

TANDEM: Tracking and Dense Mapping in Real-time using Deep Multi-view Stereo

Supplementary Material

Anonymous Author(s)

Affiliation

Address

email

In this supplementary material, we briefly introduce the volumetric mapping of TANDEM (section 1), discuss the initialization in section 2, show further experimental results (section 3), give more details regarding our experiments (section 4) and, finally, provide implementation details for TANDEM in section 5 including the possibility to deploy TANDEM on an embedded device (cf. section 5.3). Upon publication, we will release our code for TANDEM as well as the rendered Replica sequences to facilitate the reproduction of our results.

We also urge the readers to watch the **supplementary video** which shows the real-time demos of TANDEM running on the unseen sequences.

1 Volumetric Mapping

We use TSDF fusion [1] to fuse the per-keyframe estimated depth maps into a globally consistent and dense 3D model. Storing the TSDF values within a dense voxel grid has a cubic memory requirement in the spatial resolution of the grid and is thus unpractical. We employ voxel hashing [2] to alleviate this issue. The TSDF fusion is run on the GPU to ensure real-time performance.

For each voxel of the grid we store the estimated TSDF value $D_i(x) \in \mathbb{R}$, the estimated RGB values $C_i(x) \in \mathbb{R}^3$, and the weight $W_i(x) \in \mathbb{R}$, where x indexes the 3D location and i indexes the time. After our CVA-MVSNet has predicted a new depth map, we iterate over all voxels within the truncation distance and update their stored quantities. Let $d_{i+1}(x) \in \mathbb{R}$ be the projective TSDF value of the voxel at x based on the depth map from the CVA-MVSNet. Furthermore, let $c_{i+1}(x)$ be the associated RGB value from the input image, then

$$D_{i+1}(x) = \frac{W_i(x)D_i(x) + d_{i+1}}{W_i(x) + 1}, \quad (1)$$

$$C_{i+1}(x) = \frac{W_i(x)C_i(x) + c_{i+1}}{W_i(x) + 1}, \quad (2)$$

$$W_{i+1}(x) = \min(W_i(x) + 1, 64). \quad (3)$$

The weight $W_i(x)$ is truncated at 64 (cf. Equation 3) to ensure that new measurements can influence the estimated quantities and avoid over-saturation. Depth maps are rendered from the TSDF volume using raycasting [1] and used for the proposed dense front-end tracking.

Because TANDEM is a monocular, geometry-based method the overall scale of the scene is not observable. Therefore, the voxel grid is scaled with the same scale as the visual odometry. The grid uses a voxel size of 0.01 which roughly corresponds to 2.5cm depending on the exact scale of a particular scene and run. We use a truncation distance for the TSDF of 0.1 which roughly corresponds to 25cm. The global scale ambiguity that is common to all monocular, geometry-based methods renders the voxel size scene dependent. For our experiments the scene scale does not vary considerably and therefore we use a fixed voxel size. However, when considering scenes of different extent, the voxel size and truncation distance have to be set accordingly, which is also

31 necessary when running classical TSDF fusion. The internal scale of TANDEM is generally such
 32 that the mean depth of the sparse points in the first frame is approximately identity.

33 The CVA-MVSNet requires a minimum and maximum depth value for inference, which is usually
 34 given in the dataset. However, during the live operation of TANDEM, we do not know the relative
 35 scale between the world and our reconstruction. To facilitate the live operation of TANDEM, we
 36 choose a simple strategy: the minimum depth is set 0.01, which is small enough to capture all
 37 objects, and the maximum depth is initially set to ten times the mean depth of the sparse points. For
 38 each following invocation of the CVA-MVSNet, we set the maximum depth to 1.5 times the previous
 39 maximum depth estimated by the network, which ensures that the whole scene is covered while
 40 providing good depth resolution. We found this simple scheme to work well in our experiments.

41 2 Initialization

42 The proposed CVA-MVSNet operates on a keyframe window and thus the tracking is initialized the
 43 same way as in DSO [3] with non-linear optimization on poses and the sparse depth. The TSDF
 44 volume is initialized to represent empty space, i.e. with zero weights $W_0(x) = 0 \forall x$. Because we
 45 use voxel hashing [2] to represent the TSDF volume, empty space can be represented very efficiently
 46 by not allocating any voxel blocks. After the first dense depth map is predicted and integrated into
 47 the TSDF volume, TANDEM uses the rendered nearly dense depth maps for tracking as can be seen
 48 from Figure 5.

49 3 Further Experimental Results

50 We show the trajectory evaluation on ICL-NUIM in Figure 1 and Table 1. Similar to the evaluation
 51 on the EuRoC dataset in the main paper, we see that the proposed TANDEM shows better results
 52 than DSO and DeepFactors [4]. Furthermore, TANDEM performs favourably in comparison to the
 53 tracking component of DeepTAM [5] that uses ground truth depth maps. We compare the trajectories
 54 for DeepFactors and TANDEM on the office1 sequence of the ICL-NUIM dataset in Figure 2. In
 55 Figure 3 we reproduce Figure 3 from the main paper in higher resolution to enable closer inspection
 56 of the generated depth maps. Furthermore, we show additional qualitative depth map comparisons
 57 in Figure 4. In section 3.1 we discuss failure cases and challenges for TANDEM. The nearly dense
 58 depth maps used for tracking by TANDEM and the sparse depth maps from the photometric bundle
 59 adjustment are shown in Figure 5 and Figure 6.

60 3.1 Failure Cases and Challenges

61 Dense monocular SLAM is a challenging problem [6] and most systems fail in certain scenarios or
 62 perform worse given certain challenges. Any system that involves deep learning has to consider the
 63 generalization capability of neural network since, for SLAM, the training and testing datasets will
 64 practically always differ. Pure rotational motion is a known challenge for both visual SLAM and 3D
 65 reconstruction due to the unobservability of depth. Dynamic scenes, rolling shutter, and geometric
 66 errors are well known failure cases in SLAM if they are not modelled explicitly. While some of
 67 the aforementioned failure cases and challenges can be overcome by explicit modelling or through
 68 employing different techniques for tracking or mapping, this often leads to an increase in runtime
 69 and system complexity. In the following, we describe the weaknesses and the failure cases of the
 70 proposed system in order to facilitate further research in this direction:

71 **Neural Network Generalization** Generalization of deep neural networks is well known to pose
 72 challenges [7] and a general solution has yet to be proposed. For deep MVS multiple remedies
 73 have been proposed, including diverse training datasets [8], specific architecture choices [9, 10],
 74 and self-supervised adaption [11, 12]. While our CVA-MVSNet already shows good generalization
 75 capabilities as it was trained on either the ScanNet or the Replica dataset and generalizes to the ICL-
 76 NUIM and EuRoC datasets, the error metrics are better for the training datasets. Although initial
 77 experiments with specific architectures as well as self-supervised adaption didn't show convincing
 78 results for our training and testing datasets, we consider this a good direction for future research. For
 79 practical applications we consider the utilization of diverse training data together with re-training for
 80 new scenarios the most promising direction.

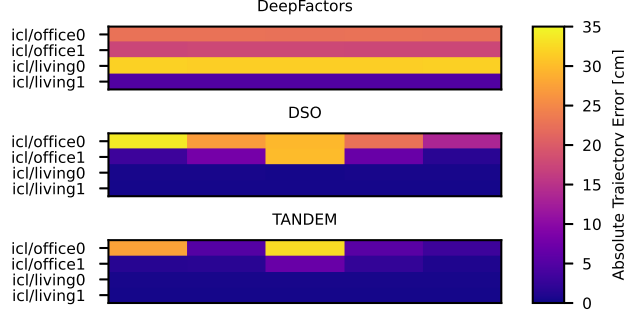


Figure 1: Trajectory evaluation on the ICL-NUIM dataset. Because all systems use monocular images we perform Sim(3) alignment w.r.t. the ground truth trajectories. We show the color-coded absolute trajectory error (ATE) for five runs. The results show that our dense tracking improves the robustness over DSO. DeepFactors produces nearly identical results for all five runs in this experiment but shows a higher median error.

Pure Rotational Motion Pure rotational motion is critical because the depth of a point cannot be inferred, which leads to problems for tracking as well as depth estimation using MVS. While some purely rotational motion can be handled by TANDEM if the system has been initialized well, rotation-only motion during initialization will lead to failure in tracking. We experimentally validate this using the sequence office0 from the ICL-NUIM dataset. This sequence contains highly rotational motion and thus the results of TANDEM and DSO are worse in comparison to the office1 sequence or the living room sequences (cf. Table 1). We investigate the effect of rotational motion during initialization by starting the office0 sequence at frame 300, which is followed by a rotation along the ceiling of the office. Although the subsequence is shorter than the full sequence, the mean RMSE of the absolute pose error for TANDEM increases from 0.056 m to 0.354 m, which shows that rotational motion during initialization remains an open challenge.

Dynamic Scenes While the photometric bundle adjustment of TANDEM can implicitly handle dynamic objects if they constitute a relatively small fraction of the scene, the deep MVS fully relies on a static scene. Such scenarios are favourable for mono-to-depth-based methods as they are independent of the scene dynamics when predicting the depth. On the other hand, dynamic objects can be explicitly modelled within the cost volume [13].

Rolling Shutter The rolling shutter effect is known to be problematic for direct SLAM because not all pixels correspond to the same time and thus the same pose [14]. To investigate the effect of rolling shutter on TANDEM we use the ICL-NUIM living room sequences with synthetic rolling shutter proposed in [15]. For the sequences living0 and living1 the mean ATE RMSE increases from 0.006 m to 0.018 m and from 0.005 m to 0.074 m, respectively.

Geometric Error Photometric methods are susceptible to geometric errors, e.g. from an inaccurate camera calibration, because these geometric errors are, in contrast to feature-based methods, not modeled explicitly. We simulate geometric errors by using an incorrect calibration for the ICL-NUIM living room sequences, specifically we add 5 px to f_x , f_y , c_x , and c_y . For the sequences living0 and living1 the mean ATE RMSE increases from 0.006 m to 0.041 m and from 0.005 m to 0.026 m, respectively.

4 Experimental Details

4.1 Poses and Alignment

For the ablation study, we use ground-truth poses and a fixed set of keyframe windows to enable a fair and accurate comparison.

For the depth evaluation on ICL-NUIM and EuRoC, we run our visual odometry once to generate poses as well as keyframe windows. We use the same poses and keyframe windows to evaluate Cas-

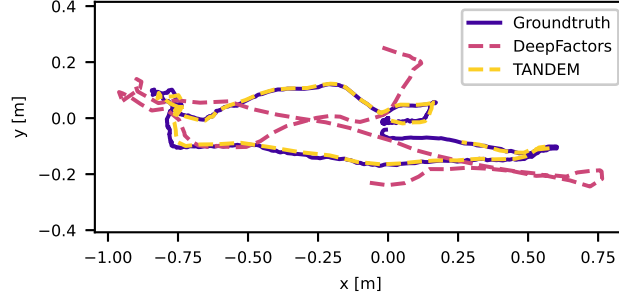


Figure 2: Estimated trajectories of DeepFactors and TANDEM on icl/office1. For both methods we select the run with the median error (cf. Figure 1).

Table 1: Pose evaluation on ICL-NUIM [18]. All the methods except DeepTAM are Sim(3) aligned w.r.t. the ground-truth trajectories. The mean absolute pose errors in meters and the standard deviations over five runs are shown. [†]For DeepTAM the official implementation contains only a pure tracking mode that requires ground truth depth maps for tracking, therefore, SE(3) alignment is performed because the scale is observable through the depth maps. Note that we show the result of a single run for DeepTAM since the results are consistent across multiple runs.

Sequence	DeepTAM [†] [5]	DeepFactors [4]	DSO [3]	TANDEM
icl/office0	<u>0.086</u> (-)	0.226 (± 0.000)	0.270 (± 0.070)	0.056 (± 0.126)
icl/office1	<u>0.030</u> (-)	0.175 (± 0.001)	0.069 (± 0.102)	0.017 (± 0.021)
icl/living0	<u>0.025</u> (-)	0.315 (± 0.001)	0.006 (± 0.000)	<u>0.006</u> (± 0.000)
icl/living1	0.034 (-)	0.049 (± 0.000)	<u>0.005</u> (± 0.000)	0.005 (± 0.000)

114 MVSNet, Ours (ScanNet), and Ours (Replica) to enable a fair and accurate comparison. Evaluating
 115 based on the full system would introduce significant non-determinism into the results. Because
 116 TANDEM is a monocular method, we use the scale estimated from the Sim(3) alignment of the
 117 trajectory to scale the depth maps. Since DeepFactors does not estimate very accurate poses on the
 118 EuRoC dataset (cf. Table 4 in the main paper), we scale align each depth map individually using
 119 the median scale between the ground-truth depth map and the predicted depth map. This procedure
 120 potentially overestimates the accuracy of DeepFactors, which is important to note when comparing
 121 with other methods.

122 For the comparison to iMAP, we use the Sim(3) alignment of the trajectory of TANDEM and the
 123 ground truth trajectory to transform the mesh into the same coordinate frame as the reference.

124 For the comparison on EuRoC we rectify the images s.t. the resulting images have resolution
 125 640×480 and evaluate all methods except CodeVIO on the same rectified images for a fair com-
 126 parison. For CodeVIO no public source code is available and we thus use the numbers published by
 127 the authors. Using a different rectification protocol can result in different results, which is often not
 128 obvious from the paper.

129 4.2 Depth Evaluation Metrics

130 For the following we will use y_i^* to denote the ground-truth depth value in meters and y_i to denote
 131 the corresponding predicted depth value. All metrics are computed per image first and then the
 132 average is taken over all images within the sequence. We let the index $i = 1, \dots, N$ enumerate all
 133 pixels with valid ground-truth depth for a given image. Note that thus there holds $y_i^* > 0, \forall i$.

134 The paper uses inconsistent metrics, e.g. a_1 and d_1 , because prior works, e.g. DeepFactors and
 135 CodeVIO, used inconsistent metrics. Additionally, for some prior works, e.g. CodeVIO and CNN-
 136 SLAM, the source code is not available and therefore we cannot evaluate all methods using one
 137 consistent metric.

138 For the comparison on ICL-NUIM we follow DeepFactors and use the a_1 metric

$$a_1 = \frac{100}{N} \sum_{i=1}^N \mathbb{1}(|y_i - y_i^*|/y_i^* < 0.1), \quad (4)$$

139 where $\mathbb{1}(\cdot)$ is the indicator function. The a_1 metric gives the percentage of pixels for which the
140 estimated depth falls within 10% of the ground-truth depth.

141 For the ablation study, we follow DeepFactors and use the a_1 metric. We found that this metric is
142 saturated because all configurations reach more than 98% and thus also show the stricter a_2 and a_3
143 metrics defined by

$$a_2 = \frac{100}{N} \sum_{i=1}^N \mathbb{1}(|y_i - y_i^*|/y_i^* < 0.01), \quad (5)$$

$$a_3 = \frac{100}{N} \sum_{i=1}^N \mathbb{1}(|y_i - y_i^*|/y_i^* < 0.001). \quad (6)$$

144 The a_2 and a_3 metrics give the percentage of pixels for which the estimated depth falls within 1%
145 and 0.1% of the ground truth depth. Additionally, we show the mean absolute error in centimeters

$$Abs = \frac{100}{N} \sum_{i=1}^N |y_i - y_i^*|. \quad (7)$$

146 For the comparison on EuRoC we follow CodeVIO and use the d_1 metric as introduced by Eigen et
147 al. [19]

$$d_1 = \frac{100}{N} \sum_{i=1}^N \mathbb{1}\left(\max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) < 1.25\right). \quad (8)$$

148 5 Implementation Details

149 5.1 CVA-MVSNet

150 We list all the hyperparameters and their corresponding values in Table 5.

151 **Architecture.** The proposed CVA-MVSNet consists of three trainable components: the feature
152 extraction network (cf. Table 2), the view aggregation network (cf. Table 3), and the cost regulariza-
153 tion network (cf. Table 4). The feature extraction network takes as input each frame I_i and outputs
154 the corresponding multi-scale feature maps \mathbf{F}_i^s , which are denoted by `out.stage1`, `out.stage2`,
155 `out.stage3` in Table 2. The weights are shared for all frames. The view aggregation network takes
156 as inputs the single frame cost volumes $(\mathbf{V}_i^s - \mathbf{V}_j^s)^2$ and outputs the aggregation weights \mathbf{W}_i^s . The
157 weights are shared for all frames but are not shared for the three stages. The cost regularization
158 network takes as input the aggregated cost volume

$$\mathbf{C}^s = \frac{\sum_{i=1, i \neq j}^n (1 + \mathbf{W}_i^s) \odot (\mathbf{V}_i^s - \mathbf{V}_j^s)^2}{n - 1},$$

159 and outputs the probability volume \mathbf{P}^s . There is a single network for all frames and the weights are
160 not shared for the three stages.

161 5.2 Runtime

162 We report the mean runtimes for the single components of TANDEM in Table 6. Overall, TANDEM
163 requires an average of 47 ms of processing time per frame, which gives a throughput of ca. 21 FPS.
164 Additionally, Table 6 shows that using asynchronicity and parallelism between CPU and GPU is
165 necessary to achieve real-time capability.

Table 2: Feature extraction network. Layers are 2D convolutions or nearest neighbor interpolation with a scale factor 2, which we denote by \uparrow . For all layers we show the input and the channels (**chns**). For convolutions, we additionally show the kernel size (**k**), the stride (**s**), the padding (**p**), if the layer uses batch normalization (**bn**), and the activation function (**act**). Only the layers `skip.stage2` and `skip.stage3` have a bias term. We use batch normalization before the activation function.

Layer	input	k	s	p	chns	bn	act
conv0.0	image	3	1	1	3 \rightarrow 8	✓	ReLU
conv0.1	conv0.0	3	1	1	8 \rightarrow 8	✓	ReLU
conv1.0	conv0.1	5	2	2	8 \rightarrow 16	✓	ReLU
conv1.1	conv1.0	3	1	1	16 \rightarrow 16	✓	ReLU
conv1.2	conv1.1	3	1	1	16 \rightarrow 16	✓	ReLU
conv2.0	conv1.2	5	2	2	16 \rightarrow 32	✓	ReLU
conv2.1	conv2.0	3	1	1	32 \rightarrow 32	✓	ReLU
conv2.2	conv2.1	3	1	1	32 \rightarrow 32	✓	ReLU
out.stage1	conv2.2	1	1	0	32 \rightarrow 32		
skip.stage2	conv1.2	1	1	0	16 \rightarrow 32		
inter.stage2	\uparrow conv2.2 + skip.stage2				32 \rightarrow 32		
out.stage2	inter.stage2	3	1	1	32 \rightarrow 16		
skip.stage3	conv0.1	1	1	0	8 \rightarrow 32		
inter.stage3	\uparrow inter.stage2 + skip.stage3				32 \rightarrow 32		
out.stage3	inter.stage3	3	1	1	32 \rightarrow 8		

Table 3: View aggregation network. All layers are 3D convolutions. We show the input, the kernel size (**k**), the stride (**s**), the padding (**p**), the channels (**chns**), if the layer uses batch normalization (**bn**), and the activation function (**act**). All layers have a bias term. We use batch normalization before the activation function. The single frame cost volumes (single frame cost volume) have 32 channels for stage 1, 16 channels for stage 2, and 8 channels for stage 3.

Layer	input	k	s	p	chns	bn	act
conv0	single frame cost volume	1	1	0	(32, 16, 8) \rightarrow 1	✓	ReLU
conv1	conv0	1	1	0	1 \rightarrow 1	✓	ReLU

166 5.3 Deployment on an Embedded Device

167 For a SLAM system in the context of mobile robotics the deployment to an embedded platform is
168 necessary for operation in the real world, if no uninterrupted connection to a server is available to
169 offload computation. While we consider the actual deployment to an embedded system outside the
170 scope of this research work, we show in the following that such a deployment is possible while
171 maintaining accurate results.

172 An embedded system can benefit from software optimizations such as 16-bit float inference or
173 NVIDIA TensorRT [24], which can bring speedups of up to $2\times$ and $2.5\times$, respectively [25]. How-
174 ever, we consider these engineering-focused optimizations outside the scope of this work.

175 The inference time for our CVA-MVSNet given in the main paper is 158 ms and decreases to 133
176 ms by switching from PyTorch 1.5 to PyTorch 1.9. The 2D feature extraction network is run on all
177 images from the keyframe window, while for all but one image the feature maps have been computed
178 before. This fact can be used to save computational cost, however, to simplify the implementation,
179 we did not use this optimization for the timings in Table 1. If the feature extraction network is run
180 for only one image, the overall inference time decreases from 133 ms to 118 ms.

181 For the EuRoC experiment in the main paper we use images of size 640×480 , which is relatively
182 high in comparison to the image size used by DeepFactors of 256×192 . Using this smaller image
183 size decreases the inference time from 118 ms to 26.5 ms, a 77.5% relative improvement. The

Table 4: Cost regularization network. All layers are either 3D convolutions (conv) or 3D transposed convolutions (convT). For all layers, we show the input, the kernel size (**k**), the stride (**s**), the padding (**p**), the channels (**chns**), if the layer uses batch normalization (**bn**), and the activation function (**act**). For transposed convolutions we show the output padding (**op**). We use batch normalization before the activation function. No layer has a bias term. The stride S (cf. conv5 and convT7) is 2 for the first stage of CVA-MVSNet and $(1, 2, 2)$ for stages 2 and 3 because of the reduced number of depth planes. For the same reason, the output padding P (cf. convT7) is 1 for stage 1 and $(0, 1, 1)$ for stages 2 and 3. The cost volume (aggregated cost volume) has 32 channels for stage 1, 16 channels for stage 2, and 8 channels for stage 3.

Layer	input	k	s	p	op	chns	bn	act
conv0	aggregated cost volume	3	1	1		$(32, 16, 8) \rightarrow 8$	✓	ReLU
conv1	conv0	3	2	1		$8 \rightarrow 16$	✓	ReLU
conv2	conv1	3	1	1		$16 \rightarrow 16$	✓	ReLU
conv3	conv2	3	2	1		$16 \rightarrow 32$	✓	ReLU
conv4	conv3	3	1	1		$32 \rightarrow 32$	✓	ReLU
conv5	conv4	3	S	1		$32 \rightarrow 64$	✓	ReLU
conv6	conv5	3	1	1		$64 \rightarrow 64$	✓	ReLU
convT7	conv6	3	S	1	P	$64 \rightarrow 32$	✓	ReLU
convT8	conv4 + convT7	3	2	1	1	$32 \rightarrow 16$	✓	ReLU
convT9	conv2 + convT8	3	2	1	1	$16 \rightarrow 8$	✓	ReLU
prob	conv0 + convT9	3	1	1		$8 \rightarrow 1$		Softmax _{depth}

model trained on 640×480 images can be used for any resolution due to the fully-convolutional and geometry-based CVA-MVSNet and achieves an absolute error of 6.52 cm on the Replica validation set in comparison to 2.33 cm for the full resolution. However, when evaluating on EuRoC using the lower resolution results in an average accuracy (d_1) of 91.43% in comparison to 94.40%. The much smaller difference is due to the domain shift and additionally due to the relative laxness of the d_1 metric. Overall, the lower-resolution model is computationally much more efficient at a reasonable accuracy decrease.

Finally, TANDEM produces new keyframes often and marginalizes them early, which has been shown to aid tracking [3, 26]. A new keyframe is created roughly every 5 frames and the MVS network is called for each new keyframe. It would be possible to estimate depth only for every second keyframe to trade-off runtime and reconstruction quality.

Combining all the aforementioned measures decreases the inference time from 158 ms to 25.6 ms, a 83.2% decrease, while maintaining reasonable reconstruction quality and without optimizations like Nvidia TensorRT. Additionally, the network could be used on every second keyframe, i.e. with ca. 2 Hz. This combination makes our CVA-MVSNet $20\times$ real-time on an RTX 2080 super. The embedded Nvidia Jetson AGX Xavier has ca. 15% of the computing power of the RTX 2080 super and thus our CVA-MVSNet could potentially reach real-time on this device even without NVIDIA TensorRT. The classic SLAM components of TANDEM can be run in real-time on an embedded device [27], hence the proposed TANDEM could likely be implemented on an embedded device in real-time while maintaining reasonable reconstruction quality.

Table 5: Hyperparameter Table.

	Value	Description
min depth	0.01	Minimal depth value in meters used for generating the depth planes for our CVA-MVSNet.
max depth	10.0	Maximal depth value in meters used for generating the depth planes for our CVA-MVSNet.
depth planes	(48, 4, 4)	Number of depth planes for each stage.
depth intervals	(0.213, 0.106, 0.053)	The distance in meters between two depth planes for each stage. For the first stage, the planes are evenly spaced between the minimum and maximum depth. For stages 2 and 3 the interval is divided by 2 and 4, respectively.
optimizer	Adam	We use the default parameters: $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$.
learning rate	0.004	Learning rate at the start of the schedule.
learning rate schedule	linear decay	Linear decay from $1r$ to $1r/100$.
epochs	50	The number of training epochs on the Replica dataset.
batch size	4×2	We train with 4 GPUs with batch size 2 on each GPU without synchronized batch norm.
BN momentum	0.1	Batch normalization momentum.
image size	640×480	The size of images and depth maps on the finest scale, i.e. stage three.

Table 6: Timing results for TANDEM on EuRoC/V101. We show the averaged per-frame times and the averaged per-keyframe times and timed 2685 single frames for which 722 keyframes were created. Processing one frame takes 47 ms, which gives a throughput of ca. 21 FPS. The CVA-MVSNet, TSDF fusion, and Bundle Adjustment are run only for keyframes and thus we show the time taken per keyframe as well as the average time taken per frame, which is ca. four times lower due to the ratio of frames and keyframes. Note that the CVA-MVSNet per-keyframe time is lower than what we reported in the main paper (cf. Table 1) because we use asynchronicity and parallelism between CPU and GPU to ensure real-time performance.

	Per Frame [ms]	Per Keyframe [ms]
Number	2685	722
CVA-MVSNet	22.6	53.0
TSDF Fusion		29.3
Coarse Tracking	10.5	
Bundle Adjustment	13.9	45.2
Sum	47.0	

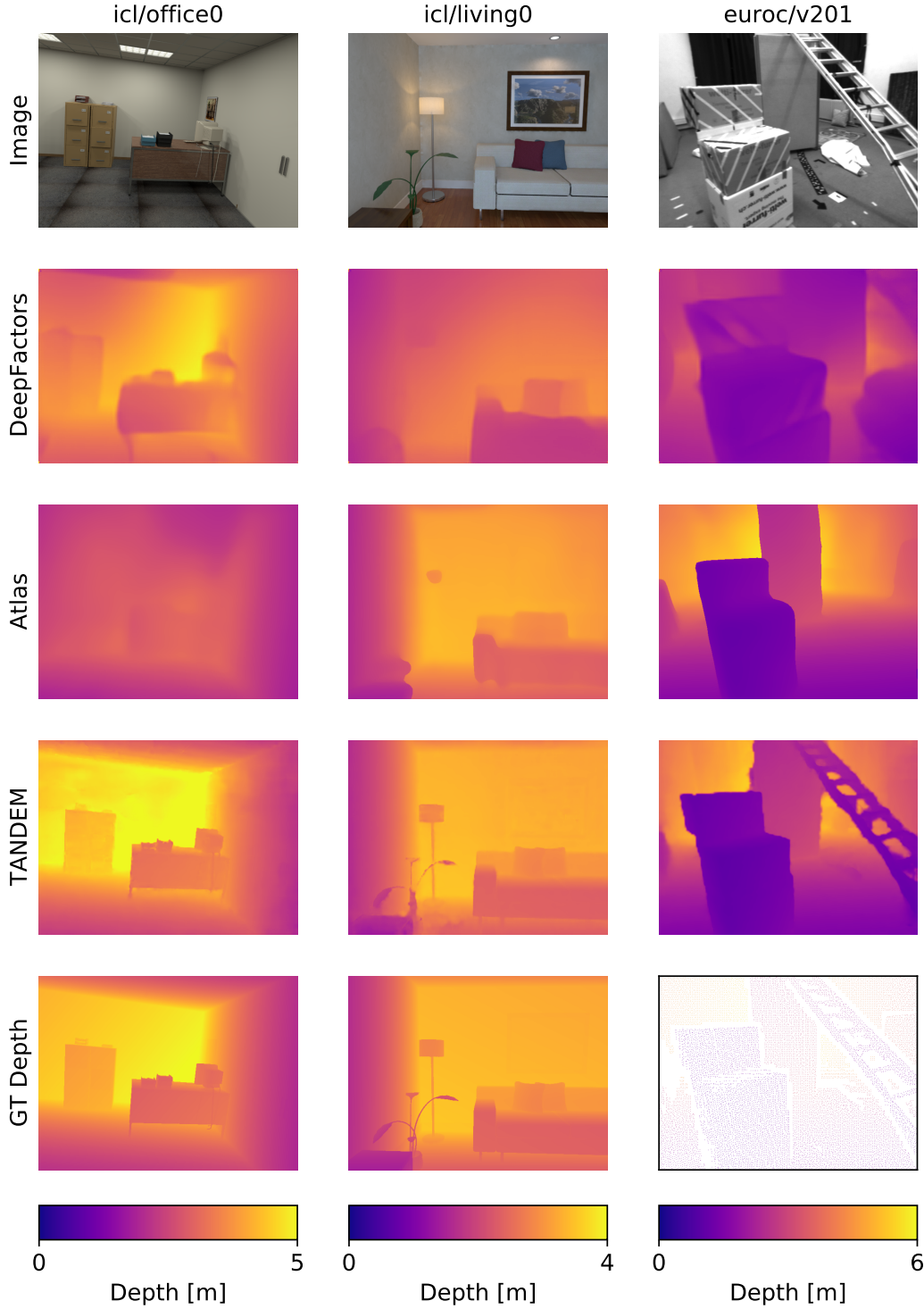


Figure 3: Depth comparison for DeepFactors [4], Atlas [16], and TANDEM on unseen sequences. TANDEM produces finer-scale details, e.g. the plant in the second column, or the ladder in the third column. For EuRoC only sparse ground-truth depth is available. This is a high-resolution version of Figure 3 from the main paper. In Figure 4 we show further qualitative results.

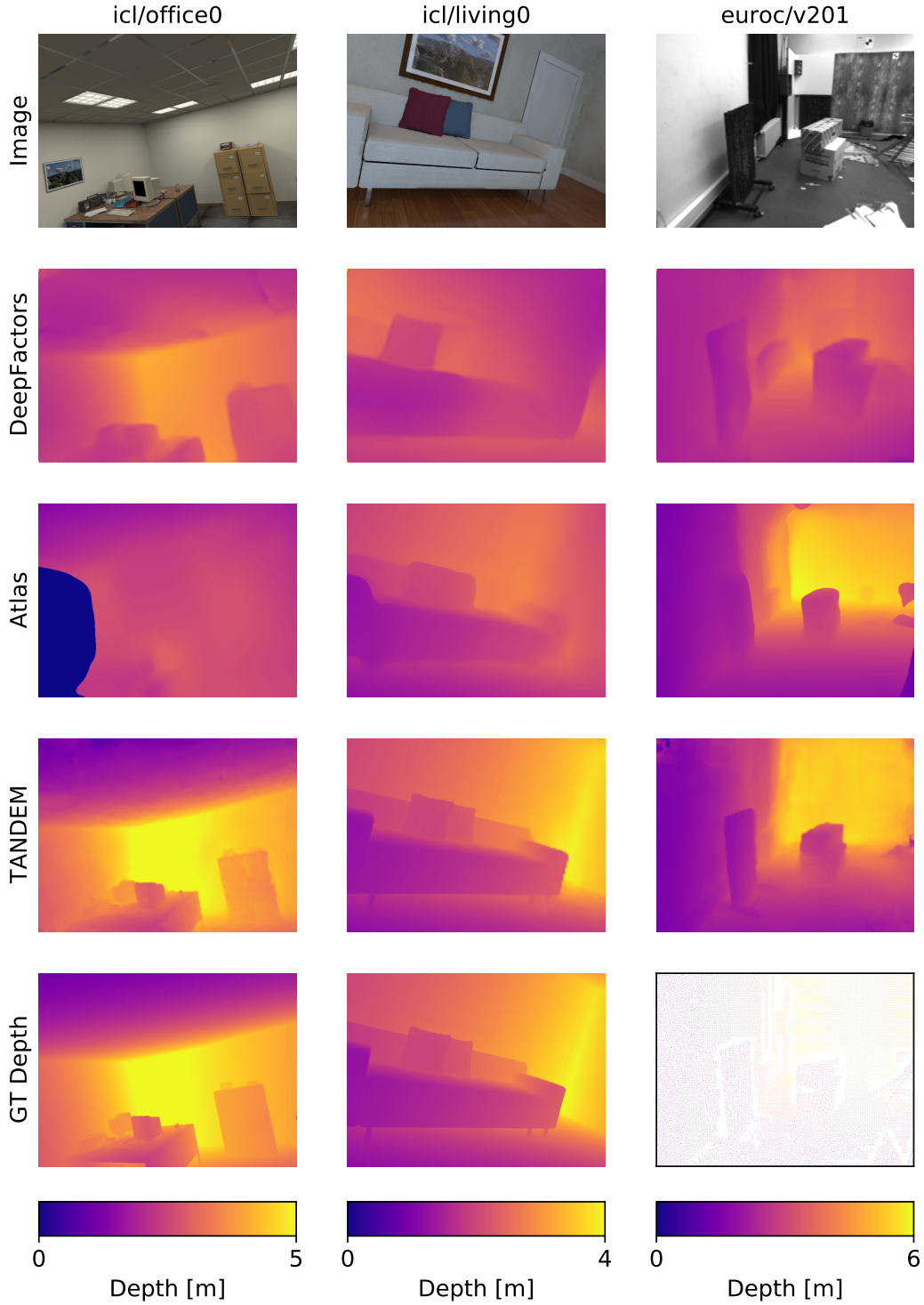


Figure 4: Further depth comparison for DeepFactors [4], Atlas [16], and TANDEM on unseen sequences. TANDEM produces finer-scale details, e.g. the computers in the first column, the pillows in the second column, or the radiator in the third column. However, TANDEM can produce outliers due to the cost volume-based formulation (cf. third column upper left corner of the image). For Atlas, the reconstruction of office0 is too small and thus the rendered depth has an invalid region (the blue blob in bottom left). For EuRoC only sparse ground-truth depth is available.

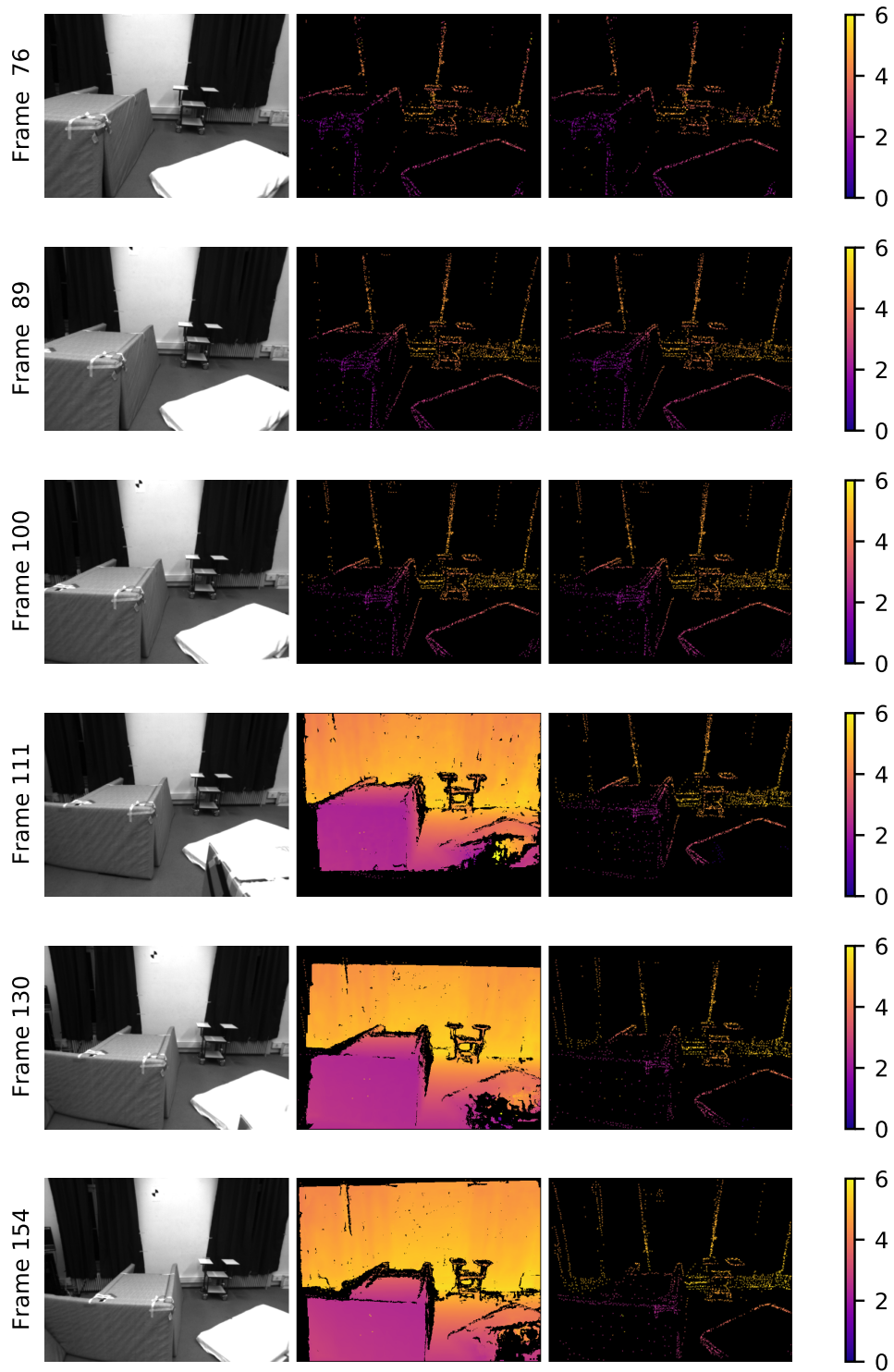


Figure 5: Tracking Depth Maps for EuRoC/V201. For every second keyframe we show the image (*left*), the depth map used for tracking by TANDEM (*middle*), and the sparse depth map that would be used without the dense tracking (*right*). For the first few keyframes no dense depth is available and thus TANDEM uses the sparse depth map for tracking. Note that, as in DSO, the sparse depth maps are slightly dilated before they are used for tracking. Further frames are shown in [Figure 6](#).

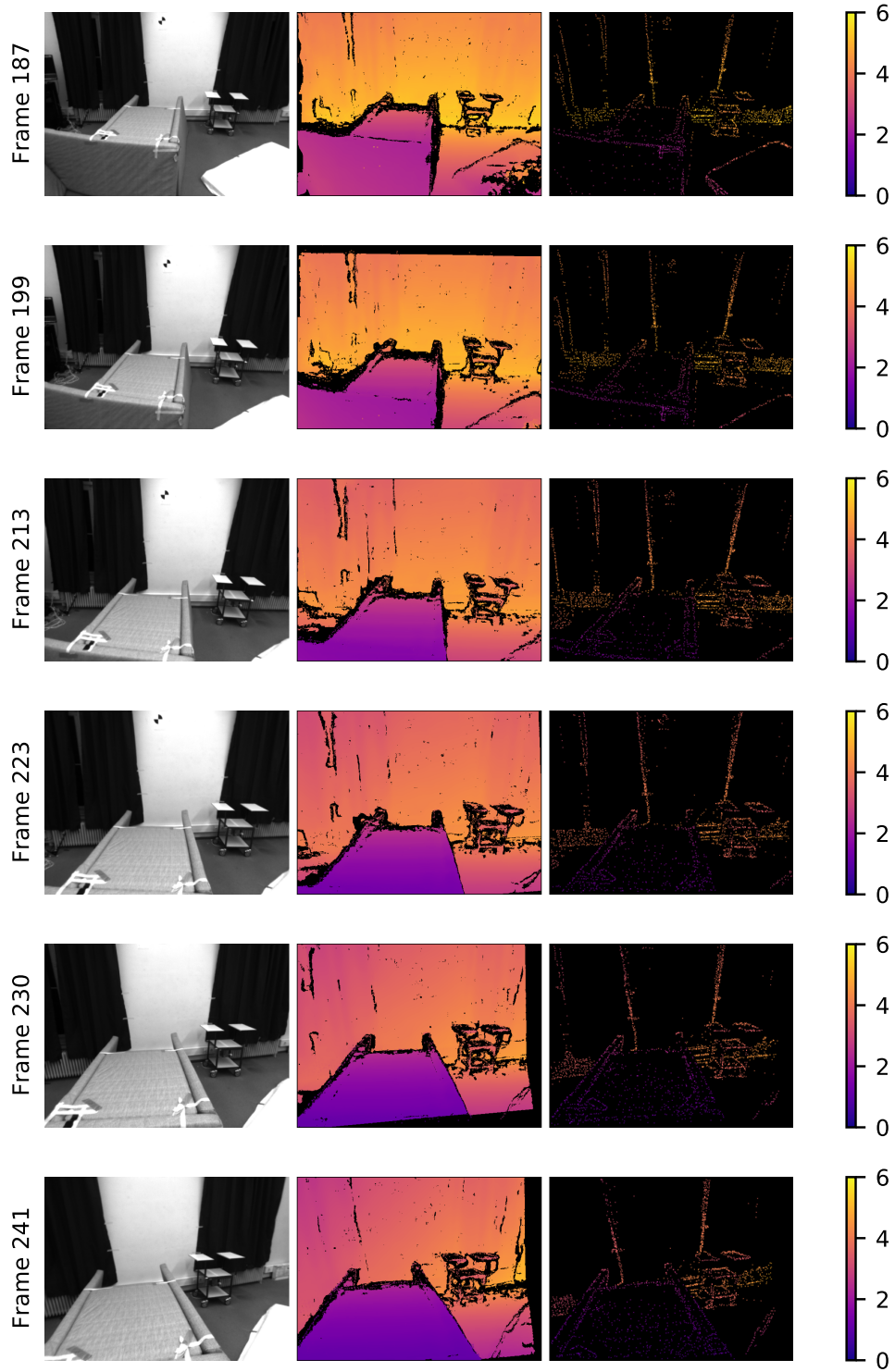


Figure 6: Tracking Depth Maps for EuRoC/V201. For every second keyframe we show the image (*left*), the depth map used for tracking by TANDEM (*middle*), and the sparse depth map that would be used without the dense tracking (*right*). The nearly dense depth map represents the scene well and thus the dense tracking can give more accurate results than the sparse tracking [17, 6]. The dense depth map also incorporates the sparse depth values which can be seen at occlusion boundaries. Note that, as in DSO, the sparse depth maps are slightly dilated. Previous frames are shown in Figure 5.

References

- [1] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1996.
- [2] M. Niessner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 2013.
- [3] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- [4] J. Czarowski, T. Laidlow, R. Clark, and A. J. Davison. DeepFactors: Real-time probabilistic dense monocular SLAM. *IEEE Robotics and Automation Letters*, 2020.
- [5] H. Zhou, B. Ummenhofer, and T. Brox. DeepTAM: Deep tracking and mapping. In *European Conference on Computer Vision (ECCV)*, 2018.
- [6] R. Newcombe. *Dense visual SLAM*. PhD thesis, Imperial College London, UK, 2012.
- [7] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio. Generalization in deep learning. *CoRR*, abs/1710.05468, 2017.
- [8] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [9] C. Cai, M. Poggi, S. Mattoccia, and P. Mordohai. Matching-space stereo networks for cross-domain generalization. In *3DV*, 2020.
- [10] F. Zhang, X. Qi, R. Yang, V. Prisacariu, B. Wah, and P. Torr. Domain-invariant stereo matching networks. In *European Conference on Computer Vision (ECCV)*, 2020.
- [11] H. Xu, Z. Zhou, Y. Qiao, W. Kang, and Q. Wu. Self-supervised multi-view stereo via effective co-segmentation and data-augmentation. In *AAAI Conference on Artificial Intelligence*, 2021.
- [12] M. Poggi, A. Tonioni, F. Tosi, S. Mattoccia, and L. Di Stefano. Continual adaptation for deep stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2021.
- [13] F. Wimbauer, N. Yang, L. von Stumberg, N. Zeller, and D. Cremers. Monorec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [14] D. Schubert, N. Demmel, V. Usenko, J. Stückler, and D. Cremers. Direct sparse odometry with rolling shutter. In *European Conference on Computer Vision (ECCV)*, 2018.
- [15] C. Kerl, J. Stückler, and D. Cremers. Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [16] Z. Murez, T. van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich. Atlas: End-to-end 3D scene reconstruction from posed images. In *European Conference on Computer Vision (ECCV)*, 2020.
- [17] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual SLAM: why filter? *Image Vis. Comput.*, 30(2):65–77, 2012.
- [18] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- [19] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Conference on Neural Information Processing Systems (NeurIPS)*, pages 2366–2374. Curran Associates, 2014.
- [20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

- 250 [21] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a
251 mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, 1982.
- 252 [22] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In
253 *International Conference on Machine Learning (ICML)*, 2010.
- 254 [23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International*
255 *Conference on Learning Representations (ICLR)*, 2015.
- 256 [24] NVIDIA. TensorRT. <https://developer.nvidia.com/tensorrt>, 2018.
- 257 [25] N. Dasan, C. Gottbrath, and J. Park. PyTorch-TensorRT: Accelerating inference in Py-
258 Torch with TensorRT. In *GPU Technology Conference (GTC)*, 2020. Available at [https:](https://developer.nvidia.com/gtc/2020/video/s21671)
259 [//developer.nvidia.com/gtc/2020/video/s21671](https://developer.nvidia.com/gtc/2020/video/s21671).
- 260 [26] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: A versatile and accurate
261 monocular SLAM system. *IEEE Transactions on Robotics*, 2015.
- 262 [27] T. Schöps, J. Engel, and D. Cremers. Semi-dense visual odometry for AR on a smartphone. In
263 *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2014.