]	M	640 x 192	K	0.120	0.789	4.755	0.177	0.856	0.961	0.987
	Monodepth2‡ [18]	M	640 x 192	K	0.090	0.545	3.942	0.137	0.914	0.983	0.995
	Kuznietsov et al.‡ [28]	D	621 x 187	K	0.089	0.478	3.610	0.138	0.906	0.980	0.995
	DORN‡ [14]	D	513 x 385	K	0.072	**0.307**	**2.727**	0.120	0.932	0.984	0.995
	PackNet-SfM	M	640 x 192	K	0.078	0.420	3.485	0.121	0.931	0.986	0.996
	PackNet-SfM	M	1280 x 384	CS + K	**0.071**	0.359	3.153	**0.109**	**0.944**	**0.990**	**0.997**
	PackNet-SfM	M+v	1280 x 384	CS + K	0.075	0.384	3.293	0.114	0.938	0.984	0.995

Table 3: **Quantitative performance comparison of PackNet-SfM on the KITTI dataset** for distances up to 80m. For Abs Rel, Sq Rel, RMSE and RMSE$_{log}$ lower is better, and for $\delta < 1.25$, $\delta < 1.25^2$ and $\delta < 1.25^3$ higher is better. In the *Dataset* column, CS+K refers to pretraining on CityScapes (CS) and fine-tuning on KITTI (K). M refers to methods that train using monocular (M) images, and M+v refers to added velocity weak supervision (v), as shown in Section 3.2. ‡ indicates ImageNet [9] pretraining. *Original* uses raw depth maps from [13] for evaluation, and *Improved* uses annotated depth maps from [41]. At test-time, all monocular methods (M) scale estimated depths with median ground-truth LiDAR information. Velocity-scaled (M+v) and supervised (D) methods are *not* scaled in such way, since they are already metrically accurate.

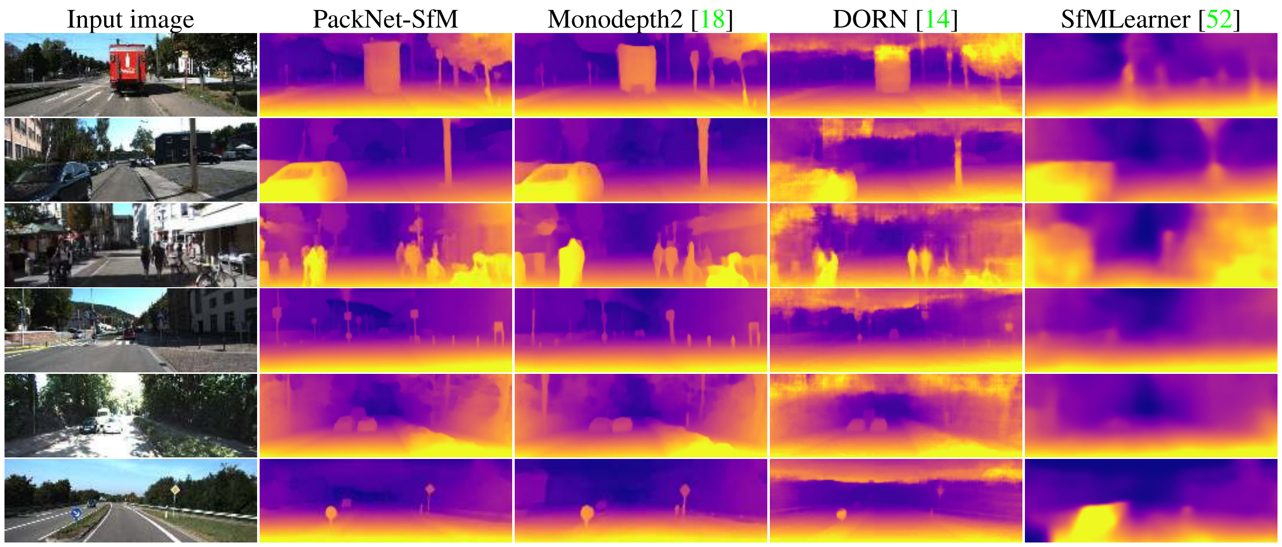


Figure 7: **Qualitative monocular depth estimation performance** comparing *PackNet* with previous methods, on frames from the KITTI dataset (Eigen test split). Our method is able to capture sharper details and structure (e.g., on vehicles, pedestrians, and thin poles) thanks to the learned preservation of spatial information.

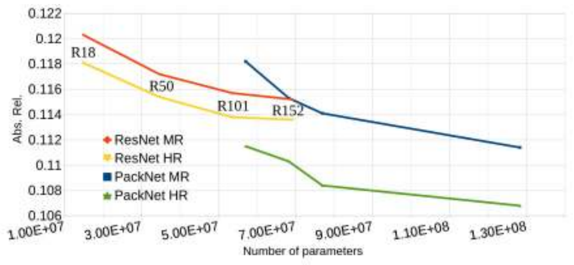


Figure 8: **Performance of different depth network architectures for varying numbers of parameters** on the original KITTI Eigen split [13] with resolutions of 640 x 192 (MR) and 1280 x 384 (HR). While the *ResNet* family plateaus at 70M parameters, the *PackNet* family matches its performance at the same number of parameters for MR, outperforms it clearly for HR, and improves significantly with more parameters in both cases without overfitting.

The *PackNet* family is also consistently better at higher resolution, as it properly preserves and propagates spatial information between layers. In contrast, as reported in prior works [18], *ResNet* architectures do not scale well, with only minor improvements at higher resolution.

5.6. Ablation Studies

To further study the performance improvements that *PackNet* provides, we perform an ablative analysis on the different architectural components introduced, as depicted in Table 4. We show that the base architecture, without the proposed packing and unpacking blocks, already produces a strong baseline for the monocular depth estimation task. The introduction of packing and unpacking boosts depth estimation performance, especially as more 3D convolutional filters are added, with new state-of-the-art results being achieved by the architecture described in Table 1.

As mentioned in [14, 18], *ResNet* architectures highly benefit from ImageNet pretraining, since they were originally developed for classification tasks. Interestingly, we also noticed that the performance of pretrained *ResNet* architectures degrades in longer training periods, due to catastrophic forgetting that leads to overfitting. The proposed *PackNet* architecture, on the other hand, achieves state-of-the-art results from randomly initialized weights, and can be further improved by self-supervised pretraining on other datasets, thus properly leveraging the large-scale availability of unlabeled information thanks to its structure.

5.7. Generalization Capability

We also investigate the generalization performance of *PackNet*, as evidence that it does not simply memorize training data but learns transferable discriminative features. To assess this, we evaluate on the recent NuScenes dataset [4] models trained on a combination of CityScapes and KITTI (CS+K), without any fine-tuning. Results in Table 5 show *PackNet* indeed generalizes better across a large spectrum of vehicles and countries (Germany for CS+K, USA + Singapore for NuScenes), outperforming standard architectures in all considered metrics without the need for large-scale supervised pretraining on ImageNet.

Depth Network	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta_{1.25}$
ResNet18	0.133	1.023	5.123	0.211	0.845
ResNet18‡	0.120	0.896	4.869	0.198	0.868
ResNet50	0.127	0.977	5.023	0.205	0.856
ResNet50‡	0.117	0.900	4.826	0.196	0.873
PackNet (w/o pack/unpack)	0.122	0.880	4.816	0.198	0.864
PackNet ($D = 0$)	0.121	0.922	4.831	0.195	0.869
PackNet ($D = 2$)	0.118	0.802	4.656	0.194	0.868
PackNet ($D = 4$)	0.113	0.818	4.621	0.190	0.875
PackNet ($D = 8$)	0.111	0.785	4.601	0.189	0.878

Table 4: **Ablation study on the PackNet architecture**, on the standard KITTI benchmark for 640 x 192 resolution. *ResNetXX* indicates that specific architecture [20] as encoder, with and without ImageNet [9] pretraining (denoted with ‡). We also show results with the proposed *PackNet* architecture, first without packing and unpacking (replaced respectively with convolutional striding and bilinear upsampling) and then with increasing numbers of 3D convolutional filters ($D = 0$ indicates no 3D convolutions and the corresponding reshape operations).

Method	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta_{1.25}$
ResNet18	0.218	2.053	8.154	0.355	0.650
ResNet18‡	0.212	1.918	7.958	0.323	0.674
ResNet50	0.216	2.165	8.477	0.371	0.637
ResNet50‡	0.210	2.017	8.111	0.328	0.697
PackNet	**0.187**	**1.852**	**7.636**	**0.289**	**0.742**

Table 5: **Generalization capability of different depth networks**, trained on both KITTI and CityScapes and evaluated on NuScenes [4], for 640 x 192 resolution and distances up to 80m. ‡ denotes ImageNet [9] pretraining.

6. Conclusion

We propose a new convolutional network architecture for *self-supervised* monocular depth estimation: *PackNet*. It leverages novel, symmetrical, detail-preserving *packing* and *unpacking* blocks that jointly learn to compress and decompress high resolution visual information for fine-grained predictions. Although purely trained on unlabeled monocular videos, our approach outperforms other existing self- and semi-supervised methods and is even competitive with fully-supervised methods while able to run in real-time. It also generalizes better to different datasets and unseen environments without the need for ImageNet pretraining, especially when considering longer depth ranges, as assessed up to 200m on our new DDAD dataset. Additionally, by leveraging during training only weak velocity information, we are able to make our model scale-aware, i.e. producing metrically accurate depth maps from a single image.

References

[1] TensorRT python library. https://developer.nvidia.com/tensorrt. Accessed: 2019-11-09. 6

[2] Aayush Bansal, Xinlei Chen, Bryan Russell, Abhinav Gupta, and Deva Ramanan. Pixelnet: Representation of the pixels, by the pixels, and for the pixels. *arXiv preprint arXiv:1702.06506*, 2017. 2

[3] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. *arXiv preprint arXiv:1811.00995*, 2018. 4

[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. *CoRR*, 2019. 2, 5, 8

[5] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *AAAI*, 2019. 1, 2, 3, 6, 7

[6] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1256–1272, 2017. 4

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2016. 4

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 5, 6

[9] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 6, 7, 8

[10] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *ICLR*, 2017. 4

[11] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, Feb. 2016. 4

[12] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 2

[13] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 2, 5, 6, 7, 8

[14] 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. 6, 7, 8

[15] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016. 2, 3

[16] 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, 5

[17] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017. 2, 3

[18] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. In *ICCV*, 2019. 3, 6, 7, 8

[19] Benjamin Graham. Fractional max-pooling. *arXiv:1412.607*, 2015. 2, 4

[20] 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. 1, 8

[21] Jrn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. In *International Conference on Learning Representations*, 2018. 4

[22] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 2

[23] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018. 1

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[25] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, 2018. 4

[26] Maria Klodt and Andrea Vedaldi. Supervising the new with the old: Learning sfm from sfm. In *European Conference on Computer Vision*, pages 713–728. Springer, 2018. 2

[27] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. *arXiv preprint arXiv:1901.09005*, 2019. 1

[28] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6647–6655, 2017. 6, 7

[29] Chen-Yu Lee, Patrick Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. 2

[30] Kuan-Hui Lee, German Ros, Jie Li, and Adrien Gaidon. Spigan: Privileged adversarial learning from simulation. In *ICLR*, 2019. 1

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Pro-*

ceedings of the IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015. 2

[32] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *arXiv preprint arXiv:1810.06125*, 2018. 7

[33] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018. 1, 2, 3, 6, 7

[34] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[35] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016. 2, 5

[36] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *22nd international conference on Machine learning*, pages 593–600. ACM, 2005. 1

[37] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 5

[38] Sudeep Pillai, Rares Ambrus, and Adrien Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. In *Robotics and Automation (ICRA), 2019 IEEE International Conference on*, 2018. 2, 6

[39] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016. 2, 4

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 4

[41] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. *3DV*, 2017. 5, 6, 7

[42] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on computer vision and pattern recognition (CVPR)*, volume 5, page 6, 2017. 2

[43] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, 2018. 2, 7

[44] 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. 2, 3

[45] Yuxin Wu and Kaiming He. Group normalization. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pages 3–19, 2018. 4

[46] Nan Yang, Rui Wang, Jörg Stückler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. *arXiv preprint arXiv:1807.02570*, 2018. 2

[47] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018. 1, 2, 7

[48] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[49] Hao Zhang and Jianwei Ma. Hartley spectral pooling for deep learning. *Computing Research Repository*, abs/1810.04028, 2018. 4

[50] Junsheng Zhou, Yuwang Wang, Naiyan Wang, and Wenjun Zeng. Unsupervised high-resolution depth learning from videos with dual networks. In *Inter. Conf. on Computer Vision*. IEEE, IEEE, 2019. 7

[51] Lipu Zhou, Jiamin Ye, Montiel Abello, Shengze Wang, and Michael Kaess. Unsupervised learning of monocular depth estimation with bundle adjustment, super-resolution and clip loss. *arXiv preprint arXiv:1812.03368*, 2018. 2

[52] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017. 2, 3, 5, 6, 7

[53] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV*, 2018. 1, 2, 7