Self-supervised training for blind multi-frame video denoising Supplementary material

This document contains supplementary material for our paper "Self-supervised training for blind multi-frame video denoising". It is not intended to be self-contained. It follows the notation introduced in the main paper.

A. Pseudocodes

The pseudocodes of the online and offline versions of our method are shown in Algorithms 1 and 2. In both cases, the input video is made of frames $\{f_t\}_{t\in\{1,\dots,T\}}$. For the online method, the weights are updated by N iterations of Adam optimization.

The offline method uses mini-batches of N_b frames. The weights are updated for N_A steps of Adam optimizer.

The source code of the proposed method will be made available.

B. Convergence of the offline fine-tuning

In the offline fine-tuning we estimate the gradient using mini-batches corresponding to 20 frames randomly sampled throughout the video. For each sampled frame the denoised frame is computed using the corresponding training stack. The weights are updated using the Adam update. This is repeated for $N_A = 200$ iterations.

In Fig. 1 we show the evolution of the average PSNR over the complete sequence with respect to the number of weight updates. The PSNR grows fast during the first 100 iterations. After that it continues to grow at a slower rate or plateaus. Based on this evolution, we set $N_A = 200$ iterations which is a reasonable trade-off between fine-tuning time and denoising performance.

C. Computation of the mask

The proposed MF2F loss penalizes the difference between the network output at t with the previous noisy frame f_{t-1}. The network output is aligned to frame $t-1$ by the warping operator $W_{t,t-1}$ which results from the optical flow computed between $t-1$ and t. Alignment errors have a negative impact in the training and are removed with a mask κ_t. This mask is the product (or logical AND) of two binary masks: $\kappa_t(x) = \kappa_t^{\text{OCC}}(x)\kappa_t^{\text{W}}(x)$. The first factor estimates occlusions by looking at collisions in the optical flow, similar in spirit to [1]. The second factor is explained below.

The mask κ_t^{W} is zero in areas where the warping residue is larger than a threshold and one elsewhere. For a pixel x at time t, the warping residue is computed as

$$r_{t,t-1}(x) = g_1(x) * |g_2(x) * f_{t-1}(x) - g_2(x) * W_{t,t-1} f_t(x)|_1, \quad (1)$$

where g_1 and g_2 are two Gaussian convolution kernels and the 1-norm $|\cdot|_1$ means the sum of the errors (in absolute value) for each channel at pixel x. We smooth the images with the kernel g_2 to remove some of the noise. Then, this pixelwise norm is smoothed again by the kernel g_1. In practice, we used a Gaussian kernel with $\sigma = 2$ for both g_1 and g_2. In order to further reduce noise, we downsample f_{t-1} and f_t by a factor 2. The final warping residue $r_{t,t-1}$ is then upsampled to the original resolution.

The distribution of the warping residuals can be considered as a mixture of two components. One due to the residuals caused by the noise, and the other due to registration errors. We compute a threshold such that values above that threshold are likely to be registration errors and not just differences caused by the noise. Computing such a threshold is difficult without making any assumption on the noise distribution. We will assume that the distribution of residual caused by the noise is unimodal. We compute the threshold automatically for each frame as

$$\tau_t = m_t + s_t f,$$

where m_t is the mode of the histogram of residuals $r_{t,t-1}$ and $s_t = m_t - p_t$, the difference between the mode and the 10% percentile p_t (thus we are also assuming that the mode is larger than the 10%). The mode of the histogram serves as a robust estimation of the position of the distribution, whereas s is a measure of the spread of the distribution. We use the distance between the mode and a low percentile, since we expect warping errors to affect the tail of the distribution (values larger than the mode). The histogram is smoothed by a Gaussian kernel.

In Figure 2 we show an example of warping mask computed with this strategy. In this example, the motion is very fast between two consecutive frames f_{t-1} and f_t. The arms,

Algorithm 1: Online fine-tuning

input : Noisy video $\{f_t\}_{t\in\{1,\dots,T\}}$, initial weights θ_0, number of Adam step N^S
output: Denoised video $\hat{u}$
for $t = 2, \dots, T$ **do**
  $\boldsymbol{v}_{t-1,t} \leftarrow \texttt{optical-flow}(f_{t-1}, f_t)$
  $W_{t,t-1} \leftarrow \texttt{warping-operator}(\boldsymbol{v}_{t-1,t})$
  $\kappa_t \leftarrow \texttt{alignment-error-mask}(\boldsymbol{v}_{t-1,t}, W_{t,t-1}f_t, f_{t-1})$
  $\mathcal{S}'_t \leftarrow [f_{t-4}, f_{t-2}, f_t, f_{t+2}, f_{t+4}]$ *// training input stack*
  $\mathcal{S}_t \leftarrow [f_{t-2}, f_{t-1}, f_t, f_{t+1}, f_{t+2}]$ *// inference input stack*
  // update the network
  for $i = 1, \dots, N^{\mathrm{S}}$ **do**
    $\theta_t \leftarrow \texttt{adam-step}(\ell_1^{\mathrm{MF2F}}(\mathcal{F}_\theta(\mathcal{S}'_t), f_{t-1}, W_{t,t-1}, \kappa_t))$
  // denoise the frame t
  $\hat{u}_t \leftarrow \mathcal{F}_{\theta_t}(\mathcal{S}_t)$

Algorithm 2: Offline fine-tuning

input : Noisy video f, initial weights θ_0, number of Adam updates N_A, mini-batch size N_b
output: Denoised video $\hat{u}$
for $i = 1, \dots, N_A$ **do**
  $loss \leftarrow 0$
  for $j = 1, \dots, N_b$ **do**
    $t \leftarrow \texttt{randint}(1, T)$ *// choose a random frame*
    $\boldsymbol{v}_{t-1,t} \leftarrow \texttt{optical-flow}(f_{t-1}, f_t)$
    $W_{t,t-1} \leftarrow \texttt{warping-operator}(\boldsymbol{v}_{t-1,t})$
    $\kappa_t \leftarrow \texttt{alignment-error-mask}(\boldsymbol{v}_{t-1,t}, W_{t,t-1}f_t, f_{t-1})$
    $\mathcal{S}'_t \leftarrow [f_{t-4}, f_{t-2}, f_t, f_{t+2}, f_{t+4}]$ *// training input stack*
    $\mathcal{S}_t \leftarrow [f_{t-2}, f_{t-1}, f_t, f_{t+1}, f_{t+2}]$ *// inference input stack*
    // accumulate gradients
    $loss \leftarrow loss + (\ell_1^{\mathrm{MF2F}}(\mathcal{F}_\theta(\mathcal{S}'_t), f_{t-1}, W_{t,t-1}, \kappa_t))$
  // update student network with Adam step
  $\theta_t \leftarrow \texttt{adam-step}(\text{loss})$
// Process the denoising of the entire video
for $t = 1, \dots, T$ **do**
  $\hat{u}_t \leftarrow \mathcal{F}_{\theta_t}(\mathcal{S}_t)$ *// denoise the frame t*

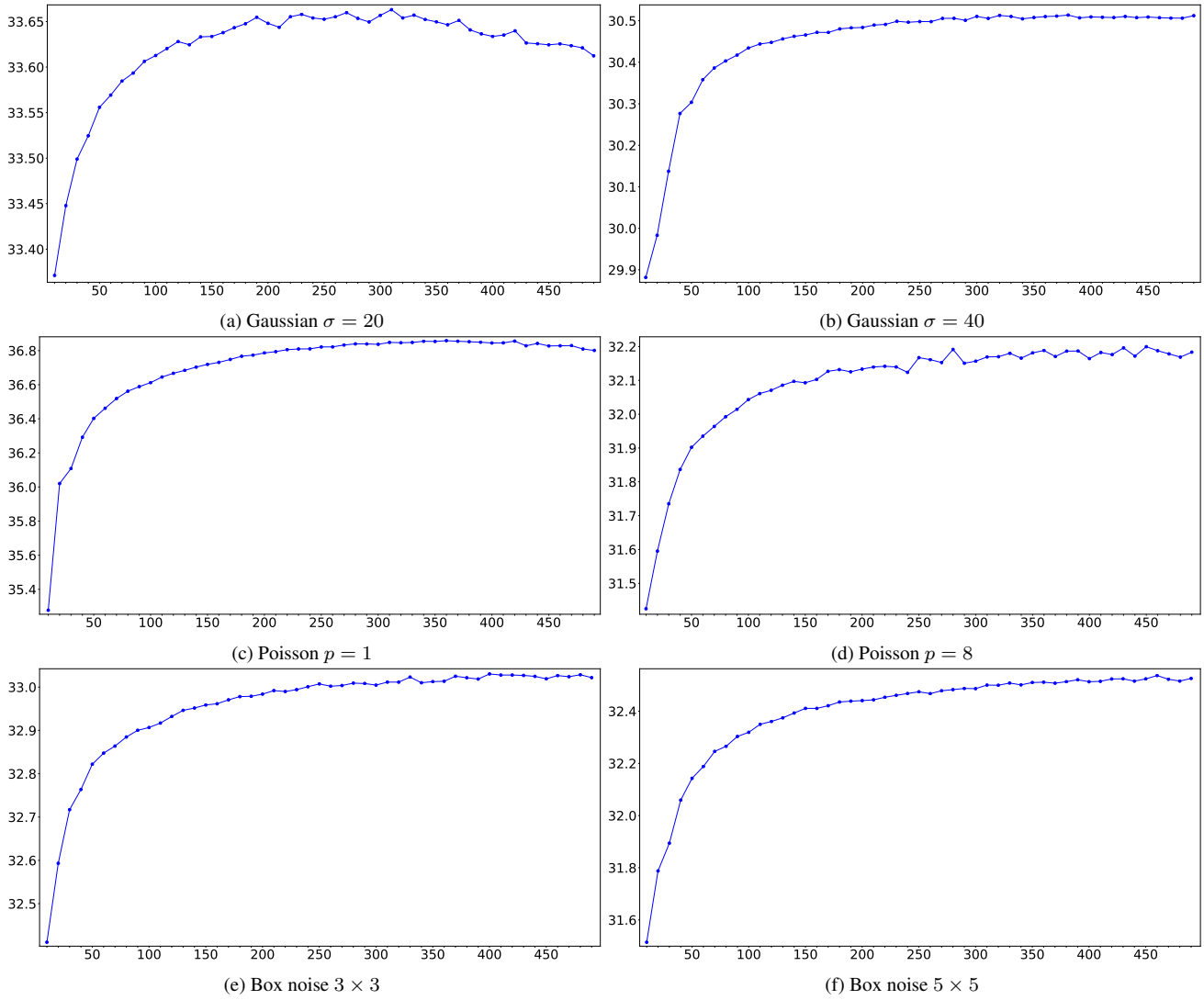
the knee of the skater and the skate itself moves quickly. This fast motion is not tracked well by the optical flow and leads to inconsistent warping for those regions. The mask κ_t removes these pixels from the loss. Figures 2e and 2f illustrate mask overlaid on the target frame (f_{t-1}) and the warped central frame of the stack (f_t) .

D. Experiment on the stack and target position configurations

The position of the target frame and the choice of the stack have already been discussed in the main paper. The table 1 extends the table 1 from the main paper by showing in addition the gain obtained by switching to the natural stack at inference compared with keeping the training one. Note that for the first row, the training stack $\mathcal{S}'_t = [f_{t-2}, f_{t-1}, f_t, f_{t+1}, f_{t+2}]$ is precisely the natural stack ($\mathcal{S}_t$). Thus both results for $\mathcal{S}_t$ and $\mathcal{S}'_t$ are equal. In the same way, for comparison, a row with the noise-specific supervised FastDVDnet was added. FastDVDnet was trained on the natural stack, The evalutation of FastDVDnet on our training stack does not make sense as it absurdly degrades its performance. We omitted them in the table.

E. Impact of pre-trained network

The proposed fine-tuning scheme can be applied to any denoising network and any pre-trained weights can be used as a starting point. Here we evaluate the impact of the choice of the pre-trained weights. This issue is related



(a) Gaussian $\sigma = 20$ (b) Gaussian $\sigma = 40$

(c) Poisson $p = 1$ (d) Poisson $p = 8$

(e) Box noise 3×3 (f) Box noise 5×5

Figure 1: Justification of the number of iterations in the offline framework: average PSNR on the whole sequence as a function of the number of Adam updates done during the fine-tuning.

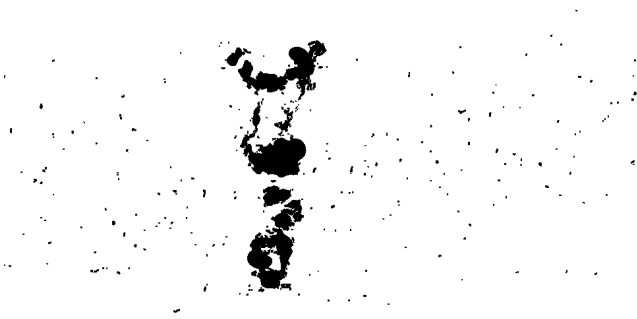
Training stack $\mathcal{S}'_t$	ref.	Box 3×3, 40 Inference stack		Gaussian 20 Inference stack		Poisson 8 Inference stack	
		$\mathcal{S}_t$	$\mathcal{S}'_t$	$\mathcal{S}_t$	$\mathcal{S}'_t$	$\mathcal{S}_t$	$\mathcal{S}'_t$
$f_{t-2}, f_{t-1}, f_t, f_{t+1}, f_{t+2}$	f_{t-3}	28.93		28.78		28.13	
$f_{t-3}, f_{t-1}, f_t, f_{t+1}, f_{t+2}$	f_{t-2}	32.02	31.90	32.25	32.12	31.18	31.03
$f_{t-3}, f_{t-2}, f_t, f_{t+1}, f_{t+2}$	f_{t-1}	**36.23**	35.98	37.25	37.10	**35.20**	35.02
$f_{t-4}, f_{t-2}, f_t, f_{t+2}, f_{t+4}$	f_{t-1}	**36.22**	35.78	**37.32**	36.94	**35.21**	34.86
FastDVDnet superv.	n/a	36.58	n/a	37.29	n/a	35.82	n/a

Table 1: PSNR results for different reference frames and training stacks $\mathcal{S}'_t$. $\mathcal{S}_t$ denotes the natural input stack. This test was carried out on the datasets Derf [4] and Vid30C-10 [2] using the online MF2F fine-tuning. The reported PSNRs are the average on all the sequences, but excluding the first 10 frames (to avoid perturbations due to the adaptation time).

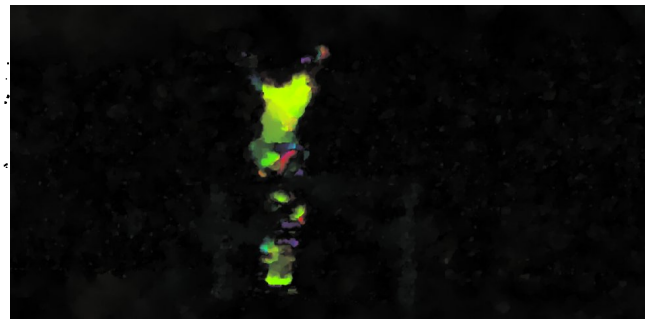
to transfer learning and domain adaptation, where a pre-trained network is re-targeted for a different task or input data distribution. In [7] it is shown that effectiveness of the transference depends on the similarity between the source

(a) Frame $t-1$

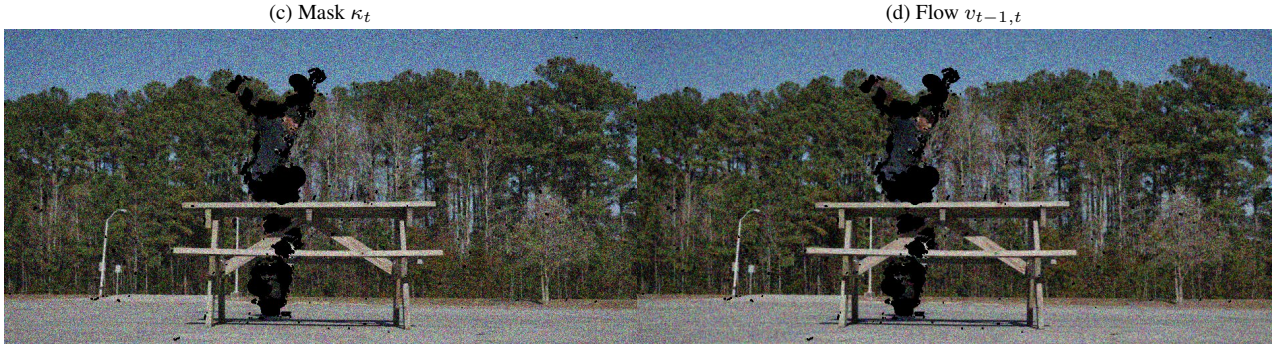
(b) Frame t



(c) Mask κ_t



(d) Flow $v_{t-1,t}$



(e) Masked target frame $\kappa_t \circ f_{t-1}$

(f) Masked warped frame $\kappa_t \circ W_{t,t-1} f_t$

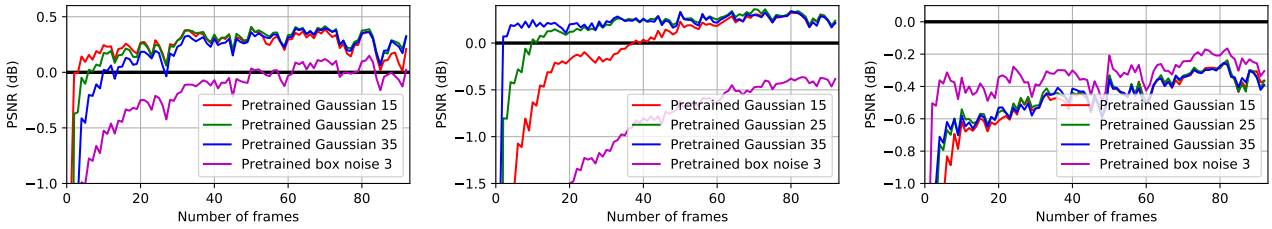
Figure 2: Example of mask, flow and warping in case of a very fast motion. The notations are those of section A.

and target tasks.

In the same spirit as [7], we tested our fine-tuning starting from weights pre-trained for four types of noise: AWGN with $\sigma = 15, 25, 35$ and box noise with with kernel size 3×3 and $\sigma = 40$. For the AWGN noise we used the pre-trained network provided by [5]. We fine-tuned those pre-trained networks for three different target noises: AWGN with small $\sigma = 10$, a stronger AWGN with $\sigma = 40$ and box noise with kernel size 5×5 and $\sigma = 65$.

Fig. 3 shows three plots, one per target noise. We consider the online version of our fine-tuning to evaluate the convergence speed. The fine-tunings were performed independently on sequences of 100 frames. For each frame, we average the PSNR obtained at that frame for the seven sequences of the Derf dataset. We plot the evolution of the difference between the average per-frame PSNR for our fine-tuned network and a network which was trained with supervision specifically for each target noise type.

As expected, the similarity between the source and target noise distributions impacts the convergence speed of the online fine-tuning. For both AWGN noise targets the weights pre-trained for AWGN with the closest σ show the fastest convergence. Similarly, the weights pre-trained for box noise work better when the target noise is also box noise. In all cases the fine-tuned network achieves a performance comparable to the supervised network (within a 0.4dB range), and even surpasses it in the case of AWGN. It seems to be easier for the weights pre-trained for AWGN to adapt to the box noise than the other way around. For this reason, all our experiments were done starting our fine-tunings from the weights pre-trained for AWGN with $\sigma = 25$.



(a) Target noise AWGN with $\sigma = 10$ (b) Target noise AWGN with $\sigma = 40$ (c) Target noise 5×5 box noise with $\sigma = 65$

Figure 3: Online MF2F fine-tuning starting from different pre-trained weights for different target noise types. For each frame, we plot the difference in PSNR with respect to the result of a noise-specific network trained with supervision. The per-frame PSNRs are averaged over the seven videos of the Derf dataset.

F. Fine-tuning only the variance map

Figure 5 shows results obtained for Poisson noise by fine-tuning the variance map. We compare the results obtained for the constant variance map and the per-level variance map (with $K = 8$ levels). Figure 6 shows an example of the per-level variance map. We recall this variance map is built by first segmenting the image in K regions based on the pixel intensity of the noisy image. To each region we assign a variance σ_i^2, and the fine-tuning is applied to these $K = 8$ variances.

The results with constant variance map clearly contain remaining noise and also over-smoothed areas, whereas the results with our variance map are uniformly denoised.

In Fig. 10, we compare some results of the MF2F fine-tuning using the per-level noise map and the spatially variant noise map on Poisson noise. The spatially variant noise map is particularly suitable for Heteroscedastic AWGN. Although in principle, the same strategy as for the space varying noise case could be also used to estimate a time-varying variance map $\Sigma_t(t)$, the results with the per-level noise map are more accurate. This is because the spatially variant noise map requires more iterations per frame (due to the slower convergence) to adapt to a temporally varying noise pattern (leading to a higher computational cost per-frame).

In Fig 4, we show an example frame of the noisy video, contaminated with the spatial Gaussian noise from the spatial noise map in figure 6 of the main paper. In the same figure, we show the corresponding denoised frame.

G. Fine-tuning half of the weights

The FastDVDnet architecture [5] consists of two cascaded blocs of U-net. In the previous section, we showed that fine-tuning the parameters of the noise map while leaving the weights fixed can achieve good results for certain types of noise. While, in the main article, we have seen that fine-tuning all the weights of the network permits to adapt to a wider range of noise types. In this section we investigate if we can update a smaller part of the network in order to attain the same adaptation capacity.

We will fine-tune half of the network weights. For that we consider four ways of splitting the weights. We can fine-tune the weights corresponding to the first Unet (denoted *first bloc*), or the second one (*second bloc*), while leaving the other fixed. But also we can fine-tune the weights of the encoder parts of both Unets (denoted *"encoder"*) or the decoder parts (*"decoder"*).

The average PSNR obtained with this fine-tuning experiments are reported in Table 2. The averages are computed over seven video sequences of the Derf dataset and ten video sequences of the Vid3oC-10 dataset. Surprisingly, one of the configuration for half fine-tuning competes with the full training. Indeed training only the encoder parts of both Unet consistently attains the performance obtained by fine-tuning the *full weights*. This is true for all the tested noises. This seems to indicate that most of the "noise-specific" work is being done in the encoders.

We can also observe that the other half-fine-tuning configurations reach a good performance and sometimes overtaking the noise-specific FastDVDnet trained with supervision.

Furthermore, in case of fine-tuning the end of the network (decoder of both Unet or encoder & decoder of the last Unet), fine-tuning half of the network does not require to back-propagate through the whole network . Thus, this allows to reduce the computational memory needed (however, the table 2 shows it slightly affects the performance compared with a full-weights training).

H. Additional results

In this section we present more results obtained with the proposed methods for different noise types. All the fine-tuned networks are obtained from the same pre-trained FastDVDnet network, trained in a supervised setting for Gaussian noise with noise level $\sigma = 25$. This network is fine-tuned blindly with the proposed method and we show that it behaves as the supervised one for many types and levels of noise.

The MF2F method was tried on two AWGN, two corre-

Figure 4: A noisy frame with spatial noise map from figure 6 in the main paper and the corresponding denoised by the self-supervised online MF2F, when fine-tuning the noise map input and keeping the network weights fixed.

Dataset & noise		Encoder	Decoder	Bloc1	Bloc2	Full weights
Derf	Gaussian 20	37.28	37.28	37.03	37.21	**37.42**
	Gaussian 40	34.19	33.89	33.94	33.27	**34.24**
	Poisson 1	40.32	40.07	39.92	38.55	**40.39**
	Poisson 8	**35.56**	35.49	35.34	35.48	**35.57**
	Box 40 3	**35.47**	35.39	35.18	35.12	**35.50**
	Box 65 5	33.85	33.64	33.43	32.96	**34.29**
	Demosaicked 4	34.70	34.60	34.46	34.49	**34.75**
Vid3oC-10	Gaussian 20	**37.35**	**37.33**	37.26	37.09	**37.32**
	Gaussian 40	**34.19**	33.90	34.10	33.12	**34.17**
	Poisson 1	**40.05**	39.70	39.79	38.00	**40.01**
	Poisson 8	**35.00**	34.76	34.90	34.44	**34.99**
	Box 40 3	**36.65**	36.56	36.47	35.72	**36.65**
	Box 65 5	**35.65**	35.46	35.42	34.32	**35.65**
	Demosaicked4	**33.96**	33.66	33.83	33.26	**33.95**

Table 2: Comparison of average PSNR over all the sequences for a given dataset and type of noise when fine-tuning the all weights or only half of them. The best PSNR is in bold. The second blind is in gray.

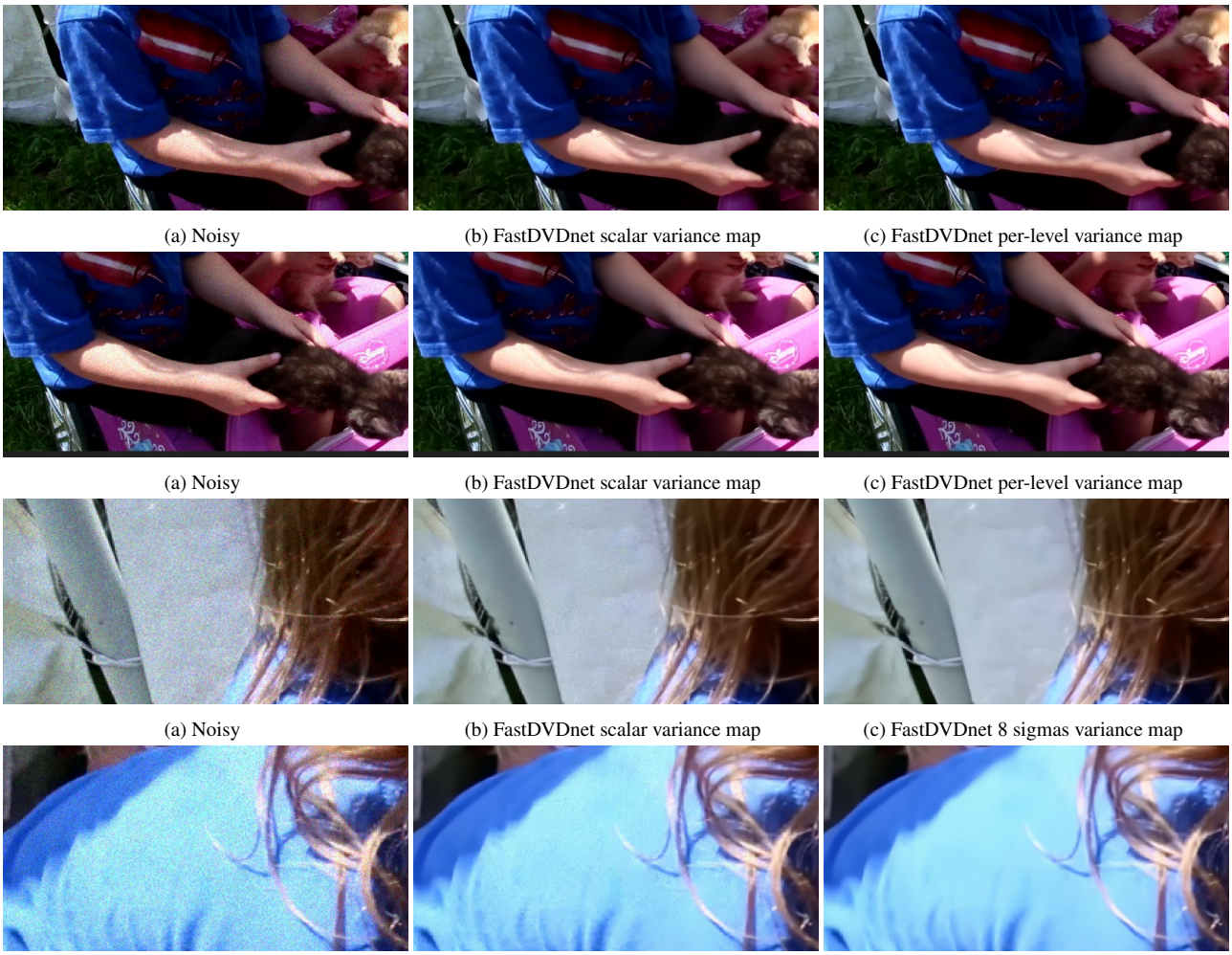
lated noise that we call "box noise" consisting of AWGN and filtered with a box filter. Finally we also tested on two scaled Poisson noise as well as on demosaicking noise (Poisson noise follows by a demosaicking algorithm).

Figures 7 and 8 illustrate the results for all the synthetic noise types used in the table 2 in the main paper, for two video sequences. In all the cases the starting point were the weights pre-trained for AWGN25. The results of our offline MF2F attains the performance of the noise-specific FastDVDnet trained in a supervised settings. We also display the results when fine-tuning the per-level variance map. From the PSNR and SSIM tables (see the main paper) we can see that for the AWGN and Poisson noise this fine-tuning yields results similar to the noise-specific FastDVDnet trained in a supervised settings. For the box noise and the demosaicked noise, the performance of MF2F is slightly below the result of the noise-specific FastDVDnet. Yet, we can see from this figure that qualitatively the results are comparable.

Additional results obtained with the proposed fine-tuning for both online and offline versions are shown in Figure 9. Those results are compared with the ones obtained by evaluation of the noise-specific FastDVDnet trained in a supervised settings in case of AWGN20. It shows that both the online and offline method achieve the performance of the supervised network and surpasses it.

Figure 11 shows results on videos with real noise from the FLIR ADAS thermal infra-red dataset, both online and offline methods are compared. Results of the last row were displayed using a *jet* color map.

Figure 12 shows the results on real noise from the CRVD dataset. In this figure we compare the results on a same scene but with different ISO levels: 1600, 3200, 6400,



(a) Noisy (b) FastDVDnet scalar variance map (c) FastDVDnet per-level variance map

(a) Noisy (b) FastDVDnet scalar variance map (c) FastDVDnet per-level variance map

(a) Noisy (b) FastDVDnet scalar variance map (c) FastDVDnet 8 sigmas variance map

(a) Noisy (b) FastDVDnet scalar variance map (c) FastDVDnet per-level variance map

Figure 5: Comparison between a constant variance map and per-level variance map for an image with Poisson noise of $p = 1$. Results with the constant variance map still contain remaining noise for bright areas. Contrast has been linearly scaled for visualization. Notice that no color variance was applied (it is not within the scope of this work to reproduce a complete image pipeline)

12800, 25600. The visual quality of denoising is not affected by the ISO level since the method quickly adapts to those different noise level. We display the results obtained both by the online MF2F and the offline MF2F. For comparison, we also added the results of online F2F and RViDeNet [6]. MF2F extends the performance of F2F and can adapt specifically to the noise of the video. As a results, it gives sharper results and with more details than RViDeNet. To illustrate that, more crops are shown in Fig. 13). RViDenet poorly reconstructs the texture of the trees, the sidewalk and even the folds in the coat. On the crops showing the legs, we see that RViDeNet has also *ghosting effect* which is not present on the results of MF2F. An illustration of this ghosting effect is also shown in Fig. 14 (see in front of the motorbike)

I. Running time

Table 3 reports the running time needed to process one color frame of 800×540 pixels (including file IO) for all the proposed online methods. The times were measured on a multi-core server with a *NVIDIA RTX 2080 TI* GPU. The online methods compute 20 Adam weight updates of the network (FastDVDnet) for each frame of the sequence. The offline method, on the other hand, performs a fixed number of Adam update steps regardless of the length of the video. A comparison with the inference time of the FastDVDnet network is also provided.

Note that fine-tuning the variance map or all the weights of the network requires roughly the same amount of time. This is because in both cases we need to back-propagate through the entire network and we perform the same num-

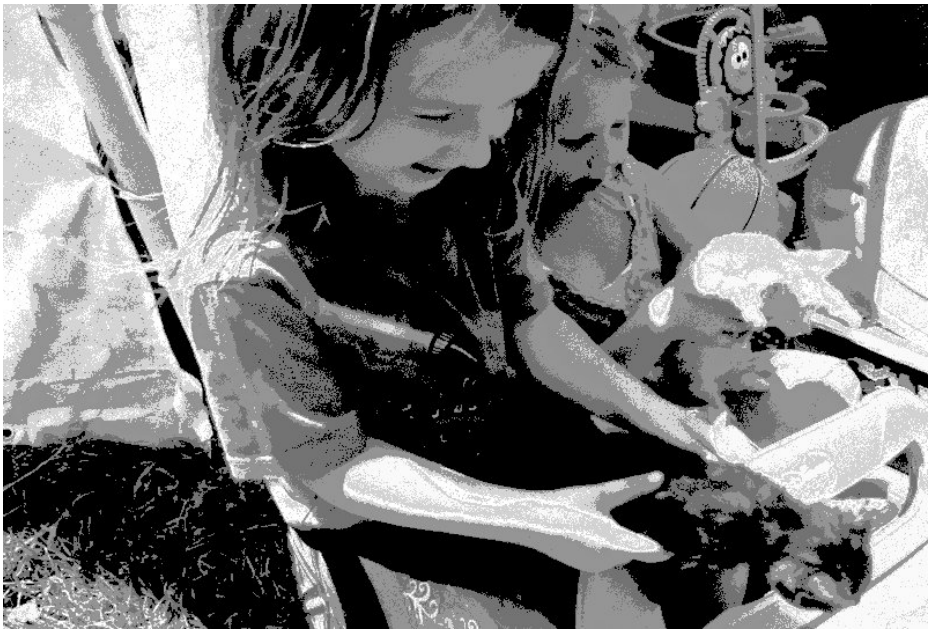


Figure 6: Obtained variance map for Poisson noise $p = 1$ (we display the square root of the variance map, *i.e.* the standard deviation). The σ found by fine-tuning the constant variance map was 10.41.

Method	time (in s)
MF2F (Online fine-tuning)	6.78
MF2F fine-tuning the 8 levels variance map	4.45
FastDVDnet (inference)	0.56

Table 3: Running time needed to process one color frame (800×540) with the online algorithm. The fine-tunings are all on the FastDVDnet network.

ber of weight update steps.

Figure 7: Comparison on all synthetic noise types. From left to right: the noisy image, the result of the noise-specific FastDVDnet (*supervised*), the result of our offline MF2F fine-tuning (*self-supervised*) and the per-level variance map MF2F (*self-supervised*). From the top to the bottom: AWGN20, AWGN40, Poisson1, Poisson8, box noise $3 \times 3, \sigma = 40$, box noise $5 \times 5, \sigma = 65$ and the demosaicking noise.

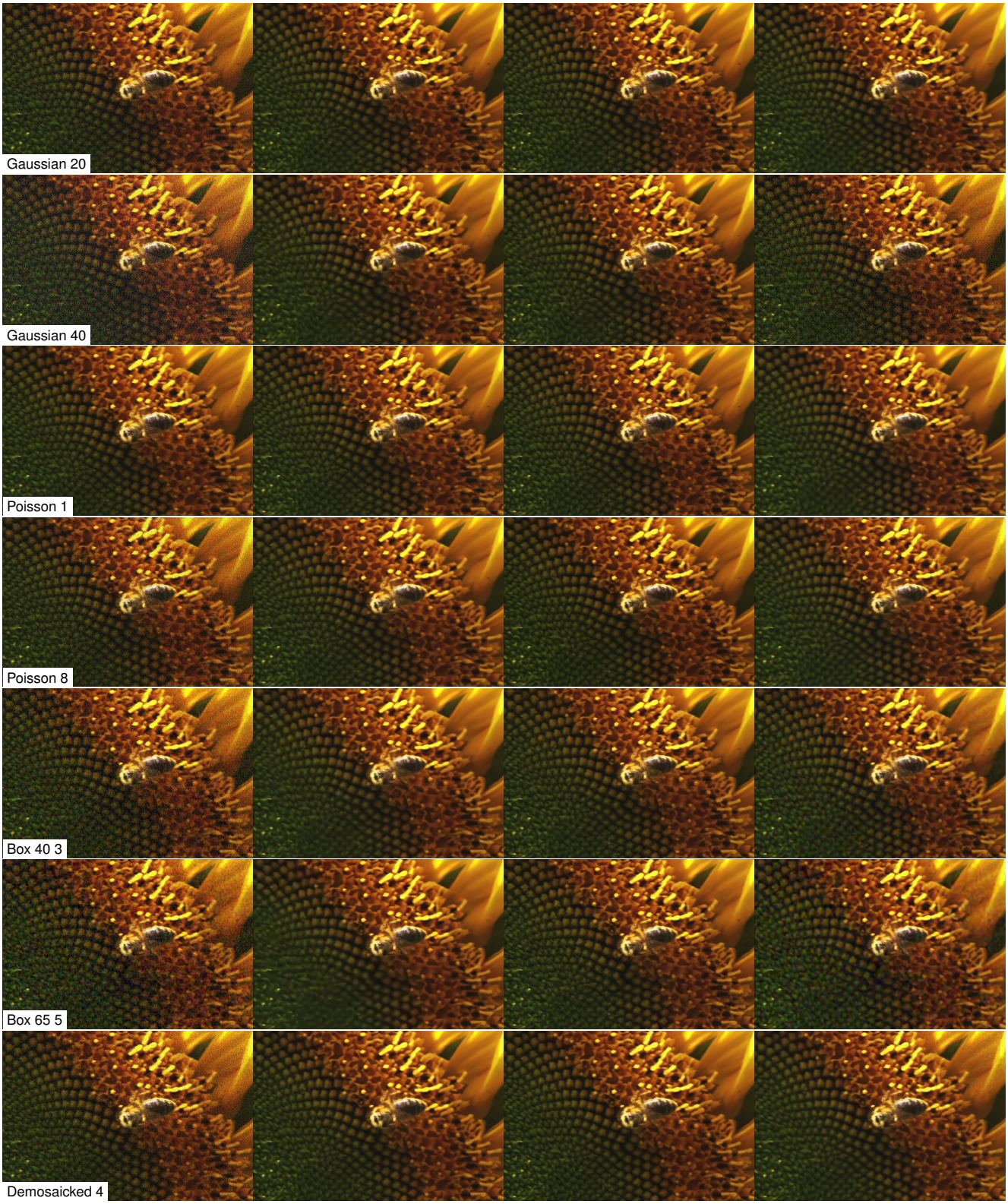


Figure 8: Comparison on synthetic noise types. From left to right: the noisy image, the result of the noise-specific FastDVDnet (*supervised*), the result of our offline MF2F fine-tuning (*self-supervised*) and the per-level variance map MF2F (*self-supervised*). From the top to the bottom: AWGN20, AWGN40, Poisson1, Poisson8, box noise $3 \times 3, \sigma = 40$, box noise $5 \times 5, \sigma = 65$ and the demosaicking noise.

Figure 9: Comparison of results obtained with the online and offline MF2F (both self-supervised) on Gaussian 20. From left to right: noise-specific FastDVDnet (*supervised*), online MF2F (*self-supervised*) and offline MF2F (*self-supervised*)

Figure 10: Comparison of results obtained with the per-level and the spatially variant variance map on poisson noise $p = 1$ (first row) and $p = 8$ (second row). From left to right: noisy, per-level variance map and spatially variant variance map.

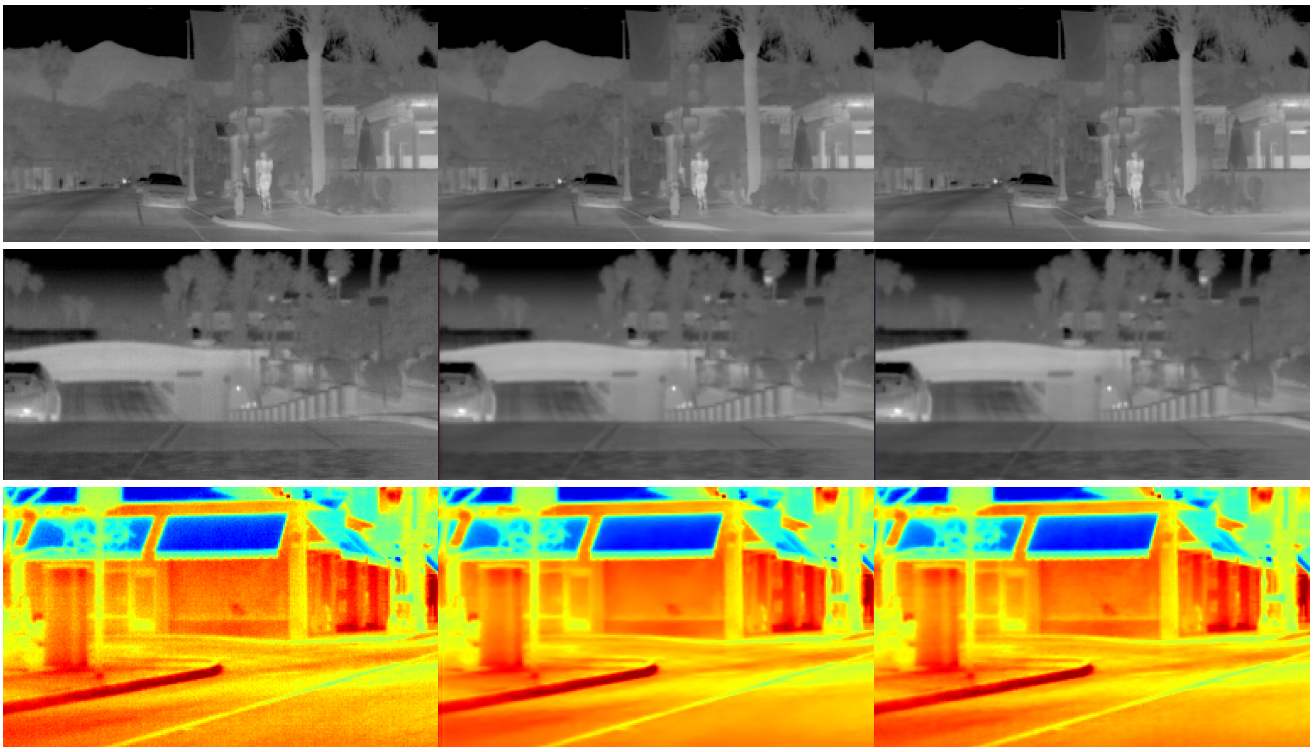


Figure 11: Results on real noise from a thermal camera (FLIR ADAS dataset). From left to right: noisy, online MF2F and offline MF2F.

(a) ISO1600 (b) ISO3200 (c) ISO6400 (d) ISO12800 (e) ISO25600

Figure 12: Real noise sequence: comparison of the same scene with different ISO levels. From top to bottom: noisy, online MF2F, offline MF2F, RViDeNet and online F2F.

(a) RViDeNet (b) Offline MF2F

(a) RViDeNet (b) Offline MF2F

(a) RViDeNet (b) Offline MF2F

Figure 13: Comparison between RViDeNet and MF2F on real noisy images [3]. The texture of trees, the coat, and the legs are poorly reconstructed by RViDeNet. On the contrary, MF2F produces results with more details. Furthermore, on the legs, we can see a ghosting effect on the result of RViDenet.

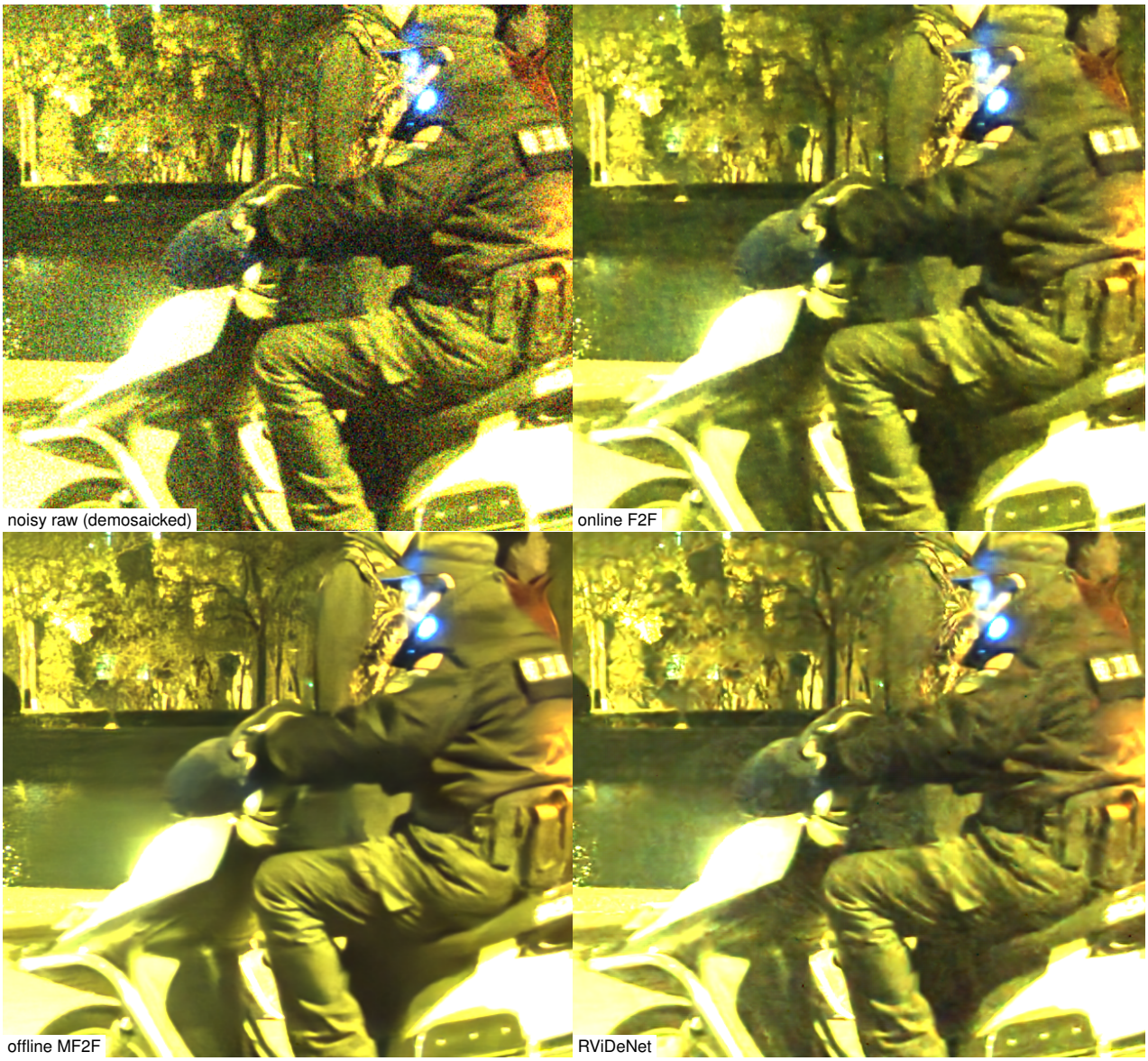


Figure 14: A frame from a denoised raw video (ISO 12800) processed by F2F, offline MF2F, and RViDeNet. The contrast was changed for display purposes. All images are demosaicked and gamma corrected. The result of RViDeNet suffers from a strong ghosting effect and the people are poorly reconstructed. On the contrary the proposed MF2F gives better results.

References

[1] Thibaud Ehret, Axel Davy, Jean-Michel Morel, Gabriele Facciolo, and Pablo Arias. Model-blind video denoising via frame-to-frame training. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[2] Sohyeong Kim, Guanju Li, Dario Fuoli, Martin Danelljan, Zhiwu Huang, Shuhang Gu, and Radu Timofte. The vid3oc and intvid datasets for video super resolution and quality mapping. In The International Conference on Computer Vision Workshop (ICCVW), pages 3609–3616. IEEE, 2019.

[3] Yoonsik Kim, Jae Woong Soh, Gu Yong Park, and Nam Ik Cho. Transfer Learning From Synthetic to Real-Noise Denoising With Adaptive Instance Normalization. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3479–3489. IEEE, jun 2020.

[4] Chris Montgomery et al. Xiph. org video test media (derf's collection), the xiph open source community, 1994. Online, https://media.xiph.org/video/derf.

[5] Matias Tassano, Julie Delon, and Thomas Veit. Fastdvdnet: Towards real-time deep video denoising without flow estimation. In The IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pages 1354–1363, June 2020.

[6] Huanjing Yue, Cong Cao, Lei Liao, Ronghe Chu, and Jingyu Yang. Supervised raw video denoising with a benchmark dataset on dynamic scenes. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2301–2310, June 2020.

[7] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3712–3722, June 2018.