Non-Local Video Denoising by CNN

Axel Davy Thibaud Ehret Jean-Michel Morel
Pablo Arias Gabriele Facciolo
CMLA, ENS Cachan, CNRS
Université Paris-Saclay, 94235 Cachan, France
axel.davy@ens-cachan.fr*

Abstract

Non-local patch based methods were until recently state-of-the-art for image denoising but are now outperformed by CNNs. Yet they are still the state-of-the-art for video denoising, as video redundancy is a key factor to attain high denoising performance. The problem is that CNN architectures are hardly compatible with the search for self-similarities. In this work we propose a new and efficient way to feed video self-similarities to a CNN. The non-locality is incorporated into the network via a first non-trainable layer which finds for each patch in the input image its most similar patches in a search region. The central values of these patches are then gathered in a feature vector which is assigned to each image pixel. This information is presented to a CNN which is trained to predict the clean image. We apply the proposed architecture to image and video denoising. For the latter patches are searched for in a 3D spatio-temporal volume. The proposed architecture achieves state-of-the-art results. To the best of our knowledge, this is the first successful application of a CNN to video denoising.

1 Introduction

Advances in image sensor hardware have steadily improved the acquisition quality of image and video cameras. However, a low signal-to-noise ratio is unavoidable in low lighting conditions if the exposure time is limited (for example to avoid motion blur). This results in high levels of noise, which negatively affects the visual quality of the video and hinders its use for many applications. As a consequence, denoising is a crucial component of any camera pipeline. Furthermore, by interpreting denoising algorithms as proximal operators, several inverse problems in image processing can be solved by iteratively applying a denoising algorithm [42]. Hence the need for video denoising algorithms with a low running time.

Literature review on image denoising. Image denoising has a vast literature where a variety of methods have been applied: PDEs and variational methods (including MRF models) [45, 11, 43], transform domain methods [18], non-local (or patch-based) methods [7, 17], multiscale approaches [21], etc. See [30] for a review. In the last two or three years, CNNs have taken over the state-of-the-art. In addition to attaining better results, CNNs are amenable to efficient parallelization on GPUs potentially enabling real-time performance. We can distinguish two types of CNN approaches: *trainable inference networks* and *black box* networks.

In the first type, the architecture mimics the operations performed by a few iterations of optimization algorithms used for MAP inference with MRFs prior models. Some approaches are based on the Field-of-Experts model [44], such as [5, 48, 14]. The architecture of [52] is based on EPLL [57], which models the *a priori* distribution of image patches as a Gaussian mixture model. Trainable inference networks reflect the operations of an optimization algorithm, which leads in some cases to unusual architectures, and to some restrictions in the network design. For example, in the *trainable reaction diffusion network* (TRDN) of [14] even layers must be an image (i.e. have only one feature). As pointed out in [28] these architectures have strong similarities with the residual networks of [23].

The black-box approaches treat denoising as a standard regression problem. They do not use much of the domain knowledge acquired during decades of research in denoising. In spite of this, these techniques are currently topping the list of state-of-the-art algorithms. The first denoising approaches using neural networks were proposed in the mid and late 2000s. Jain and Seung [26] proposed a five layer CNN with 5×5 filters, with 24 features in the hidden layers and sigmoid activation functions. Burger et al. [10] reported

*Work supported by IDEX Paris-Saclay IDI 2016, ANR-11-IDEX-0003-02, ONR grant N00014-17-1-2552, CNES MISS project, DGA Astrid ANR-17-ASTR-0013-01, DGA ANR-16-DEFA-0004-01. The TITAN V used for this research was donated by the NVIDIA Corporation.

the first state-of-the-art results with a multilayer perceptron trained to denoise 17×17 patches, but with a heavy architecture. More recently, DnCNN [55] obtained impressive results with a far lighter 17 layer deep CNN with 3×3 convolutions, ReLU activations and batch normalization [25]. This work also proposes a blind denoising network that can denoise an image with an unknown noise level $\sigma \in [0, 55]$, and a multi-noise network trained to denoise blindly three types of noise. A faster version of DnCNN, named FFDNet, was proposed in [56], which also allows handling noise with spatially variant variance $\sigma(x)$ by adding the noise variance map as an additional input. The architectures of DnCNN and FFDnet keep the same image size throughout the network. Other architectures [36, 47, 12] use pooling or strided convolutions to downscale the image, and then up-convolutional layers to upscale it back. Skip connections connect the layers before the pooling with the output of the up-convolution to avoid loss of spatial resolution. Skip connections are used extensively in [51].

Although these architectures produce very good results, for textures formed by repetitive patterns, non-local patch-based methods still perform better [55, 10]. Some works have therefore attempted to incorporate the non-local patch similarity into a CNN framework. Qiao *et al*. [41] proposed inference networks derived from the non-local FoE MRF model [50]. This can be seen as a non-local version of the TRDN network of [14]. A different non-local TRDN was introduced by [31]. BM3D-net [54] pre-computes for each pixel a stack of similar patches which are fed into a CNN, which reproduces the operations done by (the first step of) the BM3D algorithm: a linear transformation of the group of patches, a non-linear shrinkage function and a second linear transform (the inverse of the first). The authors train the linear transformations and the shrinkage function. In [15] the authors propose an iterative approach that can be used to reinforce non-locality to any denoiser. Each iteration consists of the application of the denoiser followed by a non-local filtering step using a fixed image (denoised with BM3D) for computing the non-local correspondences. This approach obtains good results and can be applied to any denoising network. An inconvenience is that the resulting algorithm requires to iterate the denoising network. Trainable non-local modules have been proposed recently by using differentiable relaxations of the 1 nearest neighbors [32] and k nearest neighbors [39] selection rules.

Literature review on video denoising. CNNs have been successfully applied to several video processing tasks such as deblurring [49], video frame synthesis [33] or super-resolution [24, 46], but their application to video denoising has been limited so far. In [13] a recurrent architecture is proposed, but the results are below the state-of-the-art. Some works have tackled the related problem of burst denoising. Recently [22, 38] focused on the related problem of image burst denoising reporting very good results.

In terms of output quality the state-of-the-art is achieved by patch-based methods [16, 35, 3, 19, 9, 53]. They exploit drastically the self-similarity of natural images and videos, namely the fact that most patches have several similar patches around them (spatially and temporally). Each patch is denoised using these similar patches, which are searched for in a region around it. The search region generally is a space-time cube, but more sophisticated search strategies involving optical flow have also been used. Because of the use of such broad search neighborhoods these methods are called *non-local*. While these video denoising algorithms perform very well, they often are computationally costly. Because of their complexity they are usually unfit for high resolution video processing.

Patch-based methods usually follow three steps that can be iterated: (1) search for similar patches, (2) denoise the group of similar patches, (3) aggregate the denoised patches to form the denoised frame. VBM3D [16] improves the image denoising algorithm BM3D [17] by searching for similar patches in neighboring frames using a "predictive search" strategy which speeds up the search and gives some temporal consistency. VBM4D [35] generalizes this idea to 3D patches. In VNLB [2] spatio-temporal patches that were not motion compensated are used to improve the temporal consistency. In [19] a generic search method extends every patch-based denoising algorithm into a global video denoising algorithm by extending the patch search to the entire video. SPTWO [9] and DDVD [8] use optical flow to warp the neighboring frames to each target frame. Each patch of the target frame is then denoised using the similar patches in this volume with a Bayesian strategy similar to [29]. Recently, [53] proposed to learn an adaptive optimal transform using batches of frames.

Patch-based approaches achieve also the state-of-the-art among frame-recursive methods [20, 4]. These methods compute the current frame using only the current noisy frame and the previous denoised frame. They achieve lower results than non-recursive methods, but have a lower memory footprint and (potentially) lower computational cost.

Contributions. In this work we propose a non-local architecture for image and video denoising that does not suffer from the restrictions of trainable inference networks.

The method first computes for each image patch the n most similar neighbors in a rectangular spatio-temporal search window and gathers the center pixel of each similar patch forming a feature vector which is assigned to each image location. This results in an image with n channels, which is fed to a CNN trained to predict the clean image from this high dimensional vector. We trained our network for grayscale and color video denoising. Practically training

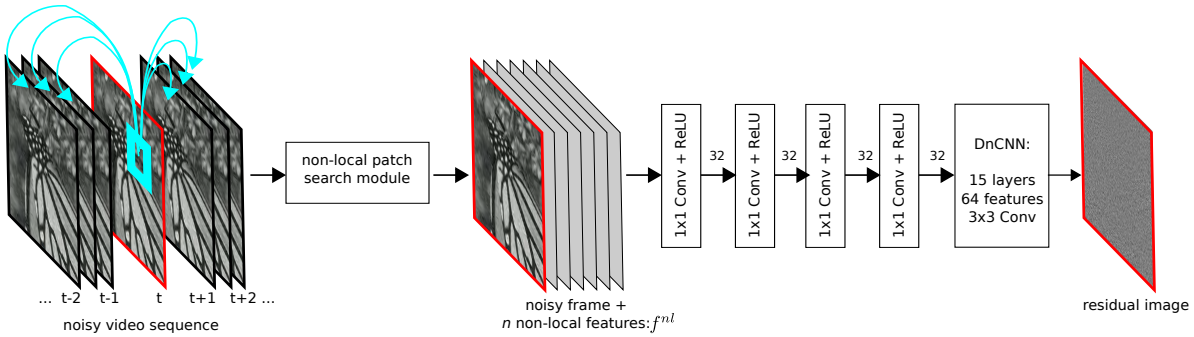

Figure 1: The architecture of the proposed method. The first module performs a patch-wise nearest neighbor search across neighboring frames. Then, the current frame, and the feature vectors f^{nl} of each pixel (the center pixels of the nearest neighbors) are fed into the network. The first four layers of the network perform 1×1 convolutions with 32 feature maps. The resulting feature maps are the input of a simplified DnCNN [55] network with 15 layers.

this architecture is made possible by a GPU implementation of the patch search that allows computing the nearest neighbors efficiently. The self-similarity present temporally in videos enables strong denoising results with our proposal.

To summarize our contributions, in this paper we present a new video denoising CNN method incorporating non-local information in a simple way. To the best of our knowledge, the present work is the first CNN-based video denoising method to attain state-of-the-art results.

2 Proposed method

Let u be a video and $u(x,t)$ denote its value at position x in frame t. We observe v, a noisy version of u contaminated by additive white Gaussian noise:

$$v = u + r,$$

where $r(x,t) \sim \mathcal{N}(0, \sigma^2)$.

Our video denoising network processes the video frame by frame. Before it is fed to the network, each frame is processed by a non-local patch search module which computes a non-local feature vector at each image position. A diagram of the proposed network is shown in Figure 1.

2.1 Non-local features

Let $P_{x,t}v$ be a patch centered at pixel x in frame t. The patch search module computes the distances between the patch $P_{x,t}v$ and the patches in a 3D rectangular search region $\mathcal{R}_{x,t}$ centered at (x,t) of size $w_s \times w_s \times w_t$, where w_s and w_t are the spatial and temporal sizes. The positions of these n similar patches are (x_i, t_i) (ordered according to a criterion specified later). Note that $(x_1, t_1) = (x,t)$.

The pixel values at those positions are gathered as an n-dimensional non-local feature vector

$$f^{\text{nl}}(x,t) = [v(x_1,t_1), ..., v(x_n,t_n)].$$

The image of non-local features f^{nl} is considered as a 3D tensor with n channels. This is the input to the network. Note that the first channel of the feature images corresponds to the noisy image v.

2.2 Network architecture

Our network can be divided in two stages: a non-local stage and a local stage. The non-local stage consists of four 1×1 convolution layers with 32 kernels. The rationale for these layers is to allow the network to compute pixel-wise features out of the raw non-local features f^{nl} at the input.

The second stage receives the features computed by the first stage. It consists of 14 layers with 64 3×3 convolution kernels, followed by batch normalization and ReLU activations. The output layer is a 3×3 convolution. Its architecture is similar to the DnCNN network introduced in [55], although with 15 layers instead of 17 (as in [56]). As for DnCNN, the network outputs a residual image, which has to be subtracted to the noisy image to get the denoised one. The training loss is the averaged mean square error between the residual and the noise. For RGB videos, we use the same number of layers, but triple the number of features for each layer.

3 Training and dataset

3.1 Datasets

For the training and validation sets we used a database of short segments of 16 frames extracted from YouTube

videos. Only HD videos with Creative Commons license were used. From each video we extracted several segments, separated by at least $10s$. In total the database consists of 16950 segments extracted from 1068 videos, organized in 64 categories (such as antelope, cars, factory, etc.). The segments were downscaled to have 540 lines and, when training the grayscale networks, converted to grayscale. An antialising filter was applied before downscaling. To avoid dataset biases, we randomized the filter width. We separated 6% of the videos of the database for the validation (one video for each category).

For training we ignored the first and last frames of each segment for which the 3D patch search window did not fit in the video. For grayscale networks the images were converted to grayscale, before the synthetic Gaussian noise was added.

During validation we only considered the central frame of each sequence. The resulting validation score is thus computed on 503 sequences (1 frame each). [1]

For testing we used two datasets. One of them is a set of seven sequences from the Derf's Test Media collection[2] used in [1]. This set used exactly the processing pipeline used in [1]: The original videos are RGB of size 1920×1080, and sequences of 100 frames were extracted and down-sampled by a factor two (the resolution is thus 960×540). The grayscale versions were obtained by averaging the channels. The second dataset is the `test-dev` split of the DAVIS video segmentation challenge [40]. It consists of 30 videos having between 25 and 90 frames. The videos are stored as sequences of JPEG images. There are two versions of the dataset: the full resolution (ranging between HD and 4K) and 480p. We used the full resolution set and applied our own downscaling to 540 rows. In this way we reduced the compression artifacts.

3.2 Epochs

At each training epoch a new realization of the noise is added to generate the noisy samples. To speed the training up, we pre-compute the non-local patch search on every video (after noise generation). A random set of (spatio-temporal) patches is drawn from the dataset to generate the mini-batches. We only consider patches such that the $w_s \times w_s \times w_t$ search window fits in the video (for instance, we exclude the first and last $w_t/2$ frames). At testing time, we simply extended the video by mirroring it at the start and the end of the sequence. An epoch comprised 14000 batches of size 128, composed of image patches of size 44×44. We trained for 20 epochs with Adam [27] and reduced the learning rate at epochs 12 and 17 (from $1e^{-3}$ to $1e^{-4}$ and $1e^{-6}$ respectively). Training a network took 16 hours on an NVIDIA TITAN V for grayscale videos, and 72 hours for color videos.

[1] The code to reproduce our results and the database can be found at https://github.com/axeldavy/vnlnet.

[2] https://media.xiph.org/video/derf

4 Experimental results

We will first show some experiments to highlight relevant aspects of the proposed approach. Then we compare with the state-of-the-art.

Method	No patch	Without oracle	With oracle
PSNR	31.24	31.28	31.85

Table 1: PSNR on the CBSD68 dataset (noise standard deviation of 25) for the proposed method on still images. Two variants of our method and a baseline ("No patch") are compared. "No patch" corresponds to the baseline CNN with no nearest neighbor information. The other two versions collect 9 neighbors by comparing 9×9 patches. But while the former searches them on the noisy image, the latter determines the patch position on the noise-free image (oracle). In both cases the pixel values for the non-local features are taken from the noisy image.

The untapped potential of non-locality. Although the focus of this work is in video denoising, it is still interesting to study the performance of the proposed non-local CNN on images. Figure 2 shows a comparison of a baseline CNN (a 15 layer version of DnCNN [55], as in our network) and a version of our method trained for still image denoising (it collects 9 neighbors by comparing 9×9 patches). The results with and without non-local information are very similar, this is confirmed on Table 1. The only difference is visible on very self-similar parts like the blinds that are shown in the detail of Figure 2. The average PSNR on the CBSD68 dataset [37, 55] (noise with $\sigma = 25$) obtained for the baseline CNN is of 31.24dB. The non-local CNN only leads to a 0.04dB improvement (31.28dB). The figure and table also show the result of an oracular method: the nearest neighbor search is performed on the noise-free image, though the pixel values are taken from the noisy image. The oracular results show that non-locality has a great potential to improve the results of CNNs. The oracular method obtains an average PSNR of 31.85dB, 0.6dB over the baseline. However, this improvement is hindered by the difficulty of finding accurate matches in the presence of noise. A way to reduce the matching errors is to use larger patches. But on images, larger patches have fewer similar patches. In contrast, as we will see below, the temporal redundancy of videos allows using very large patches.

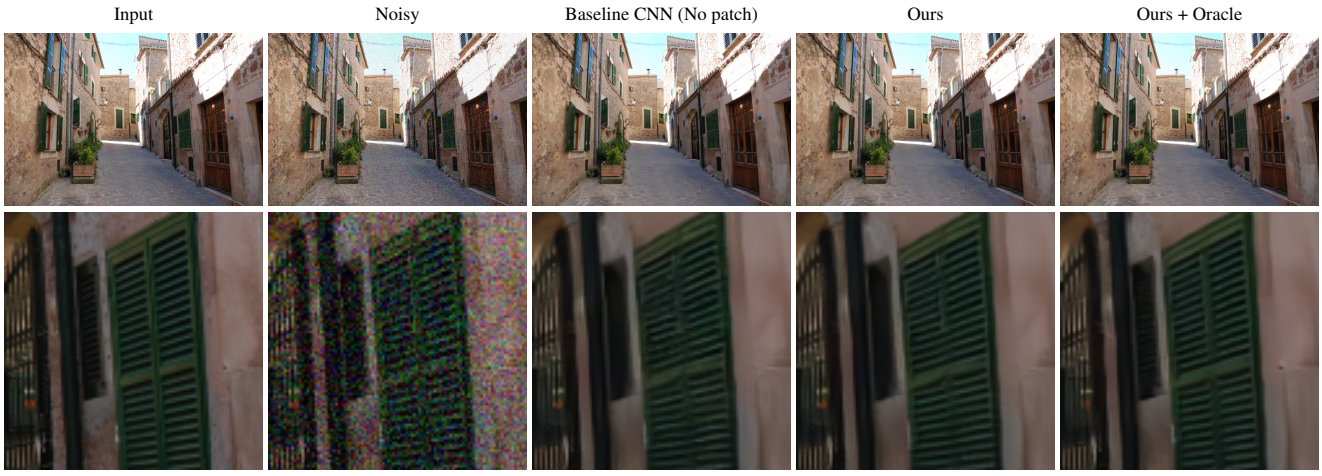

Figure 2: Results on a color image (noise standard deviation of 25). The compared methods are the ones introduced in Table 1.

4.1 Parameter tuning

Non-local search has three main parameters: The patch size, the number of retained matches and the number of frames in the search region. We expect the best matches to be past or future versions of the current patch, so we set the number of matches as the number of frames on which we search.

Patch size	no patch	9×9	15×15	21×21	31×31	41×41
PSNR	33.75	35.62	36.40	36.84	37.11	37.22

Table 2: Impact of the patch size on the PSNR computed on the validation set (noise standard deviation of 20). The tested sizes are 9×9, 15×15, 21×21, 31×31 and 41×41. No patch corresponds to the baseline simplified DnCNN.

# search frames	no patch	3	7	11	15
PSNR	33.75	35.35	36.50	36.97	37.22

Table 3: Impact of the number of frames considered in the 3D search window, on the PSNR computed on the validation set for a noise standard deviation of 20. (respectively no patch search, 3, 7, 11 and 15)

In Table 2, we explore the impact of the patch size used for the matching. Figure 3 shows visual results corresponding to each parameter. Surprisingly, we obtain better and better results by increasing the size of the patches. The main reason for this is that the match precision is improved, as the impact of noise on the patch distance shrinks. The bottom row of Figure 3 shows an area of the ground only affected by slight camera motion and on the top row an area with complex motion (a person moving his feet). We can see that the former is clearly better denoised using large patches, while the latter remains unaffected around the motion area. This indicates that the network is able to determine when the provided non-local information is not accurate and to fall back to a result similar to DnCNN in this case (single image denoising), which can be noticed on the last row of Figure 6. Further increasing the patch size would result in more areas being processed as single images. As a result, we see that the performance gain from 31×31 to 41×41 is rather small. With such large patches, only matches of the same objects from different frames are likely to be taken as neighbors. Thus we go a step further by enforcing matches to come from different frames, which improves slightly the performance. This is shown on Figure 5 and Table 4. Note the network is retrained as the patch distribution is impacted. Indeed when no restriction are imposed, the neighbors are sorted by increasing distance. While, in this variant, neighbors are sorted by frame index.

Patch search	no restriction	one neighbor per frame
PSNR	37.22	37.46

Table 4: Impact of allowing patches to be selected anywhere on the 3D search region, or having exactly one neighbor per frame. The PSNR is computed on the validation set (noise standard deviation of 20), with a patch size 41×41 and a search region of 15 frames.

In Table 3 and Figure 4, we see the impact of the number of frames used in the search window (and thus the number of nearest neighbors). One can see that the more frames, the better. Increasing the number of frames beyond 15 (7 past, current, and 7 future) does not justify the small increase of performance. Foreground moving objects are unlikely

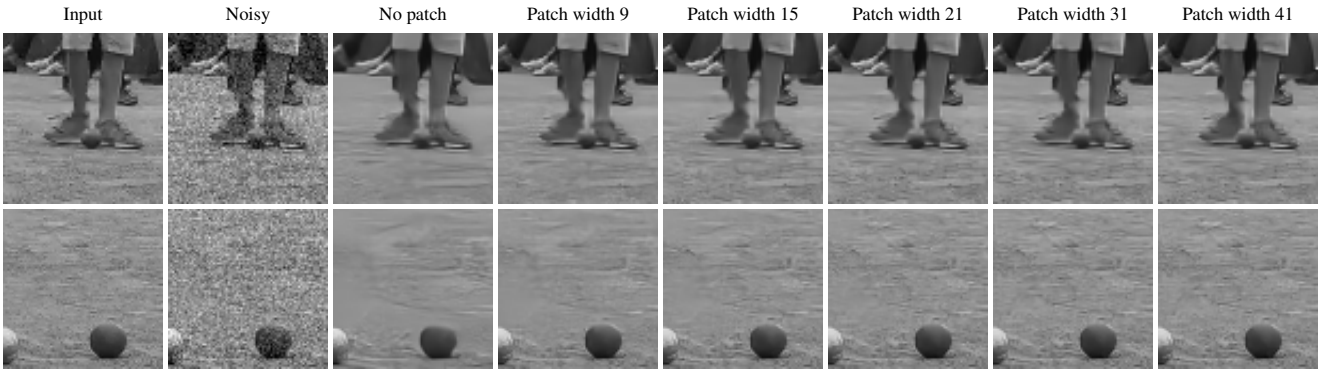

Figure 3: Example of denoised results with our method when changing the patch size, respectively no patch search, 9×9, 15×15, 21×21, 31×31 and 41×41 patches. The 3D search window has 15 frames for these experiments.

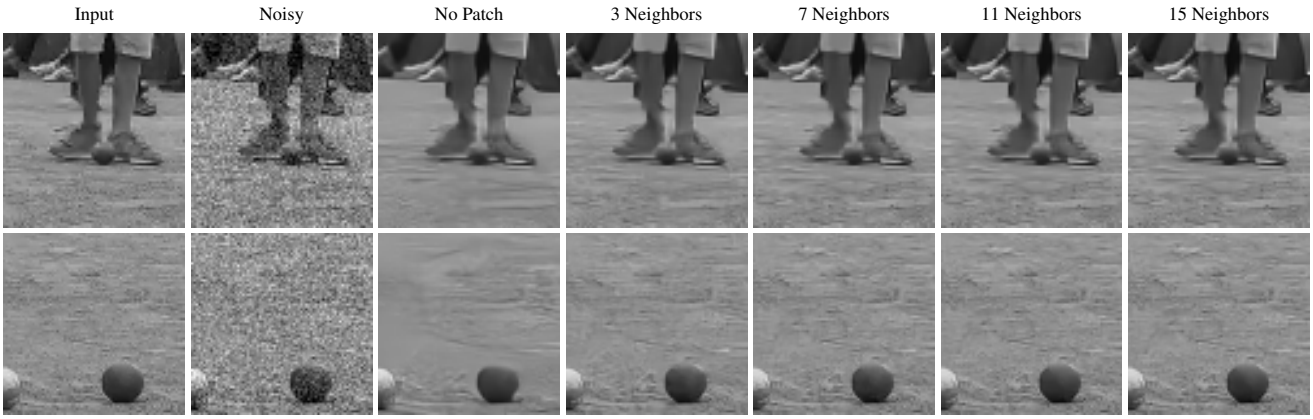

Figure 4: Example of denoised results with our method when changing the number of frames considered in the 3D search window (respectively no patch search, 3, 7, 11 and 15). 41×41 patches were used for these experiments.

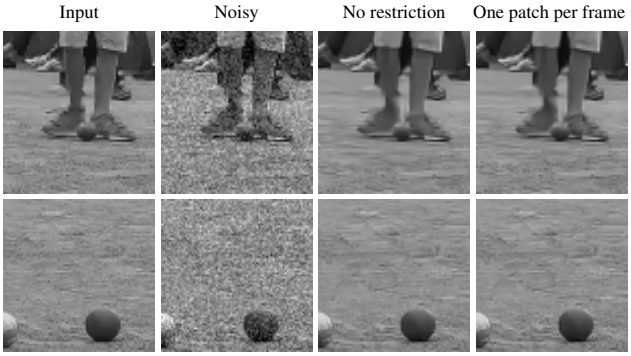

Figure 5: Example of denoised results with our method when allowing patches to be selected anywhere on the 3D search region, or when having exactly one patch neighbor per frame. 41×41 patches and a search region of 15 frames were used for these experiments.

to get good neighbors for the selected patch size, unlike background objects, thus it comes to no surprise that the visual quality of the background improves with the number of patches, while foreground moving objects (for example the legs on Figure 4) do not improve much.

In the following experiments, we shall use 41×41 patches and 15 frames. Another parameter for non-local search is the spatial width of the search window, which we set to 41 pixels (the center pixel of the tested patches must reside inside this region). We trained grayscale and color networks for AGWN of σ 10, 20 and 40. To highlight the fact that a CNN method can adapt to many noise types, unlike traditional methods, we also trained a grayscale network for Gaussian noise correlated by a 3×3 box kernel such that the final standard deviation is $\sigma = 20$, and 25% uniform Salt and Pepper noise (removed pixels are replaced by random uniform noise).

σ	Method	crowd	park joy	pedestrians	station	sunflower	touchdown	tractor	average
10	SPTWO	36.57 / .9651	35.87 / .9570	41.02 / .9725	41.24 / .9697	42.84 / .9824	40.45 / .9557	**38.92** / .9701	39.56 / .9675
	VBM3D	35.76 / .9589	35.00 / .9469	40.90 / .9674	39.14 / .9651	40.13 / .9770	39.25 / .9466	37.51 / .9575	38.24 / .9599
	VBM4D	36.05 / .9535	35.31 / .9354	40.61 / .9712	40.85 / .9466	41.88 / .9696	39.79 / .9440	37.73 / .9533	38.88 / .9534
	VNLB	**37.24** / **.9702**	**36.48** / **.9622**	**42.23** / **.9782**	**42.14** / **.9771**	**43.70** / **.9850**	**41.23** / **.9615**	**40.20** / **.9773**	**40.57** / **.9731**
	DnCNN	34.39 / .9455	33.82 / .9329	39.46 / .9641	37.89 / .9412	40.20 / .9702	38.28 / .9269	36.91 / .9568	37.28 / .9482
	VNLnet	**37.00** / **.9727**	**36.39** / **.9665**	**41.96** / **.9779**	**42.44** / **.9766**	**43.76** / **.9861**	**41.05** / **.9609**	38.89 / **.9718**	**40.21** / **.9732**
20	SPTWO	32.94 / .9319	32.35 / .9161	37.01 / .9391	38.09 / **.9461**	38.83 / .9593	**37.55** / **.9287**	35.15 / .9363	35.99 / .9368
	VBM3D	32.34 / .9093	31.50 / .8731	37.06 / .9423	35.91 / .9007	36.25 / .9393	36.17 / .9065	33.53 / .8991	34.68 / .9100
	VBM4D	32.40 / .9126	31.60 / .8832	36.72 / .9344	36.84 / .9224	37.78 / .9517	36.44 / .9034	33.95 / .9104	35.10 / .9169
	VNLB	**33.49** / **.9335**	**32.80** / **.9154**	**38.61** / **.9583**	**38.78** / **.9470**	**39.82** / **.9698**	**37.47** / **.9220**	**36.67** / **.9536**	**36.81** / **.9428**
	DnCNN	30.47 / .8890	30.03 / .8625	35.81 / .9302	34.37 / .8832	36.19 / .9361	35.35 / .8782	32.99 / .9019	33.60 / .8973
	VNLnet	**33.40** / **.9415**	**32.84** / **.9271**	**38.32** / **.9565**	**38.49** / .9454	**39.88** / **.9700**	37.11 / .9102	**35.23** / **.9390**	**36.47** / **.9414**
40	SPTWO	29.02 / .8095	28.79 / .8022	31.32 / .7705	32.37 / .7922	32.61 / .7974	31.80 / .7364	30.61 / .8223	30.93 / .7901
	VBM3D	28.73 / .8295	27.93 / .7663	33.00 / .8828	32.57 / .8239	32.39 / .8831	33.38 / **.8624**	29.80 / .8039	31.11 / .8360
	VBM4D	28.72 / .8339	27.99 / .7751	32.62 / .8683	32.93 / .8441	33.66 / .8999	33.68 / .8603	30.20 / .8205	31.40 / .8432
	VNLB	**29.88** / **.8682**	**29.28** / **.8309**	**34.68** / **.9167**	**34.65** / **.8871**	**35.44** / **.9329**	**34.18** / **.8712**	**32.58** / **.8921**	**32.95** / **.8856**
	DnCNN	26.85 / .7979	26.65 / .7525	32.01 / .8660	30.96 / .7899	32.13 / .8705	32.78 / .8346	29.25 / .7976	30.09 / .8156
	VNLnet	**29.69** / **.8727**	**29.29** / **.8397**	**34.21** / **.9089**	**33.96** / **.8686**	**35.12** / **.9224**	**33.88** / .8495	**31.41** / **.8647**	**32.51** / **.8752**

Table 5: Quantitative denoising results (PSNR and SSIM) for seven grayscale test sequences of size 960×540 from the *Derf's Test Media collection* on several state-of-the-art video denoising algorithms versus DnCNN and our method. Three noise standard deviations σ are tested (10, 20 and 40). Compared methods are SPTWO [9], VBM3D [16], VBM4D [34], VNLB [2], DnCNN [55] and VNLnet (ours). We highlighted the best performance in black and the second best in brown.

σ	Method	crowd	park joy	pedestrians	station	sunflower	touchdown	tractor	average
10	VBM3D	36.03 / .9625	35.01 / .9451	41.19 / .9738	38.53 / .9463	39.58 / .9599	39.91 / .9486	37.10 / .9555	38.19 / .9560
	VNLB	**38.33** / **.9773**	**37.09** / **.9708**	**42.77** / **.9800**	**42.83** / **.9784**	**43.23** / **.9820**	**42.16** / **.9677**	**40.07** / **.9760**	**40.93** / **.9760**
	DnCNN	35.41 / .9576	34.37 / .9454	40.26 / .9701	38.73 / .9536	40.10 / .9675	39.77 / .9485	37.37 / .9600	38.00 / .9575
	VNLnet	**37.74** / **.9758**	**36.63** / **.9690**	**42.56** / **.9805**	**42.22** / **.9765**	**43.12** / **.9819**	**42.26** / **.9705**	**38.90** / **.9699**	**40.49** / **.9749**
20	VBM3D	32.54 / .9284	31.58 / .8930	37.73 / .9505	35.28 / .8962	36.01 / .9250	36.89 / .9091	33.56 / .9131	34.80 / .9165
	VNLB	**34.78** / **.9529**	**33.53** / **.9365**	**39.63** / **.9640**	**39.67** / **.9565**	**39.84** / **.9667**	**38.80** / **.9353**	**37.08** / **.9575**	**37.62** / **.9528**
	DnCNN	31.49 / .9144	30.62 / .8875	36.92 / .9459	35.44 / .9094	36.24 / .9373	36.70 / .9064	33.65 / .9190	34.44 / .9171
	VNLnet	**34.45** / **.9556**	**33.40** / **.9407**	**39.63** / **.9670**	**39.57** / **.9592**	**40.06** / **.9695**	**39.30** / **.9465**	**35.78** / **.9456**	**37.46** / **.9549**
40	VBM3D	29.23 / .8688	28.43 / .8152	34.11 / .9056	32.45 / .8255	32.82 / .8792	34.17 / .8604	30.30 / .8431	31.65 / .8568
	VNLB	**31.24** / **.9052**	**30.23** / **.8730**	**35.97** / **.9291**	**35.88** / **.9074**	**35.77** / **.9285**	**35.19** / **.8719**	**33.47** / **.9140**	**33.97** / **.9042**
	DnCNN	27.99 / .8429	27.50 / .7998	33.50 / .9001	32.29 / .8350	32.58 / .8845	33.94 / .8566	30.17 / .8454	31.14 / .8520
	VNLnet	**31.13** / **.9144**	**30.23** / **.8848**	**36.19** / **.9390**	**36.12** / **.9172**	**36.36** / **.9411**	**35.64** / **.8872**	**32.44** / **.8985**	**34.02** / **.9117**

Table 6: Quantitative denoising results (PSNR and SSIM) for seven color test sequences of size 960×540 from the *Derf's Test Media collection* on several state-of-the-art video denoising algorithms versus DnCNN and our method. Three noise standard deviations σ are tested (10, 20 and 40). Compared methods are VBM3D [16], VNLB [2], DnCNN [55] and VNLnet (ours). We highlighted the best performance in black and the second best in brown.

Method	Corr. Gaussian noise	Uniform S&P 25%
VNLB	**25.39** / **.5922**	**23.49** / **.7264**
VNLnet	**30.94** / **.9452**	**48.12** / **.9951**

Table 7: Performance (PSNR and SSIM) of VNLB and VNLnet (our method) on the grayscale DERF dataset (Table 5) for non-standard noises.

4.2 Comparison with state-of-the-art

On tables 5 and 6, we show a comparison of DnCNN (applied frame-by-frame) and the proposed method *Video Non-Local Network* (VNLnet) with other state-of-the-art video denoising methods [1] for the DERF dataset. The state-of-the-art methods include SPTWO [9], VBM3D [16], VBM4D [34] and VNLB [2]. Figures 6, 9 and 8 show results for some of the most relevant methods.

DnCNN, VBM3D and VNLB were also compared on

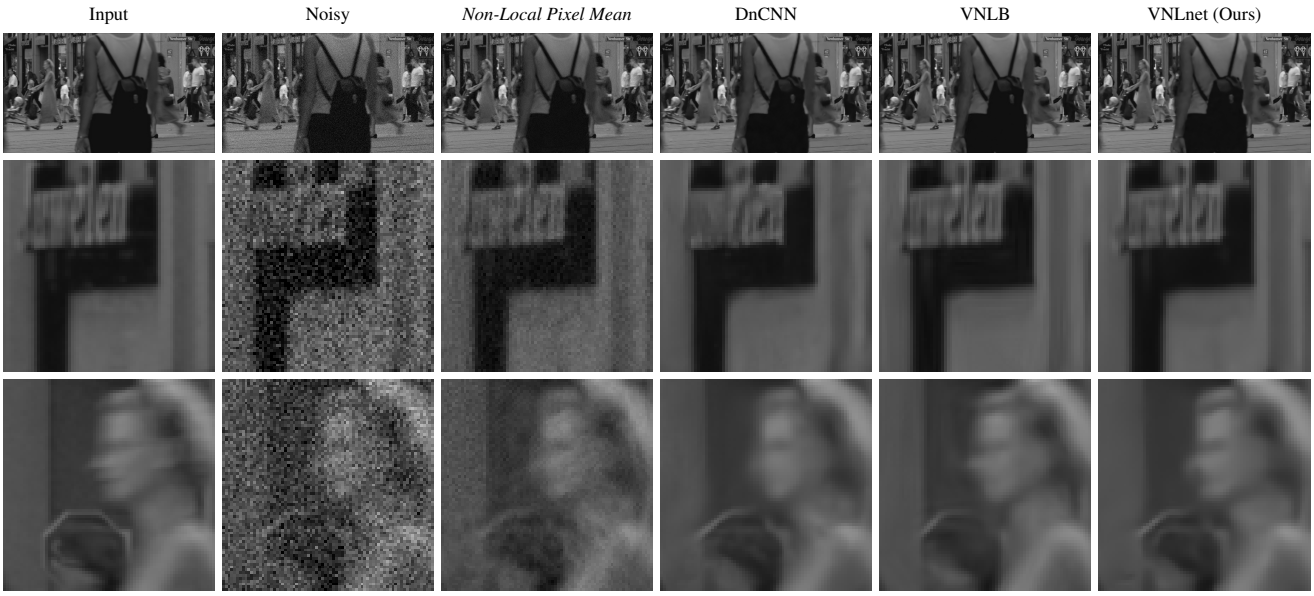

Figure 6: Example of denoised result for several algorithms (noise standard deviation of 20). The two crops highlight the results on a non-moving and a moving part of the video. *Non-Local Pixel Mean* corresponds to the average of the output of the non-local search layer.

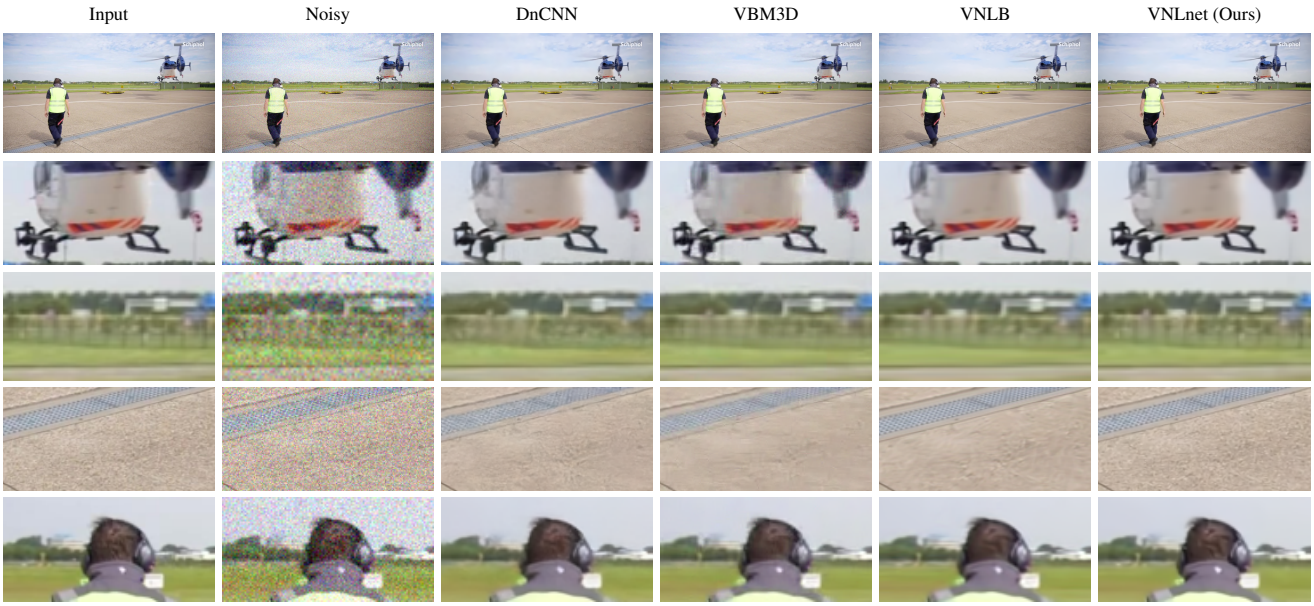

Figure 7: Example of denoised result for several algorithms (noise standard deviation of 20) on a sequence of the color DAVIS dataset [40]. The crops highlight the results on non-moving and moving parts of the video.

the DAVIS `test-dev` dataset [40]. Results are shown in Tables 8, 9 and Figure 7. VNLnet is the best performing method in the DAVIS dataset, but it is outperformed by VNLB (Video Non-Local Bayes) on the DERF dataset. This could be explained by the fact that the DERF dataset is blurrier compared to DAVIS and to VNLnet's training set. This blur is caused by a wider antialiasing filter used when downscaling the original videos in [1]. We notice DnCNN also underperforms on this dataset compared to the sharper DAVIS dataset.

A comparison of tables 8 and 9 reveals that CNN based methods are better in exploiting the correlations between

Method	$\sigma = 10$	$\sigma = 20$	$\sigma = 40$
VBM3D	37.43 / .9425	33.75 / .8870	30.12 / .8068
VNLB	**38.84 / .9634**	**35.26 / .9240**	**31.88 / .8622**
DnCNN	36.80 / .9451	32.94 / .8878	28.69 / .7940
VNLnet	**39.07 / .9663**	**35.46 / .9299**	**31.90 / .8659**

Table 8: Performance (PSNR and SSIM) of DnCNN, VBM3D and VNLnet (our method) on the grayscale DAVIS dataset [40] for several noise levels σ (10, 20 and 40).

Method	$\sigma = 10$	$\sigma = 20$	$\sigma = 40$
VBM3D	38.43 / .9591	34.74 / .9157	31.38 / .8473
VNLB	**40.31 / .9725**	**36.79 / .9420**	**33.34 / .8896**
DnCNN	38.91 / .9655	35.24 / .9278	31.81 / .8637
VNLnet	**40.71 / .9760**	**37.39 / .9534**	**33.96 / .9091**

Table 9: Performance (PSNR and SSIM) of DnCNN, VBM3D and VNLnet (our method) on the color DAVIS dataset [40] for several noise levels σ (10, 20 and 40).

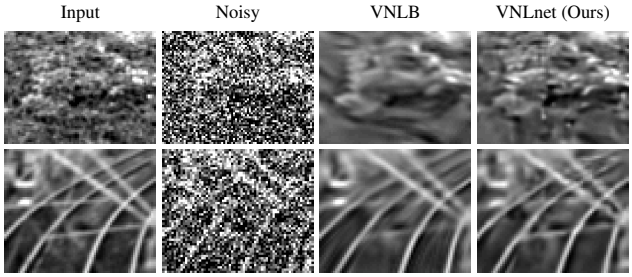

Figure 8: Examples of background details more properly recovered by VNLnet compared to VNLB (enhanced contrast).

color channels: while for grayscale, VBM3D was significantly outperforming DnCNN in PSNR on the DAVIS dataset, the reverse occurs for color. In addition, the gap between VNLnet and VNLB is widened in color. This should not come as a surprise, since the way in which VBM3D and VNLB treat color is rather heuristic: an orthogonal color transformed is applied to the video which is supposed to decorrelate color information. Based on this, the processing of each color channel of a group of patches in done independently. These results make VNLnet the state-of-the-art method for video denoising and the first so of the neural kind.

In order to better compare qualitative aspects of the results we show some details in Figures 6, 9 and 8. Figure 6 shows some results on an video of DERF. The two bottom rows show details on two different types of areas (background and moving object). We include as reference the *Non-Local Pixel Mean*, which is just the result of the averaging of the matches presented to the network. As noise remains, one can thus see that the network does more than averaging the data on static areas (last two rows). On background objects, the denoising performance is significantly improved compared to DnCNN and is similar to VNLB (middle row). For some scenes, VNLnet recovers significantly more details in the background, as shown on Figure 8 and Figure 9. On moving foreground objects - thus with bad matches - our method performs similar to DnCNN, as can be observed on the last row of Figure 6. In general, we observe VNLnet has better background reconstruction than VNLB. Both methods achieve high temporal consistency, which is an important quality requirement for video denoising.

One of the benefits of CNNs over traditional model-based approaches is that they can be easily retargeted to handle other noise models. To illustrate this we compare the classical best performing method VNLB and our method on non-standard noises on Table 7. As expected, VNLnet significantly beats VNLB for these non-Gaussian noise distributions.

VBM3D	DnCNN	VBM4D	VNLB	SPTWO
1.3s	13s	52s	140s	210s

Table 10: Running time per frame on a 960×540 video for VBM3D, DnCNN, VBM4D, VNLB and SPTWO on single CPU core.

Non-local search	Rest of the network	DnCNN
850 ms	80 ms	95 ms

Table 11: Running time per frame on a 960×540 video on a NVIDIA TITAN V (41×41 patches at every position, $41 \times 41 \times 15$ 3D windows, the default parameters).

A note on running times. On Table 10, we compare the CPU running time of VBM3D, DnCNN and VNLB when denoising a video frame. While we do not a have a CPU implementation of the patch search layer, the GPU runtimes of Table 11 point out that on CPU our method should be 10 times slower than DnCNN. The non-local search is particularly costly because we search matches on 15 frames for patches centered in every pixel of our image. The patch search could be made significantly faster by reducing the size of the 3D window using tricks explored in other papers. VBM3D for example centers the search on each frame on small windows around the best matches found in the previous frame. A related acceleration is to use a search strategy based on PatchMatch [6].

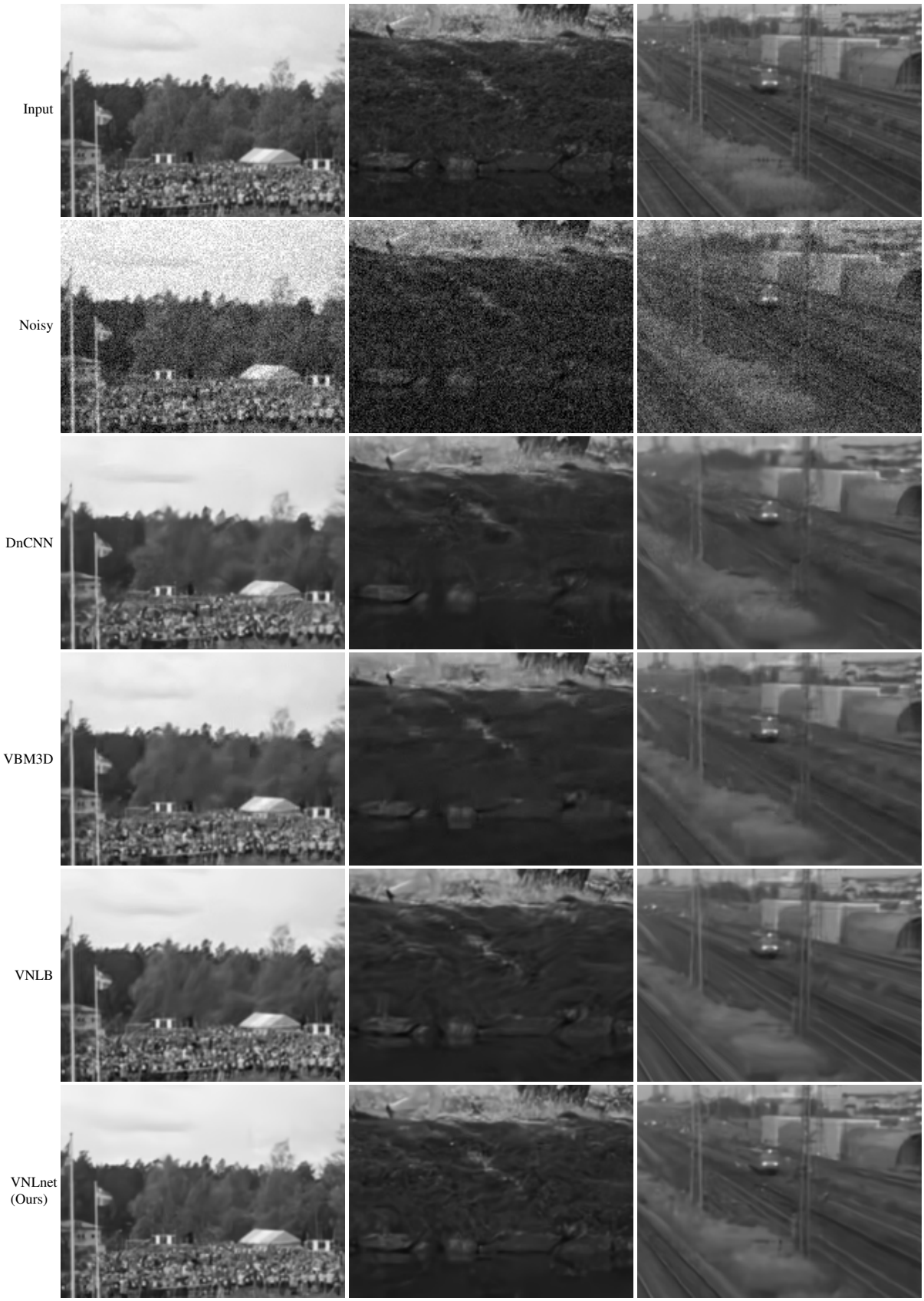

Figure 9: Examples of areas where the level of restored detail of the methods differs significantly (noise standard deviation of 20) on *crowd, park* and *station*.

5 Implementation details

The patch search requires the computation of the distance between each patch in the image and the patches in the search region. If implemented naïvely, this operation can be prohibitive. Patch-based methods require a patch search step. To reduce the computational cost, a common approach is to search the nearest neighbors only for the patches in a subgrid of the image. For example BM3D processes 1/9th of the patches with default parameters. Since the processed patches overlap, the aggregation of the denoised patches covers the whole image.

Our proposed method does not have any aggregation. We compute the neighbors for all image patches, which is costly. In the case of video, best results are obtained with large patches and a large search region (both temporally and spatially). Therefore we need a highly efficient patch search algorithm.

Our implementation uses an optimized GPU kernel which searches for the locations in parallel. For each patch, the best distances with respect to all other patches in the search volume are maintained in a table. We split the computation of the distances is two steps: first compute the sum of squares across columns:

$$D^{\text{col}}(x', y', t') = \sum_{h=-s/2}^{s/2} (v(x, y+h, t) - v(x', y'+h, t')^2.$$

Then the distances can be obtained by applying a horizontal box filter of size s on the volume D^{col} composed by the neighboring GPU threads. The resulting implementation has linear complexity in the size of the search region and the patch width.

To optimize the speed of the algorithm we use the GPU shared memory as cache for the memory accesses thus reducing bandwidth limitations. In addition, for sorting the distances the ordered table is stored into GPU registers, and written to memory only at the end of the computation. The computation of the L2 distances and the maintenance of the ordered table have about the same order of computation cost. More details about the implementation can be found in Appendix A.

6 Conclusions

We described a simple yet effective way of incorporating non-local information into a CNN for video denoising. The proposed method computes for each image patch the n most similar neighbors on a spatio-temporal window and gathers the value of the central pixel of each similar patch to form a non-local feature vector which is given to a CNN. Our method yields a significant gain compared to using the single frame baseline CNN on each video frame. Compared to other video denoising algorithms, it achieves state-of-the-art performance and attaining the highest PSNR on a downscaled version of the DAVIS dataset.

Our contribution places neural networks among the best video denoising methods and opens the way for new works in this area.

We have seen the importance of having reliable matches: On the validation set, the best performing method used patches of size 41×41 for the patch search. We have also noticed that on regions with non-reliable matches (complex motion), the network reverts to a result similar to single image denoising. Thus we believe future works should focus on improving this area, by possibly adapting the size of the patch and passing information about the quality of the matches to the network.

A More implementation details on the non-local search

In this section we describe in more details our GPU implementation of the non-local search.

As mentioned in the main paper, a naïve implementation of the patch search can be very inefficient.

The patch search algorithm can be divided conceptually into two parts: First, computing for all positions in the search window the L2 distance between the reference patch and the target patch, both of size $K \times K$. Second, retaining the best N distances and positions. Both parts need to be implemented efficiently.

Algorithm 1: Keeping an ordered table of distances and positions

```
Input: New position p and distance d
Input: Tables Positions and Distances of length N.
if d < Distances[N-1] then
    for i from N-1 to 1 do
        insert ← Distances[i-1] ≤ d
        Positions[i] ← p if insert else Positions[i-1]
        Distances[i] ← d if insert else Distances[i-1]
        quit function if insert
    end
    Positions[0] ← p
    Distances[0] ← d
end
```

Of the two parts, the most critical one is the maintenance of the table of the best N distances and positions. Our implementation maintains the distances and positions in an ordered table stored into the GPU registers. Indeed GPUs have many available registers: a maximum of 128 on

Intel, and 256 on AMD and NVIDIA for the current generation, although consuming fewer registers helps reducing the latency. Thus if N is small enough, both tables (distances and positions) can be stored in the register table. The tables need to be accessed very frequently, so not using registers leads to a much slower code. Our algorithm is summarized on Algorithm 1.

Algorithm 2: Summary of our patch search implementation

```
On CPU: Divide the image into regions of
  overlapping row segments of length 128. The
  overlap should be of length patch_width - 1.
On GPU:
Assign one thread to each element of an horizontal
  segment.
for each offset (dz, dy, dx) in the 3D search window
 do
    Compute squared L2 distance between reference
      and target center columns
    Write result into GPU shared memory
    Sum results of neighboring threads
    Maintain table of best distances and positions
end
Only save valid results (border threads can't
  compute the full distance)
```

Then comes the computation of the L2 distances between patches. A naïve algorithm would, for every patch pair, read the reference patch, read the target patch, and compare them pixel-wise, without reusing any computation or memory access. Our optimized algorithm uses the fact that the L2 distances computation share a lot of common elements with the same computations after a translation of the positions of the reference and the target patches. This avoids both computation and memory accesses. We organize GPU threads into groups treating an horizontal segment each. Each thread will compute the distance between the reference and the target patch for the center column only, and shared GPU memory will be used to read the results of neighboring threads and compute the full distance. Since we only need to compare a column of the patches, that column can be stored into GPU registers, thus avoiding to reload the reference patch data every iteration. Threads at the border of the segments can't compute the full distance as some results are missing, thus some overlap between the segments are required. We found a length of 128 to be a good compromise for the length of these segments. For increased speed, we cache our memory accesses with the GPU shared memory between the computing threads. The process is summarized in Algorithm 2.

Both our implementation and the naïve implementation have a linear complexity in the size of the 3D search window, but our algorithm has a linear complexity with respect to the width of the patches, while it is quadratic for the naïve algorithm. One should know, if not familiar with patch search, that the 3D search window defines the search region for all patches whose center lie inside the 3D search window, thus the patches do not have to fit completely inside the region. For the default VNLnet parameters, our implementation is 25 times faster (on a NVIDIA TITAN V) than the naïve implementation (both using Algorithm 1 for the tables of distances and positions).

References

[1] P. Arias, G. Facciolo, and J.-M. Morel. A comparison of patch-based models in video denoising. In *2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pages 1–5. IEEE, 2018. 4, 7, 8

[2] P. Arias and J.-M. Morel. Towards a bayesian video denoising method. In *Advanced Concepts for Intelligent Vision Systems*, LNCS. Springer, 2015. 2, 7

[3] P. Arias and J.-M. Morel. Video denoising via empirical bayesian estimation of space-time patches. *Journal of Mathematical Imaging and Vision*, 60(1):70–93, Jan 2018. 2

[4] P. Arias and J.-M. Morel. Kalman filtering of patches for frame-recursive video denoising. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 2

[5] A. Barbu. Training an active random field for real-time image denoising. *IEEE Transactions on Image Processing*, 18(11):2451–2462, Nov 2009. 1

[6] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch. In *ACM SIGGRAPH 2009 papers on - SIGGRAPH '09*, page 1, New York, New York, USA, 2009. ACM Press. 9

[7] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. *Computer Vision and Pattern*, 2(0):60–65, 2005. 1

[8] A. Buades and J. L. Lisani. Dual domain video denoising with optical flow estimation. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2986–2990, Sep. 2017. 2

[9] A. Buades, J.-L. Lisani, and M. Miladinović. Patch-based video denoising with optical flow estimation. *IEEE Transactions on Image Processing*, 25(6):2573–2586, June 2016. 2, 7

[10] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2392–2399, June 2012. 1, 2

[11] V. Caselles, A. Chambolle, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. *Theoretical Foundations and Numerical Methods for Sparse Recovery, De Gruyter, Radon Series Comp. Appl. Math.*, 9:263–340, 2010. 1

[12] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to see in the dark. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[13] X. Chen, L. Song, and X. Yang. Deep rnns for video denoising. In *Applications of Digital Image Processing*, 2016. 2

[14] Y. Chen and T. Pock. Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 6 2017. 1, 2

[15] C. Cruz, A. Foi, V. Katkovnik, and K. Egiazarian. Nonlocality-reinforced convolutional neural networks for image denoising. *IEEE Signal Processing Letters*, 25(8):1216–1220, Aug 2018. 2

[16] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3D transform-domain collaborative filtering. In *EUSIPCO*, 2007. 2, 7

[17] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 2007. 1, 2

[18] D. L. Donoho and J. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994. 1

[19] T. Ehret, P. Arias, and J.-M. Morel. Global patch search boosts video denoising. In *International Conference on Computer Vision Theory and Applications*, 2017. 2

[20] T. Ehret, J. Morel, and P. Arias. Non-Local Kalman: A recursive video denoising algorithm. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3204–3208, Oct 2018. 2

[21] G. Facciolo, N. Pierazzo, and J. Morel. Conservative scale recomposition for multiscale denoising (the devil is in the high frequency detail). *SIAM Journal on Imaging Sciences*, 10(3):1603–1626, 2017. 1

[22] C. Godard, K. Matzen, and M. Uyttendaele. Deep burst denoising. *arXiv*, 2017. 2

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. 1

[24] Y. Huang, W. Wang, and L. Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 235–243. Curran Associates, Inc., 2015. 2

[25] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015. 2

[26] V. Jain and S. Seung. Natural image denoising with convolutional networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 769–776. Curran Associates, Inc., 2009. 1

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[28] E. Kobler, T. Klatzer, K. Hammernik, and T. Pock. Variational networks: Connecting variational methods and deep learning. In V. Roth and T. Vetter, editors, *Pattern Recognition*, pages 281–293, Cham, 2017. Springer International Publishing. 1

[29] M. Lebrun, A. Buades, and J.-M. Morel. A nonlocal bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 2013. 2

[30] M. Lebrun, M. Colom, A. Buades, and J.-M. Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21(1):475–576, 2012. 1

[31] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5882–5891, July 2017. 2

[32] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang. Non-local recurrent network for image restoration. In *Advances in Neural Information Processing Systems*, pages 1680–1689, 2018. 2

[33] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4473–4481, Oct 2017. 2

[34] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian. Video Denoising Using Separable 4D Nonlocal Spatiotemporal Transforms. In *Proc. of SPIE*, 2011. 7

[35] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian. Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms. *IEEE Transactions on Image Processing*, 2012. 2

[36] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2802–2810. Curran Associates, Inc., 2016. 2

[37] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*. IEEE, 2001. 4

[38] B. Mildenhall, J. T. Barron, J. Chen, D. Sharlet, R. Ng, and R. Carroll. Burst denoising with kernel prediction networks. In *CVPR*, 2018. 2

[39] T. Plötz and S. Roth. Neural nearest neighbors networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 1095–1106, USA, 2018. Curran Associates Inc. 2

[40] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 4, 8, 9

[41] P. Qiao, Y. Dou, W. Feng, R. Li, and Y. Chen. Learning non-local image diffusion for image denoising. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, pages 1847–1855, New York, NY, USA, 2017. ACM. 2

[42] Y. Romano, M. Elad, and P. Milanfar. The little engine that could: Regularization by denoising (red). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017. 1

[43] S. Roth and M. J. Black. Fields of experts: a framework for learning image priors. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 860–867 vol. 2, June 2005. 1

[44] S. Roth and M. J. Black. Fields of experts: a framework for learning image priors. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 860–867 vol. 2, June 2005. 1

[45] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, Nov. 1992. 1

[46] M. S. M. Sajjadi, R. Vemulapalli, and M. Brown. Frame-Recurrent Video Super-Resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[47] V. Santhanam, V. I. Morariu, and L. S. Davis. Generalized deep image to image regression. *CoRR*, abs/1612.03268, 2016. 2

[48] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2781, June 2014. 1

[49] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang. Deep video deblurring for hand-held cameras. In *IEEE CVPR*, 2017. 2

[50] J. Sun and M. F. Tappen. Learning non-local range markov random field for image restoration. In *CVPR 2011*, pages 2745–2752, June 2011. 2

[51] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4549–4557, 2017. 2

[52] R. Vemulapalli, O. Tuzel, and M. Liu. Deep gaussian conditional random field network: A model-based deep network for discriminative denoising. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4801–4809, June 2016. 1

[53] B. Wen, Y. Li, L. Pfister, and Y. Bresler. Joint adaptive sparsity and low-rankness on the fly: an online tensor reconstruction scheme for video denoising. In *IEEE ICCV*, 2017. 2

[54] D. Yang and J. Sun. Bm3d-net: A convolutional neural network for transform-domain collaborative filtering. *IEEE Signal Processing Letters*, 25(1):55–59, Jan 2018. 2

[55] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 7 2017. 2, 3, 4, 7

[56] K. Zhang, W. Zuo, and L. Zhang. FFDNet: Toward a Fast and Flexible Solution for {CNN} based Image Denoising. *CoRR*, abs/1710.0, 2017. 2, 3

[57] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486, Nov 2011. 1