3D Packing for Self-Supervised Monocular Depth Estimation

Vitor Guizilini  Rares 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 of work by incorporating these methods, additional 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 measurements 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 \( f_D : I \rightarrow D \), that predicts the scale-ambiguous depth \( D = f_D(I(p)) \) for every pixel \( p \) in the target image \( I \); and (ii) a monocular ego-motion estimator \( f_x : (I_t, I_S) \rightarrow x_{t \rightarrow S} \), that predicts the set of 6-DoF rigid transformations for all \( s \in S \) given by \( x_{t \rightarrow s} = (R^t_s I_s) \in \text{SE}(3) \), 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 \( f_d : I \rightarrow f_D^{-1}(I) \) instead, along with the ego-motion estimator \( f_x \). 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:

$$L(I_t, \hat{I}_t) = L_p(I_t, I_S) \odot M_p \odot M_t + \lambda_1 L_s(\hat{D}_t)$$  \hspace{1cm} (1)

where $M_t$ 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, $\lambda_1$ 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.

$$L_p(I_t, \hat{I}_t) = \alpha \frac{1-\text{SSIM}(I_t, \hat{I}_t)}{2} + (1 - \alpha) \|I_t - \hat{I}_t\|$$  \hspace{1cm} (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:

$$L_p(I_t, I_S) = \min_{I_S} L_p(I_t, \hat{I}_t)$$  \hspace{1cm} (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 $L_p(I_t, \hat{I}_t)$ higher than their corresponding unwarped photometric loss $L_p(I_t, I_s)$, 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.

$$M_p = \min_{I_S} L_p(I_t, I_s) > \min_{I_S} L_p(I_t, \hat{I}_t)$$  \hspace{1cm} (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 0th pyramid level.

$$L_s(\hat{D}_t) = |\delta_x \hat{D}_t| e^{-|\delta_x I_t|} + |\delta_y \hat{D}_t| e^{-|\delta_y I_t|}$$  \hspace{1cm} (5)

**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 $L_v$ between the magnitude of the pose-translation component of the pose network prediction $\hat{t}$ and the measured instantaneous velocity scalar $v$ multiplied by the time difference between target and source frames $\Delta T_{t\rightarrow s}$, as shown below:

$$L_v(\hat{t}_{t\rightarrow s}, v) = \|\hat{t}_{t\rightarrow s}\| - |v| \Delta T_{t\rightarrow s}$$  \hspace{1cm} (6)

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

$$L_{\text{scale}}(I_t, \hat{I}_t, v) = L(I_t, \hat{I}_t) + \lambda_2 L_v(\hat{t}_{t\rightarrow s}, v)$$  \hspace{1cm} (7)

where $\lambda_2$ 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).
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.

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.
4.3. Detail-Preserving Properties

In Fig. 4, we illustrate the detail-preserving properties of our packing / unpacking combination, showing how 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 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 $1 \times 1$ 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 $s \in S$. 

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.
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.

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 5: PackNet pointcloud reconstructions on DDAD.

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

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<sub>log</sub> lower is better, and for <br>\(\delta < 1.25\), <br>\(\delta < 1.25^2\) and <br>\(\delta < 1.25^3\) 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+<sup>v</sup> 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+<sup>v</sup>) 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, outperforming 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 datasets, being achieved by the architecture described in Table 1.

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).

<table>
<thead>
<tr>
<th>Method</th>
<th>Abs Rel</th>
<th>Sq Rel</th>
<th>RMSE</th>
<th>RMSE_{log}</th>
<th>δ₁₂₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet18</td>
<td>0.218</td>
<td>2.053</td>
<td>8.154</td>
<td>0.355</td>
<td>0.650</td>
</tr>
<tr>
<td>ResNet18†</td>
<td>0.212</td>
<td>1.918</td>
<td>7.958</td>
<td>0.323</td>
<td>0.674</td>
</tr>
<tr>
<td>ResNet50</td>
<td>0.216</td>
<td>2.165</td>
<td>8.477</td>
<td>0.371</td>
<td>0.637</td>
</tr>
<tr>
<td>ResNet50†</td>
<td>0.210</td>
<td>2.017</td>
<td>8.111</td>
<td>0.328</td>
<td>0.697</td>
</tr>
<tr>
<td>PackNet</td>
<td>0.187</td>
<td>1.852</td>
<td>7.636</td>
<td>0.289</td>
<td>0.742</td>
</tr>
</tbody>
</table>

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.

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.

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


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


