

# DeepTAM: Deep Tracking and Mapping

Huizhong Zhou\* Benjamin Ummenhofer\* Thomas Brox

University of Freiburg  
{zhouh, ummenhof, brox}@cs.uni-freiburg.de

**Abstract.** We present a system for keyframe-based dense 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 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.

**Keywords:** Camera tracking, Multi-view stereo, ConvNets

## 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 [30] 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.

---

\*Equal contribution

- 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 [23], 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 [3]. 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 [26,4], scene understanding [11,12], image dehazing [8,36,35].

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 [22] 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 [23]. 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 [30], 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. Agrawal *et al.* [2] trains a neural network to estimate the egomotion, which mainly serves as a supervision for feature learning. Kendall *et al.* [16] apply deep learning to the camera localization task and Valada *et al.* [31] show that the visual localization and odometry can be solved jointly within one network. DeepVO [33] 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.* [37] 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 [32], too, uses unsupervised learning ideas, and (despite its title) does not use the motion parallax for depth estimation. UnDeepVO [20] 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 [37,33,20] focus on the KITTI dataset [10]. 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 [23] that are worth mentioning. LSD-SLAM [7] 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.* [6] 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.* [18] carefully model the noise distribution to improve robustness. Again, this comes for free in a learning-based approach. CNN-SLAM [29] 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.

### 3 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  $4 \times 4$  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 [23], 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  $f$ .

### 3.1 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.

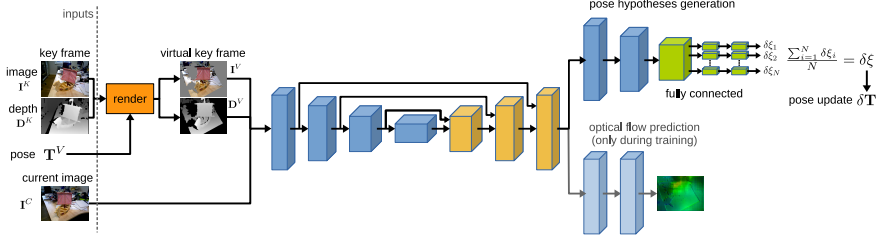
Since camera motion can only be estimated by relating the keyframe to the current image, we make use of optical flow as an auxiliary task. The predicted optical flow ensures that the network learns to exploit the relationship between both frames. We demonstrate the importance of the flow prediction in 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 the 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

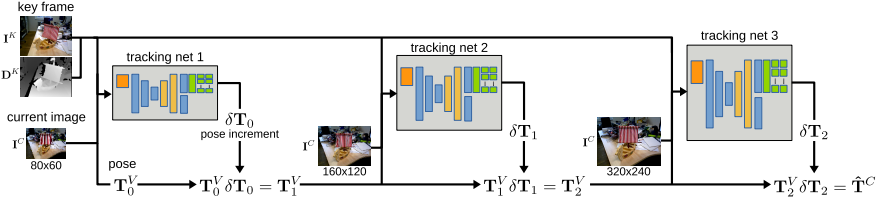
$$\delta\xi = \frac{1}{N} \sum_{i=1}^{N=64} \delta\xi_i. \quad (3)$$

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.





**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, which then splits into  $N = 64$  fully connected branches sharing 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 ( $80 \times 60$ ), ( $160 \times 120$ ) and ( $320 \times 240$ ). 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  $\delta\mathbf{T}_i$ .

### 3.2 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 [10], 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.

**Datasets** We train on image pairs from the SUN3D dataset [34] and the SUNCG dataset [27]. 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.

**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:

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

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

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  $\alpha$  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  $\Sigma$  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. The function  $K_v$  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  $\delta\xi_i$ .

We optimize using Adam [19] with the learning rate schedule proposed in [21]. We implement and train the networks with Tensorflow [1]. Training the tracking network takes less than a day on an NVIDIA GTX1080Ti. We provide the detailed training parameters in the supplementary material.

## 4 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 photoconsistency 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 photoconsistency 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  $3 \times 3$  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  $\mathcal{W}(\mathbf{I}_i, \mathbf{T}_i(\mathbf{T}^K)^{-1}, d)$ , 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

$$w_i(\mathbf{x}) = 1 - \frac{1}{N-1} \sum_{d \in B_{\text{fb}} \setminus \{d^*\}} \exp\left(-\alpha \cdot (\rho_i(\mathbf{x}, d) - \rho_i(\mathbf{x}, d^*))^2\right). \quad (10)$$

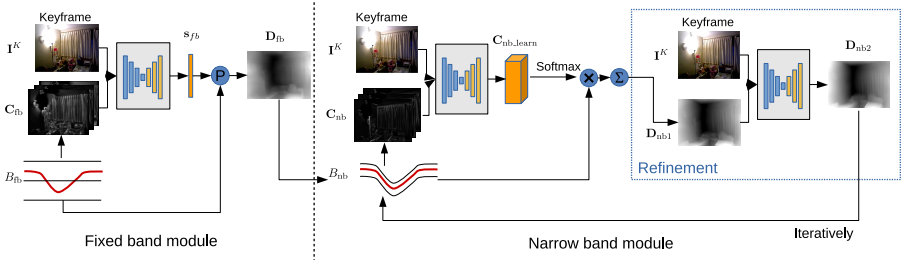
$w_i$  describes the matching confidence and is close to 1 if there is a clear and unique minimum  $\rho_i(\mathbf{x}, d^*)$  with  $d^* = \arg \min_d \rho_i(\mathbf{x}, d)$ .

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 [13,9,14] 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  $d_{\text{prev}}$  as

$$B_{\text{nb}} = \{b_i | b_i = d_{\text{prev}} + i \cdot \sigma_{\text{nb}} \cdot d_{\text{prev}}, i = -\frac{N}{2}, \dots, \frac{N-2}{2}\}. \quad (11)$$

$\sigma_{\text{nb}}$  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.



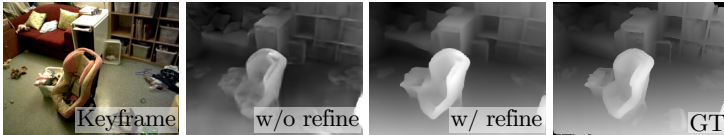
**Fig. 3.** Mapping networks overview. Mapping consists of a fixed band module and a narrow band module, which is based on an encoder-decoder architecture. **Fixed band module:** This module takes the keyframe image  $\mathbf{I}^K$  ( $320 \times 240 \times 3$ ) and the cost volume  $\mathbf{C}_{fb}$  ( $320 \times 240 \times 32$ ) 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  $\sigma_{nb}$  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 [15]:  $\mathbf{D}_{nb1} = \sum_{d \in \mathbf{B}_{nb}} \mathbf{B}_{nb} \times \text{softmax}(-\mathbf{C}_{nb,learn})$ . 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  $\mathbf{D}_{nb2}$ .

#### 4.1 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.



**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.

## 4.2 Training

We train our mapping networks from scratch using Adam [19] based on the Tensorflow [1] framework. 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.

**Datasets** We train our mapping networks on various datasets to avoid overfitting. SUN3D [34] has a large variety of indoor scenes. For ground truth we take the improved Kinect depths with multi-frame TSDF filling. SUNCG [27] 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 and SUN3D, we generate a dataset –in the following called MVS– with the COLMAP structure from motion pipeline [24,25]. 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  $(320 \times 240)$  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.

**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 [30]:

$$\mathcal{L}_{\text{sc-inv-grad}} = \sum_{h \in \{1,2,4\}} \sum_{i,j} \|\mathbf{g}_h[\mathbf{D}](i,j) - \mathbf{g}_h[\mathbf{D}_{\text{gt}}](i,j)\|_2, \quad (12)$$

where

$$\mathbf{g}_h[\mathbf{D}](i, j) = \left( \frac{\mathbf{D}(i+h, j) - \mathbf{D}(i, j)}{|\mathbf{D}(i+h, j)| + |\mathbf{D}(i, j)|}, \frac{\mathbf{D}(i, j+h) - \mathbf{D}(i, j)}{|\mathbf{D}(i, j+h)| + |\mathbf{D}(i, j)|} \right)^\top. \quad (13)$$

$\mathbf{g}_h[\mathbf{D}](i, j)$  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  $\mathbf{g}_h$ .

## 5 Experiments

### 5.1 Tracking evaluation

Tab. 1 shows the performance of our tracking network on the RGB-D benchmark [28]. 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.* [17]. 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 [17] 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. In addition, the results also show that forcing the network to predict multiple pose hypotheses further reduces the translational drift on most sequences. The results show also the generalization capabilities as we did not train or finetune on any sequences of the benchmark.

### 5.2 Mapping evaluation

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

$$\text{sc-inv}(\mathbf{D}, \mathbf{D}_{\text{gt}}) = \sqrt{\frac{1}{n} \sum_{i,j} \mathbf{E}(i, j)^2 - \frac{1}{n^2} \left( \sum_{i,j} \mathbf{E}(i, j) \right)^2}, \quad (14)$$

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

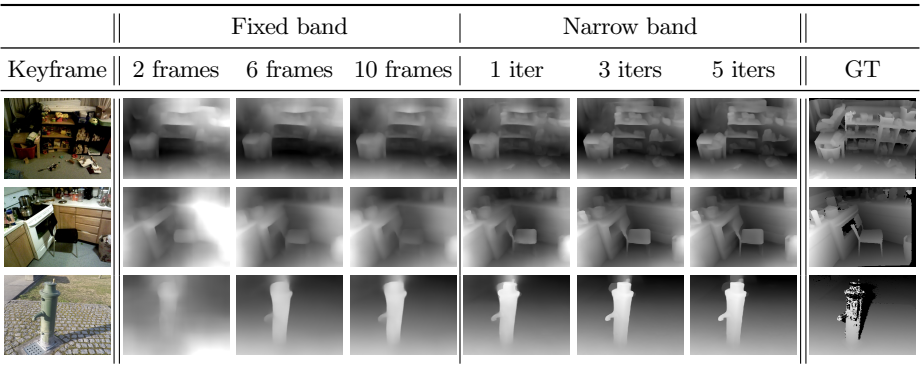
$$\text{L1-rel}(\mathbf{D}, \mathbf{D}_{\text{gt}}) = \frac{1}{n} \sum_i \frac{|\mathbf{D}(i, j) - \mathbf{D}_{\text{gt}}(i, j)|}{\mathbf{D}_{\text{gt}}(i, j)} \quad \text{and} \quad (15)$$

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

sc-inv is a scale invariant metric introduced in [5]. 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.

	Tracking				Tracking and mapping	
Sequence	RGB-D SLAM	Ours	Ours	Ours	CNN-SLAM*	Ours
	Kerl <i>et al.</i> [17]	(w/o flow)	(w/o hypotheses)		Tateno <i>et al.</i> [29]	
fr1/360	0.125	0.069	0.065	<b>0.054</b>	0.500	<b>0.116</b>
fr1/desk	0.037	0.042	0.031	<b>0.027</b>	0.095	<b>0.078</b>
fr1/desk2	0.020	0.025	0.020	<b>0.017</b>	0.115	<b>0.055</b>
fr1/plant	0.062	0.063	0.060	<b>0.057</b>	<b>0.150</b>	0.165
fr1/room	0.042	0.051	0.041	<b>0.039</b>	0.445	<b>0.084</b>
fr1/rpy	0.082	0.070	<b>0.063</b>	0.065	0.261	<b>0.052</b>
fr1/xzy	0.051	0.030	0.021	<b>0.019</b>	0.206	<b>0.054</b>
average	0.060	0.050	0.043	<b>0.040</b>	0.253	<b>0.086</b>

**Table 1.** Evaluation of our tracking (left part) and the combined mapping and tracking (right part) on the validation sets of RGB-D benchmark [28]. 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.* [17]. Numbers for Kerl *et al.* [17] correspond to the frame-to-keyframe odometry evaluation and have been copied from their paper. Kerl *et al.* [17] uses 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 (w/o flow)** does not learn optical flow. **Ours (w/o hypotheses)** is a network which just predicts a single pose. **Ours** uses optical flow to learn motion features and predicts multiple pose hypotheses. **Tracking and mapping:** We compare our tracking and mapping against CNN-SLAM by Tateno *et al.* [29]. \* 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.



**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.

		Fixed band			Narrow band			Mapping comparison			
		2frames	6frames	10frames	1iter	3iters	5iters	SGM	DTAM	DeMoN	Ours
MVS	L1-inv	0.117	0.085	<b>0.083</b>	0.076	0.065	<b>0.064</b>	-	0.086	0.059	<b>0.036</b>
	L1-rel	0.239	0.163	<b>0.159</b>	0.142	0.113	<b>0.111</b>	-	0.557	0.240	<b>0.171</b>
	sc-inv	0.193	0.160	<b>0.159</b>	0.156	0.132	<b>0.130</b>	0.251	0.305	0.246	<b>0.146</b>
SUNCG	L1-inv	0.075	<b>0.065</b>	0.067	0.049	0.039	<b>0.036</b>	-	0.142	0.169	<b>0.036</b>
	L1-rel	0.439	<b>0.418</b>	0.423	0.304	0.213	<b>0.171</b>	-	0.380	0.533	<b>0.083</b>
	sc-inv	0.213	<b>0.199</b>	0.200	0.174	0.152	<b>0.146</b>	0.248	0.343	0.383	<b>0.128</b>
SUN3D	L1-inv	0.097	0.067	<b>0.065</b>	0.050	<b>0.035</b>	0.036	-	0.210	0.197	<b>0.064</b>
	L1-rel	0.288	0.198	<b>0.193</b>	0.141	<b>0.082</b>	0.083	-	0.423	0.412	<b>0.111</b>
	sc-inv	0.206	0.174	<b>0.172</b>	0.155	<b>0.125</b>	0.128	0.146	0.374	0.340	<b>0.130</b>

**Table 2.** Keyframe depth map errors on the test split of our training data sets. **Fixed band:** The influence of the number of frame 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  $\sigma_{nb}$  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).

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 [30] and two strong classic dense mapping methods DTAM [23] and SGM [13]. We use the publicly available reimplementations OpenDTAM\* 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  $320 \times 240$ . 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

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

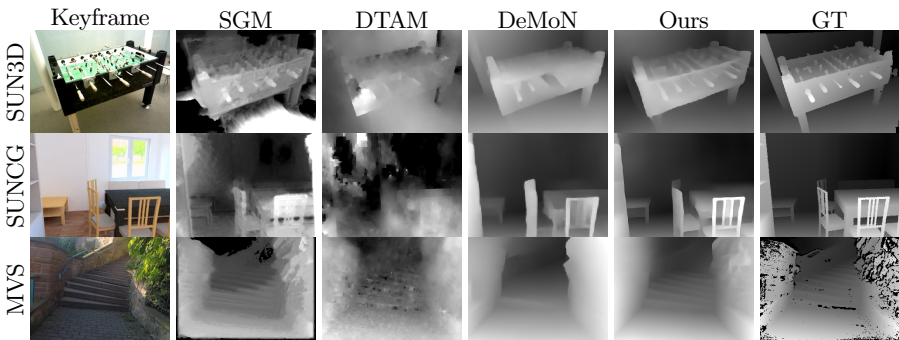


a fair comparison. For DeMoN –which is limited to two images– we give the first and last frame from the sequence to provide enough motion parallax.

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. 8. 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 [29] 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 RGB-D benchmark datasets [28]. 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 seldom fails and overall yields more reliable trajectories.

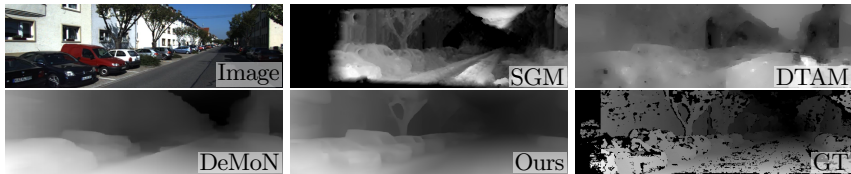
To further verify our generalization ability, we test our model on KITTI [10] without finetuning. Fig. 7 shows a qualitative comparison.



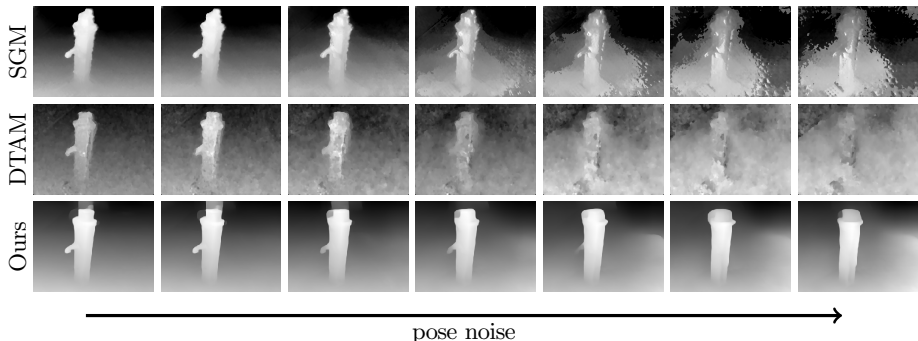
**Fig. 6.** Qualitative depth prediction comparison for sequences with 10 frames. DTAM has problems with short sequences and textureless scenes. SGM shares the same problems but works reasonably well if enough texture is present. DeMoN work well even in homogeneous image regions but misses many details.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

## 6 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



**Fig. 7.** Generalization experiment on KITTI [10]. 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. 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.



**Fig. 8.** 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.

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

**Acknowledgements** 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.

## References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., 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, 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.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015), software available from tensorflow.org
2. Agrawal, P., Carreira, J., Malik, J.: Learning to See by Moving. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 37–45 (Dec 2015). <https://doi.org/10.1109/ICCV.2015.13>
3. Collins, R.T.: A space-sweep approach to true multi-image matching. pp. 358–363. IEEE (Jun 1996). <https://doi.org/10.1109/CVPR.1996.517097>
4. Dhiman, V., Tran, Q.H., Corso, J.J., Chandraker, M.: A Continuous Occlusion Model for Road Scene Understanding. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4331–4339 (Jun 2016). <https://doi.org/10.1109/CVPR.2016.469>
5. Eigen, D., Puhrsch, C., Fergus, R.: Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. arXiv:1406.2283 [cs] (Jun 2014)
6. Engel, J., Koltun, V., Cremers, D.: Direct Sparse Odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(3), 611–625 (Mar 2018). <https://doi.org/10.1109/TPAMI.2017.2658577>
7. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: European Conference on Computer Vision. pp. 834–849. Springer (2014)
8. Fattal, R.: Single image dehazing. In: ACM SIGGRAPH 2008 Papers. pp. 72:1–72:9. SIGGRAPH '08, ACM, New York, NY, USA (2008). <https://doi.org/10.1145/1399504.1360671>, <http://doi.acm.org/10.1145/1399504.1360671>
9. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient Belief Propagation for Early Vision. International Journal of Computer Vision **70**(1), 41–54 (Oct 2006). <https://doi.org/10.1007/s11263-006-7899-4>
10. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference On. pp. 3354–3361. IEEE (2012)
11. Gupta, S., Arbellez, P., Malik, J.: Perceptual organization and recognition of indoor scenes from rgb-d images. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 564–571 (June 2013). <https://doi.org/10.1109/CVPR.2013.79>
12. Gupta, S., Arbeláez, P., Girshick, R., Malik, J.: Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation. Int. J. Comput. Vision **112**(2), 133–149 (Apr 2015). <https://doi.org/10.1007/s11263-014-0777-6>, <http://dx.doi.org/10.1007/s11263-014-0777-6>
13. Hirschmüller, H.: 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 (Jun 2005). <https://doi.org/10.1109/CVPR.2005.56>
14. Hosni, A., Rhemann, C., Bleyer, M., Rother, C., Gelautz, M.: Fast Cost-Volume Filtering for Visual Correspondence and Beyond. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(2), 504–511 (2013). <https://doi.org/10.1109/TPAMI.2012.156>

15. Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P.: End-to-End Learning of Geometry and Context for Deep Stereo Regression. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 66–75 (Oct 2017). <https://doi.org/10.1109/ICCV.2017.17>
16. Kendall, A., Cipolla, R.: Geometric loss functions for camera pose regression with deep learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017)
17. Kerl, C., Sturm, J., Cremers, D.: Dense visual SLAM for RGB-D cameras. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2100–2106 (Nov 2013). <https://doi.org/10.1109/IROS.2013.6696650>
18. Kerl, C., Sturm, J., Cremers, D.: Robust odometry estimation for RGB-D cameras. In: 2013 IEEE International Conference on Robotics and Automation. pp. 3748–3754 (May 2013). <https://doi.org/10.1109/ICRA.2013.6631104>
19. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* (Dec 2014)
20. Li, R., Wang, S., Long, Z., Gu, D.: UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. *arXiv:1709.06841 [cs]* (Sep 2017)
21. Loshchilov, I., Hutter, F.: SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv:1608.03983 [cs, math]* (Aug 2016)
22. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**(2), 91–110 (Nov 2004). <https://doi.org/10.1023/B:VISI.0000029664.99615.94>
23. Newcombe, R.A., Lovegrove, S., Davison, A.: DTAM: Dense tracking and mapping in real-time. In: 2011 IEEE International Conference on Computer Vision (ICCV). pp. 2320–2327 (2011). <https://doi.org/10.1109/ICCV.2011.6126513>
24. Schönberger, J.L., Frahm, J.M.: Structure-from-Motion Revisited. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4104–4113 (Jun 2016). <https://doi.org/10.1109/CVPR.2016.445>
25. Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise View Selection for Unstructured Multi-View Stereo. In: *Computer Vision – ECCV 2016*. pp. 501–518. Springer International Publishing (Oct 2016). [https://doi.org/10.1007/978-3-319-46487-9\\_31](https://doi.org/10.1007/978-3-319-46487-9_31)
26. Song, S., Chandraker, M.: 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 (Jun 2015). <https://doi.org/10.1109/CVPR.2015.7298997>
27. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic Scene Completion from a Single Depth Image. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 190–198 (Jul 2017). <https://doi.org/10.1109/CVPR.2017.28>
28. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 573–580 (Oct 2012). <https://doi.org/10.1109/IROS.2012.6385773>
29. Tateno, K., Tombari, F., Laina, I., Navab, N.: 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 (Jul 2017). <https://doi.org/10.1109/CVPR.2017.695>
30. Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., Brox, T.: DeMoN: Depth and Motion Network for Learning Monocular Stereo. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017)
31. Valada, A., Radwan, N., Burgard, W.: Deep Auxiliary Learning for Visual Localization and Odometry. *arXiv:1803.03642 [cs]* (Mar 2018)

32. Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., Fragkiadaki, K.: SfM-Net: Learning of Structure and Motion from Video. arXiv:1704.07804 [cs] (Apr 2017)
33. Wang, S., Clark, R., Wen, H., Trigoni, N.: 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 (May 2017). <https://doi.org/10.1109/ICRA.2017.7989236>
34. Xiao, J., Owens, A., Torralba, A.: SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In: 2013 IEEE International Conference on Computer Vision (ICCV). pp. 1625–1632 (Dec 2013). <https://doi.org/10.1109/ICCV.2013.458>
35. Zhang, H., Patel, V.M.: Densely connected pyramid dehazing network. In: CVPR (2018)
36. Zhang, H., Patel, V.M.: Density-aware single image de-raining using a multi-stream dense network. In: CVPR (2018)
37. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised Learning of Depth and Ego-Motion from Video. arXiv:1704.07813 [cs] (Apr 2017)

# DeepTAM: Deep Tracking and Mapping

## Supplementary Material

### 1 Tracking Network Implementation Details

Fig. 1 shows the operations and parameters of the ConvNet parts for the three tracking networks.

#### 1.1 Rendering

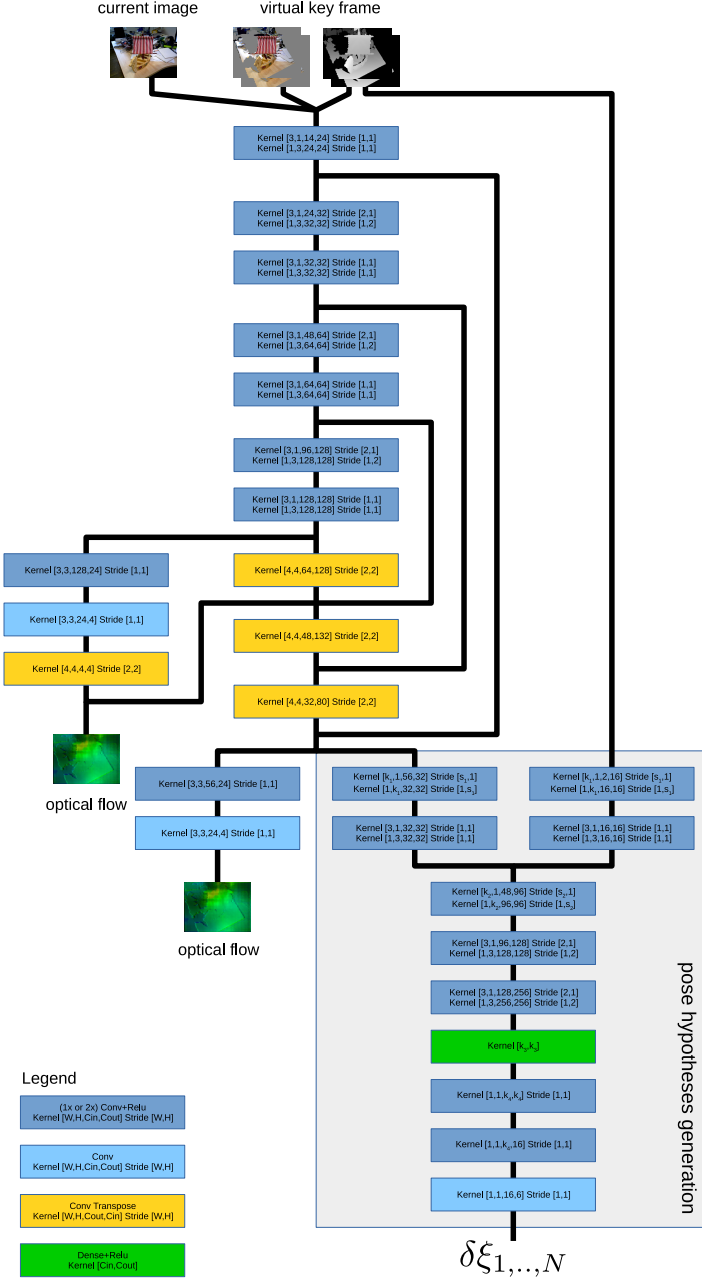
The input to the ConvNets is the current camera image and the rendered virtual keyframe. To pass as much information as possible to the ConvNet, we render two image and depth map pairs. We use the depth tests GREATER and LESS to render the images and depth maps. The depth test GREATER generates an image and an inverse depth map pair which corresponds to what can actually be seen from the specified viewpoint. The depth test LESS generates an image and depth map which shows the occluded parts (we assume that there are at most two depth layers). We use simple point-based rendering to generate images and depth maps.

### 2 Mapping Network Implementation

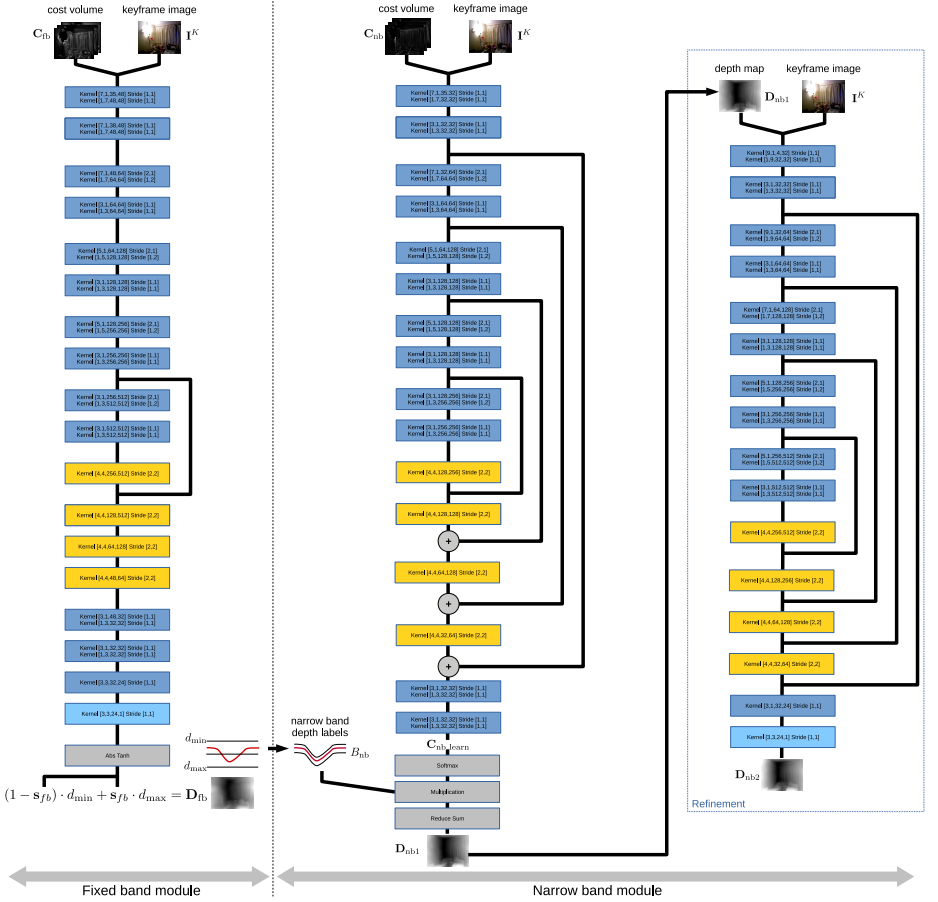
Fig. 3 shows the operations and parameters of the ConvNets of the mapping module.

### 3 Runtimes

Tab. 1 shows the runtimes of each network component of our approach. The tracking can run in real-time. For the mapping we use the cost volume to collect information from incoming frames. We do not run the fixed band and narrow band modules at frame rate, but invoke them once enough frames have been collected.



**Fig. 1.** Tracking network parameters. The shared encoder decoder configuration is identical for all resolution levels. The pose hypotheses generation part uses slightly different kernel size and stride parameters for each resolution level. The changing parameters for the resolution levels  $[60 \times 80, 120 \times 160, 240 \times 320]$  are:  $k_1 : [3, 5, 5]$ ,  $s_1 : [2, 4, 4]$ ,  $k_2 : [3, 3, 5]$ ,  $s_2 : [2, 2, 4]$ ,  $k_3 : [1536, 1536, 2048]$ ,  $k_4 : [24, 24, 32]$ .



**Fig. 2.** Mapping networks parameters. The mapping consists of two modules: fixed band module and narrow band module. **Fixed band module:** This module takes the keyframe image  $I^K$  (320 × 240 × 3) and the cost volume  $C_{fb}$  (320 × 240 × 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 × 240 × 1). The fixed band depth estimation is computed as  $D_{fb} = (1 - s_{fb}) \cdot d_{min} + s_{fb} \cdot d_{max}$ . **Narrow band module:** The narrow band module is run iteratively; in each iteration we build a cost volume  $C_{nb}$  from a set of depth labels distributed around the current depth estimation with a band width  $\sigma_{nb}$  of 0.0125. It consists of two encoder-decoder pairs. The first pair gets the cost volume  $C_{nb}$  (320 × 240 × 32) and the keyframe image  $I^K$  (320 × 240 × 3) as inputs and generates a learned cost volume  $C_{nb,learn}$  (320 × 240 × 32). The depth map is then obtained using a differentiable soft argmin operation [3]:  $D_{nb1} = \sum_{d \in B_{nb}} B_{nb} \times \text{softmax}(-C_{nb,learn})$ . The second encoder-decoder pair gets the current depth estimation  $D_{nb1}$  and the keyframe image  $I^K$  and produces a refined depth  $D_{nb2}$ .



	Tracking Cost volume Fixed band Narrow band			
Mean	0.0227	0.0164	0.0181	0.0359
Min	0.0203	0.0153	0.0171	0.0347
Max	0.0251	0.0168	0.0190	0.0393

**Table 1.** Runtime in seconds for each component of our system. The statistic is computed excluding outliers. The **Tracking** time is the time required for a forward pass through all three resolution levels including render time for the virtual keyframe. The isolated tracking component runs with about 44 Hz. The runtimes for **Cost volume** describe the time to compute and add the matching costs of a new frame. The cost volume generation is implemented with Tensorflow ops and has some overhead. The **Narrow band** time is the time per iteration. All runtimes have been measured on an NVIDIA GTX 1070.

## 4 Results

Tab. 2 shows an extended evaluation on the TUM RGB-D benchmark for the freiburg1 sequences with public ground truth and the validation sets with secret ground truth.

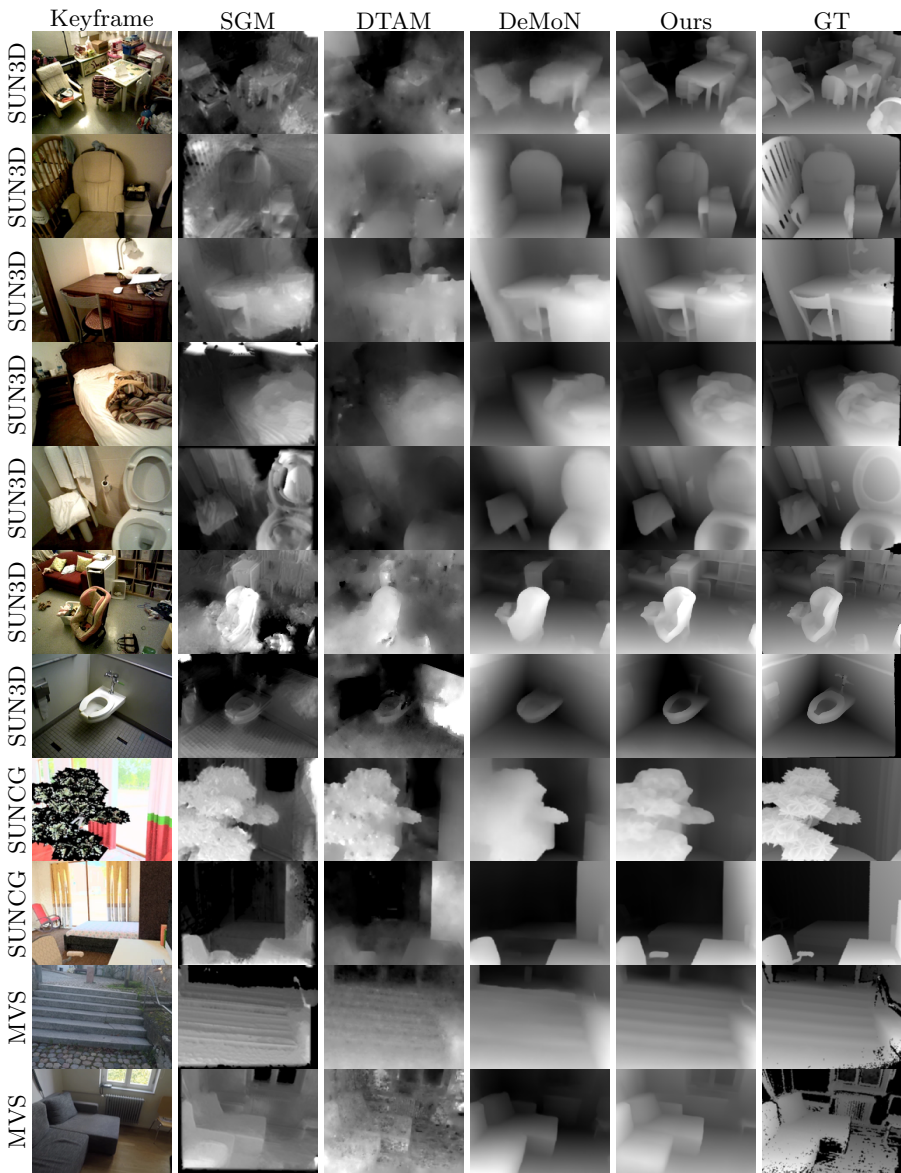
Fig. 3 shows more examples of our mapping component in comparison with SGM [2], DTAM [5] and DeMoN [8].

## 5 Generalization

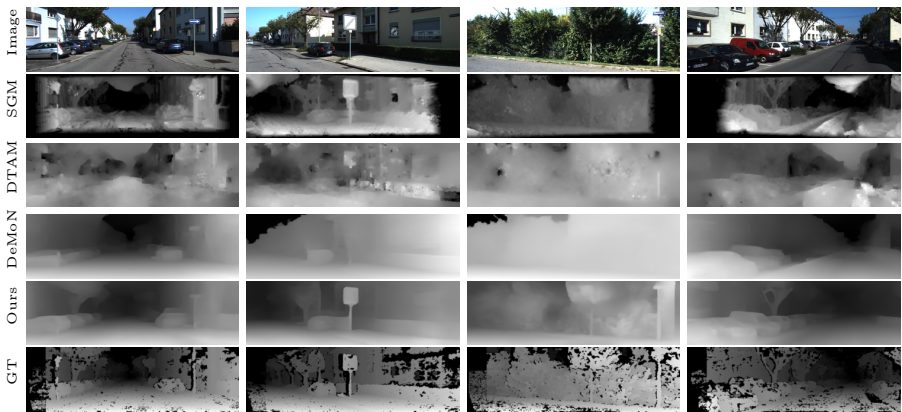
It is well known that learning-based methods easily overfit. To prevent this problem we trained and tested on diverse datasets, which covers indoor and outdoor scenarios and consists of realistic and artificial data. Additionally, we carefully design our network architecture to avoid overfitting. Tab. 2 shows our results on the TUM RGB-D benchmark, which was not part of our training data. In addition, Fig. 4 demonstrates that our mapping module also generalizes well to KITTI datasets without any finetuning.

	Tracking				Tracking and mapping	
Sequence	RGB-D SLAM Kerl <i>et al.</i> [4]	Ours (w/o flow)	Ours (single pose)	Ours	CNN-SLAM* Tateno <i>et al.</i> [7]	Ours
fr1/360	0.119	0.079	0.070	<b>0.063</b>	0.839	<b>0.133</b>
fr1/360 (v)	0.125	0.069	0.065	<b>0.054</b>	0.500	<b>0.116</b>
fr1/desk	<b>0.030</b>	0.051	0.048	0.033	0.175	<b>0.130</b>
fr1/desk (v)	0.037	0.042	0.031	<b>0.027</b>	0.095	<b>0.078</b>
fr1/desk2	0.055	0.064	0.054	<b>0.046</b>	0.236	<b>0.124</b>
fr1/desk2 (v)	0.020	0.025	0.020	<b>0.017</b>	0.115	<b>0.055</b>
fr1/floor	0.090	0.095	0.091	<b>0.081</b>	<b>0.282</b>	<b>0.282</b>
fr1/plant	0.036	0.038	0.028	<b>0.027</b>	<b>0.178</b>	0.299
fr1/plant (v)	0.062	0.063	0.060	<b>0.057</b>	<b>0.150</b>	0.165
fr1/room	0.048	0.059	0.048	<b>0.040</b>	0.169	<b>0.138</b>
fr1/room (v)	0.042	0.051	0.041	<b>0.039</b>	0.445	<b>0.084</b>
fr1/rpy	<b>0.043</b>	0.052	0.045	0.046	0.074	<b>0.046</b>
fr1/rpy (v)	0.082	0.070	<b>0.063</b>	0.065	0.261	<b>0.052</b>
fr1/teddy	0.067	0.067	<b>0.058</b>	0.059	0.207	<b>0.164</b>
fr1/xzy	0.024	0.029	0.021	<b>0.017</b>	0.060	<b>0.045</b>
fr1/xzy (v)	0.051	0.030	0.021	<b>0.019</b>	0.206	<b>0.054</b>
average	0.058	0.055	0.047	<b>0.043</b>	0.250	<b>0.123</b>

**Table 2.** Evaluation of our tracking (left part) and the combined mapping and tracking (right part) on the RGB-D benchmark [6]. The values describe the translational RMSE in  $[m/s]$ . The validation sets are marked with (v). **Tracking:** We compare the performance of our tracking network against the RGB-D SLAM method of Kerl *et al.* [4]. Numbers for Kerl *et al.* [4] correspond to the frame-to-keyframe odometry evaluation and have been copied from their paper. Note that Kerl *et al.* [4] uses 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 (single pose)** is our tracking with generating just a single pose hypothesis and deactivated  $\mathcal{L}_{uncertainty}$ . **Ours** is our tracking network with 64 pose hypotheses and the mean as final estimate. Both versions use the depth stream to obtain the depth for keyframes. **Tracking and mapping:** We compare our tracking and mapping against CNN-SLAM by Tateno *et al.* [7]. \* 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.



**Fig. 3.** Qualitative depth prediction comparison for sequences with 10 frames. DeMoN uses only the first and last frame of each sequence.



**Fig. 4.** Generalization experiment on KITTI [1]. 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. 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.

## References

1. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference On. pp. 3354–3361. IEEE (2012)
2. Hirschmüller, H.: 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 (Jun 2005). <https://doi.org/10.1109/CVPR.2005.56>
3. Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P.: End-to-End Learning of Geometry and Context for Deep Stereo Regression. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 66–75 (Oct 2017). <https://doi.org/10.1109/ICCV.2017.17>
4. Kerl, C., Sturm, J., Cremers, D.: Dense visual SLAM for RGB-D cameras. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2100–2106 (Nov 2013). <https://doi.org/10.1109/IROS.2013.6696650>
5. Newcombe, R.A., Lovegrove, S., Davison, A.: DTAM: Dense tracking and mapping in real-time. In: 2011 IEEE International Conference on Computer Vision (ICCV). pp. 2320–2327 (2011). <https://doi.org/10.1109/ICCV.2011.6126513>
6. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 573–580 (Oct 2012). <https://doi.org/10.1109/IROS.2012.6385773>
7. Tateno, K., Tombari, F., Laina, I., Navab, N.: 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 (Jul 2017). <https://doi.org/10.1109/CVPR.2017.695>
8. Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., Brox, T.: DeMoN: Depth and Motion Network for Learning Monocular Stereo. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)