DRO: Deep Recurrent Optimizer for Structure-from-Motion

Xiaodong Gu\textsuperscript{1\*}  Weihao Yuan\textsuperscript{1\*}  Zuozhuo Dai\textsuperscript{1}  Siyu Zhu\textsuperscript{1}  Chengzhou Tang\textsuperscript{12}  Ping Tan\textsuperscript{12}

\textsuperscript{1}Alibaba A.I. Labs  \textsuperscript{2}Simon Fraser University

Abstract

There are increasing interests of studying the structure-from-motion (SfM) problem with machine learning techniques. While earlier methods directly learn a mapping from images to depth maps and camera poses, more recent works enforce multi-view geometry through optimization embed in the learning framework. This paper presents a novel optimization method based on recurrent neural networks to further exploit the potential of neural networks in SfM. Our neural optimizer alternatively updates the depth and camera poses through iterations to minimize a feature-metric cost. Two gated recurrent units are designed to trace the historical information during the iterations. Our network works as a zeroth-order optimizer, where the computation and memory expensive cost volume or gradients are avoided. Experiments demonstrate that our recurrent optimizer effectively reduces the feature-metric cost while refining the depth and poses. Our method outperforms previous methods and is more efficient in computation and memory consumption than cost-volume-based methods. The code of our method will be made public.

1. Introduction

Structure-from-motion (SfM) [30] is a fundamental task in computer vision and essential for numerous applications such as robotics, autonomous driving, augmented reality, and 3D reconstruction. Given a sequence of images, SfM methods optimize depth maps and camera poses to recover the 3D structure of a scene. Traditional methods solve the Bundle-Adjustment (BA) problem [35], where the reprojection error between reprojected 3D scene points and 2D image feature points are minimized iteratively.

Recently, deep-learning-based methods have dominated most benchmarks and demonstrated advantages over traditional methods [36, 17, 32, 21, 40, 33]. Earlier learning-based methods [36, 14, 23, 26] directly regress the depth maps and camera poses from the input images, but the domain knowledge such as multi-view geometry is ignored. To combine the strength of neural networks and traditional geometric methods, more recent works formulate the geometric-based optimization as differentiable layers and embed them in a learning framework [32, 33, 45].

We follow the approach of combining neural networks and optimization methods with some novel insights. Firstly, previous methods [32, 12, 33] adopt gradient-based optimization such as Levenberg-Marquardt or Gauss-Newton methods. However, the gradients could be noisy and misleading especially for the high-dimensional optimization problem in dense depth map computation. Careful regularization such as the depth bases [32] or manifold embedding [6, 7] is often required. Furthermore, a multi-resolution strategy is needed to gradually compute the solution from coarse to fine. In comparison, we employ a gated recurrent neural network for optimization as inspired by [34] as illustrated in Figure 1. Our method does not compute gradients and works on the high resolution image directly without regularization which might limit the algorithm generalization.

Secondly, some methods [40, 33, 48, 45] build cost volumes to solve the dense depth maps. Similar cost volume is also employed in [34] to compute optical flow. A cost vol-
volume encodes the errors of multiple different depth values at each pixel. It evaluates the result quality within a large spatial neighborhood in the solution space in a discrete fashion. While cost volumes have been demonstrated effective in computing depth maps [43, 20, 40], they are inefficient in time and space because they exhaustively evaluate results in a large spatial neighborhood. We argue that a gated recurrent network [11] can minimize the feature-metric error to compute dense depth without resorting to compute such a cost volume. In particular, the gated recurrent network only looks at the result quality at the current solution (i.e., a single point in the solution space) and those of the previous iterations to update the results. In spirit, our learned optimizer is zeroth-order and exploits temporal information during iterations, while gradient based methods or cost volume based methods rely only on the spatial information. In this way, our method has the potential of better running time and memory efficiency.

In experiments, our method demonstrates better accuracy than previous methods in both indoor and outdoor data. Our method is good at dealing with small-size, thin, and distinct objects. We also show that the recurrent optimizer reduces the feature-metric cost over iterations and produces gradually improved depth maps and camera poses.

Our contributions can be summarized as follows:

1) We propose a novel zeroth-order recurrent optimizer for joint depth and camera pose optimization where gradients or cost volumes are not involved for better memory and computation efficiency.

2) The depths and poses are alternatively updated to uncouple the mutual influence by the GRU module for effective optimization.

3) Our optimizer outputs better results than previous methods in both supervised and self-supervised settings.

2. Related work

**Supervised Deep SfM.** Deep neural networks can learn to solve the SfM problem directly from data [36, 48, 40]. With the ground-truth information, DeMoN [36] trains two network branches to regress structures and motions separately with an auxiliary flow prediction task to exploit feature correspondences. Some methods adopt a discrete sampling strategy to achieve high-quality depth maps [48, 33]. They generate depth hypotheses and utilize multiple images to construct a cost volume. Furthermore, the pose volume is also introduced in [40]. They take the feature maps to build two cost volumes and employ 3D convolutions to regularize.

There are also methods to directly regress scene depth from a single input image [14, 17, 26], which is an ill-posed problem. These methods rely heavily on the data fitting of the neural networks. Therefore, their network structure and feature volumes are usually bulky, and their performance are limited in unseen scenes.

**Self-supervised Deep SfM.** Supervised methods, nevertheless, require collecting a large number of training data with ground-truth depth and camera poses. Recently, many unsupervised works [49, 18, 28, 46, 27, 37, 42, 44, 29, 4, 21, 38] have been proposed to train a depth and pose estimation model from only monocular RGB images. They employ the predicted depths and poses to warp the neighbor frames to the reference frame, such that a photometric constraint is created to serve as a self-supervision signal. In this case, the dynamic objects is a problem and would generate errors in the photometric loss. To address this, semantic mask [24] and optical flow [50, 47, 5] are proposed to exclude the influence of moving objects. Another challenge is the visibility problem between different frames. To deal with this, a minimum re-projection loss are designed in [18, 21] to handle the occluded regions. Despite these efforts, there is still a gap between the self-supervised methods and the supervised methods.

**Learning to Optimize.** Traditional computer vision methods usually formulate the tasks as optimization problems according to the first principles such as photo-consistency, multi-view geometry, etc. Inspired by this, recently many works are seeking to combine the strength of neural network and traditional optimization-based methods. There are mainly two approaches in learning to optimize. One approach [3, 2, 32, 33] employs a network to predict the inputs or parameters of an optimizer, which is implemented as some layers in a large neural network for end-to-end training. On the contrary, the other approach directly learns to update optimization variables from the data [1, 9, 16, 12, 34].

However, the first approach needs to explicitly formulate the solver and is limited to problems where the objective function can be easily defined [3, 2, 32, 33]. Furthermore, the methods in [12, 32] need to explicitly evaluate gradients of the objective function, which is hard in many problems. Besides, the methods in [33, 34] adopt cost volumes, which make the model heavy to apply.

In comparison, our method does not require gradients computation or cost volume aggregation. It only evaluates the result quality at a single point in the solution space at each step. In this sense, it is a zeroth-order optimizer embedded in a network. By accumulating temporal evidence from previous iterations, our GRU module learns to minimize the objective function. Unlike the method in [34] which still relies on a cost volume, our method is more computation and memory efficient. Besides, two updaters in our framework, one for depth and the other one for pose, are alternatively updated, which is inspired by the traditional bundle adjustment algorithm.
### 3. Deep Recurrent Optimizer

#### 3.1. Overview

Given the reference image $I_0$ and $N$ neighboring images \{I_i\}_{i=1}^N, our method outputs the depth $D$ of the reference image and the relative camera poses \{T_i\}_{i=1}^N for images \{I_i\}_{i=1}^N as shown in Figure 2. Images are first fed into a shared feature extraction module to produce features $F_i$ for each image, then a depth head and a pose head take these features in and output the initial depth map and relative poses. Finally, the initial depth map and relative poses are refined by the depth and the pose GRU-optimizers alternatively, and converge to the final depth and poses.

#### 3.2. Feature Extraction and Cost Construction

Similar to BA-Net [32], we construct a photometric cost in feature space as the energy function to minimize. This cost measures the distance between aligned feature maps. Given the depth map $D$ for the reference image $I_0$ and the relative camera pose $T_i$ of $I_i$ respect to $I_0$, the cost is defined at each pixel $x$ in the reference image $I_0$:

$$C_i(x) = ||F_i(\pi(T_i, D(x) \cdot \pi^{-1}(x))) - F_0(x)||_2,$$  

(1)

where $||\cdot||_2$ is the L2 norm, and $\pi$ is the projection function. Thus, $D(x) \cdot \pi^{-1}(x)$ is the 3D point corresponding to the pixel location $x$, and $T_i$ transforms 3D points from the camera space of the image $I_0$ to that of $I_i$. Note that the feature-metric error in BA-Net [32] would further sum the cost over all pixels as $\sum_x C_i(x)$. However, in this work, we maintain a cost map $C(x)$ that has the same resolution with the feature map $F_i$. In the following of this paper, we refer $C(x)$ as cost map instead of feature-metric error.

**Depth and pose cost.** When there are multiple neighboring images, we average multiple cost values \{C_i(x)\}_{i=1}^N as $C_d(x)$ for the depth value at each pixel:

$$C_d(x) = \frac{1}{N} \sum_{i=1}^N C_i(x).$$  

(2)

For the pose cost, we directly use $C_i(x)$ on each image $I_i$ because the pose $T_i$ only associates with $I_i$ when the depth map $D$ is fixed in our alternative optimization.

**Feature extraction.** There are two feature extraction modules. One is denoted as base feature network for extracting the aforementioned feature maps \{F_i\}_{i=0}^N, while the other one is denoted as contextual feature network for providing the initial hidden state and the contextual feature for the GRU optimizer. We use ResNet18 [22] as our backbone to extract features. The resolution of the feature maps is $1/8$ of the original input images. The feature of the reference image is used for depth branch, while the feature of the concatenated image pair is used for pose branch.

#### 3.3. Iterative Optimization

We then minimize the cost map $C_d$ in an iterative manner. At each iteration, the optimizer outputs an update of the depth $\Delta D$ and that of the pose $\Delta T$. Inspired by [34], we utilize a gated recurrent unit to compute these updates, since a GRU can memorize the status at the previous results during the optimization and the gated activation makes the update easier to converge.

##### 3.3.1 Initialization

The initial depth and pose are from two simple initial networks, which are adding a depth head and a pose head upon the base feature network, respectively. The depth head is composed of two convolutional neural layers, and the pose head is plus another average pooling layer. The hidden state is initialized by the contextual feature network, with the tanh function as the activation.

##### 3.3.2 Recurrent Update

We design two GRU modules, one for updating the depth and the other one for updating the camera pose. Each GRU module receives the current cost map $C_i^{t-1}$ and the current
estimated variables $V^{t-1}$ (depth map $D^{t-1}$ or camera pose $T^{t-1}$) and outputs an incremental update $\Delta V^t$ to update the results as $V^t = V^{t-1} + \Delta V^t$.

Specifically, we first project the variable $V^{t-1}$ and the cost $C^{t-1}$ into the feature space with two convolutional layers $\mathcal{P}_v$ and $\mathcal{P}_c$, respectively, and then concatenate $\mathcal{P}_v(V^{t-1}), \mathcal{P}_c(C^{t-1})$, and the image contextual feature $\mathcal{F}_c$ into $M^{t-1}$. Therefore, the structure inside each GRU unit is as follows:

$$
\begin{align*}
    z^t &= \sigma(\text{Conv}_{5 \times 5}([h^{t-1}, M^{t-1}], W_z)) \\
    r^t &= \sigma(\text{Conv}_{5 \times 5}([h^{t-1}, M^{t-1}], W_r)) \\
    h^t &= \text{tanh}(\text{Conv}_{5 \times 5}(\{r^t \odot h^{t-1}, x^{t-1}\}, W_h)) \\
    \tilde{h}^t &= (1 - z^t) \odot h^{t-1} + z^t \odot \tilde{h}^t,
\end{align*}
$$

(3)

where $\text{Conv}_{5 \times 5}$ represents a separable $5 \times 5$ convolution, $\odot$ is the element-wise multiplication, $\sigma$ and $\text{tanh}$ are the sigmoid and the tanh activation functions. Finally, the depth maps or the camera poses are predicted from the hidden state $h^t$ by similar structures to the initial depth or the camera pose head in Sec. 3.3.1.

With this optimizer, from the initial point, the estimated depth and pose are iteratively refined as the optimization iteration proceeds. Finally they will both converge to fixed points $D^t \rightarrow D^*$ and $T^t \rightarrow T^*$.

### 3.3.3 Alternative Optimization

After defining the structure of the GRU unit, we update the depth map $D^t$ and the camera transformation $T^t$ alternatively in totally $m$ stages. As shown in Figure 3, at each stage, we first freeze the camera pose and update the depth map as $D^t = D^{t-1} + \Delta D^t$, which is repeated by $n$ times. Then we freeze the depth map $D$ and switch to the camera pose updating, where $T^t = T^{t-1} + \Delta T^t$ is also repeated by $n$ times. This alternative optimization leads to more stable optimization and easier training empirically. In our experiments, $m$ is set as 3 and $n$ is set as 4 if not particularly specified.

To gain more insights into the recurrent process and demonstrate the GRU unit behaves as a recurrent optimizer, we visualize how the feature-metric error decreases over the GRU iterations in Figure 4. This figure shows that both the depths and the poses are refined step-by-step to the optimum along with a decreasing cost. Eventually, the warped neighbor image is aligned seamlessly with the reference image, and the estimated depth is close to the ground truth. This indicates that our optimizer refines the outputs by learning to minimize the feature-metric error.

### 3.4. Training Loss

#### 3.4.1 Supervised Case

When ground truth is available, we supervise the training by evaluating the depth and pose errors.

**Depth supervision** $\mathcal{L}_{\text{depth}}$ computes the L1 distance between the predicted depth map $\hat{D}$ and the ground-truth depth map $D$ in each stage:

$$
\mathcal{L}_{\text{depth}} = \sum_{s=1}^{m} \gamma^{m-s} ||D^s - \hat{D}||_1,
$$

(4)

where $\gamma$ is a discounting factor.
4. Experiments

This section first presents the implementation details and then evaluates our supervised and self-supervised models on outdoor and indoor datasets, which are followed by the ablation studies.

4.1. Implementation Details

Our work is implemented in Pytorch and trained on Nvidia GTX 2080 Ti GPUs. The network is optimized end-to-end with Adam ($\beta_1 = 0.9, \beta_2 = 0.999$) and the learning rate decreases from $2 \times 10^{-4}$ to $5 \times 10^{-5}$ in 50 epochs.
For supervised training, we use the ground truth from the datasets to supervise the training with the losses described in section 3.4.1, where $\gamma$ is set as 0.85. For self-supervised training, without any ground-truth information, geometric constraints are leveraged to provide the supervision as depicted in section 3.4.2, where $\alpha$ is set as 0.85 and $\lambda$ is set as 0.01.

4.2. Datasets

**KITTI dataset.** The KITTI dataset is a widely used benchmark for depth evaluation, where outdoor scenes are captured from a moving vehicle. We adopt the training/test split proposed by Eigen et al. [14], resulting in 22,600 images for training and 697 images for testing. There are two types of ground-truth depth. One is the original Velodyne Lidar points which are quite sparse. The other one is the improved annotated depth map, which uses five successive images to accumulate the Lidar points and stereo images to handle moving objects. For the improved depth type there are 652 images for testing.

**ScanNet dataset.** ScanNet [13] is a large indoor dataset consisting of 1,513 RGB-D videos in 707 distinct environments. The raw data is captured from a depth camera. The
the results of our method and some previous methods on the KITTI dataset in Table 1. State-of-the-art single-frame depth estimation methods and deep SfM methods are listed. For a fair comparison, all SfM methods are evaluated under the two-view setting. From the results, our approach outperforms other methods by a large margin in both the supervised setting and the self-supervised setting. Also, the performance of our self-supervised model already surpasses most previous supervised methods. The qualitative results of these outdoor scenes are shown in Figure 5, from which we can see our approach estimates better depth for distant and small-size or thin objects, e.g., people, motorbike, and guidepost. Also, we predict sharper edges at object boundaries. Thin structures are usually recovered by fine updates in the last few iterations.

**Evaluation on ScanNet.** For indoor scenes, we evaluate our method on the ScanNet dataset in Table 2. For a fair comparison, all methods are evaluated under the two-view setting since there are only 2 images in the testing split. The results of Photometric BA and DeMoN are cited from [32]. The results show that our model outperforms previous methods on both depth accuracy and pose accuracy. Our self-supervised model is already able to predict the results that are comparable to supervised methods, especially on the pose accuracy. Among previous methods, DeepV2D performs best but it requires pre-training a complex pose solver first. Also, the inference time of their method is much longer than ours. Even using five views their performance is still not comparable to ours, of which their depth error is 0.057. From the qualitative results shown in Figure 6, our model predicts the finer depth of the indoor objects and is robust in clutter.

**Evaluation on SUN3D, RGB-D SLAM, and Scenes11.** We further evaluate our approach on the SUN3D, RGB-D SLAM, and Scenes11 datasets using the data pre-processed by [36] for a fair comparison. In the experiments, however, we find some of the picked image sequences are not suitable for training an SfM model because there is no enough overlap between neighbor images, which especially influences the prediction of the depth maps. Still, we achieve decent performance that is comparable to previous methods, especially in the pose accuracy, which is not seriously affected by the lack of overlap.

| Method                  | Supervised | Abs Rel | Sq Rel | RMSE | RMSE<sub>log</sub> | SI Inv | Rot (deg) | Tr (deg) | Tr (cm) | Time (s) |
|-------------------------|------------|---------|--------|------|-------------------|--------|-----------|----------|---------|---------|
| Photometric BA [15]     | ✓          | 0.268   | 0.427  | 0.788| 0.330             | 0.323  | 4.409     | 34.36    | 21.40   | –       |
| DeMoN [36]              | ✓          | 0.231   | 0.520  | 0.761| 0.289             | 0.284  | 3.791     | 31.626   | 15.50   | –       |
| BANet [32]              | ✓          | 0.161   | 0.092  | 0.346| 0.214             | 0.184  | 1.018     | 20.577   | 3.39    | –       |
| DeepV2D (2-view) [33]   | ✓          | 0.069   | 0.018  | 0.196| 0.099             | 0.097  | 0.092     | 11.731   | 1.902   | 0.95    |
| DRO (ours)              | ✗          | 0.140   | 0.127  | 0.496| 0.212             | 0.210  | 0.091     | 11.702   | 1.647   | 0.11    |
| DRO (ours)              | ✓          | 0.053   | 0.017  | 0.168| 0.081             | 0.079  | 0.047     | 9.219    | 1.160   | 0.11    |

Table 2: Quantitative results on the ScanNet dataset. Five metrics of the depth and two metrics of the pose are reported.

**Figure 6:** Qualitative results on the ScanNet dataset.
is that we do not need a heavy cost volume for optimization. Here, we also replace the feature-metric error with a $H \times W \times 64$ cost volume as the minimization object. The cost volume is in a cascaded structure, i.e., in each stage the depth range of the volume is dynamically adjusted according to the last estimated depth. From the results shown in Table 4, the performance of using this heavy cost volume is similar to using the cost map, which proves that employing information in temporal domain can make up the lack of neighborhood information in spatial domain. Also, we test the performance of a model without the cost input. As expected, the error of the depth estimation is large since the optimizer loses the objective to minimize.

### 5. Conclusion

In this work, we have proposed a zeroth-order deep recurrent optimizer for addressing the structure from motion problem. Two gated recurrent units have been introduced to serve as an optimizer considering both the spatial neighbor information and the temporal historical trajectories, such that the heavy cost volume could be replaced by a lightweight cost, and the gradient is not needed to be explicitly calculated in this end-to-end optimizer. The experiments have demonstrated our approach outperforms all previous methods on both the outdoor and indoor datasets, in both the supervised and self-supervised settings, which suggests that leveraging the information in the time domain can make up the lack of information in the spatial domain for an optimization problem.

---

### 4.4. Ablation Study

To inspect the effect of our framework, we evaluate each module of our model on the KITTI dataset and present the results in Table 4.

**GRU module.** The core module of our framework is the recurrent optimizer. To see how the recurrent module helps the optimization, we replace the GRU block with three convolutional layers. In the training, the depth error decreases to 0.058 in the first few epochs but then the network diverges. We think this is because the gate mechanism not only avoids the gradient explosion but also regularizes the optimization and makes the convergence stable. Furthermore, historical information is leveraged to guide the updating to avoid diverging directions.

**Alternant update.** It is important to alternatively update the depth and pose to decouple their influence in the feature-metric error. To see how they influence each other, we train a model where the optimizer predicts the depth and pose simultaneously. The depth accuracy in this setting is almost as poor as the one without GRU. This alternation is critical, for instance, when updating the depth, the training of the depth optimizer will be confused if the pose can be changed at the same time.

**Cost Volume.** One of the advantages of our method is that we do not need a heavy cost volume for optimiza-

| Method         | SUN3D       | RGB-D       | Scenes11     |
|----------------|-------------|-------------|--------------|
|                | L1-inv Sc-inv L1-rel Rot Tran | L1-inv Sc-inv L1-rel Rot Tran | L1-inv Sc-inv L1-rel Rot Tran |
| DeMoN [36]     | 0.019 0.114 0.172 1.801 18.811 | 0.028 0.130 0.212 2.641 20.585 | 0.019 0.315 0.248 0.809 8.918 |
| LS-Net [12]    | 0.015 0.189 0.650 1.521 14.347 | 0.019 0.090 0.301 1.010 22.100 | 0.010 0.410 0.210 0.910 8.210 |
| BANet [32]     | 0.015 0.110 **0.060** 1.729 13.260 | 0.008 0.087 0.050 2.459 14.90 | 0.080 0.210 0.130 1.298 10.370 |
| DeepSfM [40]   | 0.013 **0.093** 0.072 1.704 13.107 | 0.011 **0.071** 0.126 1.862 14.570 | **0.007 0.112 0.064** 0.403 5.828 |
| DRO (ours)     | **0.011** 0.108 0.076 **1.334 10.988** | **0.004** 0.087 **0.046** 2.839 **11.390** | 0.010 0.167 0.096 **0.366 3.705** |

Table 3: Quantitative results on the SUN3D, RGB-D SLAM, and Scenes11 datasets.

| Setting     | Abs Rel | Sq Rel | RMSE  | $R_{log}$   | $1.25$ | $1.25^2$ |
|-------------|---------|--------|-------|-------------|--------|----------|
| w/o GRU     | 0.058   | 0.258  | 2.953 | 0.097       | 0.955  | 0.992    |
| w/o Alter   | 0.055   | 0.247  | 2.952 | 0.094       | 0.959  | 0.992    |
| w/o Cost    | 0.065   | 0.324  | 3.270 | 0.112       | 0.940  | 0.988    |
| Cost volume | 0.049   | 0.214  | 2.804 | 0.086       | 0.966  | 0.994    |
| Full-setting| 0.047   | 0.199  | 2.629 | 0.082       | 0.970  | 0.994    |
| 0           | 0.094   | 0.529  | 4.014 | 0.150       | 0.891  | 0.974    |
| 4           | 0.059   | 0.266  | 2.992 | 0.099       | 0.951  | 0.992    |
| 8           | 0.049   | 0.208  | 2.687 | 0.084       | 0.968  | 0.994    |
| 16          | 0.046   | 0.198  | 2.623 | 0.081       | 0.970  | 0.994    |
| 24          | 0.046   | 0.199  | 2.626 | 0.082       | 0.970  | 0.994    |

Table 4: Ablation study on the KITTI dataset. The first six metrics of those used in Table 1 are reported here.
References

[1] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017. 2

[2] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, pages 9558–9570, 2019. 2

[3] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017. 2

[4] Jia-Wang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 2

[5] Jia-Wang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 2

[6] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2560–2568, 2018. 1

[7] Michael Bloesch, Tristan Laidlow, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Learning meshes for dense visual slam. In *Proceedings of the International Conference on Computer Vision*, pages 5855–5864, 2019. 1

[8] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 7

[9] Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando Freitas. Learning to learn without gradient descent. In *Proceedings of the International Conference on Machine Learning*, pages 748–756. PMLR, 2017. 2

[10] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the International Conference on Computer Vision*, pages 7063–7072, 2019. 6

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 2

[12] Ronald Clark, Michael Bloesch, Jan Czarnowski, Stefan Leutenegger, and Andrew J Davison. Ls-net: Learning to solve nonlinear least squares for monocular stereo. *arXiv preprint arXiv:1809.02966*, 2018. 1, 2, 8

[13] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6

[14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pages 2366–2374, 2014. 1, 2, 6

[15] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd: Large-scale direct monocular slam. In *Proceedings of the European Conference on Computer Vision*, pages 834–849. Springer, 2014. 7

[16] Lijie Fan, Wenbing Huang, Chuang Gan, Stefano Ermon, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6016–6025, 2018. 2

[17] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018. 1, 2, 6

[18] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the International Conference on Computer Vision*, pages 3828–3838, 2019. 2, 6

[19] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the International Conference on Computer Vision*, pages 3828–3838, 2019. 5

[20] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2495–2504, 2020. 2

[21] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2485–2494, 2020. 1, 2, 6

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 3

[23] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421, 2015. 1

[24] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *Proceedings of the European Conference on Computer Vision*, pages 582–600. Springer, 2020. 2

[25] Yevhen Kuznietsov, Jörg Stücker, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2215–2223, 2017. 6

[26] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar...
null