Self-Supervised Multi-Frame Monocular Scene Flow
– Supplementary Material –

Junhwa Hur[1] Stefan Roth[1,2]
[1]Department of Computer Science, TU Darmstadt [2] hessian.AI

In this supplementary material, we introduce our 3D scene flow color coding scheme and provide details on our refined backbone architecture, self-supervised loss function, implementation, and computational cost. Then, we demonstrate additional results on temporal consistency, qualitative comparisons with the direct two-frame baseline (Self-Mono-SF [23]), and more experimental results for the generalization to other datasets. Lastly, we provide preliminary results of our model trained on a vast amount of unlabeled web videos in a self-supervised manner. We discuss the results as well as a current limitation of our method.

A. Scene Flow Color Coding

For visualizing 3D scene flow in 2D image coordinates, we use the CIE-LAB color space, as visualized in Fig. 12.

B. Refined Backbone Architecture

We provide a more in-depth analysis of our refined backbone two-frame architecture introduced in Sec. 3.1 of the main paper. We first present a simple empirical study of key findings from [28] and discuss which key factors can be carried over to monocular scene flow estimation. Afterward, we demonstrate an accuracy analysis and the improved training stability of our refined architecture by discarding the context network and splitting the decoder.

Empirical study on key findings from [28]. Jonschkowski *et al.* [28] provide a systematical analysis of the key design factors for highly accurate self-supervised optical flow. We conduct an empirical study on which of their key findings are beneficial in the context of monocular 3D scene flow. We report results on *KITTI Scene Flow Training* [43, 44] using the scene flow metrics (*cf*. Sec. 4.2 in the main paper).

Table 8 provides our empirical study on adopting each key factor on top of our baseline and reports the scene flow accuracy. We follow the training setup from [23]. We first apply cost volume normalization (CV. Norm.) on the model from [23] without the context network (Cont. Net.).[1] Cost volume normalization clearly improves the accuracy on all metrics, up to 4.4% (relative improvement) in terms of the scene flow accuracy. We choose this model as the baseline and conduct further empirical study on top of it.

[1]We use the model without the context network for more stable training, see main paper.

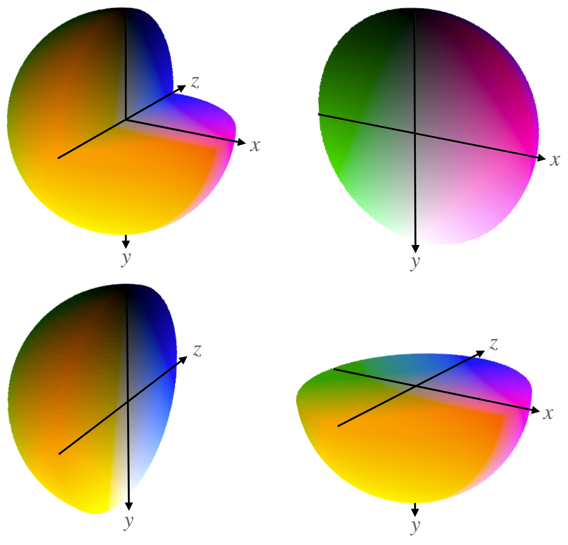
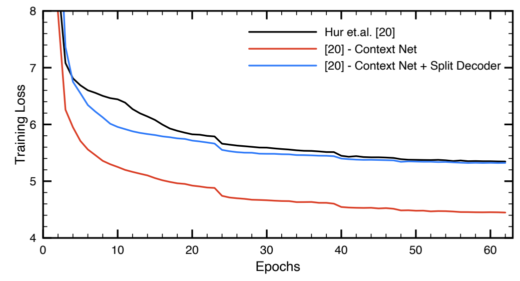


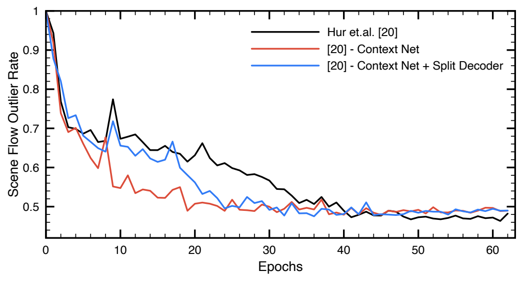
Figure 12. **3D scene flow color coding scheme using the CIE-LAB color space:** Each figure shows a sliced sphere along each plane for ease of visualization.

Model	D1-all	D2-all	Fl-all	SF-all
[23] − Cont. Net.	34.24	37.32	25.06	50.49
[23] − Cont. Net. + CV. Norm. (**Baseline**)	**31.91**	**35.31**	**24.80**	**48.29**
Applying each row on top of the baseline above:				
Census loss	32.52	34.54	21.79	45.61
Using one less pyramid level	33.68	35.03	23.98	47.77
Data distillation	34.62	35.74	24.09	48.11
Using 640 × 640 resolution	33.12	34.59	22.43	48.28
Level dropout	(*not converged*)			

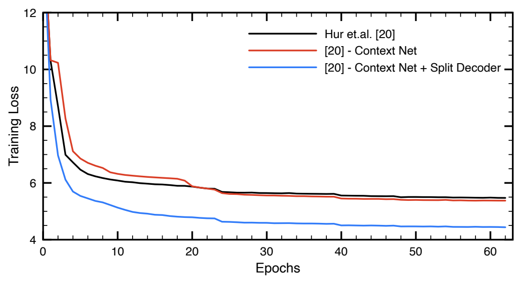
Table 8. **An empirical study of the key findings from [28]**: We take [23] after discarding the context network (Cont. Net.) and applying the cost volume normalization (CV. Norm.) as the baseline network. Then we apply each key factor to the baseline and compare the scene flow accuracy. Numbers colored in blue outperform the baseline accuracy.



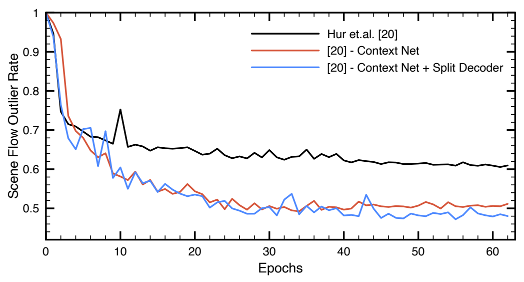
(a) Training loss, original train split [23]



(b) Scene flow outlier rate, original train split [23]



(c) Training loss, our train split



(d) Scene flow outlier rate, our train split

Figure 13. **An analysis of training stability**: We compare the training loss and the accuracy of three models (*i.e*., direct baseline [23], [23] without context network, and [23] without context network and with split decoder), training on two different splits (*i.e*. the original training split from [23] and our multi-frame train split). Discarding the context network offers more stable, faster convergence of the accuracy on both splits. Further applying the split decoder improves the accuracy.

Model	D1-all	D2-all	Fl-all	SF-all
Forward-backward consistency	27.99	30.51	20.35	40.80
Using disocclusion	**27.33**	**30.44**	**18.92**	**39.82**

Table 9. **Comparison of different occlusion estimation strategies**: Using disocclusion produces more accurate scene flow estimates than using a forward-backward consistency check.

We find the census loss and using one less pyramid level to be beneficial for the monocular scene flow setup as well. On the other hand, using data distillation or a 640×640 pixel resolution only brings marginal improvements, but requires a much longer training time; thus we decide not to apply them here. We observe that using a square resolution improves the optical flow accuracy but hurts the disparity accuracy; this offsets the improvement in the end. When using level dropout, which randomly skips some pyramid levels when training, training unfortunately did not converge.

In Table 9, we also compare different occlusion estimation techniques, specifically disocclusion detection and the forward-backward consistency check as described in [28]. We use our final model with multi-frame estimation for the study. Unlike the conclusion from [28], we find that using disocclusion information for occlusion detection produces more accurate scene flow than using a forward-backward consistency check.

Improved training stability. As discussed in Sec. 3.1 of the main paper, discarding the context network and splitting the decoder improve the training stability with faster convergence. Fig. 13 plots the training loss and the scene flow outlier rate of the direct baseline [23], the baseline without the context network, and additionally applying our split decoder design. We demonstrate the results on using two different train splits, the original split from [23] and our multi-frame train split (see Sec. 4.1).

We make the following main observations: *(i)* The direct baseline [23] shows a significant accuracy drop when using our train split. *(ii)* Discarding the context network resolves the issue and offers more stable, faster convergence regarding the accuracy on both train splits. *(iii)* After applying our split decoder design, the model further improves the accuracy, showing more stable and faster convergence regarding the training loss on both train splits.

Note that a lower training loss does not always directly translate to better accuracy, since the model optimizes the self-supervised proxy loss. For this reason we also plot the scene flow outlier rate (on KITTI Scene Flow Training).

C. Self-Supervised Loss

We provide further details on the self-supervised proxy loss introduced in Sec. 3.3 in the main paper. The weighting constant λ_{sf} in Eq. (2) in the main paper is calculated at every iteration step to make the scene flow loss L_{sf} equal to the disparity loss L_{d}, which previous work [23] empirically found to be better than using a fixed constant.

Disparity loss. Following Godard *et al*. [12, 13], we use the right view of a stereo image pair for the guidance of disparity estimation at training time; the second view is not used at test time. The disparity loss consists of a photometric loss $L_{\text{d,ph}}$ and a smoothness loss $L_{\text{d,sm}}$ with regularization constant $\lambda_{\text{d,sm}} = 0.1$,

$$L_{\text{d}} = L_{\text{d,ph}} + \lambda_{\text{d,sm}} L_{\text{d,sm}} \tag{4a}$$

with

$$L_{\text{d,ph}} = \rho_{\text{census}}\big(\mathbf{I}_t, \tilde{\mathbf{I}}_t^{\text{disp}}, \mathbf{O}_t^{\text{disp}}\big). \tag{4b}$$

The photometric loss $L_{\text{d,ph}}$ in Eq. (4b) penalizes the photometric difference between the left view $\mathbf{I}_t$ and the synthesized left view $\tilde{\mathbf{I}}_t^{\text{disp}}$, obtained from the output disparity $\mathbf{d}_t$ and the given right view $\mathbf{I}_t^{\text{r}}$ via backward warping [81]. To calculate the photometric difference, we use our new occlusion-aware census loss ρ_{census} in Eq. (3a) in the main paper. As in [23], we obtain the disparity occlusion mask $\mathbf{O}_t^{\text{disp}}$ from forward-warping the right disparity map by inputting the right view $\mathbf{I}_t^{\text{r}}$ into the network.

We use an edge-aware 2nd-order smoothness term [23, 28] to define the disparity smoothness $L_{\text{d,sm}}$ in Eq. (4a).

$$L_{\text{d,sm}} = \frac{1}{N}\sum_{\mathbf{p}}\sum_{i\in\{x,y\}} \left|\nabla_i^2 \mathbf{d}_t(\mathbf{p})\right| \cdot e^{-\beta\|\nabla_i \mathbf{I}_t(\mathbf{p})\|_1}, \tag{5}$$

with $\beta = 150$, divided by the number of pixels N [28].

Scene flow loss. The scene flow loss consists of three terms [23]: a photometric loss $L_{\text{sf,ph}}$, a 3D point reconstruction loss $L_{\text{sf,pt}}$, and a scene flow smoothness loss $L_{\text{sf,sm}}$,

$$L_{\text{sf}} = L_{\text{sf,ph}} + \lambda_{\text{sf,pt}} L_{\text{sf,pt}} + \lambda_{\text{sf,sm}} L_{\text{sf,sm}}, \tag{6a}$$

with

$$L_{\text{sf,ph}} = \rho_{\text{census}}\big(\mathbf{I}_t, \tilde{\mathbf{I}}_t^{\text{sf}}, \mathbf{O}_t^{\text{sf}}\big), \tag{6b}$$

and regularization weights $\lambda_{\text{sf,pt}} = 0.2$, $\lambda_{\text{sf,sm}} = 1000$.

The scene flow photometric loss in Eq. (6b) penalizes the photometric difference between the reference image $\mathbf{I}_t$ and the synthesized reference image $\tilde{\mathbf{I}}_t^{\text{sf}}$, obtained from the camera intrinsics $\mathbf{K}$, estimated disparity $\mathbf{d}_t$, and the scene flow $\mathbf{s}_t^{\text{f}}$ (*cf*. Fig. 4a in [23]). Here we also apply our novel occlusion-aware census transform ρ_{census} from Eq. (3a). The scene flow occlusion mask $\mathbf{O}_t^{\text{sf}}$ is obtained by the disocclusion from the backward scene flow $\mathbf{s}_{t+1}^{\text{b}}$.

The 3D reconstruction loss $L_{\text{sf,pt}}$ in Eq. (6a) penalizes the Euclidean distance between the corresponding 3D points, $\mathbf{P}'_t$ and $\mathbf{P}'_{t+1}$, however only for visible pixels:

$$L_{\text{sf,pt}} = \frac{1}{\sum_{\mathbf{q}} \mathbf{O}_t^{\text{sf}}(\mathbf{q})} \sum_{\mathbf{p}} \frac{\mathbf{O}_t^{\text{sf}}(\mathbf{p}) \cdot \left\|\mathbf{P}'_t - \mathbf{P}'_{t+1}\right\|_2}{\left\|\mathbf{P}_t\right\|_2} \tag{7a}$$

with

$$\mathbf{P}'_t = \hat{\mathbf{d}}_t(\mathbf{p}) \cdot \mathbf{K}^{-1}\mathbf{p} + \mathbf{s}_t^{\text{f}}(\mathbf{p}) \tag{7b}$$

$$\mathbf{P}'_{t+1} = \hat{\mathbf{d}}_{t+1}(\mathbf{p}') \cdot \mathbf{K}^{-1}\mathbf{p}', \tag{7c}$$

and

$$\mathbf{p}' = \mathbf{K}\Big(\hat{\mathbf{d}}_t(\mathbf{p}) \cdot \mathbf{K}^{-1}\mathbf{p} + \mathbf{s}_t^{\text{f}}(\mathbf{p})\Big), \tag{7d}$$

where $\mathbf{p}'$ is the corresponding pixel of $\mathbf{p}$ given the scene flow and disparity estimate. $\hat{\mathbf{d}}_t$ and $\hat{\mathbf{d}}_{t+1}$ are the depth maps at time t and $t+1$ respectively. The depth $\hat{\mathbf{d}}$ is trivally converted from the disparity estimates given the camera focal length f_{focal} and the baseline of the stereo rig b, specifically $\hat{\mathbf{d}} = f_{\text{focal}} \cdot b / \mathbf{d}$. Here, we assume that the camera focal length and the stereo baseline is given so that the network outputs disparity (or depth) on a certain, fixed scale.

The loss is normalized by the 3D distance of each point $\mathbf{P}_t$ to the camera to penalize the relative distance to camera.

The same edge-aware smoothness loss is applied to 3D scene flow, yielding $L_{\text{sf,sm}}$ in Eq. (6a), also normalized by its 3D distance to camera:

$$L_{\text{sf,sm}} = \frac{1}{N}\sum_{\mathbf{p}} \frac{\sum_{i\in\{x,y\}} \left|\nabla_i^2 \mathbf{s}_t^{\text{f}}(\mathbf{p})\right| \cdot e^{-\beta\|\nabla_i \mathbf{I}_t(\mathbf{p})\|_1}}{\left\|\mathbf{P}_t\right\|_2}, \tag{8}$$

with $\beta = 150$ and N being the number of pixels.

D. Implementation Details

As briefly discussed in Sec. 4.1 of the main paper, we use the augmentation scheme and training configuration suggested by [23]. The geometric augmentation consists of horizontal flips (with 50% probability), random scaling, cropping, and resizing into 256×832 pixels. Then, a photometric augmentation is applied with 50% probability, consisting of gamma adjustment, random brightness and color changes. The augmentation parameters are uniformly sampled from the ranges given in Table 10. We use the same augmentation parameters for all consecutive frames included in the same mini-batch.

For training, we use the Adam optimizer [82] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We do not apply weight decay because we found that it harms the accuracy. The learning rate schedule from [23] is used. That is, for self-supervised training for 400k iterations, the initial learning rate starts at 2×10^{-4}, being halved at 150k, 250k, 300k, and 350k iteration steps. Afterwards, for semi-supervised fine-tuning for 45k iterations, the learning rate starts from 4×10^{-5}, being halved at 10k, 20k, 30k, 35k, and 40k iteration steps.

Augmentation type	Sampling range
Random scaling	$[0.93, 1.0]$
Random cropping	$[-3.5\%, 3.5\%]$
Gamma adjustment	$[0.8, 1.2]$
Brightness change multiplication factor	$[0.5, 2.0]$
Color change multiplication factor	$[0.8, 1.2]$

Table 10. **Augmentation parameters**: The augmentation parameters are uniformly sampled from the given sampling ranges.

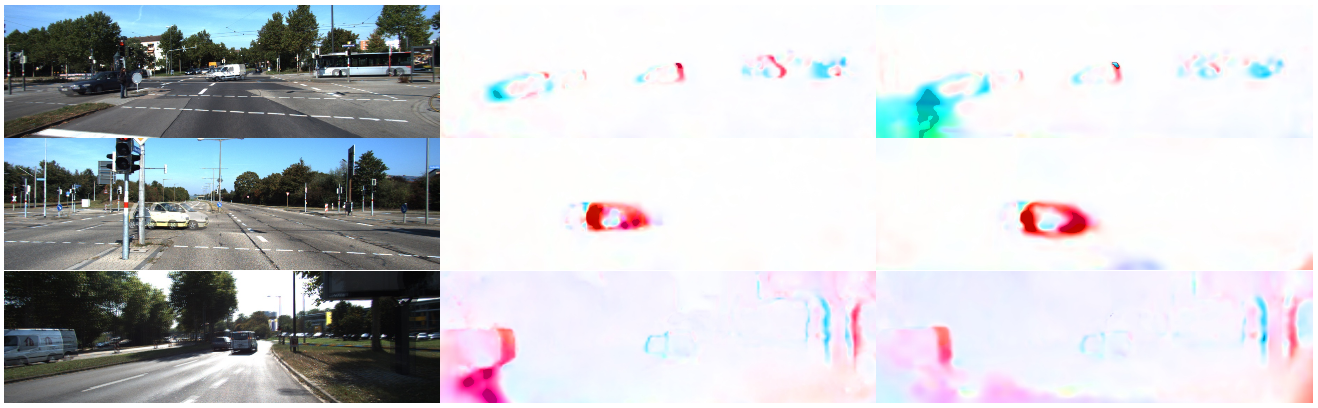


Figure 14. **Qualitative comparison of temporal consistency**: Each scene shows *(a)* overlayed input images and scene flow difference maps of *(b)* our method, and *(c)* the direct two-frame baseline of [23], visualized using optical flow color coding. Our method provides more temporally consistent estimates near moving objects and out-of-bound regions.

Method	Self-Mono-SF [23]	**Multi-Mono-SF (ours)**
AEPE	0.0969	**0.0911**

Table 11. **Temporal consistency evaluation**: Our method shows lower average end-point error (AEPE) between two temporally consecutive estimates.

E. Computational Cost and Training Time

Our model takes 153.1G FLOPS of computation per frame pair, with a model size of 7.537M parameters. It requires only one GPU to train, consumes only 4.89G GPU memory, and trains for 4.5 days (on a single NVIDIA GTX 1080 Ti GPU).

F. Temporal Consistency

We provide an additional analysis of the temporal consistency, continuing from Sec. 4.4 in the main paper. Fig. 14 visualizes additional comparisons of the scene flow difference map, comparing to the direct two-frame baseline [23]. Our model produces visibly more temporally consistent scene flow, especially near moving objects and out-of-bound regions.

In Table 11, we also quantitatively evaluate the temporal consistency on *KITTI Scene Flow Training* by calculating the average Euclidean distance of two corresponding scene flow vectors between the two temporally consecutive estimates. The corresponding scene flow is found using the provided ground truth labels. While this is not an ideal way for measuring temporal consistency, it shows how much each corresponding scene flow vector changes over time, assuming constant velocity. Comparing to the direct two-frame baseline [23], our method gives lower AEPE between two temporally corresponding scene flow vectors, which indicates more temporally consistent estimates.

Dataset	Optical flow EPE	Scene flow EPE
Driving	3.6613	0.8845
Monkaa	12.8881	4.2982

Table 12. **Scene flow accuracy on Driving and Monkaa datasets [40] using the End-Point Error (EPE) metric**: The accuracy of our model is generally low in the synthetic domain but is better on Driving than on Monkaa.

G. Qualitative Comparison

In Fig. 15, we provide additional qualitative comparisons with the direct two-frame baseline [23] as in Sec. 4.5 in the main paper. Supporting the same conclusion as in the main paper, our approach produces more accurate 3D scene flow on out-of-bound regions, foreground objects, and planar road surfaces.

H. Generalization to Other Datasets

Continuing from Sec. 4.5 of the main paper, we provide more qualitative results on the nuScenes [8], DAVIS [49], Driving and Monkaa [40] datasets. Fig. 16 provides both successful cases and failure cases on those three datasets, respectively. Our model, trained only on the KITTI dataset, generalizes well to the nuScenes dataset [8], which is reasonably close in domain (*i.e.*, driving scenes). However, there exist some failure cases with inaccurate depth estimation as well as occasional artifacts on the image boundary. On the DAVIS [49] dataset, our model generalizes surprisingly well to completely unseen domains, yet depth estimation on unseen objects (*e.g.*, horse, cat) can sometimes fail.

In the synthetic domain (Driving and Monkaa [40] datasets), however, our model demonstrates less accurate results, as can be expected. Typical failure cases are again inaccurate depth estimation on completely unseen synthetic objects or reflective road surfaces. In Table 12, we eval-

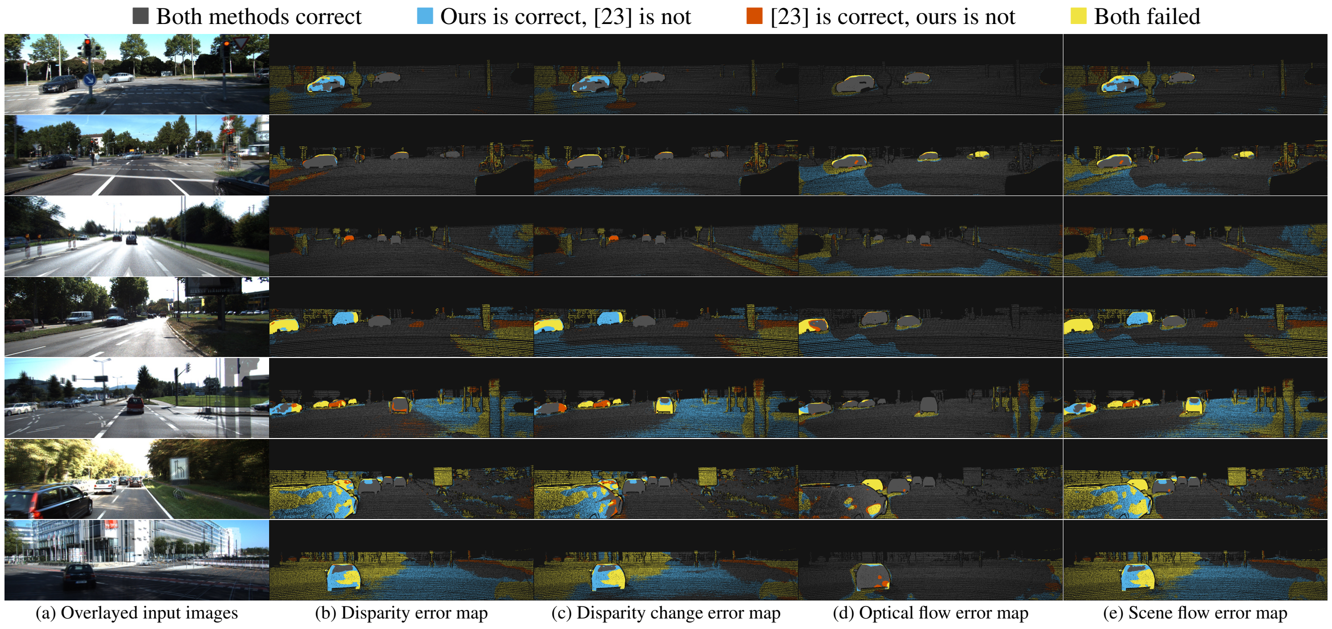

Both methods correct
Ours is correct, [23] is not
[23] is correct, ours is not
Both failed
(a) Overlayed input images
(b) Disparity error map
(c) Disparity change error map
(d) Optical flow error map
(e) Scene flow error map

Figure 15. **Qualitative comparison with the direct baseline of [23]**: Each scene shows *(a)* overlayed input images and error maps for *(b)* disparity, *(c)* disparity change, *(d)* optical flow, and *(e)* scene flow. Please refer to the color code above the images.

uate the scene flow accuracy of our model on the Driving and Monkaa [40] datasets, using the End-Point-Error (EPE) metric. Though the accuracy is quite low in general, the accuracy on Driving is much better than that on Monkaa as can be expected.

These overall results suggest that the accuracy of our self-supervised model depends on the training domain as well as the presence of target objects in the training dataset. From this observation, we can conclude that better generalization requires to train the model on a dataset with both diverse domains and objects.

I. Self-Supervised Learning in the Wild

Through self-supervised learning, our method can in principle leverage vast amounts of unlabeled stereo web videos. However unlike training on a single, calibrated dataset (*e.g.*, KITTI), this comes with several new technical challenges. Each stereo video is captured with different camera intrinsics and stereo configurations, whose values are even unknown. Without knowing them, the self-supervised loss in Eq. (2) in the main paper cannot be directly applied because it assumes a fixed (or given) focal length and stereo baseline. We provide preliminary experiments to assess the feasibility of this scenario.

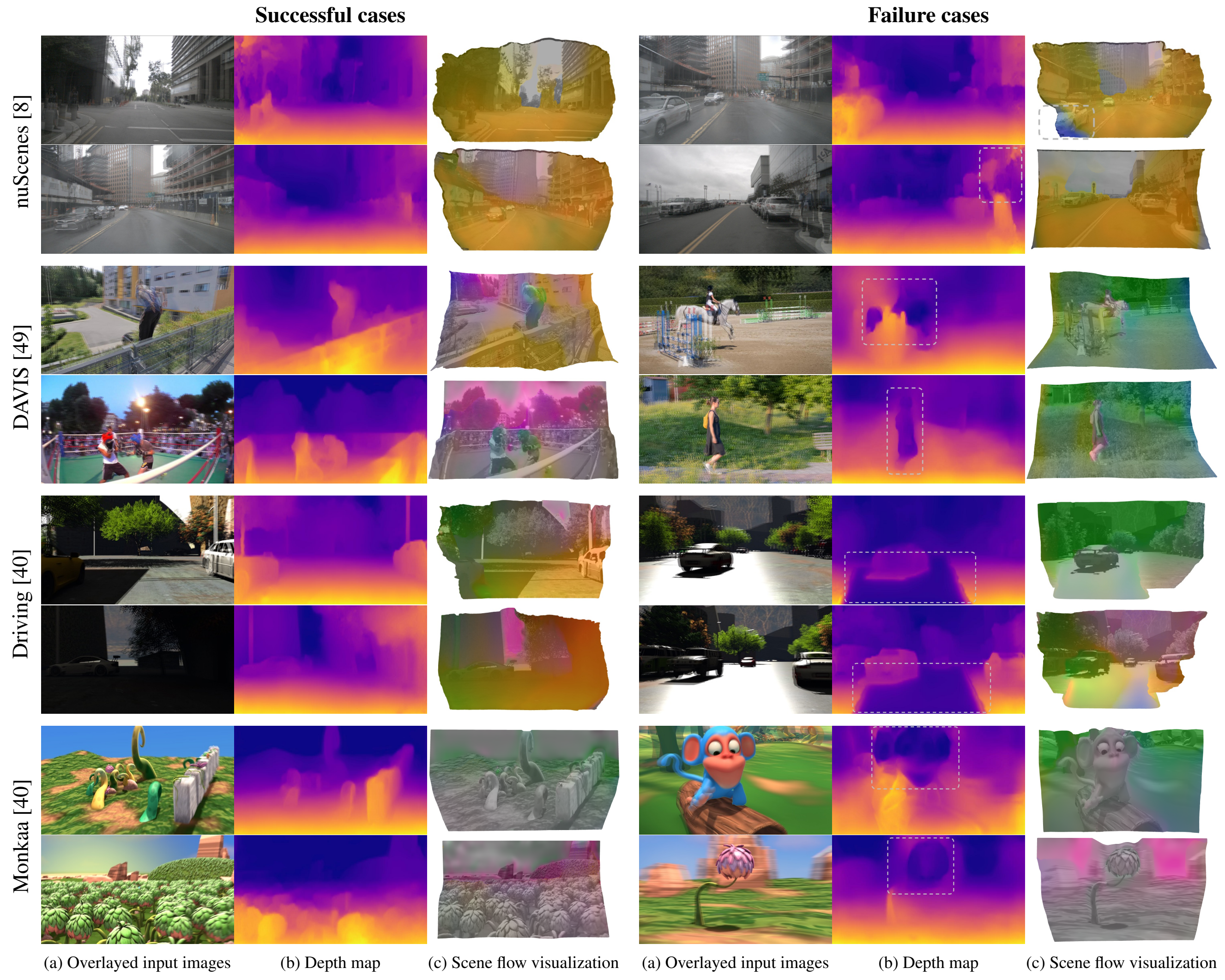
To train the network despite these unknowns, we first assume all videos share the same focal length. Then, we normalize the output disparity (say, d_{norm}) to be in a fixed, normalized scale and use it for the scene flow loss. For the disparity loss, we further linearly transform the disparity d_{norm} to match the actual disparity scale of each given stereo input:

$$d_{\text{actual}} = a_{\text{scale}} \cdot d_{\text{norm}} + b_{\text{scale}}. \tag{9}$$

To obtain the coefficients a_{scale} and b_{scale}, we estimate optical flow between the stereo pair using our network, take the horizontal flow as pseudo disparity d_{pseudo}, and use the least squares between the pseudo disparity d_{pseudo} and the normalized disparity d_{norm}. Though our network now outputs disparity and scene flow on a normalized scale, it is still able to estimate optical flow (by projecting scene flow to image coordinates) on the correct scale due to being supervised by the 2D view-synthesis proxy loss.

For our preliminary experiments, we use the WSVD dataset [83], which is a collection of stereo videos from YouTube, for training and test on the DAVIS [49] dataset.

Training dataset preparation. When training on such diverse data collected on the web, it is important to make sure that the dataset is free of outliers. For preparing the training data, we carefully pre-process the WVSD dataset by first discarding videos with low resolution, poor image quality, texts, or watermarks. We further discard videos having vertical disparity, lens distortion, and narrow stereo baselines for better stereo supervision. Then, we check every frame and remove black-colored edges on the image boundaries, if applicable. Also, we find that many videos contain static scenes; thus we sample every 4th frame and 2nd sequence for having more dynamic motion in the training sequences. This pre-processing step, in the end, results in 58k training images from about 1.5M raw frames. Given the pre-trained model on KITTI, we further train the model on this curated dataset for 300k iteration steps.



Successful cases
Failure cases
nuScenes [8]
DAVIS [49]
Driving [40]
Monkaa [40]
(a) Overlayed input images
(b) Depth map
(c) Scene flow visualization
(a) Overlayed input images
(b) Depth map
(c) Scene flow visualization

Figure 16. **Generalization to nuScenes, DAVIS, Driving, and Monkaa datasets**: Each scene shows *(a)* overlayed input images, *(b)* depth, and *(c)* 3D scene flow visualization. The *left side* demonstrates good generalization of our model to nuScenes, DAVIS, Driving, and Monkaa datasets. Nonetheless, failure cases do exist, provided on the *right side* and highlighted with dashed gray squares. A typical failure mode is inaccurate depth estimation of foreground objects that are not seen in the training set as well as reflective road surfaces.

Result and discussion. Fig. 17 demonstrates the test result on the DAVIS [49] dataset. Comparing to our KITTI-trained model (*cf*. Fig. 16), our model trained on the WSVD dataset [83] is able to correctly estimate depth on diverse scenes from the DAVIS [49] dataset. This also confirms our observation from the generalization analysis in Appendix H that better generalization can be achieved by training on a dataset with diverse scenes and objects.

However, our model unfortunately fails to correctly estimate scene flow: the network outputs z components of the scene flow that are nearly zero and only estimates x and y components (refer the scene flow color coding in Fig. 12). We also observe that the 3D reconstruction loss $L_{\text{sf,pt}}$ in Eq. (7a) does not converge at all. We conjecture that the failure comes from the issue that the model does not estimate scale-consistent depth but instead normalized depth for each frame, which makes it difficult to determine the correct z component of the scene flow for each sequence. This connects to a current limitation that our approach requires a fixed (or given) focal length and a stereo baseline for training. However, we expect that this can be overcome once the network is able to output scale-consistent depth across sequences or videos while being trained on videos with diverse camera settings. We leave this for future work.

References

[81] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NIPS*2015*, pages 2017–2025. 3

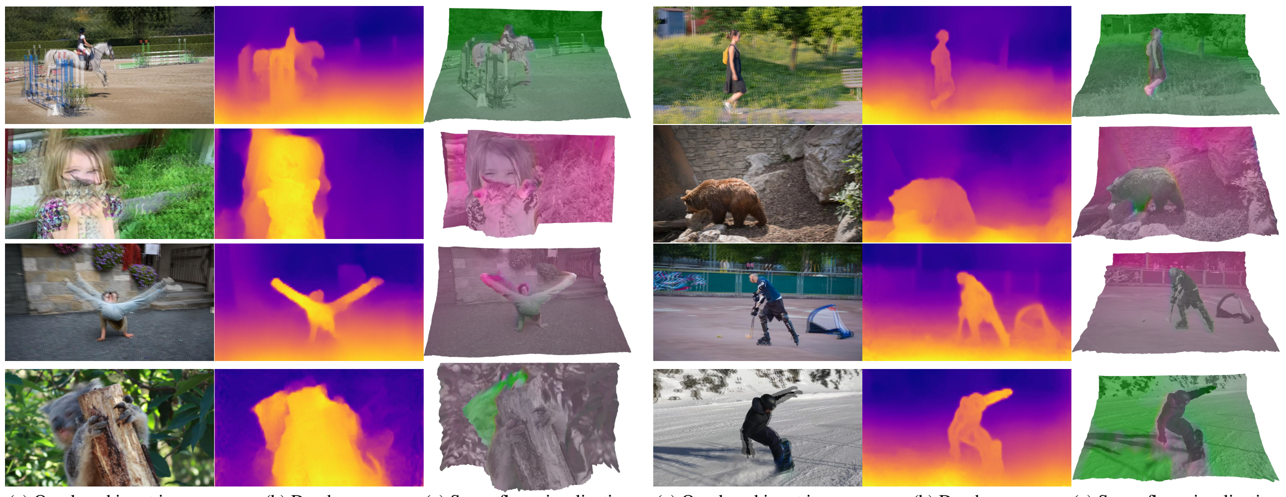

(a) Overlayed input images
(b) Depth map
(c) Scene flow visualization
(a) Overlayed input images
(b) Depth map
(c) Scene flow visualization

Figure 17. **Self-supervised learning in the wild and generalization to DAVIS dataset [49]**: Our model trained on the WSVD dataset [83] (a large amount of web videos) generalizes well to diverse scenes from the DAVIS dataset [49].

[82] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3

[83] Chaoyang Wang, Simon Lucey, Federico Perazzi, and Oliver Wang. Web stereo video supervision for depth prediction from dynamic scenes. In *3DV*, pages 348–357, 2019. 5, 6, 7