Digging Into Self-Supervised Monocular Depth Estimation

Clément Godard[1] Oisin Mac Aodha[2] Michael Firman[3] Gabriel Brostow[3,1]
[1]UCL [2]Caltech [3]Niantic
www.github.com/nianticlabs/monodepth2

Abstract

Per-pixel ground-truth depth data is challenging to acquire at scale. To overcome this limitation, self-supervised learning has emerged as a promising alternative for training models to perform monocular depth estimation. In this paper, we propose a set of improvements, which together result in both quantitatively and qualitatively improved depth maps compared to competing self-supervised methods.

Research on self-supervised monocular training usually explores increasingly complex architectures, loss functions, and image formation models, all of which have recently helped to close the gap with fully-supervised methods. We show that a surprisingly simple model, and associated design choices, lead to superior predictions. In particular, we propose (i) a minimum reprojection loss, designed to robustly handle occlusions, (ii) a full-resolution multi-scale sampling method that reduces visual artifacts, and (iii) an auto-masking loss to ignore training pixels that violate camera motion assumptions. We demonstrate the effectiveness of each component in isolation, and show high quality, state-of-the-art results on the KITTI benchmark.

1. Introduction

We seek to automatically infer a dense depth image from a single color input image. Estimating absolute, or even relative depth, seems ill-posed without a second input image to enable triangulation. Yet, humans learn from navigating and interacting in the real-world, enabling us to hypothesize plausible depth estimates for novel scenes [18].

Generating high quality depth-from-color is attractive because it could inexpensively complement LIDAR sensors used in self-driving cars, and enable new single-photo applications such as image-editing and AR-compositing. Solving for depth is also a powerful way to use large unlabeled image datasets for the pretraining of deep networks for downstream discriminative tasks [23]. However, collecting large and varied training datasets with accurate *ground truth* depth for supervised learning [55, 9] is itself a formidable challenge. As an alternative, several recent self-supervised approaches have shown that it is instead possible to train monocular depth estimation models using only synchronized *stereo pairs* [12, 15] or *monocular video* [76].

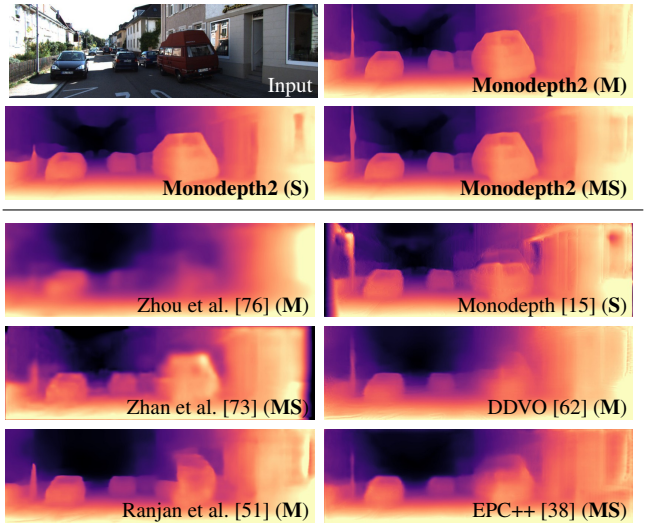

Figure 1. **Depth from a single image.** Our self-supervised model, **Monodepth2**, produces sharp, high quality depth maps, whether trained with monocular (M), stereo (S), or joint (MS) supervision.

Among the two self-supervised approaches, monocular video is an attractive alternative to stereo-based supervision, but it introduces its own set of challenges. In addition to estimating depth, the model also needs to estimate the ego-motion between temporal image pairs during training. This typically involves training a pose estimation network that takes a finite sequence of frames as input, and outputs the corresponding camera transformations. Conversely, using stereo data for training makes the camera-pose estimation a one-time offline calibration, but can cause issues related to occlusion and texture-copy artifacts [15].

We propose three architectural and loss innovations that combined, lead to large improvements in monocular depth estimation when training with monocular video, stereo pairs, or both: (1) A novel appearance matching loss to address the problem of occluded pixels that occur when using monocular supervision. (2) A novel and simple *auto-masking* approach to ignore pixels where no relative camera

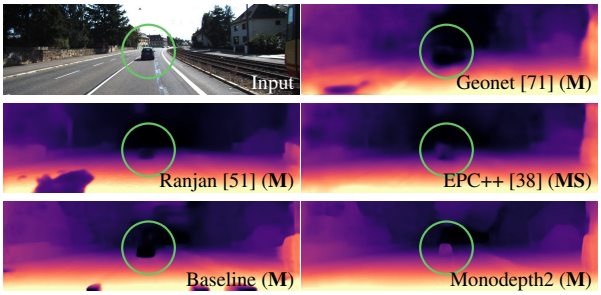

Figure 2. **Moving objects.** Monocular methods can fail to predict depth for objects that were often observed to be in motion during training *e.g*. moving cars – including methods which explicitly model motion [71, 38, 51]. Our method succeeds here where others, and our baseline with our contributions turned off, fail.

motion is observed in monocular training. (3) A multi-scale appearance matching loss that performs all image sampling at the input resolution, leading to a reduction in depth artifacts. Together, these contributions yield state-of-the-art monocular and stereo self-supervised depth estimation results on the KITTI dataset [13], and simplify many components found in the existing top performing models.

2. Related Work

We review models that, at test time, take a single color image as input and predict the depth of each pixel as output.

2.1. Supervised Depth Estimation

Estimating depth from a single image is an inherently ill-posed problem as the same input image can project to multiple plausible depths. To address this, learning based methods have shown themselves capable of fitting predictive models that exploit the relationship between color images and their corresponding depth. Various approaches, such as combining local predictions [19, 55], non-parametric scene sampling [24], through to end-to-end supervised learning [9, 31, 10] have been explored. Learning based algorithms are also among some of the best performing for stereo estimation [72, 42, 60, 25] and optical flow [20, 63].

Many of the above methods are fully supervised, requiring ground truth depth during training. However, this is challenging to acquire in varied real-world settings. As a result, there is a growing body of work that exploits weakly supervised training data, *e.g*. in the form of known object sizes [66], sparse ordinal depths [77, 6], supervised appearance matching terms [72, 73], or unpaired synthetic depth data [45, 2, 16, 78], all while still requiring the collection of additional depth or other annotations. Synthetic training data is an alternative [41], but it is not trivial to generate large amounts of synthetic data containing varied real-world appearance and motion. Recent work has shown that conventional structure-from-motion (SfM) pipelines can generate sparse training signal for both camera pose and depth [35, 28, 68], where SfM is typically run as a pre-processing step decoupled from learning. Recently, [65] built upon our model by incorporating noisy depth hints from traditional stereo algorithms, improving depth predictions.

2.2. Self-supervised Depth Estimation

In the absence of ground truth depth, one alternative is to train depth estimation models using image reconstruction as the supervisory signal. Here, the model is given a set of images as input, either in the form of stereo pairs or monocular sequences. By hallucinating the depth for a given image and projecting it into nearby views, the model is trained by minimizing the image reconstruction error.

Self-supervised Stereo Training

One form of self-supervision comes from stereo pairs. Here, synchronized stereo pairs are available during training, and by predicting the pixel disparities between the pair, a deep network can be trained to perform monocular depth estimation at test time. [67] proposed such a model with discretized depth for the problem of novel view synthesis. [12] extended this approach by predicting continuous disparity values, and [15] produced results superior to contemporary supervised methods by including a left-right depth consistency term. Stereo-based approaches have been extended with semi-supervised data [30, 39], generative adversarial networks [1, 48], additional consistency [50], temporal information [33, 73, 3], and for real-time use [49].

In this work, we show that with careful choices regarding appearance losses and image resolution, we can reach the performance of stereo training using only monocular training. Further, one of our contributions carries over to stereo training, resulting in increased performance there too.

Self-supervised Monocular Training

A less constrained form of self-supervision is to use monocular videos, where consecutive temporal frames provide the training signal. Here, in addition to predicting depth, the network has to also estimate the camera pose between frames, which is challenging in the presence of object motion. This estimated camera pose is only needed during training to help constrain the depth estimation network.

In one of the first monocular self-supervised approaches, [76] trained a depth estimation network along with a separate pose network. To deal with non-rigid scene motion, an additional motion explanation mask allowed the model to ignore specific regions that violated the rigid scene assumption. However, later iterations of their model available online disabled this term, achieving superior performance. Inspired by [4], [61] proposed a more sophisticated motion model using multiple motion masks. However, this was not fully evaluated, making it difficult to understand its utility. [71] also decomposed motion into rigid and non-rigid components, using depth and optical flow to explain object motion. This improved the *flow* estimation, but they reported no improvement when jointly training for flow and depth

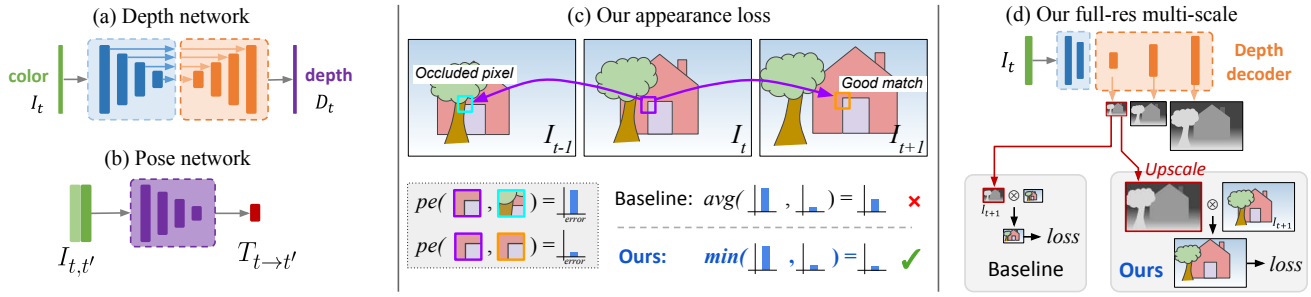

Figure 3. **Overview. (a) Depth network:** We use a standard, fully convolutional, U-Net to predict depth. **(b) Pose network:** Pose between a pair of frames is predicted with a separate pose network. **(c) Per-pixel minimum reprojection:** When correspondences are *good*, the reprojection loss should be *low*. However, occlusions and disocclusions result in pixels from the current time step not appearing in both the previous and next frames. The baseline *average* loss forces the network to match occluded pixels, whereas our *minimum reprojection* loss only matches each pixel to the view in which it is visible, leading to sharper results. **(d) Full-resolution multi-scale:** We upsample depth predictions at intermediate layers and compute all losses at the input resolution, reducing texture-copy artifacts.

estimation. In the context of optical flow estimation, [22] showed that it helps to explicitly model occlusion.

Recent approaches have begun to close the performance gap between monocular and stereo-based self-supervision. [70] constrained the predicted depth to be consistent with predicted surface normals, and [69] enforced edge consistency. [40] proposed an approximate geometry based matching loss to encourage temporal depth consistency. [62] use a depth normalization layer to overcome the preference for smaller depth values that arises from the commonly used depth smoothness term from [15]. [5] make use of pre-computed instance segmentation masks for known categories to help deal with moving objects.

Appearance Based Losses

Self-supervised training typically relies on making assumptions about the appearance (*i.e.* brightness constancy) and material properties (*e.g.* Lambertian) of object surfaces between frames. [15] showed that the inclusion of a local structure based appearance loss [64] significantly improved depth estimation performance compared to simple pairwise pixel differences [67, 12, 76]. [28] extended this approach to include an error fitting term, and [43] explored combining it with an adversarial based loss to encourage realistic looking synthesized images. Finally, inspired by [72], [73] use ground truth depth to train an appearance matching term.

3. Method

Here, we describe our depth prediction network that takes a single color input I_t and produces a depth map D_t. We first review the key ideas behind self-supervised training for monocular depth estimation, and then describe our depth estimation network and joint training loss.

3.1. Self-Supervised Training

Self-supervised depth estimation frames the learning problem as one of novel view-synthesis, by training a network to predict the appearance of a target image from the viewpoint of *another* image. By constraining the network to perform image synthesis using an intermediary variable, in our case depth or disparity, we can then extract this interpretable depth from the model. This is an ill-posed problem as there is an extremely large number of possible incorrect depths per pixel which can correctly reconstruct the novel view given the relative pose between those two views. Classical binocular and multi-view stereo methods typically address this ambiguity by enforcing smoothness in the depth maps, and by computing photo-consistency on patches when solving for per-pixel depth via global optimization *e.g.* [11].

Similar to [12, 15, 76], we also formulate our problem as the minimization of a photometric reprojection error at training time. We express the relative pose for each source view $I_{t'}$, with respect to the target image I_t's pose, as $T_{t\to t'}$. We predict a dense depth map D_t that minimizes the photometric reprojection error L_p, where

$$L_p = \sum_{t'} pe(I_t, I_{t'\to t}), \tag{1}$$

$$\text{and} \quad I_{t'\to t} = I_{t'}\left\langle proj(D_t, T_{t\to t'}, K)\right\rangle. \tag{2}$$

Here pe is a photometric reconstruction error, *e.g.* the L1 distance in pixel space; $proj()$ are the resulting 2D coordinates of the projected depths D_t in $I_{t'}$ and $\langle\rangle$ is the sampling operator. For simplicity of notation we assume the pre-computed intrinsics K of all the views are identical, though they can be different. Following [21] we use bilinear sampling to sample the source images, which is locally sub-differentiable, and we follow [75, 15] in using L1 and SSIM [64] to make our photometric error function pe, *i.e.*

$$pe(I_a, I_b) = \frac{\alpha}{2}(1 - \text{SSIM}(I_a, I_b)) + (1-\alpha)\|I_a - I_b\|_1,$$

where $\alpha = 0.85$. As in [15] we use edge-aware smoothness

$$L_s = |\partial_x d_t^*|\, e^{-|\partial_x I_t|} + |\partial_y d_t^*|\, e^{-|\partial_y I_t|}, \tag{3}$$

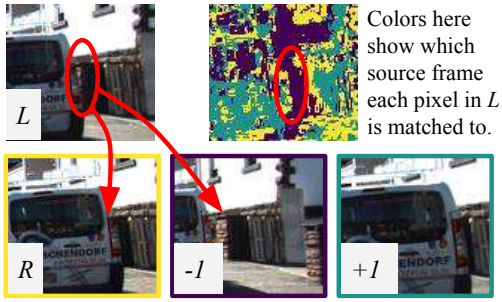

Figure 4. **Benefit of min. reprojection loss in MS training**. Pixels in the the circled region are occluded in I_R so no loss is applied between (I_L, I_R). Instead, the pixels are matched to I_{-1} where they are visible. Colors in the top right image indicate which of the source images on the bottom are selected for matching by Eqn. 4.

where $d_t^* = d_t/\overline{d_t}$ is the mean-normalized inverse depth from [62] to discourage shrinking of the estimated depth.

In stereo training, our source image $I_{t'}$ is the second view in the stereo pair to I_t, which has known relative pose. While relative poses are not known in advance for monocular sequences, [76] showed that it is possible to train a second pose estimation network to predict the relative poses $T_{t\to t'}$ used in the projection function $proj$. During training, we solve for camera pose and depth simultaneously, to minimize L_p. For monocular training, we use the two frames temporally adjacent to I_t as our source frames, *i.e.* $I_{t'} \in \{I_{t-1}, I_{t+1}\}$. In mixed training (MS), $I_{t'}$ includes the temporally adjacent frames and the opposite stereo view.

3.2. Improved Self-Supervised Depth Estimation

Existing monocular methods produce lower quality depths than the best fully-supervised models. To close this gap, we propose several improvements that significantly increase predicted depth quality, without adding additional model components that also require training (see Fig. 3).

Per-Pixel Minimum Reprojection Loss

When computing the reprojection error from multiple source images, existing self-supervised depth estimation methods average together the reprojection error into each of the available source images.This can cause issues with pixels that are visible in the target image, but are *not visible* in some of the source images (Fig. 3(c)). If the network predicts the correct depth for such a pixel, the corresponding color in an occluded source image will likely *not* match the target, inducing a high photometric error penalty. Such problematic pixels come from two main categories: out-of-view pixels due to egomotion at image boundaries, and occluded pixels. The effect of out-of-view pixels can be reduced by masking such pixels in the reprojection loss [40, 61], but this does not handle disocclusion, where average reprojection can result in blurred depth discontinuities.

We propose an improvement that deals with both issues

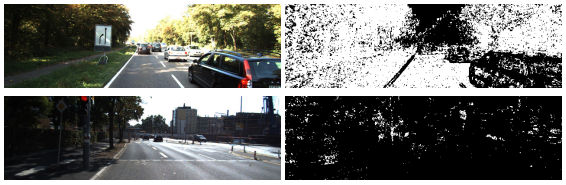

Figure 5. **Auto-masking.** We show auto-masks computed after one epoch, where black pixels are removed from the loss (*i.e.* $\mu = 0$). The mask prevents objects moving at similar speeds to the camera (top) and whole frames where the camera is static (bottom) from contaminating the loss. The mask is computed from the input frames and network predictions using Eqn. 5.

at once. At each pixel, instead of averaging the photometric error over all source images, we simply use the minimum. Our final *per-pixel* photometric loss is therefore

$$L_p = \min_{t'} pe(I_t, I_{t'\to t}). \tag{4}$$

See Fig. 4 for an example of this loss in practice. Using our minimum reprojection loss significantly reduces artifacts at image borders, improves the sharpness of occlusion boundaries, and leads to better accuracy (see Table 2).

Auto-Masking Stationary Pixels

Self-supervised monocular training often operates under the assumptions of a moving camera and a static scene. When these assumptions break down, for example when the camera is stationary or there is object motion in the scene, performance can suffer greatly. This problem can manifest itself as 'holes' of infinite depth in the predicted test time depth maps, for objects that are typically observed to be moving during training [38] (Fig. 2). This motivates our second contribution: a simple auto-masking method that filters out pixels which do not change appearance from one frame to the next in the sequence. This has the effect of letting the network ignore objects which move at the same velocity as the camera, and even to ignore whole frames in monocular videos when the camera stops moving.

Like other works [76, 61, 38], we also apply a per-pixel mask μ to the loss, selectively weighting pixels. However in contrast to prior work, our mask is binary, so $\mu \in \{0, 1\}$, and is computed automatically on the forward pass of the network, instead of being learned or estimated from object motion. We observe that pixels which remain the same between adjacent frames in the sequence often indicate a static camera, an object moving at equivalent relative translation to the camera, or a low texture region. We therefore set μ to only include the loss of pixels where the reprojection error of the warped image $I_{t'\to t}$ is lower than that of the original, unwarped source image I'_t, *i.e.*

$$\mu = \Big[\min_{t'} pe(I_t, I_{t'\to t}) < \min_{t'} pe(I_t, I_{t'}) \Big], \tag{5}$$

where $[\,]$ is the Iverson bracket. In cases where the camera and another object are both moving at a similar velocity,

μ prevents the pixels which remain stationary in the image from contaminating the loss. Similarly, when the camera is static, the mask can filter out all pixels in the image (Fig. 5). We show experimentally that this simple and inexpensive modification to the loss brings significant improvements.

Multi-scale Estimation

Due to the gradient locality of the bilinear sampler [21], and to prevent the training objective getting stuck in local minima, existing models use multi-scale depth prediction and image reconstruction. Here, the total loss is the combination of the individual losses at each scale in the decoder. [12, 15] compute the photometric error on images at the resolution of each decoder layer. We observe that this has the tendency to create 'holes' in large low-texture regions in the intermediate lower resolution depth maps, as well as texture-copy artifacts (details in the depth map incorrectly transferred from the color image). Holes in the depth can occur at low resolution in low-texture regions where the photometric error is ambiguous. This complicates the task for the depth network, now freed to predict incorrect depths.

Inspired by techniques in stereo reconstruction [56], we propose an improvement to this multi-scale formulation, where we decouple the resolutions of the disparity images and the color images used to compute the reprojection error. Instead of computing the photometric error on the ambiguous low-resolution images, we first upsample the lower resolution depth maps (from the intermediate layers) to the input image resolution, and then reproject, resample, and compute the error pe at this higher input resolution (Fig. 3 (d)). This procedure is similar to matching patches, as low-resolution disparity values will be responsible for warping an entire 'patch' of pixels in the high resolution image. This effectively constrains the depth maps at each scale to work toward the same objective *i.e*. reconstructing the high resolution input target image as accurately as possible.

Final Training Loss

We combine our per-pixel smoothness and masked photometric losses as $L = \mu L_p + \lambda L_s$, and average over each pixel, scale, and batch.

3.3. Additional Considerations

Our depth estimation network is based on the general U-Net architecture [53], *i.e*. an encoder-decoder network, with skip connections, enabling us to represent both deep abstract features as well as local information. We use a ResNet18 [17] as our encoder, which contains 11M parameters, compared to the larger, and slower, DispNet and ResNet50 models used in existing work [15]. Similar to [30, 16], we start with weights pretrained on ImageNet [54], and show that this improves accuracy for our compact model compared to training from scratch (Table 2). Our depth decoder is similar to [15], with sigmoids at the output and ELU nonlinearities [7] elsewhere. We convert the sigmoid output σ to depth with $D = 1/(a\sigma + b)$, where a and b are chosen to constrain D between 0.1 and 100 units. We make use of reflection padding, in place of zero padding, in the decoder, returning the value of the closest border pixels in the source image when samples land outside of the image boundaries. We found that this significantly reduces the border artifacts found in existing approaches, *e.g*. [15].

For pose estimation, we follow [62] and predict the rotation using an axis-angle representation, and scale the rotation and translation outputs by 0.01. For monocular training, we use a sequence length of three frames, while our pose network is formed from a ResNet18, modified to accept a pair of color images (or six channels) as input and to predict a single 6-DoF relative pose. We perform horizontal flips and the following training augmentations, with 50% chance: random brightness, contrast, saturation, and hue jitter with respective ranges of ± 0.2, ± 0.2, ± 0.2, and ± 0.1. Importantly, the color augmentations are only applied to the images which are fed to the networks, not to those used to compute L_p. All three images fed to the pose and depth networks are augmented with the same parameters.

Our models are implemented in PyTorch [46], trained for 20 epochs using Adam [26], with a batch size of 12 and an input/output resolution of 640×192 unless otherwise specified. We use a learning rate of 10^{-4} for the first 15 epochs which is then dropped to 10^{-5} for the remainder. This was chosen using a dedicated validation set of 10% of the data. The smoothness term λ is set to 0.001. Training takes 8, 12, and 15 hours on a single Titan Xp, for the stereo (S), monocular (M), and monocular plus stereo models (MS).

4. Experiments

Here, we validate that (1) our reprojection loss helps with occluded pixels compared to existing pixel-averaging, (2) our auto-masking improves results, especially when training on scenes with static cameras, and (3) our multi-scale appearance matching loss improves accuracy. We evaluate our models, named **Monodepth2**, on the KITTI 2015 stereo dataset [13], to allow comparison with previously published monocular methods.

4.1. KITTI Eigen Split

We use the data split of Eigen *et al*. [8]. Except in ablation experiments, for training which uses monocular sequences (*i.e*. monocular and monocular plus stereo) we follow Zhou *et al*.'s [76] pre-processing to remove static frames. This results in 39,810 monocular triplets for training and 4,424 for validation. We use the same intrinsics for all images, setting the principal point of the camera to the image center and the focal length to the average of all the focal lengths in KITTI. For stereo and mixed training

Method	Train	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen [9]	D	0.203	1.548	6.307	0.282	0.702	0.890	0.890
Liu [36]	D	0.201	1.584	6.471	0.273	0.680	0.898	0.967
Klodt [28]	D*M	0.166	1.490	5.998	-	0.778	0.919	0.966
AdaDepth [45]	D*	0.167	1.257	5.578	0.237	0.771	0.922	0.971
Kuznietsov [30]	DS	0.113	0.741	4.621	0.189	0.862	0.960	0.986
DVSO [68]	D*S	0.097	0.734	4.442	0.187	0.888	0.958	0.980
SVSM FT [39]	DS	0.094	0.626	4.252	0.177	0.891	0.965	0.984
Guo [16]	DS	0.096	0.641	4.095	0.168	0.892	0.967	0.986
DORN [10]	D	**0.072**	**0.307**	**2.727**	**0.120**	**0.932**	**0.984**	**0.994**
Zhou [76]†	M	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Yang [70]	M	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Mahjourian [40]	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
GeoNet [71]†	M	0.149	1.060	5.567	0.226	0.796	0.935	0.975
DDVO [62]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974
DF-Net [78]	M	0.150	1.124	5.507	0.223	0.806	0.933	0.973
LEGO [69]	M	0.162	1.352	6.276	0.252	-	-	-
Ranjan [51]	M	0.148	1.149	5.464	0.226	0.815	0.935	0.973
EPC++ [38]	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Struct2depth '(M)' [5]	M	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Monodepth2 w/o pretraining	M	0.132	1.044	5.142	0.210	0.845	0.948	0.977
Monodepth2	M	**0.115**	**0.903**	**4.863**	**0.193**	**0.877**	**0.959**	**0.981**
Monodepth2 (1024 × 320)	M	**0.115**	**0.882**	**4.701**	**0.190**	**0.879**	**0.961**	**0.982**
Garg [12]†	S	0.152	1.226	5.849	0.246	0.784	0.921	0.967
Monodepth R50 [15]†	S	0.133	1.142	5.533	0.230	0.830	0.936	0.970
StrAT [43]	S	0.128	1.019	5.403	0.227	0.827	0.935	0.971
3Net (R50) [50]	S	0.129	0.996	5.281	0.223	0.831	0.939	0.974
3Net (VGG) [50]	S	0.119	1.201	5.888	0.208	0.844	0.941	**0.978**
SuperDepth + pp [47] (1024 × 382)	S	0.112	0.875	**4.958**	**0.207**	0.852	0.947	0.977
Monodepth2 w/o pretraining	S	0.130	1.144	5.485	0.232	0.831	0.932	0.968
Monodepth2	S	**0.109**	**0.873**	4.960	0.209	**0.864**	**0.948**	0.975
Monodepth2 (1024 × 320)	S	**0.107**	**0.849**	**4.764**	**0.201**	**0.874**	**0.953**	0.977
UnDeepVO [33]	MS	0.183	1.730	6.57	0.268	-	-	-
Zhan FullNYU [73]	D*MS	0.135	1.132	5.585	0.229	0.820	0.933	0.971
EPC++ [38]	MS	0.128	0.935	5.011	0.209	0.831	0.945	**0.979**
Monodepth2 w/o pretraining	MS	0.127	1.031	5.266	0.221	0.836	0.943	0.974
Monodepth2	MS	**0.106**	**0.818**	**4.750**	**0.196**	**0.874**	**0.957**	**0.979**
Monodepth2 (1024 × 320)	MS	**0.106**	**0.806**	**4.630**	**0.193**	**0.876**	**0.958**	**0.980**

Table 1. **Quantitative results.** Comparison of our method to existing methods on KITTI 2015 [13] using the Eigen split. Best results in each category are in **bold**; second best are underlined.
All results here are presented without post-processing [15]; see supplementary Section F for improved post-processed results. While our contributions are designed for monocular training, we still gain high accuracy in the stereo-only category.
We additionally show we can get higher scores at a larger 1024 × 320 resolution, similar to [47] – see supplementary Section G. These high resolution numbers are bolded if they beat all other models, including our low-res versions.

Legend
D – Depth supervision
D* – Auxiliary depth supervision
S – Self-supervised stereo supervision
M – Self-supervised mono supervision
† – Newer results from github.
+ pp – With post-processing

(monocular plus stereo), we set the transformation between the two stereo frames to be a pure horizontal translation of fixed length. During evaluation, we cap depth to 80m per standard practice [15]. For our monocular models, we report results using the per-image median ground truth scaling introduced by [76]. See also supplementary material Section D.2 for results where we apply a single median scaling to the whole test set, instead of scaling each image independently. For results that use any stereo supervision we do not perform median scaling as scale can be inferred from the known camera baseline during training.

We compare the results of several variants of our model, trained with different types of self-supervision: monocular video only (M), stereo only (S), and both (MS). The results in Table 1 show that our monocular method outperforms all existing state-of-the-art self-supervised approaches. We also outperform recent methods ([38, 51]) that explicitly compute optical flow as well as motion masks. Qualitative results can be seen in Fig. 7 and supplementary Section E. However, as with all image reconstruction based approaches to depth estimation, our model breaks when the scene contains objects that violate the Lambertian assumptions of our appearance loss (Fig. 8).

As expected, the combination of M and S training data increases accuracy, which is especially noticeable on metrics that are sensitive to large depth errors *e.g*. RMSE. Despite our contributions being designed around monocular training, we find that the in the stereo-only case we still perform well. We achieve high accuracy despite using a lower resolution than [47]'s 1024×384, with substantially less training time (20 vs. 200 epochs) and no use of post-processing.

4.1.1 KITTI Ablation Study

To better understand how the components of our model contribute to the overall performance in monocular training, in Table 2(a) we perform an ablation study by changing various components of our model. We see that the baseline model, without any of our contributions, performs the worst. When combined together, all our components lead to a significant improvement (**Monodepth2** (full)). More experiments turning parts of our full model off in turn are shown in supplementary material Section C.

Benefits of auto-masking The full Eigen [8] KITTI split contains several sequences where the camera does not move between frames *e.g*. where the data capture car was stopped at traffic lights. These 'no camera motion' sequences can cause problems for self-supervised monocular training, and as a result, they are typically excluded at training time using expensive to compute optical flow [76]. We report monocular results trained on the full Eigen data split in Table 2(c), *i.e*. without removing frames. The baseline model trained on the full KITTI split performs worse than our full model.

		Auto-masking	Min. reproj.	Full-res multi-scale	Pretrained	Full Eigen split	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
(a)	Baseline				✓		0.140	1.610	5.512	0.223	0.852	0.946	0.973
	Baseline + min reproj.		✓		✓		0.122	1.081	5.116	0.199	0.866	0.957	0.980
	Baseline + automasking	✓					0.124	0.936	5.010	0.206	0.858	0.952	0.977
	Baseline + full-res m.s.			✓	✓		0.124	1.170	5.249	0.203	0.865	0.953	0.978
	Monodepth2 w/o min reprojection	✓		✓	✓		0.117	0.878	4.846	0.196	0.870	0.957	0.980
	Monodepth2 w/o auto-masking		✓	✓	✓		0.120	1.097	5.074	0.197	0.872	0.956	0.979
	Monodepth2 w/o full-res m.s.	✓	✓		✓		0.117	**0.866**	4.864	0.196	0.871	0.957	**0.981**
	Monodepth2 with [76]'s mask		✓	✓	✓		0.123	1.177	5.210	0.200	0.869	0.955	0.978
	Monodepth2 smaller (416 × 128)	✓	✓	✓	✓		0.128	1.087	5.171	0.204	0.855	0.953	0.978
	Monodepth2 (full)	✓	✓	✓	✓		**0.115**	0.903	**4.863**	**0.193**	**0.877**	**0.959**	**0.981**
(b)	Baseline w/o pt						0.150	1.585	5.671	0.234	0.827	0.938	0.971
	Monodepth2 w/o pt	✓	✓	✓			**0.132**	**1.044**	**5.142**	**0.210**	**0.845**	**0.948**	**0.977**
(c)	Baseline (full Eigen dataset)				✓	✓	0.146	1.876	5.666	0.230	0.848	0.945	0.972
	Monodepth2 (full Eigen dataset)	✓	✓	✓	✓	✓	**0.116**	**0.918**	**4.872**	**0.193**	**0.874**	**0.959**	**0.981**

Table 2. **Ablation.** Results for different variants of our model (**Monodepth2**) with monocular training on KITTI 2015 [13] using the Eigen split. **(a)** The baseline model, with none of our contributions, performs poorly. The addition of our minimum reprojection, auto-masking and full-res multi-scale components, significantly improves performance. **(b)** Even without ImageNet pretrained weights, our much simpler model brings large improvements above the baseline – see also Table 1. **(c)** If we train with the full Eigen dataset (instead of the subset introduced for monocular training by [76]) our improvement over the baseline increases.

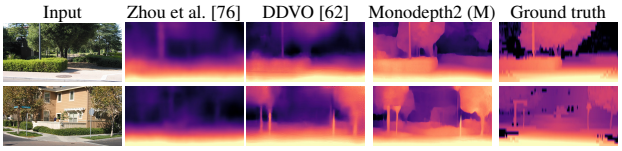

Figure 6. **Qualitative Make3D results.** All methods were trained on KITTI using monocular supervision.

Further, in Table 2(a), we replace our auto-masking loss with a re-implementation of the predictive mask from [76]. We find that this ablated model is worse than using no masking at all, while our auto-masking improves results in all cases. We see an example of how auto-masking works in practice in Fig. 5.

Effect of ImageNet pretraining We follow previous work [14, 30, 16] in initializing our encoders with weights pretrained on ImageNet [54]. While some other monocular depth prediction works have elected not to use ImageNet pretraining, we show in Table 1 that even without pretraining, we still achieve state-of-the-art results. We train these 'w/o pretraining' models for 30 epochs to ensure convergence. Table 2 shows the benefit our contributions bring both to pretrained networks and those trained from scratch; see supplementary material Section C for more ablations.

4.2. Additional Datasets

KITTI Odometry In Section A of the supplementary material we show odometry evaluation on KITTI. While our focus is better depth estimation, our pose network performs on par with competing methods. Competing methods typically feed more frames to their pose network which may improve their ability to generalize.

KITTI Depth Prediction Benchmark We also perform experiments on the recently introduced KITTI Depth Prediction Evaluation dataset [59], which features more accurate ground truth depth, addressing quality issues with the standard split. We train models using this new benchmark split, and evaluate it using the online server [27], and provide results in supplementary Section D.3. Additionally, 93% of the Eigen split test frames have higher quality ground truth depths provided by [59]. Like [1], we use these instead of the reprojected LIDAR scans to compare our method against several existing baseline algorithms, still showing superior performance.

	Type	Abs Rel	Sq Rel	RMSE	$\log_{10}$
Karsch [24]	D	0.428	5.079	8.389	0.149
Liu [37]	D	0.475	6.562	10.05	0.165
Laina [31]	D	**0.204**	**1.840**	**5.683**	**0.084**
Monodepth [15]	S	0.544	10.94	11.760	0.193
Zhou [76]	M	0.383	5.321	10.470	0.478
DDVO [62]	M	0.387	4.720	8.090	0.204
Monodepth2	M	**0.322**	**3.589**	**7.417**	**0.163**
Monodepth2	MS	0.374	3.792	8.238	0.201

Table 3. **Make3D results.** All M results benefit from median scaling, while MS uses the unmodified network prediction.

Make3D In Table 3 we report performance on the Make3D dataset [55] using our models trained on KITTI. We outperform all methods that do not use depth supervision, with the evaluation criteria from [15]. However, caution should be taken with Make3D, as its ground truth depth and input images are not well aligned, causing potential evaluation issues. We evaluate on a center crop of 2×1 ratio, and apply median scaling for our M model. Qualitative results can be seen in Fig. 6 and in supplementary Section E.

5. Conclusion

We have presented a versatile model for self-supervised monocular depth estimation, achieving state-of-the-art depth predictions. We introduced three contributions: (i) a minimum reprojection loss, computed for each pixel, to deal

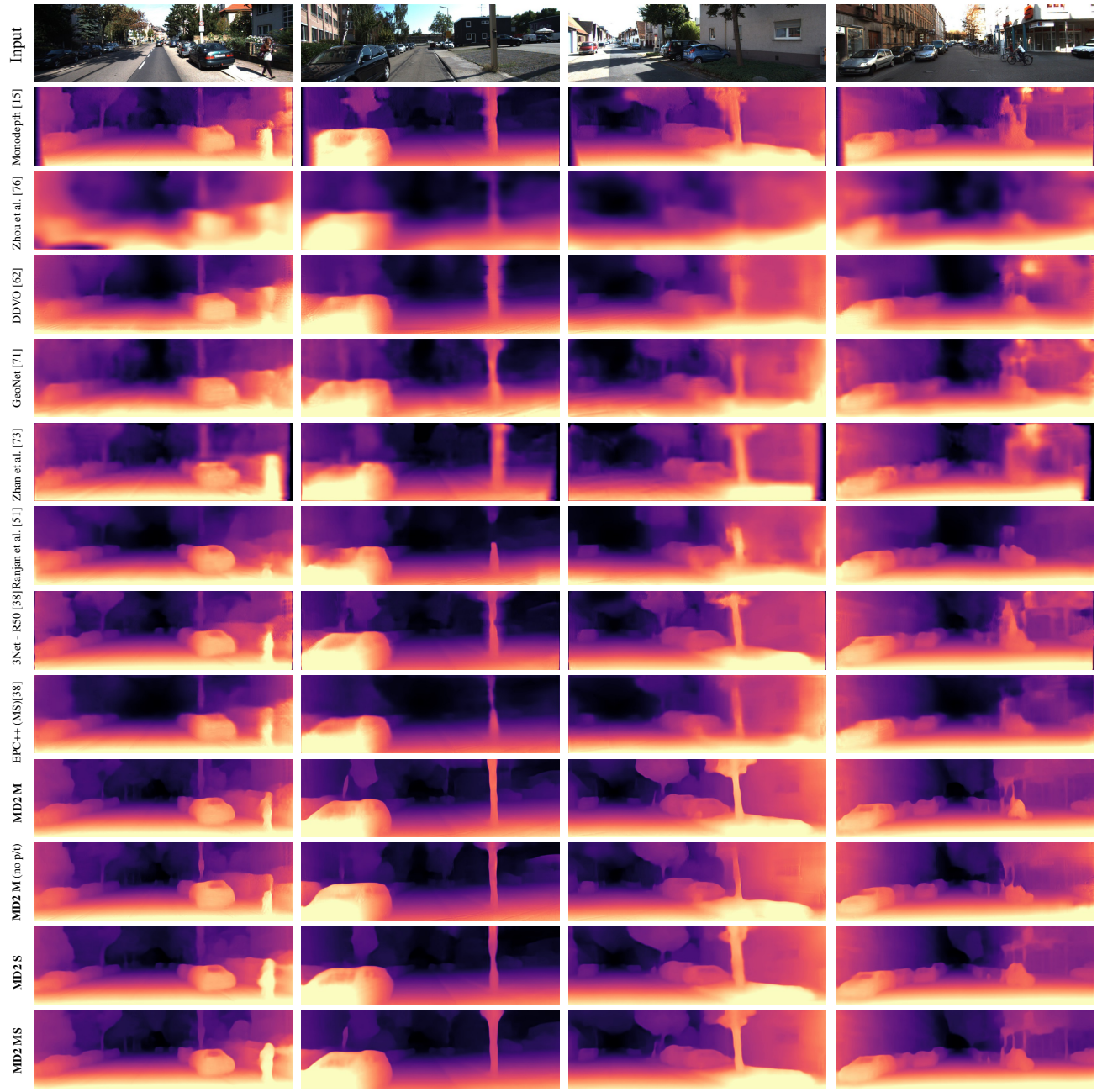

Figure 7. **Qualitative results on the KITTI Eigen split.** Our models (MD2) in the last four rows produce the sharpest depth maps, which are reflected in the superior quantitative results in Table 1. Additional results can be seen in the supplementary materiale Section E.

Figure 8. **Failure cases. Top:** Our self-supervised loss fails to learn good depths for distorted, reflective and color-saturated regions. **Bottom:** We can fail to accurately delineate objects where boundaries are ambiguous (left) or shapes are intricate (right).

with occlusions between frames in monocular video, (ii) an auto-masking loss to ignore confusing, stationary pixels, and (iii) a full-resolution multi-scale sampling method. We showed how together they give a simple and efficient model for depth estimation, which can be trained with monocular video data, stereo data, or mixed monocular and stereo data.

Acknowledgements Thanks to the authors who shared their results, and Peter Hedman, Daniyar Turmukhambetov, and Aron Monszpart for their helpful discussions.

References

[1] Filippo Aleotti, Fabio Tosi, Matteo Poggi, and Stefano Mattoccia. Generative adversarial networks for unsupervised monocular depth prediction. In *ECCV Workshops*, 2018.

[2] Amir Atapour-Abarghouei and Toby Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *CVPR*, 2018.

[3] V Madhu Babu, Kaushik Das, Anima Majumdar, and Swagat Kumar. Undemon: Unsupervised deep network for depth and ego-motion estimation. In *IROS*, 2018.

[4] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *ICRA*, 2017.

[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.

[6] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *NeurIPS*, 2016.

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv*, 2015.

[8] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[9] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014.

[10] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[11] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2015.

[12] Ravi Garg, Vijay Kumar BG, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[15] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[16] Xiaoyang Guo, Hongsheng Li, Shuai Yi, Jimmy Ren, and Xiaogang Wang. Learning monocular depth by distilling cross-domain stereo networks. In *ECCV*, 2018.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] Carol Barnes Hochberg and Julian E Hochberg. Familiar size and the perception of depth. *The Journal of Psychology*, 1952.

[19] Derek Hoiem, Alexei A Efros, and Martial Hebert. Automatic photo pop-up. *TOG*, 2005.

[20] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet2: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017.

[21] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015.

[22] Joel Janai, Fatma Güney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV*, 2018.

[23] Huaizu Jiang, Erik Learned-Miller, Gustav Larsson, Michael Maire, and Greg Shakhnarovich. Self-supervised relative depth learning for urban scene understanding. In *ECCV*, 2018.

[24] Kevin Karsch, Ce Liu, and Sing Bing Kang. Depth transfer: Depth extraction from video using non-parametric sampling. *PAMI*, 2014.

[25] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017.

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

[27] KITTI Single Depth Evaluation Server. http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_prediction. 2017.

[28] Maria Klodt and Andrea Vedaldi. Supervising the new with the old: learning SFM from SFM. In *ECCV*, 2018.

[29] Shu Kong and Charless Fowlkes. Pixel-wise attentional gating for parsimonious pixel labeling. *arXiv*, 2018.

[30] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *CVPR*, 2017.

[31] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, 2016.

[32] Bo Li, Yuchao Dai, and Mingyi He. Monocular depth estimation with hierarchical fusion of dilated cnns and soft-weighted-sum inference. *Pattern Recognition*, 2018.

[33] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. UnDeepVO: Monocular visual odometry through unsupervised deep learning. *arXiv*, 2017.

[34] Ruibo Li, Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, and Lingxiao Hang. Deep attention-based classification network for robust depth prediction. *ACCV*, 2018.

[35] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018.

[36] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *PAMI*, 2015.

[37] Miaomiao Liu, Mathieu Salzmann, and Xuming He. Discrete-continuous depth estimation from a single image. In *CVPR*, 2014.

[38] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3D holistic understanding. *arXiv*, 2018.

[39] Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. Single view stereo matching. In *CVPR*, 2018.

[40] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In *CVPR*, 2018.

[41] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *IJCV*, 2018.

[42] Nikolaus Mayer, Eddy Ilg, Philip Häusser, 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 *CVPR*, 2016.

[43] Ishit Mehta, Parikshit Sakurikar, and PJ Narayanan. Structured adversarial training for unsupervised monocular depth estimation. In *3DV*, 2018.

[44] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *Transactions on Robotics*, 2015.

[45] Jogendra Nath Kundu, Phani Krishna Uppala, Anuj Pahuja, and R. Venkatesh Babu. AdaDepth: Unsupervised content congruent adaptation for depth estimation. In *CVPR*, 2018.

[46] 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 *NeurIPS-W*, 2017.

[47] Sudeep Pillai, Rares Ambrus, and Adrien Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. In *ICRA*, 2019.

[48] Andrea Pilzer, Dan Xu, Mihai Marian Puscas, Elisa Ricci, and Nicu Sebe. Unsupervised adversarial depth estimation using cycled generative networks. In *3DV*, 2018.

[49] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on cpu. In *IROS*, 2018.

[50] Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Learning monocular depth estimation with unsupervised trinocular assumptions. In *3DV*, 2018.

[51] Anurag Ranjan, Varun Jampani, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *CVPR*, 2019.

[52] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, 2017.

[53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[55] Ashutosh Saxena, Min Sun, and Andrew Ng. Make3d: Learning 3d scene structure from a single still image. *PAMI*, 2009.

[56] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[58] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.

[59] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant CNNs. In *3DV*, 2017.

[60] 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 *CVPR*, 2017.

[61] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. SfM-Net: Learning of structure and motion from video. *arXiv*, 2017.

[62] Chaoyang Wang, Jose Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *CVPR*, 2018.

[63] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, and Wei Xu. Occlusion aware unsupervised learning of optical flow. In *CVPR*, 2018.

[64] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004.

[65] Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019.

[66] Yiran Wu, Sihao Ying, and Lianmin Zheng. Size-to-depth: A new perspective for single image depth estimation. *arXiv*, 2018.

[67] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In *ECCV*, 2016.

[68] Nan Yang, Rui Wang, Jörg Stückler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*, 2018.

[69] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. LEGO: Learning edge with geometry all at once by watching videos. In *CVPR*, 2018.

[70] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. In *AAAI*, 2018.

[71] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018.

[72] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 2016.

[73] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *CVPR*, 2018.

[74] Zhenyu Zhang, Chunyan Xu, Jian Yang, Ying Tai, and Liang Chen. Deep hierarchical guidance and regularization learning for end-to-end depth estimation. *Pattern Recognition*, 2018.

[75] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *Transactions on Computational Imaging*, 2017.

[76] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[77] Daniel Zoran, Phillip Isola, Dilip Krishnan, and William T Freeman. Learning ordinal relationships for mid-level vision. In *ICCV*, 2015.

[78] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV*, 2018.

Supplementary Material

Note on arXiv versions In an earlier pre-print of this paper, *1806.01260v1*, we included a shared encoder for pose and depth. While this reduced the number of training parameters, we have since found that we can gain even higher results with a separate ResNet pose encoder which accepts a stack of two frames as input (see ablation study in Section H). Since *v1*, we have also introduced *auto-masking* to help the model ignore pixels that violate our motion assumptions.

A. Odometry Evaluation

In Table 4 we evaluate our pose estimation network following the protocol in [76]. We trained our models on sequences 0-8 from the KITTI odometry split and tested on sequences 9 and 10. As in [76], the absolute trajectory error is then averaged over all overlapping five-frame snippets in the test sequences. Here, unlike [76] and others who use *custom* models for the odometry task, we use the *same* architecture for this task as our other results, and simply train it again from scratch on these new sequences.

Baselines such as [76] use a pose network which predicts transformations between sets of five frames simultaneously. Our pose network only takes two frames as input, and ouputs a single transformation between that pair of frames. In order to evaluate our two-frame model on the five-frame test sequences, we make separate predictions for each of the four frame-to-frame transformation for each set of five frames and combine them to form local trajectories. For completeness we repeat the same process with [76] predicted poses, which we denote as 'Zhou*'. As we can see in Table 4, our frame-to-frame poses come close to the accuracy of methods trained on blocks of five frames at a time.

B. Network Details

Except where stated, for all experiments we use a standard ResNet18 [17] encoder for both depth and pose networks. Our pose encoder is modified to accept a pair of frames, or six channels, as input. Our pose encoder therefore has convolutional weights in the first layer of shape $6\times64\times3\times3$, instead of the ResNet default of $3\times64\times3\times3$. When using pretrained weights for our pose encoder, we duplicate the first pretrained filter tensor along the channel dimension to make a filter of shape $6 \times 64 \times 3 \times 3$. This allows for a six-channel input image. All weights in this new expanded filter are divided by 2 to make the output of the convolution in the same numerical range as the original, one-image ResNet. In Table 5 we describe the parameters of each layer used in our depth decoder and pose network. Our pose network is larger and deeper than previous works [76, 62], and we only feed two frames at a time to the

	Sequence 09	**Sequence 10**	**# frames**
ORB-Slam [44]	0.014±0.008	0.012±0.011	-
DDVO [62]	0.045±0.108	0.033±0.074	3
Zhou* [76]	0.050±0.039	0.034±0.028	$5 \rightarrow 2$
Zhou [76]	0.021±0.017	0.020±0.015	5
Zhou [76]†	0.016±0.009	0.013±0.009	5
Mahjourian [40]	0.013±0.010	0.012±0.011	3
GeoNet [71]	0.012±0.007	0.012±0.009	5
EPC++ M [38]	0.013±0.007	**0.012±0.008**	3
Ranjan [51]	**0.012**±0.007	**0.012±0.008**	5
EPC++ MS [38]	**0.012±0.006**	**0.012±0.008**	3
Monodepth2 M*	0.017±0.008	0.015±0.010	2
Monodepth2 MS*	0.017±0.008	0.015±0.010	2
Monodepth2 M w/o pretraining*	0.018±0.010	0.015±0.010	2
Monodepth2 MS w/o pretraining*	0.018±0.009	0.015±0.010	2

Table 4. **Odometry results on the KITTI [13] odometry dataset.** Results show the average absolute trajectory error, and standard deviation, in meters.

† – newer results from the respective online implementations.

* – evaluation on trajectories made from pairwise predictions – see text for details.

'# frames' is the number of input frames used for pose prediction. To evaluate our method we chain integrate the poses from four pairs to make five frames for evaluation.

Depth Decoder						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
upconv5	3	1	256	32	econv5	ELU [7]
iconv5	3	1	256	16	↑upconv5, econv4	ELU
upconv4	3	1	128	16	iconv5	ELU
iconv4	3	1	128	8	↑upconv4, econv3	ELU
disp4	3	1	1	1	iconv4	Sigmoid
upconv3	3	1	64	8	iconv4	ELU
iconv3	3	1	64	4	↑upconv3, econv2	ELU
disp3	3	1	1	1	iconv3	Sigmoid
upconv2	3	1	32	4	iconv3	ELU
iconv2	3	1	32	2	↑upconv2, econv1	ELU
disp2	3	1	1	1	iconv2	Sigmoid
upconv1	3	1	16	2	iconv2	ELU
iconv1	3	1	16	1	↑upconv1	ELU
disp1	3	1	1	1	iconv1	Sigmoid

Pose Decoder						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
pconv0	1	1	256	32	econv5	ReLU
pconv1	3	1	256	32	pconv0	ReLU
pconv2	3	1	256	32	pconv1	ReLU
pconv3	1	1	6	32	pconv3	-

Table 5. **Our network architecture** Here **k** is the kernel size, **s** the stride, **chns** the number of output channels for each layer, **res** is the downscaling factor for each layer relative to the input image, and **input** corresponds to the input of each layer where ↑ is a 2× nearest-neighbor upsampling of the layer.

		Auto-masking	Min. reproj.	Full-res multi-scale	Encoder	Pretrained	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
(a)	Baseline				R18	✓	0.140	1.610	5.512	0.223	0.852	0.946	0.973
	Monodepth2 w/o min reprojection	✓		✓	R18	✓	0.117	0.878	4.846	0.196	0.870	0.957	0.980
	Monodepth2 w/o auto-masking		✓	✓	R18	✓	0.120	1.097	5.074	0.197	0.872	0.956	0.979
	Monodepth2 w/o full-res m.s.	✓	✓		R18	✓	0.117	0.866	4.864	0.196	0.871	0.957	**0.981**
	Monodepth2 w/o SSIM	✓	✓	✓	R18	✓	0.118	**0.853**	**4.824**	0.198	0.868	0.956	0.980
	Monodepth2 with [76]'s mask		✓	✓	R18	✓	0.123	1.177	5.210	0.200	0.869	0.955	0.978
	Monodepth2 (full)	✓	✓	✓	R18	✓	**0.115**	0.903	4.863	**0.193**	**0.877**	**0.959**	**0.981**
(b)	Baseline w/o pt				R18		0.150	1.585	5.671	0.234	0.827	0.938	0.971
	Monodepth2 w/o pt or auto-masking		✓	✓	R18		0.138	1.197	5.369	0.215	0.842	0.945	0.975
	Monodepth2 w/o pt or min reproj	✓		✓	R18		0.133	**1.021**	5.219	0.214	0.841	0.945	0.976
	Monodepth2 w/o pt or full-res m.s.	✓	✓		R18		**0.131**	1.030	5.206	**0.210**	**0.846**	**0.948**	**0.978**
	Monodepth2 w/o pt	✓	✓	✓	R18		0.132	1.044	**5.142**	**0.210**	0.845	**0.948**	0.977
(c)	**Monodepth2** ResNet18 w/o pt	✓	✓	✓	R18		0.132	1.044	5.142	0.210	0.845	0.948	0.977
	Monodepth2 ResNet18	✓	✓	✓	R18	✓	0.115	0.903	4.863	0.193	0.877	0.959	0.981
	Monodepth2 ResNet 50 w/o pt	✓	✓	✓	R50		0.131	1.023	5.064	0.206	0.849	0.951	0.979
	Monodepth2 ResNet 50	✓	✓	✓	R50	✓	**0.110**	**0.831**	**4.642**	**0.187**	**0.883**	**0.962**	**0.982**

Table 6. **Ablation.** Results for different variants of our model (**Monodepth2**) with monocular training (except where specified) on KITTI 2015 [13].

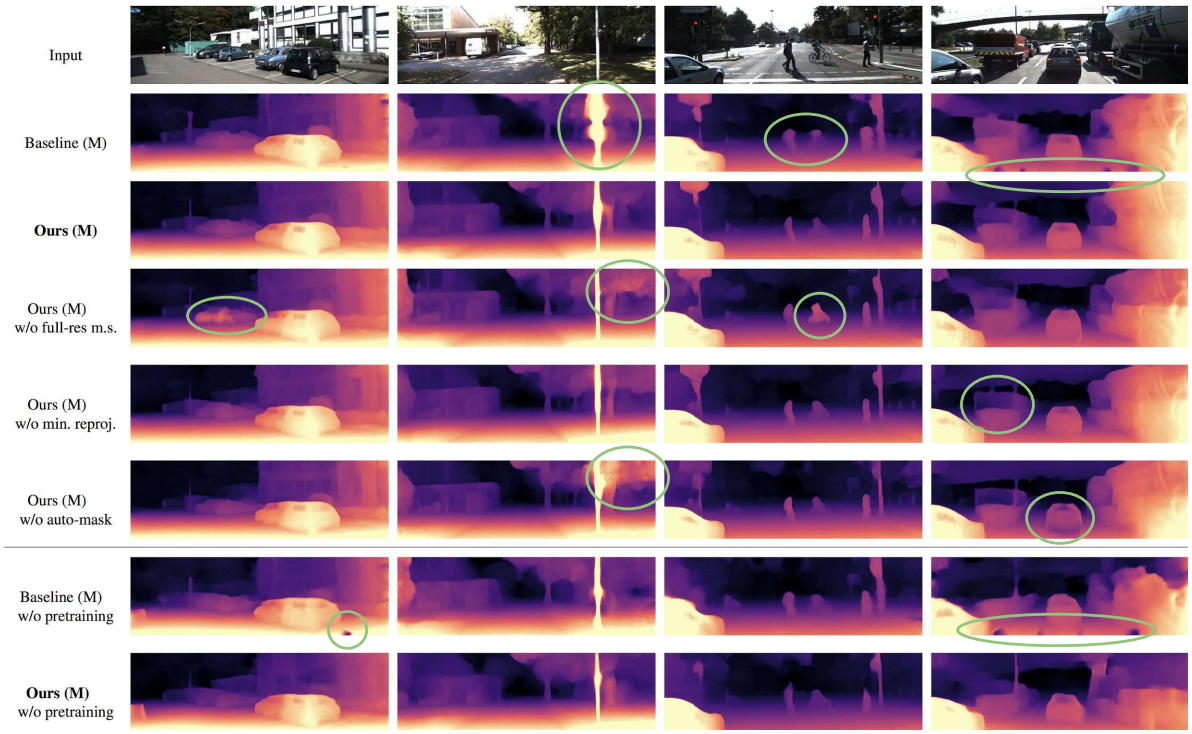

Figure 9. **Qualitative ablation study.** We can see that our model with all components added result in the smallest amount of depth artifacts. 'Baseline (M)' is our model without our full-resolution multi-scale appearance loss, minimum reprojection loss, or auto-masking loss.

pose network in contrast to previous works which use three [76, 62] or more for their depth estimation experiments. In Section H we validate the benefit of bringing additional parameters to the pose network.

C. Additional Ablation Experiments

In Table 6 we show a full ablation study on our algorithm, turning on and off different components of the system. We confirm the finding of the main paper, that all our components together gives the highest quality model, and that pretraining helps. We observe in Table 6 (d) that our results with ResNet 50 are even higher than our ResNet18 models. ResNet 50 is a standard encoder used by previous works *e.g.* [15, 50]. However, training with a 50-layer ResNet comes at the expense of longer training and test times. In Fig. 9 we show additional qualitative results for the monocular trained variants of our model from Table 6. We observe 'depth holes' in both non-pretrained and pretrained versions of the baseline model compared to ours.

Method	Train	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Zhou [76]†	M	0.176	1.532	6.129	0.244	0.758	0.921	0.971
Mahjourian [40]	M	0.134	0.983	5.501	0.203	0.827	0.944	0.981
GeoNet [71]	M	0.132	0.994	5.240	0.193	0.833	0.953	0.985
DDVO [62]	M	0.126	0.866	4.932	0.185	0.851	0.958	0.986
Ranjan [51]	M	0.123	0.881	4.834	0.181	0.860	0.959	0.985
EPC++ [38]	M	0.120	0.789	4.755	0.177	0.856	0.961	0.987
Monodepth2 w/o pretraining	M	0.112	0.715	4.502	0.167	0.876	0.967	0.990
Monodepth2	M	**0.090**	**0.545**	**3.942**	**0.137**	**0.914**	**0.983**	**0.995**
Monodepth [15]	S	0.109	0.811	4.568	0.166	0.877	0.967	0.988
3net [50] (VGG)	S	0.119	0.920	4.824	0.182	0.856	0.957	0.985
3net [50] (ResNet 50)	S	0.102	0.675	4.293	0.159	0.881	0.969	0.991
SuperDepth [47] + pp	S	0.090	0.542	3.967	0.144	0.901	0.976	**0.993**
Monodepth2 w/o pretraining	S	0.110	0.849	4.580	0.173	0.875	0.962	0.986
Monodepth2	S	**0.085**	**0.537**	**3.868**	**0.139**	**0.912**	**0.979**	**0.993**
Zhan FullNYU [73]	D*MS	0.130	1.520	5.184	0.205	0.859	0.955	0.981
EPC++ [38]	MS	0.123	0.754	4.453	0.172	0.863	0.964	0.989
Monodepth2 w/o pretraining	MS	0.107	0.720	4.345	0.161	0.890	0.971	0.989
Monodepth2	MS	**0.080**	**0.466**	**3.681**	**0.127**	**0.926**	**0.985**	**0.995**

Table 7. **KITTI improved ground truth.** Comparison to existing methods on KITTI 2015 [13] using 93% of the Eigen split and the improved ground truth from [59]. Baseline methods were evaluated using their provided disparity files, which were either available publicly or from private communication with the authors.

Legend
D* – Auxiliary depth supervision
S – Self-supervised stereo supervision
M – Self-supervised mono supervision
† – Newer results from the respective online implementations.
+ pp – With post-processing

D. Additional Evaluation

D.1. Improved Ground Truth

As mentioned in the main paper, the evaluation method introduced by Eigen [8] for KITTI uses the reprojected LIDAR points but does not handle occlusions, moving objects, or the fact that the car is moving. [59] introduced a set of high quality depth maps for the KITTI dataset, making use of 5 consecutive frames and handling moving objects using the stereo pair. This improved ground truth depth is provided for 652 (or 93%) of the 697 test frames contained in the Eigen test split [8]. We evaluate our results on these 652 improved ground truth frames and compare to existing published methods without having to retrain each method, see Table 7. We present results for all other methods for which we have obtained predictions from the authors. We use the same error metrics from the standard evaluation, and clip the predicted depths to 80 meters to match the Eigen evaluation. We evaluate on the full image and do not crop, unlike with the Eigen evaluation. We can see that our method still significantly outperforms all previously published methods on all metrics. While Superdepth [47] comes a close second to our algorithm in the S category, they are run at high resolution (1024×384 vs. our 640×192), and in Table 1 we show that at higher resolutions our model's performance also increases.

D.2. Single-Scale Evaluation

Our monocular trained approach, like all self-supervised baselines, has no guarantee of producing results with a metric scale. Nonetheless, we anticipate that there could be value in estimating depth-outputs that are, without special measures, consistent with each other across all predictions. In [76], the authors independently scale each predicted depth map by the ratio of the median of the ground truth and predicted depth map – for *each* individual test image. This is in contrast to stereo based training where the scale is known and as a result no additional scaling is required during the evaluation *e.g.* [12, 15]. This per-image depth scaling hides unstable scale estimation in both depth and pose estimation and presents a best-case scenario for the monocular training case. If a method outputs wildly varying scales for each sequence, then this evaluation protocol will hide the issue. This gives an unfair advantage over stereo trained methods that do not perform per-image scaling.

We thus modified the original protocol to instead use a single scale for all predicted depth maps of each method. For each method, we compute this single scale by taking the median of all the individual ratios of the depth medians on the *test* set. While this is still not ideal as it makes use of the ground truth depth, we believe it to be fairer and representative of the performance of each method. We also calculated the standard deviation σ_{scale} of the individual scales, where lower values indicate more consistent output depth map scales. As can be seen in Table 8, our method outperforms previously published self-supervised monocular methods, especially in the near range depth values *i.e.* $\delta < 1.25$, and is more stable overall.

D.3. KITTI Evaluation Server Benchmark

Here, we report the performance of our self-supervised monocular plus stereo model on the online KITTI single image depth prediction benchmark evaluation server [27]. [27] uses a different split of the data, which is not the same as the Eigen split. As a result, we train a new model on the provided training data. At the time of writing, there were no published self-supervised approaches among the submissions on the leaderboard. Despite not using any ground truth data during training, our monocular only predictions are competitive with fully supervised methods, see Table 9. Adding stereo data and a more powerful encoder at training time results in even better performance (**Monodepth2**

Method	σ_{scale}	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Zhou [76]†	0.210	0.258	2.338	7.040	0.309	0.601	0.853	0.940
Mahjourian [40]	0.189	0.221	1.663	6.220	0.265	0.665	0.892	0.962
GeoNet [71]	0.172	0.202	1.521	5.829	0.244	0.707	0.913	0.970
Ranjan [51]	0.162	0.188	1.298	5.467	0.232	0.724	0.927	0.974
EPC++ [38]	0.123	0.153	0.998	5.080	0.204	0.805	0.945	0.982
DDVO [62]	0.108	0.147	1.014	5.183	0.204	0.808	0.946	0.983
Monodepth2	**0.093**	**0.109**	**0.623**	**4.136**	**0.154**	**0.873**	**0.977**	**0.994**

Table 8. **Single scale monocular evaluation.** Comparison to existing monocular supervised methods on KITTI 2015 [13] using the Eigen split with improved ground truth from [59] using a *single* scale for each method. † indicates newer results from the online implementation.

Method	Train	SILog	sqErrorRel	absErrorRel	iRMSE
DORN [10]	D	11.77	2.23	8.78	12.98
DABC [34]	D	14.49	4.08	12.72	15.53
APMoE [29]	D	14.74	3.88	11.74	15.63
CSWS [32]	D	14.85	3.48	11.84	16.38
DHGRL [74]	D	15.47	4.04	12.52	15.72
Monodepth [15]	S	22.02	20.58	17.79	21.84
Monodepth2	M	15.57	4.52	12.98	16.70
Monodepth2	MS	15.07	4.16	11.64	15.27
Monodepth2 (ResNet 50)	MS	14.41	3.67	11.22	14.73

Table 9. **KITTI depth prediction benchmark.** Comparison of our monocular plus stereo approaches to fully supervised methods on the KITTI depth prediction benchmark [27]. D indicates models that were trained with ground truth depth supervision, while M and S are monocular and stereo self-supervision respectively.

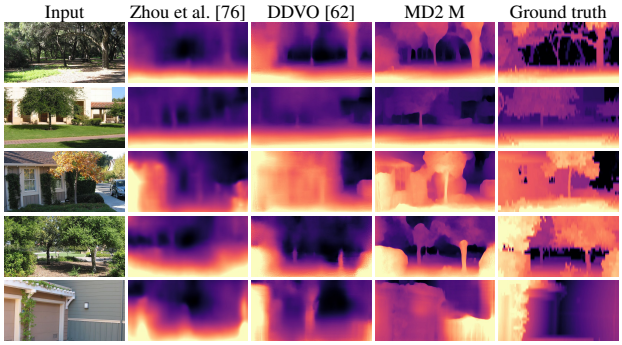

Figure 10. **Additional Make3D results.** Our model (MD2 M) trained on KITTI results in plausible depths, predicting more detail than existing monocular methods. The last row is an interesting failure for all methods as it contains an image that is very different than those from the KITTI training set.

(ResNet50)).

Because the evaluation server does not do median scaling (required for monocular methods), we needed a way to find the correct scaling for our mono-only model, which makes unscaled predictions. We make predictions with our mono-model on 1,000 images from the KITTI training set which have ground truth depths available, and for each of the 1,000 images we find the scale factor which best align the depth maps [76]. Finally, we take the median of these 1,000 scale factors as the single factor which we use to scale all predictions from our mono model. Note that, to remain true to our 'self-supervised' philosophy, we never do any other form of validation, model selection or parameter tuning using ground truth depths. For comparison, we trained a version of the original Monodepth [15] using the online code[1] on the same benchmark split.

E. Additional Qualitative Comparisons

We include additional qualitative results from the KITTI test set in Fig. 13. We can see that our models generate higher quality outputs and do not produce 'holes' in the depth maps or border artifacts that can be seen in many existing baselines *e.g.* [76, 51, 15, 73]. We also show additional results from Make3D in Fig. 10.

F. Results with Post-Processing

Post-processing, introduced by [15], is a technique to improve test time results on stereo-trained monocular depth estimation methods by running each test image through the network twice, once unflipped and then flipped. The two predictions are then masked and averaged. This has been shown to bring significant gains in accuracy for stereo results, at the expense of requiring two forward-passes through the network at test time [15, 50]. In Table 10 we show, for the first time, that post-processing also improves quantitative performance in the monocular only (M) and mixed (MS) training cases.

[1] https://github.com/mrharicot/monodepth

G. Effect of Image Resolution

In the main paper, we presented results at our standard resolution (640×192). We also showed additional results at higher (1024×320) and lower (416×128) resolutions. In Table 11 we show a full set of results at all three resolutions. We see that higher resolution helps, confirming the finding in [47]. We also include an ablation showing that, even at the highest resolution, our full-res multi-scale still provides benefit beyond just higher resolution training (*vs.* '**Ours** w/o full-res multi-scale').

Our high resolution models were initialized using the weights from our standard resolution (640×192) model after 10 epochs of training. We then trained our high resolution models for 5 epochs with a learning rate of 10^{-5}. We used a batch size of 4 to enable this higher resolution model to fit on a single 12GB Titan X GPU.

Qualitative results of the effect of resolution are illus-

Method	Train	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2 w/o pretraining	M	0.132	1.044	5.142	0.210	0.845	0.948	0.977
Monodepth2 w/o pretraining + pp	M	0.129	1.003	5.072	0.207	0.848	0.949	0.978
Monodepth2	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981
Monodepth2 + pp	M	**0.112**	0.851	4.754	0.190	0.881	0.960	0.981
Monodepth2 (1024 × 320)	M	0.115	0.882	4.701	0.190	0.879	0.961	**0.982**
Monodepth2 (1024 × 320) + pp	M	**0.112**	**0.838**	**4.607**	**0.187**	**0.883**	**0.962**	**0.982**
Monodepth2 w/o pretraining	S	0.130	1.144	5.485	0.232	0.831	0.932	0.968
Monodepth2 w/o pretraining + pp	S	0.128	1.089	5.385	0.229	0.832	0.934	0.969
Monodepth2	S	0.109	0.873	4.960	0.209	0.864	0.948	0.975
Monodepth2 + pp	S	0.108	0.842	4.891	0.207	0.866	0.949	0.976
Monodepth2 (1024 × 320)	S	0.107	0.849	4.764	0.201	0.874	0.953	**0.977**
Monodepth2 (1024 × 320) + pp	S	**0.105**	**0.822**	**4.692**	**0.199**	**0.876**	**0.954**	**0.977**
Monodepth2 w/o pretraining	MS	0.127	1.031	5.266	0.221	0.836	0.943	0.974
Monodepth2 w/o pretraining + pp	MS	0.125	1.000	5.205	0.218	0.837	0.944	0.974
Monodepth2	MS	0.106	0.818	4.750	0.196	0.874	0.957	0.979
Monodepth2 + pp	MS	**0.104**	0.786	4.687	0.194	0.876	0.958	0.980
Monodepth2 (1024 × 320)	MS	0.106	0.806	4.630	0.193	0.876	0.958	0.980
Monodepth2 (1024 × 320) + pp	MS	**0.104**	**0.775**	**4.562**	**0.191**	**0.878**	**0.959**	**0.981**

Table 10. **Effect of post-processing.** We observe that post-processing, originally motivated only for stereo training, also brings consistent benefits to all our monocular-trained models. Interestingly, for some metrics post-processing results in a larger quantitative gain than models trained at higher resolution.

	Train	Resolution	Full-res multi-scale	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Train. time (h)
Monodepth2	M	416 × 128	✓	0.128	1.087	5.171	0.204	0.855	0.953	0.978	9
Monodepth2	M	640 × 192	✓	**0.115**	0.903	4.863	0.193	0.877	0.959	0.981	12
Monodepth2	M	1024 × 320	✓	**0.115**	**0.882**	**4.701**	**0.190**	**0.879**	**0.961**	**0.982**	6 + 9 †
Monodepth2	S	416 × 128	✓	0.118	0.971	5.231	0.218	0.848	0.943	0.973	6
Monodepth2	S	640 × 192	✓	0.109	0.873	4.960	0.209	0.864	0.948	0.975	8
Monodepth2	S	1024 × 320	✓	**0.105**	**0.822**	**4.692**	**0.199**	**0.876**	**0.954**	**0.977**	4 + 8 †
Monodepth2	MS	416 × 128	✓	0.118	0.935	5.119	0.210	0.852	0.949	0.976	11
Monodepth2	MS	640 × 192	✓	**0.106**	0.818	4.750	0.196	**0.874**	0.957	0.979	15
Monodepth2	MS	1024 × 320	✓	**0.106**	**0.806**	**4.630**	**0.193**	**0.876**	**0.958**	**0.980**	7.5 + 10 †

Table 11. **Ablation study on the input/output resolutions of our model**. †Timings for the highest resolution models comprise 10 epochs training of the 640 × 192 model and 5 epochs of the 1024 × 320 model.

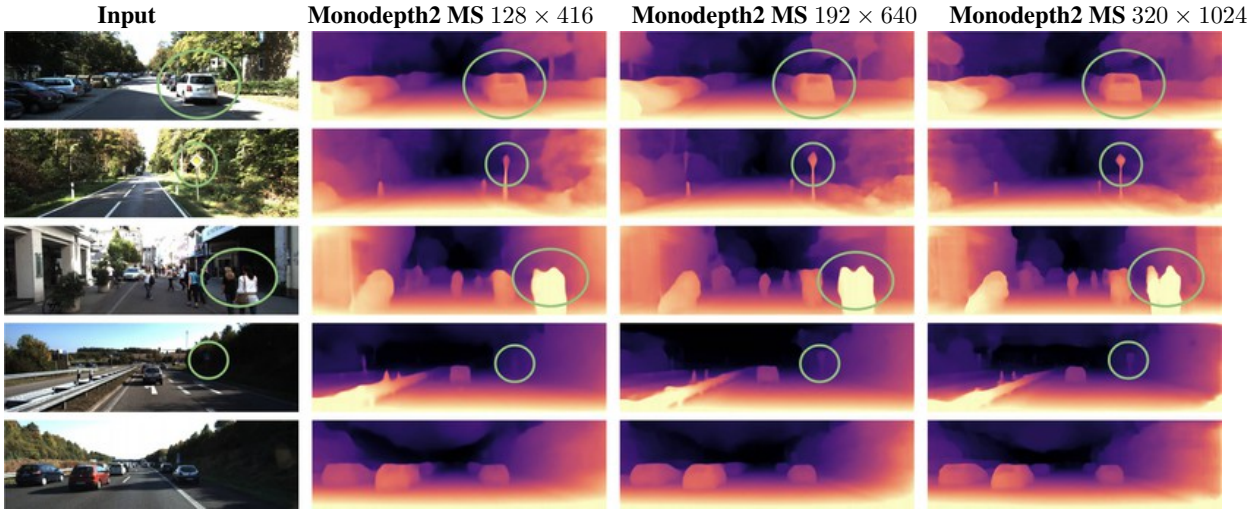

Figure 11. **Effect of varying resolutions on the KITTI Eigen split.** All predicted disparity maps have been resized to the same size for visualization. Our lowest resolution model (128 × 416) captures the broad shape of the scene successfully, but struggles with thin objects and sometimes fails to accurately capture the shape of depth discontinuities around object boundaries.

trated in Fig. 11. It is clear that all resolutions accurately capture the overall shape of the scene. However, only the highest resolution model accurately represents the shape of thin objects.

	Pose network architecture	Input frames	Pretrained	Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	δ <1.25^2	δ < 1.25^3
	PoseCNN [62]	2	✓	0.138	1.122	5.308	0.209	0.840	0.950	0.978
	PoseCNN [62]	3	✓	0.148	1.211	5.595	0.219	0.815	0.942	0.976
	Shared encoder (*arXiv v1*)	2	✓	0.125	0.986	5.070	0.201	0.857	0.954	0.979
	Shared encoder (*arXiv v1*)	3	✓	0.123	1.031	5.052	0.199	0.863	0.954	0.979
Monodepth2 ⇒	Separate ResNet	2	✓	**0.115**	0.919	4.863	**0.193**	**0.877**	0.959	**0.981**
	Separate ResNet	3	✓	**0.115**	**0.902**	**4.847**	**0.193**	**0.877**	**0.960**	**0.981**
	PoseCNN [62]	2		0.147	1.164	5.445	0.221	0.818	0.940	0.974
	PoseCNN [62]	3		0.147	1.117	5.403	0.222	0.815	0.940	0.976
	Shared encoder (*arXiv v1*)	2		0.149	1.153	5.567	0.229	0.807	0.934	0.972
	Shared encoder (*arXiv v1*)	3		0.145	1.159	5.482	0.224	0.818	0.937	0.973
Monodepth2 ⇒	Separate ResNet	2		**0.132**	1.044	**5.142**	**0.210**	**0.845**	**0.948**	**0.977**
	Separate ResNet	3		**0.132**	**1.017**	5.169	0.211	0.842	0.947	**0.977**

Table 12. **Ablation of the effect of pose networks on depth prediction.** Results shown are on depth prediction on the KITTI dataset, when trained from monocular sequences only. 'Input Frames' indicate how many frames are fed to the pose network. 'Shared encoder (*arXiv v1*)' denotes the architecture proposed in *v1* of this paper.

H. Comparison of Pose Encoder

In Table 12 we evaluate different pose encoders. In an earlier version of this paper, we proposed the use of a shared pose encoder that shared features with the depth network. This resulted in fewer parameters to optimize during training, but also results in a decrease in depth prediction accuracy, see Table 12. As a baseline we compare against the pose network used by [62], which builds upon [76] with an additional scaling of the translation by the mean of the inverse depth. Overall, our separate encoder is superior for both pretrained and non-pretrained variants, whether we use two or three frames as input.

I. Supplementary Video Results

In the supplementary video, we show results on 'Wander', a monocular dataset collected from the 'Wind Walk Travel Videos' YouTube channel.[2] This dataset is quite different from the car mounted videos of KITTI as it only features a *monocular* hand-held camera in a non-European environment. We train on four sequences and present results on a fifth unseen sequence. We use an input/output resolution of 128×224. As with our KITTI experiments we train for 20 epochs with a batch size of 12, with a learning rate of 10^{-4} which is reduced by a factor of 10 for the final 5 epochs. For these handheld videos we found that the SSIM loss produced artifacts at object edges. As a result, we used a feature reconstruction loss in the appearance matching term, as in [52, 58, 73], by computing the L1 distance on the reprojected and normalized `relu1_1` features from an ImageNet pretrained VGG16 [57] as our pe function. This takes significantly longer to train, but results in qualitatively better depth maps on this dataset. Examples of predicted depths can be seen in Fig. 12.

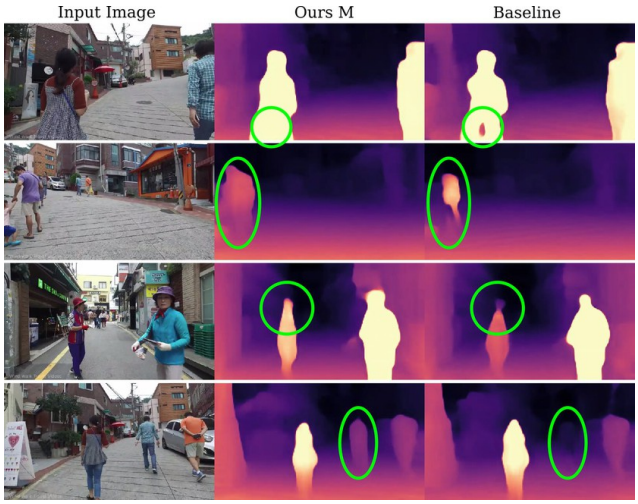

Figure 12. **Additional Wander results.** We observe that our model (Ours M) results in fewer visual artifacts when compared to the the baseline (*i.e.* the same model including VGG loss, but without our contributions).

[2] `https://www.youtube.com/channel/UCPur06mx78RtwgHJzxpu2ew`

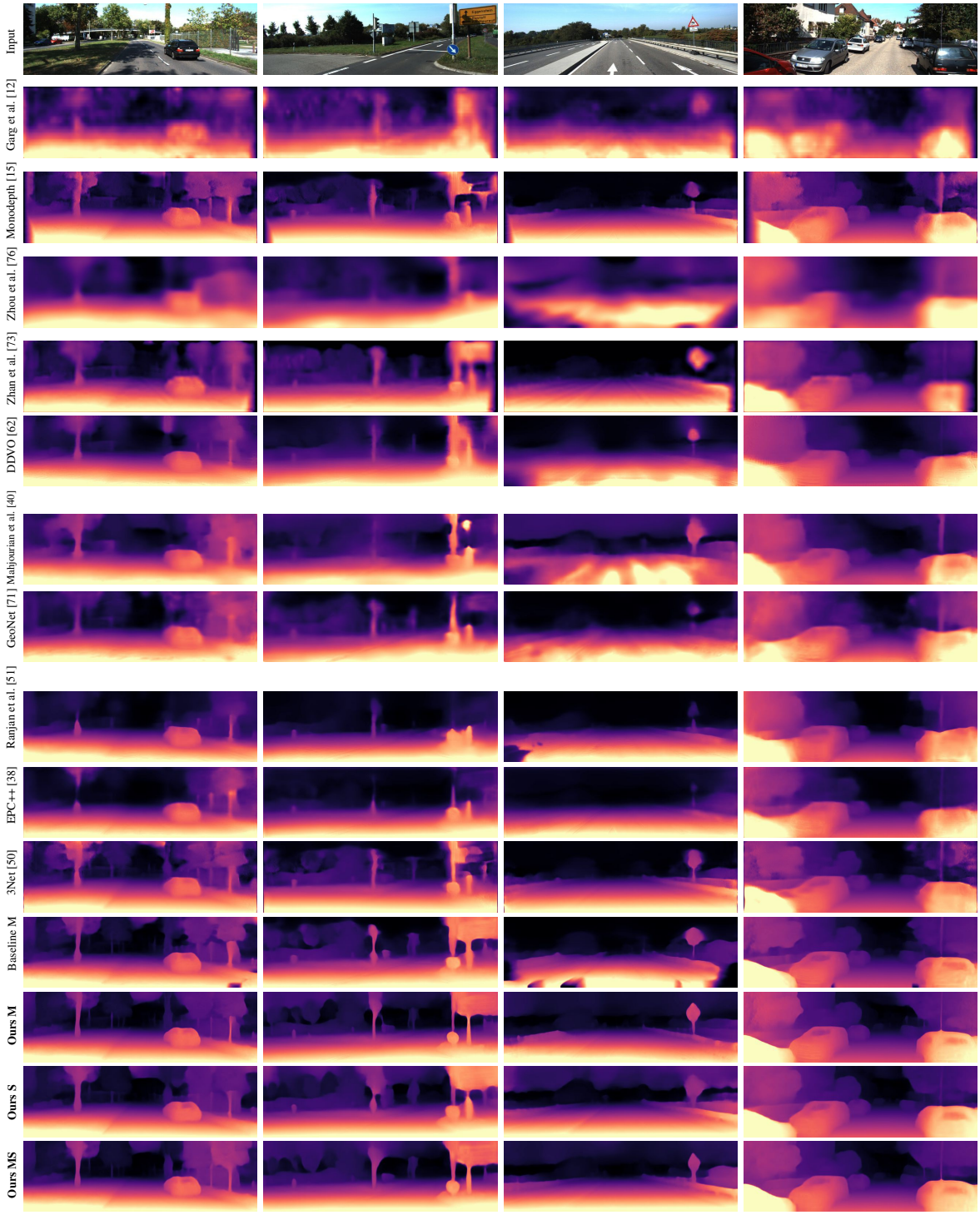

Figure 13. **Additional KITTI Eigen split test results.** We can see that our approaches in the last three rows produce the sharpest depth maps. 'Baseline M' is our model without our contributions.