Open access • Proceedings Article • DOI:10.1109/CVPR.2017.596

DeMoN: Depth and Motion Network for Learning Monocular Stereo — Source link

Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer ...+3 more authors

Institutions: University of Freiburg

Published on: 07 Dec 2016 - arXiv: Computer Vision and Pattern Recognition

Topics: Structure from motion and Optical flow

DeMoN: Depth and Motion Network for Learning Monocular Stereo

Benjamin Ummenhofer[*,1] Huizhong Zhou[*,1]
{ummenhof, zhouh}@cs.uni-freiburg.de

Jonas Uhrig[1,2] Nikolaus Mayer[1] Eddy Ilg[1] Alexey Dosovitskiy[1] Thomas Brox[1]
[1]University of Freiburg [2]Daimler AG R&D
{uhrigj, mayern, ilg, dosovits, brox}@cs.uni-freiburg.de

Abstract

In this paper we formulate structure from motion as a learning problem. We train a convolutional network end-to-end to compute depth and camera motion from successive, unconstrained image pairs. The architecture is composed of multiple stacked encoder-decoder networks, the core part being an iterative network that is able to improve its own predictions. The network estimates not only depth and motion, but additionally surface normals, optical flow between the images and confidence of the matching. A crucial component of the approach is a training loss based on spatial relative differences. Compared to traditional two-frame structure from motion methods, results are more accurate and more robust. In contrast to the popular depth-from-single-image networks, DeMoN learns the concept of matching and, thus, better generalizes to structures not seen during training.

1. Introduction

Structure from motion (SfM) is a long standing task in computer vision. Most existing systems, which represent the state of the art, are carefully engineered pipelines consisting of several consecutive processing steps. A fundamental building block of these pipelines is the computation of the structure and motion for two images. Present implementations of this step have some inherent limitations. For instance, it is common to start with the estimation of the camera motion before inferring the structure of the scene by dense correspondence search. Thus, an incorrect estimate of the camera motion leads to wrong depth predictions. Moreover, the camera motion is estimated from sparse correspondences computed via keypoint detection and descriptor matching. This low-level process is prone to outliers and does not work in non-textured regions. Finally, all existing SfM approaches fail in case of small camera translation. This is because it is hard to integrate priors that could provide reasonable solutions in those degenerate cases.

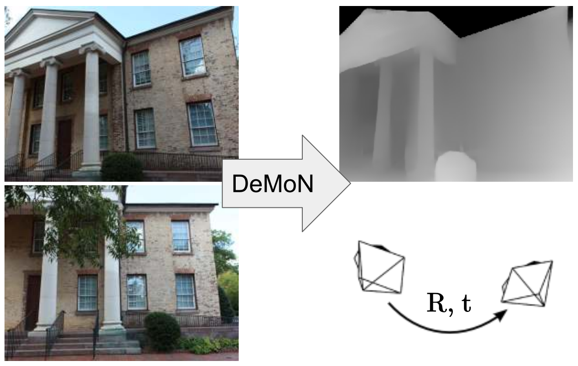


Figure 1. Illustration of DeMoN. The input to the network is two successive images from a monocular camera. The network estimates the depth in the first image and the camera motion.

In this paper, we succeed for the first time in training a convolutional network to jointly estimate the depth and the camera motion from an unconstrained pair of images. This approach is very different from the typical SfM pipeline in that it solves the problems of motion and dense depth estimation jointly. We cannot yet provide a full learning-based system for large-scale SfM, but the two-frame case is a crucial first step towards this goal. In the longer term, the learning approach has large potential, since it integrates naturally all the shape from X approaches: multi-view, silhouettes, texture, shading, defocus, haze. Moreover, strong priors on objects and structure can be learned efficiently from data and regularize the problem in degenerate cases; see Fig. 6 for example. This potential is indicated by our results for the two-frame scenario, where the learning approach clearly outperforms traditional methods.

Convolutional networks recently have shown to excel at depth prediction from a single image [7, 8, 24]. By learning priors about objects and their shapes these networks reach remarkably good performance in restricted evaluation sce-

*Equal contribution

narios such as indoor or driving scenes. However, single-image methods have more problems generalizing to previously unseen types of images. This is because they do not exploit stereopsis. Fig. 9 shows an example, where depth from a single image fails, because the network did not see similar structures before. Our network, which learned to exploit the motion parallax, does not have this restriction and generalizes well to very new scenarios.

To exploit the motion parallax, the network must put the two input images in correspondence. We found that a simple encoder-decoder network fails to make use of stereo: when trained to compute depth from two images it ends up using only one of them. Depth from a single image is a shortcut to satisfy the training objective without putting the two images into correspondence and deriving camera motion and depth from these correspondences.

In this paper, we present a way to avoid this shortcut and elaborate on it to obtain accurate depth maps and camera motion estimates. The key to the problem is an architecture that alternates optical flow estimation with the estimation of camera motion and depth; see Fig. 3. In order to solve for optical flow, the network *must* use both images. To this end, we adapted the FlowNet architecture [5] to our case. Our network architecture has an iterative part that is comparable to a recurrent network, since weights are shared. Instead of the typical unrolling, which is common practice when training recurrent networks, we append predictions of previous training iterations to the current minibatch. This training technique saves much memory and allows us to include more iterations for training. Another technical contribution of this paper is a special gradient loss to deal with the scale ambiguity in structure from motion. The network was trained on a mixture of real images from a Kinect camera, including the SUN3D dataset [43], and a variety of rendered scenes that we created for this work.

2. Related Work

Estimation of depth and motion from pairs of images goes back to Longuet-Higgins [25]. The underlying 3D geometry is a consolidated field, which is well covered in textbooks [17, 10]. State-of-the-art systems [14, 42] allow for reconstructions of large scenes including whole cities. They consist of a long pipeline of methods, starting with descriptor matching for finding a sparse set of correspondences between images [26], followed by estimating the essential matrix to determine the camera motion. Outliers among the correspondences are typically filtered out via RANSAC [11]. Although these systems use bundle adjustment [39] to jointly optimize camera poses and structure of many images, they depend on the quality of the estimated geometry between image pairs for initialization. Only after estimation of the camera motion and a sparse 3D point cloud, dense depth maps are computed by exploiting the epipolar geometry [4]. LSD-SLAM [9] deviates from this approach by jointly optimizing semi-dense correspondences and depth maps. It considers multiple frames from a short temporal window but does not include bundle adjustment. DTAM [30] can track camera poses reliably for critical motions by matching against dense depth maps. However, an external depth map initialization is required, which in turn relies on classic structure and motion methods.

Camera motion estimation from dense correspondences has been proposed by Valgaerts et al. [41]. In this paper, we deviate completely from these previous approaches by training a single deep network that includes computation of dense correspondences, estimation of depth, and the camera motion between two frames.

Eigen et al. [7] trained a ConvNet to predict depth from a single image. Depth prediction from a single image is an inherently ill-posed problem which can only be solved using priors and semantic understanding of the scene – tasks ConvNets are known to be very good at. Liu et al. [24] combined a ConvNet with a superpixel-based conditional random field, yielding improved results. Our two-frame network also learns to exploit the same cues and priors as the single-frame networks, but in addition it makes use of a pair of images and the motion parallax between those. This enables generalization to arbitrary new scenes.

ConvNets have been trained to replace the descriptor matching module in aforementioned SfM systems [6, 44]. The same idea was used by Žbontar and LeCun [45] to estimate dense disparity maps between stereo images. Computation of dense correspondences with a ConvNet that is trained end-to-end on the task, was presented by Dosovitskiy et al. [5]. Mayer et al. [28] applied the same concept to dense disparity estimation in stereo pairs. We, too, make use of the FlowNet idea [5], but in contrast to [28, 45], the motion between the two views is not fixed, but must be estimated to derive depth estimates. This makes the learning problem much more difficult.

Flynn et al. [12] implicitly estimated the 3D structure of a scene from a monocular video using a convolutional network. They assume known camera poses – a large simplification which allows them to use the plane-sweeping approach to interpolate between given views of the scene. Moreover, they never explicitly predict the depth, only RGB images from intermediate viewpoints.

Agrawal et al. [2] and Jayaraman & Grauman [19] applied ConvNets to estimating camera motion. The main focus of these works is not on the camera motion itself, but on learning a feature representation useful for recognition. The accuracy of the estimated camera motion is not competitive with classic methods. Kendall et al. [21] trained a ConvNet for camera relocalization — predicting the location of the camera within a known scene from a single image. This is mainly an instance recognition task and requires retraining

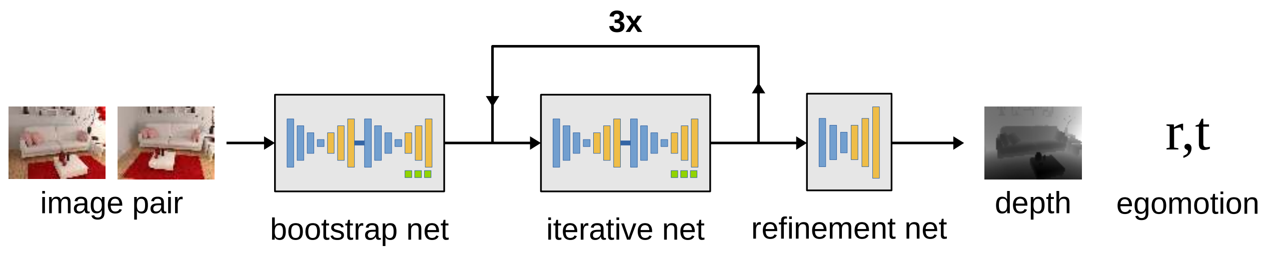


Figure 2. Overview of the architecture. DeMoN takes an image pair as input and predicts the depth map of the first image and the relative pose of the second camera. The network consists of a chain of encoder-decoder networks that iterate over optical flow, depth, and egomotion estimation; see Fig. 3 for details. The refinement network increases the resolution of the final depth map.

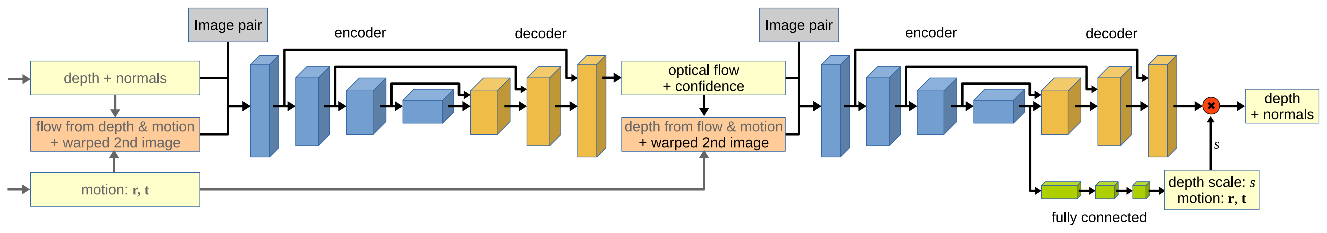


Image pair
encoder
decoder
depth + normals
flow from depth & motion + warped 2nd image
motion: r, t
optical flow + confidence
depth from flow & motion + warped 2nd image
Image pair
encoder
decoder
depth + normals
s
depth scale: s motion: r, t
fully connected

Figure 3. Schematic representation of the encoder-decoder pair used in the bootstrapping and iterative network. Inputs with gray font are only available for the iterative network. The first encoder-decoder predicts optical flow and its confidence from an image pair and previous estimates. The second encoder-decoder predicts the depth map and surface normals. A fully connected network appended to the encoder estimates the camera motion $\mathbf{r}, \mathbf{t}$ and a depth scale factor s. The scale factor s relates the scale of the depth values to the camera motion.

for each new scene. All these works do not provide depth estimates.

3. Network Architecture

The overall network architecture is shown in Fig. 2. DeMoN is a chain of encoder-decoder networks solving different tasks. The architecture consists of three main components: the bootstrap net, the iterative net and the refinement net. The first two components are pairs of encoder-decoder networks, where the first one computes optical flow while the second one computes depth and camera motion; see Fig. 3. The iterative net is applied recursively to successively refine the estimates of the previous iteration. The last component is a single encoder-decoder network that generates the final upsampled and refined depth map.

Bootstrap net. The bootstrap component gets the image pair as input and outputs the initial depth and motion estimates. Internally, first an encoder-decoder network computes optical flow and a confidence map for the flow (the left part of Fig. 3). The encoder consists of pairs of convolutional layers with 1D filters in y and x-direction. Using pairs of 1D filters as suggested in [37] allows us to use spatially large filter while keeping the number of parameters and runtime manageable. We gradually reduce the spatial resolution with a stride of 2 while increasing the number of channels. The decoder part generates the optical flow estimate from the encoder's representation via a series of up-convolutional layers with stride 2 followed by two convolutional layers. It outputs two components of the optical

Method	L1-inv	sc-inv	L1-rel
Single image	0.080	0.159	0.696
Naïve image pair	0.079	0.165	0.722
DeMoN	**0.012**	**0.131**	**0.097**

Table 1. Naïve two-frame depth estimation does not perform better than depth from a single image on any of the error measures (smaller is better). The architecture of DeMoN forces the network to use both images, yielding a large performance improvement.

flow field and an estimate of their confidence. Details on the loss and the training procedure are described in Section 5.

The second encoder-decoder, shown in the right part of Fig. 3, takes as input the optical flow, its confidence, the image pair, and the second image warped with the estimated flow field. Based on these inputs it estimates depth, surface normals, and camera motion. The architecture is the same as above, apart from the extra 3 fully connected layers that compute the camera motion and a scaling factor for the depth prediction. The latter reflects the inherent connection between depth and motion predictions due to scale ambiguity; see Section 4.

By feeding optical flow estimate into the second encoder-decoder we let it make use of motion parallax. Tab. 1 shows that an encoder-decoder network trained to estimate depth and camera motion directly from an image pair (naïve image pair) fails to make use of stereo cues and performs on par with a single-image network. DeMoN, on the other hand, performs significantly better.

Iterative net. The iterative net is trained to improve existing depth, normal, and motion estimates. The architecture of this encoder-decoder pair is identical to the bootstrap net, but it takes additional inputs. We convert the depth map and camera motion estimated by the bootstrap net or a previous iteration of the iterative net into an optical flow field, and feed it into the first encoder-decoder together with other inputs. Likewise, we convert the optical flow to a depth map using the previous camera motion prediction and pass it along with the optical flow to the second encoder-decoder. In both cases the networks are presented with a prediction proposal generated from the predictions of the previous encoder-decoder.

Fig. 4 shows how the optical flow and depth improve with each iteration of the network. The iterations enable sharp discontinuities, improve the scale of the depth values, and can even correct wrong estimates of the initial bootstrapping network. The improvements largely saturate after 3 or 4 iterations. Quantitative analysis is shown in the supplementary material.

During training we simulate 4 iterations by appending predictions from previous training iterations to the minibatch. Unlike unrolling, there is no backpropagation of the gradient through iterations. Instead, the gradient of each iteration is described by the losses on the well defined network outputs: optical flow, depth, normals, and camera motion. Compared to backpropagation through time this saves a lot of memory and allows us to have a larger network and more iterations. A similar approach was taken by Li *et al.* [23], who train each iteration in a separate step and therefore need to store predictions as input for the next stage. We also train the first iteration on its own, but then train all iterations jointly which avoids intermediate storage.

Refinement net. While the previous network components operate at a reduced resolution of 64×48 to save parameters and to reduce training and test time, the final refinement net upscales the predictions to the full input image resolution (256×192). It gets as input the full resolution first image and the nearest-neighbor-upsampled depth and normal field. Fig. 5 shows the low-resolution input and the refined high-resolution output.

A forward pass through the network with 3 iterations takes 110ms on an Nvidia GTX Titan X. Implementation details and exact network definitions of all network components are provided in the supplementary material.

4. Depth and Motion Parameterization

The network computes the depth map in the first view and the camera motion to the second view. We represent the relative pose of the second camera with $\mathbf{r}, \mathbf{t} \in \mathbb{R}^3$. The rotation $\mathbf{r} = \theta \mathbf{v}$ is an angle axis representation with angle θ and axis $\mathbf{v}$. The translation $\mathbf{t}$ is given in Cartesian coordinates.

It is well-known that the reconstruction of a scene from

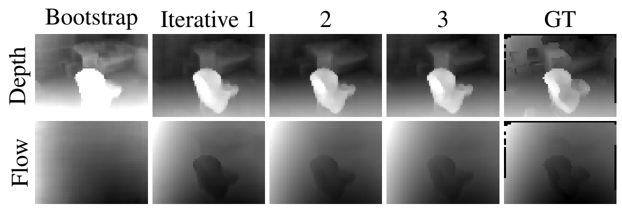


Figure 4. **Top:** Iterative depth refinement. The bootstrap net fails to accurately estimate the scale of the depth. The iterations refine the depth prediction and strongly improve the scale of the depth values. The L1-inverse error drops from 0.0137 to 0.0072 after the first iteration. **Bottom:** Iterative refinement of optical flow. Images show the x component of the optical flow for better visibility. The flow prediction of the bootstrap net misses the object completely. Motion edges are retrieved already in the first iteration and the endpoint error is reduced from 0.0176 to 0.0120.

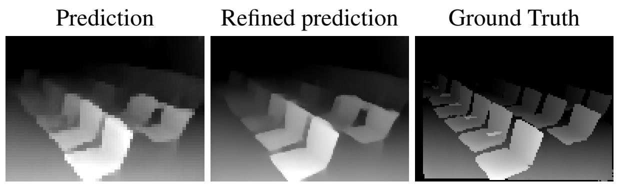


Figure 5. The refinement net generates a high-resolution depth map (256×192) from the low-resolution estimate (64×48) and the input image. The depth sampling preserves depth edges and can even repair wrong depth measurements.

images with unknown camera motion can be determined only up to scale. We resolve the scale ambiguity by normalizing translations and depth values such that $\|\mathbf{t}\| = 1$. This way the network learns to predict a unit norm translation vector.

Rather than the depth z, the network estimates the inverse depth $\xi = 1/z$. The inverse depth allows representation of points at infinity and accounts for the growing localization uncertainty of points with increasing distance.To match the unit translation, our network predicts a scalar scaling factor s, which we use to obtain the final depth values $s\xi$.

5. Training Procedure

5.1. Loss functions

The network estimates outputs of very different nature: high-dimensional (per-pixel) depth maps and low-dimensional camera motion vectors. The loss has to balance both of these objectives, and stimulate synergy of the two tasks without over-fitting to a specific scenario.

Point-wise losses. We apply point-wise losses to our outputs: inverse depth ξ, surface normals $\mathbf{n}$, optical flow $\mathbf{w}$, and optical flow confidence $\mathbf{c}$. For depth we use an L1 loss directly on the inverse depth values:

$$\mathcal{L}_{\text{depth}} = \textstyle\sum_{i,j} |s\xi(i,j) - \hat{\xi}(i,j)|, \qquad (1)$$

with ground truth $\hat{\xi}$. Note that we apply the predicted scale s to the predicted values ξ.

For the loss function of the normals and the optical flow we use the (non-squared) L2 norm to penalize deviations from the respective ground truths $\hat{\mathbf{n}}$ and $\hat{\mathbf{w}}$

$$\begin{aligned}\mathcal{L}_{\text{normal}} &= \sum_{i,j} \|\mathbf{n}(i,j) - \hat{\mathbf{n}}(i,j)\|_2 \\ \mathcal{L}_{\text{flow}} &= \sum_{i,j} \|\mathbf{w}(i,j) - \hat{\mathbf{w}}(i,j)\|_2 \,.\end{aligned} \tag{2}$$

For optical flow this amounts to the usual endpoint error.

We train the network to assess the quality of its own flow prediction by predicting a confidence map for each optical flow component. The ground truth of the confidence for the x component is

$$\hat{c}_x(i,j) = e^{-|\mathbf{w}_x(i,j) - \hat{\mathbf{w}}_x(i,j)|}, \tag{3}$$

and the corresponding loss function reads as

$$\mathcal{L}_{\text{flow confidence}} = \sum_{i,j} |c_x(i,j) - \hat{c}_x(i,j)| \,. \tag{4}$$

Motion losses. We use a minimal parameterization of the camera motion with 3 parameters for rotation $\mathbf{r}$ and translation $\mathbf{t}$ each. The losses for the motion vectors are

$$\begin{aligned}\mathcal{L}_{\text{rotation}} &= \|\mathbf{r} - \hat{\mathbf{r}}\|_2 \\ \mathcal{L}_{\text{translation}} &= \|\mathbf{t} - \hat{\mathbf{t}}\|_2 .\end{aligned} \tag{5}$$

The translation ground truth is always normalized such that $\|\hat{\mathbf{t}}\|_2 = 1$, while the magnitude of $\hat{\mathbf{r}}$ encodes the angle of the rotation.

Scale invariant gradient loss. We define a discrete scale invariant gradient $\mathbf{g}$ as

$$\mathbf{g}_h[f](i,j) = \left(\frac{f(i+h,j) - f(i,j)}{|f(i+h,j)| + |f(i,j)|}, \frac{f(i,j+h) - f(i,j)}{|f(i,j+h)| + |f(i,j)|} \right)^\top . \tag{6}$$

Based on this gradient we define a scale invariant loss that penalizes relative depth errors between neighbouring pixels:

$$\mathcal{L}_{\text{grad}\,\xi} = \sum_{h \in \{1,2,4,8,16\}} \sum_{i,j} \left\| \mathbf{g}_h[\xi](i,j) - \mathbf{g}_h[\hat{\xi}](i,j) \right\|_2 . \tag{7}$$

To cover gradients at different scales we use 5 different spacings h. This loss stimulates the network to compare depth values within a local neighbourhood for each pixel. It emphasizes depth discontinuities, stimulates sharp edges in the depth map and increases smoothness within homogeneous regions as seen in Fig. 10. Note that due to the relation $\mathbf{g}_h[\xi](i,j) = -\mathbf{g}_h[z](i,j)$ for $\xi, z > 0$, the loss is the same for the actual non-inverse depth values z.

We apply the same scale invariant gradient loss to each component of the optical flow. This enhances the smoothness of estimated flow fields and the sharpness of motion discontinuities.

Weighting. We individually weigh the losses to balance their importance. The weight factors were determined empirically and are listed in the supplementary material.

5.2. Training Schedule

The network training is based on the Caffe framework [20]. We train our model from scratch with Adam [22] using a momentum of 0.9 and a weight decay of 0.0004. The whole training procedure consists of three phases.

First, we sequentially train the four encoder-decoder components in both bootstrap and iterative nets for 250k iterations each with a batch size of 32. While training an encoder-decoder we keep the weights for all previous components fixed. For encoder-decoders predicting optical flow, the scale invariant loss is applied after 10k iterations.

Second, we train only the encoder-decoder pair of the iterative net. In this phase we append outputs from previous three training iterations to the minibatch. In this phase the bootstrap net uses batches of size 8. The outputs of the previous three network iterations are added to the batch, which yields a total batch size of 32 for the iterative network. We run 1.6 million training iterations.

Finally, the refinement net is trained for 600k iterations with all other weights fixed. The details of the training process, including the learning rate schedules, are provided in the supplementary material.

6. Experiments

6.1. Datasets

SUN3D [43] provides a diverse set of indoor images together with depth and camera pose. The depth and camera pose on this dataset are not perfect. Thus, we sampled image pairs from the dataset and automatically discarded pairs with a high photoconsistency error. We split the dataset so that the same scenes do not appear in both the training and the test set.

RGB-D SLAM [36] provides high quality camera poses obtained with an external motion tracking system. Depth maps are disturbed by measurement noise, and we use the same preprocessing as for SUN3D. We created a training and a test set.

MVS includes several outdoor datasets. We used the Citywall and Achteckturm datasets from [15] and the Breisach dataset [40] for training and the datasets provided with COLMAP [33, 34] for testing. The depth maps of the reconstructed scenes are often sparse and can comprise reconstruction errors.

Scenes11 is a synthetic dataset with generated images of virtual scenes with random geometry, which provide perfect depth and motion ground truth, but lack realism.

Thus, we introduce the **Blendswap** dataset which is based on 150 scenes from `blendswap.com`. The dataset provides a large variety of scenes, ranging from cartoon-like to photorealistic scenes. The dataset contains mainly indoor scenes. We used this dataset only for training.

NYUv2 [29] provides depth maps for diverse indoor scenes but lacks camera pose information. We did not train on NYU and used the same test split as in Eigen et al. [7]. In contrast to Eigen et al., we also require a second input image that should not be identical to the previous one. Thus, we automatically chose the next image that is sufficiently different from the first image according to a threshold on the difference image.

In all cases where the surface normals are not available, we generated them from the depth maps. We trained DeMoN specifically for the camera intrinsics used in SUN3D and adapted all other datasets by cropping and scaling to match these parameters.

6.2. Error metrics

While single-image methods aim to predict depth at the actual physical scale, two-image methods typically yield the scale relative to the norm of the camera translation vector. Comparing the results of these two families of methods requires a scale-invariant error metric. We adopt the scale-invariant error of [8], which is defined as

$$\text{sc-inv}(z, \hat{z}) = \sqrt{\tfrac{1}{n} \textstyle\sum_i d_i^2 - \tfrac{1}{n^2} \left(\textstyle\sum_i d_i\right)^2}, \qquad (8)$$

with $d_i = \log z_i - \log \hat{z}_i$. For comparison with classic structure from motion methods we use the following measures:

$$\text{L1-rel}(z, \hat{z}) = \tfrac{1}{n} \textstyle\sum_i \frac{|z_i - \hat{z}_i|}{\hat{z}_i} \qquad (9)$$

$$\text{L1-inv}(z, \hat{z}) = \tfrac{1}{n} \textstyle\sum_i |\xi_i - \hat{\xi}_i| = \tfrac{1}{n} \textstyle\sum_i \left| \frac{1}{z_i} - \frac{1}{\hat{z}_i} \right| \qquad (10)$$

L1-rel computes the depth error relative to the ground truth depth and therefore reduces errors where the ground truth depth is large and increases the importance of close objects in the ground truth. L1-inv behaves similarly and resembles our loss function for predicted inverse depth values (1).

For evaluating the camera motion estimation, we report the angle (in degrees) between the prediction and the ground truth for both the translation and the rotation. The length of the translation vector is 1 by definition.

The accuracy of optical flow is measured by the average endpoint error (EPE), that is, the Euclidean norm of the difference between the predicted and the true flow vector, averaged over all image pixels. The flow is scaled such that the displacement by the image size corresponds to 1.

6.3. Comparison to classic structure from motion

We compare to several strong baselines implemented by us from state-of-the-art components ("Base-*"). For these baselines, we estimated correspondences between images, either by matching SIFT keypoints ("Base-SIFT") or with the FlowFields optical flow method from Bailer et al. [3] ("Base-FF"). Next, we computed the essential matrix with the normalized 8-point algorithm [16] and RANSAC. To further improve accuracy we minimized the reprojection error using the *ceres* library [1]. Finally, we generated the depth maps by plane sweep stereo and used the approach of Hirschmueller et al. [18] for optimization. We also report the accuracy of the depth estimate when providing the ground truth camera motion ("Base-Oracle"). ("Base-Matlab") and ("Base-Mat-F") are implemented in Matlab. ("Base-Matlab") uses Matlab implementations of the KLT algorithm [38, 27, 35] for correspondence search while ("Base-Mat-F") uses the predicted flow from DeMoN. The essential matrix is computed with RANSAC and the 5-point algorithm [31] for both.

		Depth			Motion			Depth
	Method	L1-inv	sc-inv	L1-rel	rot	trans	Method	sc-inv
MVS	Base-Oracle	**0.019**	**0.197**	**0.105**	0	0		
	Base-SIFT	0.056	0.309	0.361	21.180	60.516		
	Base-FF	0.055	0.308	0.322	**4.834**	17.252	Liu indoor	0.260
	Base-Matlab	-	-	-	10.843	32.736	Liu outdoor	0.341
	Base-Mat-F	-	-	-	5.442	18.549	Eigen VGG	0.225
	DeMoN	0.047	0.202	0.305	5.156	**14.447**	DeMoN	**0.203**
Scenes11	Base-Oracle	0.023	0.618	0.349	0	0		
	Base-SIFT	0.051	0.900	1.027	6.179	56.650		
	Base-FF	0.038	0.793	0.776	1.309	19.425	Liu indoor	0.816
	Base-Matlab	-	-	-	0.917	14.639	Liu outdoor	0.814
	Base-Mat-F	-	-	-	2.324	39.055	Eigen VGG	0.763
	DeMoN	**0.019**	**0.315**	**0.248**	**0.809**	**8.918**	DeMoN	**0.303**
RGB-D	Base-Oracle	**0.026**	0.398	0.336	0	0		
	Base-SIFT	0.050	0.577	0.703	12.010	56.021		
	Base-FF	0.045	0.548	0.613	4.709	46.058	Liu indoor	0.338
	Base-Matlab	-	-	-	12.831	49.612	Liu outdoor	0.428
	Base-Mat-F	-	-	-	2.917	22.523	Eigen VGG	0.272
	DeMoN	0.028	**0.130**	**0.212**	**2.641**	**20.585**	DeMoN	**0.134**
Sun3D	Base-oracle	0.020	0.241	0.220	0	0		
	Base-SIFT	0.029	0.290	0.286	7.702	41.825		
	Base-FF	0.029	0.284	0.297	3.681	33.301	Liu indoor	0.214
	Base-Matlab	-	-	-	5.920	32.298	Liu outdoor	0.401
	Base-Mat-F	-	-	-	2.230	26.338	Eigen VGG	0.175
	DeMoN	**0.019**	**0.114**	**0.172**	**1.801**	**18.811**	DeMoN	**0.126**
NYUv2	Base-oracle	-	-	-	-	-		
	Base-SIFT	-	-	-	-	-		
	Base-FF	-	-	-	-	-	Liu indoor	0.210
	Base-Matlab	-	-	-	-	-	Liu outdoor	0.421
	Base-Mat-F	-	-	-	-	-	Eigen VGG	**0.148**
	DeMoN	-	-	-	-	-	DeMoN	0.180

Table 2. **Left:** Comparison of two-frame depth and motion estimation methods. Lower is better for all measures. For a fair comparison with the baseline methods, we evaluate depth only at pixels visible in both images. For Base-Matlab depth is only available as a sparse point cloud and is therefore not compared to here. We do not report the errors on NYUv2 since motion ground truth (and therefore depth scale) is not available. **Right:** Comparison to single-frame depth estimation. Since the scale estimates are not comparable, we report only the scale invariant error metric.

Tab. 2 shows that DeMoN outperforms all baseline methods both on motion and depth accuracy by a factor of 1.5 to 2 on most datasets. The only exception is the MVS dataset where the motion accuracy of DeMoN is on par with the strong baseline based on FlowFields optical flow. This demonstrates that traditional methods work well on the texture rich scenes present in MVS, but do not perform well for example on indoor scenes, with large homogeneous regions or small baselines where priors may be very useful. Besides

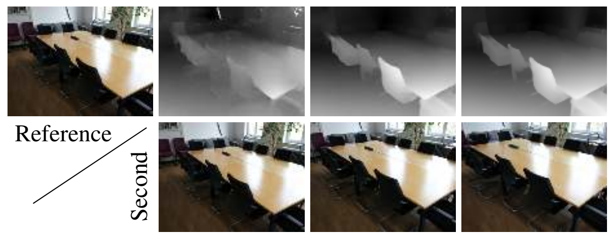


Figure 6. Qualitative performance gain by increasing the baseline between the two input images for DeMoN. The depth map is produced with the top left reference image and the second image below. The first output is obtained with two identical images as input, which is a degenerate case for traditional structure from motion.

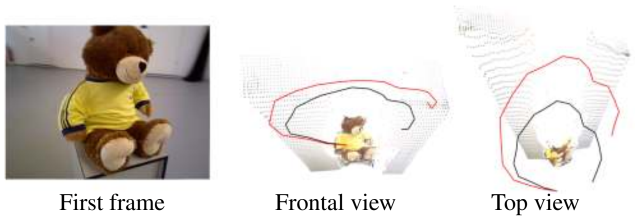


Figure 7. Result on a sequence of the RGB-D SLAM dataset [36]. The accumulated pairwise pose estimates by our network (red) are locally consistent with the ground truth trajectory (black). The depth prediction of the first frame is shown. The network also separates foreground and background in its depth output.

that, all Base-* methods use images at the full resolution of 640×480, while our method uses downsampled images of 256×192 as input. Higher resolution gives the Base-* methods an advantage in depth accuracy, but on the other hand these methods are more prone to outliers. For detailed error distributions see the supplemental material. Remarkably, on all datasets except for MVS the depth estimates of DeMoN are better than the ones a traditional approach can produce given the ground truth motion. This is supported by qualitative results in Fig. 8. We also note that DeMoN has smaller motion errors than ("Base-Mat-F"), showing its advantage over classical methods in motion estimation.

In contrast to classical approaches, we can also handle cases without and with very little camera motion, see Fig. 6. We used our network to compute camera trajectories by simple concatenation of the motion of consecutive frames, as shown in Fig. 7. The trajectory shows mainly translational drift. We also did not apply any drift correction which is a crucial component in SLAM systems, but results convince us that DeMoN can be integrated into such systems.

6.4. Comparison to depth from single image

To demonstrate the value of the motion parallax, we additionally compare to the single-image depth estimation methods by Eigen & Fergus [7] and Liu et al. [24]. We compare to the improved version of the Eigen & Fergus method, which is based on the VGG network architecture, and to two

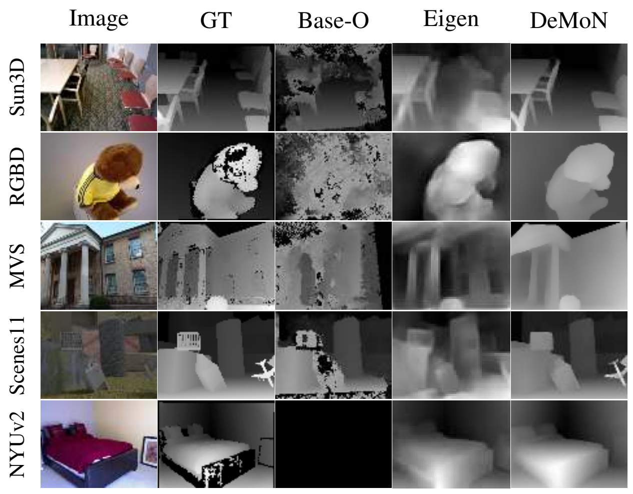


Figure 8. **Top:** Qualitative depth prediction comparison on various datasets. The predictions of DeMoN are very sharp and detailed. The Base-Oracle prediction on NYUv2 is missing because the motion ground truth is not available. Results on more methods and examples are shown in the supplementary material.

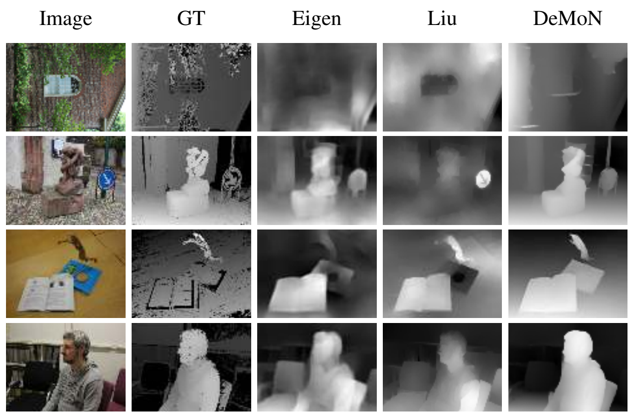
models by Liu et al.: one trained on indoor scenes from the NYUv2 dataset ("indoor") and another, trained on outdoor images from the Make3D dataset [32] ("outdoor").

The comparison in Fig. 8 shows that the depth maps produced by DeMoN are more detailed and more regular than the ones produced by other methods. This becomes even more obvious when the results are visualized as a point cloud; see the videos in the supplemental material.

On all but one dataset, DeMoN outperforms the single-frame methods also by numbers, typically by a large margin. Notably, a large improvement can be observed even on the indoor datasets, Sun3D and RGB-D, showing that the additional stereopsis complements the other cues that can be learned from the large amounts of training data available for this scenario. Only on the NYUv2 dataset, DeMoN is slightly behind the method of Eigen & Fergus. This is because the comparison is not totally fair: the network of Eigen & Fergus as well as Liu indoor was trained on the training set of NYUv2, whereas the other networks have not seen this kind of data before.

6.4.1 Generalization to new data

Scene-specific priors learned during training may be useless or even harmful when being confronted with a scene that is very different from the training data. In contrast, the geometric relations between a pair of images are independent of the content of the scene and should generalize to unknown scenes. To analyze the generalization properties of DeMoN, we compiled a small dataset of images showing uncommon or complicated scenes, for example abstract sculptures, close-ups of people and objects, images rotated by 90 degrees.



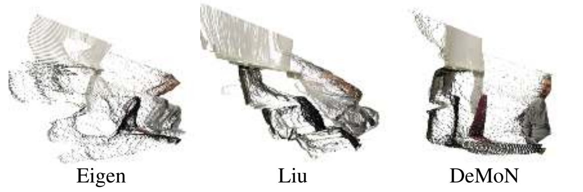


Figure 9. Visualization of DeMoN's generalization capabilities to previously unseen configurations. Single-frame methods have severe problems in such cases, as most clearly visible in the point cloud visualization of the depth estimate for the last example.

Method	L1-inv	sc-inv	L1-rel
Liu [24]	0.055	0.247	0.194
Eigen [7]	0.062	0.238	0.185
DeMoN	**0.041**	**0.183**	**0.130**

Table 3. Quantitative generalization performance on previously unseen scenes, objects, and camera rotations, using a self-recorded and reconstructed dataset. Errors after optimal log-scaling. The best model of Eigen et al. [7] for this task is based on VGG, for Liu et al. [24], the model trained on Make3D [13] performed best. DeMoN achieved best performance after two iterations.

Fig. 9 and Tab. 3 show that DeMoN, as to be expected, generalizes better to these unexpected scenes than single-image methods. It shows that the network has learned to make use of the motion parallax.

6.5. Ablation studies

Our architecture contains some design decisions that we justify by the following ablation studies. All results have been obtained on the Sun3D dataset with the bootstrap net.

Choice of the loss function. Tab. 4 shows the influence of the loss function on the accuracy of the estimated depth and motion. Interestingly, while the scale invariant loss greatly improves the prediction qualitatively (see Fig. 10), it has negative effects on depth scale estimation. This leads to weak performance on non-scale-invariant metrics and the motion accuracy. Estimation of the surface normals slightly improves all results. Finally, the full architecture with the scale invariant loss, normal estimation, and a loss on the flow, leads to the best results.

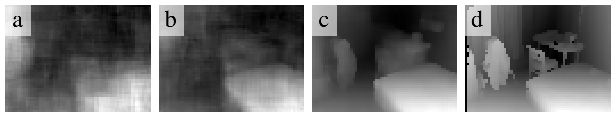


Figure 10. Depth prediction comparison with different outputs and losses. **(a)** Just L1 loss on the absolute depth values. **(b)** Additional output of normals and L1 loss on the normals. **(c)** Like (b) but with the proposed gradient loss. **(d)** Ground truth.

			Depth			Motion	
grad	norm	flow	L1-inv	sc-inv	L1-rel	rot	tran
no	no	no	0.040	0.211	0.354	3.127	30.861
yes	no	no	0.057	0.159	0.437	4.585	39.819
no	yes	no	0.037	0.190	0.336	2.570	29.607
no	yes	yes	**0.029**	0.184	**0.266**	**2.359**	**23.578**
yes	yes	yes	0.032	**0.150**	0.276	2.479	24.372

Table 4. The influence of the loss function on the performance. The gradient loss improves the scale invariant error, but degrades the scale-sensitive measures. Surface normal prediction improves the depth accuracy. A combination of all components leads to the best tradeoff.

	Depth			Motion		Flow
Confidence	L1-inv	sc-inv	L1-rel	rot	tran	EPE
no	0.030	0.028	0.26	2.830	25.262	0.027
yes	0.032	0.027	0.28	2.479	24.372	0.027

Table 5. The influence of confidence prediction on the overall performance of the different outputs.

Flow confidence. Egomotion estimation only requires sparse but high-quality correspondences. Tab. 5 shows that given the same flow, egomotion estimation improves when given the flow confidence as an extra input. Our interpretation is that the flow confidence helps finding most accurate correspondences.

7. Conclusions and Future Work

DeMoN is the first deep network that has learned to estimate depth and camera motion from two unconstrained images. Unlike networks that estimate depth from a single image, DeMoN can exploit the motion parallax, which is a powerful cue that generalizes to new types of scenes, and allows to estimate the egomotion. This network outperforms traditional structure from motion techniques on two frames, since in contrast to those, it is trained end-to-end and learns to integrate other shape from X cues. When it comes to handling cameras with different intrinsic parameters it has not yet reached the flexibility of classic approaches. The next challenge is to lift this restriction and extend this work to more than two images. As in classical techniques, this is expected to significantly improve the robustness and accuracy.

Acknowledgements We acknowledge funding by the ERC Starting Grant VideoLearn, the DFG grant BR-3815/5-1, and the EU project Trimbot2020.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres Solver. 6

[2] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015. 2

[3] C. Bailer, B. Taetz, and D. Stricker. Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation. In *IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015. 6

[4] R. T. Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR '96, 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996*, pages 358–363, June 1996. 2

[5] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015. 2

[6] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1734–1747, Oct 2016. TPAMI-2015-05-0348.R1. 2

[7] D. Eigen and R. Fergus. Predicting Depth, Surface Normals and Semantic Labels With a Common Multi-Scale Convolutional Architecture. In *IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015. 1, 2, 6, 7, 8

[8] 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, *Advances in Neural Information Processing Systems 27*, pages 2366–2374. Curran Associates, Inc., 2014. 1, 6

[9] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014. 2

[10] O. Faugeras. *Three-dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA, USA, 1993. 2

[11] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981. 2

[12] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[13] D. A. Forsyth. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840, May 2009. 8

[14] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a Cloudless Day. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *European Conference on Computer Vision (ECCV)*, number 6314 in Lecture Notes in Computer Science, pages 368–381. Springer Berlin Heidelberg, 2010. 2

[15] S. Fuhrmann, F. Langguth, and M. Goesele. Mve-a multiview reconstruction environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, volume 6, page 8, 2014. 5

[16] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997. 6

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 2

[18] H. Hirschmüller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 807–814, June 2005. 6

[19] D. Jayaraman and K. Grauman. Learning image representations tied to egomotion. In *ICCV*, 2015. 2

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014. 5

[21] A. Kendall and R. Cipolla. Modelling Uncertainty in Deep Learning for Camera Relocalization. In *International Converence on Robotics and Automation (ICRA)*, 2016. 2

[22] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, Dec. 2014. arXiv: 1412.6980. 5

[23] K. Li, B. Hariharan, and J. Malik. Iterative Instance Segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3659–3667, June 2016. 4

[24] F. Liu, C. Shen, G. Lin, and I. Reid. Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. 1, 2, 7, 8

[25] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, Sept. 1981. 2

[26] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004. 2

[27] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. 6

[28] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[29] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *European Conference on Computer Vision (ECCV)*, 2012. 6

[30] R. A. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327, 2011. 2

[31] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004. 6

[32] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *In NIPS 18*. MIT Press, 2005. 7

[33] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[34] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[35] J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994. 6

[36] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 5, 7

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567 [cs]*, Dec. 2015. 3

[38] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991. 6

[39] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment A Modern Synthesis Vision Algorithms: Theory and Practice. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883, pages 153–177. Springer Berlin / Heidelberg, Apr. 2000. 2

[40] B. Ummenhofer and T. Brox. Global, dense multiscale reconstruction for a billion points. In *IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. 5

[41] L. Valgaerts, A. Bruhn, M. Mainberger, and J. Weickert. Dense versus Sparse Approaches for Estimating the Fundamental Matrix. *International Journal of Computer Vision*, 96(2):212–234, Jan. 2012. 2

[42] C. Wu. Towards Linear-Time Incremental Structure from Motion. In *International Conference on 3D Vision (3DV)*, pages 127–134, June 2013. 2

[43] J. Xiao, A. Owens, and A. Torralba. SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1625–1632, Dec. 2013. 2, 5

[44] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[45] J. Žbontar and Y. LeCun. Computing the Stereo Matching Cost With a Convolutional Neural Network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

DeMoN: Depth and Motion Network for Learning Monocular Stereo
– Supplementary Material –

A. Network Architecture Details

Our network is a chain of encoder-decoder networks. Figures 15 and 16 explain the details of the two encoder-decoders used in the bootstrap and iterative net part. Fig. 17 gives implementation details for the refinement net.

The encoder-decoders for the bootstrap and iterative net use additional inputs which come from previous predictions. Some of these inputs, like warped images or depth from optical flow, need to be generated with special layers, which we describe here.

Warped second image We warp the second image using the predicted or generated optical flow field. We compute all values with bilinear interpolation and fill values that fall outside the image boundaries with zeros.

After warping with ground truth optical flow, the second image should look like the first image with the exception of occlusion regions. Comparing first image and warped image allows to assess the quality of the optical flow.

Flow from depth and motion We generate an optical flow estimate based on a depth and camera motion estimate for the first encoder-decoder of the iterative net. This optical flow can be used as an initial estimate, which can be further improved by the iterative net.

Depth from optical flow and motion In contrast to generating optical flow from a depth map and camera motion, computing depth from optical flow and camera motion is not straightforward. To compute the depth value of a pixel, we first project the corresponding point in the second image to the epipolar line and then triangulate the 3D point to retrieve the depth.

This generated depth map provides an estimate which combines optical flow and camera motion. Note that using the estimated camera motion here ensures that the depth values of the estimate are correctly scaled according to the camera motion. In case of ground truth optical flow and camera motion this yields the true depth map.

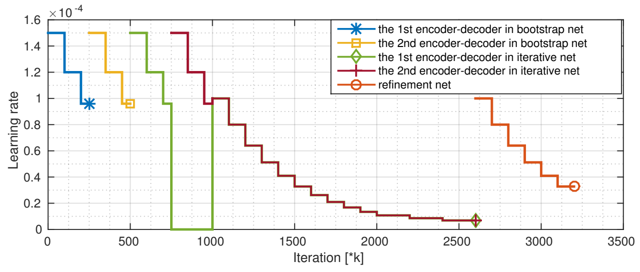


Figure 1. Learning rate schedule. During the training of each encoder-decoder a multistep learning rate is applied. At each step the learning rate is reduced by 20%.

B. Training Schedule

We train our model from scratch for 3200k iterations in total. Fig. 1 shows our learning rate schedule. From 0 to 1000k iterations we sequentially train the four encoder-decoders, then the two encoder-decoders of the iterative net are jointly trained to iteration 2600k, finally we train the refinement net for another 600k iterations. This training schedule can be further optimized.

We normalize all point-wise losses with the number of pixels. The loss weights for flow, flow confidence, depth and normals are 1000, 1000, 300 and 100 respectively. The loss weight for the scale invariant gradient loss on flow is 1000. For depth we weight the scale invariant gradient loss with 1500, because we consider sharp edges and smooth surfaces more important for depth prediction. The translation and rotation losses are weighted with a factor of 15 and 160 respectively to balance their importance with respect to the other losses.

C. Datasets

Our training procedure requires ground truth camera poses and ground truth depth for the first image.

The datasets we use for training can be divided in two groups: synthetic and real datasets. Real datasets provide natural images, but do only provide sparse pseudo ground truth due to measurement and reconstruction errors. Synthetic datasets provide perfect ground truth but often feature unrealistic images. Since no single dataset is perfect, we train on a set of five datasets with different properties. Tab. 1 provides an overview of the properties of the datasets. Figures 11, 12, 13, 14 and 3 show examples from the datasets we used.

SUN3D The dataset from [11] is a set of video sequences with camera poses and depth maps. Camera poses and depth maps are inaccurate because they have been created via a reconstruction process. The raw sensor data for this dataset was recorded with a structured light depth sensor which reports absolute depth values, thus the reconstruction scale for this dataset is known. The dataset features indoor scenes of offices, apartments, hotel rooms and university buildings.

To sample image pairs suitable for training, we apply a simple filtering process. To avoid image pairs with large pose or depth error, we filter out pairs with a large photo-consistency error. We also filter out image pairs with less than 50% of the pixels from the first image visible in the second image. Further we discard images with a baseline smaller than 5cm.

We have split the datasets into training and testing scenes. The training set contains 253 scenes and the test set contains 16 scenes. For training we sample a total of 117768 image pairs from all image sequences. For testing we generate a diverse set of 80 image pairs which minimizes overlap within image sequences. This way we obtain a small test set with large diversity.

RGB-D SLAM The dataset from [9] is a set of video sequences with camera poses and depth maps. Camera poses are accurate since they come from an external motion tracking system. The scale of the reconstructed camera poses is known. Depth maps have been recorded with a structured light sensor and are therefore affected by measurement noise and limited to a fixed depth range. Due to the use of an external tracking system, the data is restricted to indoor scenes in a lab environment.

We use the same sampling process as for SUN3D. The training set contains 45276 image pairs, while the test set features 80 diverse image pairs.

MVS In contrast to SUN3D and RGB-D SLAM, MVS is a dataset with outdoor images. We use the Citywall and Achteckturm datasets provided with [4] and the Breisach dataset from [10] for training. We compute depth maps and camera poses with the Multiview Reconstruction Environment by [4].

For testing we use datasets provided with the COLMAP software from Schönberger *et al.* [7, 8], which are called Person-Hall, Graham-Hall, and South-Building. These datasets show the exterior of the eponymous buildings. Further, we have recorded a small scene which shows a sculpture and can be seen in the third row of Fig. 13. We use the COLMAP software for computing the depth and camera poses for these scenes.

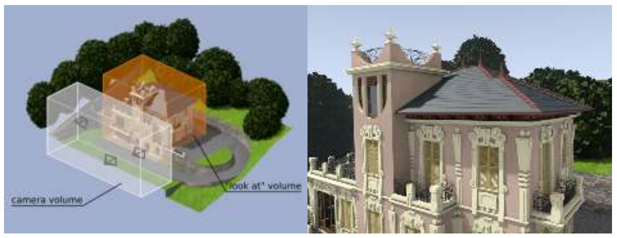


Figure 2. Annotation volumes for the Blendswap dataset. **Left:** We annotate each scene with volumes that describe valid camera positions (white box) and valid positions for the "look at" target (orange box). **Right:** Image from a randomly sampled camera and target.

Due to the scale ambiguity in the reconstruction process, the absolute scale is unknown. The datasets are small and not very diverse. Again we use the same image pair sampling strategy as for the previous datasets. We sample 16152 image pairs for training and 66 image pairs for testing.

Scenes11 Scenes11 is a synthetic dataset with randomly generated geometry and objects from ShapeNet [3]. All scenes have a ground plane which makes the scenes look like outdoor scenes. We use the open source software Blender [2] for rendering.

Due to the synthetic nature of the dataset, the accuracy of the camera poses and depth maps is perfect. Disadvantages are the artificial look and a simplistic model of the camera movement. We sample camera parameters from Gaussian and uniform distributions, which bears the risk of learning the underlying model by heart when training only on this data.

The absolute scale for this dataset is meaningless since the scale of the objects is arbitrary and inconsistent. We generate 239508 image pairs from 19959 unique scenes for training and 128 image pairs from 128 unique scenes for testing.

Blendswap The blendswap dataset is an artificial dataset generated with 150 scenes from `blendswap.com`. Scenes range from cartoon-like to photorealistic. Blendswap contains some outdoor scenes, but a large part of the scenes shows interiors. We annotated each of the 150 scenes with a camera volume, which marks free space that can be used for camera placement, and a "look at" volume for placing target points defining the

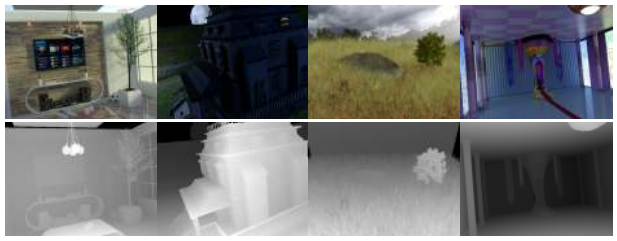


Figure 3. Images and the corresponding depth maps from the Blendswap dataset. Blendswap contains 150 distinct scenes, with a large variety of different styles and settings.

camera viewing direction. We sample camera positions and targets uniformly over the annotated volumes. Fig. 2 shows an example of an annotated scene and an automatically generated image. When sampling a camera pair we use the same target point but add individual uniform noise to its position. The noise level depends on the distance of each camera to this target point.

We sample 34320 image pairs from the 150 annotated scenes for training. Fig. 3 shows images from this dataset and the corresponding ground truth depth maps. The generated data set contains a diverse set of scenes with many realistic images and actual ground truth data, and remedies the main disadvantages of Scenes11. However, adding new scenes to this dataset requires manual annotation, which is a time-consuming process. Due to the small number of scenes we only use this data for training. We plan to gradually extend this dataset in the future with more scenes.

Generalization test data In Section 6.4.1 of the main paper, we compare the generalization capabilities of DeMoN and other learned approaches. To evaluate this, we reconstructed seven self-recorded scenes using COLMAP [7, 8], and generated 16 views of these scenes with corresponding depth maps. Examples are shown in Fig. 4. Content of the scenes is very different from the data DeMoN has been trained on: Close-ups of unique objects such as a miniature bridge, figurine, and ship, as well as a person. We also include some outdoor scenes containing different architecture as well as 90-degree rotated images and a unique sculpture.

Scale normalization: To compensate for the inherent scale ambiguity of our reconstruction, we compute all errors in Tab. 3 of the main paper with optimally scaled depth predictions. The scale $s_{\log} = \exp(\frac{1}{n}\sum \log \hat{z} - \log z)$ is computed to minimize the mean logarithmic difference between ground truth and prediction.

D. Experiments with Ground Truth

DeMoN iterates between estimating optical flow and estimating depth and camera motion. This is supposed to gradually refine the depth and motion estimates. The architecture is justified by a strong mathematical relation between optical flow, depth and camera motion. Optical flow can be computed analytically from depth and motion, and vice-versa, depth and motion can be computed from optical flow in non-degenerate cases. But can a convolutional network fully exploit this mathematical relation? Given perfect optical flow, can it indeed estimate depth and camera motion very accurately?

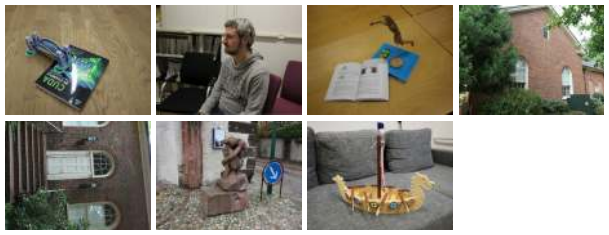


Figure 4. Samples of the seven scenes used for the generalization experiment.

To answer these questions we trained networks to estimate depth and motion from ground truth optical flow, and, other way round, to estimate optical flow given ground truth depth and motion. Results on the SUN3D dataset are reported in Table 2. In all cases performance dramatically improves when the ground truth input is provided, compared to only taking an image pair as input. The network can use an accurate optical flow estimate to refine the depth and motion estimates, and vice-versa.

At the same time, this experiment provides an upper bound on the performance we can expect from the current architecture. DeMoN's performance is still far below this bound, meaning that the difficulties come from estimating optical flow, depth and motion from images, and not from converting between these modalities.

Ground truth		Depth			Motion		Flow
flow	dep+mot	L1-inv	sc-inv	L1-rel	rot	tran	EPE
yes	no	0.007	0.058	0.066	-	-	-
yes	no	-	-	-	0.340	2.235	-
no	no	0.058	0.163	0.603	4.472	41.766	-
no	yes	-	-	-	-	-	0.005
no	no	-	-	-	-	-	0.027

Table 2. The effect of providing the ground truth optical flow (flow) or ground truth depth and motion (dep+mot) to the network. A network can be trained to produce very accurate depth and motion estimates given the ground truth optical flow, and vice-versa, a network can estimate the optical flow very well given the ground truth depth and motion.

Dataset	Perfect GT	Photorealistic	Outdoor scenes	Rot. avg	Rot. stddev	Tri. angle avg	Tri. angle stddev
SUN3D	no	yes	no	10.6	7.5	5.2	4.6
RGBD	no	yes	no	10.4	8.3	6.8	4.5
Scenes11	yes	no	(yes)	3.3	2.1	5.3	4.4
MVS	no	yes	yes	34.3	24.7	28.9	17.5
Blendswap	yes	(yes)	(yes)	23.1	17.1	20.1	13.6

Table 1. Training dataset properties. **Perfect GT:** Perfect ground truth camera poses and depth maps are only available for the synthetic datasets Scenes11 and Blendswap. **Photorealistic:** The synthetic Blendswap dataset features some photorealistic scenes, while Scenes11 looks entirely artificial. The other datasets use real images. **Outdoor scenes:** MVS is the only outdoor dataset with real images. Images from Scenes11 show wide open spaces, similar to outdoor data. Blendswap contains some outdoor scenes, but is biased towards indoor environments. **Rotation and Triangulation angle:** Camera rotation and triangulation angles are given in degree. The rotation and triangulation angle is similar for SUN3D and RGB-D SLAM. Both datasets are indoor video sequences. MVS and Blendswap also show similar characteristics, which means that the sampling procedure for Blendswap mimics the camera poses of a real outdoor dataset.

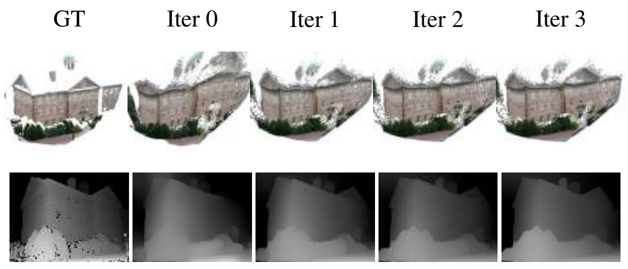


Figure 5. Effect of the iterative net on depth values.

		Depth						Motion		Flow
		L1-inv	sc-inv	L1-rel	$\delta<1.25$	$\delta<1.25^2$	$\delta<1.25^3$	rot	tran	EPE
Iteration	0	0.029	0.145	0.244	0.587	0.844	0.940	2.18	20.27	0.030
	1	0.024	0.130	0.207	0.679	0.891	0.961	1.94	17.25	0.020
	2	0.022	0.131	0.187	0.688	0.900	0.982	1.87	18.31	0.019
	3	0.021	0.132	0.179	0.698	0.912	0.981	1.80	18.81	0.019
	4	0.021	0.133	0.184	0.690	0.908	0.975	1.79	18.94	0.019
	5	0.021	0.133	0.185	0.692	0.910	0.970	1.79	19.65	0.019

Table 3. The effect of iterative refinement of the predictions. The performance is computed on the SUN3D dataset. The results do not significantly improve beyond 3 iterations. The threshold metric is defined as the percentage of pixel z_i so that $\max(\frac{z_i}{\hat{z}_i}, \frac{\hat{z}_i}{z_i}) = \delta < thr$.

E. Effect of Iterative Refinement

The quantitative evaluation of iterative refinement is shown in Table 3. Both depth and motion accuracy significantly improve up to iteration 3.

Fig. 5 shows the effect on one sample of the MVS dataset. The iterative net improves the depth values, which is visible as a reduced distortion of the building in the point cloud.

F. Error Distributions

We show the per-pixel error distributions and means for Base-Oracle and DeMoN on the MVS and SUN3D dataset in Fig. 6. Note that the average values in Tab. 2 in the main paper have been computed over test samples and here we average over pixels.

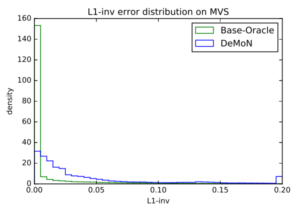
The distributions on the highly textured MVS show that Base-Oracle produces many very accurate depth estimates, while the distribution of errors is more spread for DeMoN. This is an effect of Base-Oracle using higher resolution images (640×480) than DeMoN (256×192). Base-Oracle also uses the motion ground truth, which again helps finding the correct depth.

For SUN3D distributions are more similar and the resolution adavantage of Base-Oracle is less pronounced. This can be explained by the more homogeneous image regions, which make matching difficult on this dataset. Base-Oracle suffers significantly more from outliers than DeMoN. Depending on the task higher depth accuracy can be more important than less outliers. It also shows the importance of lifting restrictions on the camera intrinsics and supporting higher resolutions in future works.

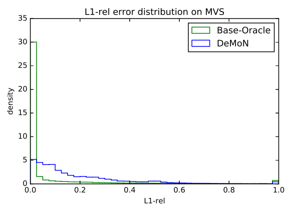
We show the distributon of the motion errors for Base-FF, Base-Mat-F and DeMoN in Fig. 7. Base-FF uses the FlowFields algorithm for correspondence search [1] and our baseline implementation using the noramlized 8-point algorithm [5] and RANSAC to compute the relative camera motion. Base-Mat-F uses the optical flow of DeMoN predicted after three iterations for correspondences and uses the Matlab implementations of [6] and RANSAC to estimate the camera motion.

The distribution is similar for Base-FF and DeMoN on the MVS dataset with Base-FF being slightly more accurate for rotation and DeMoN being more accurate for translation. Base-Mat-F is less accurate than both comparisons.

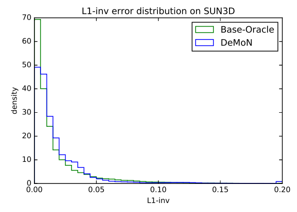
On SUN3D DeMoN can estimate the motion more accurately than the comparisons. Base-FF produces some outliers for rotation while DeMoN and Base-Mat-F have almost no outliers. Base-FF also fails to estimate accurate translation directions on SUN3D. We show some failure cases in Fig. 8. On SUN3D baselines are usually smaller than on MVS.



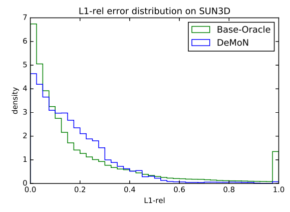
Base-Oracle: mean 0.018 (0.013)
DeMoN: mean 0.045 (0.037)



Base-Oracle: mean 0.103 (0.064)
DeMoN: mean 0.272 (0.177)

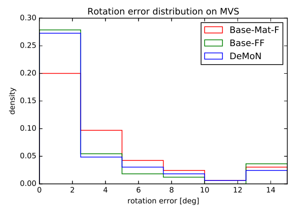


Base-Oracle: mean 0.021 (0.018)
DeMoN: mean 0.019 (0.018)

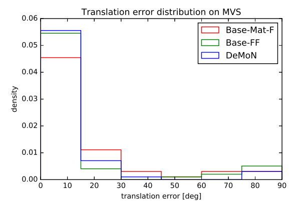


Base-Oracle: mean 0.218 (0.174)
DeMoN: mean 0.172 (0.169)

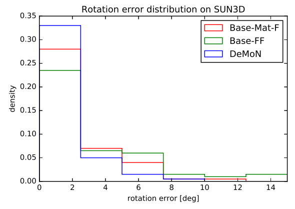
Figure 6. Error histograms and mean for per pixel depth errors (L1-inv and L1-rel) on MVS (top) and SUN3D (bottom). The last bin includes samples with errors above of the shown range. The second mean value in parenthesis excludes the last bin of the respective histogram.



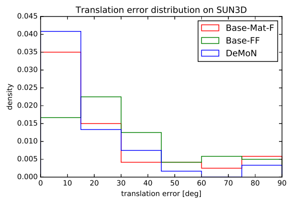
Base-FF: mean 4.834
Base-Mat-F: mean 5.442
DeMoN: mean 5.156



Base-FF: mean 17.252
Base-Mat-F: mean 18.549
DeMoN: mean 14.447



Base-FF: mean 3.681
Base-Mat-F: mean 2.230
DeMoN: mean 1.801



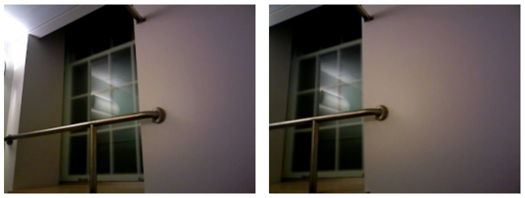
Base-FF: mean 33.301
Base-Mat-F: mean 26.338
DeMoN: mean 18.811

Figure 7. Error distributions for rotation and translation on MVS (top) and SUN3D (bottom). The translation error is given as angle because the length of the translation is 1 by definition. The last bin includes samples with errors above of the shown range.

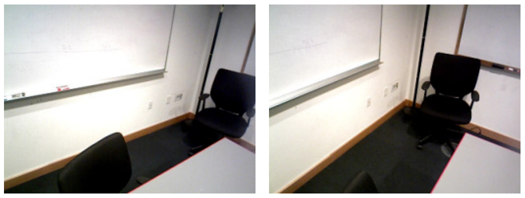
G. Qualitative Depth Estimation Results

We show more qualitative results of depth estimation on the test sets of SUN3D, RGB-D SLAM, MVS, Scenes11 and NYUv2 in Fig. 11 to Fig. 10 respectively. Our method presents smooth depth estimations while preserving sharp edges. This advantage can be observed more clearly by the point clouds, which we show in Fig. 9.

(a) DeMoN: tran 24.096, rot 0.878
Base-FF: tran 71.871, rot 2.564



(b) DeMoN: tran 4.804, rot 1.237
Base-FF: tran 56.948 , rot 2.087



(c) DeMoN: tran 11.725, rot 1.628
Base-FF: tran 110.516, rot 15.197

Figure 8. Comparison of the motion error (in degrees) in some special cases. The classic method Base-FF fails when the baseline of the camera motion is small shown in (a) and the scenes have many homogeneous regions like (b) and (c).

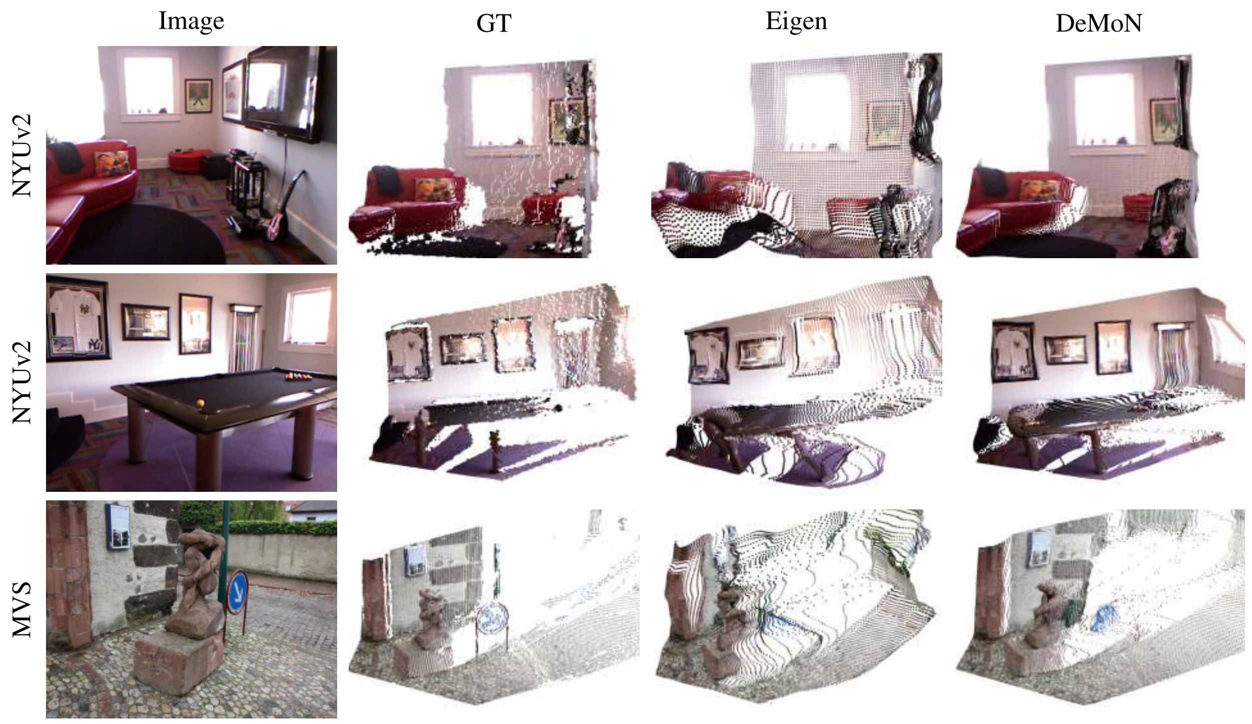


Figure 9. Qualitative point clouds comparison on NYUv2 and MVS.

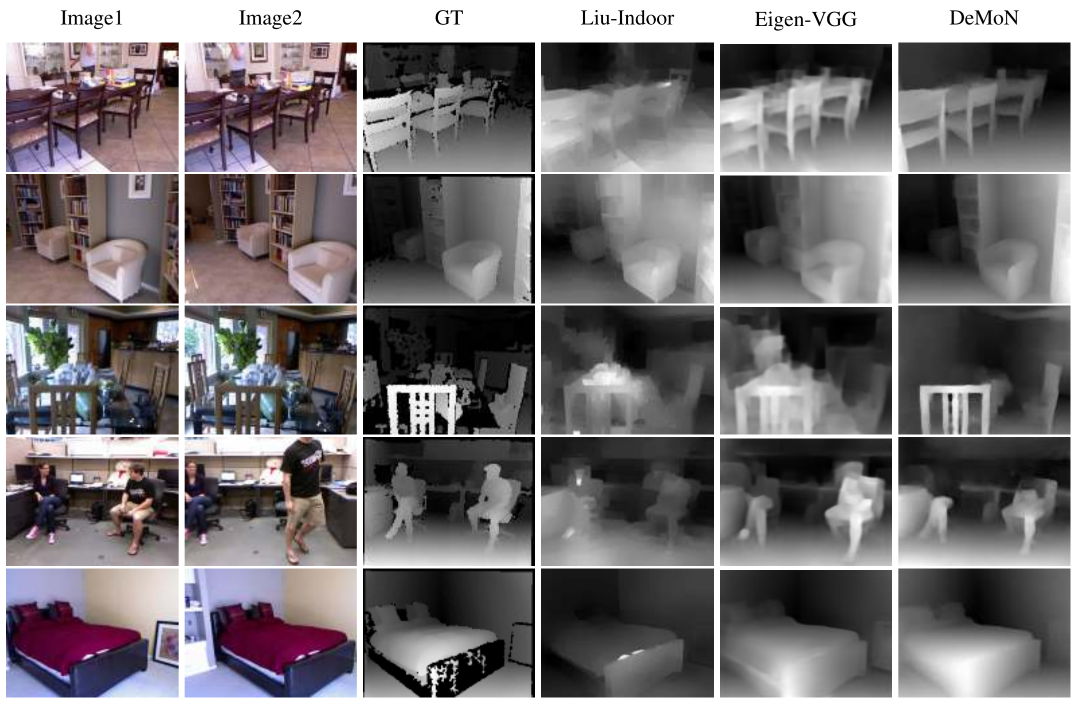


Figure 10. Qualitative depth prediction comparison on NYUv2. The Base-Oracle prediction is not available because there is no motion ground truth. Our method fails to predict the upper bodies of the persons in the fourth example because the persons move between the two frames.

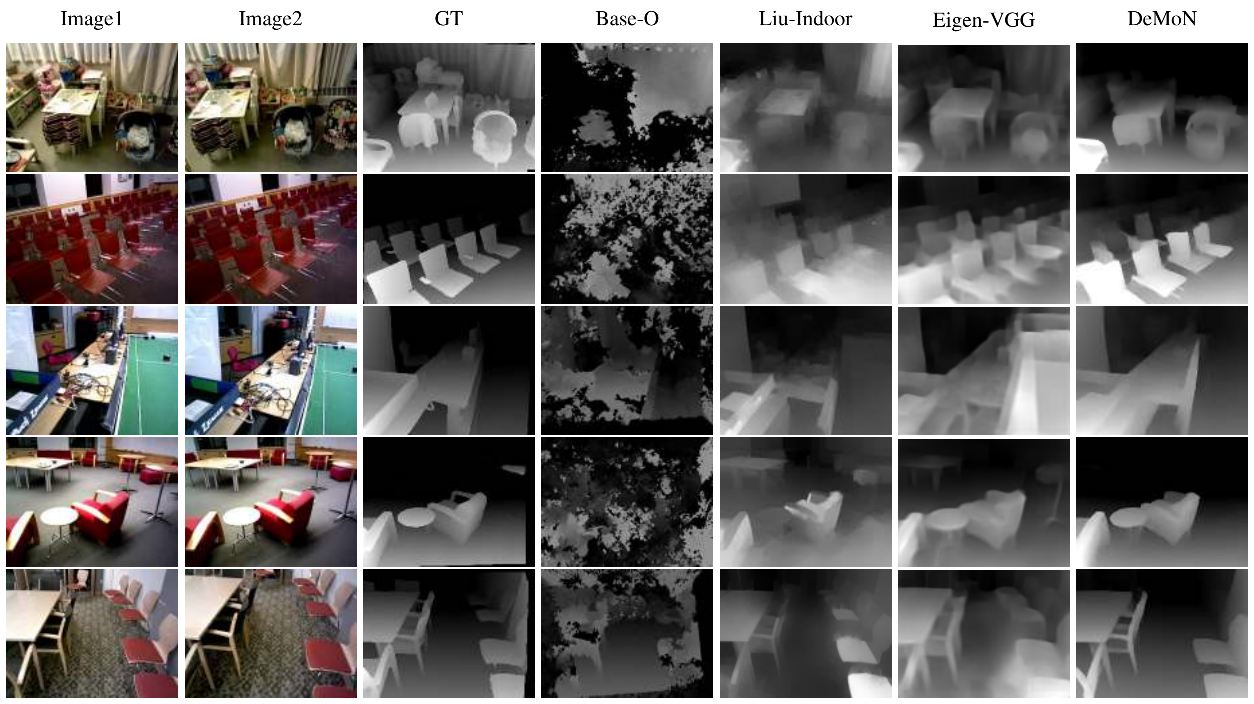


Figure 11. Qualitative depth prediction comparison on SUN3D. The Base-Oracle performs badly due to inaccurate motion ground truth. Eigen-VGG, which was trained on NYUv2, works well for many images. SUN3D is similar to the NYUv2 dataset shown in Fig. 10.

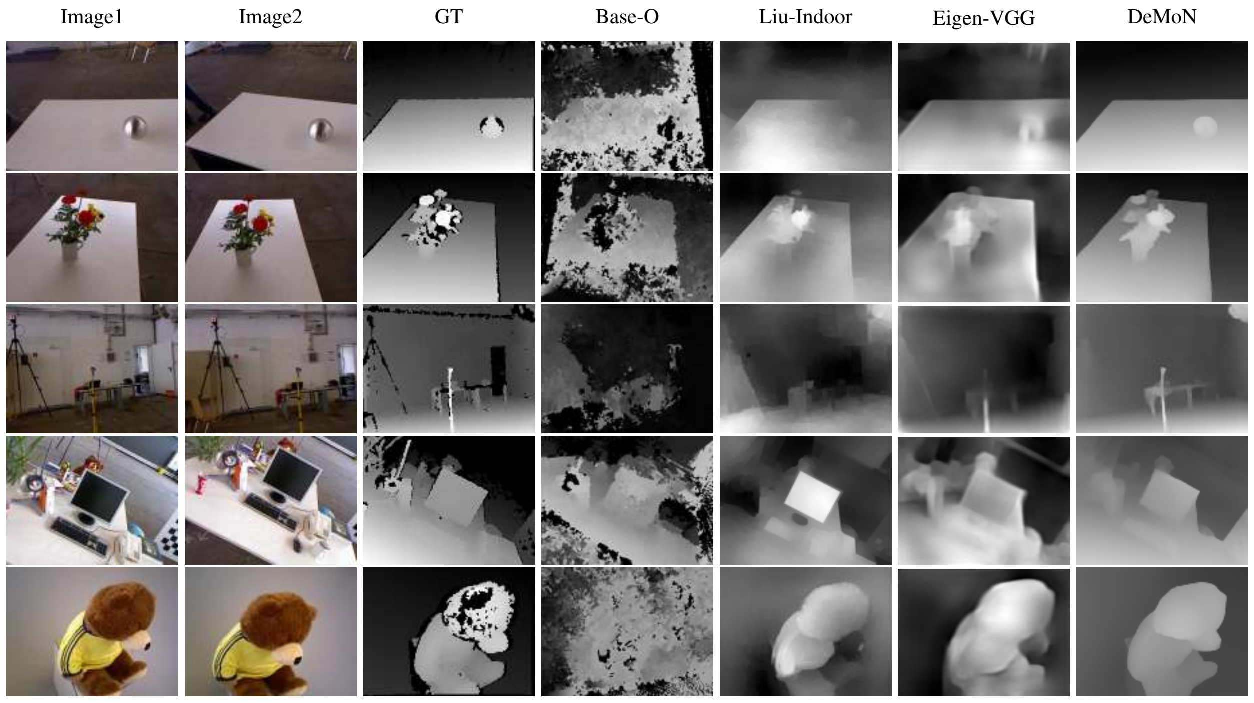


Figure 12. Qualitative depth prediction comparison on RGB-D SLAM. Our method can deal with very thin objects (third row).

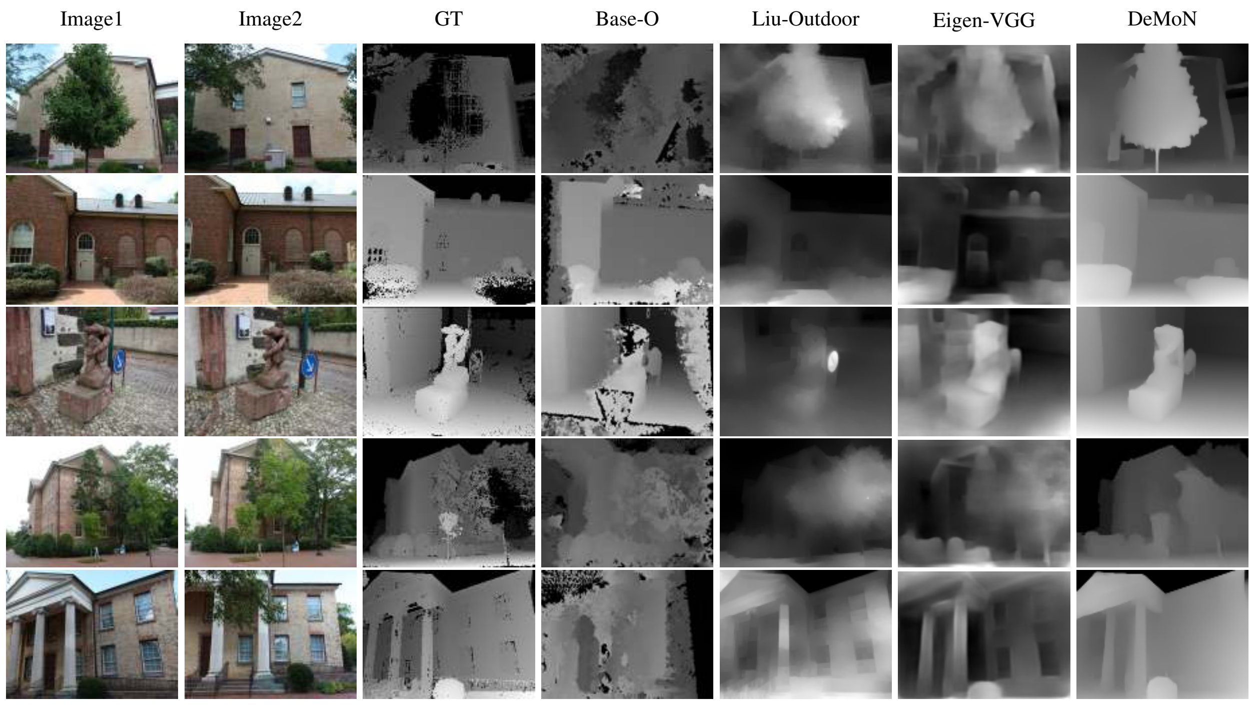


Figure 13. Qualitative depth prediction comparison on MVS. The single image methods Liu-Outdoor and Eigen-VGG do not generalize well to new datasets. The depth maps show coarse outliers caused by hallucinating wrong depth values at object contours like the street sign in the third row or the windows in the last row. The Base-Oracle method performs well on this data. Most outliers fall into image regions not visible in the second image.

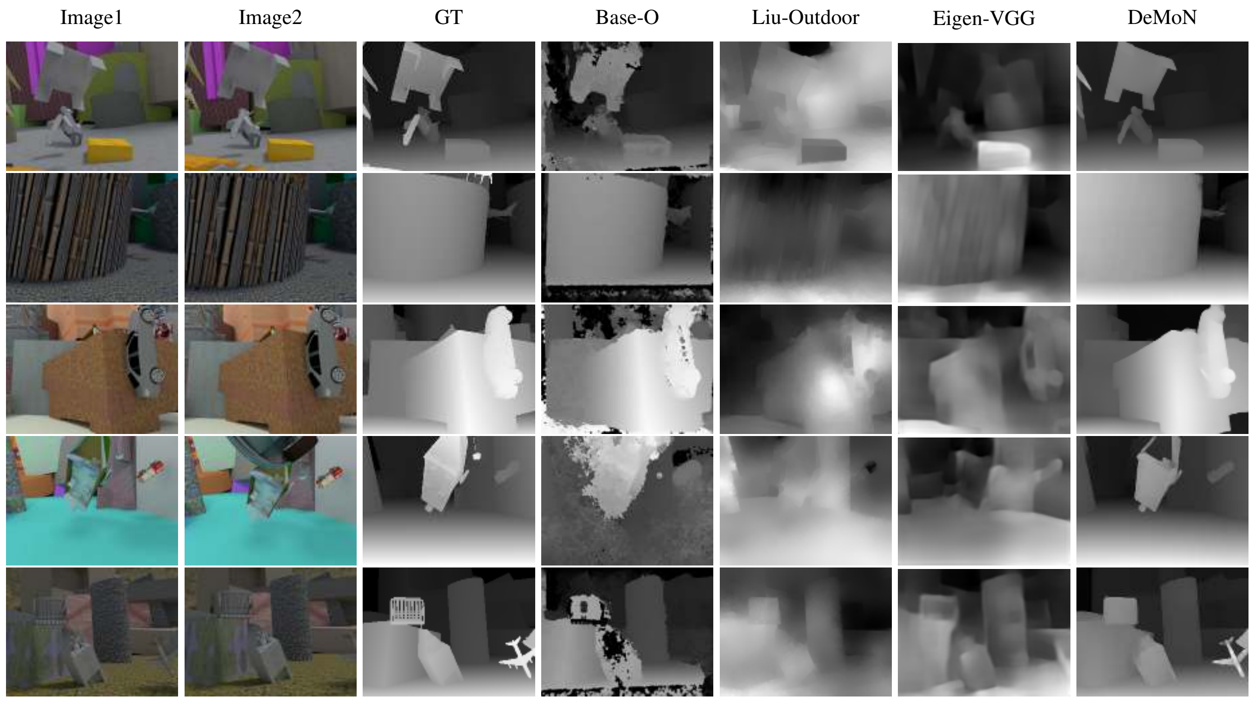


Figure 14. Qualitative depth prediction comparison on Scenes11. Base-Oracle and DeMoN give the best results on this dataset.

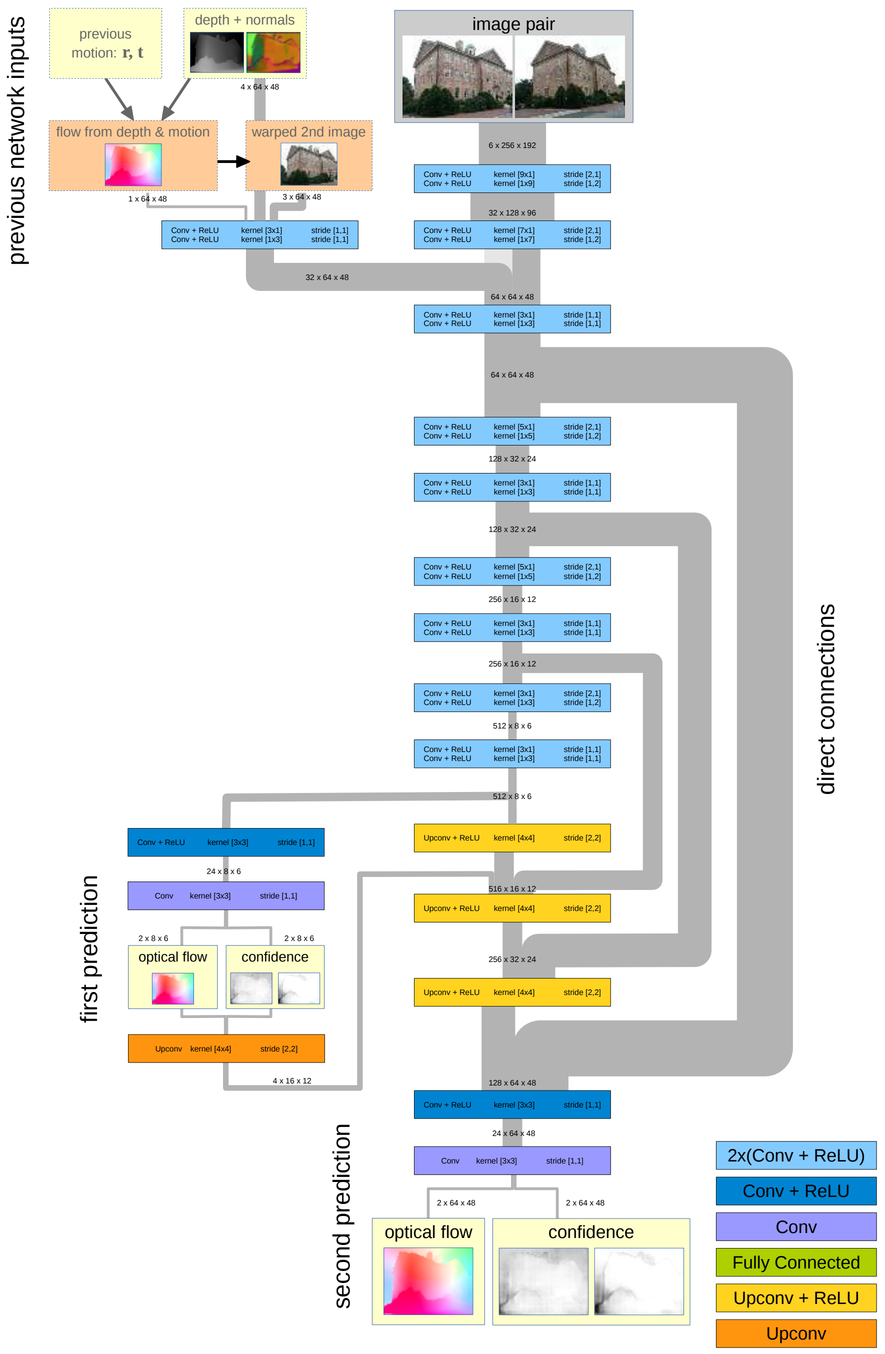

previous network inputs
previous motion: r, t
depth + normals
4 x 64 x 48
flow from depth & motion
warped 2nd image
1 x 64 x 48
3 x 64 x 48
image pair
6 x 256 x 192
Conv + ReLU kernel [9x1] stride [2,1]
Conv + ReLU kernel [1x9] stride [1,2]
32 x 128 x 96
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
Conv + ReLU kernel [7x1] stride [2,1]
Conv + ReLU kernel [1x7] stride [1,2]
32 x 64 x 48
64 x 64 x 48
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
64 x 64 x 48
Conv + ReLU kernel [5x1] stride [2,1]
Conv + ReLU kernel [1x5] stride [1,2]
128 x 32 x 24
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
128 x 32 x 24
Conv + ReLU kernel [5x1] stride [2,1]
Conv + ReLU kernel [1x5] stride [1,2]
256 x 16 x 12
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
256 x 16 x 12
Conv + ReLU kernel [3x1] stride [2,1]
Conv + ReLU kernel [1x3] stride [1,2]
512 x 8 x 6
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
512 x 8 x 6
direct connections
first prediction
Conv + ReLU kernel [3x3] stride [1,1]
24 x 8 x 6
Conv kernel [3x3] stride [1,1]
2 x 8 x 6
2 x 8 x 6
optical flow
confidence
Upconv kernel [4x4] stride [2,2]
4 x 16 x 12
Upconv + ReLU kernel [4x4] stride [2,2]
516 x 16 x 12
Upconv + ReLU kernel [4x4] stride [2,2]
256 x 32 x 24
Upconv + ReLU kernel [4x4] stride [2,2]
128 x 64 x 48
Conv + ReLU kernel [3x3] stride [1,1]
24 x 64 x 48
Conv kernel [3x3] stride [1,1]
second prediction
2 x 64 x 48
2 x 64 x 48
optical flow
confidence
2x(Conv + ReLU)
Conv + ReLU
Conv
Fully Connected
Upconv + ReLU
Upconv

Figure 15. Encoder-decoder architecture for predicting optical flow. The depicted network is used in the bootstrap net and the iterative net part. Inputs with gray font (*depth and normals*, *flow from depth & motion*, *warped 2nd image*) are only available for the iterative net. The encoder-decoder predicts optical flow and its confidence at two different resolutions. The first prediction is directly appended to the end of the encoder and its output resolution is 8×6. We also apply our losses to this small prediction. We also feed this prediction back into the decoder part after upsampling it with an upconvolutional layer. The second prediction is part of the decoder and predicts all outputs with a resolution of 64×48, which is four times smaller than the original image dimensions (256×192). Due to this resolution difference we concatenate inputs from the previous network at the respective spatial resolution level within the encoder. We use direct connections from the encoder to the decoder, which allows the decoder to reuse features from the respective spatial resolution levels from the encoder.

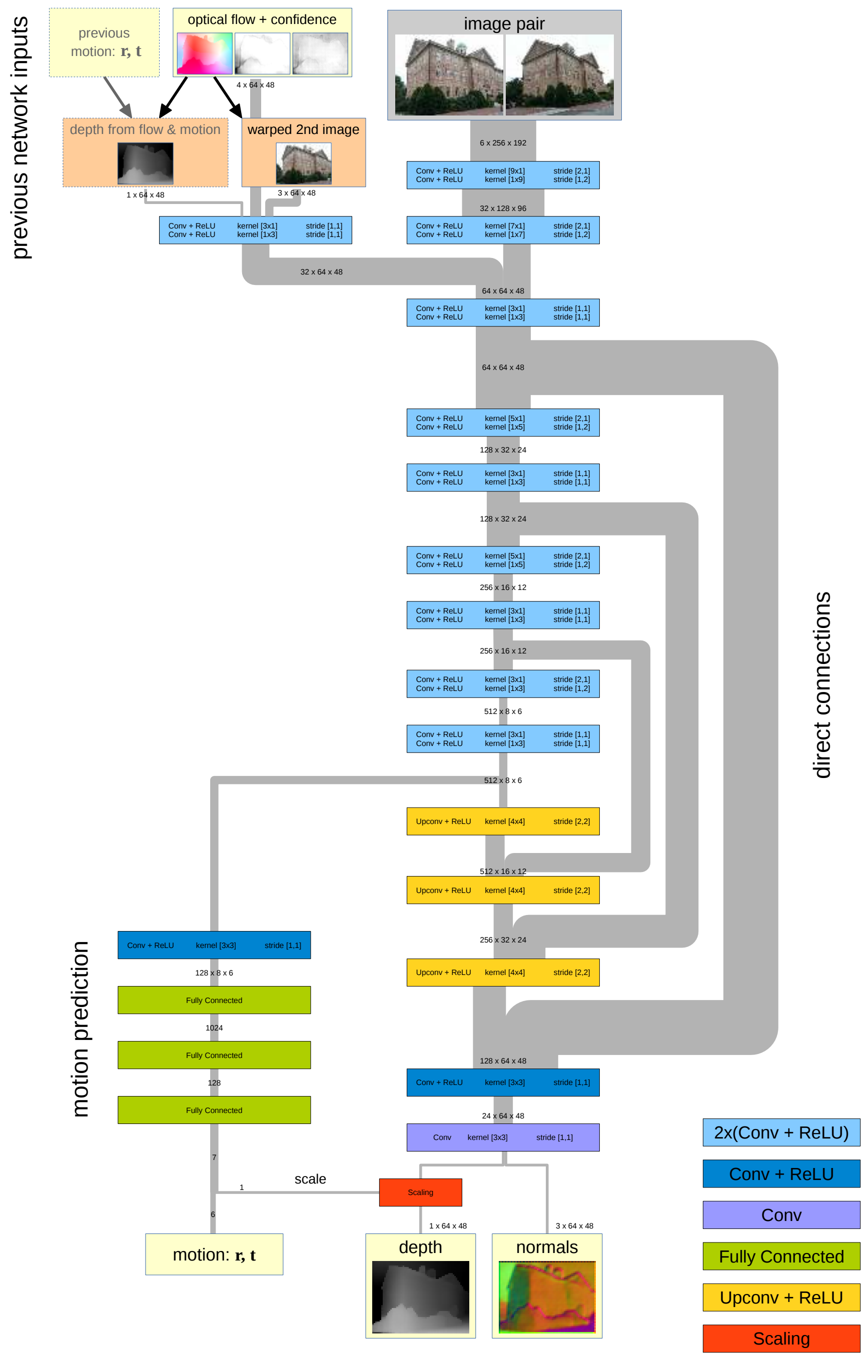
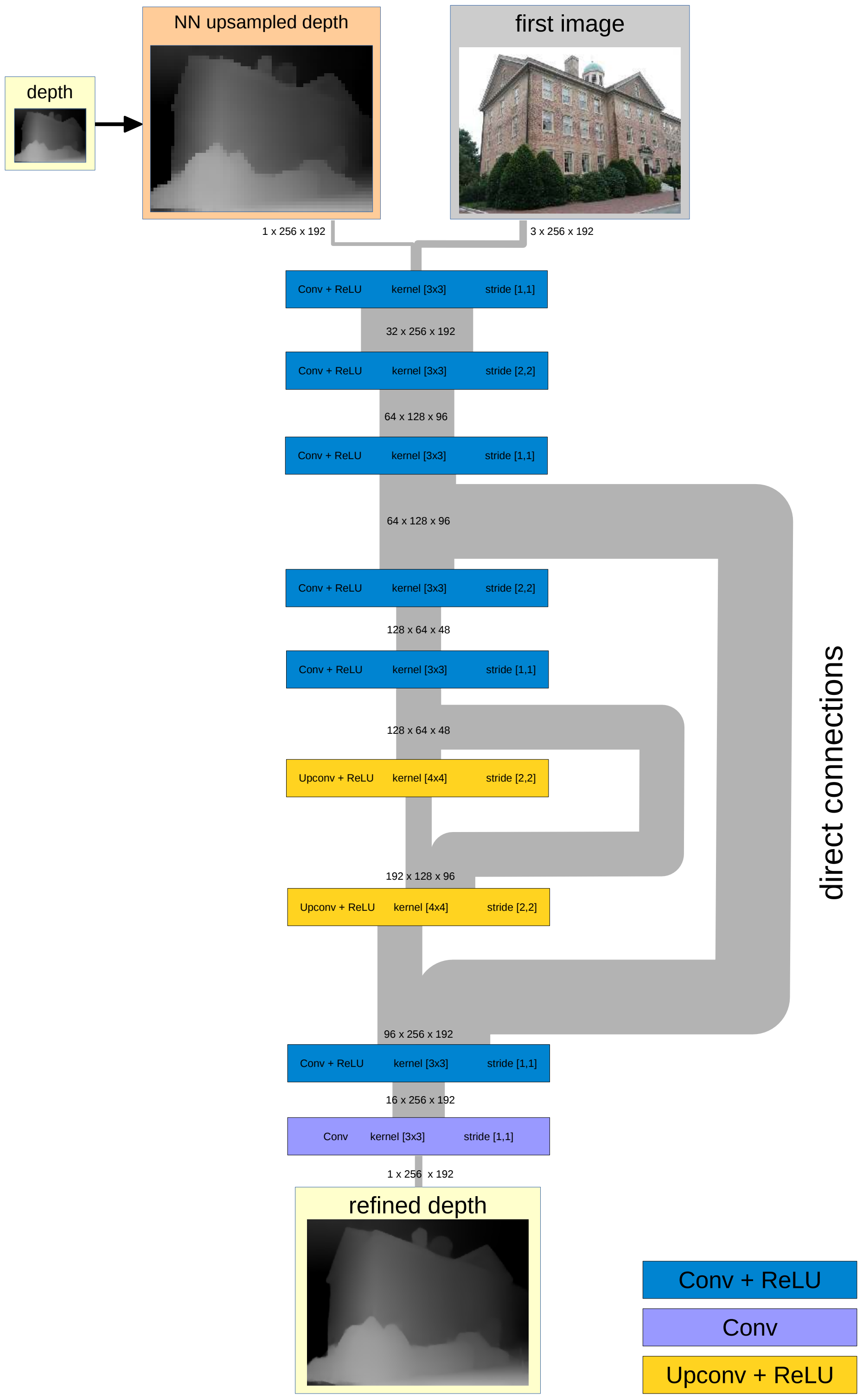

previous network inputs
previous motion: r, t
optical flow + confidence
image pair
4 x 64 x 48
depth from flow & motion
warped 2nd image
6 x 256 x 192
Conv + ReLU kernel [9x1] stride [2,1]
Conv + ReLU kernel [1x9] stride [1,2]
1 x 64 x 48
3 x 64 x 48
32 x 128 x 96
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
Conv + ReLU kernel [7x1] stride [2,1]
Conv + ReLU kernel [1x7] stride [1,2]
32 x 64 x 48
64 x 64 x 48
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
64 x 64 x 48
Conv + ReLU kernel [5x1] stride [2,1]
Conv + ReLU kernel [1x5] stride [1,2]
128 x 32 x 24
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
128 x 32 x 24
Conv + ReLU kernel [5x1] stride [2,1]
Conv + ReLU kernel [1x5] stride [1,2]
256 x 16 x 12
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
256 x 16 x 12
Conv + ReLU kernel [3x1] stride [2,1]
Conv + ReLU kernel [1x3] stride [1,2]
512 x 8 x 6
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
512 x 8 x 6
direct connections
Upconv + ReLU kernel [4x4] stride [2,2]
512 x 16 x 12
Upconv + ReLU kernel [4x4] stride [2,2]
256 x 32 x 24
Upconv + ReLU kernel [4x4] stride [2,2]
motion prediction
Conv + ReLU kernel [3x3] stride [1,1]
128 x 8 x 6
Fully Connected
1024
Fully Connected
128
Fully Connected
128 x 64 x 48
Conv + ReLU kernel [3x3] stride [1,1]
24 x 64 x 48
Conv kernel [3x3] stride [1,1]
7
scale
1
Scaling
6
1 x 64 x 48
3 x 64 x 48
motion: r, t
depth
normals
2x(Conv + ReLU)
Conv + ReLU
Conv
Fully Connected
Upconv + ReLU
Scaling

Figure 16. Encoder-decoder architecture for predicting depth and camera motion. The depicted network is used in the bootstrap net and the iterative net part. Inputs with gray font (*previous motion*, *depth from flow & motion*) are only available for the iterative net. The encoder-decoder predicts the depth map, the normals and the camera motion for an image pair. Similar to the encoder-decoder shown in Fig. 15, this encoder-decoder features direct connections and integrates previous network inputs at the corresponding resolution level into the encoder. This encoder-decoder predicts a camera motion vector and depth and normal maps. While all predictions share the encoder part, the camera motion prediction uses a separate fully connected network for its prediction. The depth and normal prediction is integrated in the decoder part. The scale of the depth values and the camera motion are highly related, therefore the motion prediction part also predicts a scale factor that we use to scale the final depth prediction.



depth
NN upsampled depth
first image
1 x 256 x 192
3 x 256 x 192
Conv + ReLU kernel [3x3] stride [1,1]
32 x 256 x 192
Conv + ReLU kernel [3x3] stride [2,2]
64 x 128 x 96
Conv + ReLU kernel [3x3] stride [1,1]
64 x 128 x 96
Conv + ReLU kernel [3x3] stride [2,2]
128 x 64 x 48
Conv + ReLU kernel [3x3] stride [1,1]
128 x 64 x 48
Upconv + ReLU kernel [4x4] stride [2,2]
192 x 128 x 96
Upconv + ReLU kernel [4x4] stride [2,2]
96 x 256 x 192
Conv + ReLU kernel [3x3] stride [1,1]
16 x 256 x 192
Conv kernel [3x3] stride [1,1]
1 x 256 x 192
refined depth
direct connections
Conv + ReLU
Conv
Upconv + ReLU

Figure 17. Encoder-decoder architecture for refining the depth prediction. The refinement network is a simple encoder-decoder network with direct connection. Input to this network is the first image and the upsampled depth map with nearest neighbor interpolation. Output is the depth map with the same resolution as the input image.

References

[1] C. Bailer, B. Taetz, and D. Stricker. Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation. In *IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015. 4

[2] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2016. 2

[3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 2

[4] S. Fuhrmann, F. Langguth, and M. Goesele. Mve-a multiview reconstruction environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, volume 6, page 8, 2014. 2

[5] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997. 4

[6] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004. 4

[7] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3

[8] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 3

[9] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 2

[10] B. Ummenhofer and T. Brox. Global, dense multiscale reconstruction for a billion points. In *IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. 2

[11] J. Xiao, A. Owens, and A. Torralba. SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1625–1632, Dec. 2013. 2