Weakly-Supervised Optical Flow Estimation for Time-of-Flight Supplementary Material

Michael Schelling, Pedro Hermosilla, Timo Ropinski
Ulm University, Germany

1. Contents

This supplementary material provides additional information about our method in Sec. 2 and Sec. 3 and further details on how the experiments were conducted in Sec. 4. Finally, we show additional qualitative results in Sec. 5.

Our code, trained networks and the additional scenes to expand the CB-dataset [11], containing moving objects, are available at `https://github.com/schellmi42/WFlowToF`.

2. Phase Unwrapping of the ToF Loss Function

In this section we provide more information on the phase unwrapping of the gradients of the ToF loss function $\mathcal{L}_{ToF}$, which is given through

$$s = \text{sign}(\hat{m}_0 - \hat{m}_2) \tag{1}$$

$$\hat{d} = \frac{c}{4\pi f} \arctan\left(\frac{\hat{m}_3 - \hat{m}_1}{\hat{m}_0 - \hat{m}_2 + s \cdot \epsilon}\right), \tag{2}$$

$$\mathcal{L}_{ToF} = \|\hat{d} - d_{ToF}\|_1, \tag{3}$$

where $\hat{m}_i$ are the iToF measurements after warping, and ϵ is a small positive constant. While the standard $\arctan$ function has a range limited to a semi-circle $(-\pi/2, \pi/2)$, the sign of the numerator and the denominator in Eq. (2) can be used to extend the range to a full circle $(-\pi, \pi]$. This method is commonly referred to as the $\arctan 2$ function

$$x = \hat{m}_0 - \hat{m}_2, \tag{4}$$

$$y = \hat{m}_3 - \hat{m}_1, \tag{5}$$

$$\hat{d} = \frac{c}{4\pi f} \arctan 2(y, x + s \cdot \epsilon). \tag{6}$$

As a consequence, the $\arctan 2$ function has multiple branches, corresponding to the sign of its arguments, as can be seen in Fig. 1.

The figure also illustrates the difference of $d_{max}/2$ between the two branches in this case. As a result if the target (red point) is on the different branch than the current depth estimate (green point), the direction of the optimization needs to be inverted in order to move the estimate through the phase wrapping of the $\arctan 2$ function and change to the correct branch. This is realized by our proposed gradient correction presented in the main paper

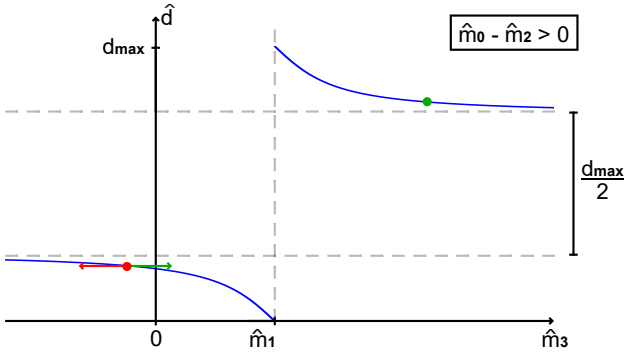

Figure 1. Reconstructed depth $\hat{d}$ dependent on the measurement $\hat{m}_0$ in the case of a positive denominator. The $\arctan 2$ function changes branches at $\hat{m}_3 = \hat{m}_1$, which introduces a discontinuity. As a result, if the prediction (red point) and the target value (green point) are on separate branches, the gradient points in the wrong direction (red arrow), and is corrected by our method (green arrow). The branches are separated by $d_{max}/2$, in line with Eq. (7).

$$\nabla \mathcal{L}_{ToF,PU} = \begin{cases} \nabla \mathcal{L}_{ToF}, & 0 \leq \mathcal{L}_{ToF} < d_{max}/2, \\ -\nabla \mathcal{L}_{ToF}, & \mathcal{L}_{ToF} \geq d_{max}/2. \end{cases} \tag{7}$$

To show the influence on the optimization we conduct a toy experiment, in which we formulate a simple reconstruction task. We assume m_0, m_1, m_2 and d_{ToF} are given and the task is to reconstruct the measurement m_3 by minimizing the ToF loss $\mathcal{L}_{ToF}$

$$\min_{\hat{m}_3} \mathcal{L}_{ToF} \tag{8}$$

$$= \min_{\hat{m}_3} \left\| \frac{c}{4\pi f} \arctan 2\left(\hat{m}_3 - m_1, m_0 - m_2\right) - d_{ToF} \right\|_1. \tag{9}$$

We initialize $\hat{m}_0 = 0$ and optimize it using simple gradient descent, with and without applying our gradient correction method. Without gradient correction only parts initialized

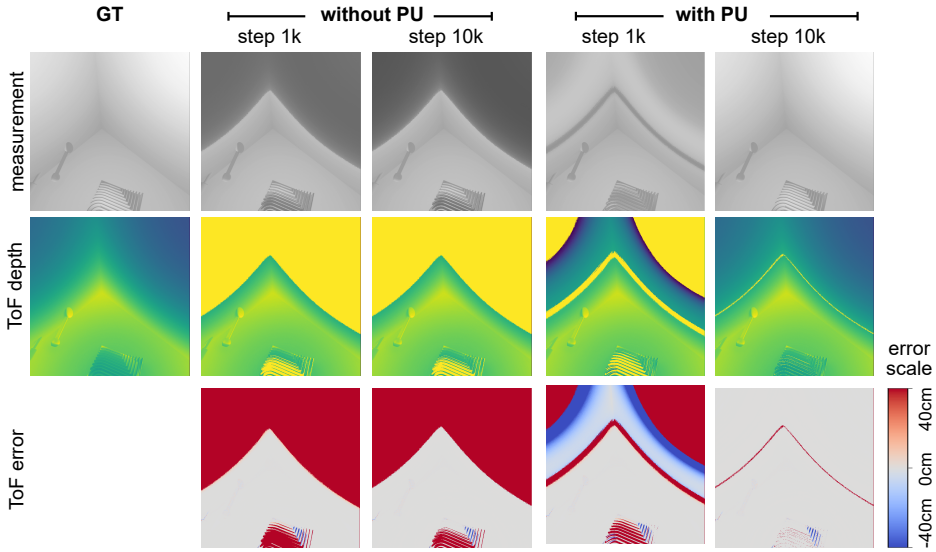

Figure 2. Reconstruction of measurement $\hat{m}_3$ by minimizing the ToF loss. Without phase unwrapping (PU) only partial reconstruction is possible. After correcting the gradients with our method, the phase wrapping is successively resolved by the optimization (right).

on the correct branch are reconstructed, wheres after gradient correction all parts can be reconstructed, as is shown in Fig. 2.

3. Regularization Losses

This section provides more insights on the effect of the regularization losses.

Without regularizations the problem of supervising the four measurements with a single depth is under-determined, *e.g.* in Eq. (6) the ratio of y/x is the determining factor, but not the individual values. While the search space is already limited when predicting optical flows, as not arbitrary values are allowed, but only values from a local neighborhood can be warped to a certain position, still regularization is necessary to further restrict the network predictions. Moreover, unsupervised Optical Flow (OF) networks require such regularizations in general to achieve competitive performance [5].

In our work we introduced two main regularizations which measure consistency in the image space. The smoothing loss $\mathcal{L}_{smooth}$ measures region consistency between the predicted flow and the input image, and is applied before warping. The edge-aware loss $\mathcal{L}_{edge}$ measures edge consistency between the warped image and the target image, and is applied after warping.

The impact of these losses on the warped images can be seen in Fig. 3

4. Experiments

In this section we provide detailed information about the hyperparameters used for training the networks.

4.1. Implementation

All custom implementations were done in `PyTorch 1.10.+cu102` [10] and `Python 3.6`. For the OF networks FFN [7] and PWC [13], and the warping operation, we use implementations provided in the PyTorch library `ptlflow 0.2.5` [9].

4.2. Motion Compensation

Our Method. Both OF backbone networks FFN and MOM [4] are trained with the ADAM [6] optimizer using a learning rate scheduler which decays the learning rate by

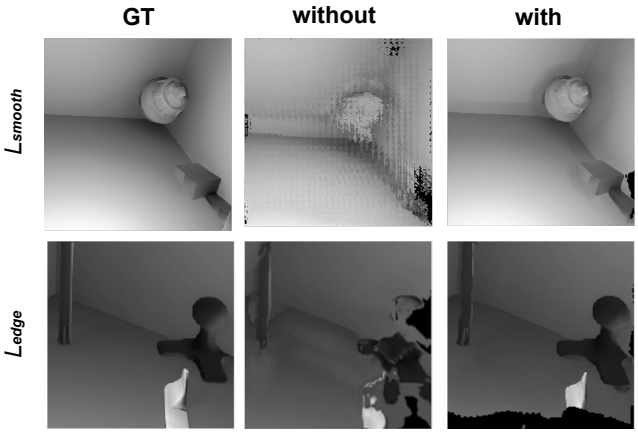

Figure 3. Effect of the regularization losses on the warped image. The smoothing loss $\mathcal{L}_{smooth}$ ensures that pixels that belong to a visually similar region are moved in the same direction (top row). With the edge aware loss $\mathcal{L}_{edge}$ the edges of the warped image align with the target image, preserving object boundaries and details. (bottom row)

a factor 0.5 when the ToF loss on the validation set did not decrease for 50 epochs.

We augment the input data by simulating shot noise on the iToF measurements, following the noise model described by Schelling *et al*. [11]. Additionally, we use random image rotations by $0°, 90°, 180°, 270°$, random mirroring along the image axes, and crop random 512×512 image patches during training.

We train the FFN network with the combination of all losses

$$\mathcal{L}_{ToF} + \lambda_{smoooth} \cdot \mathcal{L}_{smooth} + \lambda_{edge} \cdot \mathcal{L}_{edge} + \lambda_{sim} \cdot \mathcal{L}_{sim}. \tag{10}$$

We compared the following values to select the hyperparameters: weights λ_i from {1, 1e-1, 1-e2, 1e-3, 1e-4}, and the shift parameter s in the edge-aware loss from {1e-2, 1-e1, 0, 1e1, 1e2, 1e3}. The results reported in the main paper were achieved with $\lambda_{smooth} = 1, \lambda_{edge} =$ 1e-1, $\lambda_{sim} =$ 1e-2, $s =$ 1e2 in the single frequency case, and in the multi frequency case only the similarity weight was changed to $\lambda_{sim} = 1e-2$. In all experiments the cosine similarity was used in $\mathcal{L}_{sim}$. In the single frequency experiments we train with a batch size of 8, and in the multi-frequency experiment with a batch size of 4. The initial learning rate is set to 1e-3.

For the MOM network we do not use the similarity loss, as the encoder decoder architecture does not have latent features for a cost volume computation, thus we set $\lambda_{sim} = 0$. We perform the same hyperparameter tuning as for the FFN network. The results in the main paper were achieved with$\lambda_{smooth} = 1, \lambda_{edge} = 1, s =$ 1e3 in both the single and the multi-frequency experiments. We train with a batch size of 1 and an initial learning rate of 1e-5, as recommended by the authors of MOM [4],

Pre-Trained Networks. The pretrained networks, FFN and PWC were trained on RGB data and hence require three input channels. In our experiments we normalize the iToF measurements to a range of [0, 255] and repeat the the scalar image three times to match the RBG input. We use weights pre-trained on the Sintel dataset [2].

UFlow. The UFlow [5] method is trained on the same data as our method, using the TensorFlow2 implementation provided by the authors. We use the hyperparameters recommended in the documentation for custom datasets, which correspond to the settings for the Flying Chairs dataset [3] in the UFlow paper.

Lindner Method. For the Lindner method, we match the input dimensionality of the pre-trained networks using the same scheme as above on the intensity computed with Lindner's method.

	SF 1T	SF 2T	MF 1T	MF 2T	MF 4T
PWC (PT)	13.70	4.03	16.01	7.51	5.41
RAFT (PT)	8.48	4.08	14.72	6.55	4.63
Our (FFN)	5.81	3.66	13.77	4.43	3.03

Table 1. Resulting $\mathcal{L}_{ToF}$ of a pre-trained (PT) RAFT in comparison to a pre-trained PWC and our method using FFN as backbone. Results on the test set.

Comparison to RAFT (SotA) As the task involves the prediction of multiple optical flows at once, only lightweight OF networks, such as FFN, can be trained in this setting. More advanced architectures, such as RAFT [14], which is currently the State-of-the-Art (SotA) in supervised RGB OF prediction, increase the computational cost and we were unable to train it as a backbone network. While a pre-trained RAFT model achieves a better performance than the simpler PWC (see Tab 1), the inference times are much slower (see Tab. 2). Still the pre-trained RAFT model is outperformed by our method with a much smaller FFN backbone (see Tab. 1).

4.3. Inference time

As our approach is a training algorithm, it does not affect the evaluation times of the backbone OF networks. As stated in the main paper, the encoder decoder architecture of MOM allows fast execution times almost independent of the number of predicted flows. In contrast, the runtime of networks using derivatives of the more advanced cost-volume architecture [3] grows linearly corresponding to the number of predicted flows. Prediction times for the networks used in this work and a comparison to the SotA RAFT network are shown in Tab. 2.

4.4. Motion Compensation and Error Correction

We implement the CFN [1] in PyTorch and also adapt it to the single frequency case by reducing the input dimension to one. For the other approaches DeepToF [8], E2E [12] and RADU [11] we use the TensorFlow2 implementations by Schelling *et al*. [11]. We train all networks using the respective hyperparameters reported by Schelling *et al*. for their CB-Dataset.

	SF 1T	SF 2T	MF 1T	MF 2T	MF 4T
MOM	0.002	0.002	0.002	0.002	0.002
FFN	0.067	0.025	0.230	0.107	0.047
PWC	0.092	0.054	0.342	0.154	0.062
RAFT	0.342	0.110	1.275	0.578	0.228

Table 2. Inference times in *s* of the OF backbone networks, averaged over the test set, on a GTX 1080 GPU.

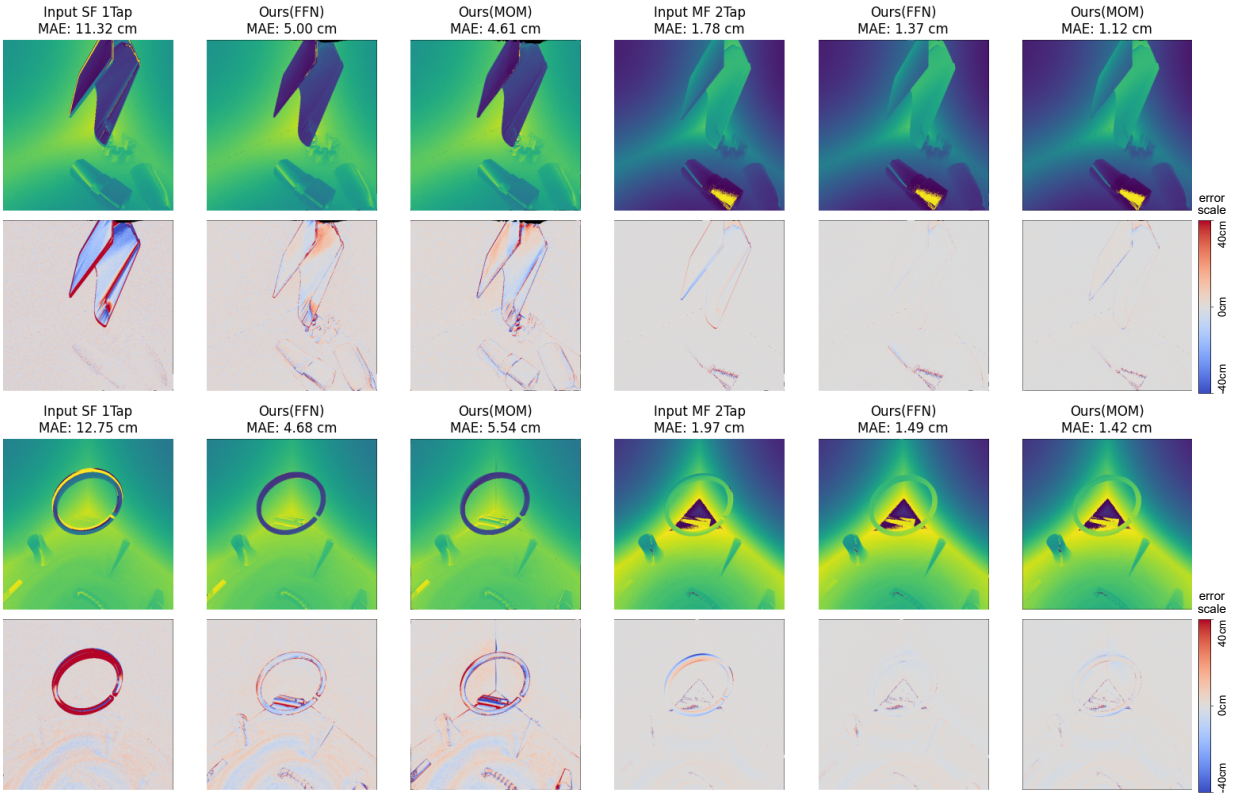

Figure 4. Results of our method for single frequency single-tap (SF 1Tap, left) and multi frequency two-tap (MF 2Tap, right). The scenes contain moving objects. First row shows ToF depths, second row shows error maps. Please note that the ToF depths are not phase unwrapped.

4.5. Ablation: Similarity Loss Function

We train the FFN network with the different similarity measures in the similarity loss function $\mathcal{L}_{sim}$ and optimize their weight λ_{sim} from {1e1, 1, 1e-1, 1e-2, 1e-3, 1e-4} for each measure. When training with input generated by Lindner's method in the multi frequency two-tap case, we reduce the network input dimension to one. The other hyperparameters, including the weights of the other losses, are set as in the main experiments. The results in the main paper were achieved with the following weights:
SF 1Tap: L1: 1e-2, L2: 1e-1, Cost: 1e-2, Cosine: 1e-3
MF 2Tap: L1: 1e-3, L2: 1e-2, Cost: 1, Cosine: 1e-3

5. Qualitative Results

Results of our method using the FFN and the MOM network can be seen in Fig. 4, 5, 6 and 7. The figures show one frame per scene from the test set. To cover both single and multi frequency and single and multi-tap the two cases single frequency single tap and multi frequency two tap are shown.

In Fig. 8 we show additional results for the combined correction of motion artifacts and Multi-Path-Interference (MPI) using the CFN as error correction network.

References

[1] Gianluca Agresti and Pietro Zanuttigh. Deep learning for multi-path error removal in ToF sensors. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.

[3] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[4] Qi Guo, Iuri Frosio, Orazio Gallo, Todd Zickler, and Jan Kautz. Tackling 3D ToF artifacts through learning and the FLAT dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 368–383, 2018.

[5] Rico Jonschkowski, Austin Stone, Jonathan T Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova. What matters in unsupervised optical flow. *arXiv preprint arXiv:2006.04902*, 2020.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Lingtong Kong, Chunhua Shen, and Jie Yang. FastFlowNet: A lightweight network for fast optical flow estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10310–10316. IEEE, 2021.

[8] Julio Marco, Quercus Hernandez, Adolfo Munoz, Yue Dong, Adrian Jarabo, Min H Kim, Xin Tong, and Diego Gutierrez. DeepToF: off-the-shelf real-time correction of multipath interference in time-of-flight imaging. *ACM Transactions on Graphics (ToG)*, 36(6):1–12, 2017.

[9] Henrique Morimitsu. Pytorch lightning optical flow. `https://github.com/hmorimitsu/ptlflow`, 2021.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] Michael Schelling, Pedro Hermosilla, and Timo Ropinski. RADU: Ray-aligned depth update convolutions for ToF data denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 671–680, 2022.

[12] Shuochen Su, Felix Heide, Gordon Wetzstein, and Wolfgang Heidrich. Deep end-to-end time-of-flight imaging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6383–6392, 2018.

[13] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[14] Z. Teed and J. Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *Proc. of ECCV*, 2020.

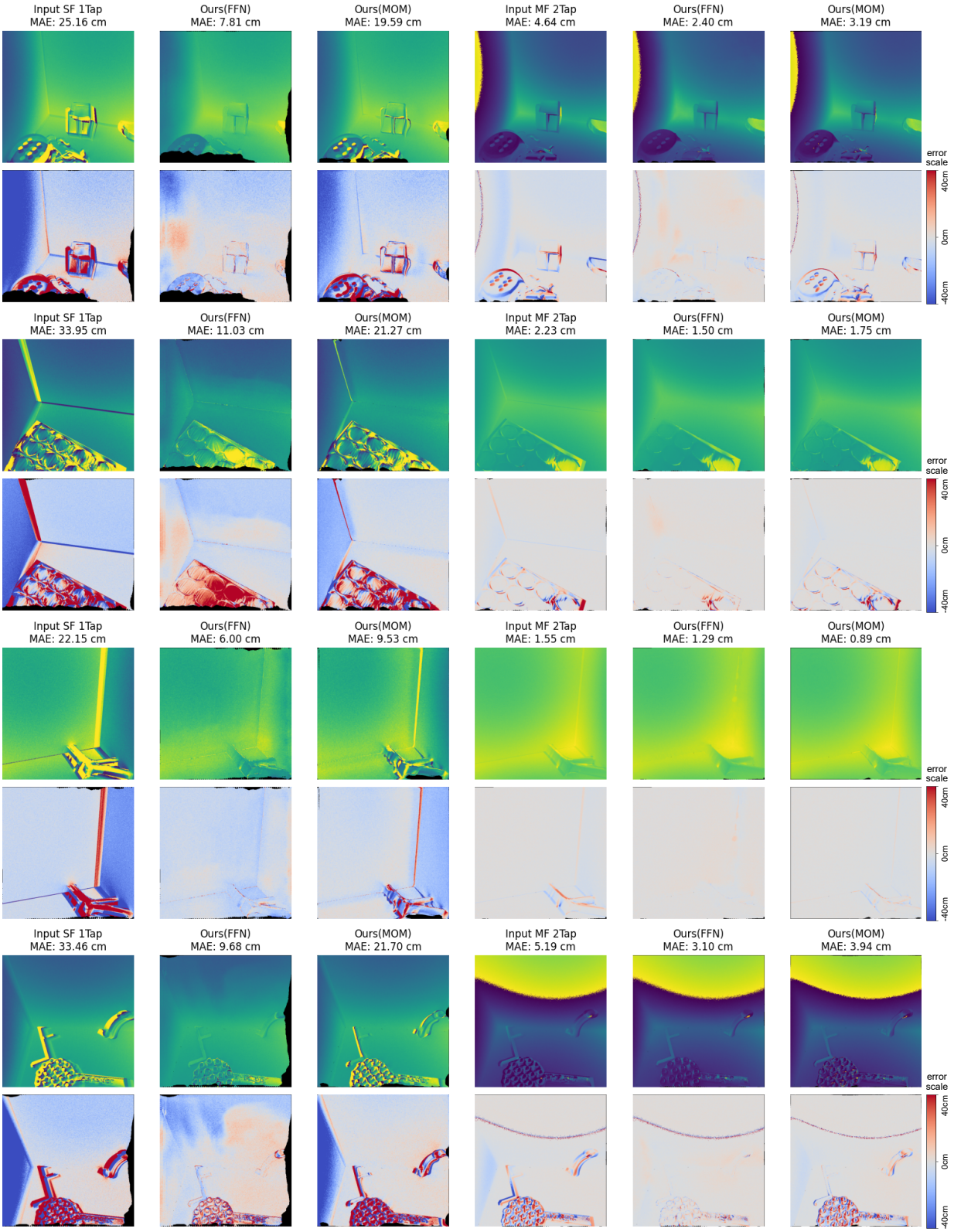

Figure 5. Results of our method for single frequency single-tap (SF 1Tap, left) and multi frequency two-tap (MF 2Tap, right). First row shows ToF depths, second row shows error maps. ToF depths are not phase unwrapped.

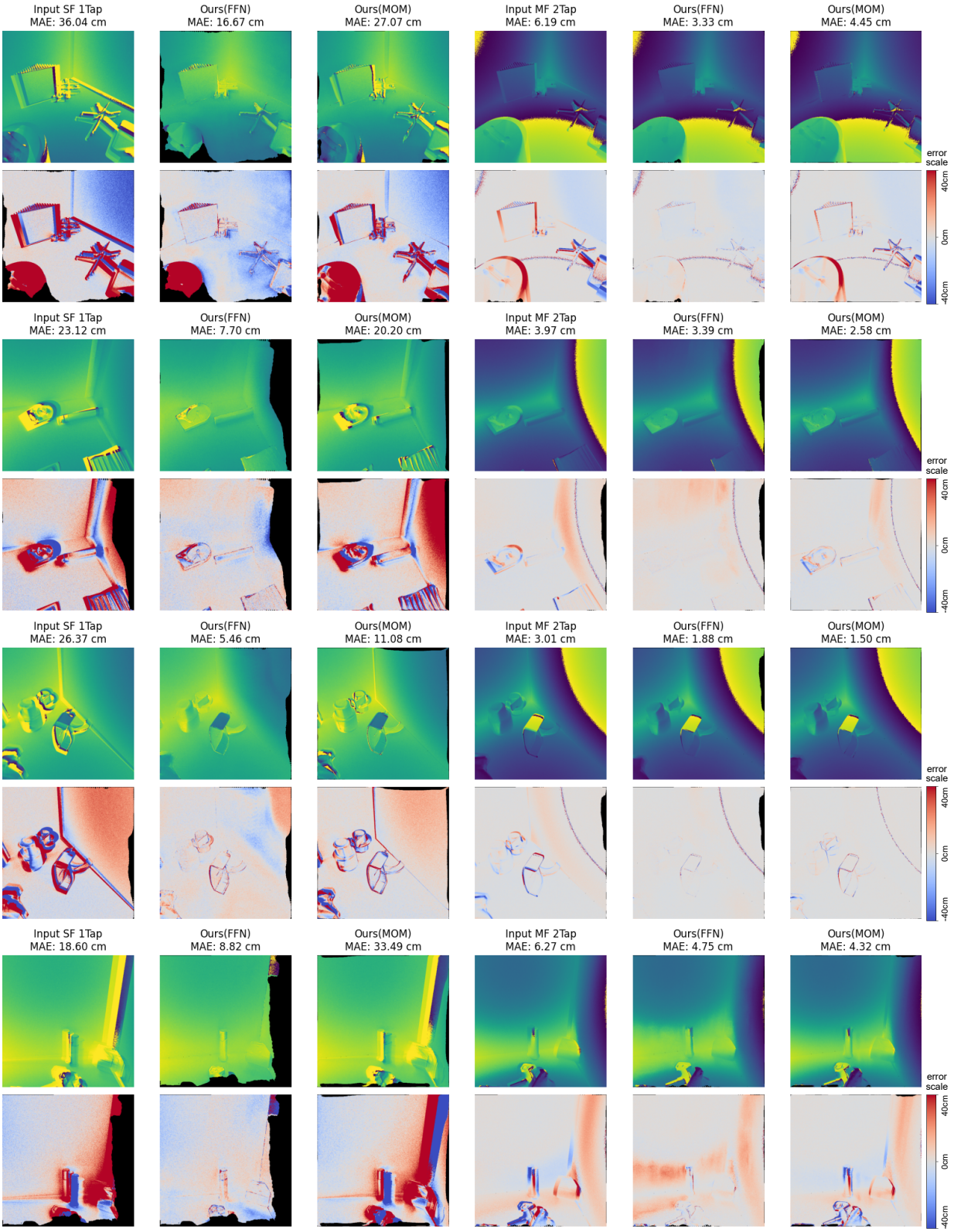

Figure 6. Results of our method for single frequency single-tap (SF 1Tap, left) and multi frequency two-tap (MF 2Tap, right). First row shows ToF depths, second row shows error maps. Please note that the ToF depths are not phase unwrapped.

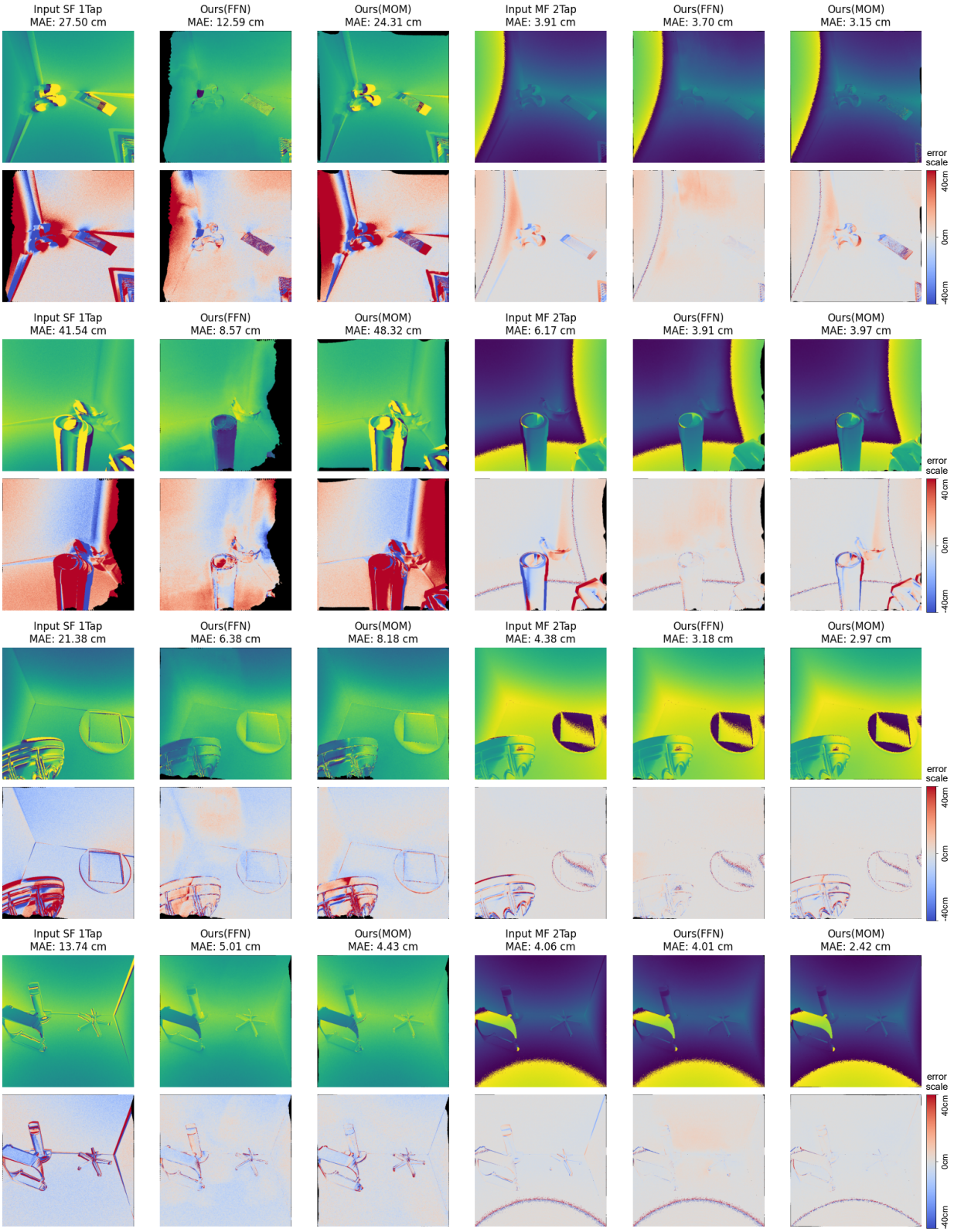

Figure 7. Results of our method for single frequency single-tap (SF 1Tap, left) and multi frequency two-tap (MF 2Tap). First row shows ToF depths, second row shows error maps. Please note that the ToF depths are not phase unwrapped.

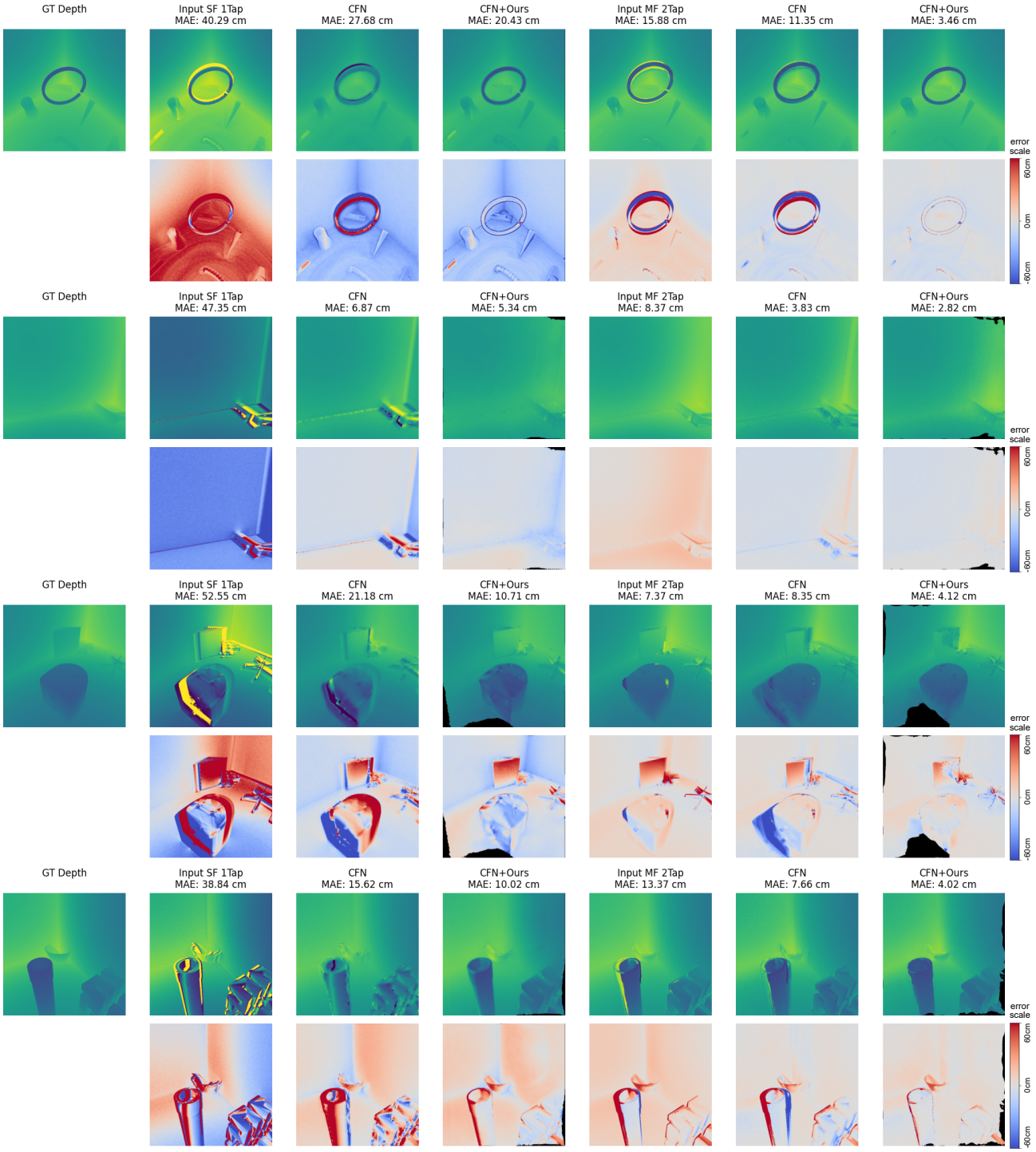

Figure 8. Results of combined motion and MPI correction using the CFN network. First row shows depths, second row shows error maps. Input in the MF 2Tap case is shown at highest frequency. First scene contains a moving object. Please note that CFN receives input from all frequencies, which can result in additional motion artifacts in the prediction compared to the high frequency input ToF depth.