Patch Craft: Video Denoising by Deep Modeling and Patch Matching

Gregory Vaksman
CS Department - The Technion
Technion City, Haifa, Israel
grishav@campus.technion.ac.il

Michael Elad
Google Research
Mountain-View, California
melad@google.com

Peyman Milanfar
Google Research
Mountain-View, California
milanfar@google.com

Abstract

The non-local self-similarity property of natural images has been exploited extensively for solving various image processing problems. When it comes to video sequences, harnessing this force is even more beneficial due to the temporal redundancy. In the context of image and video denoising, many classically-oriented algorithms employ self-similarity, splitting the data into overlapping patches, gathering groups of similar ones and processing these together somehow. With the emergence of convolutional neural networks (CNN), the patch-based framework has been abandoned. Most CNN denoisers operate on the whole image, leveraging non-local relations only implicitly by using a large receptive field. This work proposes a novel approach for leveraging self-similarity in the context of video denoising, while still relying on a regular convolutional architecture. We introduce a concept of ***patch-craft frames*** *– artificial frames that are similar to the real ones, built by tiling matched patches. Our algorithm augments video sequences with patch-craft frames and feeds them to a CNN. We demonstrate the substantial boost in denoising performance obtained with the proposed approach.*

1. Introduction

In this paper we put our emphasis on the denoising task, removing white additive Gaussian noise of known variance from visual content, focusing on video sequences. Image and video denoising is a rich and heavily studied topic, with numerous classically oriented methods and ideas that span decades of extensive research activity. The recent emergence of deep learning has brought a further boost to this field, with better performing solutions. Our goal in this paper to propose a novel video denoising strategy that builds on a synergy between the classics and deep neural networks. A key feature we build upon is the evident spatio-temporal self-similarity existing in video sequences.

Natural images are known to have a spatial *self-similarity* property – local image components tend to repeat themselves inside the same picture [33]. Imagine an image split into overlapping patches of small size (e.g. 7×7). Many of these are likely to have several similar twins in different locations in the same image. This property has been exploited extensively by classically oriented algorithms for solving various image processing problems – denoising and many other tasks. These algorithms usually split the processed image into fully overlapping patches and arrange them into some structure according to their similarity. For example, the well-known Non-Local-Means algorithm [3] filters each patch by averaging it with similar ones. The methods reported in [5, 13] group the similar patches and denoise them jointly. Alternatively, the authors of [16, 25] chain all the patches into a shortest-path, using this as a regularization for solving various inverse problems. Other methods go even farther and construct more complicated structures, such as binary trees [17] or graphs [27], and use these structures for solving image reconstruction tasks.

In recent years convolutional neural networks (CNN) entered the image restoration field and took the lead, showing impressive results (e.g., [30, 31, 14, 21, 11, 10, 32]). With this trend in place, the patch-based framework has been nearly abandoned, despite its success and popularity in classical algorithms. Most CNN based schemes work globally, operating on the whole image rather than splitting it into patches, leveraging self-similarity only implicitly by using a large receptive field. This trend has two origins: First, the reconstructed patches tend to be inconsistent on their overlaps. This undesirable and challenging phenomenon is referred to in the literature as the *local-global* gap, handled typically by plain averaging [18, 2, 20, 34]. The second reason for abandoning the patch-based framework is the difficulty of combining it with CNNs. The convolutional architecture has been shown to be a very successful choice, achieving state-of-the-art (SOTA) results in many image restoration tasks (e.g., [30, 14, 21, 11, 32]). However, such an architecture implies working on the whole image uniformly, and thus combining it with a patch-based framework is not straightforward.

Several recent denoisers have combined the patch-based

point of view within a deep-learning solution (e.g. [8, 9, 24]). Without diving into their details, these algorithms split a noisy image into fully overlapping patches, augment each with a group of similar ones and feed these groups to a denoising network. All three algorithms have managed to achieve near SOTA results while using a small number of trainable parameters. However, their performance is still challenged by leading convolutional networks, such as DnCNN and other networks [30, 32, 11, 21].

When turning to video processing, self similarity is further amplified due to the temporal redundancy. Thus, harnessing non-local processing in video is expected to be even more effective and beneficial. Many classical algorithms have successfully exploited spatio-temporal self-similarity by working on 2D or even 3D patches. For example, the V-BM4D [12] groups similar 3D patches and denoises them by a joint transform and thresholding, extending the well-known BM3D to video [5]. VNLB [1] also relies on groups of such patches, employing a joint empirical Bayes estimation for each group, under a Gaussianity modeling.

In contrast to the activity in image denoising, where many CNN-based schemes surpass classical algorithms, only a few video denoising networks have been shown to be competitive with classical methods. The recently published DVDnet [22] and FastDVDnet [23] are such networks, obtaining SOTA results, the first leveraging motion compensation, and the second combining several U-Net [19] networks for increasing its receptive field. Other CNN-based video denoisers in recent literature are [29, 4, 26, 7]

While CNN-based algorithms for video denoising, such as DVDnet and FastDVDnet, choose to work on whole frames rather than on patches, neural network based video denoising may still exploit self-similarity. For example, VNLnet [6], the first good-performing such denoiser, combines the patch-based framework with DnCNN [30] architecture, augmenting noisy frames with auxiliary feature maps that consist of central pixels taken from similar patches. While VNLnet's strategy introduces a non-locality flavor into the denoising process, it is limited in its effectiveness due to the use of central pixels instead of full neighboring patches, as evident from its performance.

Our work proposes a novel, intuitive, and highly effective way to leverage non-local self-similarity within a CNN architecture. We introduce the concept of *patch-craft* frames and use these as feature maps within the denoising process. For constructing the patch-craft frames, we split each video frame into fully overlapping patches. For each patch we find its n nearest neighbors in a spatio-temporal window, and those are used to build f (patch size) groups of corresponding n patch-craft frames. These are augmented to the real video frames and fed into a spatio-temporal denoising network. This way, self-similarity is fully leveraged, while preserving the CNN's nature of operating on whole frames, and overcoming the local-global gap.

Augmenting video sequences with patch-craft frames requires processing large amounts of data in producing each output frame. To overcome this difficulty, we use a CNN composed of multidimensional *separable convolutional* (SepConv) layers. A SepConv layer applies a series of convolutional filters, each working on a sub-group of dimensions while referring to the rest as independent tensors. Such layers implement a multidimensional separable convolution, which allows reducing the number of trainable parameters and expediting the inference.

The processing pipeline of our proposed augmentation scheme for video denoising is composed of two stages. First, we augment a noisy video sequence with patch-craft frames and feed the augmented sequence into a CNN built of SepConv layers. This stage functions mostly as a spatial filtering. At the second stage, we apply a temporal filtering, using a 3D extension of the DnCNN [30] architecture. This filter works in a sliding window manner, getting as input the outcome of the first stage with the original noisy video and producing reconstructed video at its output. Through extensive experiments, we show that the proposed method leads to a substantial boost in video denoising performance compared with leading SOTA algorithms.

To summarise, the contributions of this work are the following: We propose a neural network based video denoising scheme that consists of an augmentation of patch-craft frames, followed by a spatial and a temporal filtering. The proposed augmentation leverages non-local self-similarity using the patch-based framework, while allowing the denoising network to operate on whole frames. The deployed SepConv layers, which are used as building blocks of the spatial filtering CNN, allow reasonable memory and computational complexities for inference and learning, despite the large number of the patch-craft frames. The proposed method shows SOTA video denoising performance when compared to leading alternative algorithms.

2. Patch-Craft Frames

Let us start by motivating our approach. Consider a video frame that should be denoised by a neural network. Imagine that one could construct an artificial frame identical to the real one but with a different noise realization. Such a synthetic frame would be beneficial for denoising because it holds additional information about the processed frame. More specifically, this synthetic frame could be used as an additional feature map representing the real frame. Following this motivation, we define the patch-craft frames as such auxiliary artificial frames built of patches taken from the current and surrounding frames.

Constructing of the patch-craft frames is carried out as follows: We start by extracting all possible overlapping patches of size $\sqrt{F} \times \sqrt{F}$ from the processed frame, which

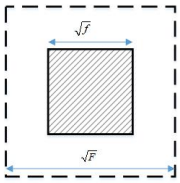
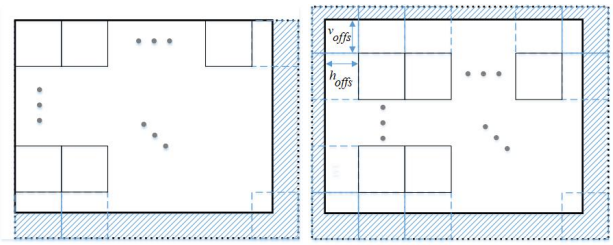


Figure 1: Patches of size $\sqrt{F} \times \sqrt{F}$ are used for nearest neighbor search. Their central $\sqrt{f} \times \sqrt{f}$ part are used for constructing patch-craft frames.



(a) Offset $[0, 0]$ (b) Offset $[v_{offs}, h_{offs}]$

Figure 2: Splitting a frame to non-overlapping patches with different offsets. Figure 2a shows a splitting to patches without an offset ($[0, 0]$), while 2b shows a splitting to patches with an offset $[v_{offs}, h_{offs}]$. The white rectangle represents the processed frame, where the blue area represents a mirror reflection of the frame pixels.

we refer to as the *current* frame, and find n nearest neighbors (most similar patches) for each extracted patch. We use the L_2 norm as distance metrics and limit the nearest neighbor search to a spatio-temporal $3D$ box of size $B \times B \times (2T_s + 1)$, where B refers to spatial axes and $2T_s + 1$ stands for the temporal window used - T_s backward and T_s forward. The n found neighbor patches are used for building the patch-craft frames where we utilize only their central parts of size $\sqrt{f} \times \sqrt{f}$, as shown in Figure 1.

The patch-craft frames are created by stitching non-overlapping patches together. More specifically, we build f groups of $n + 1$ frames. Each group contains a copy of the processed frame and n patch-craft ones. The first patch-craft frame is built by stitching the first nearest neighbors together, the second frame is constructed from the second nearest neighbors, and so on, till the last (n-th) nearest neighbor. The f groups differ by the patches' offsets. For building the first group, we use the neighbors of patches with no offset (i. e., with offset $[0, 0]$). The second group is constructed using the neighbors of patches with offset $[0, 1]$, and so on, till an offset $[\sqrt{f} - 1, \sqrt{f} - 1]$. For handling boundary pixels, we extrapolate the frame with a mirror reflection of itself and cut the leftovers after stitching the neighbors. Splitting of a frame to non-overlapping patches with different offsets is shown schematically in Figure 2.

Clearly, stitching patches with no overlaps to form an image may lead to the block boundary artifacts. Such artifacts appear, for example, in heavy JPEG compression. Therefore, the reader may wonder how do we avoid these artifacts in the patch-craft frames? Indeed, a naive attempt to construct patch-craft frames from a clean video sequence may lead to a significant block boundary artifacts. These are clearly seen in Figures 3b and 3f, which show an example of stitching together clean fifth nearest neighbors.

In order to explain why this problem is avoided in our case, we draw intuition from dithering methods. It is well-known that adding random noise before quantization causes a reduction in visual artifacts. More generally, adding random noise to a signal can help to combat structural noise. An adaptation of this idea for our case is immediate: the fact that the handled video sequence is already noisy leads to reduced artifacts, as can be seen in an example in Figure 3. A comparison between Figures 3b, 3f and Figures 3d, 3h exposes the benefit of the added random noise.

In addition to the f groups of $n + 1$ frames, we provide the denoising network with $n+1$ feature maps of scores that indicate the reliability of the patch-craft frames. A natural measure for this reliability is a patchwise squared distance between the processed and the patch-craft frames. If we denote the processed frame by y, and the jth patch-craft frame in group i by $\tilde{y}_{ij}$, then the patchwise squared distance d_{ij} can be calculated by subtracting the frames, computing the pointwise square of the difference, and convolving the result with a uniform kernel,

$$d_{ij} = \text{conv2d}\left((y - \tilde{y}_{ij}) * *2, \text{ones}\left(\sqrt{f}, \sqrt{f}\right)\right) . \quad (1)$$

Since the neural network can learn and absorb convolution kernels, we omit the last convolution. Thus we build the feature maps of scores by calculating average pointwise squared distances between the processed and patch-craft frames. More specifically, these feature maps is a group of $n + 1$ frames $\{d_j\}_{j=0}^{n}$[1], where

$$d_j = \frac{1}{f} \sum_{i=0}^{f-1} (y - \tilde{y}_{ij}) * *2 . \quad (2)$$

We concatenate these feature maps of scores with the f groups of $n + 1$ frames along the f dimension, passing the denoising network $f + 1$ groups of $n + 1$ frames for each processed original frame.

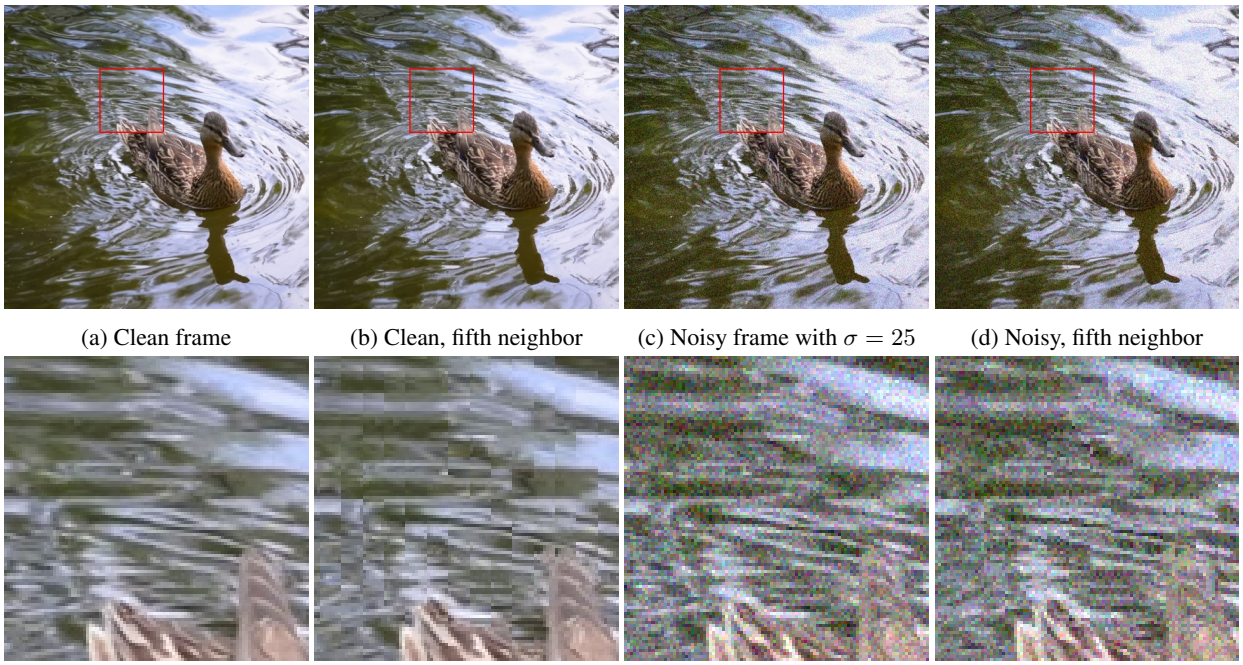
3. The Proposed Algorithm

In this section we present the proposed architecture covering the separable spatial and the temporal filters.

3.1. Separable Convolutional Neural Network

As described in the previous section, the spatial denoising network gets as input $f + 1$ groups of $n + 1$ feature maps for each processed frame. For instance, if the patch

[1] We set $\tilde{y}_{0j} = y$, thus d_0 is zero, used for preserving tensor size.



(a) Clean frame (b) Clean, fifth neighbor (c) Noisy frame with $\sigma = 25$ (d) Noisy, fifth neighbor

(e) Clean frame (f) Clean, fifth neighbor (g) Noisy frame with $\sigma = 25$ (h) Noisy, fifth neighbor

Figure 3: An example of the block boundary artifacts and the influence of the additive random Gaussian noise. This figure shows the fifth patch-craft frame, i.e., built of fifth neighbors, for frame 6 of the sequence *mallard-water*. As can be seen by comparing 3b, 3f and 3d, 3h, the patch-craft frame built using a clean sequence suffers from block boundary artifacts while the noisy data leads to artifact reduction.

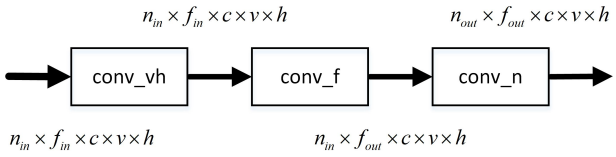


Figure 4: The SepConv layer.

size is 7×7 (i.e., $f = 49$) and $n = 14$, the number of feature maps pushed into the network for each processed frame (3 color channels) is $(49 + 1) \times (14 + 1) \times 3 = 2250$. A regular convolutional neural network would be inapplicable for processing such an amount of data, as even using the smallest possible filters of size 3×3 requires learning kernels of size $3 \times 3 \times 2250$. To overcome this difficulty, we use separable convolutional layers.

The proposed SepConv layer is shown schematically in Figure 4. It is a separable convolutional layer composed of three convolutional filters, $conv\_vh$, $conv\_f$, and $conv\_n$. Each filter works on a sub-group of dimensions and refers to the rest as independent tensors. The input and output of SepConv are five dimensional tensors of sizes $n_{in} \times f_{in} \times c \times v \times h$ and $n_{out} \times f_{out} \times c \times v \times h$ respectively, where $[v, h]$ is the frame size, c is the number of color layers, f is the patch-size, and n is the number of neighbors to be used.

The $conv\_vh$ filter applies $2D$ convolutions with kernels of size $m \times m$ referring to dimension c as input channels and dimensions n and f as independent ones. This filter represents a local spatial prior, having $n_{in} f_{in}$ groups of trainable convolution kernels. The $conv\_f$ filter applies $2D$ convolutions with 1×1 kernels referring to dimensions c and f as input channels and n as independent. This filter represents a weighted patch averaging, having n_{in} groups of trainable kernels. $conv\_n$ applies $2D$ convolutions with 1×1 kernels referring to n as input channels, while c and f are referred to as independent. This kernel represents a weighted neighbor averaging, having $f_{out} c$ groups of trainable kernels.

The spatial denoising network (S-CNN) is composed of blocks as shown in Figure 5. The first block includes the SepConv layer followed by ReLU, the middle blocks are similar to the first but with a Batch Normalization (BN) between SepConv and ReLU, and the last block consists of a single SepConv layer. Each SepConv layer reduces the number of neighbors by a factor of 2, i.e., $n_{out} = \lceil n_{in}/2 \rceil$. The network operates in the residual domain predicting the noise z_s. The output frame $\hat{y}$ is obtained by subtracting the predicted noise from the corrupted frame y.

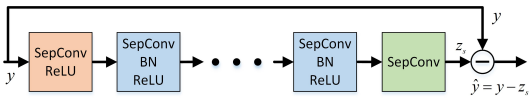
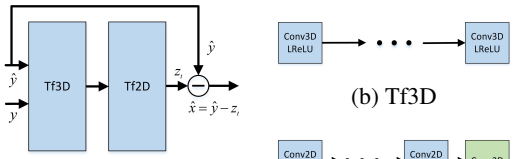
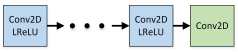


Figure 5: Our spatial denoising network S-CNN.



(a) Temporal denoising network T-CNN

(b) Tf3D



(c) Tf2D

Figure 6: Our temporal filtering network.

3.2. Temporal Filtering

Although S-CNN processes information from adjacent frames due to the augmentation, it does not guarantee temporal continuity. More generally, S-CNN does not impose an explicit temporal prior on the denoised video sequence. Thus, we apply temporal post-filtering, T-CNN, on the S-CNN output. The architecture of T-CNN is shown in Figure 6, working in a sliding window – getting as input $2T_t + 1$ frames for any output one. Each T-CNN input frame is a concatenation (along the color dimension) of the S-CNN input and output frames y and $\hat{y}$. Similar to S-CNN, T-CNN works in the residual domain, predicting the noise z_t. The output frame $\hat{x}$ is obtained by subtracting the predicted noise from the partially denoised frame $\hat{y}$. The network architecture should remind the reader of DnCNN [30]. The first part of it, Temporal Filter 3D (Tf3D), is composed of T_t blocks consisting of a $3D$ convolutions with $3 \times 3 \times 3$ kernels followed by Leaky ReLUs (LReLU). The second part, Temporal Filter 2D (Tf2D), consists of $2D$ convolutions with 3×3 kernels followed by LReLU. Each 3D kernel of Tf3D applies no padding in the temporal dimension, while padding with zeros spatially. The kernels of Tf2D apply padding with zeros as well.

4. Experimental Results

4.1. Video Denoising

In this section we report the denoising performance of our scheme, while comparing it with leading algorithms. We refer hereafter to our method as Patch-Craft Network (PaCNet). For quantitative comparisons, we use the PSNR metric, which is a commonly used measure of distortion for reconstructed video. In addition, we present qualitative comparisons between the reconstructed videos. Among classical video denoisers, we compare with V-BM4D [12] and VNLB [1] as they are the best performing classical schemes in terms of PSNR. In comparisons with CNN-based denoisers, we include VNLnet [6] since it has a flavor of non-locality, and the two SOTA networks, DVDnet [22] and FastDVDnet [23].

We test our network with additive white Gaussian noise of known variance, a standard and common scenario. The algorithm parameters are as follows: $T_s = 3$, $B = 89$, $\sqrt{F} = 15$, $\sqrt{f} = 7$, $n = 14$. S-CNN has 5 blocks (3 inner SepConv + BN + ReLU blocks) with $m = 7$. For the first SepConv layer $n_{in} = n + 1$, and $f_{in} = f + 1$. For all layers except the last $f_{out} = f_{in}$, while for the last $f_{out} = 1$. T-CNN has 17 Conv2D layers with 96 channels each, Conv3D layers have 48 channels, and $T_t = 3$. Our network is trained on 90 video sequences at 480p resolution – the DAVIS dataset [15]. The spatial and the temporal CNNs are trained separately, both using the Mean Squared Error (MSE) loss. We start by training the spatial CNN alone, and then fix its parameters and train the temporal CNN. Both networks are trained using Lamb optimizer [28] with a decreasing learning rate, starting from $5 \cdot 10^{-3}$ for the spatial and $2 \cdot 10^{-3}$ for the temporal CNNs. Our network has in total $2.87 \cdot 10^6$ trainable parameters, where $1.34 \cdot 10^6$ are S-CNN parameters and $1.53 \cdot 10^6$ are T-CNN parameters. The inference time for video resolution of 854×480 pixels is about 0.5 minute per frame on Nvidia Quadro RTX 8000 GPU or about 5.5 minutes per frame on CPU.

Table 1 reports the average PSNR performance per noise level for 30 test video sequences from the DAVIS dataset (Test-Dev 2017) at 480p resolution. As can be seen, PaCNet shows a substantial boost in denoising performance of 0.5 dB - 1.2 dB, compared with the existing SOTA algorithms. When compared to FastDVDnet and DVDnet, the PSNR benefit decreases with the increase in noise level. This behavior can be explained by the deterioration of the nearest neighbor search for higher noise levels.

Figures 7 and 8 present qualitative comparisons of our algorithm with leading alternatives. As can be seen, our method reconstructs video frames more faithfully than the competing algorithms. For example, in Figure 7, PaCNet manages to recover the eyes and preserves more details in the background trees. The comparison with VNLB [1] and FastDVDnet [23] shows that PaCNet tends to produce sharper frames with more details. The strength of the FastDVDnet is its reliance on a plain CNN architecture, but its weakness is lack of explicitly harnessing non-local self-similarity. In contrast, while VNLB leverages non-local redundancy, it is still inferior to a supervised trained CNN. PaCNet enjoys both worlds, as it combines a CNN processing with leveraging of non-local self-similarity. The synergy between these two leads to SOTA denoising performance both visually and in terms of PSNR.

We also compare PaCNet with VNLnet [6], which combines nearest neighbor search with CNN for video denois-

Noise σ	Method							
	V-BM4D [12]	VNLB [1]	VNLnet [6]	DVDnet [22]	FastDVDnet[2] [23]	S-CNN-0 (single frame)	S-CNN-3 (no T-CNN)	PaCNet (ours)
10	37.58	38.85	35.83	38.13	38.93	38.38	39.90	39.97
20	33.88	35.68	34.49	35.70	35.88	34.85	36.48	37.10
30	31.65	33.73	-[3]	34.08	34.12	32.86	34.34	35.07
40	30.05	32.32	32.32	32.86	32.87	31.56	32.78	33.57
50	28.80	31.13	31.43	31.85	31.90	30.51	31.55	32.39
Average	32.39	34.34	-	34.52	34.74	33.63	35.01	35.62

Table 1: Video denoising performance : Best PSNR is marked in red.

Method	Noise σ			Average
	10	30	50	
ViDeNN [4]	37.13	32.24	29.77	33.05
FastDVDnet[2]	38.65	33.59	31.28	34.51
PaCNet (ours)	40.13	34.92	32.15	35.73

Table 2: Denoising for clipped Gaussian noise.

Method	Noise σ			Average
	15	25	50	
LIDIA [24]	34.03	31.31	27.99	31.11
S-CNN-0 (ours)	33.95	31.22	27.93	31.03

Table 3: Single image denoising performance comparison.

ing. Although VNLnet performs non-local filtering, its effectiveness is limited due to the restricted use of central pixels of patches. As can be seen in Figure 7j, VNLnet creates a sharp frame with a good recovery, but suffers from artifacts along edges, as reflected by a 2.5dB drop compared to PaCNet in Figure 7l. In Figure 7j, these artifacts can be seen on the man's cap. We bring more qualitative comparisons in the supplementary material. Beyond high PSNR and the sharp reconstructed frames, our method produces video sequences with low flickering – this can be seen in video sequences in the supplementary material.

In addition to the above, we evaluate the denoising performance of PaCNet in experiments with a clipped Gaussian noise (i.e., truncation of noisy pixels to $[0, 1]$) and provide a comprehensive comparison to recent SOTA algorithms for this type of distortion, ViDeNN [4] and FastDVDnet [23]. In this experiment we use the same network parameters and the same training and test sets as above. Average PSNR results are reported in Table 2, exposing a similar trend to previous experiments. More specifically, PaCNet shows considerable improvement in PSNR of 0.8 dB - 1.4 dB, while the improvement decreases with an increase of noise level.

4.2. Single Image Denoising

The proposed algorithm can be easily reduced to a single image denoiser by omitting the temporal denoising network and setting $T_s = 0$. We refer to this as S-CNN-0. Among existing image denoisers, this configuration is most similar to LIDIA [24], as both methods perform nearest neighbor search as an augmentation for denoising. Table 3 shows that these two denoisers have very similar performance. In order to demonstrate the impact of each component of PaCNet, we add two columns to Table 1: S-CNN-0 and S-CNN-3. S-CNN-0 reports the performance of our scheme in an intra-frame denoising configuration, in which the video is denoised frame by frame independently. S-CNN-3 shows the PaCNet performance without the temporal filtering. In this scenario, we set $T_s = 3$, extending the nearest neighbor search in 7 adjacent frames. As can be seen, extending the nearest neighbor search to nearby frames gains more than 1 dB in PSNR, compared to a frame by frame denoising. Temporal filtering adds 0.15-0.8 dB, where this benefit increases with the increase in noise level. In addition, the temporal filter plays a key role in the reduction of flickering.

5. Conclusion

This work presents a novel algorithm for video denoising. Our method augments the processed video with patchcraft frames and applies spatial and temporal filtering on the enlarged sequence. The augmentation leverages non-local redundancy, similar to the way the patch-based framework operates. The spatial denoising network consists of separable convolutional layers, which allow for reasonable memory and computational complexities. The temporal CNN reduces flickering by imposing temporal continuity. We demonstrate the proposed method in extensive tests.[4]

[2]FastDVDnet [23] PSNR values are obtained from the released code. The rest of the values reported in Tables 1 and 2 are taken from [23].

[3]The PSNR value for VNLnet [6] with $\sigma = 30$ is missing as [6] did not provide a model for this noise level.

[4]The code reproducing the results of this paper is available at TBD.

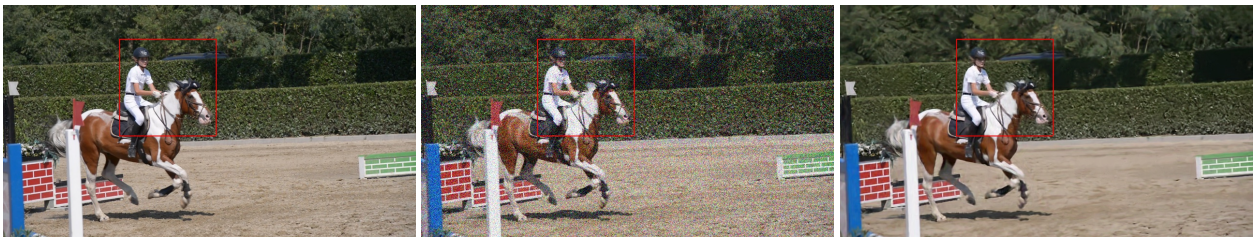
(a) Original (b) Noisy with $\sigma = 20$ (c) VNLB [1], PSNR = 36.39dB

(d) VNLnet [6], PSNR = 33.41dB (e) FastDVDnet [23], PSNR = 35.15dB (f) PaCNet (ours), PSNR = 37.31dB

(g) Original (h) Noisy with $\sigma = 20$ (i) VNLB [1], PSNR = 35.21dB

(j) VNLnet [6], PSNR = 33.61dB (k) FastDVDnet [23], PSNR = 35.00dB (l) PaCNet (ours), PSNR = 36.11dB

Figure 7: Denoising example with $\sigma = 20$. The figure shows frame 61 of the sequence *skate-jump*. The PSNR values appearing in 7c, 7d, 7e and 7f refer to the whole frame, whereas those in 7i, 7j, 7k and 7l refer to the cropped area. As can be seen, PaCNet leads to better reconstructed result – see the eyes and the details in the background trees.



(a) Original (b) Noisy with $\sigma = 40$ (c) VNLB [1], PSNR = 28.66dB

(d) VNLnet [6], PSNR = 29.03dB (e) FastDVDnet [23], PSNR = 29.27dB (f) PaCNet (ours), PSNR = 29.73dB

(g) Original (h) Noisy with $\sigma = 40$ (i) VNLB [1], PSNR = 27.92dB

(j) VNLnet [6], PSNR = 27.95dB (k) FastDVDnet [23], PSNR = 28.23dB (l) PaCNet (ours), PSNR = 29.07dB

Figure 8: Denoising example with $\sigma = 40$. The figure shows frame 48 of the sequence *horsejump-stick*. The PSNR values appearing in 8c, 8d, 8e and 8f refer to the whole frame, whereas those in 8i, 8j, 8k and 8l refer to the cropped area. As can be seen, PaCNet leads to better reconstructed results – see the face and the details in the background shrubs.

References

[1] Pablo Arias and Jean-Michel Morel. Video denoising via empirical bayesian estimation of space-time patches. *Journal of Mathematical Imaging and Vision*, 60(1):70–93, 2018. 2, 5, 6, 7, 8, 11, 13, 14, 15

[2] Dmitry Batenkov, Yaniv Romano, and Michael Elad. On the global-local dichotomy in sparsity modeling. In *Compressed Sensing and its Applications*, pages 1–53. Springer, 2017. 1

[3] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005. 1

[4] Michele Claus and Jan van Gemert. Videnn: Deep blind video denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2, 6

[5] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 1, 2

[6] Axel Davy, Thibaud Ehret, Jean-Michel Morel, Pablo Arias, and Gabriele Facciolo. A non-local cnn for video denoising. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2409–2413. IEEE, 2019. 2, 5, 6, 7, 8, 11, 13, 14, 15

[7] Thibaud Ehret, Axel Davy, Jean-Michel Morel, Gabriele Facciolo, and Pablo Arias. Model-blind video denoising via frame-to-frame training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11369–11378, 2019. 2

[8] Stamatios Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3587–3596, 2017. 2

[9] Stamatios Lefkimmiatis. Universal denoising networks: a novel cnn architecture for image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3204–3213, 2018. 2

[10] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. Non-local recurrent network for image restoration. *arXiv preprint arXiv:1806.02919*, 2018. 1

[11] Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. Multi-level wavelet-cnn for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 773–782, 2018. 1, 2

[12] Matteo Maggioni, Giacomo Boracchi, Alessandro Foi, and Karen Egiazarian. Video denoising using separable 4d non-local spatiotemporal transforms. In *Image Processing: Algorithms and Systems IX*, volume 7870, page 787003. International Society for Optics and Photonics, 2011. 2, 5, 6

[13] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *2009 IEEE 12th international conference on computer vision*, pages 2272–2279. IEEE, 2009. 1

[14] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *arXiv preprint arXiv:1603.09056*, 2016. 1

[15] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 5

[16] Idan Ram, Israel Cohen, and Michael Elad. Patch-ordering-based wavelet frame and its use in inverse problems. *IEEE transactions on image processing*, 23(7):2779–2792, 2014. 1

[17] Idan Ram, Michael Elad, and Israel Cohen. Generalized tree-based wavelet transform. *IEEE Transactions on Signal Processing*, 59(9):4199–4209, 2011. 1

[18] Yaniv Romano and Michael Elad. Patch-disagreement as away to improve k-svd denoising. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1280–1284. IEEE, 2015. 1

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[20] Jeremias Sulam and Michael Elad. Expected patch log likelihood with a sparse prior. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 99–111. Springer, 2015. 1

[21] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE international conference on computer vision*, pages 4539–4547, 2017. 1, 2

[22] Matias Tassano, Julie Delon, and Thomas Veit. Dvdnet: A fast network for deep video denoising. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1805–1809. IEEE, 2019. 2, 5, 6

[23] Matias Tassano, Julie Delon, and Thomas Veit. Fastdvdnet: Towards real-time deep video denoising without flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1354–1363, 2020. 2, 5, 6, 7, 8, 11, 13, 14, 15

[24] Gregory Vaksman, Michael Elad, and Peyman Milanfar. Lidia: Lightweight learned image denoising with instance adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 524–525, 2020. 2, 6

[25] Gregory Vaksman, Michael Zibulevsky, and Michael Elad. Patch ordering as a regularization for inverse problems in image processing. *SIAM Journal on Imaging Sciences*, 9(1):287–319, 2016. 1

[26] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019. 2

[27] Yael Yankelevsky and Michael Elad. Dual graph regularized dictionary learning. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):611–624, 2016. 1

[28] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Dem-

mel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019. 5

[29] Huanjing Yue, Cong Cao, Lei Liao, Ronghe Chu, and Jingyu Yang. Supervised raw video denoising with a benchmark dataset on dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2301–2310, 2020. 2

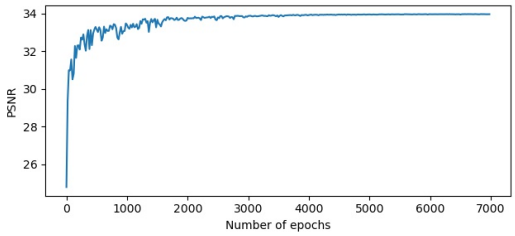
[30] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017. 1, 2, 5

[31] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018. 1

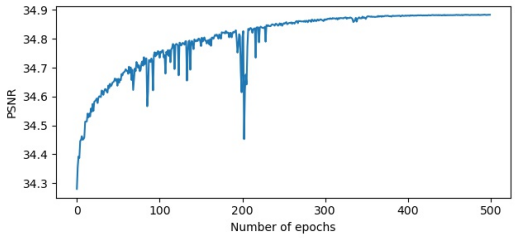
[32] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1, 2

[33] Maria Zontak and Michal Irani. Internal statistics of a single natural image. In *CVPR 2011*, pages 977–984. IEEE, 2011. 1

[34] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486. IEEE, 2011. 1



(a) S-CNN



(b) T-CNN

Figure 9: PSNR vs. the number of epochs for the validation set during training of the spatial and the temporal denoising networks, S-CNN and T-CNN, for noise level $\sigma = 30$. (We use different validation sets for each network).

Appendices

A. Additional Details Regarding Training

Figures 9a and 9b present graphs of PSNR versus number of epochs during training of our networks. The values shown in the graphs are a rough estimation of training PSNR obtained by evaluating the networks on a small set of short videos randomly cropped from the training set. The spatial network, S-CNN, is trained using spatio-temporal $3D$ boxes of size $150 \times 150 \times 7$, applying denoising on the central frame of size $64 \times 64 \times 1$, where the rest of the box is used for nearest neighbor search. The boxes are randomly cropped from the training video seqiences. We use batches of size 10 and train the network for 7000 epochs. For training the temporal network, T-CNN, we use batches of 10 randomly cropped spatial-temporal boxes of size $64 \times 64 \times 7$ and run training for 500 epochs.

B. Additional Results

Figure 11 presents graphs showing PSNR versus frame number for several test video sequences comparing PaCNet performance with VNLB [1], VNLnet [6], and FasDVDnet [23]. Figures 12, 13 and 14 show visual comparisons of our method versus leading algorithms. In addition to these figures, we attach to our paper several video (AVI) files that show comparisons of video sequences. Each file

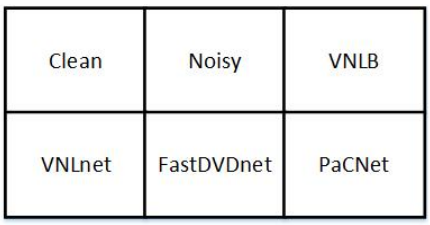


Figure 10: Video chart.

simultaneously plays the outcomes of four denoising algorithms: VNLB [1], VNLnet [6], FastDVDnet [23], and PaCNet (ours), along with the clean and the noisy sequences. These sequences are arranged according to the chart shown in Figure 10.

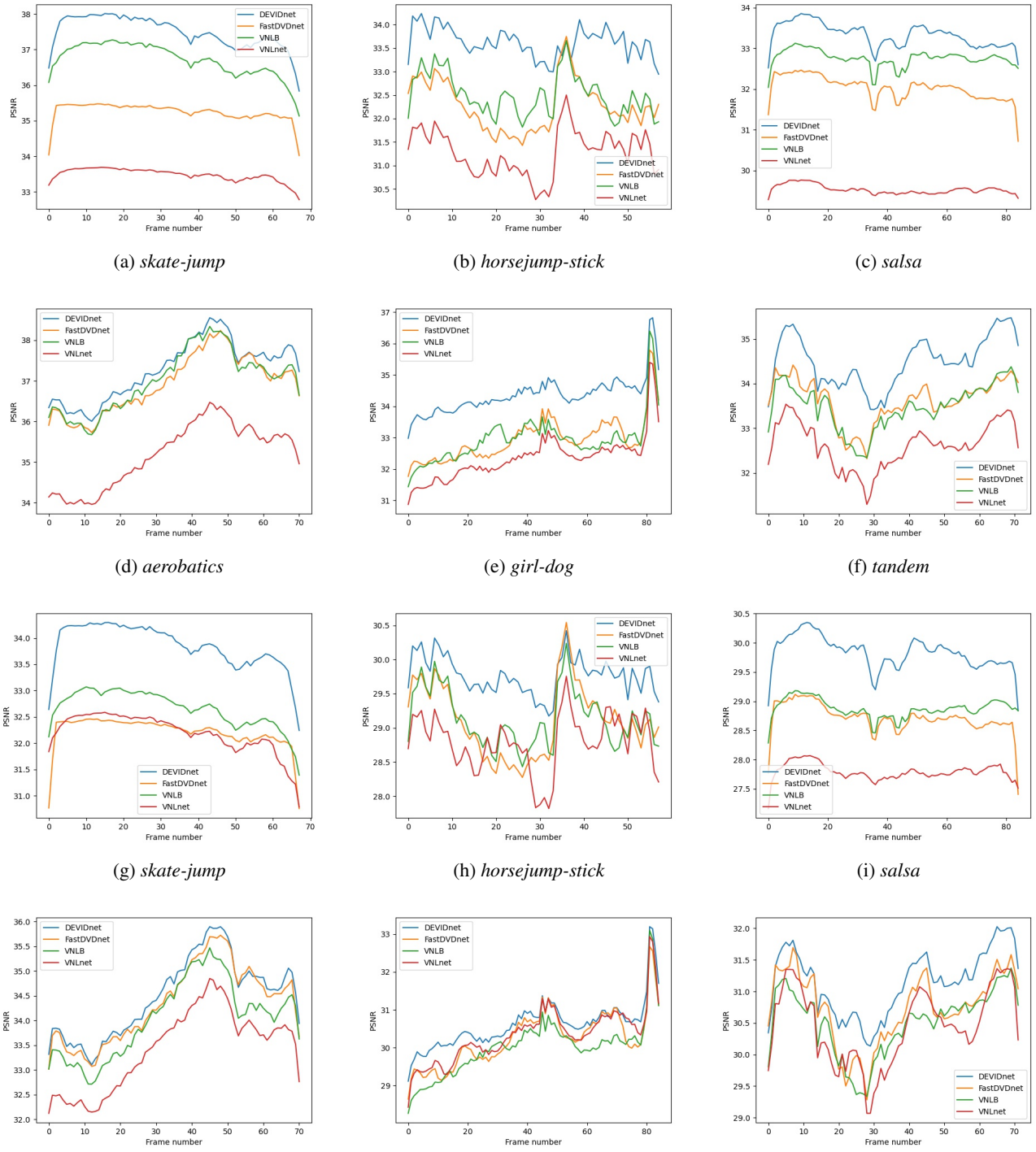
Files *salsa_s40_merge_rect.avi* and *skate-jump_s20_merge_rect.avi* show the video sequences *salsa* and *skate-jump* contaminated by noise with $\sigma = 40$ and $\sigma = 20$ respectively. There are two rectangles, red and green, in each video. The rest four files show zoom-in on the area in these rectangles:

- The green rectangle in *salsa* is shown in *salsa_s40_merge_zoom_g.avi*. As can be seen, our result is sharper than the VNLB outcome and less noisy than the outputs of FastDVDnet and VNLnet – see for example the floor. Also, observe that the VNLnet has noticeable artifacts around the legs.

- The red rectangle in *salsa* is shown in *salsa_s40_merge_zoom_r.avi*. As can be seen, PaCNet leads to better reconstruction – see for example the brick wall. Our output is sharper and less noisy than the competitors' results.

- The green and the red rectangles of *skate-jump* are shown in *skate-jump_s20_merge_zoom_g.avi* and *skate-jump_s20_merge_zoom_r.avi* respectively. As can be seen here as well, our algorithm leads to better reconstruction – e.g. see the trees.

The videos are better seen in repeat mode.



(a) *skate-jump* (b) *horsejump-stick* (c) *salsa*

(d) *aerobatics* (e) *girl-dog* (f) *tandem*

(g) *skate-jump* (h) *horsejump-stick* (i) *salsa*

(j) *aerobatics* (k) *girl-dog* (l) *tandem*

Figure 11: PSNR vs. frame number for video sequences *skate-jump*, *horsejump-stick*, *salsa*, *aerobatics*, *girl-dog*, and *tandem*. The two first rows show denoising experiments with noise level $\sigma = 20$ and the third and fourth rows with $\sigma = 40$.

(a) Original (b) Noisy with $\sigma = 40$ (c) VNLB [1], PSNR = 29.14dB

(d) VNLnet [6], PSNR = 28.03dB (e) FastDVDnet [23], PSNR = 29.03dB (f) PaCNet (ours), PSNR = 30.15dB

(g) Original (h) Noisy with $\sigma = 40$ (i) VNLB [1], PSNR = 27.68dB

(j) VNLnet [6], PSNR = 26.69dB (k) FastDVDnet [23], PSNR = 27.56dB (l) PaCNet (ours), PSNR = 28.31dB

Figure 12: Denoising example with $\sigma = 40$. The figure shows frame 9 of the sequence *salsa*. The PSNR values appearing in 12c, 12d, 12e and 12f refer to the whole frame, whereas those in 12i, 12j, 12k and 12l refer to the cropped area. As can be seen, PaCNet leads to better reconstructed results – see the face and the details in the background building.

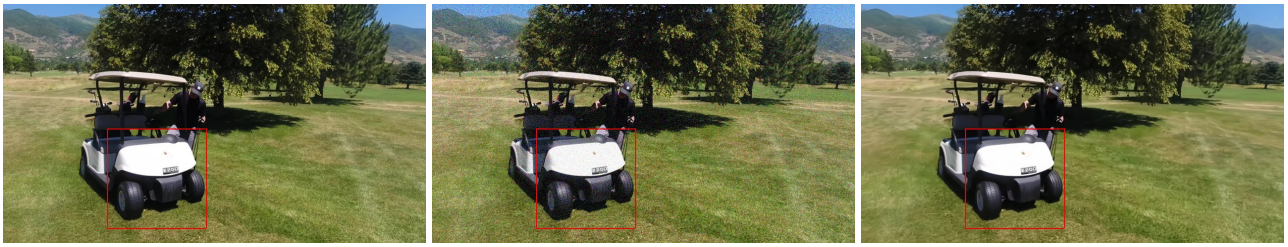
(a) Original (b) Noisy with $\sigma = 20$ (c) VNLB [1], PSNR = 34.18dB

(d) VNLnet [6], PSNR = 33.26dB (e) FastDVDnet [23], PSNR = 34.42dB (f) PaCNet (ours), PSNR = 34.74dB

(g) Original (h) Noisy with $\sigma = 20$ (i) VNLB [1], PSNR = 38.43dB

(j) VNLnet [6], PSNR = 37.71dB (k) FastDVDnet [23], PSNR = 38.53dB (l) PaCNet (ours), PSNR = 39.09dB

Figure 13: Denoising example with $\sigma = 20$. The figure shows frame 23 of the sequence *tractor*. The PSNR values appearing in 13c, 13d, 13e and 13f refer to the whole frame, whereas those in 13i, 13j, 13k and 13l refer to the cropped area. As can be seen, PaCNet leads to better reconstructed results – see the text on the trailer.



(a) Original (b) Noisy with $\sigma = 20$ (c) VNLB [1], PSNR = 31.22dB

(d) VNLnet [6], PSNR = 30.19dB (e) FastDVDnet [23], PSNR = 31.29dB (f) PaCNet (ours), PSNR = 32.14dB

(g) Original (h) Noisy with $\sigma = 20$ (i) VNLB [1], PSNR = 32.75dB

(j) VNLnet [6], PSNR = 31.15B (k) FastDVDnet [23], PSNR = 32.40dB (l) PaCNet (ours), PSNR = 33.42dB

Figure 14: Denoising example with $\sigma = 20$. The figure shows frame 18 of the sequence *golf*. The PSNR values appearing in 14c, 14d, 14e and 14f refer to the whole frame, whereas those in 14i, 14j, 14k and 14l refer to the cropped area. As can be seen, PaCNet leads to better reconstructed results – see the pattern on wheels.