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.

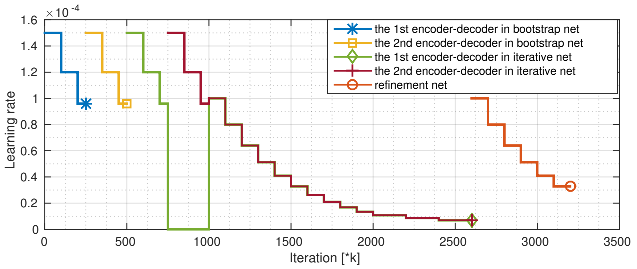


Figure 1. Learning rate schedule. During the training of each encoder-decoder a multistep learning rate is applied. At each step the learning rate is reduced by 20%.

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 photoconsistency 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.

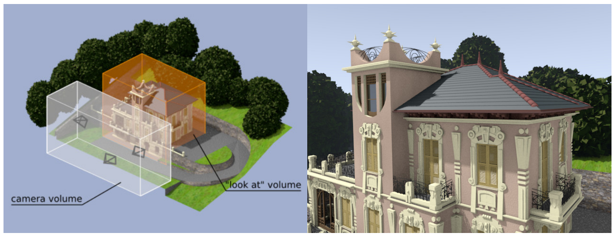


Figure 2. Annotation volumes for the Blendswap dataset. **Left:** We annotate each scene with volumes that describe valid camera positions (white box) and valid positions for the ”look at” target (orange box). **Right:** Image from a randomly sampled camera and target.

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

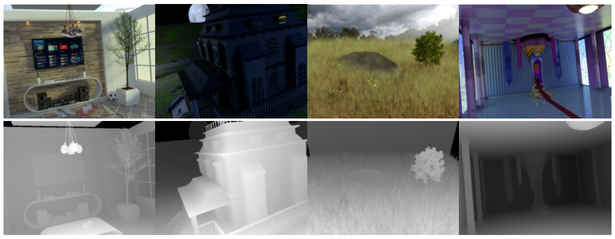


Figure 3. Images and the corresponding depth maps from the Blendswap dataset. Blendswap contains 150 distinct scenes, with a large variety of different styles and settings.

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?

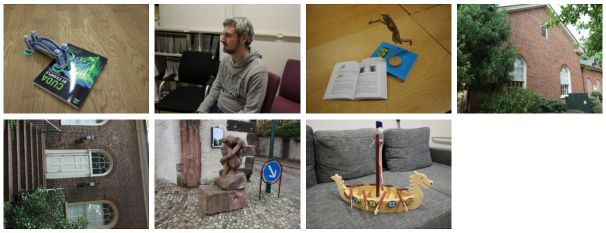


Figure 4. Samples of the seven scenes used for the generalization experiment.

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.

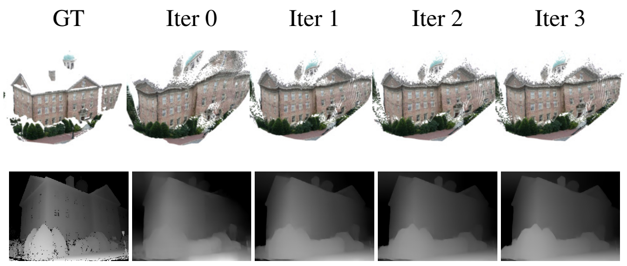


Figure 5. Effect of the iterative net on depth values.

		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.

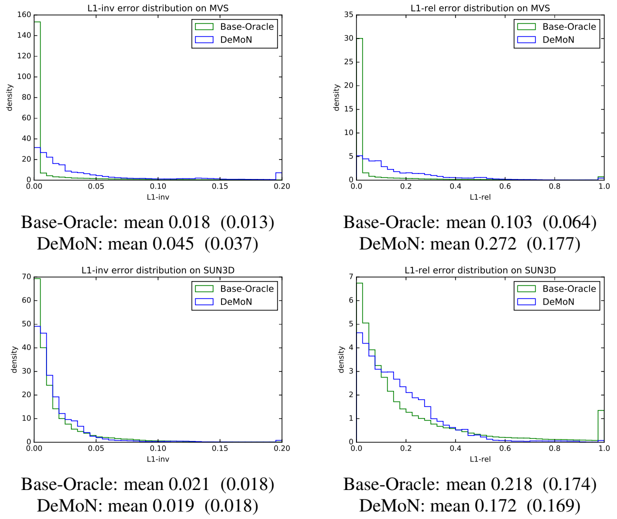
The distributions on the highly textured MVS show that Base-Oracle produces many very accurate depth estimates, while the distribution of errors is more spread for DeMoN. This is an effect of Base-Oracle using higher resolution images (640 × 480) than DeMoN (256 × 192). Base-Oracle also uses the motion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



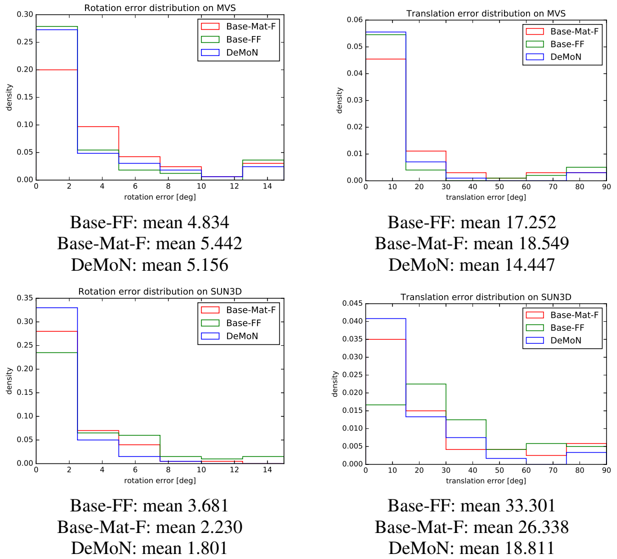
Base-Oracle: mean 0.018 (0.013)
DeMoN: mean 0.045 (0.037)

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

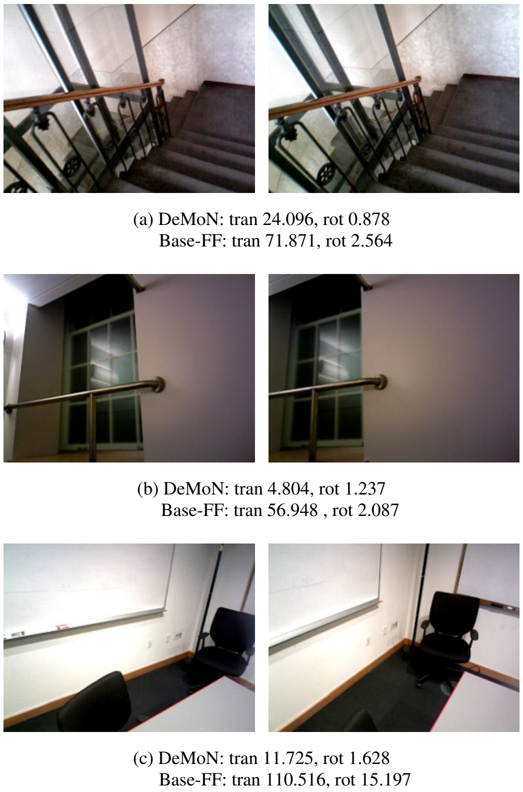
Base-FF: mean 3.681
Base-Mat-F: mean 2.230
DeMoN: mean 1.801

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).

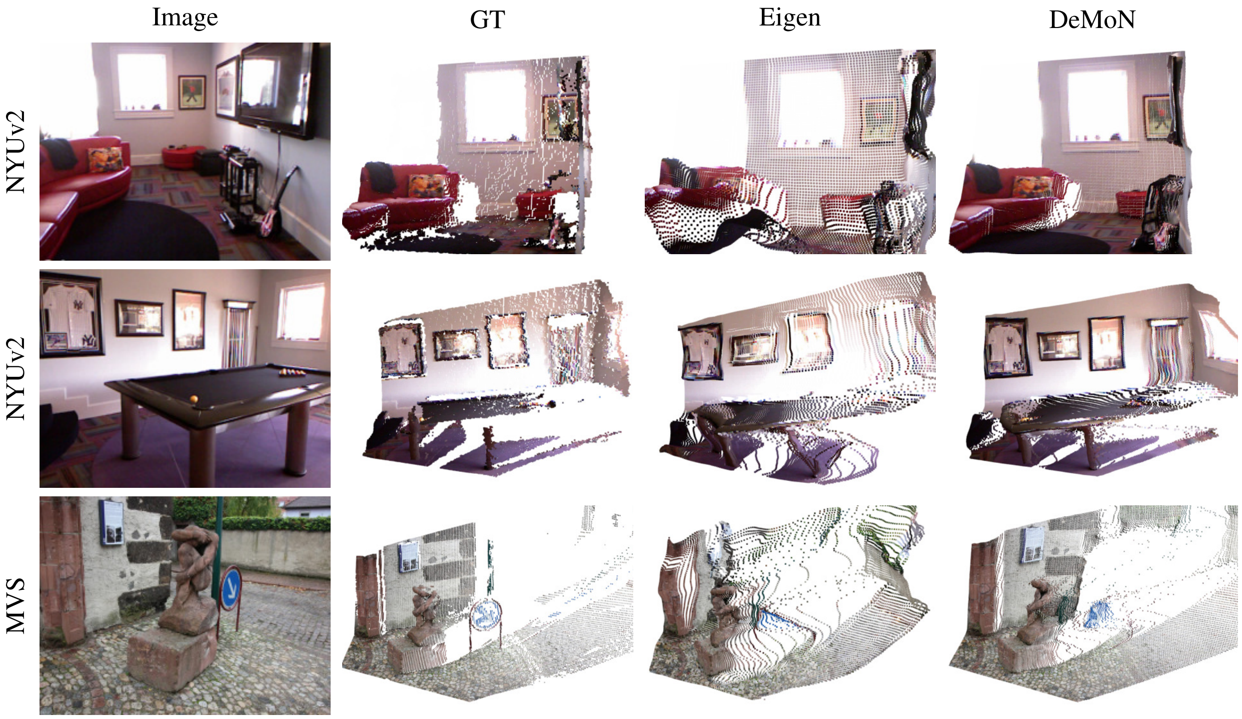


Figure 9. Qualitative point clouds comparison on NYUv2 and MVS.

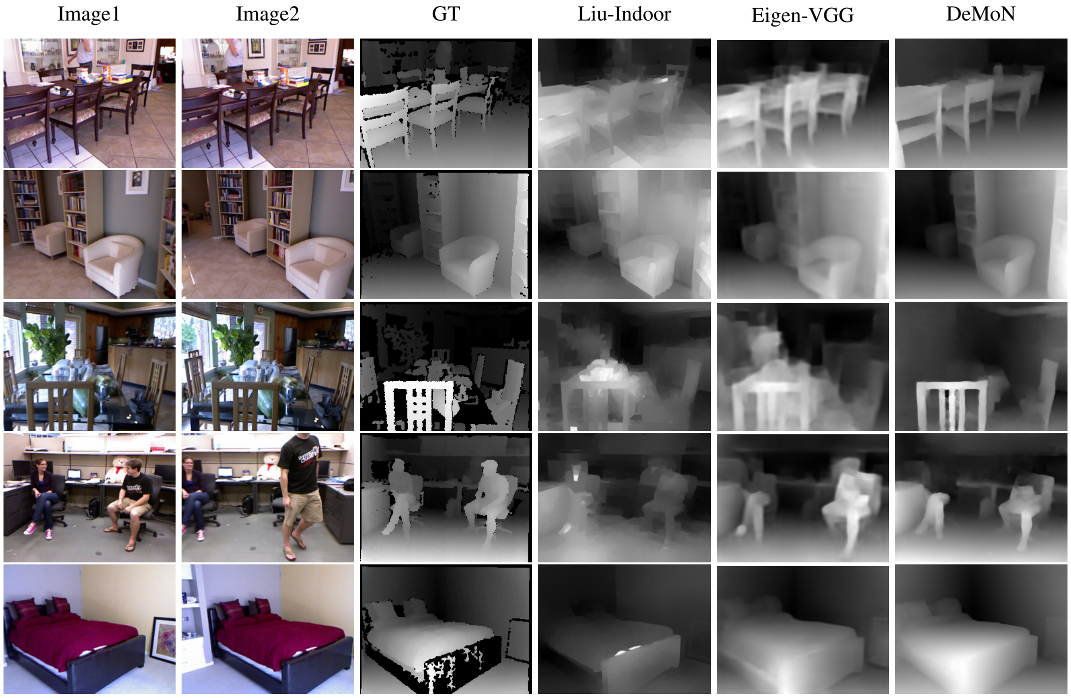


Figure 10. Qualitative depth prediction comparison on NYUv2. The Base-Oracle prediction is not available because there is no motion ground truth. Our method fails to predict the upper bodies of the persons in the fourth example because the persons move between the two frames.

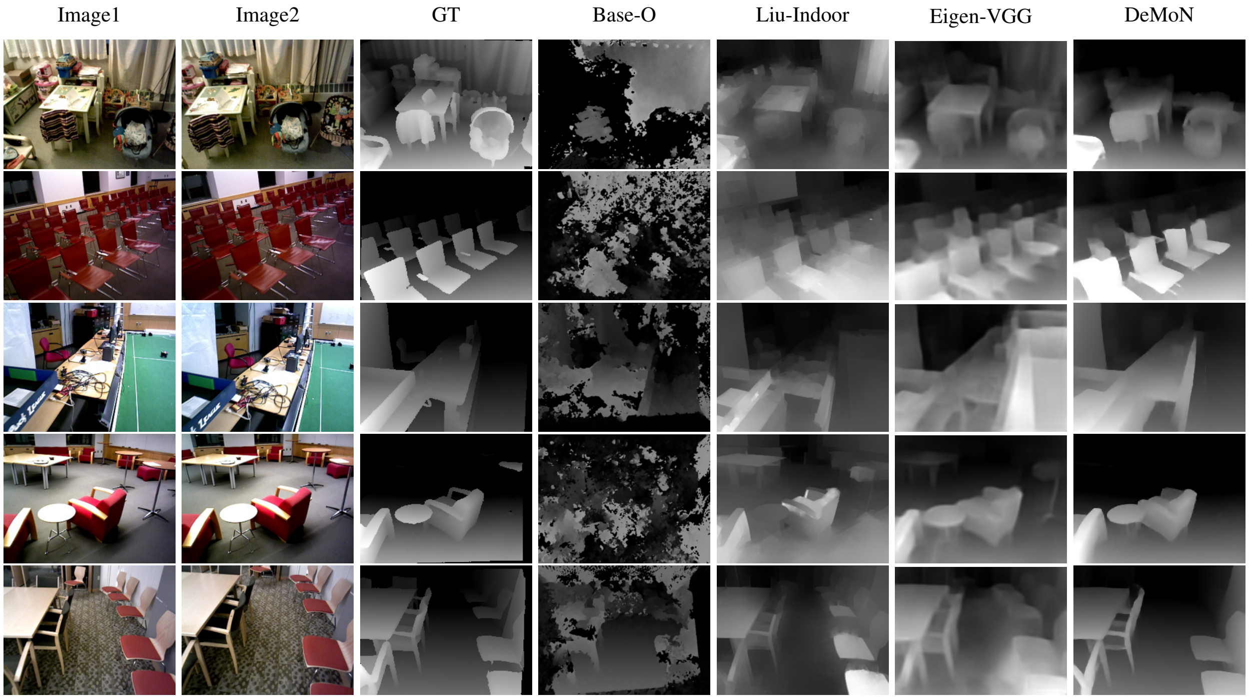


Figure 11. Qualitative depth prediction comparison on SUN3D. The Base-Oracle performs badly due to inaccurate motion ground truth. Eigen-VGG, which was trained on NYUv2, works well for many images. SUN3D is similar to the NYUv2 dataset shown in Fig. 10.

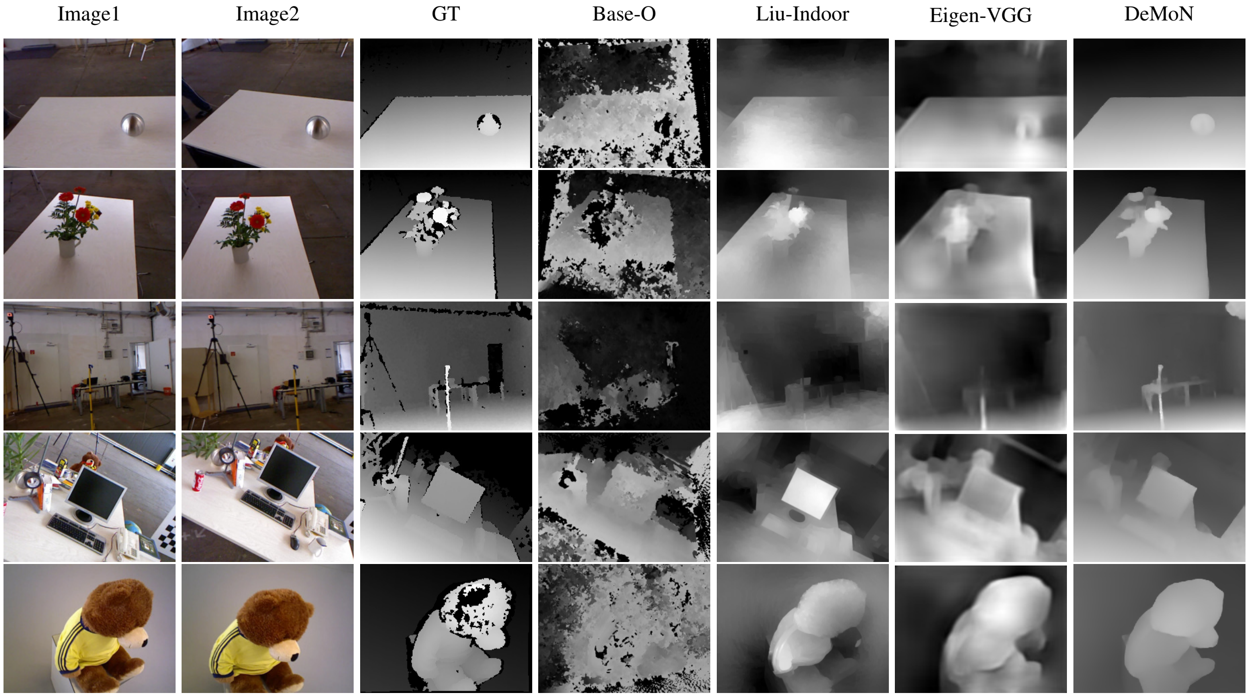


Figure 12. Qualitative depth prediction comparison on RGB-D SLAM. Our method can deal with very thin objects (third row).

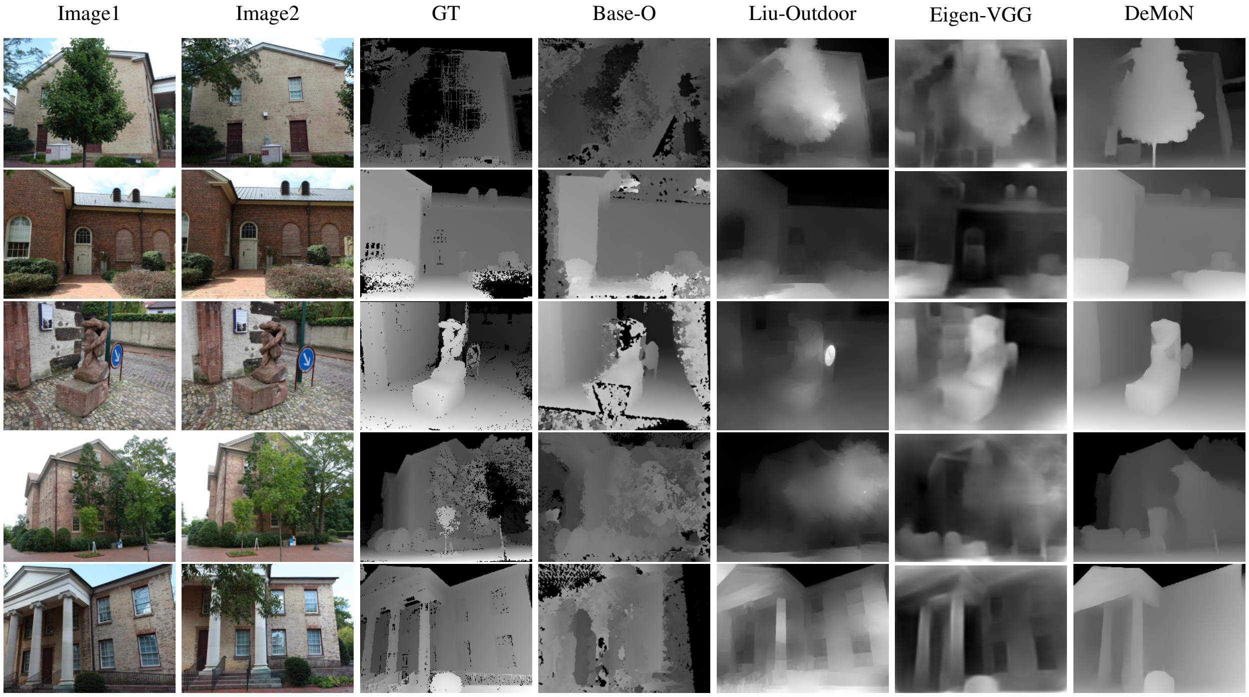


Figure 13. Qualitative depth prediction comparison on MVS. The single image methods Liu-Outdoor and Eigen-VGG do not generalize well to new datasets. The depth maps show coarse outliers caused by hallucinating wrong depth values at object contours like the street sign in the third row or the windows 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.

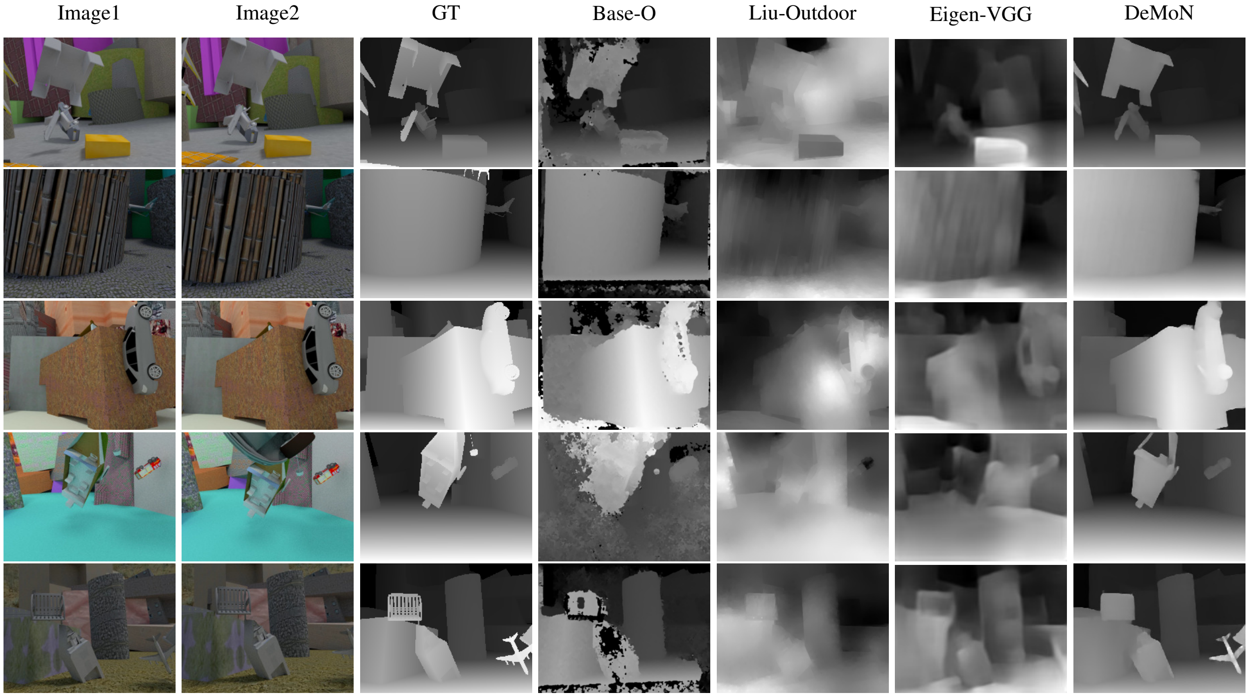


Figure 14. Qualitative depth prediction comparison on Scenes11. Base-Oracle and DeMoN give the best results on this dataset.

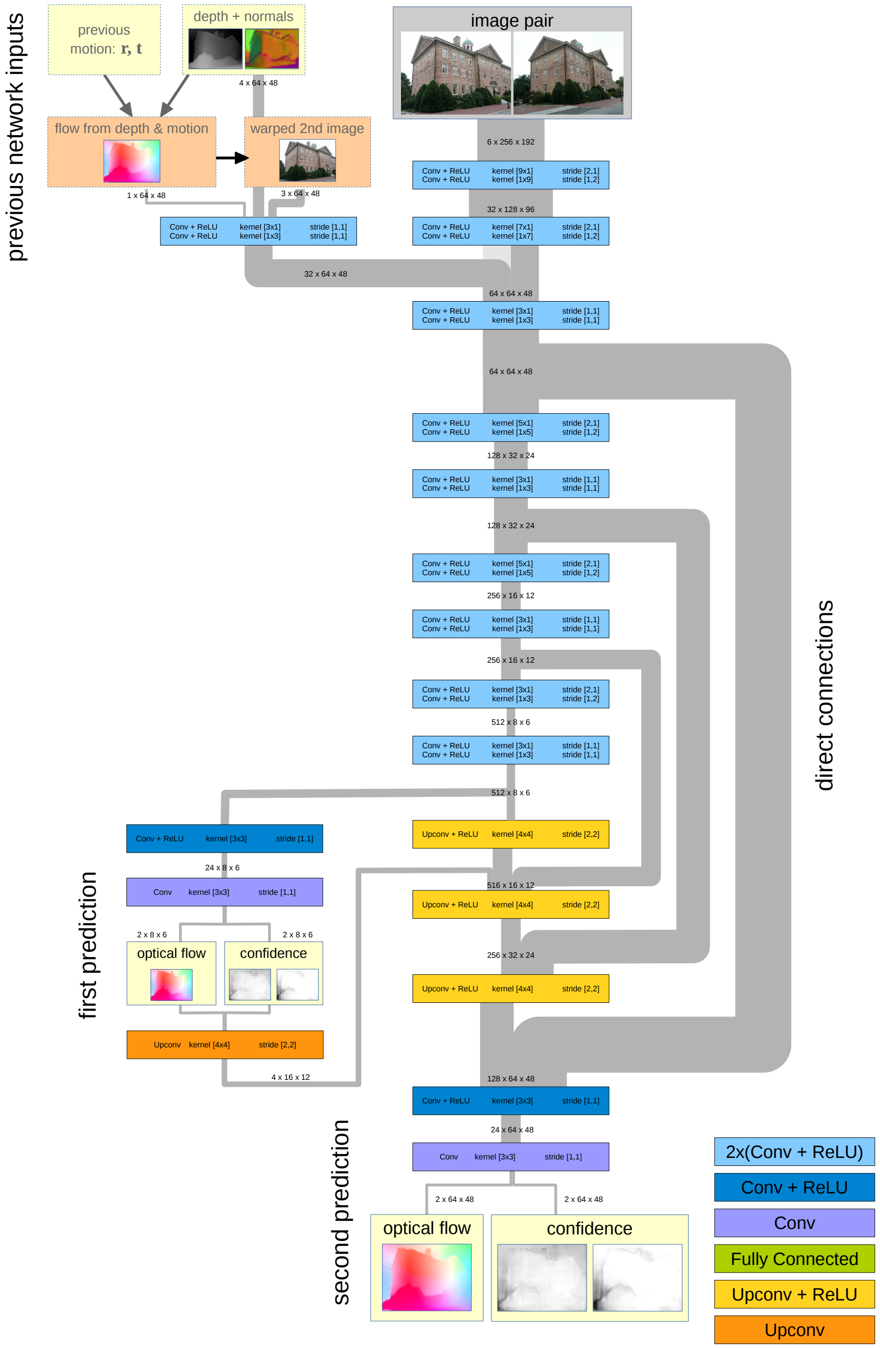
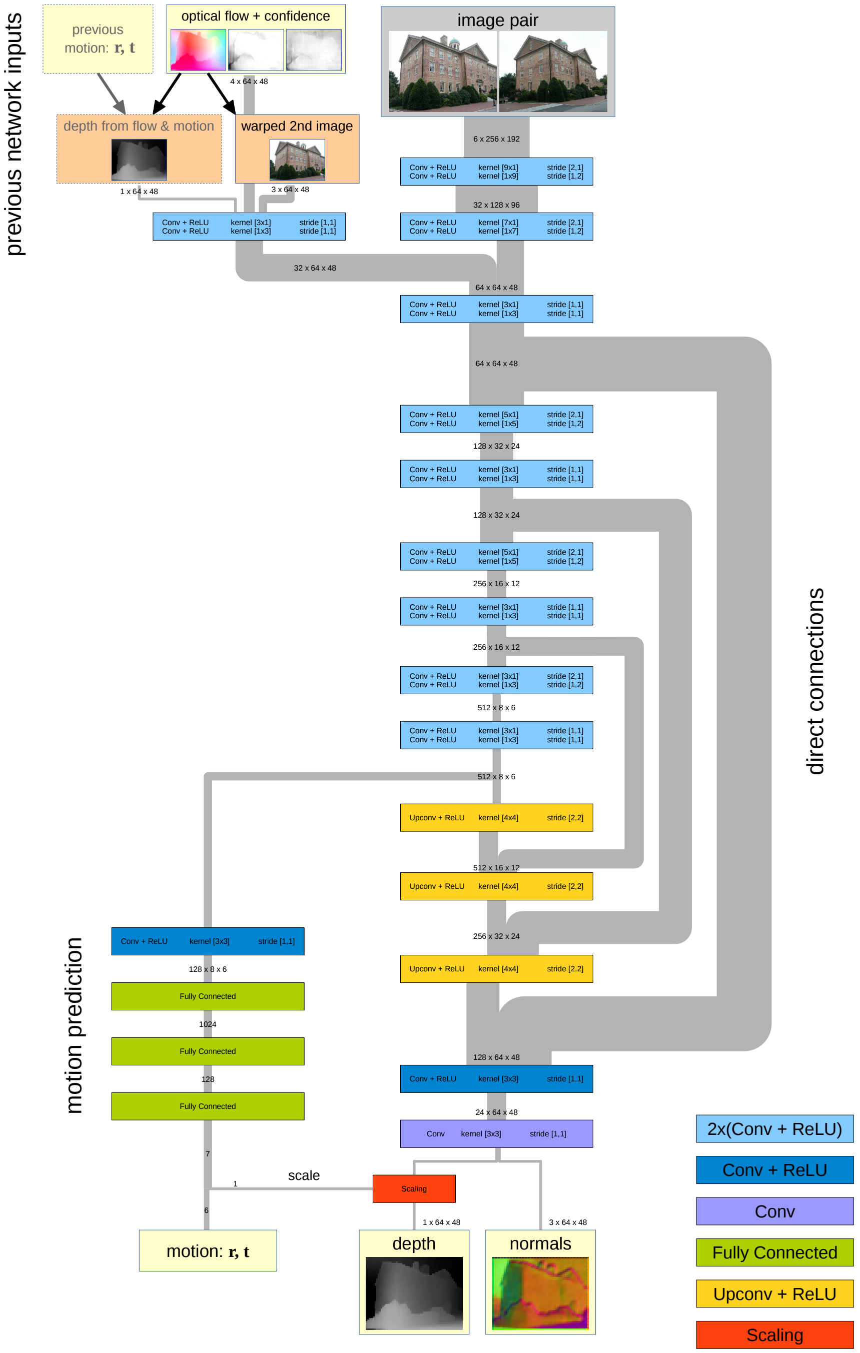

previous network inputs
previous motion: **r, t**
depth + normals
4 x 64 x 48
flow from depth & motion
warped 2nd image
1 x 64 x 48
3 x 64 x 48
image pair
6 x 256 x 192
Conv + ReLU kernel [9x1] stride [2,1]
Conv + ReLU kernel [1x9] stride [1,2]
32 x 128 x 96
Conv + ReLU kernel [7x1] stride [2,1]
Conv + ReLU kernel [1x7] stride [1,2]
Conv + ReLU kernel [3x1] stride [1,1]
Conv + ReLU kernel [1x3] stride [1,1]
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]
2 x 64 x 48
2 x 64 x 48
second prediction
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]
direct connections
512 x 8 x 6
Upconv + ReLU kernel [4x4] stride [2,2]
512 x 16 x 12
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
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]
7
1
scale
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.

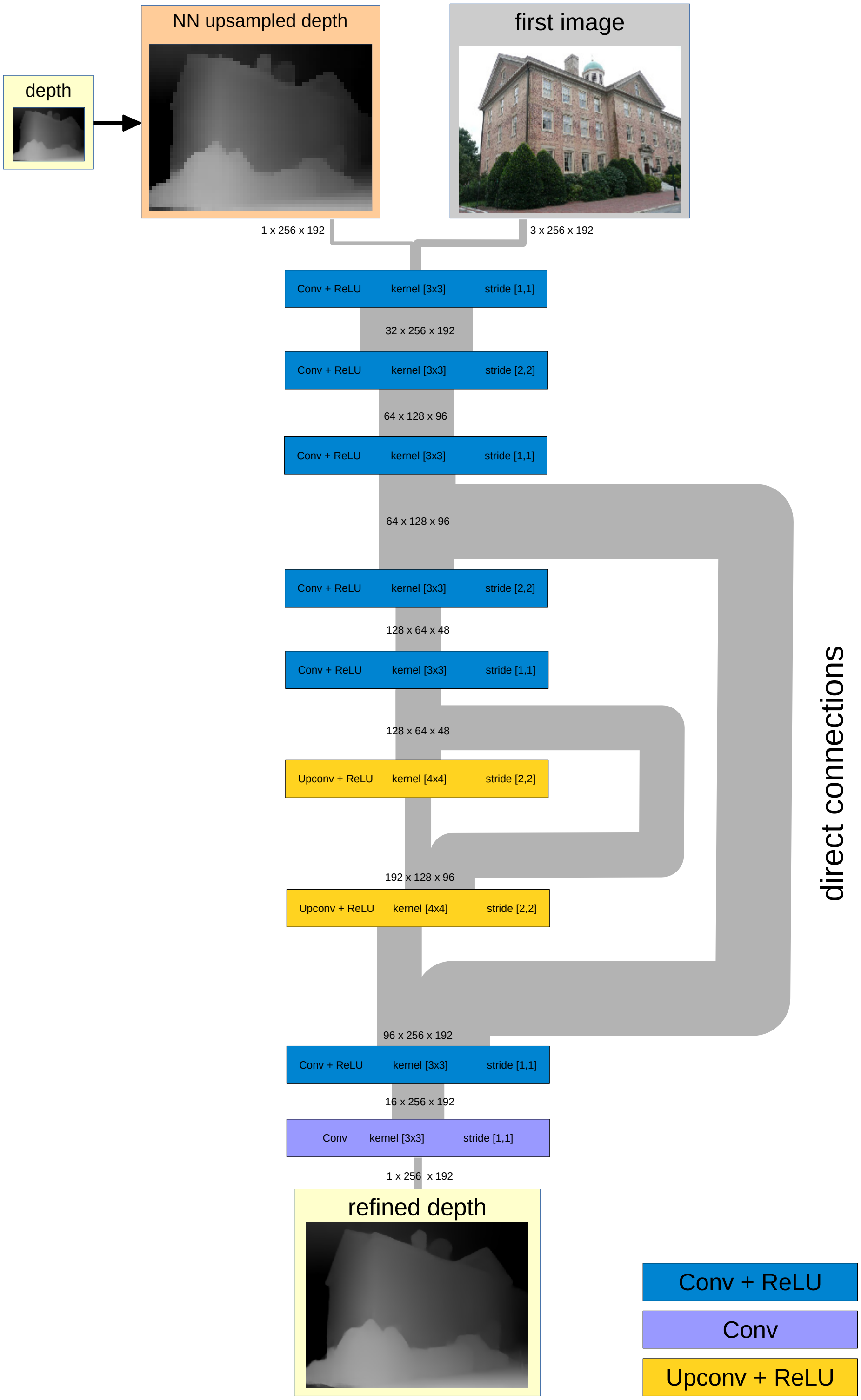

depth
NN upsampled depth
first image
1 x 256 x 192
3 x 256 x 192
Conv + ReLU kernel [3x3] stride [1,1]
32 x 256 x 192
Conv + ReLU kernel [3x3] stride [2,2]
64 x 128 x 96
Conv + ReLU kernel [3x3] stride [1,1]
64 x 128 x 96
Conv + ReLU kernel [3x3] stride [2,2]
128 x 64 x 48
Conv + ReLU kernel [3x3] stride [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