DeMoN: Depth and Motion Network for Learning Monocular Stereo

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

Jonas Uhrig1,2 Nikolaus Mayer1 Eddy Ilg1 Alexey Dosovitskiy1 Thomas Brox1
1University of Freiburg 2Daimler 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.

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

Iterative net. The iterative net is trained to improve estimates for each new scene. All these works do not provide depth estimates.

<table>
<thead>
<tr>
<th>Method</th>
<th>L1-inv</th>
<th>sc-inv</th>
<th>L1-rel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single image</td>
<td>0.080</td>
<td>0.159</td>
<td>0.696</td>
</tr>
<tr>
<td>Naïve image pair</td>
<td>0.079</td>
<td>0.165</td>
<td>0.722</td>
</tr>
<tr>
<td>DeMoN</td>
<td>0.012</td>
<td>0.131</td>
<td>0.097</td>
</tr>
</tbody>
</table>

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.
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 mini-batch. 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 $64 \times 48$ to save parameters and to reduce training and test time, the final refinement net scales the predictions to the full input image resolution ($256 \times 192$). 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 $r,t \in \mathbb{R}^3$. The rotation $r = \theta w$ is an angle axis representation with angle $\theta$ and axis $w$. The translation $t$ is given in Cartesian coordinates.

It is well-known that the reconstruction of a scene from 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 mini-batch. 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 $64 \times 48$ to save parameters and to reduce training and test time, the final refinement net scales the predictions to the full input image resolution ($256 \times 192$). 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 $r,t \in \mathbb{R}^3$. The rotation $r = \theta w$ is an angle axis representation with angle $\theta$ and axis $w$. The translation $t$ is given in Cartesian coordinates.

It is well-known that the reconstruction of a scene from 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 $||t|| = 1$. 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 $\xi$, surface normals $n$, optical flow $w$, and optical flow confidence $c$. For depth we use an L1 loss directly on the inverse depth values:

$$L_{\text{depth}} = \sum_{i,j} |s \xi(i,j) - \hat{\xi}(i,j)|,$$  

(1)
with ground truth \( \hat{\xi} \). Note that we apply the predicted scale \( s \) to the predicted values \( \xi \).

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 \( \hat{n} \) and \( \hat{w} \)

\[
L_{\text{normal}} = \sum_{i,j} \| \hat{n}(i,j) - \hat{\hat{n}}(i,j) \|_2^2
\]

\[
L_{\text{flow}} = \sum_{i,j} \| \hat{w}(i,j) - \hat{\hat{w}}(i,j) \|_2^2.
\]

(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

\[
\hat{c}_x(i,j) = e^{-|w_x(i,j) - \hat{w}_x(i,j)|},
\]

(3)

and the corresponding loss function reads as

\[
L_{\text{flow confidence}} = \sum_{i,j} |c_x(i,j) - \hat{c}_x(i,j)|.
\]

(4)

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

\[
L_{\text{rotation}} = \| r - \hat{r} \|_2^2
\]

\[
L_{\text{translation}} = \| t - \hat{t} \|_2^2.
\]

(5)

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

Scale invariant gradient loss. We define a discrete scale invariant gradient \( g \) as

\[
g_h[f](i,j) = \left( \frac{f(i+h,j) - f(i,j)}{f(i,j+h) + f(i,j) + f(i+h,j) + f(i,j)} \right) \top.
\]

(6)

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

\[
L_{\text{grad}} \xi = \sum_{h \in \{1,2,4,8,16\}} \sum_{i,j} \| g_h[\xi](i,j) - g_h[\hat{\xi}](i,j) \|_2^2.
\]

(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 \( g_h[\xi](i,j) = -g_h[z](i,j) \) 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.

Sceens11 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 Blendedswap 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

$$\text{sc-inv}(z, \hat{z}) = \sqrt{\frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} (\sum_i d_i)^2},$$

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

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

$$\text{L1-inv}(z, \hat{z}) = \frac{1}{n} \sum_i |z_i - \hat{z}_i| = \frac{1}{n} \sum_i \left| \frac{1}{z_i} - \frac{1}{\hat{z}_i} \right|$$

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. 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 Hirschmüller 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.

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
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 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.
show that DeMoN, as to be expected, shows the influence greatly improves the prediction qualitatively (see Fig. 9).

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.

### 6.5. Ablation studies

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.

### 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


