Non-Local Video Denoising by CNN

Axel Davy Thibaud Ehret Gabriele Facciolo
Jean-Michel Morel Pablo Arias
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 best ones 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, specially in video denoising, where it outperforms most state-of-the-art methods. 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 [31]. Hence the need for video denoising algorithms with a low running time.

Literature review on image denoising. Image denoising has a vast literature where very varied methods have been used: PDEs and variational methods (including MRF models), transform domain methods, non-local (or patch-based) methods, etc. 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 [32], such as [4, 35, 10]. The architecture of [39] is based on EPPL [44], 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 [10] even layers must be an image (i.e. have only one feature). As pointed out in [21] these architectures have strong similarities with the residual networks of [16].

The black-box approaches treat the denoising problem as a standard regression problem. They don’t 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 [19] proposed a five layer CNN with $5 	imes 5$ filters, with 24 features in the hidden layers and sigmoid activation functions. Burger et al. [7] reported the first state-of-the-art results with a multilayer perceptron trained to denoise $17 \times 17$ patches, but...
with a heavy architecture. More recently, DnCNN [42] obtained impressive results with a far lighter 17 layer deep CNN with $3 \times 3$ convolutions, ReLU activations and batch normalization [18]. 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 [43], 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 [27, 34, 8] use pulling or strided convolutions to downscale the image, and then up-convolutional layers to upscale it back. Skip connections connect the layers before the pulling with the output of the up-convolution to avoid loss of spatial resolution. Skip connections are used extensively in [38].

Although these architectures produce very good results, for textures formed by repetitive patterns, non-local patch-based methods still perform better [42, 7]. Some works have therefore attempted to incorporate the non-local patch similarity into a CNN framework. Qiao et al. [30] proposed inference networks derived from the non-local FoE MRF model [37]. This can be seen as a non-local version of the TRDN network of [10]. A different non-local TRDN was introduced by [25]. BM3D-net [41] 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 [11] 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.

In summary, existing non-local CNNs are either trainable versions of previous non-local approaches (such as BM3D or the non-local FoE model) or iterative meta denoisers that apply a non-local filtering separately from the denoising network.

**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 is made possible by a GPU implementation of the patch search that allows computing the nearest neighbors efficiently. 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 video denoising. The non-locality 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. Compared to other works incorporating non-locality to neural networks on images, our proposal doesn’t have the limitations of the ones derived from variational/MRF models.

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 \((x_i, t_i)\). Note that \((x_1, t_1) = (x, t)\).

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

\[
f_{nl}(x, t) = [v(x_1, t_1), ..., v(x_n, t_n)].
\]

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 consists of two stages: a non-local stage and a local stage. The non-local stage consists of four \( 1 \times 1 \) 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 \( 64 \ 3 \times 3 \) convolution kernels, followed by batch normalization and ReLU activations. The output layer is a \( 3 \times 3 \) convolution. Its architecture is similar to the DnCNN network introduced in [42], although with 15 layers instead of 17 (as in [43]). 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.

3. Training and dataset

3.1. Datasets

For the training and validation sets we used a database of short segments of YouTube videos. The videos were selected by searching for 64 keywords. Only HD videos with Creative Commons license were used. From each video segments of 16 frames were extracted and downscaled to have 540 lines (typically \( 960 \times 540 \), but the number of columns depends on the aspect ratio). The separations between segments is at least 10s. In total the database consists of around 1120 extracts with 16 frames. We separated 6.5% of the videos of the database for the validation (one for each category).

For training we ignored the first and last frames of each segment for which the 3D patch search window couldn’t fit in the video. 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]. We used this set for comparing with previous denoising methods. The sequences have a resolution of \( 960 \times 540 \) with 100 frames. The original videos are RGB of size \( 1920 \times 1080 \), and were converted to grayscale by averaging the channels, and then down-sampled by a factor two.

The second dataset is the test-dev split of the DAVIS video segmentation challenge [29]. 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 artifacts caused by JPEG compression.

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 padding with black frames at the start and the end of the sequence. An epoch was composed of 14000 batches of size 128, composed of image patches of

---

\(^1\)The code to reproduce our results and the database is available at https://github.com/axeldavy/vnlnet.

\(^2\)https://media.xiph.org/video/derf
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 layer of the network perform $1 \times 1$ convolutions with 32 feature maps. The resulting feature maps are the input of a simplified DnCNN [42] network with 15 layers.

size $44 \times 44$. We trained for 20 epochs with Adam [20] and reduced the learning rate at epochs 12 and 17 (from $1e^{-3}$ to $1e^{-4}$ and $1e^{-5}$ respectively).

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 a simplified DnCNN with no nearest neighbor information. The other two versions collect 9 neighbors by comparing $9 \times 9$ 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 are taken on 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 simplified DnCNN (A standard DnCNN but with 15 layers, as in our network) and our method on a static image. 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 figure and table also show the result of an oracular method: the nearest neighbor search is performed on the noisy 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, yielding an improvement of 0.6dB. However, this improvement is hindered by the difficulty of finding accurate matches in the presence of noise. A standard 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.

4.1. Parameter tuning

Non-local search has three main parameters: The patch size, the number of retained matches and the number of frames on which we search them. Intuitively, we want the matches to be past or future positions of the current patch. Thus we set the number of matches to be the number of frames on which we search.

| Patch Width | No patch | 9    | 15   | 21   | 31   | 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 $9 \times 9$, $15 \times 15$, $21 \times 21$, $31 \times 31$ and $41 \times 41$. No patch corresponds to the baseline simplified DnCNN.

| Num Neighbors | No patch | 3    | 7    | 11   | 15   |
|---------------|----------|------|------|------|------|
| PSNR          | 33.75    | 35.35| 36.50| 36.97| 37.22|

Table 3. Impact of the depth of the 3D patch search window, i.e., the number of frames considered in the search, 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 6 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 6 shows an area of the ground only affected
by slight camera motion and on the top row an area with complex motion (human movement). We can see that the former is clearly better denoised using large patches, while the latter remains unaffected around the motion. 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 (obtained by single image denoising). 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 $31 \times 31$ to $41 \times 41$ is rather small.

In Table 3 and Figure 3, we see the impact of the number of frames used. One can see that the more frames, the better. Increasing the number of frames beyond 15 (7 past, current, and 7 future) doesn’t justify the small increase of performance.

In the following experiments, we shall use $41 \times 41$ patches and 15 frames. Another parameter for non-local search is the spatial width of the search window, which we set to 41 pixels. We trained three networks for Gaussian noise of standard deviation 10, 20 and 40.

### 4.2. Comparison with state-of-the-art

In Table 4, we show a comparison of DnCNN and the proposed method Video Non-Local Network (VNLNet) with other state-of-the-art video denoising methods [1]. The state-of-the-art methods include SPTWO [6], VBM3D [12], VBM4D [25] and VNLB [2]. We note that overall VNLB (Video Non-Local Bayes) is the best performing method. Yet, as we shall see later, it comes at a high computational cost. Our method comes second and beats the other methods, which makes it a state-of-the-art method for video denoising and the first so of the neural kind. Figure 5 shows in detail the results for the most relevant methods (VBM3D being the most popular method).

In Figure 6, we show a detail of a sequence to highlight the result on two different types of areas. 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 (middle row). Our result has more details than DnCNN and is on par visually...
Table 5. Performance (PSNR) of DnCNN, VBM3D and VNLnet (our method) on the DAVIS test-dev dataset [29] for several noise levels $\sigma$ (10, 20 and 40).

Table 4. Quantitative denoising results (PSNR and SSIM) for seven grayscale test sequences of size $960 \times 540$ 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 $\sigma$ are tested (10, 20 and 40). Compared methods are SPTWO [6], VBM3D [12], VBM4D [25], VNLB [2], DnCNN [42] and VNLnet (ours). We highlighted the best performance in black and the second best in brown.

Table 6. Running time per frame on a $960 \times 540$ video for VBM3D, DnCNN, VBM4D, VNLB and SPTWO on single CPU core.

The results are summarized on Table 5. VNLB wasn’t considered due to its computation time. We note that the proposed method outperforms DnCNN, a frame by frame denoiser, and VBM3D, a state-of-the-art video denoising method.

A note on running times. On Table 6, we compare the CPU running time of VBM3D, DnCNN and VNLB when denoising a video frame. While we do not have a CPU

| Method | $\sigma = 10$ | $\sigma = 20$ | $\sigma = 40$ |
|--------|---------------|---------------|---------------|
| DnCNN  | 36.80         | 32.94         | 28.69         |
| VBM3D  | 37.43         | 33.75         | 30.12         |
| VNLnet | 39.08         | 35.44         | 31.79         |

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

Table 6. Running time per frame on a $960 \times 540$ video for VBM3D, DnCNN, VBM4D, VNLB and SPTWO on single CPU core.

| Method   | 1.3s | 13s | 52s | 140s | 210s |
|----------|------|-----|-----|------|------|
| VBM3D    |      |     |     |      |      |
| DnCNN    |      |     |     |      |      |
| VBM4D    | 1.3s | 13s | 52s | 140s | 210s |
| VNLB     |      |     |     |      |      |
| SPTWO    | 1.3s | 13s | 52s | 140s | 210s |
Figure 5. Visual comparison of the denoising results on the seven images reported in Table 4 for noise $\sigma = 20$.

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.
| Non-local search | Rest of the network | DnCNN |
|------------------|---------------------|-------|
| 932 ms           | 80 ms               | 95 ms |

Table 7. Running time per frame on a 960 × 540 video on a Nvidia Titan V (41 × 41 patches at every position, 41 × 41 × 15 3D windows, the default parameters).

implementation of the patch search layer, the GPU runtimes of Table 7 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 [5].

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 naively, 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:

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

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 the supplementary material.

6. Conclusions

We have proposed a simple but efficient way of incorporating non-local information into a Neural network architecture for denoising. Our proposed method first computes for each image patch the $n$ most similar neighbors on a spatio-temporal window and gathers the center pixel of each similar patch forming a non-local feature vector which is given to a simplified DnCNN. Our method yields a significant gain compared to using DnCNN directly on each video frame. But it also outperforms many state-of-the-art video denoising algorithms, including the popular VBM3D.

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.

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.

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

[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.

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

[5] 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.

[6] 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.
[7] 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

[8] 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

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

[10] 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, 2017. 1, 2

[11] 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

[12] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3D transform-domain collaborative filtering. In EU-SIPCO, 2007. 2, 5, 6

[13] 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. 2

[14] 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

[15] C. Godard, 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

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

[17] 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

[18] 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

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

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

[21] 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

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

[23] 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

[24] 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

[25] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian. Video Denoising Using Separable 4D Nonlocal Spatiotemporal Transforms. In Proc. of SPIE, 2011. 5, 6

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

[27] 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

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

[29] 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. 3, 6

[30] 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

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

[32] 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

[33] 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
[39] 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

[40] 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

[41] 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

[42] 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. 1, 2, 3, 4, 6

[43] 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

[44] 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
1. More implementation details on the non-local search

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

As mentioned in the main paper, a naive 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 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

```plaintext
Input: New position $p$ and distance $d$
Tables Positions and Distances of length $N$ each.
if $d < \text{Distances}[N-1]$ then
    for $i$ from $N-1$ to 1 do
        insert ← $\text{Distances}[i-1] \leq d$
        Positions[$i$] ← $p$ if insert else Positions[$i-1$]
        Distances[$i$] ← $d$ if insert else Distances[$i-1$]
        quit function if insert
    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. Though, 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 horizontal 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 naive 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 naive 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 naive 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 naive implementation (both using Algorithm 1 for the tables of distances and positions).

2. Some more results

Visualizing the videos is important to evaluate the qualitative aspects of the results that cannot be appreciated from a single image, such as the temporal consistency. When visualizing them, we can observe that the result of VNLB seems the most temporally consistent, which can be attributed to the use of 3D spatio-temporal patches. The proposed VNLnet is still overall temporally consistent, even though each frame is processed independently. DnCNN is not temporally consistent. Another visible aspect is that VNLnet restores more details. In Figure 1, we highlight some areas of the sequences where the level of recovered detail by each method is quite different. The images are from the Derf’s Test Media collection used for the test set.

\[1\]https://media.xiph.org/video/derf
Figure 1. Examples of areas where the level of restored detail of the methods differs significantly on crowd, park and station.