

DeepTAM: Deep Tracking and Mapping with Convolutional Neural Networks

Huizhong Zhou^{*,1} Benjamin Ummenhofer^{*,2} Thomas Brox¹

Received: 31 January 2019 / Accepted: 26 August 2019

Abstract We present a system for dense keyframe-based camera tracking and depth map estimation that is entirely learned. For tracking, we estimate small pose increments between the current camera image and a synthetic viewpoint. This formulation significantly simplifies the learning problem and alleviates the dataset bias for camera motions. Further, we show that generating a large number of pose hypotheses leads to more accurate predictions. For mapping, we accumulate information in a cost volume centered at the current depth estimate. The mapping network then combines the cost volume and the keyframe image to update the depth prediction, thereby effectively making use of depth measurements and image-based priors. Our approach yields state-of-the-art results with few images and is robust with respect to noisy camera poses.

We demonstrate that the performance of our 6 DOF tracking competes with RGB-D tracking algorithms. We compare favorably against strong classic and deep learning powered dense depth algorithms.

*Equal contribution.

This project was in large parts funded by the EU Horizon 2020 project Trimbot2020. We also thank the bwHPC initiative for computing resources, Facebook for their P100 server donation and gift funding.

Huizhong Zhou
zhouh@cs.uni-freiburg.de

Benjamin Ummenhofer
benjamin.ummehofer@intel.com

Thomas Brox
brox@cs.uni-freiburg.de

¹ University of Freiburg

² Intel Labs

1 Introduction

In contrast to recognition, there is limited work on applying deep learning to camera tracking or 3D mapping tasks. This is because, in contrast to recognition, the field of 3D mapping is already in possession of very good solutions. Nonetheless, learning approaches have much to offer for camera tracking and 3D mapping. On the limited number of subtasks, where deep learning has been applied, it has outperformed classical techniques: on disparity estimation all leading approaches are based on deep networks, and the first work on dense motion stereo (Ummenhofer et al, 2017) immediately achieved state-of-the-art performance on this task.

In this work, we extend the domain of learning-based mapping approaches further towards full-scale SLAM systems. We present a deep learning approach for the two most important components in visual SLAM: camera pose tracking, and dense mapping.

The main contribution of the paper is a learned tracking network and a mapping network, which generalize well to new datasets and outperform strong competing algorithms. This is achieved by the following key components:

- a tracking network architecture for incremental frame to keyframe tracking designed to reduce the dataset bias problem.
- a multiple hypothesis approach for camera poses which leads to more accurate pose estimation.
- a mapping network architecture that combines depth measurements with image-based priors, which is highly robust and yields accurate depth maps.
- an efficient depth refinement strategy combining a network with the narrow band technique.

The most related classical approach is DTAM (Newcombe et al, 2011), which stands for Dense Tracking And Mapping. Conceptually we follow a very similar approach, except that we formulate it as a learning problem. Consequently, we call our approach DeepTAM.

For tracking, DeepTAM uses a neural network for aligning the current camera image to a keyframe—color and depth image—to infer the camera pose. To this end we use a small and fast stack of networks which implement a coarse-to-fine approach. The network stack incrementally refines the estimated camera pose. In each step we update a virtual keyframe, thereby improving convergence of the predicted camera pose. This incremental formulation significantly simplifies the learning task and reduces the effects of dataset bias. In addition, we show that generating a large number of hypotheses improves the pose accuracy.

Our mapping network is built upon the plane sweep stereo idea (Collins, 1996). We first accumulate information from multiple images in a cost volume, then extract the depth map using a deep network by combining image-based priors with the accumulated depth measurements. To further improve the depth prediction we append a network, which iteratively refines the prediction using a cost volume defined on a narrow band around the previous surface estimate. The obtained depth can be a valuable cue for many vision tasks, e.g. object localization (Song and Chandraker, 2015; Dhiman et al, 2016), scene understanding (Gupta et al, 2013, 2015), image dehazing (Fattal, 2008; Zhang and Patel, 2018a,b).

As a learning approach, DeepTAM is very good at integrating various cues and learning implicit priors about the used camera. This is in contrast to classic approaches which fundamentally rely on handcrafted features like SIFT (Lowe, 2004) and photoconsistency maximization. A well known problem of learning-based approaches is overfitting, and we took special care in the design of the architecture and the definition of the learning problem so that the network cannot learn simple shortcuts that would not generalize.

As a consequence, DeepTAM generalizes well to new datasets and is the first learned approach with full 6 DOF keyframe pose tracking and dense mapping. On standard benchmarks, it compares favorably to state-of-the-art RGB-D tracking, while using less data. DeepTAM employs dense mapping that can process arbitrary many frames and runs at interactive frame rates.

2 Related work

The most related work is DTAM (Newcombe et al, 2011). We build on the same generic idea: drift-free

camera pose tracking via a dense depth map towards a keyframe and aggregation of depth over time. However, we use completely different technology to implement this concept. In particular, both the tracking and the mapping are implemented by deep networks, which solely learn the task from data.

Most related with regard to the learning methodology is DeMoN (Ummenhofer et al, 2017), which implements 6 DOF egomotion and depth estimation for two images as a learning problem. In contrast to DeMoN, we process more than two images. We avoid drift by the use of keyframes, and we can refine the depth map as more frames are coming in.

A few more works based on deep learning have appeared recently that have a weak connection to the present work. Weerasekera et al (2018) and Weerasekera et al (2017) present a dense monocular reconstruction framework which uses convolutional neural networks to extract visual features that are best suited for the reconstruction. Agrawal et al (2015) trains a neural network to estimate the egomotion, which mainly serves as a supervision for feature learning. Kendall and Cipolla (2017) apply deep learning to the camera localization task and Valada et al (2018) show that the visual localization and odometry can be solved jointly within one network. DeepVO (Wang et al, 2017) runs a deep network for visual odometry, i.e., regressing the egomotion between two frames. There is no mapping part, and the egomotion estimation only works for environments seen during training. Zhou et al (2017) and Zhan et al (2018) presented a deep network for egomotion and depth estimation that can be trained with an unsupervised loss. The approach uses two images for depth estimation during training. However, it ignores the second image when estimating the depth at runtime, hence ignoring the motion parallax. SfM-Net (Vijayanarasimhan et al, 2017), too, uses unsupervised learning ideas, and (despite its title) does not use the motion parallax for depth estimation. UnDeepVO (Li et al, 2018) proposed egomotion estimation and depth estimation again based on an unsupervised loss. All these works are like DeMoN limited to the joint processing of two frames and limited to the motions present in the datasets.

Training and experiments in most of these previous works (Zhou et al, 2017; Wang et al, 2017; Li et al, 2018) focus on the KITTI dataset (Geiger et al, 2012). These driving scenarios mostly show 3 DOF motion in a plane, which is induced by a 2 DOF action space (accelerate/brake, steer left/steer right). In particular the hard ambiguities between camera translation and rotation do not exist, since the car cannot move sideward. In contrast, the present work yields full 6 DOF pose

tracking, can handle these ambiguities, and we evaluate on a 6 DOF benchmark.

We cannot cover the full literature on classical tracking and mapping techniques, but there are some related works besides DTAM (Newcombe et al, 2011) that are worth mentioning. LSD-SLAM (Engel et al, 2014) is a state-of-the-art SLAM approach that uses direct measures for optimization. It provides a full SLAM pipeline with loop closing. In contrast to DTAM and our approach, LSD-SLAM only yields sparse depth estimates. Engel et al (2018) propose a sparse direct approach. They show that integrating a sophisticated model of the image formation process significantly improves the accuracy. For our learning-based approach, accounting for the characteristics of the imaging process is covered by the training process. Similarly, Kerl et al (2013b) carefully model the noise distribution to improve robustness. Again, this comes for free in a learning-based approach. CNN-SLAM (Tateno et al, 2017) extends LSD-SLAM with single image depth maps. In contrast to our approach, tracking and mapping are not coupled in a dense manner. In particular, the tracking uses a semi-dense subset of the depth map.

This paper is an extended version of the published work (Zhou et al, 2018). We extend our previous work by a thorough ablation study and cross-dataset evaluation. We analyze the design of our tracking and mapping module in more detail. In addition, to validate our generalization ability we perform a series of cross-dataset experiments. We show that our system can achieve state-of-the-art comparable performance by training only on the synthetic datasets and we can achieve higher accuracy with the help of real datasets. Besides, we evaluate our methods on KITTI (Geiger et al, 2012) without any finetuning.

3 System Overview

Similar as in existing SLAM methods like (Klein and Murray, 2007; Newcombe et al, 2011; Engel et al, 2014), we separate the tasks of tracking and mapping into two components. Both components are mutually dependent on each other and require the output of the respective other component. The tracking component computes the current camera pose with respect to a keyframe, which is a part of the map, by aligning the current camera image with the keyframe. The mapping component consumes the current camera pose and camera image to update the depth map associated with the keyframe. This separation of tasks provides us with the flexibility at runtime to run the two components at different update rates, i.e. we can compute the camera pose for each frame but collect data for the keyframe update over

multiple frames. Further, it allows to train networks for both components in parallel. In the following we present the design of both components and details about the training procedure. We also release our source code at <https://github.com/lmb-freiburg/deeptam>.

4 Tracking

Given the current camera image \mathbf{I}^C and a keyframe, which consists of an image \mathbf{I}^K and an inverse depth map \mathbf{D}^K , we want to estimate the 4×4 transformation matrix \mathbf{T}^{KC} that maps a point in the keyframe coordinate system to the coordinate system of the current camera frame. The keyframe pose \mathbf{T}^K and the current camera pose \mathbf{T}^C are related by

$$\mathbf{T}^C = \mathbf{T}^K \mathbf{T}^{KC}, \quad \text{with } \mathbf{T}^C, \mathbf{T}^K, \mathbf{T}^{KC} \in \mathbf{SE}(3). \quad (1)$$

Learning to compute \mathbf{T}^{KC} is related to finding 2D-3D correspondences between the current image \mathbf{I}^C and the keyframe $(\mathbf{I}^K, \mathbf{D}^K)$. It is well known that the correspondence problem can be solved more efficiently and reliably if pixel displacements between image pairs are small. Since we want to track the current camera pose at interactive rates, we assume that a guess \mathbf{T}^V close to \mathbf{T}^C is available. Similar to DTAM (Newcombe et al, 2011), we generate a virtual keyframe $(\mathbf{I}^V, \mathbf{D}^V)$ that shows the content of the keyframe $(\mathbf{I}^K, \mathbf{D}^K)$ from a viewpoint corresponding to \mathbf{T}^V . Instead of directly estimating \mathbf{T}^{KC} , we learn to predict the increment $\delta\mathbf{T}$, i.e., we write the current camera pose as

$$\mathbf{T}^C = \mathbf{T}^V \delta\mathbf{T}. \quad (2)$$

This effectively reduces the problem to learning the function $\delta\mathbf{T} = f(\mathbf{I}^C, \mathbf{I}^V, \mathbf{D}^V)$. We use a deep network to learn f .

4.1 Network Architecture

We use the encoder-decoder-based architecture shown in Fig. 1 for learning to estimate the 6 DOF pose between a keyframe $(\mathbf{I}^K, \mathbf{D}^K)$ and an image \mathbf{I}^C . A detailed description of all network parameters can be found in the supplementary material.

Since camera motion can only be estimated by relating the keyframe to the current image, we make use of optical flow as an auxiliary task. The predicted optical flow ensures that the network learns to exploit the relationship between both frames. We demonstrate the importance of the flow prediction in Tab. 1. We use the features shared with the optical flow prediction task in a second network branch for generating pose hypotheses. As we show in the experiments (Tab. 1), generating

multiple hypotheses improves the accuracy of the predicted pose compared to direct prediction of the pose.

The last part of the pose generation consists of $N = 64$ branches of stacked fully connected layers sharing their weights. We found that this configuration is more stable and accurate than a single branch of fully connected layers computing N poses. Each generated pose hypothesis is a 6D pose vector $\delta\xi_i = (\mathbf{r}_i, \mathbf{t}_i)^\top$. The 3D rotation vector \mathbf{r}_i is a minimal angle axis representation with the angle encoded as the magnitude of the vector. The translation \mathbf{t}_i is encoded in 3D Cartesian coordinates. For simplicity, and because $\delta\xi_i$ are small rigid body motions, we compute the final pose estimate $\delta\xi$ as the linear combination

$$\delta\xi = \frac{1}{N} \sum_{i=1}^{N=64} \delta\xi_i. \quad (3)$$

Coarse camera motions are already visible at small image resolutions, while small motions require higher image resolutions. Thus, we use a coarse-to-fine strategy to efficiently track the camera in real time. We train three distinct tracking networks as shown in Fig. 2, which deal with the pose estimation problem at different resolutions and refine the prediction of the respective previous resolution level.

4.2 Training

A major problem of learning-based approaches is the strong dependency on suitable datasets. Datasets often do not cover all important modes, which complicates generalization to new data. An example is the KITTI dataset for autonomous driving (Geiger et al, 2012), which is limited to motion in a plane and does not cover full 6 DOF motion. As a consequence, learning-based methods easily overfit to this type of motion and do not generalize. Artificial data can be used to alleviate this problem, but it is not trivial to generate realistic imagery with ground truth depth.

We tackle this problem in two ways. First by using the incremental formulation in (2), i.e., we estimate a small increment $\delta\mathbf{T}$ instead of the absolute motion between keyframe and current camera image. This reduces the magnitude of motion and reduces the difficulty of the task. Second, we use rendered images and depth maps as a proxy for real keyframes. Given a keyframe $(\mathbf{I}^K, \mathbf{D}^K)$, we sample the initial pose guess \mathbf{T}_0^V from a normal distribution centered at the ground truth pose \mathbf{T}^C to generate the virtual frame $(\mathbf{I}^V, \mathbf{D}^V)$. This simulates all possible 6 DOF motions and, thus, effectively augments the data to overcome the limited set of motions in the dataset.

4.2.1 Datasets

We train on image pairs from the SUN3D dataset (Xiao et al, 2013) and the SUNCG dataset (Song et al, 2017). For SUN3D we sample image pairs with a baseline of up to 40cm. For SUNCG we generate images with normally distributed baselines with standard deviation 15cm and rotation angles with standard deviation 0.15 radians. When sampling an image pair we reject samples with an image overlap of less than 50%. For keyframe depth maps \mathbf{D}^K , we use the ground truth depth from the datasets during training.

4.2.2 Training Objective

The objective function for the tracking network is

$$\mathcal{L}_{\text{tracking}} = \mathcal{L}_{\text{flow}}(\mathbf{w}) + \mathcal{L}_{\text{motion}}(\delta\xi) + \mathcal{L}_{\text{uncertainty}}(\delta\xi_i). \quad (4)$$

The predicted optical flow \mathbf{w} and the predicted poses $\delta\xi_i$ are the network’s outputs.

The loss $\mathcal{L}_{\text{flow}}$ defines the auxiliary optical flow task. We use the endpoint error

$$\mathcal{L}_{\text{flow}} = \sum_{i,j} \|\mathbf{w}(i,j) - \mathbf{w}_{\text{gt}}(i,j)\|_2, \quad (5)$$

which is a common error metric for optical flow.

The two losses $\mathcal{L}_{\text{motion}}$ and $\mathcal{L}_{\text{uncertainty}}$ for the generation of pose hypotheses are defined as:

$$\mathcal{L}_{\text{motion}} = \alpha \|\mathbf{r} - \mathbf{r}_{\text{gt}}\|_2 + \|\mathbf{t} - \mathbf{t}_{\text{gt}}\|_2, \text{ and} \quad (6)$$

$$\begin{aligned} \mathcal{L}_{\text{uncertainty}} = & \frac{1}{2} \log(|\Sigma|) - 2 \log\left(\frac{\mathbf{x}^\top \Sigma^{-1} \mathbf{x}}{2}\right) \\ & - \log\left(K_\nu\left(\sqrt{2\mathbf{x}^\top \Sigma^{-1} \mathbf{x}}\right)\right). \end{aligned} \quad (7)$$

The vectors \mathbf{r} and \mathbf{t} are the rotation and translation parts of the linear combination $\delta\xi$ defined in (3). We use the parameter $\alpha = \frac{3}{2}$ to balance the importance of both components. We combine this loss, which directly acts on the predicted average motion, with $\mathcal{L}_{\text{uncertainty}}$, which is the negative log-likelihood of the multivariate Laplace distribution. We compute Σ from the predicted pose samples as $\Sigma = \frac{1}{N} \sum_i^N (\delta\xi_i - \delta\xi)(\delta\xi_i - \delta\xi)^\top$, and the vector \mathbf{x} as $\mathbf{x} = \delta\xi - \delta\xi_{\text{gt}}$. During optimization we treat \mathbf{x} as a constant and we regularize Σ by adding a diagonal matrix $\epsilon\mathbf{I}$, where ϵ takes the value of 0.1. The function K_ν is the modified Bessel function of the second kind. We empirically found that a loss based on the multivariate Laplace distribution yields better results than the multivariate Normal distribution. The uncertainty loss pushes the network to predict distinct poses $\delta\xi_i$.

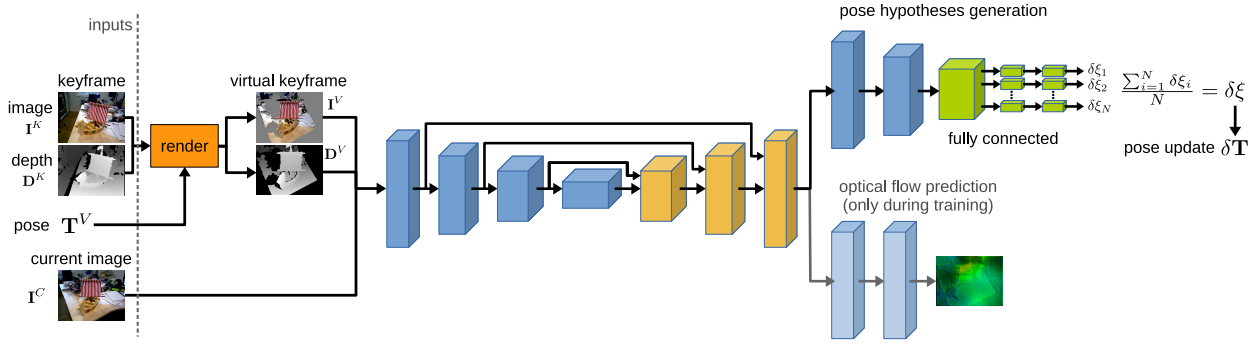


Fig. 1 The tracking network uses an encoder-decoder type architecture with direct connections between the encoding and decoding part. The decoder is used by two tasks, which are optical flow prediction and the generation of pose hypotheses. The optical flow prediction is a small stack of two convolution layers and is only active during training to stimulate the generation of motion features. The pose hypotheses generation part is a stack of downsampling convolution layers followed by a fully connected layer whose output we split into $N = 64$ chunks, which are then processed by fully connected layers with shared parameters to estimate the $\delta\xi_i$. Along with the current camera image \mathbf{I}^C we provide a virtual keyframe ($\mathbf{I}^V, \mathbf{D}^V$) as input for the network, which is rendered using the active keyframe ($\mathbf{I}^K, \mathbf{D}^K$) and the current pose estimate \mathbf{T}^V . We stack the depicted network architecture three times with each instance operating at a different resolution as shown in Fig. 2.

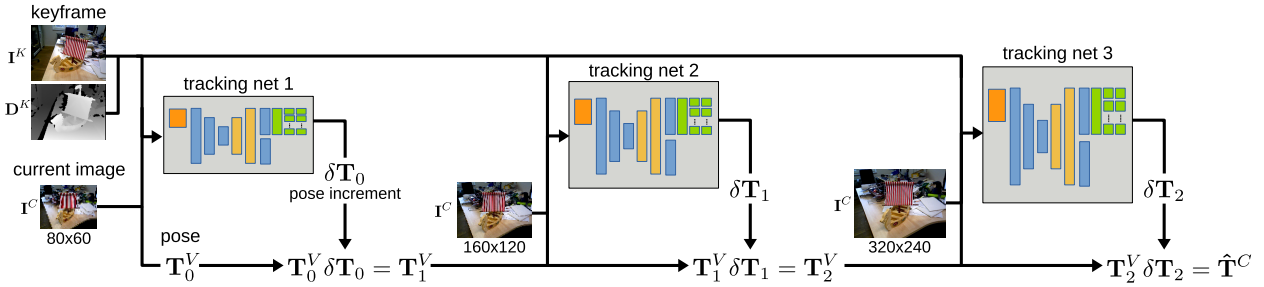


Fig. 2 Overview of the tracking networks and the incremental pose estimation. We apply a coarse-to-fine approach to efficiently estimate the current camera pose. We train three tracking networks each specialized for a distinct resolution level corresponding to the input image dimensions (80×60), (160×120) and (320×240). Each network computes a pose estimate $\delta\mathbf{T}_i$ with respect to a guess \mathbf{T}_i^V . The guess \mathbf{T}_0^V is the camera pose from the previously tracked frame. Each of the tracking networks uses the latest pose guess to generate a virtual keyframe at the respective resolution level and thereby indirectly tracking the camera with respect to the original keyframe ($\mathbf{I}^K, \mathbf{D}^K$). The final pose estimate $\hat{\mathbf{T}}^C$ is computed as the product of all incremental pose updates $\delta\mathbf{T}_i$.

We optimize using Adam (Kingma and Ba, 2015) with the learning rate schedule proposed in (Loshchilov and Hutter, 2017). We implement and train the networks with Tensorflow (Abadi et al, 2015). Training the tracking network takes less than a day on an NVIDIA GTX1080Ti. We provide the detailed training parameters in the supplementary material.

5 Mapping

We describe the geometry of a scene as a set of depth maps, which we compute for every keyframe. To achieve high quality depth maps we accumulate information from multiple images in a cost volume. The depth map is then extracted from the cost volume by means of a convolutional neural network.

Let \mathbf{C} be the cost volume and $\mathbf{C}(\mathbf{x}, d)$ the photo-consistency cost for a pixel \mathbf{x} at depth label $d \in B_{\text{fb}}$.

We define the set of N depth labels for a fixed range $[d_{\min}, d_{\max}]$ as

$$B_{\text{fb}} = \{b_i | b_i = d_{\min} + i \cdot \frac{d_{\max} - d_{\min}}{N-1}, i = 0, 1, \dots, N-1\}. \quad (8)$$

Given a sequence of m images $\mathbf{I}_1, \dots, \mathbf{I}_m$ along with their camera poses $\mathbf{T}_1, \dots, \mathbf{T}_m$, we compute the photo-consistency costs as

$$\mathbf{C}(\mathbf{x}, d) = \sum_{i \in \{1, \dots, m\}} \rho_i(\mathbf{x}, d) \cdot w_i(\mathbf{x}). \quad (9)$$

The photoconsistency $\rho_i(\mathbf{x}, d)$ is the sum of absolute differences (SAD) of 3×3 patches between the keyframe image \mathbf{I}^K and the warped image $\tilde{\mathbf{I}}_i$ at point \mathbf{x} for depth d . We obtain $\tilde{\mathbf{I}}_i$ using a warping function

$$\tilde{\mathbf{I}}_i = \mathcal{W}(\mathbf{I}_i, \mathbf{T}_i(\mathbf{T}^K)^{-1}, d), \quad (10)$$

which warps the image \mathbf{I}_i to the keyframe using the relative pose and the depth.

The weighting factor w_i is then computed as

$$w_i(\mathbf{x}) = 1 - \frac{1}{N-1} \sum_{d \in B_{fb} \setminus \{d^*\}} e^{-\nu \cdot (\rho_i(\mathbf{x}, d) - \rho_i(\mathbf{x}, d^*))^2}. \quad (11)$$

w_i describes the matching confidence and is close to 1 if there is a clear and unique minimum $\rho_i(\mathbf{x}, d^*)$ with $d^* = \arg \min_d \rho_i(\mathbf{x}, d)$. The scalar parameter ν is set to 10 in all experiments.

In classic methods the cost volume is taken as data term and a depth map can be obtained by searching for the minimum cost. However, due to noise in the cost volume, various sophisticated regularization terms and optimization techniques have been introduced (Hosni et al, 2013; Hirschmüller, 2005; Felzenszwalb and Huttenlocher, 2006) to extract the depth in a robust manner. Instead, we train a network to use the matching cost information in the cost volume and simultaneously combine it with the image-based scene priors to obtain more accurate and more robust depth estimates.

For cost-volume-based methods, accuracy is limited by the number of depth labels N . Hence, we use an adaptive narrow band strategy to increase the sampling density while keeping the number of labels constant. We define the narrow band of depth labels centered at the previous depth estimate d_{prev} as

$$B_{\text{nb}} = \{b_i | b_i = d_{\text{prev}} + i \cdot \sigma_{\text{nb}} \cdot d_{\text{prev}}, i = -\frac{N}{2}, \dots, \frac{N-2}{2}\}. \quad (12)$$

σ_{nb} determines the narrow band width. We recompute the cost volume for the narrow band for a small selection of frames and search again for a better depth estimate. The narrow band allows us to recover more details in the depth map, but also requires a good initialization and regularization to keep the band in the right place. We address these tasks using multiple encoder-decoder type networks. Fig. 3 shows an overview of the mapping architecture with the fixed band and narrow band stage.

5.1 Network Architecture

The network is trained to predict the keyframe inverse depth \mathbf{D}^K from the keyframe image \mathbf{I}^K and the cost volume \mathbf{C} computed from a set of images $\mathbf{I}_1, \dots, \mathbf{I}_m$ and camera poses $\mathbf{T}_1, \dots, \mathbf{T}_m$. \mathbf{D}^K is represented as inverse depth, which enables a more precise representation with closer distance. We apply a coarse-to-fine strategy along the depth axis. Thus, the mapping is divided into a

fixed band module and a narrow band module. The fixed band module builds the cost volume \mathbf{C}_{fb} with depth labels evenly spaced in the whole depth range, while the narrow band cost volume \mathbf{C}_{nb} centers at the current depth estimation and accumulates information in a small band close to the estimate.

The fixed band module regresses an interpolation factor between the minimum and maximum depth label as output. As a consequence, the network cannot reason about the absolute scale of the scenes, which helps to make the network more flexible and generalize better. Unlike the fixed band, which contains a set of fronto-parallel planes as depth labels, the discrete labels of the narrow band are individual for each pixel. Predicting interpolation factors is not appropriate, since the network in the narrow band module has no knowledge of the band’s shape. We intentionally do not provide the narrow band network with the band shape (i.e. the depth value for which each depth label stands), because the network tends to overfit to this straight-forward cue and ignores the cost information in the cost volume. However, the absence of the band shape makes the depth regularization difficult which can be observed in Fig. 4. Therefore we append another refine network, which focuses on the problem of depth regularization. Both networks together can be understood as solving alternately the data and smoothness terms of a variational approach. The detailed architecture is shown in Fig. 3.

5.2 Training

We train our mapping networks from scratch using Adam (Kingma and Ba, 2015) and the Tensorflow framework (Abadi et al, 2015). Our training procedure consists of multiple stages. We first train the fixed band module with subsampled video sequences of length 8. Then we fix the parameters and sequentially add the two narrow band encoder-decoder pairs to the training. In the last stage we unroll the narrow band network to simulate 3 iterations and train all parts jointly. Training the mapping networks takes about 8 days in total on an NVIDIA GTX 1080Ti.

5.2.1 Datasets

We train our mapping networks on various datasets to avoid overfitting. SUN3D (Xiao et al, 2013) has a large variety of indoor scenes. For ground truth we take the improved Kinect depths with multi-frame TSDF filling. SUNCG (Song et al, 2017) is a synthetic dataset of 3D scenes with realistic scene scale. We render SUNCG to obtain a sequence of data by randomly sampling from SUN3D pose trajectories. In addition to SUNCG

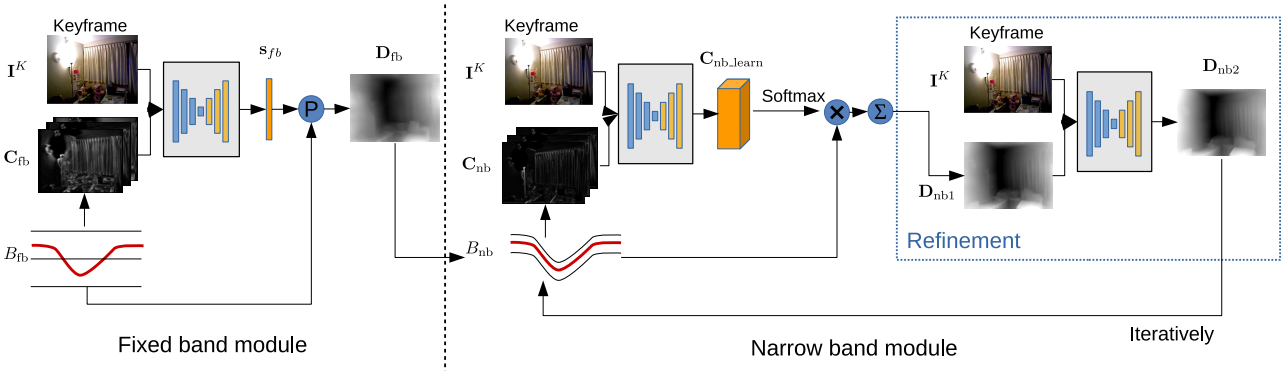


Fig. 3 Mapping networks overview. Mapping consists of a fixed band module and a narrow band module, which is based on an encoder-decoder architecture. **Fixed band module:** This module takes the keyframe image \mathbf{I}^K ($320 \times 240 \times 3$) and the cost volume \mathbf{C}_{fb} ($320 \times 240 \times 32$) generated with 32 depth labels equally spaced in the range $[0.01, 2.5]$ as inputs and outputs an interpolation factor s_{fb} ($320 \times 240 \times 1$). The fixed band depth estimation is computed as $\mathbf{D}_{fb} = (1 - s_{fb}) \cdot d_{min} + s_{fb} \cdot d_{max}$. **Narrow band module:** The narrow band module is run iteratively; in each iteration we build a cost volume \mathbf{C}_{nb} from a set of depth labels distributed around the current depth estimation with a band width σ_{nb} of 0.0125. It consists of two encoder-decoder pairs. The first pair gets the cost volume \mathbf{C}_{nb} ($320 \times 240 \times 32$) and the keyframe image \mathbf{I}^K ($320 \times 240 \times 3$) as inputs and generates a learned cost volume \mathbf{C}_{nb_learn} ($320 \times 240 \times 32$). The depth map is then obtained using a differentiable soft argmin operation (Kendall et al, 2017): $\mathbf{D}_{nb1} = \sum_{d \in \mathcal{B}_{nb}} \mathbf{B}_{nb} \times \text{softmax}(-\mathbf{C}_{nb_learn})$. The second encoder-decoder pair gets the current depth estimation \mathbf{D}_{nb1} and the keyframe image \mathbf{I}^K and produces a refined depth \mathbf{D}_{nb2} .



Fig. 4 Effects of the narrow band refinement. We apply the narrow band module for 15 iterations with and without refinement. Without the refinement, the module lacks the knowledge of the band shape and it can only make updates based on the measurements in the cost volume. This can help in capturing more details, but also causes strong artifacts. Appending a refinement network with previous depth estimation as input allows for a better regularized and more stable depth estimation.

and SUN3D, we generate a dataset—in the following called MVS—with the COLMAP structure from motion pipeline (Schönberger and Frahm, 2016; Schönberger et al, 2016). MVS contains both indoor and outdoor scenes and was captured at full image and temporal resolution ($2704 \times 1520@50\text{Hz}$) with a wide-angle GoPro camera. For training we downsample to (320×240) and use every third frame. We manually remove sequences where the reconstruction failed.

During training we use the (pseudo) ground truth camera poses from the datasets to construct the cost volume.

5.2.2 Training Objective

We use a simple L1 loss on the inverse depth maps $\mathcal{L}_{\text{depth}} = |\mathbf{D} - \mathbf{D}_{\text{gt}}|$ and the scale invariant gradient loss proposed in (Ummenhofer et al, 2017):

$$\mathcal{L}_{\text{sc-inv-grad}} = \sum_{h \in \{1, 2, 4\}} \sum_{i, j} \|\mathbf{g}_h[\mathbf{D}](i, j) - \mathbf{g}_h[\mathbf{D}_{\text{gt}}](i, j)\|_2, \quad (13)$$

where

$$\mathbf{g}_h[\mathbf{D}](i, j) = \left(\frac{\mathbf{D}(i+h, j) - \mathbf{D}(i, j)}{|\mathbf{D}(i+h, j)| + |\mathbf{D}(i, j)|}, \frac{\mathbf{D}(i, j+h) - \mathbf{D}(i, j)}{|\mathbf{D}(i, j+h)| + |\mathbf{D}(i, j)|} \right)^\top. \quad (14)$$

$\mathbf{g}_h[\mathbf{D}](i, j)$ and $\mathbf{g}_h[\mathbf{D}_{\text{gt}}](i, j)$ are gradient images of the predicted and the ground truth depth map that emphasize discontinuities. h is the step in the difference operator \mathbf{g}_h .

6 Experiments

6.1 Tracking evaluation

Tab. 1 shows the performance of our tracking network on the RGB-D benchmark (Sturm et al, 2012). The benchmark provides images and depth maps with accurate ground truth poses obtained from an external multi-camera tracking system.

We use the depth maps from the dataset during keyframe generation to measure the isolated tracking performance of our approach (left part of Tab. 1). We compare against the keyframe odometry component of the RGB-D SLAM method of Kerl et al (2013a). Their method uses the full color and depth information –for both keyframe and current frame– to compute the pose, while our method only uses the depth information from the dataset for the keyframes. During testing we generate a new keyframe if the rotational distance exceeds a threshold of 6 degrees or translational distance exceeds a 15cm threshold. The number of generated keyframes is similar to the number of keyframes reported in (Kerl et al, 2013a) for RGB-D SLAM.

Tab. 1 shows that our learning-based approach outperforms a state-of-the-art RGB-D method on most of the sequences, despite using less information. The results show also the generalization capabilities as we did not train or finetune on any sequences of the benchmark.

6.2 Mapping evaluation

For evaluating the mapping performance we use the following error metrics:

$$\text{sc-inv}(\mathbf{D}, \mathbf{D}_{\text{gt}}) = \sqrt{\frac{1}{n} \sum_{i,j} \mathbf{E}(i,j)^2 - \frac{1}{n^2} \left(\sum_{i,j} \mathbf{E}(i,j) \right)^2}, \quad (15)$$

where $\mathbf{E}(i,j) = \log \mathbf{D}(i,j) - \log \mathbf{D}_{\text{gt}}(i,j)$ and n is the number of pixels,

$$\text{L1-rel}(\mathbf{D}, \mathbf{D}_{\text{gt}}) = \frac{1}{n} \sum_i \frac{|\mathbf{D}(i,j) - \mathbf{D}_{\text{gt}}(i,j)|}{\mathbf{D}_{\text{gt}}(i,j)} \quad \text{and} \quad (16)$$

$$\text{L1-inv}(\mathbf{D}, \mathbf{D}_{\text{gt}}) = \frac{1}{n} \sum_i \left| \frac{1}{\mathbf{D}(i,j)} - \frac{1}{\mathbf{D}_{\text{gt}}(i,j)} \right|. \quad (17)$$

sc-inv is a scale invariant metric introduced in (Eigen et al, 2014). The L1-rel metric normalizes the depth error with respect to the ground truth depth value. L1-inv gives more importance to close depth values by computing the absolute difference of the reciprocal of the depth values. This metric also reflects the increasing uncertainty in the depth computation with increasing distance to the camera.

We evaluate our fixed band module and narrow band module quantitatively in Tab. 2. The results show that the fixed band module is able to exploit the accumulated information from multiple frames leading to better depth estimates. While this behaviour is taken for granted for traditional methods, this is not necessarily the case for learning-based methods. The same holds for iterative processes like the narrow band module. Running the narrow band module iteratively improves the

depth estimates. We can show this quantitatively in Tab. 2 and qualitatively in Fig. 5.

We also compare our mapping against the state-of-the-art deep learning approach DeMoN (Ummenhofer et al, 2017) and two strong classic dense mapping methods DTAM (Newcombe et al, 2011) and Semi-Global Matching (SGM) (Hirschmüller, 2005). We use the publicly available reimplementation OpenDTAM¹ and our own implementation of SGM with 16 directions. For DTAM, SGM and DeepTAM we construct a cost volume with 32 labels at the resolution of 320×240 . We use SAD as photo-consistency measure and accumulate the information of video sequences of length 10. We use the same pseudo camera pose ground truth from the datasets for a fair comparison. For DeMoN –which is limited to two images– we give the first and last frame from the sequence to provide enough motion parallax.

As shown in Tab. 2 our method achieves the best performance on all metrics and test sets. All classic methods tend to suffer from weakly-textured scenes which occur quite often in the indoor datasets and synthetic datasets. However, we are less affected by this problem by means of leveraging matching cost information together with scene priors via a neural network. This is again supported by the qualitative comparison in Fig. 6. In addition, the mapping performance of all the classic cost-volume-based methods is prone to noisy camera pose while our method is more robust, which is demonstrated in Fig. 7. More qualitative examples can be found in the supplemental video.

In the right part of Tab. 1 we compare our combined tracking and mapping against CNN-SLAM (Tateno et al, 2017) without pose graph optimization. CNN-SLAM uses a semi-dense photoconsistency optimization approach for computing camera poses and uncertainty-based depth update. We did not train on the RGB-D benchmark datasets (Sturm et al, 2012). Our learned dense tracking and mapping generalizes well and proves to be more robust and accurate on the majority of sequences. While it performs clearly worse on fr1/plant it seldomly fails and overall yields more reliable trajectories.

6.3 Ablation study

We justify our design choices with the ablation studies in Tab. 3, Tab. 4 and Fig. 9.

Tab. 3 shows results on the RGB-D benchmark for different versions of the tracking network. We test the network without the auxiliary optical flow task, without generation of multiple hypotheses and with features

¹ <https://github.com/magican/OpenDTAM.git>
SHA: 1f92a54334c233f9c4ce7d8cbaf9a81dee5e69a6

Table 1 Evaluation of our tracking (left part) and the combined mapping and tracking (right part) on the validation sets of RGB-D benchmark (Sturm et al, 2012).

Sequence	Tracking		Tracking and mapping	
	RGB-D SLAM Kerl et al (2013a)	Ours	CNN-SLAM* Tateno et al (2017)	Ours
fr1/360	0.125	0.054	0.500	0.116
fr1/desk	0.037	0.027	0.095	0.078
fr1/desk2	0.020	0.017	0.115	0.055
fr1/plant	0.062	0.057	0.150	0.165
fr1/room	0.042	0.039	0.445	0.084
fr1/rpy	0.082	0.065	0.261	0.052
fr1/xzy	0.051	0.019	0.206	0.054
average	0.060	0.040	0.253	0.086

The values describe the translational RMSE in $[m/s]$. **Tracking:** We compare the performance of our tracking network against the RGB-D SLAM method of Kerl et al (2013a). Numbers for (Kerl et al, 2013a) correspond to the frame-to-keyframe odometry evaluation and have been copied from their paper. Kerl et al (2013a) use the camera image *and* the depth stream for computing the poses, while our approach uses the depth stream only for keyframes and is limited to photometric alignment. **Ours** uses optical flow to learn motion features and predicts multiple pose hypotheses. We use the mean as the final pose estimate. **Tracking and mapping:** We compare our tracking and mapping against CNN-SLAM by Tateno et al (2017). * For a fair comparison CNN-SLAM is run without pose graph optimization. To avoid a bias in the initialization **Ours** uses the depth prediction from CNN-SLAM for the first frame of each sequence and then switches to our combined tracking and mapping.

Table 2 Keyframe depth map errors on the test split of our training data sets.

		Fixed band			Narrow band			Mapping comparison			
		2frames	6frames	10frames	1iter	3iters	5iters	SGM	DTAM	DeMoN	Ours
SUN3D	L1-inv	0.117	0.085	0.083	0.076	0.065	0.064	0.210	0.197	-	0.064
	L1-rel	0.239	0.163	0.159	0.142	0.113	0.111	0.423	0.412	-	0.111
	sc-inv	0.193	0.160	0.159	0.156	0.132	0.130	0.374	0.340	0.146	0.130
MVS	L1-inv	0.075	0.065	0.067	0.048	0.033	0.029	0.086	0.059	-	0.029
	L1-rel	0.439	0.418	0.423	0.329	0.200	0.154	0.557	0.240	-	0.154
	sc-inv	0.213	0.199	0.200	0.182	0.144	0.135	0.305	0.246	0.251	0.135
SUNCG	L1-inv	0.097	0.067	0.065	0.050	0.035	0.036	0.142	0.169	-	0.036
	L1-rel	0.288	0.198	0.193	0.141	0.082	0.083	0.380	0.533	-	0.083
	sc-inv	0.206	0.174	0.172	0.155	0.125	0.128	0.343	0.383	0.248	0.128

Fixed band: The influence of the number of frames used for computing the cost volume for the fixed band module. Accumulating information from multiple frames improves the performance and saturates after adding six or more frames. **Narrow band:** The effect of different number of iterations of the narrow band module. More iterations lead to more accurate depth maps. Depth estimations converge after about three iterations and improve only slowly with more iterations. On SUN3D results get slightly worse with more than three iterations. The narrow band width σ_{nb} is a constant number, which can be replaced by a gradually decreasing strategy or optimally by the uncertainty of the depth estimation. **Mapping comparison:** Quantitative comparison to other learning- and cost-volume-based dense mapping methods. We evaluate sequences of length 10 from our test sets and use the camera poses from the datasets to measure the isolated performance of our mapping. DeMoN just uses two input images (first and last frame of each sequence) and does not use the pose as input. Since DeMoN predicts the depth scaled with respect to its motion prediction, we compare only on the scale invariant metric sc-inv. SUNCG and SUN3D feature a large number of indoor scenes with low texture, while MVS contains a mixture of indoor and outdoor scenes and provides more texture. Our method outperforms the baselines on all datasets. The margin is especially large on the very difficult indoor datasets (SUNCG, SUN3D).

from the bottleneck of the encoder-decoder for pose hypotheses generation. The results show that attaching the pose generation branch near the end of the decoder as shown in Fig. 1 yields more accurate poses compared to features from the bottleneck and adding an auxiliary optical flow task helps to learn useful features for pose estimation. They also show that forcing the network

to predict multiple pose hypotheses further reduces the translational drift on most sequences. We further examine the covariance matrix from the multiple hypotheses at different training steps in Fig. 8.

We examine the influence of the inputs to the mapping module and show the results in Tab. 4. To identify the importance of each input we feed zeros instead of

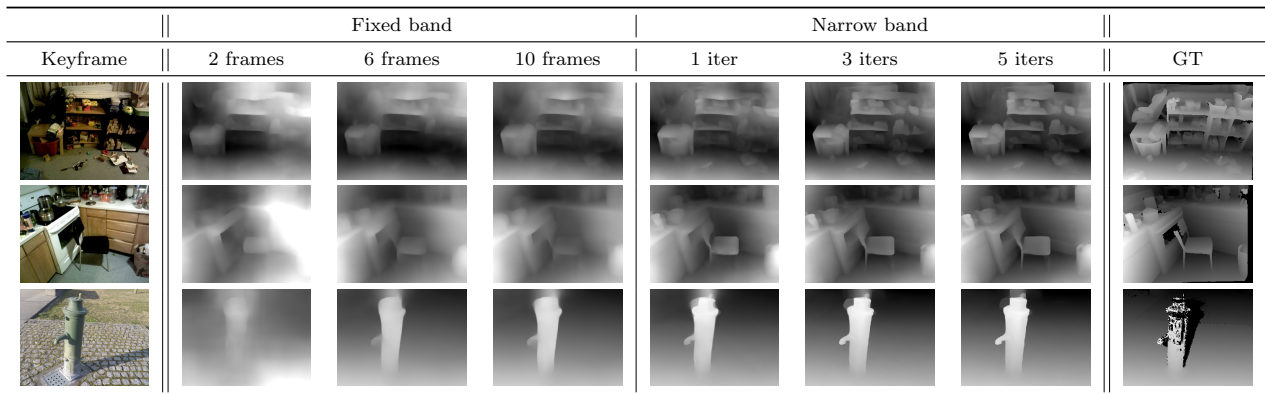


Fig. 5 Qualitative comparison of the depth prediction of the fixed band and narrow band module. We evaluate the effect of different numbers of frames used in the fixed band module and iterations used in the narrow band module. The fixed band gains in performance with more frames. The largest improvement can be observed between using only 2 frames (including keyframe) and 6 frames. The performance saturates with more frames. To further improve the quality of the depth map we use the iterative narrow band module on the 10 frames result of the fixed band. Using a narrow band around the previous depth estimation allows us to capture finer details and achieve higher accuracy.

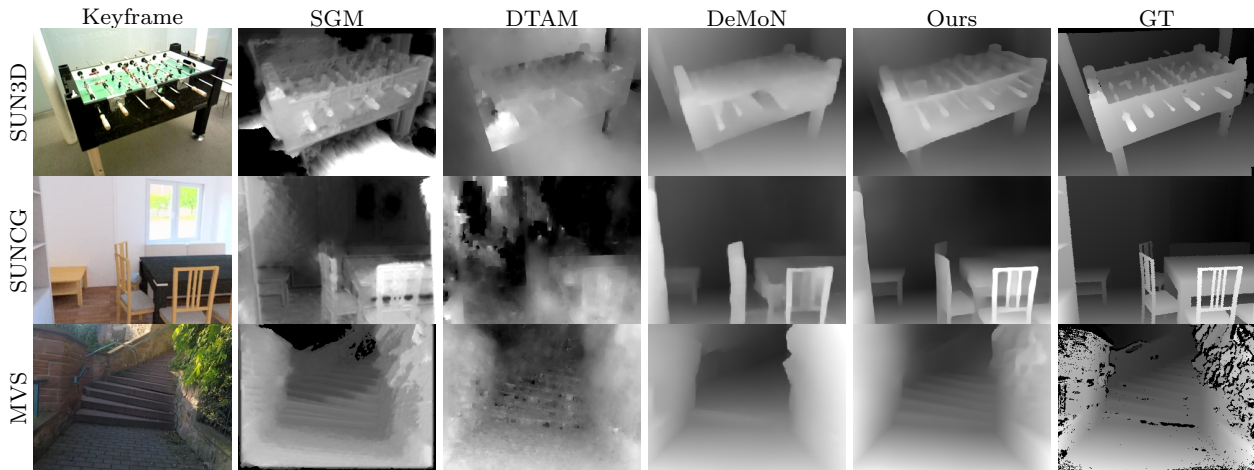


Fig. 6 Qualitative depth prediction comparison for sequences with 10 frames. DTAM has problems with short sequences and textureless scenes. SGM shares the same problems but works reasonably well if enough texture is present. DeMoN work well even in homogeneous image regions but misses many details. Our method can produce high quality depth maps using a small number of frames and captures more details compared to the other methods.

the respective input to the module and run the test for sequences of length 10 with ground truth poses. The results show that our architecture combines information from both inputs (keyframe image and cost volume). While the cost volume allows to measure depth, the keyframe image provides scene priors that improve the regularization. We show qualitative examples in Fig. 9.

6.4 Cross-dataset evaluation

We evaluate the effects of training our system on different datasets in Tab. 5 and Tab. 7. We also compare generalization performance to KITTI (Geiger et al, 2012) in Tab. 6 and Fig. 10.

Tab. 5 shows the results from a series of tracking cross-dataset experiments to assess the effects of training with and without synthetic data. Purely training on the synthetic dataset already allows us to outperform a classical baseline. By combining synthetic and real datasets, we can achieve even better performance. For the mapping module we show results of generalization from synthetic to real data in Tab. 7. We compare the fixed band module, which computes the initial depth estimate, with a single image baseline and show that making use of multiple frames improves generalization.

In Tab. 6 we show cross-dataset evaluations on KITTI for the mapping module. We also show a qualitative comparison in Fig. 10 and a failure example in Fig. 11. Although KITTI is significantly different from SUN3D

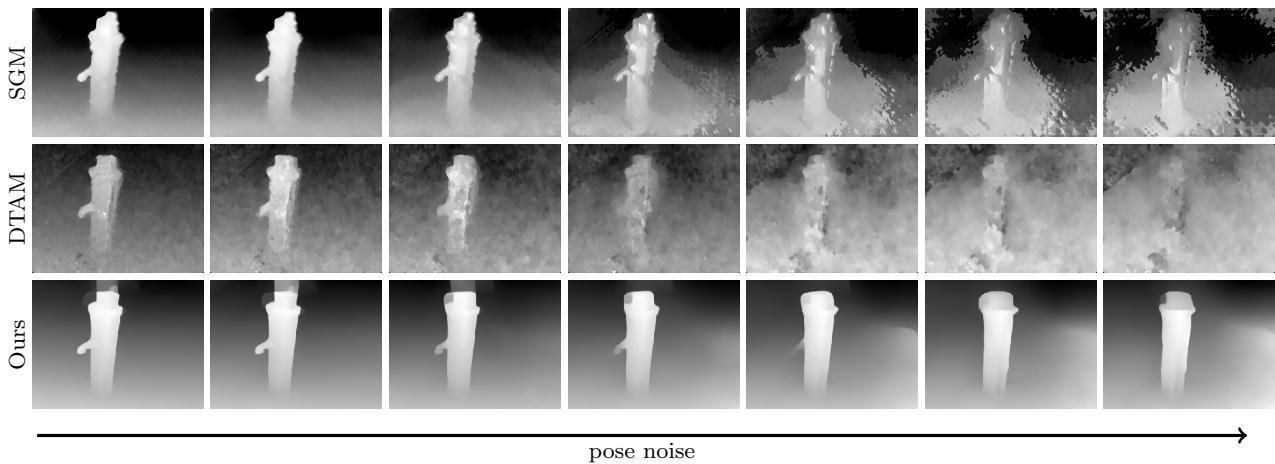


Fig. 7 Qualitative depth prediction comparison of DeepTAM, SGM, DTAM against increasing pose noise. We carefully select a well textured video sequence with 10 frames and enough motion parallax. For SGM and DTAM we use a cost volume with 64 labels, while we use 32 labels for DeepTAM. We found that using 64 instead 32 labels improves the results for both baseline methods. We apply the same normal-distributed noise vectors for all methods to the camera poses and increase the standard deviation from 0 (leftmost) to $0.6|\xi|$ (rightmost). SGM and DTAM are highly sensitive to noise and their performance degrades quickly. Our predicted depth preserves the important scene structures even under large amounts of noise. This behaviour is advantageous during tracking and improves the robustness of the overall system.

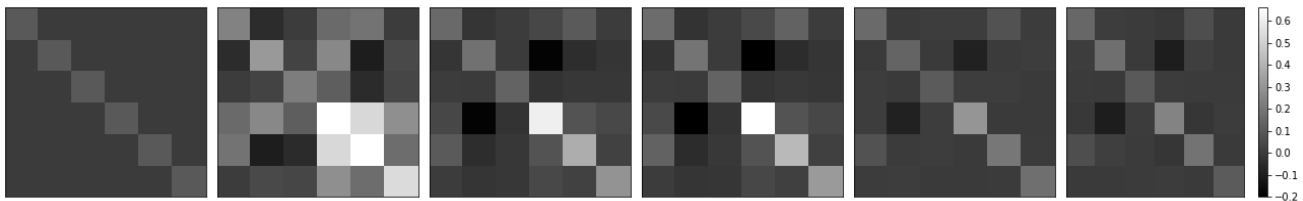


Fig. 8 Visualization of the regularized pose covariance matrices Σ at different training steps (5k, 11k, 15k, 20k, 60k and 300k) for tracking net 3. For the first 10k iterations, Σ mainly stays at its initialization state. After the network can give reasonable poses, the Σ starts to adapt. With more training, the pose estimation gets better and the absolute values of the elements in the covariance matrices become smaller. Although we use the normal distribution to sample poses for the virtual keyframe, the pose hypotheses capture the typical confusions of egomotion estimation from images. For instance, the covariance values $\Sigma_{1,5}$ and $\Sigma_{2,4}$ indicate that a rotation about the x-axis can be confused with a translation in y-direction and a negative rotation about the y-axis can be confused with a translation in x-direction, respectively. As a consequence $\mathcal{L}_{\text{uncertainty}}$ (7) pushes the network to improve the pose estimates accordingly.

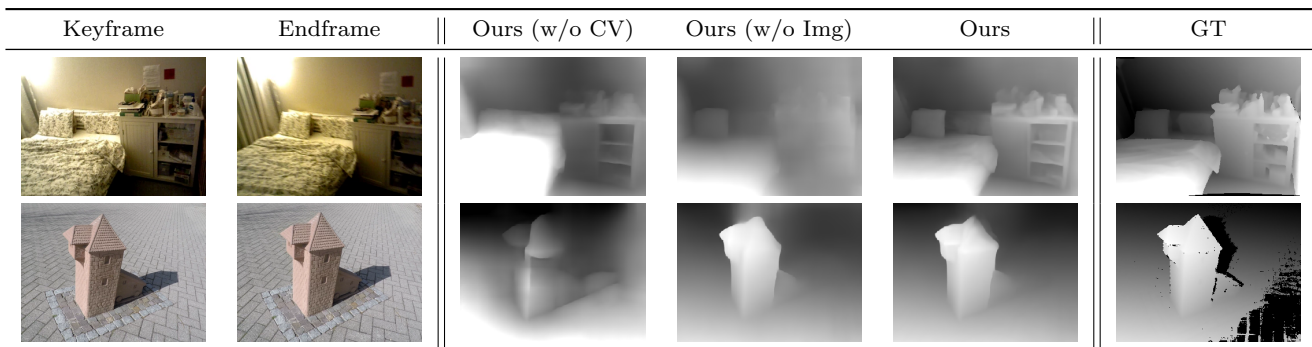


Fig. 9 Qualitative depth prediction comparison with different inputs. The first row shows an example from the SUN3D test split, while the second row is a completely different recording which was taken outdoors by myself. **Ours (w/o CV)** only uses the keyframe image and therefore cannot measure the depth and must rely on learned priors. The depth estimation is not accurate (e.g. the bed is estimated too close to the camera in the first example) or even completely fails when it comes to unseen scenes as can be seen in the example in the second row. **Ours (w/o Img)** takes the cost volume as input but not the keyframe image. It generalizes better but tends to produce blurry depth maps when the image quality is bad (see the strongly blurred depth map for the first example). **Ours** combines the advantages of both inputs and gives a robust and accurate depth estimation.

Table 3 Evaluation of different versions of our tracking network on the validation sets of the RGB-D benchmark (Sturm et al, 2012).

Sequence	Tracking Architecture			
	Ours (bottleneck features)	Ours (w/o flow)	Ours (w/o hypotheses)	Ours
fr1/360	0.064	0.069	0.065	0.054
fr1/desk	0.038	0.042	0.031	0.027
fr1/desk2	0.022	0.025	0.020	0.017
fr1/plant	0.062	0.063	0.060	0.057
fr1/room	0.051	0.051	0.041	0.039
fr1/rpy	0.078	0.070	0.063	0.065
fr1/xzy	0.024	0.030	0.021	0.019
average	0.049	0.050	0.043	0.040

The values describe the translational RMSE in $[m/s]$. **Ours (bottleneck features)** uses features from the bottleneck instead of the end of the decoder for generating the motion hypotheses. **Ours (w/o flow)** does not learn optical flow. **Ours (w/o hypotheses)** is a network which just predicts a single pose. **Ours** uses features from the decoder, uses optical flow to learn motion features and predicts multiple pose hypotheses.

Table 4 Evaluation of the importance of our mapping module’s inputs.

	SUN3D			MVS		
	L1-inv	L1-rel	sc-inv	L1-inv	L1-rel	sc-inv
Ours	0.064	0.111	0.130	0.029	0.154	0.135
Ours (w/o Img)	0.076	0.135	0.158	0.038	0.219	0.172
Ours (w/o CV)	0.150	0.304	0.208	0.222	0.842	0.266

Ours takes both the cost volume and the keyframe image as inputs and yields the best performance. In **Ours (w/o Img)**, the network is still able to measure the depth given only the cost volume. A small decrease of performance can be observed due to possible missing measurements in the cost volume and the lack of the information from the keyframe image. **Ours (w/o CV)** only makes use of the keyframe image and relies on the learned scene priors. Without the cost volume it is difficult to estimate the depth accurately, as a consequence **Ours(w/o CV)** produces the largest error.

our mapping module generalizes well compared to other learning and classical baselines.

7 Conclusions

We propose a novel deep learning architecture for real-time dense mapping and tracking. For tracking we show that generating synthetic viewpoints allows us to track incrementally with respect to a keyframe. For mapping, our methods can effectively exploit the cost volume information and image-based priors leading to accurate and robust dense depth estimations. We demonstrate that our methods outperform strong classic and deep

Table 5 Evaluation of the influence of synthetic training data for our tracking on the RGB-D benchmark (Sturm et al, 2012).

Sequence	RGB-D SLAM	Ours	Ours
	Kerl	SUNCG	SUNCG+SUN3D
fr1/360	0.119	0.072	0.061
fr1/desk	0.030	0.042	0.038
fr1/desk2	0.055	0.052	0.051
fr1/floor	0.090	0.103	0.079
fr1/plant	0.036	0.031	0.027
fr1/room	0.048	0.045	0.044
fr1/rpy	0.043	0.060	0.058
fr1/teddy	0.067	0.074	0.062
fr1/xzy	0.024	0.018	0.016
average	0.057	0.052	0.048

The values describe the translational RMSE in $[m/s]$. We compare the performance of our tracking network with different combinations of training data against the RGB-D SLAM method of Kerl et al (2013a). Results of RGB-D SLAM are taken from their paper. **Ours (SUNCG)** is the model that we train only with the synthetic dataset SUNCG (Song et al, 2017), while for **Ours (SUNCG + SUN3D)** we use a combination of the synthetic dataset SUNCG and the real dataset SUN3D (Xiao et al, 2013).

Table 6 Cross-dataset evaluation of the depth estimation.

	SUN3D → KITTI					
	Sequence 0			Sequence 10		
	L1-inv	L1-rel	sc-inv	L1-inv	L1-rel	sc-inv
Single	0.387	7.612	0.603	0.425	7.940	0.635
SGM	0.027	0.282	0.695	0.034	0.320	0.740
DeMoN	-	-	0.441	-	-	0.513
Ours	0.019	0.265	0.376	0.028	0.339	0.452

All methods are tested on KITTI without finetuning. We randomly choose sequence 0 and 10 for evaluation. **Single** is a basic encoder-decoder-based network which predicts depth from just a single image. For a fair comparison, this baseline is also trained on SUN3D as **Ours**. **SGM** and **Ours** use a sequence of 5 frames with ground truth poses, while **DeMoN** only uses the first and last frame. KITTI is an urban scene dataset where the depth values are on average a lot larger than on the SUN3D indoor dataset. The single image baseline does not adapt to the different depth range, which results in very large errors for the L1-inv and L1-rel metrics. Our mapping generalizes much better from SUN3D to KITTI.

learning algorithms. In future work, we plan to extend the presented components to build a full SLAM system.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah

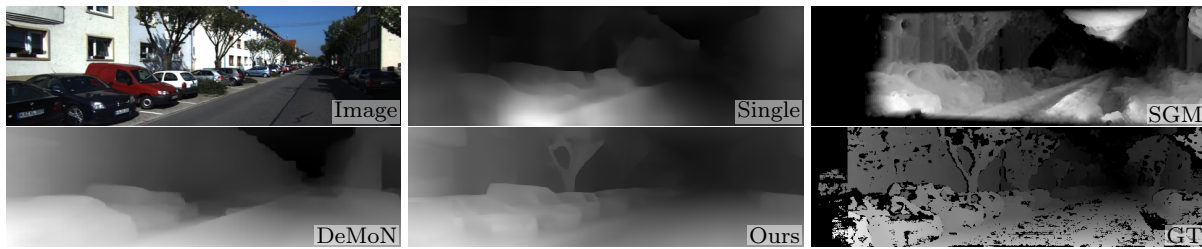


Fig. 10 Generalization experiment on KITTI (Geiger et al, 2012). SGM, DTAM and Ours use a sequence of 5 frames from the left color camera, while for DeMoN we only use the first and last frame of each sequence. Single is an encoder-decoder-based network that only gets single image as input and is trained on the same datasets as Ours. We show pseudo GT as a reference, which was obtained by computing the disparity of the corresponding rectified and synchronized stereo pairs. KITTI is an urban scene dataset captured with a wide-angle camera, which differs from our training data significantly. Further, due to the dominant forward motion pattern of the dataset the epipole is within the visible image borders, which makes depth estimation especially difficult. Without finetuning our method generalizes well to this dataset. More examples can be found in the supplementary material.



Fig. 11 Failure example on KITTI. Ours uses a sequence of 5 frames from the left color camera; the first and last frame are shown above. Our cost-volume-based approach suffers from the moving object (the silver bus in this example) and the occlusion (trees occluded by the bus).

Table 7 Evaluation of the generalization ability of the fixed band mapping module from synthetic to real data.

	SUN3D \rightarrow SUN3D			SUNCG \rightarrow SUN3D			Err. inc. Avg
	L1-inv	L1-rel	sc-inv	L1-inv	L1-rel	sc-inv	
Single	0.109	0.221	0.143	0.158	0.301	0.261	54.5%
Ours(fb)	0.083	0.158	0.159	0.096	0.171	0.201	16.7%

Single is a depth estimation network using just a single image. **Ours(fb)** is the fixed band module of our mapping part, which generates the initial depth estimates in our system. It is comparable in size to the model of the single image network. We evaluate both on the test split of the SUN3D dataset. We compare the depth estimation errors of both approaches trained on different data. Using only the synthetic SUNCG dataset for training obviously decreases the performance for both methods. However, the performance drop of **Ours(fb)** is significantly smaller than for the single image approach (see the average increase of the error in the last column). This demonstrates the robustness and generalization ability of the architecture of our fixed band module.

C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org 5, 6

Agrawal P, Carreira J, Malik J (2015) Learning to See by Moving. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp 37–45, DOI 10.1109/ICCV.2015.13 2

Collins RT (1996) A space-sweep approach to true multi-image matching. In: Proceedings CVPR IEEE

Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, pp 358–363, DOI 10.1109/CVPR.1996.517097 2

Dhiman V, Tran QH, Corso JJ, Chandraker M (2016) A Continuous Occlusion Model for Road Scene Understanding. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4331–4339, DOI 10.1109/CVPR.2016.469 2

Eigen D, Puhrsch C, Fergus R (2014) Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) Advances in Neural Information Processing Systems 27, Curran Associates, Inc., pp 2366–2374 8

Engel J, Schöps T, Cremers D (2014) LSD-SLAM: Large-scale direct monocular SLAM. In: European Conference on Computer Vision, Springer, pp 834–849 3

Engel J, Koltun V, Cremers D (2018) Direct Sparse Odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence 40(3):611–625, DOI 10.1109/TPAMI.2017.2658577 3

Fattal R (2008) Single image dehazing. In: ACM SIGGRAPH 2008 Papers, ACM, New York, NY, USA, SIGGRAPH '08, pp 72:1–72:9, DOI 10.1145/1399504.1360671 2

Felzenszwalb PF, Huttenlocher DP (2006) Efficient Belief Propagation for Early Vision. International Journal of Computer Vision 70(1):41–54, DOI 10.1007/s11263-006-7899-4 6

- Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference On*, IEEE, pp 3354–3361 [2](#), [3](#), [4](#), [10](#), [13](#)
- Gupta S, Arbellez P, Malik J (2013) Perceptual organization and recognition of indoor scenes from rgb-d images. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp 564–571, DOI 10.1109/CVPR.2013.79 [2](#)
- Gupta S, Arbelález P, Girshick R, Malik J (2015) Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation. *Int J Comput Vision* 112(2):133–149, DOI 10.1007/s11263-014-0777-6 [2](#)
- Hirschmüller H (2005) Accurate and efficient stereo processing by semi-global matching and mutual information. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol 2, pp 807–814 vol. 2, DOI 10.1109/CVPR.2005.56 [6](#), [8](#)
- Hosni A, Rhemann C, Bleyer M, Rother C, Gelautz M (2013) Fast Cost-Volume Filtering for Visual Correspondence and Beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(2):504–511, DOI 10.1109/TPAMI.2012.156 [6](#)
- Kendall A, Cipolla R (2017) Geometric loss functions for camera pose regression with deep learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* [2](#)
- Kendall A, Martirosyan H, Dasgupta S, Henry P (2017) End-to-End Learning of Geometry and Context for Deep Stereo Regression. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp 66–75, DOI 10.1109/ICCV.2017.17 [7](#)
- Kerl C, Sturm J, Cremers D (2013a) Dense visual SLAM for RGB-D cameras. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp 2100–2106, DOI 10.1109/IROS.2013.6696650 [8](#), [9](#), [12](#)
- Kerl C, Sturm J, Cremers D (2013b) Robust odometry estimation for RGB-D cameras. In: *2013 IEEE International Conference on Robotics and Automation*, pp 3748–3754, DOI 10.1109/ICRA.2013.6631104 [3](#)
- Kingma DP, Ba J (2015) Adam: A Method for Stochastic Optimization. In: *Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* [5](#), [6](#)
- Klein G, Murray D (2007) Parallel Tracking and Mapping for Small {AR} Workspaces. In: *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)* [3](#)
- Li R, Wang S, Long Z, Gu D (2018) UnDeepVO: Monocular Visual Odometry Through Unsupervised Deep Learning. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp 7286–7291, DOI 10.1109/ICRA.2018.8461251 [2](#)
- Loshchilov I, Hutter F (2017) SGDR: Stochastic Gradient Descent with Warm Restarts. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net [5](#)
- Lowe DG (2004) Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60(2):91–110, DOI 10.1023/B:VISI.0000029664.99615.94 [2](#)
- Newcombe RA, Lovegrove S, Davison A (2011) DTAM: Dense tracking and mapping in real-time. In: *2011 IEEE International Conference on Computer Vision (ICCV)*, pp 2320–2327, DOI 10.1109/ICCV.2011.6126513 [2](#), [3](#), [8](#)
- Schönberger JL, Frahm JM (2016) Structure-from-Motion Revisited. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 4104–4113, DOI 10.1109/CVPR.2016.445 [7](#)
- Schönberger JL, Zheng E, Frahm JM, Pollefeys M (2016) Pixelwise View Selection for Unstructured Multi-View Stereo. In: *Computer Vision – ECCV 2016*, Springer International Publishing, pp 501–518, DOI 10.1007/978-3-319-46487-9_31 [7](#)
- Song S, Chandraker M (2015) Joint SFM and detection cues for monocular 3D localization in road scenes. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 3734–3742, DOI 10.1109/CVPR.2015.7298997 [2](#)
- Song S, Yu F, Zeng A, Chang AX, Savva M, Funkhouser T (2017) Semantic Scene Completion from a Single Depth Image. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 190–198, DOI 10.1109/CVPR.2017.28 [4](#), [6](#), [12](#)
- Sturm J, Engelhard N, Endres F, Burgard W, Cremers D (2012) A benchmark for the evaluation of RGB-D SLAM systems. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp 573–580, DOI 10.1109/IROS.2012.6385773 [7](#), [8](#), [9](#), [12](#)
- Tateno K, Tombari F, Laina I, Navab N (2017) CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 6565–6574, DOI 10.1109/CVPR.2017.695 [3](#), [8](#), [9](#)
- Ummenhofer B, Zhou H, Uhrig J, Mayer N, Ilg E, Dosovitskiy A, Brox T (2017) DeMoN: Depth and Motion Network for Learning Monocular Stereo. In: *IEEE*

- Conference on Computer Vision and Pattern Recognition (CVPR) [1](#), [2](#), [7](#), [8](#)
- Valada A, Radwan N, Burgard W (2018) Deep Auxiliary Learning for Visual Localization and Odometry. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp 6939–6946, DOI [10.1109/ICRA.2018.8462979](#) [2](#)
- Vijayanarasimhan S, Ricco S, Schmid C, Sukthankar R, Fragkiadaki K (2017) SfM-Net: Learning of Structure and Motion from Video. arXiv:170407804 [cs] [1704.07804](#) [2](#)
- Wang S, Clark R, Wen H, Trigoni N (2017) DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp 2043–2050, DOI [10.1109/ICRA.2017.7989236](#) [2](#)
- Weerasekera CS, Latif Y, Garg R, Reid I (2017) Dense monocular reconstruction using surface normals. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp 2524–2531, DOI [10.1109/ICRA.2017.7989293](#) [2](#)
- Weerasekera CS, Garg R, Latif Y, Reid I (2018) Learning Deeply Supervised Good Features to Match for Dense Monocular Reconstruction. In: Computer Vision – ACCV 2018, Springer, Cham, pp 609–624, DOI [10.1007/978-3-030-20873-8_39](#) [2](#)
- Xiao J, Owens A, Torralba A (2013) SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In: 2013 IEEE International Conference on Computer Vision (ICCV), pp 1625–1632, DOI [10.1109/ICCV.2013.458](#) [4](#), [6](#), [12](#)
- Zhan H, Garg R, Saroj Weerasekera C, Li K, Agarwal H, Reid I (2018) Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [2](#)
- Zhang H, Patel VM (2018a) Densely connected pyramid dehazing network. In: CVPR [2](#)
- Zhang H, Patel VM (2018b) Density-aware single image de-raining using a multi-stream dense network. In: CVPR [2](#)
- Zhou H, Ummenhofer B, Brox T (2018) Deeptam: Deep tracking and mapping. In: European Conference on Computer Vision (ECCV) [3](#)
- Zhou T, Brown M, Snavely N, Lowe DG (2017) Unsupervised Learning of Depth and Ego-Motion from Video. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 6612–6619, DOI [10.1109/CVPR.2017.700](#) [2](#)