Published in Image Processing On Line on YYYY–MM–DD.
Submitted on YYYY–MM–DD, accepted on YYYY–MM–DD.
ISSN 2105–1232 © YYYY IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol

An implementation of "Efficient Multi-Stage Video Denoising Method" and some variants

Zhe Zheng, Gabriele Facciolo, Pablo Arias

Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, 91190, Gif-sur-Yvette, France

PREPRINT May 27, 2022

Abstract

Recently, the field of image and video denoising has undergone a revolution thanks to deep learning approaches. These methods outperform traditional model-based approaches in almost every image/video restoration problem. In this paper, we propose an implementation of a recent approach proposed for video denoising, namely Efficient Multi-stage Video Denoising method (EMVD) [22]. The method has a lightweight and interpretable architecture consisting of three stages: temporal fusion, denoising, and refinement stages. We reproduce this method and propose three modifications aimed at improving its performance. (1) We apply motion compensation to make better use of temporal redundancy, (2) we apply variance stabilization to help this lightweight network deal with signal-dependent noise and (3) we decouple occlusion detection and fusion weights prediction. We evaluate the original method and the proposed modifications on a task of raw video denoising.

Keywords: Video denoising, recurrent methods, convolutional neural network

1 Introduction

Digital images/videos have reached a high importance in our world. They are widely used in many fields, like computed tomography (CT) image for medical purpose, SAR or optical image for remote sensing purpose, and they are also ubiquitous in our daily

life due to the popularity of smartphones. Digital images/videos are inevitably corrupted by the nature of imaging process, like the photon counting. This results in strong noise which reduces the visual quality and limits further usage of images/videos, specially in low light imaging conditions. As a consequence, it is necessary to perform restoration techniques to improve the quality of images/videos. Denoising, aiming at estimating the true value at every pixel, is one of the most important steps among the whole pipeline, whose outputs have a direct effect on the rest of the pipeline. In the last few past decades, many methods have been proposed for this denoising purpose. The classical approach for addressing denoising problems is based on mathematical modeling of signal and noise. Research in the past decades have produced many elegant models which are capable of obtaining good results. More recently, along with the development of the computational power of GPUs, models based on deep neural networks emerged as the new state of the art, drawing almost all the attention of the community.

1.1 Related works

Image denoising Image denoising is a long-standing problem and there are a variety of methods that have been proposed for addressing it before deep learning, for example, PDE and variational methods [6, 25], transformed domain methods [9], and patch-based methods [3, 10]. The latter give good denoising results and were the state of the art for more than a decade, albeit sometimes demanding of huge computational cost [16]. In recent years, methods based on neural networks have outperformed model-based method, which brought a revolution to this field. Probably the first neural network based methods that gave competitive results was proposed in [5], where the authors used a multilayer perceptron trained to denoise 17×17 patches. This is a heavy architecture, using fully connected layers with 2048 hidden features, which makes it computationally demanding. DnCNN [31], as well as FFDnet [32] which is an improved version of DnCNN, have achieved great success in this domain since they are proposed in 2017. They adopted convolutional layers, ReLU activation and batch normalization, residual learning, which are common techniques in network structure designing. Recently, the popular Transformer has gained much attention in computer vision community, and it has also been exploited for image processing [20, 21].

Video denoising In the context of video denoising, making use of temporal redundancy is of critical importance. This characteristic of videos should facilitate denoising performance compared with single image denoising, as it provides additional information. To enforce temporal consistence in final denoising results, two factors are usually considered: use 3D spatio-temporal receptive fields and/or use motion compensation to better exploit temporal redundancy. In fact, algorithms for video processing take advantages of them explicitly or implicitly.

Deep learning based methods have been applied to video denoising since 2016 when in [7] the authors proposed to address video denoising adopting recurrent neural network. But it was not until very recently that they became the state of the art. In recent years,

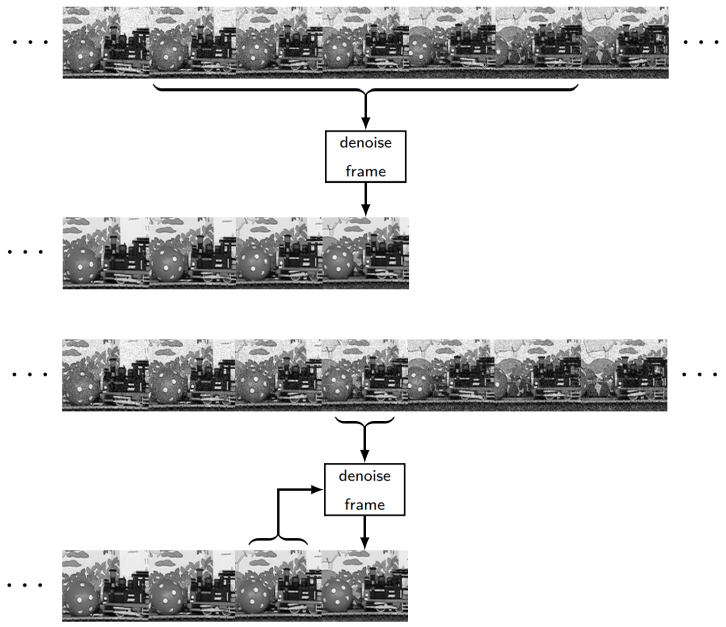

Figure 1: non-Recurrent v.s. Recurrent. A non-recurrent method processes many frames (past and future) to denoise reference frame, while a recurrent method takes the previous output, together with current noisy frame as input.

several convolutional neural network (CNN) based methods have been proposed, e.g. VNLnet [11], DVDnet [27], ViDeNN [8], fastDVDnet [28] and VRT [20]. These methods either take several consecutive frames as input, or search for similar patches among a large temporal-spatio volume, which requires a lot of computational resources.

Recurrent methods Efficiency is a key aspect for practical video restoration. In spite of their good performance, deep learning based methods require a large number of operations per pixels, which becomes prohibitively for their application in real case scenarios. Yet, methods in most current existing literature focus on quality rather than efficiency. Traditional approaches for real-time applications often rely on **recurrence**: a recurrent method receives as input its own output for the previous frame or more generally, an encoding of it, see Figure 1. This allows to incorporate information from past frames, without incurring in excessive cost [1, 14]. This can reduce computational cost and memory cost. Moreover, by incorporating past frames, temporal consistency is enforced in a natural and direct way. Recurrent CNNs for video processing have attracted attention [18, 26], but most of them focused on super resolution rather than denoising.

The first recurrent CNN for video-resolution was proposed in [17], where a heavy set of temporal connections was used, including forward and backward subnetworks. In [15,26], the authors proposed to use recurrence for video super-resolution. A small difference between these two is that [26] used frame-recurrence, while [15] proposed to exploit an encoding of the frame, namely a hidden state. In addition to exploit recurrence differently, there are also other strategies used and worth mentioning. For example, in [18], the

authors use recurrence to conduct super-resolution in a streaming way similar to [15,26]. Moreover, they proposed to decompose each frame into two components, the detail and structure components, and adapt different hidden state to each reference frame. This strategy also plays a role in the method we want to analyse.

Recurrent methods are also applied to other video recognition tasks, like [12,13] for action recognition and image captioning. But rarely can they be found in video denoising, and most of the existing methods are still prohibitively complex for real-time application. An exception is the frame-recurrent approach proposed in [22]: Efficient Multi-Stage Video Denoising (EMVD). It has a very low computational cost and and still reports a performance comparable to other state-of-the-art methods.

In this work we propose an implementation of EMVD, and study its performance. We also proposed to improve it through three modifications.

Noise model. Most methods in signal processing assume additive white Gaussian noise (AWGN). While it is true that central-limit theorem supports the Gaussian distribution of random errors, data acquired in real applications can seldom be described with good approximation by AWGN model.

Two main sources of noise in imaging process are shot noise and thermal noise, which lead to a Poisson-Gaussion model. Denoting clean and noisy images as y and z respectively, this model can be approximated by the following signal-dependent heteroskedastic Gaussian model:

$$z(x) = y(x) + \sigma(y(x))\eta(x), \tag{1}$$

where $\sigma(y(x)) = \sqrt{ay(x) + b}$ and $\eta(x) \sim \mathcal{N}(0, 1)$. In this approximation, $z|y$ follows a Gaussian distribution, i.e. $\mathcal{N}(y(x), \sigma^2(y(x)))$.

The main difference between these model and AWGN is that here the noise variance is not a constant but depends on signal. The noise model for video denoising introduces an additional temporal index, which is formulated as follows:

$$z_t(x) = y_t(x) + \sigma(y_t(x))\eta_t(x), \tag{2}$$

where $x \in X \subset \mathbb{N}^2$ denotes the spatial position in frame, and $t \in T \subset \mathbb{N}$ denotes the temporal index of the frame in a video sequence.

In this paper, we focus on RAW image processing. In practice, the image acquired by the sensor is different from the processed RGB image that we see. Each pixel in the sensor can only capture a single color channel. Half of the sensor pixels capture only green channel, a quarter only the red channel and the remaining quarter the blue channel. Pixels for three different color in the sensor are arranged in a specific pattern called Bayer pattern. Images of this type are called mosaicked images. For each pixel, only one color is known, so we need to find other two missing colors. This problem is called demosaicking. We will store an $W \times H$ raw image as a $W/2 \times H/2$ image with 4 channels which is sometimes called a *packed raw*. Two channels correspond to the red and blue pixels in the Bayer pattern and the remaining two green pixels.

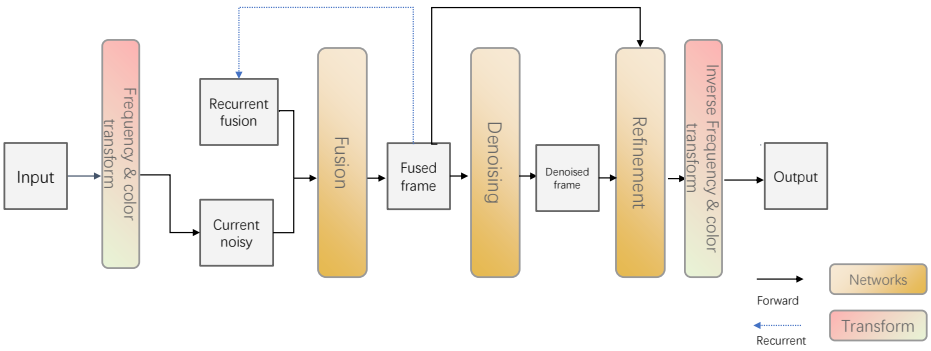

Figure 2: The architecture of the EMVD method (single scale version).

2 Review of EMVD method

The EMVD method is composed of 3 convolutional networks, which correspond to 3 processing stages: temporal fusion, pre-denoising and refinement. The input data is mapped into a transformed domain by a color and a frequency trainable transforms. The structure of a single scale EMVD is illustrated in Figure 2

2.1 Trainable transforms

Each noisy frame z_t is pre-processed by linear orthogonal trainable color and frequency transforms. They are both implemented as convolutional operations. The kernel for color transform $\mathcal{T}_c$ is a matrix $M \in \mathbb{R}^{C\times C}$, where C is the number of channels (4 for a packed RAW frame). It aims at decorrelating the color to luminance-chrominance representation. The inverse operation is also a convolution with kernel initialized as $M' = M^T$. The initial value is as follows, based on the opponents transform [4]:

$$M = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ 0.65 & 0.2784 & -0.2784 & -0.65 \\ -0.2784 & 0.65 & -0.65 & 0.2784 \end{pmatrix} \tag{3}$$

The frequency transform $\mathcal{T}_f$ decorrelates the input frequencies into four half-resolution components: low-pass LL subband and three high-pass LH, HL, HH subbands. At the output of the network the inverse transforms are applied. Four $n \times n$ kernels which are the outer product of some chosen wavelet decomposition and reconstruction filters, denoted by ψ and ϕ respectively. In our case, we use Haar transform as initialization,

Algorithm 1: EMVD: recurrent method for video denoising

Input: Noisy frames $\{z_t\}_{t=0}^{T}$, Optical flows $\{\mathbf{v}_t\}_{t=0}^{T}$
Output: A sequence of estimation $\{\hat{y}_t\}_{t=1}^{T}$

Function EMVD($z_t, \bar{y}_{t-1}$)**:**
- $z_t \leftarrow \mathrm{CT}(z_t)$; // color transform $\mathcal{T}_c$
- $z_t \leftarrow \mathrm{FT}(z_t)$; // frequency transform $\mathcal{T}_f$
- $\hat{\sigma}_t^2 \leftarrow \sigma^2(z_{t|LL})$;
- $\gamma_t \leftarrow \mathrm{FCNN}(|z_{t|LL} - \bar{y}_{t-1|LL}|, \hat{\sigma}_t^2)$; // compute fusion weight
- $\bar{y}_t \leftarrow z_t(1-\gamma_t) + \bar{y}_{t-1}\gamma_t$; // update temporal fusion
- $\bar{\sigma}_t^2 \leftarrow \gamma_t^2\bar{\sigma}_{t-1}^2 + (1-\gamma_t)^2\sigma_t^2$; // and its variance accordingly
- $\tilde{y}_t \leftarrow \mathrm{DCNN}(\bar{y}_t, x_{t|LL}, \bar{\sigma}_t^2)$; // compute pre-denoising result
- $\omega_t \leftarrow \mathrm{RCNN}(\tilde{y}_t, \bar{y}_t, \bar{\sigma}_t^2)$; // compute refine weight
- $\hat{y}_t \leftarrow \bar{y}_t\omega_t + \tilde{y}_t(1-\omega_t)$;
- $\hat{y}_t \leftarrow \mathrm{INVFT}(\hat{y}_t)$; // inverse color transform $\mathcal{T}_c^{-1}$
- $\hat{y}_t \leftarrow \mathrm{INVCT}(\hat{y}_t)$; // inverse frequency transform $\mathcal{T}_f^{-1}$
- **return** $\hat{y}_t$

$\bar{y}_0 \leftarrow \mathrm{CT}(z_0)$; // Initialize temporal fusion
$\bar{\sigma}_0^2 \leftarrow \sigma(\bar{y}_{0|LL})$; // and variance of it
for $t \leftarrow 1$ **to** T **do**
- $\bar{y}_{t-1} \leftarrow \mathrm{WARP\text{-}BICUBIC}(\bar{y}_{t-1}, \mathbf{v}_{t-1})$; // warp temporal fusion to current noisy frame
- $\bar{y}_{t-1} \leftarrow \mathrm{FT}(\bar{y}_{t-1})$; // frequency transform $\mathcal{T}_f$
- $\hat{y}_t = \mathrm{EMVD}(z_t, \bar{y}_{t-1})$;
- $\bar{y}_t \leftarrow \mathrm{INVFT}(\bar{y}_t)$; // inverse temporal fusion for next warp

and corresponding filters are as follows:

$$\phi = \begin{pmatrix} \phi_L \\ \phi_H \end{pmatrix} \qquad \psi = \begin{pmatrix} \psi_L \\ \psi_H \end{pmatrix}$$
$$= \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} \in \mathbb{R}^{2\times 2}, \quad = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} \in \mathbb{R}^{2\times 2}$$

To enforce the invertiblity of the transforms during training, two additional loss terms are added (the complete loss will be described later):

$$\mathcal{L}_c = \|MM^T - I_C\|_F^2, \quad \mathcal{L}_f = \|\psi\phi^T - I_2\|_F^2. \tag{4}$$

2.2 Temporal fusion

EMVD achieves temporal denoising by keeping running frame average $\bar{y}_t$ and its corresponding variance map $\bar{\sigma}^2$. Mathematically, it can be represented as:

$$\bar{y}_t = \bar{y}_{t-1} \odot \gamma_t + z_t \odot (1 - \gamma_t), \tag{5}$$
$$\bar{\sigma}_t^2 = \bar{\sigma}_{t-1}^2 \odot \gamma_t^2 + \sigma_t^2 \odot (1 - \gamma_t)^2, \tag{6}$$

where $\odot$ denotes the element-wise product. The average weight γ_t satisfies $\gamma_t(x) \in (0, 1)$ for any given pixel position x. It is given by a fusion CNN (denoted as FCNN in the following), which takes as input the absolute value of the difference between the LL subbands of z_t and previous average $\bar{y}_{t-1}$, together with the input's noise variance map σ_t^2:

$$\gamma_t = \text{FCNN}(|z_{LL|t} - \bar{y}_{LL|t-1}|, \hat{\sigma}_t^2), \tag{7}$$

where the estimated variance of input noisy frame z_t, computed as $\hat{\sigma}_t^2 = \sigma^2(z_{LL|t})$. To ensure fusion weights between 0 and 1, the output layer is followed by a sigmoid activation function $\text{Sigmoid}(z) = \frac{1}{1+e^{-z}}$.

The FCNN is expected to complete two tasks: 1) detecting miss-alignments between the two frames; 2) predicting reasonable weights for temporal averaging. The fusion weights allow noise reduction at locations where z_t and $\bar{y}_{t-1}$ are well aligned, and prevent averaging if changes are detected. In these locations, the output should coincide with the noisy input z_t. We initialize the previous fused image with the first noisy observation, i.e. $\bar{y}_0 = z_0$.

2.3 Pre-denoising

The second stage is called pre-denoising, where a spatial denoising is applied to the temporal fusion $\bar{y}_t$. The pre-denoising is given by a denoising network, called DCNN, which takes as input the temporal average and its variance map, plus the LL subband of the noisy frame:

$$\tilde{y}_t = \text{DCNN}(\bar{y}_t, z_{LL|t}, \bar{\sigma}_t^2). \tag{8}$$

Here $\tilde{y}_t$ is the pre-denoised image, and $\bar{\sigma}_t^2$ is the noise variance of the temporal fusion $\bar{y}_t$. The LL subband of current noisy frame $z_{LL|t}$ is also included as an input because it is expected for the network to acquire valuable structure information from this unprocessed input. The update of $\bar{\sigma}_t^2$ is given as in (5).

2.4 Refinement

The refinement stage performs a convex combination between the pre-denoised frame $\tilde{y}_t$ and the temporal fusion $\bar{y}_t$. The weights are computed by a refinement network RCNN:

$$\omega_t = \text{RCNN}(\tilde{y}_t, \bar{y}_t, \bar{\sigma}_t^2) \tag{9}$$

$$\hat{y}_t = \bar{y}_t \odot \omega_t + \tilde{y}_t \odot (1 - \omega_t). \tag{10}$$

As with the fusion network, a sigmoid activation function at the end of the network ensures that the weights are between 0 and 1. Since the image obtained after the pre-denoising stage could lose details and generate some artifacts(specially because the goal is to use shallow networks), this stage aims at adding high-frequency information back from the fused image $\bar{y}_t$.

Together with the fusion expression (5) in the processing pipeline, the output $\hat{y}_t$ can be expressed as follows

$$\hat{y}_t = \bar{y}_{t-1} \odot \gamma_t \odot \omega_t + z_t \odot (1 - \gamma_t) \odot \omega_t + \tilde{y}_t \odot (1 - \omega_t).$$

This shows that the final output is a convex combination of the noisy frame, the previous temporal average and the output of the denoising network. This also provides a way to justify which component plays the most important role by investigating how much its output counts in the final output.

2.5 CNN blocks

Following the author's suggestion on efficiency, we adopt a simple structure, where in default, all three networks are of three convolutional layers, with 16 hidden features. See Figure 3. Except for the last convolutional layer, other two are followed by a ReLU activation function. The output layer of fusion and refinement are followed by a sigmoid function, to ensure the values of weight are mapped to $[0, 1]$, while there is no activation function applied to the output layer of denoising network.

2.6 Multiscale structure

The Haar transform can also be applied to the LL subbands of transformed images. In that way, we get an architecture with multiscale structure, see Figure 4. Results obtained

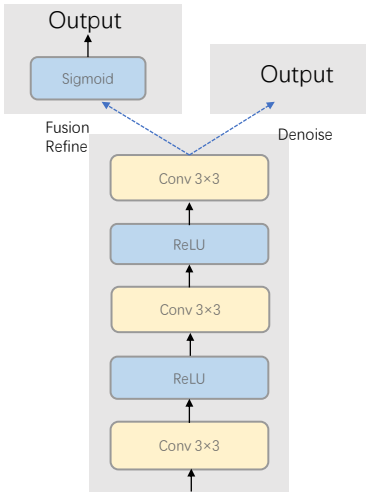

Figure 3: CNN blocks used in EMVD. Output layers of fusion and refinement are activated by a sigmoid, while no activation function applied to the output of denoising.

at this coarser scale can be used as guides for the computations in the upper scale. In the EMVD method, pipelines at coarser scale are similar to the finest scale, except for excluding the refinement stage. The inputs of the coarser scale include the LL subbands of both recurrent fused feature and current noisy frame, and the output are fusion weights and pre-denoised frame after inverse Haar transform. Fusion weights are upsampled and concatenated to the inputs of fusion stage at an upper scale, and the denoised frame is used as a guidance in the denoising stage of an upper scale. Denoting these two by γ^{co} and y_{LL}^{co}, we have new equations for fusion and denoising at a finer scale:

$$\begin{aligned} \gamma_t &= \mathrm{FCNN}(|z_{LL|t} - \bar{y}_{LL|t-1}|, \hat{\sigma}_t^2) \quad \Longrightarrow \gamma_t = \mathrm{FCNN}(|z_{LL|t} - \bar{y}_{LL|t-1}|, \hat{\sigma}_t^2, \gamma^{co}) \\ \tilde{y}_t &= \mathrm{DCNN}(\bar{y}_t, z_{LL|t}, \bar{\sigma}_t^2) \qquad\qquad \Longrightarrow \tilde{y}_t = \mathrm{DCNN}(\bar{y}_t, z_{LL|t}, \bar{\sigma}_t^2, y_{LL}^{co}). \end{aligned}$$

The EMVD with multi-scale structure is shown in Figure 4. The shaded part shows the forward pass at a lower scale.

3 Improvement on EMVD

3.1 Motion compensation.

Since the temporal fusion (5) is just a temporal average, a good frame alignment is essential to attain good results. This is especially true for videos with camera motion. In our experiment, we use the REDS dataset [23], which is captured by a hand-held camera, resulting in a lot of motion in these sequences. We choose the $TV - \ell_1$ method [24, 30] to compute the optical flow between the noisy frames t and $t-1$, which we use to warp the previous temporal average $\bar{y}_{t-1}$ and align it to z_t. This will enable us to make better use of temporal redundancy.

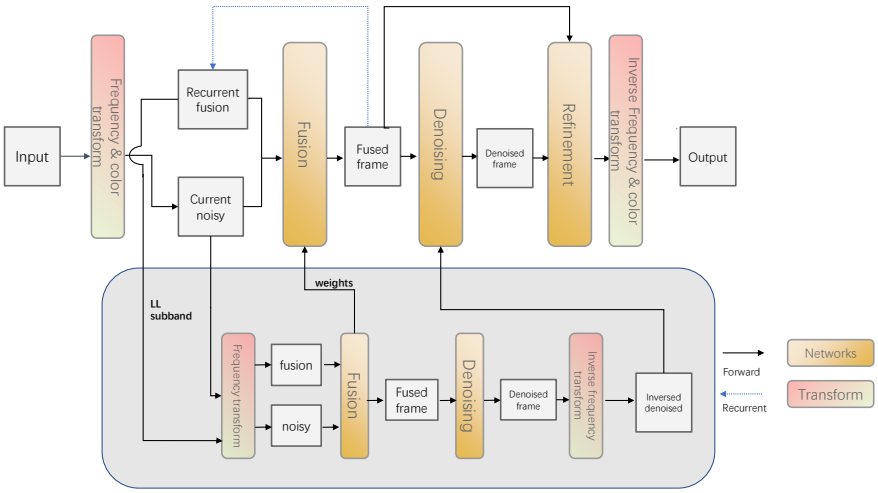

Figure 4: EMVD multiscale network architecture. The shaded part shows the work pipeline in coarser scale.

3.2 Variance stabilization.

The noise in the RAW video can be approximated as a heteroscedastic Gaussian with signal-dependent variance. A variance stabilizing transform (VST) can transform this noise to a standard AWGN. VSTs are a common way to adapt a AWGN denoiser to real noise. With learning-based methods, VSTs are not necessary anymore, as it has been shown that the networks can handle signal dependent noise. However, since the networks in EMVD are rather small, stabilizing the variance might help. The procedure of applying VST involves three steps: First, transform the noisy data by a nonlinear VST which is designed for a specific noise model. Then, we denoise such transformed data. Finally, we apply an inverse VST to the denoised data, obtaining data in the original range.

For the signal-dependent Gaussian noise we assume $n_t(x) \sim \mathcal{N}(0, ay_t(x) + b)$, and we use the following VST: $f(x) = \frac{2}{a}\sqrt{ax + b}$. After the VST, we have approximately AWGN with unit variance. We invert the VST with the algebraic inverse $f^{-1}(x) = \frac{ax^2}{4} - \frac{b}{a}$. This inverse is known to introduce a bias for dark values [**?**], but since the bias is deterministic, the network can learn to compensate it.

3.3 Decoupling occlusion detection and fusion weights prediction.

In the original paper [22], the fusion network FCNN aims at estimating the combination weight between temporal fusion and current noisy. These weights fulfill two goals: (1) They have to avoid using the motion compensated $\bar{y}_{t-1}$ in locations where the alignment failed, and (2) in the regions where the alignment is accurate, the weight has to be chosen to determine an optimal temporal fusion. For the latter, it is reasonable to give

as input to the fusion network both the estimated variances of the current noisy frame σ_t^2 and of temporal fusion $\bar{\sigma}_{t-1}^2$. Indeed, for a pixel x, and assuming that $\bar{y}_{t-1}(x)$ is an unbiased estimatior of $y_{t-1}(x)$, the ratio

$$\frac{\mathbb{V}(z_t(x))}{\mathbb{V}(\bar{y}_{t-1}(x)) + \mathbb{V}(z_t(x))}$$

leads to an optimal convex combination in the MSE sense.

We tried two ways to exploit the variance of temporal fusion $\bar{\sigma}_{t-1}^2$ described in the following.

As a direct input. The fusion weights will be predicted as follows:

$$\gamma_t = \text{FCNN}(|z_{LL|t} - \bar{y}_{LL|t-1}|, \hat{\sigma}_t^2, \bar{\sigma}_{t-1}^2), \tag{11}$$

and the update of variance of fusion keeps the same:

$$\bar{\sigma}_t^2 = \bar{\sigma}_{t-1}^2(1-\gamma_t)^2 + \sigma_t^2\gamma_t^2. \tag{12}$$

However, it is problematic to add $\bar{\sigma}_{t-1}^2$ as a direct input to FCNN: the obtained fusion weights γ_t suffered from severe artefacts – mostly horizontal and vertical short black line segements. Moreover, once these artifacts appear, they remain for several frames and move consistently with the motion in the video. This can be explained from equations (13) and (12). The FCNN tends to give small value to γ_t where the fusion variance $\bar{\sigma}_{t-1}^2$ is relatively high. This is consistent with the equation for the optimal weights. Once there is a low value in γ_t, it will output $\bar{\sigma}_t^2$ with a high value, which in turn will create a low fusion weight when processing the next frame. Since the dark artefacts sometimes form periodic arrengements, We tried to regularize this fusion weight image with a TV (Total Variation) loss. However, the artefacts are always noticeable.

As a multiplicative correction. Instead of giving $\bar{\sigma}_{t-1}^2$ as a direct input FCNN, we propose to decouple the miss-alignment detection from the fusion weight computation. To that aim, we first run the fusion network as in [22] and then we multiply its output by an the optimal fusion weights. Mathematically, the fusion weights will be predicted as follows:

$$\gamma_t^{occ} = \text{FCNN}(|z_{LL|t} - \bar{y}_{LL|t-1}|, \hat{\sigma}_t^2). \tag{13}$$

$$\bar{\gamma}_{t-1} = \gamma_t^{occ} \times \frac{\sigma_t^2}{\bar{\sigma}_{t-1}^2 + \sigma_t^2} \tag{14}$$

$$\gamma_t = 1 - \bar{\gamma}_{t-1} \tag{15}$$

The reasoning is that in this way the FCNN can focus on producing a (approximately) binary miss-alignment map. When there are no occlusions, we apply the weights that minimize the MSE of the temporal fusion. However, since $\bar{y}_t$ is not the final output of the network, it might be that the best weights for the network are not those of variance ratio (Eq. (14)). If this optimal value is smaller than the variance ratio, the FCNN can correct it via the product with γ_t^{occ}.

4 Experiments

4.1 Dataset

The dataset we use in the training is from the REDS [23] public dataset. The original videos are RGB captured at 30 frames per second. This dataset consists of 270 sequences with 90 frames of size 1280×720 pixels, split into 240 sequences for training and 30 of them for testing and validation. To simulate RAW data, we use an unprocessing pipeline similar to that of [2] . This unprocessing consist in inverting the main steps of an image camera pipeline:

1. inverse tone mapping
2. inverse Gamma curve
3. inverse color matrix
4. inverse white balance
5. apply Bayer pattern.

After unprocessing, we add noise of two levels to the RAW sequences, using noise model which are estimated from the CRVD dataset [29], ISO3200 and ISO12800. In our experiments, each frame is stored in the packed RAW way, in which a $W \times H$ raw image is stored as a $W/2 \times H/2$ image with 4 channels, one channel for red, one channel for blue, and the other two for green (packed RAW images).

4.2 Training

Loss function: The total loss is the sum of three parts: loss for color transform $\mathcal{L}_f$, loss for frequency transform $\mathcal{L}_c$, and L1 loss as image reconstruction criterion in the training $\mathcal{L}_{im}$. Since our network is recurrent, we need to apply it over several frames in order to train it, and backpropagate the gradient from the last output throughout all the applications of the network. We use the term *unrollings* to each application of the network during training. The final loss is a weighted sum of outputs produced for each unrolling. Mathematically, the image reconstruction loss for n unrollings can be formulated as:

$$\mathcal{L}_{im}(\{\hat{y}_t\}_{t=1,\dots,n}) = \sum_{t=1}^{n} w_t \ell(y_t, \hat{y}_t), \quad \ell(y_t, \hat{y}_t) = ||y_t - \hat{y}_t||_1 \tag{16}$$

where $\sum_{t=1}^{n} w_t = 1$, y_t and $\hat{y}_t$ stand for the i-th clean and denoised frame, respectively. To speed up the training, we use the following strategy: we put 100% weight on the first unrolling in the first 20 epochs, i.e.

$$w_1 = 1, \quad w_i = 0, \forall i \neq 1.$$

In this way, only the first unrolling will be penalized:

$$\mathcal{L}_{im}(\{\hat{y}_t\}_{t=1,\dots,n}) = \ell(y_1, \hat{y}_1),$$

meaning that the network is trained as a non-recurrent network. In the next 5 epochs, we use a gradual transition weight schedule, increasing the weights on the last unrolling. The final distribution of weights puts 90% of the weights on the last unrolling, and the remaining 10% weights is split on the rest unrollings uniformly, i.e.

$$w_t = \frac{1}{10(n-1)}, t = 1, 2, \ldots, n-1, \quad w_n = \frac{9}{10}.$$

The reason why we prefer putting most of the weight on the last unrolling is that, the network will be applied to a large number of frames to denoise in the testing. Though it is never trained for that number of unrollings. The last unrolling is assumed to be the closest one to the steady state operation of the network which is mostly used in testing.

We found that the pre-denoising network may not learn what it is supposed to: it may conduct a residual learning, other than a spatial denoising. To avoid this unexpected result, we add L1 loss to the output of pre-denoising network. Thus the image reconstruction loss reads:

$$\mathcal{L}_{im}(\{\tilde{y}_t, \hat{y}_t\}_{t=1,\ldots,n}) = \sum_{t=1}^{n} w_t(||y_t - \hat{y}_t||_1 + \lambda||y_t - \tilde{y}_t||_1). \tag{17}$$

Dataloader Reading from the hard-drive is slow. To avoid this, we use a dataloader that pre-loads a section of the dataset into RAM. This speeds up training (at the expense of requiring a large amount of RAM memory). At the beginning of each epoch we load into RAM a random segment of 10 consecutive frames (for more unrollings, we load more frames, as explained in the subsequent text) from each sequence in the training set, together with the optical flows. From these spatio-temporal volumes, we define a set of 3D crops with a stride of 3 pixels in all dimensions (x, y and t). During the entire epoch, the mini-batches sampled at random from these set of crops. The crops have a spatial size of 136×136 with a number of frames dictated by the network and the number of unrollings. The network processes each 3D crop in the mini-batch and returns an output consisting of n frames for a recurrent network trained with n unrollings. The number of unrollings describes the number of frames being denoised by the network. This concept appears in the recurrent network, determined by the number of frames and previous recurrent features we use. For example, in our case, the common settings are we take 5 consecutive frames, and just use one recurrent feature. This results in 4 frames to be denoised, thus 4 unrollings.

Optimizer and learning rate: We use the Adam optimizer [19] to update the weights. The total training takes 100 epochs. During the first 70 epochs, the network is trained with a fixed learning rate (typically 2e-4) and in the following 30 epochs, the learning rate would be reduced at each epoch linearly to 0.

For the noise maps, we mention that they are of only one channel, even if the RAW image is stored with four channels. This is because after the color transform (3), there will be a channel carrying most of the information (the one computed using the first row), it is reasonable to make use this transformed channel to estimate the variance.

ISO	stage	full	w/o VST	w/o VR	w/o warp	only warp (TV-ℓ_1)
12800	final	36.205	**36.322**	36.247	35.661	36.239
	pre-denoise	35.019	**35.274**	34.836	34.839	34.988
	fusion	30.310	**30.736**	29.971	27.876	30.371
3200	final	**40.948**	40.775	40.771	40.242	40.868
	pre-denoise	**39.094**	38.545	37.807	37.574	38.661
	fusion	38.366	**38.543**	38.161	35.845	38.331

Table 1: Ablation study for proposed improvements, small network, two ISOs.

ISO	stage	full	w/o VST	w/o VR	w/o warp	only warp (TV-ℓ_1)
12800	final	37.740	37.772	37.784	37.097	**37.836**
	pre-denoise	37.657	37.673	37.666	36.969	**37.752**
	fusion	30.270	**30.292**	30.038	27.809	30.139
3200	final	**42.123**	42.039	42.120	41.634	42.020
	pre-denoise	**41.916**	41.767	41.890	41.362	41.364
	fusion	38.079	**38.469**	37.919	35.743	38.388

Table 2: Ablation study for proposed improvements, big network, two ISOs.

4.3 Experimental results

We report the ablation study on the proposed modifications in Table 1 and 2. We compared the following settings:

full All three proposed modifications are applied in this setting.

w/o VST The variance stabilization is removed from full, to illustrate the usefulness.

w/o VR The variance ratio is removed from full, to compare the usefulness of it.

w/o warp The variance ratio is removed from full.

only warp Only motion compensation is applied. This one is considered as the original version in [22], since the dataset in our experiments contains a lot of motions.

To assess how the network capacity would influence the performance, we consider the EMVD with two network configurations, one with small capacity with 3 layers and 16 features for each one of the three networks (Table 1), the other one with a big capacity has 5 layers and 64 features for all three networks (Table 2).

In the tables we report the PSNR values for the three stages of the method: temporal fusion, pre-denoising and the final refined result. Figure 5 shows an example of the results tested for both ISOs, using the small networks with all the proposed modifications. All three stages' outputs are included. Noise remains in temporal fusion, and it is removed

Figure 5: Visual results of all stages. From left to right: fusion, pre-denosied, final. The first two rows show ISO 3200, and the last two are for ISO 12800.

after the pre-denoising stage. Details are added back in the final output, compared to the oversmoothed pre-denoised frame.

In all cases, warping leads to a significant improvement. This is reasonable, since there is a lot of camera motion in the REDS dataset. Without temporal alignment, the fusion weights are mostly 0, meaning that the network just performs a single image denoising with no temporal aggregation. Figure 6 compares the outputs with and without warping. It is obvious there is almost no temporal denoising in fusion stage. This also can be observed in the PSNR value of fusion output in Tables 2 and 1. Indeed, the original version of the method (which does not consider motion estimation) was intended to be use on sequences taken with a static camera thus having a static background.

For the other modifications, i.e. VST and the multiplication by the variance ratio (VR), we obtain very similar PSNR values. In Figure 7 we compare the results of adopting only warping with those where the VST and variance ratio are added respectively. For the sake of space, we just show visual results of ISO 3200. The impact of the VST is very hard to notice. The variance ratio yields a temporal fusion which has less noise in smooth areas such as the sky. This is consistent with the PSNR measures, which show that the variance ratio leads to 0.2-0.4 dB gains in the output of temporal fusion for ISO 12800, depending on the capacity of networks. However, it is very hard to notice any difference in the final results.

It is of interest to inspect how the temporal fusion is improved. In the last row of Figure 7 and in Figure 8, we show different fusion weights under different settings. We can see that without the VST, the fusion weights are more correlated with the image content. This is to be expected, as the variance is proportional to the image. It can also be observed that when adding the product with the variance ratio, the fusion weights tend to be higher (e.g. in the sky), which implies a stronger temporal averaging.

The EMVD is a light-weighted network designed for real-time applications. The small configuration gives quite good results. It is not surprising that big networks improve PSNR values of the pre-denoised and the final result. However, the PSNR values of temporal fusion are almost the same as those given by the small network. From this observation, the way of using convex combination to reduce temporal noise has reached its limit, and it follows that the only way to improve the final performance is by improving the spatial denoising. Indeed with a bigger pre-denoising network, it is possible to perform a better spatial denoising. We show in Figure 9 two pairs of refinement weights images where only capacity of networks changed. The refinement weights choose between the temporal fusion and the pre-denoised images. The ones corresponding with the big networks put more weight on pre-denoised one rather than the temporal fusion. This illustrates the importance of the spatial pre-denoising network in the result.

Lastly, we consider how the number of unrollings will influence the performance. It is possible that the limited performance of the temporal fusion is due to the fact that during the training with 4 unrollings, the network is only applied to short videos of 5 frames. We compare how the number of unrollings affects denoising preformance in Table 3 by listing the final PSNR values, together with those of intermediate results. There is an

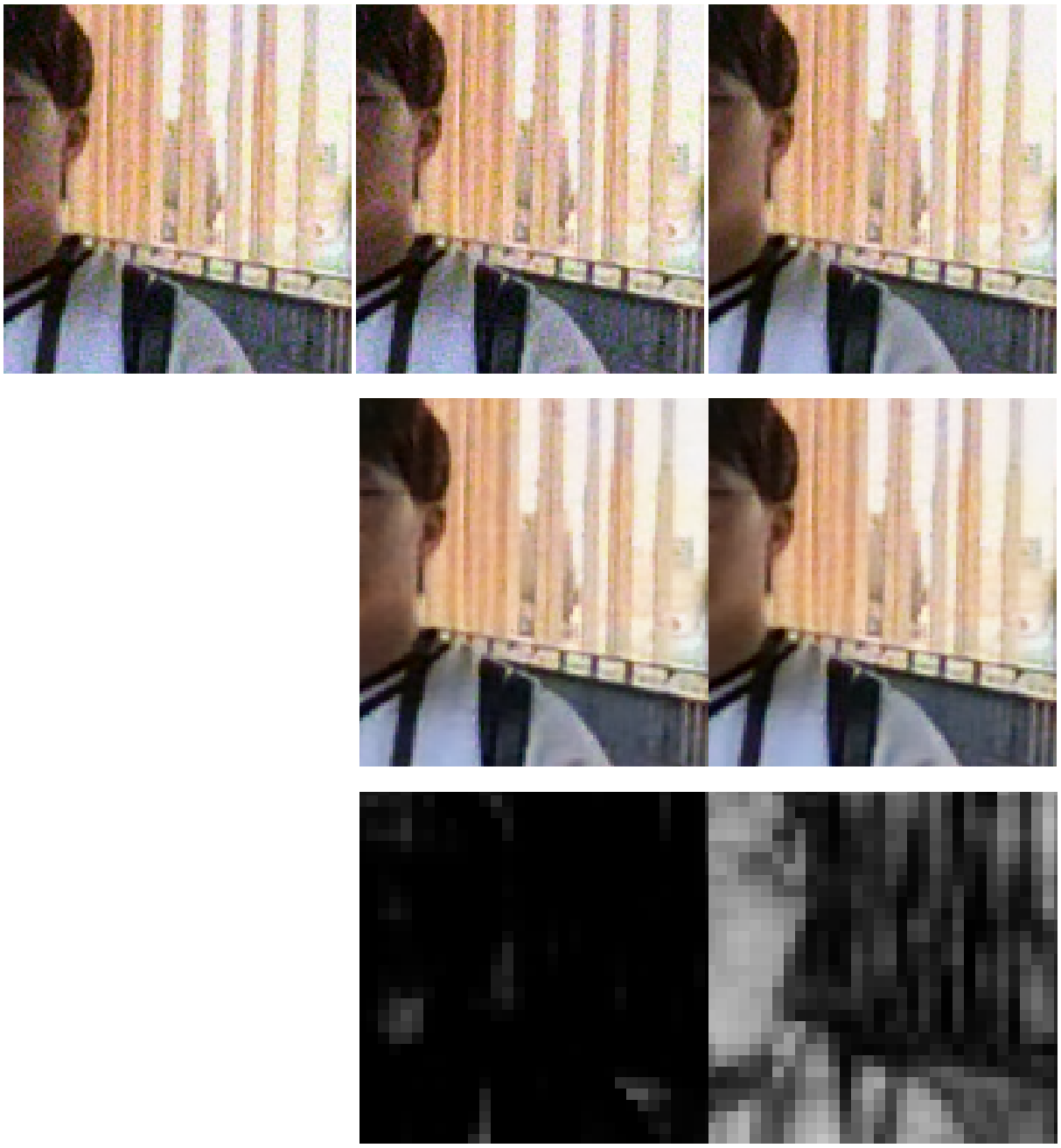

Figure 6: Comparison between results given by using warping or not, ISO 3200. From left to right, top row: noisy frame; temporal fusion without warping; temporal fusion with warping; second row: final result without warping; final result with warping; third row: fusion weight without warping; fusion weight with warping.

Figure 7: Effect of the VST and variance ratio, ISO 3200. From top to bottom: results of the temporal fusion, final results and fusion map. From left to right: only warping; VST added; variance ratio added. Note that in the fusion map we use a larger crop to show a larger context.

Figure 8: Comparison between fusion weights under different settings. From left to right: no VR, no VST, full.

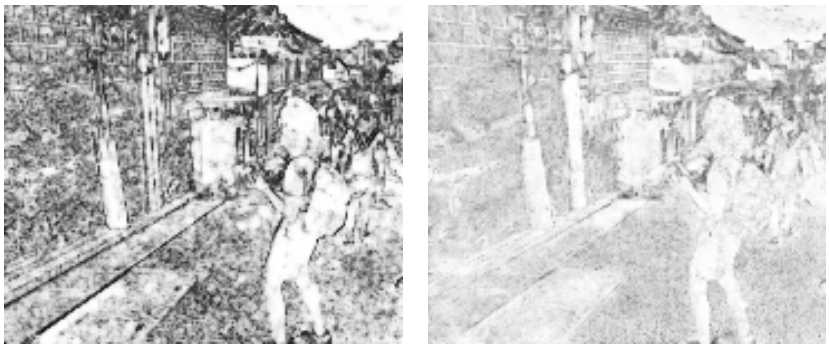

Figure 9: Comparison between refine weights under different configurations. Left: small; right: big.

# of unrolls	1	4	8
final	40.557	40.868	40.916
pre-denoise	36.123	38.661	39.602
fusion	37.756	38.331	38.481

Table 3: Performance of different number of unrollings. ISO3200, small network. Only warping is applied in this comparison.

obvious improvement from 1 unrolling to 4 unrollings, especially for the temporal fusion result, since we get to recurrent denoising from single image denoising. However, the improvement is much smaller if we increase the number of unrollings from 4 to 8.

5 Conclusion

In this paper, we propose an implementation of the EMVD method, a lightweight and interpretable recurrent video denoising CNN. Moreover, we proposed to improve this method in three ways, including applying VST, incorporating the variance of temporal fusion in the computation of the fusion weights, and motion compensation. Among these, the latter makes significant improvement, which stresses the importance of using temporal redundancy in video processing. While the other modifications do not improve the final PSNR, they have a positive influence on the temporal average.

References

[1] P. Arias and J.-M. Morel, *Kalman filtering of patches for frame-recursive video denoising*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[2] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, *Unprocessing images for learned raw denoising*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[3] A. Buades, B. Coll, and J.-M. Morel, *Denoising image sequences does not require motion estimation*, in IEEE Conference on Advanced Video and Signal Based Surveillance, 2005., IEEE, 2005, pp. 70–74.

[4] A. Buades and J. Duran, *Cfa video denoising and demosaicking chain via spatio-temporal patch-based filtering*, IEEE Transactions on Circuits and Systems for Video Technology, 30 (2019), pp. 4143–4157.

[5] 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, IEEE, 2012, pp. 2392–2399.

[6] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock, *An introduction to total variation for image analysis*, in Theoretical foundations and numerical methods for sparse recovery, de Gruyter, 2010, pp. 263–340.

[7] X. Chen, L. Song, and X. Yang, *Deep rnns for video denoising*, in Applications of digital image processing XXXIX, vol. 9971, international Society for optics and Photonics, 2016, p. 99711T.

[8] M. Claus and J. van Gemert, *Videnn: Deep blind video denoising*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[9] R. R. Coifman and D. L. Donoho, *Translation-invariant de-noising*, in Wavelets and statistics, Springer, 1995, pp. 125–150.

[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Image denoising by sparse 3-d transform-domain collaborative filtering*, IEEE Transactions on image processing, 16 (2007), pp. 2080–2095.

[11] A. Davy, T. Ehret, J.-M. Morel, P. Arias, and G. Facciolo, *A non-local cnn for video denoising*, in 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 2409–2413.

[12] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, *Long-term recurrent convolutional networks for visual recognition and description*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 2625–2634.

[13] W. Du, Y. Wang, and Y. Qiao, *Rpan: An end-to-end recurrent pose-attention network for action recognition in videos*, in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3725–3734.

[14] T. Ehret, J.-M. Morel, and P. Arias, *Non-local kalman: A recursive video denoising algorithm*, in 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE, 2018, pp. 3204–3208.

[15] D. Fuoli, S. Gu, and R. Timofte, *Efficient video super-resolution through recurrent latent space propagation*, in 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), IEEE, 2019, pp. 3476–3485.

[16] S. Gu, L. Zhang, W. Zuo, and X. Feng, *Weighted nuclear norm minimization with application to image denoising*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 2862–2869.

[17] Y. Huang, W. Wang, and L. Wang, *Bidirectional recurrent convolutional networks for multi-frame super-resolution*, Advances in neural information processing systems, 28 (2015), pp. 235–243.

[18] T. Isobe, X. Jia, S. Gu, S. Li, S. Wang, and Q. Tian, *Video super-resolution with recurrent structure-detail network*, in European Conference on Computer Vision, Springer, 2020, pp. 645–660.

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

[20] J. Liang, J. Cao, Y. Fan, K. Zhang, R. Ranjan, Y. Li, R. Timofte, and L. Van Gool, *Vrt: A video restoration transformer*, arXiv preprint arXiv:2201.12288, (2022).

[21] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, *Swinir: Image restoration using swin transformer*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 1833–1844.

[22] M. Maggioni, Y. Huang, C. Li, S. Xiao, Z. Fu, and F. Song, *Efficient multi-stage video denoising with recurrent spatio-temporal fusion*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3466–3475.

[23] S. Nah, S. Baik, S. Hong, G. Moon, S. Son, R. Timofte, and K. Mu Lee, *Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[24] J. S. Pérez, E. Meinhardt-Llopis, and G. Facciolo, *Tv-l1 optical flow estimation*, Image Processing On Line, 2013 (2013), pp. 137–150.

[25] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: nonlinear phenomena, 60 (1992), pp. 259–268.

[26] M. S. Sajjadi, R. Vemulapalli, and M. Brown, *Frame-recurrent video super-resolution*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6626–6634.

[27] M. Tassano, J. Delon, and T. Veit, *Dvdnet: A fast network for deep video denoising*, in 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 1805–1809.

[28] ——, *Fastdvdnet: Towards real-time deep video denoising without flow estimation*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1354–1363.

[29] H. Yue, C. Cao, L. Liao, R. Chu, and J. Yang, *Supervised raw video denoising with a benchmark dataset on dynamic scenes*, in CVPR, 2020.

[30] C. Zach, T. Pock, and H. Bischof, *A duality based approach for realtime tv-l 1 optical flow*, in Joint pattern recognition symposium, Springer, 2007, pp. 214–223.

[31] 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 (2017), pp. 3142–3155.

[32] K. Zhang, W. Zuo, and L. Zhang, *Ffdnet: Toward a fast and flexible solution for cnn-based image denoising*, IEEE Transactions on Image Processing, 27 (2018), pp. 4608–4622.