Eurographics Symposium on Rendering 2023
T. Ritschel and A. Weidlich
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 4

Interactive Control over Temporal Consistency while Stylizing Video Streams (Supplementary Material)

Sumit Shekhar[1*], Max Reimann[1*], Moritz Hilscher[1], Amir Semmo[1,2],
Jürgen Döllner[1], and Matthias Trapp[1]

[1]Hasso Plattner Institute for Digital Engineering, University of Potsdam, Germany
[2]Digital Masterpieces GmbH, Germany
("*" denotes equal contribution)

Some details had to be omitted from the main paper due to the page limit; we present those details here. In the following, we report on ablation experiments in Sec. 1.1, which were carried out to determine the best performing fast optical-flow network, and also expand on quantization and pruning which were employed for our mobile-optimized network. In Sec. 1.2 we expand on training and implementation details. In Sec. 2 we provide detailed numbers for the warping error. In Sec. 3 we compare our method subjectively against Shekhar *et al.* [SST*19] and Thimonier *et al.* [TDKP21] through a user study. Finally, in Sec. 4, we present further visual results of our optical-flow network.

1. Fast Optical Flow Network

1.1. Ablation Study

To analyze our optimization steps, we compare different variants of our Convolutional Neural Network (CNN). All variants are trained on the full dataset schedule unless stated otherwise. We make use of Sintel Final Train dataset [BWSB12] as a benchmark and measure accuracy (in terms of Endpoint Error (EPE)), number of parameters, and run-time for different variants. Due to different desktop and mobile Graphics Processing Unit (GPU) hardware, the run-time performance can vary between platforms, thus we measure them separately.

Table 1: Comparison of DenseNet [HLvdMW17] and light [LZH*20] connections on a desktop system and mobile devices on Sintel Final Train.

Variant	EPE (↓)	Params M (↓)	Desktop FPS (↑)	iPad Air FPS (↑)	iPad Pro FPS (↑)
dense	2.507	9.36	29.97	1.53	2.80
<u>light</u>	<u>2.825</u>	<u>5.99</u>	<u>40.26</u>	<u>2.83</u>	<u>5.09</u>

DenseNet Connection Replacement. As a first architectural improvement, we replace DenseNet [HLvdMW17] connections in the flow estimators with light connections [LZH*20]. Replacing these results in a significant run-time improvement on both desktop systems and mobile devices, with a larger relative speed-up on mobile devices (Tab. 1). We conjecture that convolutions with a large number of channels (dense architecture uses up to 565 channels) might perform worse on mobile GPUs due to smaller memory and cache sizes. Thus, reducing these high channel counts results in a larger speed-up on mobile devices. The light connections result in a loss in accuracy (Tab. 1), but due to the significant run-time improvements, we find it a reasonable trade-off and use light connections in the following experiments and in our proposed mobile architecture.

Table 2: Ablation of different channel reduction on desktop system

Variant	EPE (↓)	Params M (↓)	Desktop FPS (↑)	iPad Air FPS (↑)	iPad Pro FPS (↑)
<u>5 estimators</u>	<u>2.825</u>	<u>5.99</u>	<u>40.26</u>	<u>2.83</u>	<u>5.09</u>
-25% channels	3.659	3.55	46.86	3.95	6.65
-50% channels	4.236	1.73	56.07	6.33	10.92

Channel Reduction. We reduce the number of channels throughout the CNN [HZC*17]. In this case, the loss in accuracy and achieved trade-off is not beneficial (Tab. 2). We hypothesize that channel reduction is potentially better for high-level Computer Vision (CV) tasks, where high-dimensional convolution features are mapped to very low-dimensional results [HZC*17]. Optical flow, however, requires pixel-precise predictions of continuous values (motion vectors) and thus requires a much higher spatial fidelity. Furthermore, we observe that the relative speed-up on mobile devices is again higher than on desktop systems which supports our previous belief that larger convolutions are more difficult for mobile GPUs with smaller memory and cache sizes.

Flow Estimators. We evaluate different configurations of separable convolutions for the five flow-estimator modules. Replacing all convolutions in the flow estimators with separable convolutions leads to a significant loss in accuracy (Tab. 3). The last two flow estimators operate on the highest pyramid resolutions and have the largest impact on run-time performance. Thus, loss of accuracy

Table 3: Ablation of different light flow-estimator configurations.

	Sintel Final Train		Number of parameters		Desktop run-time		iPad Air run-time		iPad Pro run-time	
Variant	EPE (↓)		M (↓)		FPS (↑)		FPS (↑)		FPS (↑)	
5 estimators, none separated	2.825		5.99		40.26		2.83		5.09	
5 estimators, all separated	3.813	+34.9%	2.66	-55.6%	43.49	+8.0%	4.52	+59.7%	7.76	+52.4%
5 estimators, last two separated	3.104	+9.8%	4.77	-20.3%	44.06	+9.4%	4.22	+49.1%	7.28	+43.0%
4 estimators, none separated	3.229	+14.3%	5.31	-11.3%	79.12	+96.5%	6.17	+118.0%	10.41	+104.5%
4 estimators, last separated	3.460	+22.4%	4.68	-21.8%	81.28	+101.7%	7.35	+159.7%	12.80	+151.4%

Table 4: Ablation of different refinement configurations.

	Sintel Final Train		Number of parameters		Desktop run-time		iPad Air run-time		iPad Pro run-time	
Variant	EPE (↓)		M (↓)		FPS (↑)		FPS (↑)		FPS (↑)	
4 estimators, default refinement	3.229		5.31		79.12		6.17		10.41	
4 estimators, no refinement	3.258	+0.9%	4.79	-9.8%	86.72	+9.6%	7.44	+20.5%	13.22	+27.0%
4 estimators, separated refinement	3.169	-1.8%	4.85	-8.6%	83.53	+5.5%	7.23	+17.1%	12.87	+23.6%
5 estimators, default refinement	2.825		5.99		40.26		2.83		5.06	
5 estimators, no refinement	3.248	+14.97%	5.47	-8.6%	51.80	+28.6%	4.22	+49.1%	7.31	+44.4%
5 estimators, separated refinement	2.922	+3.43%	5.53	-7.6%	47.43	+17.8%	3.47	+22.6%	6.00	+18.5%

Table 5: Ablation of different pruning amounts on desktop system.

	Sintel Final Train		Number of parameters		Desktop run-time		iPad Air run-time		iPad Pro run-time	
Variant	EPE (↓)		M (↓)		FPS (↑)					
4 estimators, separated refinement	3.169		4.85		79.12		7.23		12.87	
30% pruned	3.275	+3.3%	3.38	-30.3%	86.47	-9.3%	8.79	+21.5%	16.45	+27.8%
40% pruned	3.443	+8.6%	2.83	-41.6%	86.61	-9.4%	10.00	+38.3%	18.94	+47.1%
50% pruned	4.036	+27.3%	2.24	-53.8%	88.94	-12.4%	13.76	+90.3%	27.50	+113.6%
5 estimators	2.825		5.99		40.26		2.83		5.06	
30% pruned	2.900	+2.6%	4.06	-32.2%	46.79	-16.2%	3.89	+37.4%	6.95	+37.3%
40% pruned	3.022	+6.9%	3.35	-44.0%	50.68	-25.8%	4.63	+63.6%	8.17	+61.4%
50% pruned	3.106	+9.9%	3.01	-49.7%	51.51	-27.9%	4.86	+71.7%	8.47	+57.4%

Table 6: Ablation of different quantization options on mobile devices using our lite-flow CNN.

Quantization settings		CNN file size		Sintel Final Train		iPad Air run-time		iPad Pro run-time	
Number of bits	low-prec. acc.	MB (↓)		EPE (↓)		FPS (↑)		FPS (↑)	
32-bit		16.3		3.577		9.95		18.94	
32-bit	✓	16.3		3.584	+0.2%	12.73	+27.9%	24.04	+26.9%
16-bit		8.2	-49.6%	3.577	±0.0%	10.00	+0.5%	18.64	-1.5%
16-bit	✓	8.2	-49.6%	3.584	+0.2%	12.68	+27.4%	23.96	+21.9%
8-bit		4.1	-74.8%	3.609	+0.9%	9.69	-2.6%	18.74	-1.0%
8-bit	✓	4.1	-74.8%	3.621	+1.2%	12.89	+29.5%	23.84	+27.2%
4-bit		2.1	-87.1%	6.665	+86.3%	9.77	-1.8%	18.75	-1.0%
4-bit	✓	2.1	-87.1%	6.665	+86.3%	12.93	+29.9%	23.81	+27.0%

can be minimized by using separable convolutions only for the last two flow estimators. Moreover, we find that removing only the last flow estimator leads to a larger speed-up and overall better trade-off [HTL20], both on desktop systems and mobile devices (Tab. 3). The last flow estimator – operating on quarter input resolution – comprises only 11.3 % of parameters, but removing it results in more than 100 % speed-up on mobile devices.

Refinement. For the four previously chosen flow estimators, we find that dense refinement can be replaced by separable convolutions which even results in a slight increase in accuracy on both desktop and mobile devices (Tab. 4). For five flow estimators, we observe that dense refinement has a larger impact on accuracy.

Pruning: As an additional improvement for mobile deployment, we evaluate pruning as a post-training step. Convolution filter pruning is applied as proposed by Li *et al*. [LKD*16] and a l_1 strategy combined with automatic consistency checks [Fan19] is used for selecting which filters to prune. We apply it to each convolution layer that has more than two output channels. To keep pruning simple, we prune the same percentage of filters from each layer and then perform a single re-training to account for the loss of accuracy. We find that pruning 40 % of filters achieves a good trade-off for the final architecture – reducing accuracy by less than 10 % (< 0.5px EPE) for more than 40 % speed-up (Tab. 5). We re-train the pruned CNN with the same dataset schedule and settings as initial training, except for training on FlyingChairs [FDI*15] where we start with the lower learning rate of 1×10^{-5} and train for fewer iterations as training converges quickly, e.g., a maximum of 15 epochs (1.5 hours).

We evaluate different options of pruning as post-training optimization. As our initial training consists of multiple stages with different datasets, we first evaluate after which stage to prune and at which stage to start re-training. We observe that the best accuracy is obtained when pruning the fully trained CNN and re-training from the beginning of the dataset schedule (Tab. 7). We believe this works best as learned representations after only training on Chairs [FDI*15] or Things3D [MIH*16] are not as distinctive as after full training.

Next, we evaluate which trade-offs result from different amounts of pruned channels. For mobile devices and our final CNN, we find that pruning up to 40% of channels results in significant run-time improvement with plausible accuracy loss. Pruning 50% of channels results in a substantially higher accuracy loss (Tab. 5). For desktop, pruning – similar to reducing the number of channels (Tab. 2) – results in a smaller speed-up than on mobile devices. Considering only a small improvement of the already high frame rate in exchange for a significant loss in quality, we do not recommend pruning for the desktop version.

Quantization and Mobile Deployment: For mobile deployment, we make use of CoreML [App] as the framework for executing our CNNs on Apple mobile devices. We apply 8 bit linear weight-quantization and enable the accumulation of low-precision intermediate results (Tab. 6). This minimizes the file size by 75 % (compared to 32 bit weights) and further improves run-time performance by 30 % (on mobile devices) with only negligible accuracy loss.

Table 7: Ablation of pruning and re-training at different training stages. The *Initial* model is the one with 4 estimators and separated refinement and then we have its variations with respect to pruning at different stages. Note that increase in EPE is the least for the 3rd variant.

Training:	Chairs → Things3D → Sintel	
Variant	EPE (↓) on Sintel Train	
Initial	3.169	
Prune 30% after Chairs, re-train from Chairs	3.505	+10.6%
Prune 30% after Things3D, re-train from Things3D	3.290	+3.8%
Prune 30% after Sintel, re-train from Chairs	3.275	+3.3%
Prune 30% after Sintel, re-train from Sintel	3.413	+7.7%

Further analysis shows that our method does not profit from using the on-device Neural Processing Unit (NPU).

1.2. Implementation Details

Pruning. For convolution filter pruning we use a PyTorch [P*19] implementation by Gongfan Fang [Fan19]. We use l_1 strategy for selecting filters to prune. l_2 strategy is available too but Li *et al*. [LKD*16] show that these strategies perform comparable. We round the number of resulting channels to a multiple of 8, as other filter counts result in a run-time overhead on mobile devices. We re-train the pruned CNN with the same dataset schedule and settings as initial training, except for training on FlyingChairs [FDI*15] where we start with the lower learning rate of 1×10^{-5} and train for fewer iterations as training converges quickly, e.g., a maximum of 15 epochs (1.5 hours).

Mobile execution. We use CoreML [App] as the framework for executing our CNNs on Apple mobile devices. We evaluate using iPad Pro (11-inch, 2nd gen 2020) and iPad Air (3rd gen, 2019). CoreML efficiently implements standard CNN operations, however, the two operations specific to optical flow, i.e., correlation and warping, need to be implemented as custom layers using Metal GPU shaders for parallel and efficient computation using the mobile GPU. After CNN conversion to CoreML, we apply 8-bit linear weight quantization using coremltools , low precision acculumation is enabled to enforce low precision accumulation in all operations.

1.3. Training Settings

Similar to the original PWC-Net [SYLK18], we train our mobile architecture on the training dataset schedule FlyingChairs [FDI*15] → FlyingThings3D → Sintel [BWSB12]. Tab. 8 lists training settings for respective stages. We compute the multi-scale losses by taking the per-pixel difference between the output of each flow estimator and an accordingly downscaled ground-truth optical flow. The pixel-level loss values are summed up to a final single value which is then used as a training objective by the optimizer. Like PWC-Net [SYLK18], we scale the ground-truth optical flow with

Table 8: Our training settings for different datasets. Training times are specified for one Nvidia V100 GPU.

	FlyingChairs [FDI*15]	FlyingThings3D [MIH*16]	Sintel (Final) [BWSB12]
Batch size	8	4	4
Dataset resolution	512 × 384	960 × 540	1024 × 436
Training resolution	448 × 320	768 × 384	448 × 384
Loss function	EPE (Eqn. 1)	robust l_1 (Eqn. 2) $q = 0.4$, $\varepsilon = 0.01$	robust l_1 (Eqn. 2) $q = 0.4$, $\varepsilon = 0.01$
Initial learning rate	1×10^{-4}	1×10^{-5}	$5 \cdot 1 \times 10^{-5}$
Learning rate schedule	as S_{long} in [IMS*17]	as S_{fine} in [IMS*17]	as in [SYLK18]†
Total epochs	60	10	200
Total iterations	1200K	400K	200K
Training time	6 hours	10 hours	4 hours

Table 9: Ablation of different hyperparameters for baseline training on desktop. Sintel Final EPE is measured after training the original PWC-Net [SYLK18] architecture for 30 epochs on FlyingChairs [FDI*15].

Hyperparameters				Sintel Final Train
Optimizer	Loss weights	Regularization	Gradient stopping	EPE (↓)
Adam	exponential			5.561
Adam	exponential	l_2		6.938
AdamW	exponential			5.153
AdamW	exponential	l_2		5.263
AdamW	equal	l_2		5.086
AdamW	exponential	l_2	✓	6.269
AdamW	equal	l_2	✓	5.465

Table 10: Our augmentation settings. The same settings are applied to all training stages (tab. 8), except when fine-tuning on Sintel [BWSB12] where no Gaussian noise is added.

Augmentation	Value range	Probability
Rotation	uniform in [-17°; 17°]	0.2
Random crop	training resolution, cf. tab. 8	1.0
Horizontal flip		0.5
Vertical flip		0.5
Additive brightness	normal, $\sigma = 0.02$	1.0
RGB multiplier	uniform in [0.9; 1.1]	1.0
Contrast multiplier	uniform in [0.7; 1.3]	1.0
Gamma adjustment	uniform in [0.7; 1.5]	1.0
RGB random order		1.0
Additive gaussian noise	σ uniform from [0; 0.04]	1.0

a factor of 20 prior to calculating the loss and thus have to divide the flow estimate by 20 at test time. For training we use AdamW optimizer [LH19] with $\beta_1 = 0.09$, $\beta_2 = 0.99$, and $l2$ weight regularization with trade-off $\gamma = 0.0004$.

Given predicted flow field uv_{Pred} and ground-truth flow field uv_{GT}, EPE loss (Eqn. 1) and a robust l_1 loss (Eqn. 2) is defined as follows:

$$\mathcal{L}_{\text{EPE}}(uv_{\text{Pred}}, uv_{\text{GT}}) = \frac{1}{W \cdot H} \sum_{x,y} \|uv_{\text{Pred}}(x,y) - uv_{\text{GT}}(x,y)\|_2 \quad (1)$$

$$\mathcal{L}_{\text{robust}}(uv_{\text{Pred}}, uv_{\text{GT}}) = \frac{1}{W \cdot H} \sum_{x,y} \left(\|uv_{\text{Pred}}(x,y) - uv_{\text{GT}}(x,y)\|_1 + \varepsilon\right)^q \quad (2)$$

with typical values of $\varepsilon = 0.01$ and $q = 0.4$. $q < 1$ results in less penalty to large error values and thus makes the loss more robust to outliers, which is necessary for fine-tuning on realistic, difficult datasets [SYLK18].

1.4. Hyperparameters Configuration

We evaluate different hyperparameters while training the original PWC-Net [SYLK18] to determine a baseline that achieves the best possible accuracy. We find that AdamW optimizer [LH19] reaches a significantly better accuracy than Adam [KB15] – with and without l_2 weight regularization (Tab. 9). Furthermore, we find that equally weighted multi-scale losses – as opposed to commonly used exponential weighting [SYLK18, HTL18, HTL20, YR19] – result in slightly better accuracy. Additionally, we consider gradient stopping as proposed by Hofinger *et al.* [HBP*20], but observe that it does not result in any improvement.

1.5. Data Augmentation

Dosovitskiy *et al.* [FDI*15] found that augmentations are important for learning-based optical flow methods to prevent overfitting on synthetic training data and to ensure generalization for real-world data. Similarly, we apply geometric and color transformations to input frames and corresponding flow fields, as listed in tab. 10. Geometric transformations are applied equally to both frames of an input pair, and must be reflected accordingly in the flow field. For example, a translation applied to the frames requires a translation applied to the flow field; a rotation of frames requires a rotation of the flow field and its motion vectors. Color transformations need to be applied only to the frames, not the flow field. While it would be possible to apply different transformations (geometric, colors) per frame of an input pair [TD20] – to further increase robustness against illumination changes for example – we find that transformations applied equally per frame pair are sufficient augmentations

	Warping Error: **L1**							
	DAVIS				VIDEVO			
Task	V_p	[BTS*15]	[LHW*18]	Ours	V_p	[BTS*15]	[LHW*18]	Ours
CycleGAN/photo2ukiyoe	0.037	**0.028**	0.028	0.033	0.036	**0.026**	0.028	0.032
CycleGAN/photo2vangogh	0.049	**0.033**	0.037	0.042	0.047	**0.032**	0.036	0.041
fast-neural-style/rain-princess	0.079	**0.043**	0.055	0.062	0.076	**0.040**	0.055	0.061
fast-neural-style/udnie	0.044	**0.025**	0.031	0.036	0.038	**0.021**	0.026	0.031
WCT/antimonocromatismo	0.052	**0.030**	0.036	0.041	0.046	**0.023**	0.030	0.035
WCT/asheville	0.067	**0.040**	0.049	0.056	0.061	**0.033**	0.042	0.049
WCT/candy	0.065	**0.035**	0.045	0.051	0.058	**0.029**	0.038	0.045
WCT/feathers	0.058	**0.035**	0.040	0.049	0.054	**0.030**	0.037	0.045
WCT/sketch	0.054	**0.037**	0.038	0.046	0.050	**0.032**	0.035	0.041
WCT/wave	0.052	**0.033**	0.037	0.044	0.048	**0.029**	0.034	0.040
Average	0.056	**0.034**	0.040	0.046	0.051	**0.036**	0.036	0.042

Table 11: Flow Warping Error over stylization tasks. The optical flow evaluation is computed using GMA [JCL*21]

that prevent overfitting, while not making training of small CNN variants too difficult [HZC*17].

2. Warping Error

In Tab. 11 we show the warping error (using ℓ_1 metric, as defined in the main paper) over the stylization tasks following Lai *et al*. [LHW*18].

3. Extended User Study

To also compare with the methods of Shekhar *et al*. [SST*19] and Thimonier *et al*. [TDKP21] we conducted another user study, involving only these two methods. The setup is similar to the one described in the main paper except that it was performed by a different group of participants to avoid bias. In total, 12 persons (3 female, 8 male, and 1 did not specify) between the ages of 25 to 40 years participated in the study. Fig. 1 shows that our method surpasses the other methods by a large margin.

4. More Optical-Flow Results

In Fig. 2 and Fig. 3 we show further results for our lite optical flow network (configured as presented in the main paper) compared to other methods on Sintel and DAVIS.

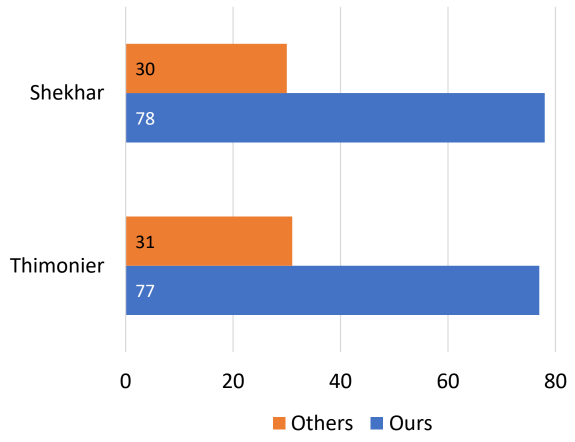


Figure 1: *Statistics of the user study results on the removal of temporal flickering from per-frame stylized videos. For* 12 *participants and* 9 *different videos, we compare our method against Shekhar et al. and Thimonier et al. through a total of* 108 *randomized A/B tests.*

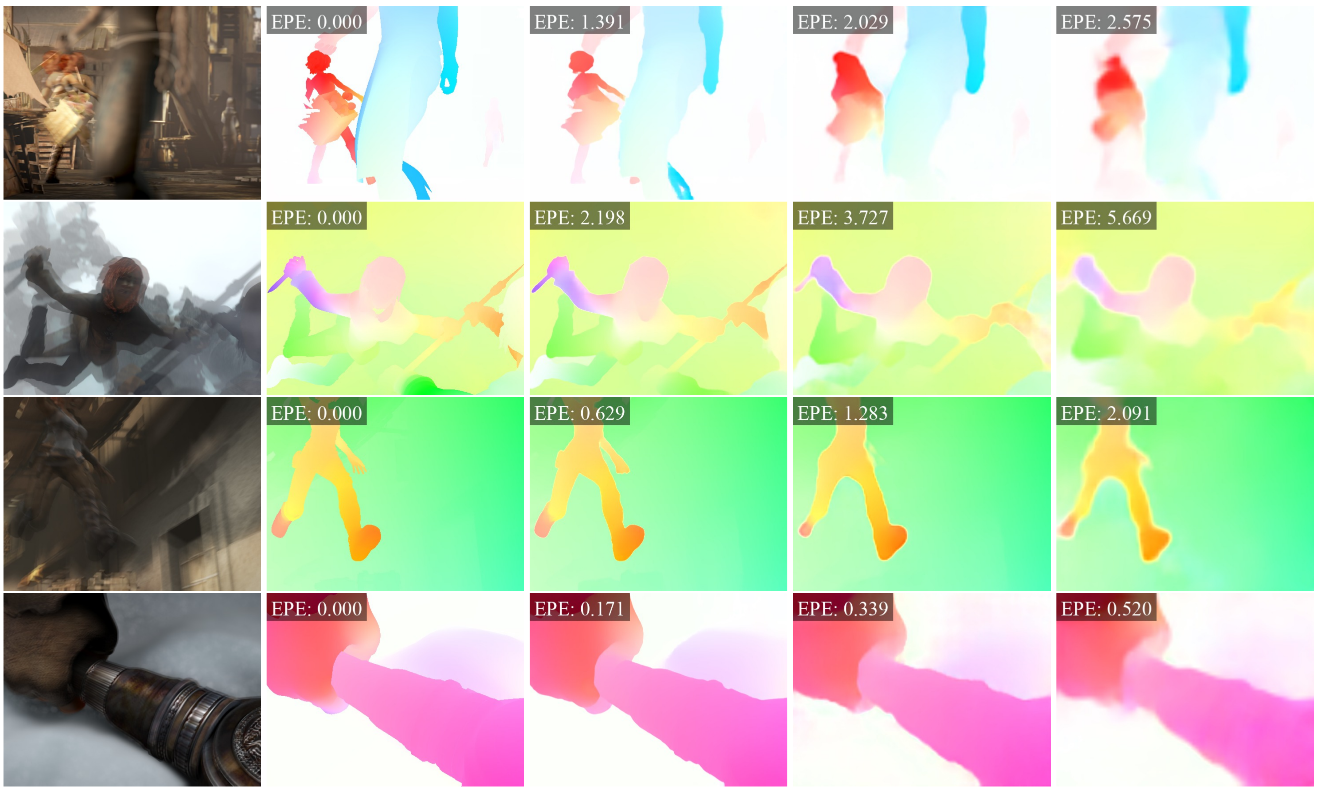
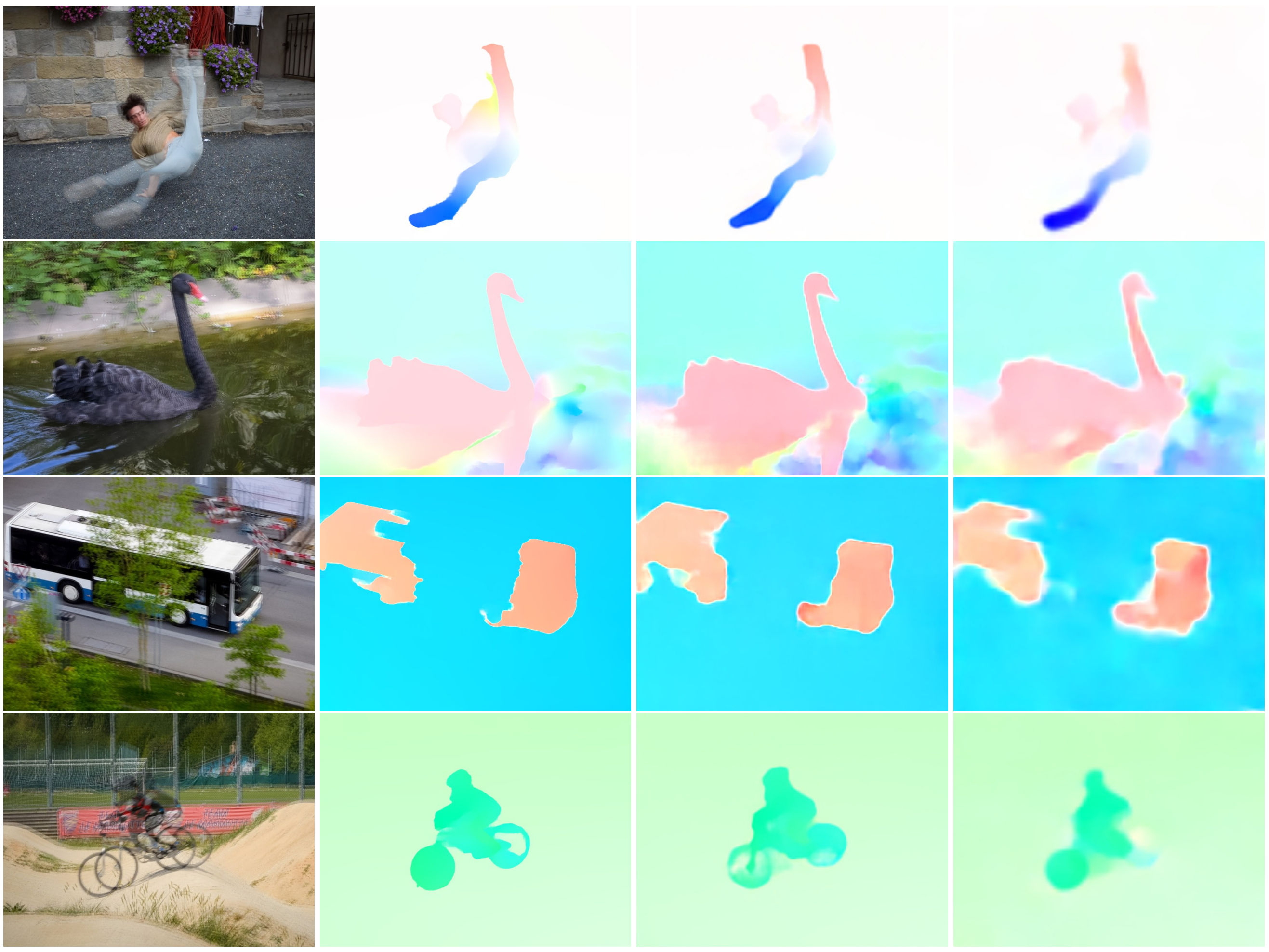


Figure 2: *Optical flow estimated using the synthetic Sintel dataset [BWSB12].*



(a) Frame Overlay *(b) RAFT [TD20]* *(c) PWC-Net [SYLK18]* *(d) Ours*

Figure 3: *Optical flow estimated for the real-world dataset DAVIS [PTPC*17].*

References

[App] Apple: Coreml. URL: https://developer.apple.com/machine-learning/core-ml. 3

[BTS*15] Bonneel N., Tompkin J., Sunkavalli K., Sun D., Paris S., Pfister H.: Blind video temporal consistency. ACM Trans. Graph. 34, 6 (oct 2015). doi:10.1145/2816795.2818107. 5

[BWSB12] Butler D. J., Wulff J., Stanley G. B., Black M. J.: A naturalistic open source movie for optical flow evaluation. In European Conference on Computer Vision (ECCV) (2012), pp. 611–625. doi:10.1007/978-3-642-33783-3_44. 1, 3, 4, 6

[Fan19] Fang G.: Torch-pruning: A pytorch pruning toolkit for structured neural network pruning and automatic layer dependency maintaining., 2019. URL: https://github.com/VainF/Torch-Pruning. 3

[FDI*15] Fischer P., Dosovitskiy A., Ilg E., Häusser P., Hazirbas C., Golkov V., van der Smagt P., Cremers D., Brox T.: Flownet: Learning optical flow with convolutional networks. In International Conference on Computer Vision (ICCV) (2015), p. 2758–2766. doi:10.1109/ICCV.2015.316. 3, 4

[HBP*20] Hofinger M., Bulò S. R., Porzi L., Knapitsch A., Pock T., Kontschieder P.: Improving optical flow on a pyramid level. In European Conference on Computer Vision (ECCV) (2020), pp. 770–786. doi:10.1007/978-3-030-58604-1_46. 4

[HLvdMW17] Huang G., Liu Z., van der Maaten L., Weinberger K. Q.: Densely connected convolutional networks. In Computer Vision and Pattern Recognition (CVPR) (2017), pp. 2261–2269. doi:10.1109/CVPR.2017.243. 1

[HTL18] Hui T.-W., Tang X., Loy C. C.: Liteflownet: A lightweight convolutional neural network for optical flow estimation. In Computer Vision and Pattern Recognition (CVPR) (2018), pp. 8981–8989. doi:10.1109/CVPR.2018.00936. 4

[HTL20] Hui T. W., Tang X., Loy C. C.: A lightweight optical flow cnn - revisiting data fidelity and regularization. In Transactions on Pattern Analysis and Machine Intelligence (TPMAI) (2020). doi:10.1109/TPAMI.2020.2976928. 3, 4

[HZC*17] Howard A. G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. In CoRR (2017). arXiv:1704.04861. 1, 5

[IMS*17] Ilg E., Mayer N., Saikia T., Keuper M., Dosovitskiy A., Brox T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In Computer Vision and Pattern Recognition (CVPR) (2017), pp. 1647–1655. doi:10.1109/CVPR.2017.179. 4

[JCL*21] Jiang S., Campbell D., Lu Y., Li H., Hartley R.: Learning to estimate hidden motions with global motion aggregation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (2021), pp. 9772–9781. 5

[KB15] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR) (2015). doi:11245/1.505367. 4

[LH19] Loshchilov I., Hutter F.: Fixing weight decay regularization in adam. In International Conference on Learning Representations (ICLR) (2019). URL: https://openreview.net/forum?id=rk6qdGgCZ. 4

[LHW*18] Lai W.-S., Huang J.-B., Wang O., Shechtman E., Yumer E., Yang M.-H.: Learning blind video temporal consistency. In Computer Vision – ECCV 2018 (2018), pp. 179–195. 5

[LKD*16] Li H., Kadav A., Durdanovic I., Samet H., Graf H. P.: Pruning filters for efficient convnets. In International Conference on Learning Representations (ICLR) (2016). URL: https://openreview.net/forum?id=rJqFGTslg. 3

[LZH*20] Liu L., Zhang J., He R., Liu Y., Wang Y., Tai Y., Luo D., Wang C., Li J., Huang F.: Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In Computer Vision and Pattern Recognition (CVPR) (2020), pp. 6488–6497. doi:10.1109/CVPR42600.2020.00652. 1

[MIH*16] Mayer N., Ilg E., Häusser P., Fischer P., Cremers D., Dosovitskiy A., Brox T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In Computer Vision and Pattern Recognition (CVPR) (2016), pp. 4040–4048. doi:10.1109/CVPR.2016.438. 3, 4

[P*19] Paszke A., et al.: Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (NIPS). 2019, pp. 8024–8035. 3

[PTPC*17] Pont-Tuset J., Perazzi F., Caelles S., Arbelaez P., Sorkine-Hornung A., Gool L. V.: The 2017 DAVIS challenge on video object segmentation. In CoRR (2017). arXiv:1704.00675. 7

[SST*19] Shekhar S., Semmo A., Trapp M., Tursun O., Pasewaldt S., Myszkowski K., Döllner J.: Consistent Filtering of Videos and Dense Light-Fields Without Optic-Flow. In Vision, Modeling and Visualization (2019), Schulz H.-J., Teschner M., Wimmer M., (Eds.). doi:10.2312/vmv.20191326. 1, 5

[SYLK18] Sun D., Yang X., Liu M.-Y., Kautz J.: PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In Computer Vision and Pattern Recognition (CVPR) (2018), pp. 8934–8943. doi:10.1109/CVPR.2018.00931. 3, 4, 6, 7

[TD20] Teed Z., Deng J.: Raft: Recurrent all-pairs field transforms for optical flow. In Computer Vision – ECCV 2020 (2020), Vedaldi A., Bischof H., Brox T., Frahm J.-M., (Eds.), pp. 402–419. doi:10.1007/978-3-030-58536-5_24. 4, 6, 7

[TDKP21] Thimonier H., Despois J., Kips R., Perrot M.: Learning long term style preserving blind video temporal consistency. In 2021 IEEE International Conference on Multimedia and Expo (ICME) (2021), IEEE, pp. 1–6. 1, 5

[YR19] Yang G., Ramanan D.: Volumetric correspondence networks for optical flow. In Advances in Neural Information Processing Systems (NIPS) (2019), pp. 794–805. URL: https://papers.nips.cc/paper/2019/hash/bbf94b34eb32268ada57a3be5062fe7d-Abstract.html. 4