The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth

Jamie Watson\textsuperscript{1} Oisin Mac Aodha\textsuperscript{2} Victor Prisacariu\textsuperscript{1,3} Gabriel Brostow\textsuperscript{1,4} Michael Firman\textsuperscript{1}
\textsuperscript{1}Niantic \textsuperscript{2}University of Edinburgh \textsuperscript{3}University of Oxford \textsuperscript{4}UCL

www.github.com/nianticlabs/manydepth

Abstract

Self-supervised monocular depth estimation networks are trained to predict scene depth using nearby frames as a supervision signal during training. However, for many applications, sequence information in the form of video frames is also available at test time. The vast majority of monocular networks do not make use of this extra signal, thus ignoring valuable information that could be used to improve the predicted depth. Those that do, either use computationally expensive test-time refinement techniques or off-the-shelf recurrent networks, which only indirectly make use of the geometric information that is inherently available.

We propose ManyDepth, an adaptive approach to dense depth estimation that can make use of sequence information at test time, when it is available. Taking inspiration from multi-view stereo, we propose a deep end-to-end cost volume based approach that is trained using self-supervision only. We present a novel consistency loss that encourages the network to ignore the cost volume when it is deemed unreliable, e.g. in the case of moving objects, and an augmentation scheme to cope with static cameras. Our detailed experiments on both KITTI and Cityscapes show that we outperform all published self-supervised baselines, including those that use single or multiple frames at test time.

1. Introduction

Knowing the depth to each pixel in an image has proved to be a useful and versatile tool, with applications ranging from augmented reality [59], autonomous driving [22], through to 3D reconstruction [64]. While specialist hardware can give per-pixel depth, e.g. from structured light or Lidar sensors, a more attractive approach is to only require a single RGB camera. Many recent monocular depth from RGB methods are trained using only self-supervision, which removes the need for expensive hardware to capture training depth data [21, 99, 23, 26]. While these approaches appear to be very promising, their test-time depth estimation performance is not yet on a par with specialist depth hardware or deep multi-view methods [73].

In an attempt to close this performance gap, we observe that in most practical scenarios more than one frame is available at test time e.g. from a camera on a moving vehicle or micro-baseline frames from a phone camera [32, 41]. Yet these additional frames are typically not exploited by current monocular methods. In this work, we use these additional frames at both training and test time, when they are available, to self-supervise a multi-frame depth estimation system. We show that a straightforward application of self-supervised training to a multi-view plane-sweep stereo architecture produces poor results, significantly worse than self-supervised single frame networks. To overcome this, we introduce several innovations to address issues caused by moving objects, scale ambiguity, and static cameras. We call our resultant multi-frame system ManyDepth.

We make the following three contributions:

1. A novel self-supervised multi-frame depth estimation model that combines the strengths of monocular and multi-view depth estimation by making use of multiple frames at test time, when they are available.
2. We show that moving objects and static scenes significantly impact self-supervised multi-view matching approaches, and we introduce efficient losses and train-
ing solutions to alleviate this problem.

3. We propose an adaptive cost volume to overcome the scale ambiguity arising from self-supervised training on monocular sequences. To the best of our knowledge, this is the first time cost volume extents have been learned from data rather than set as parameters.

Our ManyDepth model outperforms existing single and multi-frame approaches on the KITTI and Cityscapes datasets.

2. Related work

2.1. Monocular depth estimation

The goal of monocular depth estimation is to predict the depth of each pixel in a single input image. Supervised approaches either make use of dense supervision from depth sensors e.g. [15, 14, 20] or sparse supervision from human annotations e.g. [8]. Self-supervised methods remove the limitation of requiring ground truth depth supervision, instead training with image-reconstruction losses using stereo pairs [87, 21, 23] or monocular video sequences [99].

Recent advances in self-supervised training have focused on addressing various challenges resulting from learning from images alone e.g. more robust image reconstruction losses [26, 72], discrete rather than continuous depth predictions [56, 25, 40], feature space reconstruction losses [93, 72], sparse automatically generated depth supervision [49, 83], occlusion handling [26], improved network architectures [28], and moving objects during training [69, 24, 26, 78, 92, 9, 3, 75, 48, 53] and at test time [54]. Our underlying monocular architecture is based on [24], and could similarly benefit from many of the above enhancements.

2.2. Multi-frame monocular depth estimation

There is a growing number of works that extend existing self-supervised monocular models so that they can leverage temporal information at test time to improve the quality of the predicted depth. It is worth noting that there are also several non-deep-learning methods that also aim to produce consistent sequential depth estimates e.g. [96, 44], in addition to conventional SLAM based methods [65, 63, 16, 89], and SLAM methods that integrate a monocular depth estimation network [74, 4, 52]. However, here we focus on state-of-the-art neural network based depth estimation.

Test-time refinement approaches adapt monocular methods to use sequence information at test time e.g. [5, 9, 59, 62, 72, 51]. As self-supervised training does not require any ground truth depth supervision, the same losses used during training can be applied to the test frames to update the network’s parameters. The downside is that this necessitates the additional computation of multiple forward and backward model passes for a set of test frames, potentially taking several seconds to perform per set [62, 59].

<table>
<thead>
<tr>
<th>Method</th>
<th>Needs Depth</th>
<th>Needs Pose</th>
<th>Needs Object Motion</th>
<th>Single Frame</th>
<th>Multi Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Frame e.g. [2]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Supervised MVS e.g. [37]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Self-sup MVS e.g. [46]</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supervised MD e.g. [20]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Self-sup MD e.g. [99]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1. Comparison of existing approaches that estimate depth from collections of images. Our approach requires no ground-truth supervision and is robust to object motion. MVS stands for Multi-View Stereo, and MD stands for Monocular Depth.

A second broad group of approaches combine traditional monocular networks with recurrent layers to process sequences of frames e.g. [66, 81, 50, 97]. A related approach uses pairs of sequential frames at test time, sharing features between the pose and depth modules [79] or computing depth-from-flow [88]. These approaches are much more efficient compared to test-time refinement, but they can be more computationally demanding during training due to the need to extract features from multiple frames in a sequence.

A further limitation of these methods is that they do not explicitly reason about geometry during inference; they simply rely on the network having learned how to extract meaningful temporal representations.

Our experiments show that our approach often outperforms test-time refinement in terms of accuracy while retaining the efficiency of recurrent methods at inference.

2.3. Deep multi-view depth estimation

Our problem of predicting depth from multiple frames is related to multi-view depth estimation. While early deep stereo methods used mostly convolutional layers to map from images pairs to depth using ground-truth supervision e.g. [61, 77, 55], [45] showed that integrating a plane-sweep stereo cost volume significantly improved results. Recent approaches improved the underlying architectures and contributed more effective ways of regularizing the cost volume [7, 94, 95, 10]. It is also possible to train stereo networks without ground truth supervision [98, 82, 1, 36], but these models are typically outperformed by supervised variants. Some works fuse conventional matching-based stereo estimation with monocular depth cues [71, 60, 17]. In contrast, we do not require stereo pairs during training or testing.

A more general version of the stereo-matching problem is multi-view stereo (MVS), which operates on unordered image collections. Early deep MVS methods used memory-expensive 3D grid representations e.g. [39, 43]. Current supervised approaches, e.g. [37, 80, 91, 38, 57], utilize cost volumes but assume ground truth depth and camera poses.
between a supervisory signal, and do not require stereo supervision.

To predict these poses \( \theta \) deep network \( \theta \) is trained with ground truth depth. Concurrent with our work, [85] learn depth from a cost volume, but they use stereo pairs and sparse supervised depth at training time and long sequences for pose estimation.

Most related to us are self-supervised MVS methods that also do not require any ground truth depth i.e. [46, 13]. However, there are several reasons why these existing self-supervised and supervised MVS methods aren’t applicable in many scenarios: (i) they need more than one input image at test time, (ii) they assume that the camera is not static, (iii) they typically require camera poses to be provided during training and sometimes also at test time, and (iv) they assume that there are no moving objects in the scene. Our approach leverages the best of monocular and multi-view methods by making use of sequence information at test time, when it is available, while also being robust to scene motion — see Table 1.

### 3. Problem setup

The aim of depth estimation is to predict a depth map \( D_t \), pixel-aligned with an input image \( I_t \). Conventional single-image depth estimation methods, e.g. [14, 23, 99], train a deep network \( \theta_{\text{depth}} \) to map from \( I_t \) to \( D_t \),

\[
D_t = \theta_{\text{depth}}(I_t).
\]

In contrast, like other multi-frame methods [66, 81], our model accepts as input \( N \) previous temporal video frames,

\[
D_t = \theta_{\text{depth}}(I_t, I_{t-1}, \ldots, I_{t-N}).
\]

While our model makes use of information from multiple frames, it also can operate in the regime where only one frame is available at test time. Unlike similar works e.g. [50, 97, 9, 59, 5], we do not use future frames at test time e.g. \( I_{t+1} \), the use of which would preclude online applications. At training time, we exploit previous and future frames as a supervisory signal, and do not require stereo supervision. We do not assume access to the true relative camera pose between \( I_t \) and the preceding frames; instead we learn to predict these poses \( \{ T_n \}_{n=1}^N \) at training and test time with a differentiable pose network \( \theta_{\text{pose}} \), following [99]. We also do not make use of any trained semantic models to mask moving objects e.g. [5, 26, 29]. We do, however, assume known fixed camera intrinsics \( K \) — though we could relax these requirements [26, 18].

### 4. Method

Our method starts with two well established components: self-supervised reprojection based training, and a multi-view cost volume. We then introduce three important innovations that enable cost volume matching to work with self-supervised training from monocular video: (1) adaptive cost volumes, (2) a method to prevent a failure mode we refer to as ‘cost volume overfitting’, and (3) augmentation for static cameras and single frame inputs.

#### 4.1. Self-supervised monocular depth estimation

Following [99], we train a self-supervision depth network using only video frames that are temporally close to \( I_t \). We use the current estimated depth \( D_t \) and the pose network \( \theta_{\text{pose}} \)’s estimate of relative camera pose \( T_{t-t+n} \) to synthesize the scene from the same viewpoint as \( I_t \), but only using pixels from neighboring source frames i.e. \( \{ I_{t+n}, n \in \{-1,1\} \} \). The synthesized counterpart to \( I_t \) is

\[
I_{t+n\to t} = I_{t+n}(\text{proj}(D_t, T_{t-t+n}, K)),
\]

(3)

where \( () \) is the sampling operator and proj returns the 2D coordinates of the depths in \( D_t \) when reprojected into the camera of \( I_{t+n} \). Note that while our cost volume, described later, only uses preceding frames to enable online applications, at training-time our reprojection loss uses future frames too. Following [24], for each pixel we optimize the loss for the best matching source image, by selecting the per pixel minimum over the reconstruction loss \( p_e \),

\[
L_p = \min_{n} p_e(I_t, I_{t+n\to t}).
\]

(4)

We set \( p_e \) as a combination of SSIM and \( L_1 \) losses, and we minimize this loss over all the pixels in the training images over four output scales; see [24] for more details.

#### 4.2. Building a cost volume

To exploit multiple input frames, inspired by [11, 42, 45] we build a cost volume which measures the geometric compatibility at different depth values between the pixels from \( I_t \) and nearby source frames from the input video. This requires knowledge of relative pose \( T \), which we estimate with the pose network \( \theta_{\text{pose}} \), trained using a reprojection loss. We define a set of ordered planes \( \mathcal{P} \), each perpendicular to the optical axis at \( I_t \) and with depths linearly spaced between \( d_{\min} \) and \( d_{\max} \). Each frame is encoded into a deep feature map \( F_t \) and warped to the viewpoint of \( I_t \) using each of the hypothesised alternative depths \( d \in \mathcal{P} \) using the known camera intrinsics and estimated pose. This creates
a warped feature map $F_{t+1-n-t,d}$. The final cost volume is constructed as the absolute difference between the warped features and the features from $I_t$, at each $d \in \mathcal{P}$. This is averaged over all source images, following [65]. The cost volume effectively says ‘for pixel $(i,j)$, what is the likelihood of the correct depth being $d$ for each $d$ in $\mathcal{P}$?’ Following [19], the cost volume is concatenated with features $F_t$ and used as input to a convolutional decoder which regresses the depth $D_t$. See Fig. 2 for an overview.

Cost volumes have the benefit of allowing the network to leverage inputs from multiple viewing angles. However, they typically require $d_{\text{min}}$ and $d_{\text{max}}$ to be chosen as hyperparameters, and they assume that world is static. In the following sections we show how to relax these assumptions.

### 4.3. Adaptive cost volumes

Cost volume approaches have a problem of needing a known depth range i.e. $d_{\text{min}}$ and $d_{\text{max}}$. These are typically selected as hyperparameters in advance of training based on prior knowledge of the dataset [13] or from known camera poses [37]. We are unable to do this, as self-supervised depth estimation trained on monocular sequences only estimates depth ‘up to scale’. This means that while we assume that the final predicted depths, and corresponding poses from the pose network, will all end up in broad agreement with each other, they will be different from real-world depth by an unknown scaling factor.

To solve this problem we introduce a novel **adaptive cost volume**, by allowing $d_{\text{min}}$ and $d_{\text{max}}$ to be learned from the data, so they can adjust during training as the network finds its own scaling. This is done using the current predictions from the network of $D_t$, whereby we compute the average min and max of each $D_t$ over a training batch. These are then used to update an exponential moving average estimates of $d_{\text{min}}$ and $d_{\text{max}}$ with momentum 0.99. $d_{\text{min}}$ and $d_{\text{max}}$ are saved along with the model weights and then kept fixed at test time. Our approach contrasts to [27] who adapt $d_{\text{min}}, d_{\text{max}}$ at test time in a coarse-to-fine manner.

### 4.4. Addressing cost volume overfitting

We observe that our baseline cost volume model trained with monocular supervision suffers from severe artefacts, including large ‘holes’ punched on moving objects. These are similar to artefacts observed in monocular $I_t \rightarrow D_t$ models (see [5, 24] for a description). However, in our cost volume network they are far more severe (see Fig. 3 (c)).

**Why does the monocular-trained cost volume fail?** In theory, our model should do well. It is trained with a similar reprojection loss used to train state-of-the-art single-frame depth estimators, but it also has access to an additional source of information via the cost volume. However, the information contained in the cost volume is only reliable in specific scenarios e.g. in static regions with textured surfaces. In regions where objects are moving, or where surfaces are untextured, the cost volume will be an unreliable source of depth information (Fig. 3 (b)). For example, the moving car in Fig. 3 results in a match in the cost volume at an incorrect depth and corresponds to a very low reprojection loss. During training, the network becomes over-reliant on the cost volume. Instead of ignoring the cost volume around moving objects, it trusts it too much. Artefacts in the cost volume from moving objects are then introduced in the final depth map, at both training and test time. Ultimately, the final predicted depths inherit the cost volume’s mistakes. We introduce a method to correct this during training, by teaching the network not to trust the cost volume in these unreliable regions.

**Using a separate network to regularize.** We make the observation that single-image depth networks do not have a cost volume, so are unaffected by ‘cost volume overfitting’. While moving objects can still be a problem for these meth-
odds during training [5, 24, 69], in general they make far less severe mistakes on moving objects. We therefore use a monocular network at training time to help ‘teach’ our cost volume network the right answer — but only in regions where we suspect the cost volume to be problematic. This separate network $\theta_{\text{consistency}}$ produces a depth map $\hat{D}_t$ for each training image, and is discarded after training. $\theta_{\text{consistency}}$ shares $\theta_{\text{pose}}$ with our main network to help ensure scale-consistent predictions between $\theta_{\text{depth}}$ and $\theta_{\text{consistency}}$. Potentially problematic pixels in our multi-frame output are identified by a binary mask $M$. In these masked regions we apply an $L_1$ loss on $D_t$, encouraging the predictions to be similar to $\hat{D}_t$,
\[
L_{\text{consistency}} = \sum M |D_t - \hat{D}_t|.
\] (5)

Gradients to $\hat{D}_t$ are blocked, ensuring knowledge only transfers from teacher to student and not vice versa.

**Identifying unreliable pixels.** Our binary mask $M$ is 1 in regions considered to be unreliable, and 0 otherwise. To generate this mask we again make use of $\hat{D}_t$. We reason that in regions where the cost volume is reliable, the depth represented by $D_t$ will be similar to the depths represented by the argmin of the cost volume. We therefore compare the depth represented by the argmin of the cost volume (i.e. $D_{cv}$, not $D_t$) to the depth $\hat{D}_t$ predicted by $\theta_{\text{consistency}}$. The mask $M$ is set to 1 only in regions where $\hat{D}_t$ and $D_{cv}$ differ significantly, so

\[
M = \max \left(\frac{D_{cv} - \hat{D}_t}{D_t}, \frac{\hat{D}_t - D_{cv}}{D_t}\right) > 1.
\] (6)

The idea of using a separate ‘disposable’ network to help to regularize training is not new, e.g. [69, 99]. Our novelty is in using a single-frame depth network to improve a multi-frame system. Our approach is also less costly and less constrained than using offline semantic segmentation [26], and makes fewer assumptions than RANSAC-based filtering [29]. In our experiments we compare to two alternative masking schemes from [69] and [24], and show that our approach is superior.

4.5. Static cameras and start-of-sequences

Using multiple frames at test time introduces two potential challenges for our method. The first issue is when $I_{t-1}$ does not exist, i.e. when predicting depth for a single image or those at the start of a sequence. This case is trivially handled by monocular methods as they only require single frames as input. However, MVS approaches fail in these situations. To address this problem, during training with probability $p$, we replace the cost volume with a tensor of zeros. For these images, this encourages the network to learn to rely only on features directly from $I_t$. Then at test time, when $I_{t-1}$ does not exist, we simply replace the cost volume with zeros. The second case arises when the camera does not move between $I_{t-1}$ and $I_t$, e.g. a car stopped at traffic lights. Again this is another failure case for MVS methods. To address this at training time, with probability $q$, we replace the $I_{t-1}$ input to the cost volume with a color-augmented version of $I_t$, but still supervise with the ‘real’ $I_{t-1}, I_{t+1}$ in Eqn. 4. This enables the network to predict plausible depths even when the cost volume is constructed from images with no camera baseline.

Our final loss is $L = (1 - M) L_p + L_{\text{consistency}} + L_{\text{smooth}}$, where $L_{\text{smooth}}$ is the smoothness loss from [23].

5. Implementation details

We use training-time color and flip augmentations on images being fed to the depth and pose networks, using the settings from [24]. Unless otherwise stated, all our models are trained with an input and output resolution of 640×192, and we fix $N = 1$, so the cost volume is constructed with frames $\{I_t, I_{t-1}\}$, at both training and test time. In all cases self-supervision during training is from frames $\{I_{t-1}, I_t, I_{t+1}\}$. We train with Adam [47] for 20 epochs with a learning rate of $10^{-4}$, dropping by a factor of 10 for the final 5 epochs. After $Q$ epochs, we fix $\alpha_{\text{min}}$ and $\alpha_{\text{max}}$ and the weights of $\theta_{\text{pose}}$ and $\theta_{\text{consistency}}$. This allows $\theta_{\text{depth}}$ to finetune with a non-moving target. We set $Q = 15$ for KITTI, and $Q = 5$ for Cityscapes to account for the larger number of images.

---

**Figure 3.** We enable self-supervised training to work with a cost volume. When we build a cost volume from a sequence (a), moving objects create incorrect depths in the cost volume (b). These errors are propagated through to our predicted depth maps (c). ‘Traditional’ single-image depth estimation does not exhibit this failure mode, but can still produce biased depth (d). We therefore use a single-image network to help our model recover from this failure mode, but only on pixels identified to be unreliable by our mask $M$ (e). Our final prediction (f) is superior to both our baseline and the single-image network.
6. Experiments

Here we evaluate our ManyDepth model and (1) show that it gives SOTA results by comparing, in a standardized way, to both single-frame and multi-frame depth estimation and (2) validate our design decisions via ablations. Additional results are provided in the supplementary material.

We evaluate on two challenging depth estimation datasets, both of which exhibit moving objects. For both, we use the standard depth evaluation metrics from [14, 15].

(a) KITTI [22]. We use the Eigen split from [14]. This is commonly used for single frame depth estimation, but is more recently also used for multi-frame approaches e.g. [81, 66]. 22 frames in the KITTI Eigen test set are at the start of a sequence, and do not have a previous frame. We still include these images in the evaluation. For these images, the network does not have access to any other frames and thus makes a prediction based on one frame only. In the supplementary material, we additionally include models evaluated on the improved KITTI ground truth [76].

(b) Cityscapes [12]. Following [99, 90, 92], we train on 69,731 images from the monocular sequences, which we preprocess into triples using the scripts from [99]. We do not use stereo pairs or semantics. We evaluate on the 1,525 test images using the provided SGM [34] disparity maps. As with KITTI, we clip predicted depths at 80m, and only evaluate on ground truth depths less than 80m.

6.1. KITTI results

In Table 2 we compare to multi-frame approaches, some of which, e.g. [9, 5, 66, 79, 62], see more frames than ours or also use future frames e.g. [9, 5, 62]. We do not include results from [56] as they do not provide their scores on the KITTI Eigen split (see [88]). We additionally compare to the best-performing self-supervised monocular depth estimation approaches. To control for resolution, we separate low and high resolution models, and we also split methods which use expensive multi-pass test-time refinement into separate sections. We observe that our approach outperforms all previously published self-supervised methods that do not use semantic supervision on most metrics. We also
implement the test-time refinement scheme of [62] on our model, updating the weights of the depth and pose encoders using sequential pairs of images from the test set, for 50 steps. Not surprisingly, this further improves our results, and we outperform other test-time refinement methods.

Qualitative results are presented in Fig. 4. In some cases the predicted depth maps looks qualitatively similar to the monocular only models, but the error maps show the high magnitude of mistakes which can be present.

**Efficiency comparison.** Fig. 5 illustrates the runtime efficiency of our ManyDepth models (640 x 192: □, ■ and 1024 x 320: □, ◆) compared to other methods, including test-time refinement approaches (×). We report multiply-add computations (MACs) for each method and show that test-time refinement models which perform multiple forward-backward passes are too computationally demanding for use in real-time applications. The supplementary material contains a full results table and additional details.

### 6.2. KITTI ablation

In Table 4 we show the importance of the various components of our approach by turning them on and off in turn. **ManyDepth w/o motion masking:** We omit \( I_{\text{consistency}} \) from our loss and set \( M \) to zeros everywhere.

**ManyDepth w/o motion masking, w/o augmentation:**
As above, but also omitting our augmentations.
Table 3. Results on Cityscapes. Our method gives superior performance to all competing models. † is trained by us with the authors’ code, with preprocessing from [99]. Results from [67] are their ‘Half-Cycle Mono’ model, their only variant not requiring test-time stereo pairs. We evaluate using the cropping scheme of [6] following conversations with the authors; see supplementary material for specifics.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test frames</th>
<th>Semantics</th>
<th>WxH</th>
<th>Abs Rel</th>
<th>Sq Rel</th>
<th>RMSE</th>
<th>RMSE log</th>
<th>δ &lt; 1.25</th>
<th>δ &lt; 1.25²</th>
<th>δ &lt; 1.25³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Struct2Depth 2 [6]</td>
<td>1</td>
<td>*</td>
<td>416 x 128</td>
<td>0.145</td>
<td>1.137</td>
<td>7.280</td>
<td>0.205</td>
<td>0.813</td>
<td>0.942</td>
<td>0.976</td>
</tr>
<tr>
<td>Pilzer et al. [67]</td>
<td>1</td>
<td></td>
<td>512 x 256</td>
<td>0.240</td>
<td>4.264</td>
<td>8.049</td>
<td>0.334</td>
<td>0.710</td>
<td>0.871</td>
<td>0.937</td>
</tr>
<tr>
<td>Monodepth2 [23]</td>
<td>1</td>
<td></td>
<td>416 x 128</td>
<td>0.129</td>
<td>1.569</td>
<td>6.876</td>
<td>0.187</td>
<td>0.849</td>
<td>0.957</td>
<td>0.983</td>
</tr>
<tr>
<td>Videos in the Wild [26]</td>
<td>1</td>
<td>*</td>
<td>416 x 128</td>
<td>0.127</td>
<td>1.330</td>
<td>6.960</td>
<td>0.195</td>
<td>0.830</td>
<td>0.947</td>
<td>0.981</td>
</tr>
<tr>
<td>Li et al. [53]</td>
<td>1</td>
<td></td>
<td>416 x 128</td>
<td>0.119</td>
<td>1.290</td>
<td>6.980</td>
<td>0.190</td>
<td>0.846</td>
<td>0.952</td>
<td>0.982</td>
</tr>
<tr>
<td>Struct2Depth 2 [6]</td>
<td>3 (-1, 0, +1)</td>
<td></td>
<td>416 x 128</td>
<td>0.222</td>
<td>5.737</td>
<td>8.613</td>
<td>0.258</td>
<td>0.774</td>
<td>0.908</td>
<td>0.954</td>
</tr>
<tr>
<td>Struct2Depth 2 [6]</td>
<td>3 (-1, 0, +1)</td>
<td>*</td>
<td>416 x 128</td>
<td>0.151</td>
<td>2.492</td>
<td>7.024</td>
<td>0.202</td>
<td>0.826</td>
<td>0.937</td>
<td>0.972</td>
</tr>
<tr>
<td>ManyDepth</td>
<td>2 (-1, 0)</td>
<td></td>
<td>416 x 128</td>
<td>0.114</td>
<td>1.193</td>
<td>6.223</td>
<td>0.170</td>
<td>0.875</td>
<td>0.967</td>
<td>0.989</td>
</tr>
<tr>
<td>ManyDepth full</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ManyDepth (w/o motion masking)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ManyDepth (w/o motion masking, w/o aug.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ManyDepth (w/o motion masking, w/o teacher)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack of 2 frames as input (I_{t−1}, I_t)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ManyDepth (with motion masking from [69])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ManyDepth (with motion masking from [24])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Khot et al. [46]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ManyDepth with 3-frame input (I_{t−2}, I_{t−1}, I_t)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ManyDepth with 3-frame input (I_{t−1}, I_t, I_{t+1})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ManyDepth full</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Our contributions lead to better scores. Here we ablate our ManyDepth method on KITTI 2015 [22] using the Eigen split. Full numbers are in the supplementary material.

<table>
<thead>
<tr>
<th>Ablation</th>
<th>Abs Rel</th>
<th>Sq Rel</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManyDepth full</td>
<td>0.998</td>
<td>0.770</td>
<td>4.459</td>
</tr>
<tr>
<td>ManyDepth (w/o motion masking)</td>
<td>0.113</td>
<td>1.354</td>
<td>5.228</td>
</tr>
<tr>
<td>ManyDepth (w/o motion masking, w/o aug.)</td>
<td>0.284</td>
<td>11.240</td>
<td>8.516</td>
</tr>
<tr>
<td>ManyDepth (w/o motion masking, w/o teacher)</td>
<td>0.154</td>
<td>2.682</td>
<td>6.573</td>
</tr>
<tr>
<td>Stack of 2 frames as input (I_{t−1}, I_t)</td>
<td>0.121</td>
<td>1.028</td>
<td>5.016</td>
</tr>
<tr>
<td>ManyDepth (with motion masking from [69])</td>
<td>0.099</td>
<td>0.783</td>
<td>4.447</td>
</tr>
<tr>
<td>ManyDepth (with motion masking from [24])</td>
<td>0.099</td>
<td>0.780</td>
<td>4.465</td>
</tr>
<tr>
<td>Khot et al. [46] reimplementation</td>
<td>0.200</td>
<td>4.694</td>
<td>7.232</td>
</tr>
<tr>
<td>ManyDepth with 3-frame input (I_{t−2}, I_{t−1}, I_t)</td>
<td>0.098</td>
<td>0.780</td>
<td>4.430</td>
</tr>
<tr>
<td>ManyDepth with 3-frame input (I_{t−1}, I_t, I_{t+1})</td>
<td>0.097</td>
<td>0.768</td>
<td>4.431</td>
</tr>
</tbody>
</table>

Table 5. Our augmentations help in static camera and start-of-sequence cases. We compare two variants of our model, one trained with our novel data augmentations (‘Ours’) and one without. We create two artificial scenarios to test each model’s performance on start-of-sequence images (where we just input I_t) and static cameras (where both input frames are the exact same).

<table>
<thead>
<tr>
<th>Test-time input</th>
<th>Model</th>
<th>Abs Rel</th>
<th>Sq Rel</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard: (I_{t−1}, I_t)</td>
<td>No augmentation</td>
<td>ManyDepth</td>
<td>0.100</td>
<td>0.794</td>
</tr>
<tr>
<td>Monodepth2 [23]</td>
<td>No augmentation</td>
<td>ManyDepth</td>
<td>0.098</td>
<td>0.770</td>
</tr>
<tr>
<td>Start-of-sequence: (I_t)</td>
<td>Monodepth2 [23]</td>
<td>No augmentation</td>
<td>ManyDepth</td>
<td>0.115</td>
</tr>
<tr>
<td>Static camera: (I_t, I_{t+1})</td>
<td>No augmentation</td>
<td>ManyDepth</td>
<td>0.138</td>
<td>1.132</td>
</tr>
</tbody>
</table>

Benefit of our augmentations. In Table 5 we evaluate three different scenarios, comparing our model to a baseline which was trained without our augmentations from Section 4.3. When evaluating in ‘standard’ mode (i.e. using the previous and current frames as input) on the entire KITTI test set, the difference between the two models is negligible. This is partially because the KITTI test images are predominately from a moving camera. However, when we evaluate in ‘start-of-sequence’ mode (i.e. the standard monocular setting using only (I_t) as input) and ‘static camera’ evaluation mode (i.e. simulating a static camera with inputs (I_t, I_{t+1})), our augmentation scheme is significantly better.

6.3. Cityscapes results

In Table 3 we perform additional comparisons where we train and test on the Cityscapes dataset [12]. Again, we consistently outperform competing methods, even those that use semantic supervision.

7. Conclusion

We presented a fully self-supervised online method that predicts superior depths from a single image, or from multiple images when they are available. We achieve the benefits of both multi-frame and monocular methods, while being more robust on moving objects and static cameras compared to a naive integration of a cost volume. We presented state-of-the-art results on both the KITTI and Cityscapes datasets. While test-time refinement methods are close competitors in terms of depth accuracy, we have shown that our method is significantly more efficient during inference. We expect that our results could be further improved via recent complementary advances in monocular depth estimation e.g. discretized output depths [25] or feature based losses [72].

Acknowledgements. Thanks to Jamie Shotton; a conversation with him at a BMVA workshop motivated work in this area. Also to Daniyar Turmukhametov and Sara Vicente for their valuable feedback, and to the authors of [6] for CityScapes evaluation help.
References


[38] Sungshoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DPSNet: End-to-end deep plane sweep stereo. ICLR, 2019.


[58] Chenzxu 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. PAMI, 2019.


