R-MSFM: Recurrent Multi-Scale Feature Modulation for Monocular Depth Estimating

Zhongkai Zhou, Xinnan Fan, Pengfei Shi*, Yuanxue Xin
College of Internet of Things Engineering;
Hohai University, Changzhou, China
200220030004@hhu.edu.cn, fanxn@hhuc.edu.cn, {shipf, xinyx}@hhu.edu.cn

Abstract

In this paper, we propose Recurrent Multi-Scale Feature Modulation (R-MSFM), a new deep network architecture for self-supervised monocular depth estimation. R-MSFM extracts per-pixel features, builds a multi-scale feature modulation module, and iteratively updates an inverse depth through a parameter-shared decoder at the fixed resolution. This architecture enables our R-MSFM to maintain semantically richer while spatially more precise representations and avoid the error propagation caused by the traditional U-Net-like coarse-to-fine architecture widely used in this domain, resulting in strong generalization and efficient parameter count. Experimental results demonstrate the superiority of our proposed R-MSFM both at model size and inference speed, and show the state-of-the-art results on the KITTI benchmark. Code is available at https://github.com/jsczzzk/R-MSFM

1. Introduction

The objective of depth estimation is to determine the depth of each pixel in an image. From the early stages of computer vision, depth estimation from images has always been one of the major challenges for researchers. Depth estimation as a low-level task is crucial to complete higher-level tasks, including 3-D reconstruction[23], autonomous driving[6], 3-D target detection[36], underwater image restoration[43], and many more.

Depth estimation is traditionally regarded as a stereo matching problem between the left and right images, which mainly approaches as a hand-crafted optimization[18], supervised[4] or self-supervised manner[24]. Although decades of developments have significantly improved its accuracy, the time-consuming matching process inevitably limits the scope of the deployment. Inspired by the domain of traditional structure from motion (SFM)[38], recent studies[13, 2, 31, 14] have demonstrated the feasibility of estimating depth from a single image as a view-synthesis problem using a photometric reconstruction loss in a self-supervised manner. Following this successful, new design paradigm, recent works mostly focus on the designing of the specific loss function[28, 14] and the improvements to the currently widely used U-Net like coarse-to-fine architecture[19, 15].

In this paper, inspired by the domain of optical flow [37], we introduce Recurrent Multi-Scale Feature Modulation(R-MSFM) , a new and effective lightweight deep learning architecture, to extend the architecture choice for monocular depth estimation. The three most significant strengths of R-MSFM are as follows:

- **Lightweight architecture**: R-MSFM reduces the parameters of Monodepth2 by 73 percent, from 14.3M to 3.8M, which is suitable for memory-limited scenarios.
- **State-of-the-art accuracy**: R-MSFM achieves state-of-the-art performance, which gets a 4.470 $RMSE$ lower than Monodepth2 (4.701) on the KITTI Eigen split test set.
- **Reasonable inference speed.**: R-MSFM processes 640×192 videos at 44 frames per second on a RTX2060 GPU. It can flexibly choose the update number and get a trade-off between speed and accuracy with half of the overall updates, which runs on 72 frames per second while still outperforms Monodepth2.

R-MSFM consists of four main components: i) a depth encoder that extracts the per-pixel representations from ResNet18 except for the last two blocks, producing multi-scale features; ii) a parameter-shared depth decoder that iteratively updates an inverse depth initialized at zero, avoiding the spatial imprecision at coarse level propagating to fine part; iii) a parameter-learned upsampling module that adaptively upsamples the estimated inverse depth, preserv-

*Corresponding author.

ing its motion boundaries; iv) a multi-scale feature modulation module that modulates content across the multi-scale feature maps, maintaining semantically richer while spatially more precise representations for each iterative update.

At each iterative update, R-MSFM maintains and refines a single inverse depth at the fixed 1/8 input resolution and then directly upsamples it to the full resolution with the learned mask. This is different from traditional U-Net like coarse-to-fine architecture in previous works[14, 15, 45, 19], where depth is first estimated at coarse resolution(1/32 input resolution) and then gradually upsampled and refined until the full resolution. By the progressive refinement at the fixed fine resolution, R-MSFM overcomes several limitations of the coarse-to-fine architecture: the error propagation from coarse to fine resolution, the difficulty of delineating small objects, the independence of the multi-scale decoders. Experimental results show R-MSFM achieves state-of-the-art performance both at accuracy and model size with reasonable inference speed.

2. Related Work

Depth estimation is an essential part of understanding the 3D world, which has a major impact on robotic systems and many vision tasks[29, 16, 7, 1]. For traditional computer vision methods, depth estimation from a single image is an ill-posed problem without relying on the second input frame. However, humans can learn a lot of prior knowledge about 3D scene comprehension through interacting with the real world. Therefore, even with a single eye, they can still get the absolute depth of the scene. With the development of deep learning that mimics the mechanism of the human brain, many works are dedicated to extracting scene depth from monocular images. We will review these related methods in the next section.

2.1. Fully-Supervised monocular depth estimation

The fully-supervised monocular depth estimation network adopts LiDAR's ground truth as the supervised signal to regress depths. During the training process, the network can learn depth information guided by ground truth. Eigen *et al*. [9] first constructed the monocular depth estimator using deep learning technique, which infers the corresponding depth from a single input image. The estimator consists of a global estimation layer followed by a local refinement layer. Therefore, this estimator preserves depth values at the edge of the image and achieves state-of-the-art results on both NYU Depth[35] and KITTI[11] benchmarks in that year. However, the global estimation layer and the local refinement layer need to be trained separately, which increases the difficulty of the training process. In order to solve the problem above, Evan Shelhamer *et al*. [33] extended the fully convolutional network[25] specifically designed for semantic segmentation to the task of monocular depth estimation, enabling the training process to be carried out in an end-to-end manner while improving accuracy. Due to the success of deep residual learning in image recognition[17], Laina *et al*. [21] introduced it into the domain of monocular depth estimation and replaced $L2$ loss with reverse Huber loss[47], thereby further stabilizing the training process and improving the accuracy of the network. Although the monocular depth estimation network trained with ground truth achieves high accuracy, obtaining ground truth from different scenes still limits the applications of these methods in the real world.

2.2. Self-Supervised monocular depth estimation

Regrading the restriction of ground truth used in fully-supervised methods, many works focused on the geometric constraints between frames as the supervised signal due to the ubiquitous cameras in the real world. R. Garg *et al*. [10] first inferred the corresponding depth from a single image using stereo training pairs in a self-supervised manner. They synthesized a novel view to obtain a supervised signal, which comprises a photometric loss between the left input image and the warped right image. Clément Godard *et al*. [13] further improved the accuracy of monocular depth estimation by introducing a new network architecture with a novel training loss, which includes a left-right disparity consistency loss and a single scale SSIM term[41]. Since consecutive images are easier to obtain than stereo pairs in the real world, it is suitable to use them directly as the training set for monocular depth estimation network. Zhou *et al*. [46] first jointly trained a separate pose network and a depth estimation network using consecutive images self-supervised by a photometric loss along with an additional motion explanation mask. Although this work proves the feasibility of estimating depth from a single image, its robustness still suffers from occlusions and moving targets. After careful analysis, Clément Godard *et al*. [14] showed that a well-designed loss function is more effective than a complex architecture in dealing with the above problems. They proposed i) a strategy that takes the minimum of the photometric loss instead of averaging for each pixel to address occlusions in consecutive images during monocular training. ii) an approach that automatically marks pixels as static or relatively static between consecutive frames. iii) a multi-scale photometric loss that samples all the depths at intermediate layers to full resolution for better supervision. Inspired by [40, 20], Adrian Johnston *et al*. [19] introduced the self-attention mechanism and discrete disparity prediction into the field of monocular depth estimation, enabling the network to be more robust at non-contiguous regions and motion boundaries. Although the accuracy of self-supervised monocular depth estimation has been dramatically improved, it is still far from fully-supervised methods. In this work, we demonstrate the superiority of iterative

depth refinement at the fixed resolution relying on the multi-scale feature modulation module and the parameter-shared decoder.

3. Method

In the section, we describe the details of our proposed R-MSFM that takes a single RGB image to produce the corresponding depth, and the self-supervised strategy that let our network learn from unlabeled monocular videos. The overview of our model is depicted in Figure 1.

3.1. Depth Encoder

Different from the current state-of-the-art method[14], which adopts the whole ResNet18[17] as its depth encoder. Our encoder contains only a part of ResNet18, which removes the last two blocks whose feature maps are semantically strong but spatially imprecise. Specifically, our depth encoder takes a single input image I_1 and outputs the multi-scale feature maps X_1, X_2, X_3 at 1/2, 1/4, and 1/8 of the input resolution: $I_1 : R^{H\times W\times 3} \rightarrow X_1 : R^{\frac{H}{2}\times\frac{W}{2}\times C_1}, X_2 : R^{\frac{H}{4}\times\frac{W}{4}\times C_2}, X_3 : R^{\frac{H}{8}\times\frac{W}{8}\times C_3}$ where C_1, C_2, C_3 in ResNet18 are 64, 64, 128 respectively. Since our depth decoder works at the fixed 1/8 input resolution, we should ensure that the multi-scale feature maps are of uniform size. On the other hand, the multi-scale feature maps from ResNet18 are under $ReLU$ nonlinearities, which conflicts with $Tanh$ nonlinearities in the multi-scale feature modulation module. Therefore, we transform them through one or two stride-2 3×3 convolutional layers, followed by $Tanh$ nonlinearities. For instance, two consecutive stride-2 3×3 convolutional layers are applied to X_1 for $4\times$ downsampling, one stride-2 3×3 convolutional layer is applied to X_2 for $2\times$ downsampling, and an additional stride-1 3×3 convolutional layer is applied to X_3 for nonlinearity transforming. Consequently, we obtain the multi-scale feature maps X_1, X_2, X_3 with uniform size $R^{\frac{H}{8}\times\frac{W}{8}\times C_3}$.

3.2. Depth Decoder

We employ a parameter-shared architecture for our depth decoder, which works at the fixed 1/8 input resolution to avoid the error propagation caused by the traditional coarse-to-fine architecture. The depth decoder outputs the inverse depth through five consecutive convolutional layers with $Sigmoid$ at the output and $LeakyReLU$ nonlinearities elsewhere. In particular, we apply two convolutional layers to the estimated inverse depth map itself to generate depth feature maps. Therefore, the input of the third convolutional layer is a concatenation of the output from the previous convolutional layer and the depth feature maps.

3.3. Parameter-Learned Upsampling Module

We employ a parameter-learned upsampling module[37] instead of bilinear interpolation to adaptively upsample the estimated inverse depth at the fixed 1/8 input resolution to full resolution. The upsampling module treats the full resolution inverse depth at each pixel to be a convex combination of the 3×3 grid of its neighborhoods at 1/8 input resolution. It takes the feature maps from the third convolutional layer in the depth decoder, and then leverages two consecutive convolutional layers to produce a convex mask. The convex mask is then performed over $Softmax$ to control the weights for the 9 neighborhoods at 1/8 input resolution, and used to retrieve the inverse depth at full resolution.

3.4. Iterative Updates

Our update procedure produces a series of inverse depth maps $\{\hat{d}_1, ..., \hat{d}_N\}$ from an initial starting point $\hat{d}_0 = 0$. At each update, it obtains the current estimation $\hat{d}_n$ by generating an update direction $\triangle\hat{d}$ which is exploited to the last estimation $\hat{d}_{n-1}$: $\hat{d}_n = \triangle\hat{d} + \hat{d}_{n-1}$. Then, it performs a $Sigmoid$ nonlinearity over the current estimation $\hat{d}_n$ to get the inverse depth d_n: $d_n = Sigmoid(\hat{d}_n)$.

We feed the multi-scale feature maps from the depth encoder to iteratively update an inverse depth in the order of X_3, X_2, and X_1 , which mimics the steps of the traditional coarse-to-fine architecture. For maintaining semantically richer while spatially more precise representations during the iterative updates, we embed a multi-scale feature modulation module(MSFM) at the beginning of the depth decoder. This module leverages a convolution-based gated recurrent unit(GRU)[5] to modulate the content between previous activation h_{t-1} and current input x_t. The goal of the module is to find the most suitable activation h_t for each update except the first one, which can be formulated as:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{1}$$

where the update gate z_t controls how much information the current hidden activation needs to remember, and $\tilde{h}_t$ is the current hidden activation. The update gate is therefore:

$$z_t = \sigma(Conv_Z([x_t, h_{t-1}])), \tag{2}$$

where $\sigma(.)$ is a sigmoid activation function, $[.]$ is the concatenation operator, and $Conv_Z$ is a separable convolutional unit consists of two operations: one with a 1×3 convolutional layer and one with a 3×1 convolutional layer for reducing model parameters while maintaining accuracy. The current hidden activation depends on the current input x_t and previous activation h_{t-1}, which can be formulated as:

$$\tilde{h}_t = tanh(Conv_H([x_t, r_t \odot h_{t-1}])), \tag{3}$$

where the reset gate r_t modulates the extent to which the previous activation is forgotten, which is computed by:

$$r_t = \sigma(Conv_R([x_t, h_{t-1}])), \tag{4}$$

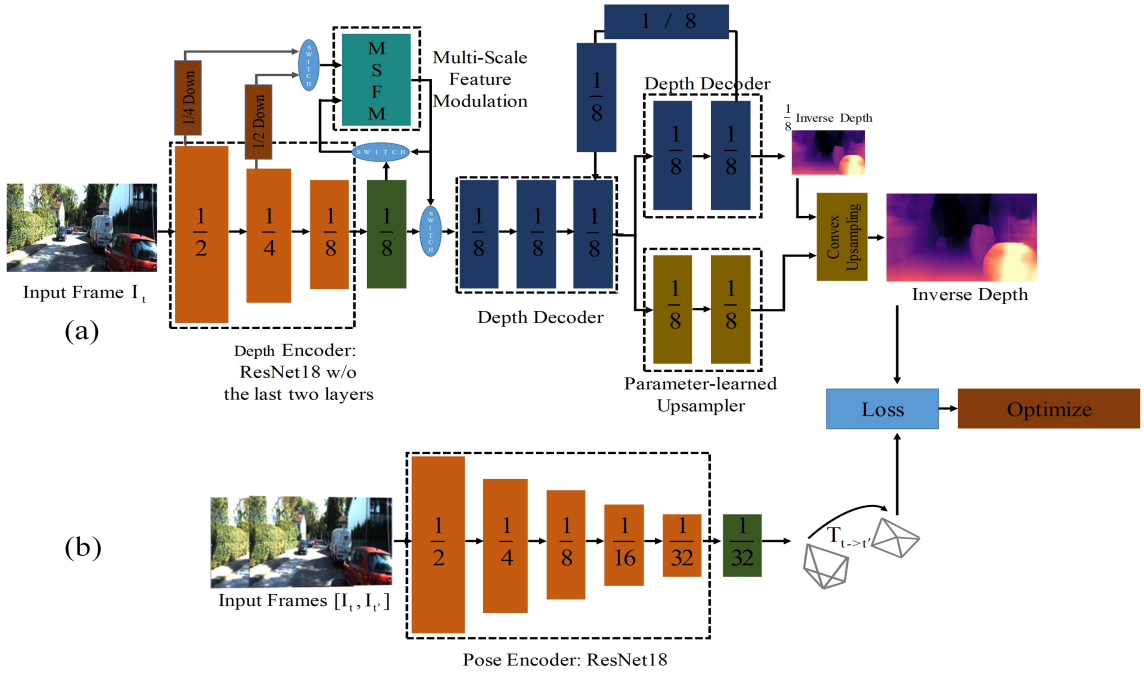

Figure 1. **Overall Architecture.** (a): The input frame is encoded by a ResNet18(w/o the last two blocks) to produce multi-scale feature maps at 1/2, 1/4, and 1/8 input resolution. These feature maps are then unified into the same size and sequentially fed to a parameter-shared depth decoder to iteratively update an inverse depth. In addition, we employ a MSFM module to maintain semantically richer while spatially more precise representations during iterative updates. Finally, we learn a convex mask to upsample the estimated inverse depth to full resolution in each update. (b) The concatenation of input frames is computed by PoseNet to get a single 6-DoF relative pose.

where $Conv_H$ and $Conv_R$ are separable convolutional units, which do not share weights. We naturally update the inverse depth for three times due to the feature maps of three scales from the depth encoder. In particular, we could apply an extra convolutional layer to the feature maps modulated by the MSFM module at each scale, resulting in six updates for the inverse depth.

3.5. Self-supervision

Following the pervious work[14], we adopt a ResNet18-based PoseNet as shown in Figure 1(b) to estimate the relative pose $T_{t->t'}$ between target image I_t and source image $I_{t'}$, which can be formulated as follows:

$$T_{t->t'} = PoseNet(I_t, I_{t'}). \tag{5}$$

In order to keep our training process robust against occlusion, the masked photometric re-projection loss L_p[14] is used, as in :

$$L_p = \sum_{i=1}^{N} \beta^{N-i} \cdot \min_{t'} \mu(I_t, I_{t'}, I^i_{t'->t}) \odot pe(I_t, I^i_{t'->t}), \tag{6}$$

where N is the number of updates, $\odot$ denotes element-wise multiplication, and update term β weights the $pe(.)$ loss exponentially increasing with the number of updates. Additionally, $t' \in (t-1, t+1)$ as source images, indicates the past and future images with respect to the target image I_t, and $\mu(.)$ is a binary mask and responsible for removing pixels that do not have relative motion between target and source images, which is defined by :

$$\mu = [\min_{t'} pe(I_t, I^i_{t'->t}) < \min_{t'} pe(I_t, I_{t'})], \tag{7}$$

where [.] denotes the Iverson bracket, $I^i_{t'->t}$ indicates the warped source images at update i, as in:

$$I^i_{t'->t} = I_{t'} \langle proj(T_{t->t'}, D^i_t, K) \rangle, \tag{8}$$

where K are the intrinsics which are identical for all images, $proj(.)$ are the resulting 2D coordinates of the projected depths D^i_t at update i in $I_{t'}$, $\langle\rangle$ is locally sub-differentiable bilinear sampler, and $pe(.)$ is minimum per-pixel photometric re-projection loss using SmoothL1[12] and SSIM[41], as in:

$$\begin{aligned} pe(I_1, I_2) = &\frac{\alpha}{2}(1 - SSIM(I_1, I_2)) \\ &+ (1-\alpha) SmoothL1(I_1, I_2), \end{aligned} \tag{9}$$

where $\alpha = 0.85$. Following [14], we use an additional edge-aware smoothness to smooth the estimated depth, which is formulated as :

$$L_s = \sum_{i=1}^{N} \beta^{N-i} |\partial_x d^{i*}_t| e^{-\partial_x I_t} + |\partial_y d^{i*}_t| e^{-\partial_y I_t}, \tag{10}$$

		Depth Error(↓)				Depth Accuracy(↑)			Model Size(↓)
Method	Train	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Parameters
Zhou[46]	M	0.183	1.595	6.709	0.270	0.734	0.902	0.959	31.6M
GeoNet[44]	M	0.149	1.060	5.567	0.226	0.796	0.935	0.975	31.6M
DDVO[39]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974	28.1M
Monodepth [13]	M	0.148	1.344	5.927	0.247	0.803	0.922	0.964	20.2M
EPC++[27]	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976	33.2M
Struct2depth[3]	M	0.141	1.026	5.291	0.215	0.816	0.945	0.979	31.6M
Monodepth2[14]	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981	14.3M
Monodepth2 (1024 × 320)[14]	M	0.115	0.882	4.701	0.190	0.879	0.961	0.982	14.3M
Johnston[19]	M	**0.106**	0.861	4.699	0.185	**0.889**	0.962	0.982	14.3M+
Zhao(832 × 256) [45]	M	0.113	**0.704**	4.581	**0.184**	0.871	0.961	**0.984**	14.3M
PackNet-SfM [15]	M	0.111	0.785	4.601	0.189	0.878	0.960	0.982	128M
PackNet-SfM(1280 × 384) [15]	M	0.107	0.802	4.538	0.186	**0.889**	0.962	0.981	128M
R-MSFM3 w/o pretraining	M	0.128	0.965	5.019	0.207	0.853	0.951	0.977	**3.5M**
R-MSFM6 w/o pretraining	M	0.126	0.944	4.981	0.204	0.857	0.952	0.978	3.8M
R-MSFM3	M	0.114	0.815	4.712	0.193	0.876	0.959	0.981	**3.5M**
R-MSFM6	M	0.112	0.806	4.704	0.191	0.878	0.960	0.981	3.8M
R-MSFM3(1024 × 320)	M	0.112	0.773	4.581	0.189	0.879	0.960	0.982	**3.5M**
R-MSFM6(1024 × 320)	M	0.108	0.748	**4.470**	0.185	**0.889**	**0.963**	0.982	3.8M
Monodepth2-R50[14]	M	0.110	0.831	4.642	0.187	0.883	0.962	0.982	**32.5M**
FeatDepth(1024 × 320)-Res50[34]	M	**0.104**	**0.729**	**4.481**	**0.179**	**0.893**	**0.965**	**0.984**	35.2M
UnDeepVO[22]	MS	0.183	1.730	6.57	0.268	-	-	-	-
EPC++[27]	MS	0.128	0.935	5.011	0.209	0.831	0.945	0.979	-
Monodepth2 [14]	MS	0.106	0.818	4.750	0.196	0.874	0.957	0.979	14.3M
Monodepth2 (1024 × 320) [14]	MS	0.106	0.806	4.630	0.193	0.876	0.958	0.980	14.3M
D3VO [42]	MS	**0.099**	0.763	4.485	**0.185**	0.885	0.958	0.979	-
R-MSFM3	MS	0.112	0.799	4.639	0.190	0.881	0.96	0.981	**3.5M**
R-MSFM6	MS	0.111	0.787	4.625	0.189	0.882	0.961	0.981	3.8M
R-MSFM3 (1024 × 320)	MS	0.112	**0.753**	4.530	0.189	0.881	0.961	**0.982**	**3.5M**
R-MSFM6 (1024 × 320)	MS	0.108	**0.753**	**4.469**	**0.185**	**0.888**	**0.963**	**0.982**	3.8M

Table 1. **Comparison of our models to existing methods on the KITTI Eigen split[8].** The best results for each metric are shown in **bold**, and the second are underlined. The table records the results of training the model using the two different strategies seen in the *Train* column: M means the model is trained with only self-supervised mono supervision, and MS means the model is trained with both self-supervised mono and stereo supervision.

where $d_t^{i*} = d_t^i / \bar{d}_t^i$ is the mean-normalized inverse depth[39] at update i , which prevents the inverse depth from approaching zero and thus to increase training stability. In the end, the final loss L is combined as the weighted sum of L_p Equation 6 and L_s Equation 10, which is formulated as :

$$L = L_p + \lambda L_s, \tag{11}$$

where λ is the smoothness regularisation term.

4. Experiments

We use the data split of Eigen *et al.* [8] to train and evaluate our model. Before training, we remove the static images from the training set following Zhou *et al.* [46]. These results in 39810 training sequences which includes three consecutive frames for monocular training and an additional stereo counter for mixed training, and 4,424 validating sequences. To recover scale information, we adopt the per-image median ground truth scaling[46]. When evaluating our model, we restrict the depth estimations to a fixed depth range between 0m and 80m and compare its performance with other state-of-the-art approaches by five widely used evaluation indicators proposed in [9]: $AbsRel$, $SqRel$, $RMSE$, $RMSElog$, and $Accuracies$ which are formulated as follows:

- $AbsRel = \frac{1}{|N|} \sum_{i \in N} \frac{|d_i - d_i^*|}{d_i^*}$,
- $SqRel = \frac{1}{|N|} \sum_{i \in N} \frac{\|d_i - d_i^*\|_2}{d_i^*}$,
- $RMSElog = \sqrt{\frac{1}{|N|} \sum_{i \in N} \|log(d_i) - log(d_i^*)\|_2}$,
- $RMSE = \sqrt{\frac{1}{|N|} \sum_{i \in N} \|d_i - d_i^*\|_2}$,
- $Accuracies = max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) = \delta < threshold$,

where N is the total number of pixels for depth ground truth, d_i denotes the predicted depth value at pixel i, and d_i^* stands for the ground truth at pixel i. In addition, $threshold$ controls the percentage of correct pixels in the estimated depth, which can be taken as 1.25, 1.25^2, 1.25^3.

We train our model in a self-supervised learning manner with different training sets(monocular triplets(M) and monocular + stereo quadruplets(MS)), updates, and input

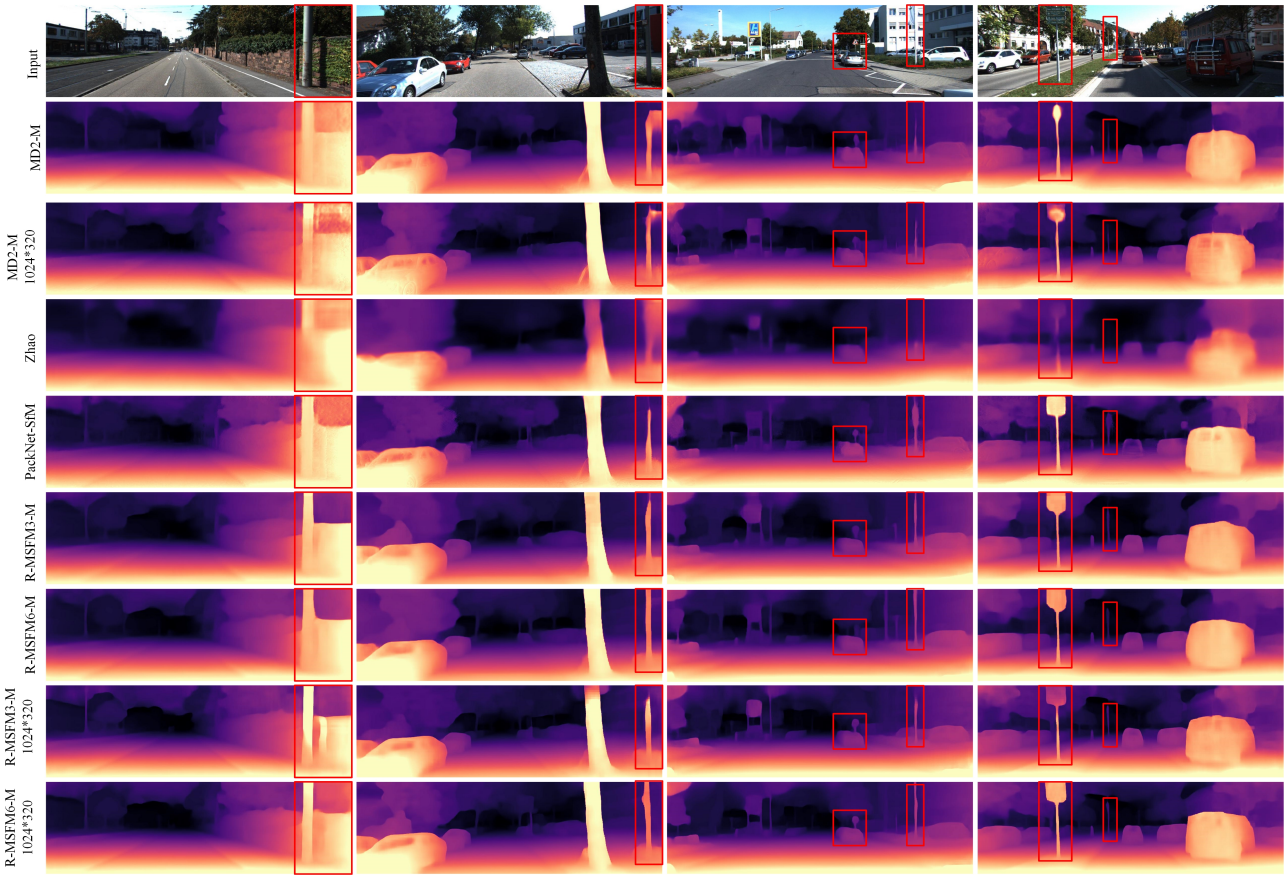

Figure 2. **Qualitative results on the KITTI Eigen split[8] test set.** Our models can robustly estimate sharper depths for intricate objects in the reflective and color-saturated regions.

Experiments	Train	Beginning	Middle	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Parameters
Monodepth2[14]	M	-	-	0.115	0.903	4.863	0.193	0.877	0.959	**0.981**	14.3M
R-MSFM3-A	M	×	×	0.120	0.889	4.869	0.199	0.867	0.956	0.980	2.9M
R-MSFM3-B	M	×	√	0.118	0.860	4.816	0.198	0.867	0.956	0.980	4.3M
R-MSFM3-C	M	√	×	**0.114**	**0.815**	4.712	0.193	0.876	0.959	**0.981**	**3.5M**
R-MSFM3-D	M	√	√	0.115	0.828	**4.702**	**0.192**	**0.877**	**0.960**	**0.981**	4.8M
R-MSFM6-C	M	√	×	**0.112**	**0.806**	4.704	**0.191**	**0.878**	**0.960**	**0.981**	3.8M

Table 2. **Ablation study on our R-MSFM architecture**, on the KITTI Eigen split[8] test set at 640×192 resolution. We evaluate the impact of the parameter-shared depth decoder, multi-scale feature modulation(MSFM), and the iterative updates. *Beginning* and *Middle* indicate the position that we embed our MSFM module. All models were trained with the same settings.

resolutions, resulting in different variants of our model. We compare our models with other state-of-the-art approaches and show that they get satisfying results with the fewest model parameters, as shown in Table 1.

4.1. Implementation Details

R-MSFM is implemented in PyTorch[30] and trained on a single Nvidia Titan RTX for 40 epochs with a batch size of 12. Following the previous work[14], we take the weights of ResNet18 on ImageNet[32] as the initialization of our depth and pose encoder. During training, both the depth and pose networks are optimized with AdamW[26] optimizer, whose initial learning rate and weight decay are set to $2e^{-4}$ and $5e^{-5}$ respectively. The gradients through our model are clipped to a fixed range[-1,1], and the resolution of input/output is resized to 640×192 by default. Additionally, the smoothness regularisation term λ and the update term β are set to 0.001 and 0.9 respectively. To alleviate the overfitting of our model during training, the following data augmentations are used with 50% chance: horizontal flips, random saturation(± 0.2), random brightness(± 0.2), random contrast(± 0.2), and hue jitter(± 0.1).

4.2. KITTI Results

The experimental results on the KITTI Eigen split test set[8]) are showed in Table 1. When comparing to oth-

Experiments	Train	Update	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
R-MSFM3 (1024 × 320)	M	1	0.137	0.957	5.086	0.216	0.825	0.945	0.978
R-MSFM3 (1024 × 320)	M	2	0.116	0.805	4.672	0.193	0.869	0.958	0.981
R-MSFM3 (1024 × 320)	M	3	**0.112**	**0.773**	**4.581**	**0.189**	**0.879**	**0.960**	**0.982**
R-MSFM6 (1024 × 320)	M	1	0.141	0.969	5.136	0.218	0.816	0.944	0.978
R-MSFM6 (1024 × 320)	M	2	0.117	0.804	4.678	0.193	0.869	0.958	**0.982**
R-MSFM6 (1024 × 320)	M	3	0.111	0.757	4.539	0.187	0.882	0.961	**0.982**
R-MSFM6 (1024 × 320)	M	4	0.109	0.755	4.502	**0.185**	0.887	0.962	**0.982**
R-MSFM6 (1024 × 320)	M	5	**0.108**	0.751	4.482	**0.185**	**0.888**	**0.963**	**0.982**
R-MSFM6 (1024 × 320)	M	6	**0.108**	**0.748**	**4.47**	**0.185**	**0.888**	**0.963**	**0.982**

Table 3. **Quantification of iterative updates.** Results of each update of our model under different iterative updates on the KITTI Eigen split test set[8].

er state-of-the-art methods which adopt a self-supervised training strategy, our method produce comparable results while significantly reducing the model parameters. As can be seen in Table 1, our method outperforms the baseline Monodepth2[14] by a significant margin using a quarter of parameters of it, and gets close to the current state-of-the-art PackNet-SfM[15] using only three percent of parameters of it. In addition, we conduct high-resolution(1024 × 320) training following the previous work [14], and this operation leads our model to outperform all the existing methods with the same training schedule[14]. Moreover, we can get comparable results to FeatDepth[34], which takes advantage of a robust feature-metric self-supervised supervision and a stronger ResNet50 encoder. The overall qualitative results are reported in Figure 2. As can be seen, our method gives satisfactory results for weak texture regions(column 1 and column 4) and thin structures(column 2, column 3, and column 4) from all three methods and their variants. These quantitative and qualitative results demonstrate the superiority of our method. In addition, this implies that our R-MSFM benefits more from iterative updates, which forces the network to learn a coarse inverse depth from high-level feature maps and then refine its boundary region from low-level ones. However, when there are moving objects in the scene, our R-MSFM fails to learn good depths for them as with all self-supervised depth estimation methods as shown in Figure 4. This is restricted by the self-supervised loss, which breaks at the regions with the moving objects.

4.3. KITTI Ablation Study

Table 2 shows an ablation study of our proposed R-MSFM, where we first start from the baseline Monodepth2[14] (row one). Next, by removing the last two blocks in its depth encoder, and employing a parameter-shared depth decoder to perform three updates for the inverse depth, we get our R-MSFM3-A model (row two). Then, by applying an additional MSFM module, we get three variants of our model: R-MSFM3-B, R-MSFM3-C, R-MSFM3-D respectively, which only differ in the position of applying the module. R-MSFM3-B model (row three) applies the module after the third convolutional layer in the depth decoder, R-MSFM3-C model (row four) applies the module at the beginning of the depth decoder, and R-MSFM3-D model (row five) applies the module in both positions. Finally, by performing three extra updates for R-MSFM3-C, we get our R-MSFM6-C model. When comparing to the baseline model Monodepth2, all the models above demonstrate their superiority.

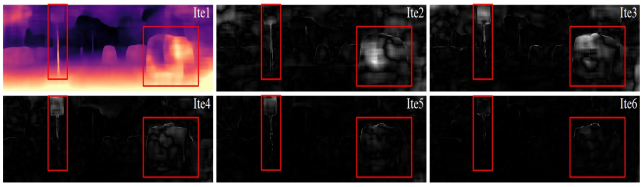

Figure 3. **Visualization of iterative updates.** We show the estimated depth from the first update and the residual refinements from subsequent updates. The brightness value of the pixel indicates the magnitude of the refinement, where the brighter the pixel, the greater the degree of the refinement. The pixels in the red box areas indicate that the refinement decreases with the number of updates and tends to become saturated after five updates.

Benefits of Parameter-Shared Depth Decoder The baseline Monodepth2[14] adopts a coarse-to-fine architecture, which gradually reduces the resolution of the input image to get aggregated strong low-level representations by a depth encoder, and then correspondingly increases its resolution until the full resolution to infer multi-scale depths by several depth decoders. This architecture works fine for most cases, however, it is limited by the excessive parameters and error propagation. Thanks to the parameter-shared depth decoder, our R-MSFM-A achieves similar results compared to Monodepth2, but its parameters are only 20% of Monodepth2, demonstrating the effectiveness of our architecture.

Benefits of MSFM Module Applying our MSFM Module to R-MSFM-A model always brings an improvement in performance. However, the position where it is embedded also affects. As can be seen in Table 2, R-MSFM3-C, which

	Encoder			Decoder			Full		
Method	Params(M)	FLOPs(B)	Speed(ms)	Params(M)	FLOPs(B)	Speed(ms)	Params(M)	FLOPs(B)	Speed(ms)
Monodepth2[14]	11.2	4.5	9.1	3.1	3.5	3.9	14.3	8.0	13.0
PackNet-SfM[15]	121	187	186.4	7	18	13.9	128	205	200.3
R-MSFM3	0.7	2.4	4.7	2.8	14.1	9.1	3.5	16.5	13.8
R-MSFM6	0.7	2.4	4.7	3.1	28.8	17.7	3.8	31.2	22.4

Table 4. **The comparison of the distribution of the parameters and inference time.** All results are tested on a single RTX2060 GPU with an input image size of 640×192. Specially, for testing speed we run 300 times and averaged the last 250 to warm up our machine.

embeds the MSFM module at the beginning of the depth decoder, provides the most incremental performance gains with the lowest computational cost. In addition, adding an extra MSFM module does not ideally lead to performance improvements since its structural complexity.

Benefits of Iterative Updates More updates always lead to performance improvement for our model, as can be seen in Table 1. Especially when feeding high-resolution input images, the accuracy of the model is dramatically improved. One reason is that doing extra updates with high-resolution feature maps provides the decoder more information than low-resolution ones. Table 3 and Figure 3 illustrate the benefits of iterative updates, indicating that the first update is responsible for coarse estimation, while the remaining updates are responsible for progressive refinements, especially in the regions containing intricate objects. It should, however, be noted that the accuracy is saturated at update 6, which primarily focuses on small boundaries rather than large targets. Finally, running for more iterations than models are trained for would degrade performance. Since the rest multi-scale feature maps are not directly involved in the optimization process of the parameter-shared decoder and GRU-based MSFM module.

4.4. Complexity Analysis

The flexibility of the monocular depth estimation system makes it expressly attractive for practical deployment. Therefore, it is necessary to analyze its complexity. Different from the existing models[14, 15, 45] based on U-Net-like coarse-to-fine architecture, our R-MSFM features a small part of the traditional encoder(excludes the last two computationally heavy blocks), such as ResNet18, and a parameter-shared decoder. Table 4 details the statuses of each part of our model including inference speed, floating-point operations (FLOPs), and parameter used, and compares them with other state-of-the-art methods. As can be seen, excluding the last two computationally heavy blocks from ResNet18 significantly decreases the parameters used, which is reduced by 94% compared to Monodepth2[14]. On the other hand, the residual addition in ResNet inevitably restricts the inference speed, resulting in its acceleration rate by 48% compared to Monodepth2. When coming to the decoder part, the procedure of our iterative update inevitably increases computational overhead(FLOPs). However, the plain topology of our decoder without residual addition makes the inference computational efficient, and takes roughly 4.75B FLOPs and 3ms per update. Finally, the parameter-economical, computationally efficient, and accurate architecture lets our R-MSFM suitable for deployment on embedded platforms.

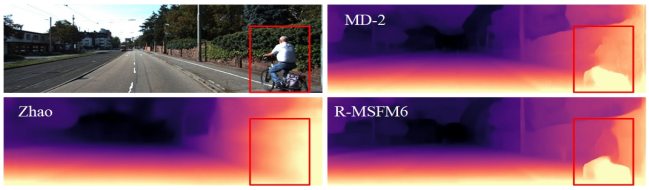

Figure 4. **Failure cases.** Moving objects have always been challenging for monocular depth estimation which will typically get the depth correct close to the ground and the accuracy will worsen as it goes up.

5. Conclusion

We have presented R-MSFM—Recurrent Multi-Scale Feature Modulation—a novel end-to-end trainable model for self-supervised monocular depth estimation. It leverages the multi-scale feature maps extracted from the depth encoder to iteratively update an inverse depth through the parameter-shared depth decoder, avoiding the error propagation from low to high resolution. Furthermore, R-MSFM embeds a multi-scale feature modulation(MSFM) module at the beginning of the depth decoder, maintaining semantically richer while spatially more precise representations during iterative updates. In addition, it adopts a parameter-learned upsampler instead of bilinear interpolation to upsample the estimated inverse depth, preserving its motion boundaries. Both the high accuracy and lightweight characteristics demonstrate that our R-MSFM is suitable in practical applications.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (61801169, 61801168) and the Fundamental Research Funds for the Central Universities (B210202087).

References

[1] Markus Achtelik, Abraham Bachrach, Ruijie He, Samuel Prentice, and Nicholas Roy. Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments. In *Unmanned Systems Technology XI*, volume 7332, page 733219. International Society for Optics and Photonics, 2009.

[2] Filippo Aleotti, Fabio Tosi, Matteo Poggi, and Stefano Mattoccia. Generative adversarial networks for unsupervised monocular depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[3] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8001–8008, 2019.

[4] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[5] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[6] Mingyu Ding, Yuqi Huo, Hongwei Yi, Zhe Wang, Jianping Shi, Zhiwu Lu, and Ping Luo. Learning depth-guided convolutions for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1000–1001, 2020.

[7] Gregory Dudek and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2010.

[8] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[9] 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.

[10] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European conference on computer vision*, pages 740–756. Springer, 2016.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[13] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[14] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 3828–3838, 2019.

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

[16] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European conference on computer vision*, pages 345–360. Springer, 2014.

[17] 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.

[18] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE, 2005.

[19] Adrian Johnston and Gustavo Carneiro. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4756–4765, 2020.

[20] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[21] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.

[22] R Li, S Wang, Z Long, and D UnDeepVO Gu. Monocular visual odometry through unsupervised deep learning. arxiv 2017. *arXiv preprint arXiv:1709.06841*.

[23] Hongmin Liu, Xincheng Tang, and Shuhan Shen. Depth-map completion for large indoor scene reconstruction. *Pattern Recognition*, 99:107112, 2020.

[24] Pengpeng Liu, Irwin King, Michael R Lyu, and Jia Xu. Flow2stereo: Effective self-supervised learning of optical flow and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6648–6657, 2020.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[27] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic un-

derstanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2624–2641, 2019.

[28] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018.

[29] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[31] Pierluigi Zama Ramirez, Matteo Poggi, Fabio Tosi, Stefano Mattoccia, and Luigi Di Stefano. Geometry meets semantics for semi-supervised monocular depth estimation. In *Asian Conference on Computer Vision*, pages 298–313. Springer, 2018.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[33] Evan Shelhamer, Jonathan T Barron, and Trevor Darrell. Scene intrinsics and depth from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 37–44, 2015.

[34] Chang Shu, Kun Yu, Zhixiang Duan, and Kuiyuan Yang. Feature-metric loss for self-supervised learning of depth and egomotion. In *European Conference on Computer Vision*, pages 572–588. Springer, 2020.

[35] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[36] Jiaming Sun, Linghao Chen, Yiming Xie, Siyu Zhang, Qinhong Jiang, Xiaowei Zhou, and Hujun Bao. Disp r-cnn: Stereo 3d object detection via shape prior guided instance disparity estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10548–10557, 2020.

[37] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *arXiv preprint arXiv:2003.12039*, 2020.

[38] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[39] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, 2018.

[40] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[42] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1281–1292, 2020.

[43] Xinchen Ye, Zheng Li, Baoli Sun, Zhihui Wang, Rui Xu, Haojie Li, and Xin Fan. Deep joint depth estimation and color correction from monocular underwater images based on unsupervised adaptation networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[44] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1983–1992, 2018.

[45] Wang Zhao, Shaohui Liu, Yezhi Shu, and Yong-Jin Liu. Towards better generalization: Joint depth-pose learning without posenet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9151–9161, 2020.

[46] Chao Zhou, Hong Zhang, Xiaoyong Shen, and Jiaya Jia. Unsupervised learning of stereo matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1567–1575, 2017.

[47] Laurent Zwald and Sophie Lambert-Lacroix. The berhu penalty and the grouped effect. *arXiv preprint arXiv:1207.6868*, 2012.