Kalman filtering of patches for frame-recursive video denoising

Pablo Arias
CMLA, CNRS, ENS Paris-Saclay
pablo.arias@cmla.ens-cachan.fr

Jean-Michel Morel
CMLA, CNRS, ENS Paris-Saclay
morels@cmla.ens-cachan.fr

Abstract

A frame recursive video denoising method computes each output frame as a function of only the current noisy frame and the previous denoised output. Frame recursive methods were among the earliest approaches for video denoising. However in the last fifteen years they have been used almost exclusively for real-time applications with denoising performance far from being state-of-the-art. In this work we propose a simple frame recursive method which is fast, has a low memory complexity and achieves results competitive with more complex state-of-the-art methods that require processing several input frames for producing each output frame. Furthermore, in terms of visual quality, the proposed approach is able to recover many details that are missed by most non-recursive methods. As an additional contribution we also propose an off-line post-processing of the denoised video that boosts denoising quality and temporal consistency.

1. Introduction

Denoising is a fundamental problem in image and video processing, and a necessary step in almost any imaging pipeline as the RAW data captured by the sensor is unavoidably corrupted by noise. After decades of research the field has evolved significantly. So much so, that in the case of still images it is difficult for a new method to obtain a significant improvement over the state of the art (for white additive Gaussian noise). Quite different is the situation in video denoising: There is still a lot of room for improvement and some approaches have been little explored.

Although evidently related, the problems of image and video denoising have important differences. The temporal consistency of videos facilitates the denoising as it provides a strong source of redundancy absent in still images. At the same time it also brings a new challenge: the output of the denoising algorithm is required to have the same temporal consistency, a key element for the perceived quality of a video [36, 28]. In addition, video denoising algorithms need to process a much larger amount of data, which results in more exigent design constraints for practical methods.

Currently, the best results are obtained by patch-based methods [13, 29, 3, 17, 9, 38] that benefit from the fact that video patches have several similar peers. They group together similar patches in highly redundant sets which can therefore be denoised effectively.

Convolutional neural networks (CNN) have been successfully applied to image denoising (e.g. [42, 43, 35]) but their application to video denoising has been limited so far. In [12] a recurrent architecture is proposed, but the results are below the state-of-the-art. Recently, [15] reported state-of-the-art results with a hybrid method which applies a CNN to an image of "non-local features": the values of the centers of the most similar patches at each location. Some works have tackled the related problem of burst denoising [21, 30], but do not report results on video.

All these methods have in common that they produce an output frame u_t at time t as a function of a number of input noisy frames f_s in a temporal vicinity: $u_t = \mathcal{D}(f_{t-h}, \dots, f_{t+h})$.[1] In spite of their good results, they have some disadvantages: (i) They tend to be computationally costly, since they have to process a volume to produce each frame; (ii) They have a latency of h frames (this could be avoided by using only past frames, possibly with a penalty in the output quality); and (iii) They lack of a direct way to control the temporal consistency of the result. Therefore they are only suited for off-line processing.

An alternative approach is given by recursive algorithms where the output at t depends on the previous output: $u_t = \mathcal{D}(f_t, u_{t-1})$. These were among the earliest methods for video denoising [7], and today are used mainly as a choice for algorithms that need to operate at real-time frame rates. The works [45, 31] combine a spatial bilateral filter with a temporal Kalman filter which is applied when no motion is detected. Recursive versions of the non-local means algorithm [8] were proposed in [23, 1]. Recently, [16] presented a multi-resolution approach for real-time video denoising. The focus of these works is reducing the number of opera-

[1] The exception are the recurrent networks [12] and [21]. The former does not achieve state-of-the-art results and the latter is for burst denoising and cannot be directly applied to video.

Figure 1. Comparison of results obtained for $\sigma = 40$. From the top: original frame, noisy, results of V-BM3D [13], VNLB [3], FNLK [18], the proposed BNLK and BNLK+S.

tions per pixel. Their results are far from the state-of-the-art in terms of denoising quality. For instance, the results reported in [1, 16] are 2dB below V-BM3D [13].

An exception is the non-local Kalman filter proposed in [18], where Kalman filters act on patches instead of single pixels as in [45, 31]. This method achieves the best results reported in the literature for a recursive method, close to those of V-BMxD [13, 29]. However the method is still costly both in terms of computation and memory usage. The reason is that that frame u_t is produced as a function $u_t = \mathcal{D}(f_t, u_{t-1}, M_{t-1})$, of the previous frame u_{t-1} and an auxiliary memory M_{t-1} containing the state of all patch Kalman filters. For patches of size $\sqrt{d} \times \sqrt{d}$, the patch Kalman filters require storing and inverting covariance matrices of size $d \times d$. As a result, the method is two to three times slower than V-BM3D.

Algorithm 1: Recursive video filtering

input: noisy video f, noise level σ
output: filtered video $\widehat{u}$

$\widetilde{u}_1 \leftarrow \texttt{nldct}(\widehat{f}_1, \sigma)$
for $t = 2, \dots, T$ **do**
  $\boldsymbol{v}_t^b \leftarrow \texttt{compute-optical-flow}(f_t, \widehat{u}_{t-1}, \sigma)$
  $\widehat{u}_{t-1}^w, \kappa_t^b \leftarrow \texttt{warp-bicubic}(\widehat{u}_{t-1}, \boldsymbol{v}_t^b)$
  $\widehat{g}_t \leftarrow \texttt{nlkalman-filter}(f_t, \widehat{u}_{t-1}^w, \kappa_t^b, \sigma)$
  $\widehat{u}_t \leftarrow \texttt{nlkalman-filter}(f_t, \widehat{u}_{t-1}^w, \kappa_t^b, \widehat{g}_t, \sigma)$
end

Algorithm 2: Recursive video smoothing

input: filtered video $\widehat{u}$, noise level σ
output: smoothed video $\widetilde{u}$

$\widetilde{u}_T \leftarrow \widehat{u}_T$
for $t = T-1, \dots, 1$ **do**
  $\boldsymbol{v}_t^f \leftarrow \texttt{compute-optical-flow}(\widehat{u}_t, \widetilde{u}_{t+1}, \sigma)$
  $\widetilde{u}_{t+1}^w, \kappa_t^f \leftarrow \texttt{warp-bicubic}(\widetilde{u}_{t+1}, \boldsymbol{v}_t^f)$
  $\widetilde{u}_t \leftarrow \texttt{nlkalman-smoother}(\widehat{u}_t, \kappa_t^f, \widetilde{u}_{t+1}^w, \sigma)$
end

Contribution. In this work we propose a purely frame recursive video denoising method. By this we mean that the output at frame t only depends on the current noisy frame and the previous denoised frame. Our work is inspired by the non-local Kalman filter of [18], but we introduce two approximations that result in a simpler and faster method. Firstly, we assume dynamic models for patches to be diagonal in the DCT basis, which reduces the computational complexity from $\mathcal{O}(d^3)$ to $\mathcal{O}(d)$. Secondly, the covariances of the previous state are estimated from the patches of the previous denoised frame u_{t-1}, thus eliminating the need of the auxiliary state memory M_{t-1}. These approximations cause a substantial drop in performance, but we are able to compensate for it by applying two denoising iterations per-frame. Our results are on par with those of [18], and are even better for higher values of noise. In terms of visual quality, our method shows less temporal consistency than [18], but in turn produces sharper frames and recovers more details. Some examples are shown in Figure 1. To the best of our knowledge, these are the best video denoising results reported with a purely frame-recursive method so far.

As an additional contribution, we also propose a "smoothing" algorithm suitable for an off-line setting. Based on the Rauch-Tung-Striebel (RTS) smoother [32], it processes the filtering output video backwards from the last frame to the first. The smoothing boosts the quality of the result (PSNR increases almost 0.7dB) and greatly improves the temporal consistency.

All results and the code are available at `http://dev.ipol.im/~pariasm/bnlk/`.

2. Overview

Algorithm 1 shows the main elements of the non-local Kalman filtering method for a video with T frames. The noisy input video is denoted by f, and a noisy frame at time t by f_t. By u we denote the unknown clean video, by $\widehat{u}$ the filtered video and by $\widetilde{u}$ the smoothed one. We assume white additive Gaussian noise of standard deviation σ, i.e. $f_t(x) = u_t(x) + r_t(x)$, where $r_t(x) \sim \mathcal{N}(0, \sigma^2)$ and x represents a pixel location in the frame domain.

The first frame is denoised with a still image denoising algorithm. We use a simplified version of BM3D [14] called non-local DCT [2], although other methods could be used as well. For the remaining frames we compute the backwards optical flow, $\boldsymbol{v}_t^b$ which estimates the motion from frame t to frame $t-1$. The backwards flow is used to warp the previous denoising output $\widehat{u}_{t-1}$. An occlusion mask κ_t^b is computing during warping essentially as the locations of strong discontinuities in the optical flow.

We apply two filtering iterations. The first iteration takes as inputs the current noisy frame f_t, the warped previous denoising output u_{t-1}^w and the occlusion mask κ_t^b. Its output, denoted by $\widehat{g}_t$, is then used as a guide for the second iteration (similar two iteration procedures are common among denoising methods [14, 26]). Note that the only information required from the previous frame is $\widehat{u}_{t-1}$.

A pseudo-code for the non-local Kalman smoothing is shown in Algorithm 2. It is a recursive method as well, but it runs backwards. It is initialized by setting $\widetilde{u}_T = \widehat{u}_T$. Similar to the filtering, for the remaining frames we compute the forward optical flow and occlusions from t to $t+1$, and warp the previous output of the smoothing recursion $\widetilde{u}_{t+1}$. For the smoother, there is no need to run two iterations as during filtering.

In the following sections we give detailed explanations of all these elements. But before, we give a brief revision of the theory of the Kalman filtering.

3. Inference in dynamical Gaussian models

The Kalman filter solves a Bayesian estimation problem for a sequence of random vectors $\boldsymbol{p}_t$ from noisy observations $\boldsymbol{q}_t$. It corresponds to the specific case in which these sequences follow a dynamic linear Gaussian model [5]. We consider a particularly simple model by assuming observation and state transition matrices equal to identity:

$$\begin{cases} \boldsymbol{p}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, P_0), & \\ \boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \boldsymbol{w}_t, & \text{where} \quad \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, W_t), \\ \boldsymbol{q}_t = \boldsymbol{p}_t + \boldsymbol{r}_t, & \text{where} \quad \boldsymbol{r}_t \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 I), \end{cases} \tag{1}$$

where the last two equations hold for $t = 1, 2, \dots$, and σ^2 is the variance of the noise (assumed to be known).

In the Kalman filter literature, the unknown vector $\boldsymbol{p}_t$ is called the *state*. The initial state $\boldsymbol{p}_0$ is Gaussian with mean $\boldsymbol{\mu}_0$ and covariance P_0. The second equation is the *state transition equation* and it describes the change in the state from $t-1$ to t. The evolution of the state is modeled by a sequence of Gaussian random vectors $\boldsymbol{w}_t \sim \mathcal{N}(\mathbf{0}, W_t)$. The *transition covariance matrices* W_t determine the modes of random variation in the state. The last equation describes the observation $\boldsymbol{q}_t$ as a noisy linear function of the state.

The parameters of this model are the state transition covariances W_t and the mean and variance of the first state. Assuming that the parameters are known, during inference at time t we are interested in the posterior distribution of the state $\boldsymbol{p}_t$ given all previous observations, i.e. $p(\boldsymbol{p}_t|\boldsymbol{q}_t, \dots, \boldsymbol{q}_1)$. This posterior distribution, sometimes referred to as the *filtering posterior*, is Gaussian, with mean $\widehat{\boldsymbol{p}}_t$ and covariance P_t. The Kalman filter equations [25] are a set of recursive equations which allow to compute $\widehat{\boldsymbol{p}}_t, P_t$ in terms of $\widehat{\boldsymbol{p}}_{t-1}, P_{t-1}$.

Several denoising methods have proposed Gaussian models as priors for image and video patches [44, 40, 11, 26, 3, 24]. In [18], the authors adopt the dynamic model (1) and apply it to video patches: $\boldsymbol{q}_t$ represents a patch from the observed noisy video and the state $\boldsymbol{p}_t$ is the unknown clean patch. Patches are squares of size $\sqrt{d} \times \sqrt{d}$, thus the identity I and the covariances P_0 and W_t have size $d \times d$.

Kalman filter. The Kalman filter equations corresponding to the model (1) are as follows:

$$\begin{aligned} K_t &= (P_{t-1} + W_t)(P_{t-1} + W_t + \sigma^2 I)^{-1}, \\ \widehat{\boldsymbol{p}}_t &= (I - K_t)\widehat{\boldsymbol{p}}_{t-1} + K_t \boldsymbol{q}_t, \\ P_t &= (I - K_t)(P_{t-1} + W_t)(I - K_t)^T + \sigma^2 K_t^2. \end{aligned} \tag{2}$$

We can gain intuition on this set of equations by expressing them in the basis of eigenvectors of $P_{t-1} + W_t$. Let $\boldsymbol{u}_{t,1}, \dots, \boldsymbol{u}_{t,d}$, and $\lambda_{t,1}, \dots, \lambda_{t,d}$ be the eigenvectors and eigenvalues of $P_{t-1} + W_t$. Then it can be seen that the eigenvectors K_t are also $\boldsymbol{u}_{t,i}$ and its eigenvalues are $s_{t,i} = \lambda_{t,i}/(\lambda_{t,i} + \sigma^2)$. It then follows that the projection of $\widehat{\boldsymbol{p}}_t$ over the ith eigenvector is computed as

$$\langle \widehat{\boldsymbol{p}}_t, \boldsymbol{u}_{t,i} \rangle = (1 - s_{t,i}) \langle \widehat{\boldsymbol{p}}_{t-1}, \boldsymbol{u}_{t,i} \rangle + s_{t,i} \langle \boldsymbol{q}_t, \boldsymbol{u}_{t,i} \rangle.$$

At time t, the Kalman filter computes the state posterior mean $\widehat{\boldsymbol{p}}_t$ as a multivalued convex combination between the new observation $\boldsymbol{q}_t$ and $\widehat{\boldsymbol{p}}_{t-1}$. The matrix K_t is called the *Kalman gain*, and controls the weights given to the previous state and to the observation. The Kalman filter provides both the mean $\widehat{\boldsymbol{p}}_t$ and covariance P_t which define the filtering posterior $p(\boldsymbol{p}_t|\boldsymbol{q}_t, \dots, \boldsymbol{q}_1)$. The mean is an optimal estimator for the unknown state. It is both the MMSE and MAP estimator in an online setting, i.e. given all observations until t.

RTS smoother. In an offline setting the full sequence of observations $\boldsymbol{q}_1, \dots, \boldsymbol{q}_T$ is available. In this case the relevant posterior is $p(\boldsymbol{p}_t|\boldsymbol{q}_T, \dots, \boldsymbol{q}_1)$, which is Gaussian with mean $\widetilde{\boldsymbol{p}}_t$ and covariance $\widetilde{P}_t$. Interestingly, this mean and covariance can be computed with the following backwards recursion on the results of the Kalman filter [5]:

$$\begin{aligned} J_t &= P_t(P_t + W_{t+1})^{-1}, \\ \widetilde{\boldsymbol{p}}_t &= (I - J_t)\widehat{\boldsymbol{p}}_t + J_t \widetilde{\boldsymbol{p}}_{t+1}, \\ \widetilde{P}_t &= P_t + J_t(\widetilde{P}_{t+1} - P_t - W_{t+1})J_t. \end{aligned} \tag{3}$$

This is called smoothing, and these equations are the *Rauch-Tung-Striebel (RTS) smoother* [32]. The matrix J_t serves a similar role as the Kalman gain, performing a multivalued weighted average between the current filtering output $\widehat{\boldsymbol{p}}_t$ and the future smoothed state $\widetilde{\boldsymbol{p}}_{t+1}$. In the smoothing equations, the smoothed means $\widetilde{\boldsymbol{p}}_t$ can be computed without computing the covariances $\widetilde{P}_t$ (which is not the case for the filtering). However, they require all the filtering covariances P_t, which have to be kept in memory if smoothing is going to be applied.

4. Kalman filters for video patches

The Kalman filter is a well-known tool and has been applied to many video processing problems: such as denoising [45, 31, 23], inpainting [10], optical flow [6, 37] and super-resolution [19, 20] problems.In most of these cases, the states are the individual pixel values. Some works treat the image as the state assuming diagonal transition and noise covariance matrices [20]. Our model is built upon the work of [18], which proposes a Kalman filter for video patches. To better motivate our contribution, we start by reviewing the approach of [18].

4.1. Review of [18]

This method works by building groups of similar patches and following them throughout the video. For this, the motion in the video is estimated using an optical flow algorithm. The patches in each group are assumed to be independent realizations of a dynamic model such as (1).

A group is created by picking a reference patch and searching for the n most similar patches in a neighborhood around it within the frame. Suppose $\boldsymbol{q}_{0,1}, \dots, \boldsymbol{q}_{0,n}$ is one of such groups created in the first frame $t = 0$. These patches are denoised using [26, 27]. The denoised patches are aggregated in the output frame, and are used to initialize n Kalman filters. The initial state means $\widehat{\boldsymbol{p}}_{0,i}$ are set to be the denoised patches. The state and transition covariances P_0 and W_0 are initialized as the sample covariance matrix of the patches in the group.

After denoising the first frame, we end up with a number of groups of patches such that the union of all these patches fully covers the frame. For each group two covariance matrices need to be stored: the state and transition covariances,

P_0 and W_0. The database with the groups of patches is denoted by M_0.

For denoising frame t, we receive M_{t-1} from the previous frame, with the groups of patches and their covariance matrices P_{t-1} and W_{t-1}. Each of such groups contains the state estimate for n patch Kalman filters $\widehat{\boldsymbol{p}}_{t-1,i}$. The forward optical flow at $t-1$, $\boldsymbol{v}^f_{t-1}$ computes the displacements from frame $t-1$ to t and is used to find the position at frame t of the patches in the group: if the ith patch is centered at pixel x_i at $t-1$, the authors assume that it will be centered at $x_i + \boldsymbol{v}^f_{t-1}(x_i)$ in frame t (i.e. the patch moves with a translation of $\boldsymbol{v}^f_{t-1}(x_i)$). The noisy observation $\boldsymbol{q}_{t,i}$ is taken as the patch from the noisy video centered $x_i + \boldsymbol{v}^f_{t-1}(x_i)$.

Before updating the estimated patches and their state covariance with the Kalman filter (2) the transition covariance W_t needs to be estimated. The authors exploit the following relation

$$\mathbb{E}\{(\boldsymbol{q}_t - \boldsymbol{q}_{t-1})(\boldsymbol{q}_t - \boldsymbol{q}_{t-1})^T\} = W_t + 2\sigma^2 I, \quad (4)$$

obtained by substituting $\boldsymbol{q}_{t-1} = \boldsymbol{p}_{t-1} + \boldsymbol{r}_{t-1}$ and $\boldsymbol{q}_t = \boldsymbol{p}_{t-1} + \boldsymbol{w}_t + \boldsymbol{r}_t$ on the left hand side. Then W_t is estimated as the sample covariance matrix of the innovations $\boldsymbol{q}_t - \boldsymbol{q}_{t-1}$:

$$\widehat{W}_t = (1-\delta)\widehat{W}_{t-1} + \delta\left(\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{q}_{t,i} - \boldsymbol{q}_{t-1,i})(\boldsymbol{q}_{t,i} - \boldsymbol{q}_{t-1,i})^T - 2\sigma^2 I\right)$$

where $\boldsymbol{q}_{t,i}$ is the tth observation for the ith patch in the group, with $i = 1, \ldots, n$.[2] The transition covariance of the previous frame $\widehat{W}_{t-1}$ is added with a weight $1-\delta$ to increase the temporal consistency.

With the estimated $\widehat{W}_t$ the patches $\widehat{\boldsymbol{p}}_{t,i}$ are estimated running the Kalman filter. The updated state means are stored in M_t and are aggregated on the output frame. After processing all groups of patches in M_{t-1} some areas in the frame might be left uncovered. This typically happens in dis-occluded areas since no patch from $t-1$ moves to those areas. To cover them, new groups are created and initialized via the spatial denoising [26] as for $t = 0$. These new groups together with the updated pre-existing ones are stored in M_t and passed to frame $t+1$.

Although this approach produces good results with strong temporal consistency, it has important drawbacks. Processing each group is costly, as it requires inverting a $d \times d$ matrix for computing the Kalman gain. Furthermore, the auxiliary memory M_t can be quite large, since for each group two $d \times d$ matrices need to be stored in addition to the n patch mean states $\widehat{\boldsymbol{p}}_{t,i}$. As noted by the authors, this is aggravated by the fact that the number of groups grows at each new frame, since new groups are created to process dis-occluded areas. This negatively affects the running time and memory usage, which in some sense defeats the purpose of using a recursive approach.

Next we describe how we can overcome these drawbacks by the use of recursive Bayesian filters which do not need to keep track of the covariance matrices P_t and W_t and operate in the DCT domain.

4.2. Kalman-like recursive Bayesian filters

At time t the Kalman filter updates the filtering posterior $p(\boldsymbol{p}_{t-1}|\boldsymbol{q}_{t-1}, \ldots, \boldsymbol{q}_1) = \mathcal{N}(\widehat{\boldsymbol{p}}_{t-1}, P_{t-1})$ to account for the new observation $\boldsymbol{q}_t$. All the information we need from the past observations is captured by $\widehat{p}_{t-1}, P_{t-1}$. Therefore, the new filtering posterior can be computed from the following model involving only $\boldsymbol{p}_{t-1}, \boldsymbol{p}_t$ and $\boldsymbol{q}_t$:

$$\begin{cases} \boldsymbol{p}_{t-1} \sim \mathcal{N}(\widehat{\boldsymbol{p}}_{t-1}, P_{t-1}), & \\ \boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \boldsymbol{w}_t, & \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, W_t), \\ \boldsymbol{q}_t = \boldsymbol{p}_t + \boldsymbol{r}_t & \boldsymbol{r}_t \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 I). \end{cases} \quad (5)$$

If we are running the Kalman filter, we obtain $\widehat{\boldsymbol{p}}_{t-1}$ and P_{t-1} from the previous iteration. Since we want to avoid having to store covariance matrices, we consider them model parameters to be estimated from data. We assume that we have available a set of n observations $\boldsymbol{p}_{t-1,i}$ and $\boldsymbol{q}_{t,i}$ which are independent samples of the model. We then estimate the parameters as follows:

$$\widehat{P}_{t-1} = \frac{1}{n}\sum_{i=1}^{n}, (\boldsymbol{p}_{t-1,i} - \widehat{\boldsymbol{p}}_{t-1,i})(\boldsymbol{p}_{t-1,i} - \widehat{\boldsymbol{p}}_{t-1,i})^T, \quad (6)$$

$$\widehat{W}_t = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{q}_{t,i} - \boldsymbol{p}_{t-1,i})(\boldsymbol{q}_{t,i} - \boldsymbol{p}_{t-1,i})^T - \sigma^2. \quad (7)$$

where $\widehat{\boldsymbol{p}}_{t-1} = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{p}_{t-1,i}$. The patches $\boldsymbol{p}_{t-1,i}$ will be extracted from the previous denoising output (warped by the optical flow) $\widehat{u}^w_{t-1}$.

The equations for inference under this model are still the Kalman filter equations (2). However, there is a subtle difference between our filtering equations and the ones from [18]. In their groups of similar patches, n Kalman filters are running in parallel. They share the same covariances P_t, W_t, but have different state means $\widehat{\boldsymbol{p}}_{t,i}$. As a result, the ith patch estimated at time t corresponds to a weighted combination between the ith observation and the ith previous state, i.e.

$$\widehat{\boldsymbol{p}}_{t,i} = (I - K_t)\widehat{\boldsymbol{p}}_{t-1,i} + K_t\boldsymbol{q}_{t,i}. \quad (8)$$

Instead, we do not consider previous patches $\boldsymbol{p}_{t-1,i}$ to be the state means, but different realizations of the previous state. As a result, all the patches at t are estimated with the same previous mean:

$$\widehat{\boldsymbol{p}}_{t,i} = (I - K_t)\widehat{\boldsymbol{p}}_{t-1} + K_t\boldsymbol{q}_{t,i}. \quad (9)$$

[2] The resulting $\widehat{W}_t$ might not be positive semi-definite. Negative eigenvalues are avoided by setting them to zero.

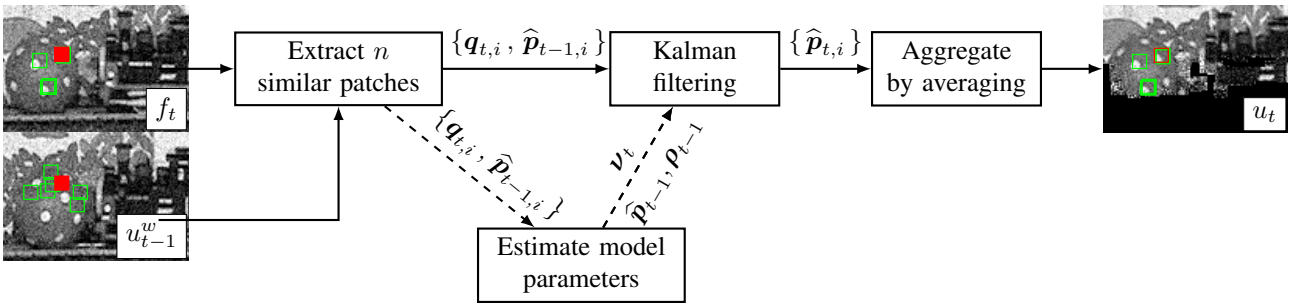

Figure 2. Overview of the proposed non-local Kalman filtering method.

This modification has important consequences. While the denoising in [18] is mainly temporal, Eq. (9) has a component of spatial denoising, as $\widehat{\boldsymbol{p}}_{t-1}$ is estimated as a non-local average of similar patches. This can be useful, but in excess it may lead to loss of detail. To control the spatial denoising, we use $m \leqslant n$ patches to estimate $\widehat{\boldsymbol{p}}_{t-1}$.

RTS-like recursive smoother. In a similar way, the model of the RTS smoother of Eq. (3) can be adapted to include the state covariances P_t and $\widetilde{P}_{t+1}$ as parameters that are estimated from a set of observations $\widehat{\boldsymbol{p}}_{t,i}$ and $\widetilde{\boldsymbol{p}}_{t+1,i}$. We omit these details for reasons of space.

4.3. Working in the DCT domain

As an additional simplification, we assume that the covariances W_t, C_{t-1} are diagonal on the DCT basis, i.e.

$$W_t = U\text{Diag}(\boldsymbol{\nu}_t)U^T, C_{t-1} = U\text{Diag}(\boldsymbol{\rho}_{t-1})U^T,$$

where U denotes the $d \times d$ matrix whose columns are the DCT basis vectors. Here $\boldsymbol{\nu}_t = (\nu_t(0), ..., \nu_t(d-1))$ is the vector of transition variances along each DCT direction: i.e. $\nu_t(j)$ represents how much the jth DCT component of the state can change from $t-1$ to t. Similarly $\boldsymbol{\rho}_{t-1} = (\rho_{t-1}(0), ..., \rho_{t-1}(d-1))$ contains the variances of the previous state along each DCT component.

This assumption greatly reduces the computational cost of the filtering, as it decouples over the DCT components. Let us denote the DCTs of the mean state and its noisy observation as: $\widehat{\boldsymbol{\alpha}}_t = \texttt{DCT}(\widehat{\boldsymbol{p}}_t) = U^T\widehat{\boldsymbol{p}}_t$ and $\boldsymbol{\beta}_t = \texttt{DCT}(\boldsymbol{q}_t) = U^T\boldsymbol{q}_t$. We can express the filtering equations for the ith DCT component as follows:

$$\begin{aligned} s_t(j) &= \frac{\rho_{t-1}(j) + \nu_t(j)}{\rho_{t-1}(j) + \nu_t(j) + \gamma\sigma^2}, \\ \widehat{\alpha}_t(j) &= (1 - s_t(j))\widehat{\alpha}_{t-1}(j) + s_t(j)\beta_t(j) \\ \rho_t(j) &= (1 - s_t(j))^2(\rho_{t-1}(j) + \nu_t(j)) + s_t(j)^2\sigma^2. \end{aligned} \tag{10}$$

Note that we have added a noise multiplier factor γ in the computation of the Kalman gain $s_t(j)$ as a parameter to control the filtering strength. A higher value of γ results in a lower value for $s_t(j)$ and thus a higher attachment to the previous state. The updated state can be computed by inverting the DCT transform $\widehat{\boldsymbol{p}}_t = \texttt{IDCT}(\widehat{\boldsymbol{\alpha}}_t) = U\widehat{\boldsymbol{\alpha}}_t$.

The computation of the variances is also simplified, since instead of having to compute a $d \times d$ covariance matrix, we need to estimate d scalar variances:

$$\widehat{\rho}_{t-1}(j) = \frac{1}{n}\sum_{i=1}^{n}(\alpha_{t-1,i}(j) - \widehat{\alpha}_{t-1,i}(j))^2, \tag{11}$$

$$\widehat{\nu}_t(j) = \left(\frac{1}{n}\sum_{i=1}^{n}(\beta_{t,i}(j) - \alpha_{t-1,i}(j))^2 - \sigma^2\right)_+, \tag{12}$$

where $(x)_+ = x$ if x is positive and 0 otherwise. Note that we need to compute the DCT of the n similar patches $\boldsymbol{q}_{t,i}$ and their corresponding previous states $\boldsymbol{p}_{t-1,i}$.

5. Filtering algorithm

We now describe the proposed method.

5.1. Optical flow and warping

The optical flow is a critical component of the method. Any optical flow algorithm could be used. For our experiments, we used the implementation [34] of the TV-L1 optical flow introduced in [41]. This is a common choice in the denoising literature [18, 3, 9] since it has a low running time (GPU versions of this code work in real-time) and is quite robust to noise (as opposed to some neural networks which have been trained on clean data). The TV-L1 method is computed in a coarse-to-fine multiscale framework. At each scale the upscaled result from the coarser scale is refined. The last scale is the most costly one. We skip the computation of the finest scale, as this results in faster running times and a more regular optical flow. Note that the optical flow is computed between the noisy frame at time t and the previous denoising output $\widehat{u}_{t-1}$, which reduces the impact of the noise.

The optical flow is used only to warp the previous filtered frame $\widehat{u}_{t-1}$ and register it with the current noisy frame:

$$\widehat{u}^w_{t-1}(x) = \widehat{u}_{t-1}(x + \boldsymbol{v}^b_t(x)),$$

where bicubic interpolation is used. Some pixels in the warped frame u^w_{t-1} are labeled as undefined in a mask κ^b_{t-1} which is returned alongside with the warped image. Undefined pixels are those which have at least one pixel in their bicubic interpolation stencil which is not visible at $t-1$. There are three reasons for this:

(i) When $t = 0$, i.e. we are denoising the first frame of the video. In this case all pixels in u^w_{t-1} are undefined.

(ii) When the pixel is out of the frame domain. This happens for example close to the frame boundaries when there is camera motion and some pixels enter the video at t that were not visible at $t-1$.

(iii) When the pixel is flagged as occluded. Occlusions are determined simply as pixels where the divergence of the optical flow is above a threshold: $|\text{div}(\boldsymbol{v}_t)(x)| \geq \tau_k$. We compute the partial derivatives of the flow with a standard forward difference approximation. The use of occlusion masks avoids artifacts located around the border of moving objects. A similar approach was used in [4, 9].

The same process is applied during smoothing to warp the smoothed frame $\widetilde{u}_{t+1}$ to the filtered frame $\widehat{u}_t$.

5.2. Non-local Kalman filtering

We now focus on the recursive Bayesian temporal filtering of patches (lines 5 and 6 in Algorithm 1).

We process the frame t by selecting reference patches in raster order with a step of $\sqrt{d}/2$ (thus the number of reference patches is the total number of patches divided by $d/4$). At each location we extract a patch $\boldsymbol{q}_t$ from the noisy frame t and the corresponding patch from the warped previous frame $\widehat{u}^w_t$, which we denote $\boldsymbol{p}_{t-1}$. If any pixel of the previous patch $\boldsymbol{p}_{t-1}$ is flagged as undefined in κ^b_{t-1}, then we use the spatial denoising algorithm to estimate $\boldsymbol{p}_t$ (explained below). If not, we proceed with the temporal filtering. It consists of (1) searching for similar patches, (2) estimating the parameters of the dynamic Gaussian model, (3) estimating the clean patch $\widehat{\boldsymbol{p}}_t$ and (4) aggregating it on the output image. This is shown by the diagram in Figure 2. We now describe each step of the algorithm. In each case we detail first the first iteration, and then explain the modifications for the second iteration with the guide.

Search for similar patches. The similar patches are searched for in a square region of size $w \times w$ centered at the location of the reference patch $\boldsymbol{q}_t$. The patch distance is the L^2 norm (or sum of square differences) between the noisy patches. We then extract the n most similar patches $\boldsymbol{q}_t = \boldsymbol{q}_{t,1}, \dots, \boldsymbol{q}_{t,n}$ and their corresponding patches from the previous warped frame, $\boldsymbol{p}_{t-1,i}$ (excluding those with undefined pixels).[3] In the second iteration the patch distance is computed as the L^2 norm between the patches of the guide. In this case we extract the n most similar patches from the guide as well $\boldsymbol{g}_{t,1}, \dots, \boldsymbol{g}_{t,n}$.

[3]The number of patches used is between 5 and 40. Higher noise levels require increased number of patches.

Parameter estimation. We then compute the DCT of the n similar patches $\boldsymbol{\beta}_{t,i} = U^T \boldsymbol{q}_{t,i}$ and $\boldsymbol{\alpha}_{t,i} = U^T \boldsymbol{p}_{t,i}$, and estimate the parameters of the dynamic model (5) in the DCT domain $\widehat{\boldsymbol{\alpha}}_{t-1}, \widehat{\boldsymbol{\rho}}_{t-1}$ and $\widehat{\boldsymbol{\nu}}_t$ as in equations (11) and (12). If a guide is available we define $\boldsymbol{\beta}_{t,i}$ as the DCT of the patches of the guide $\boldsymbol{g}_{t,i}$ and use them in the computation of $\widehat{\boldsymbol{\nu}}_t$. In this case we do not subtract σ^2 in Eq. (12).

Recursive Bayesian filtering. The m most similar patches (where $m \leqslant n$) are filtered by applying Eqs. (10), followed by the application of the inverse DCT $\widehat{\boldsymbol{p}}_{t,i} = \texttt{IDCT}(\widehat{\boldsymbol{\alpha}}_{t,i})$, for $i = 1, \dots, m$.

Weighted aggregation. The aggregation is the process by which a single output frame is computed from all the estimated patches. Since the patches overlap, a single pixel will be estimated several times, one for each denoised patch that contains the pixel. These estimates will not coincide, and thus they are aggregated by computing a weighted average. The aggregation weights are inversely proportional to the variance of each estimated patch. The variance is computed as the sum of the variances of the DCT components $\sum_{j=1}^{d} \rho_t(j)$. This is a common practice in patch-based denoising [22, 14, 33]. These weights can be shown to be optimal in the MSE sense [22].

5.3. Spatial denoising

If $\boldsymbol{p}_{t-1}(\boldsymbol{x})$ has some if its pixels flagged as undefined we apply a spatial denoising algorithm. Any spatial denoising method could be used. The one we use is a version of the non-local Bayesian method [26] in the DCT domain, which is similar to the non-local Kalman filter described.

We search for n patches $\boldsymbol{q}_t = \boldsymbol{q}_{t,1}, \dots, \boldsymbol{q}_{t,n}$ in a square search window centered at $\boldsymbol{q}_t$. We then assume that the unknown clean patches are distributed following a Gaussian prior, $\boldsymbol{p}_{t,i} \sim \mathcal{N}(\boldsymbol{\mu}, C)$, where C is assumed to be diagonal in the DCT basis, i.e. $C = U\text{Diag}(\boldsymbol{\lambda})U^T$.

We compute the DCT transform of the similar patches, $\boldsymbol{\beta}_{t,i} = \texttt{DCT}(\boldsymbol{q}_{t,i})$. The mean $\boldsymbol{\mu}$ and variances $\boldsymbol{\lambda}$ are estimated from the sample average and variances of set of similar patches along each DCT component, similar to Eqs. (**??**) and (11). The clean patches are estimated via the following Wiener filter of their DCT coefficients:

$$\widehat{\alpha}_{t,i}(j) = (1 - s(j))\widehat{\mu}(j) + s(j)\beta_{t,i}(j),$$

where $s(j) = \widehat{\lambda}(j)/(\widehat{\lambda}(j) + \gamma\sigma^2)$. Here γ is a parameter added to control the denoising strength, similar to the one in Eq. (10). Finally we aggregate the corresponding patches $\widehat{p}_{t,i} = \texttt{IDCT}(\widehat{\alpha}_{t,i})$ on the output image. The aggregation is weighted by the posterior variance, given by $\sum_{j=1}^{d} s(j)\widehat{\lambda}(j)$. The spatial denoising is iterated too using

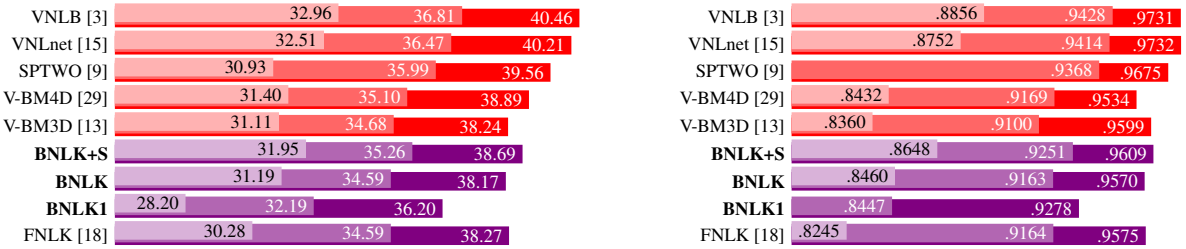

Figure 3. Average PSNR and SSIM over 7 grayscale sequences $960 \times 540 \times 100$, for noise levels $\sigma = 40, 20, 10$ (bars from left to right). The methods in red are non-recursive, and in purple are recursive. The methods proposed in this work are shown in boldface.

the first iterate as guide. As before, the guide is used to compute the patch distances and the to estimate the variances $\boldsymbol{\lambda}$.

6. Results

We fix the patch size at 8×8. The search region is of size 21×21 for the spatial denoising and 11×11 for the temporal filtering and smoothing. The remaining parameters are twelve. For iteration i of the spatial denoising: the number of patches $n_x^{(i)}$ and the noise multiplier $\gamma_x^{(i)}$; for iteration i of the temporal filtering, the numbers of patches $n_t^{(i)}$ and $m_t^{(i)}$ and the noise multiplier $\gamma_t^{(i)}$; and for the smoother the number of patches n_t^S and noise multiplier γ_t^S. These parameters were tuned using a training set of consisting of 400×400 crops extracted from 14 grayscale sequences having 20 frames each. We considered noises 10, 20 and 40 and computed the parameters as linear functions of σ.

To differentiate the proposed non-local Kalman (NLK) filter from the one in [18] we call ours backward NLK (BNLK) and one in [18] forward NLK (FNLK), the reason being that during filtering, ours uses the backwards optical flow to locate the previous patches in frame $t-1$, whereas [18] uses the forward flow to push the patches from $t-1$ to t. The proposed BNLK applies two filtering iterations. The result obtained with a single iteration is labeled BNLK1. The results obtained with the smoother are denoted as BNLK+S.

We evaluated the performance of the algorithm by computing the average PSNR and SSIM over the test set used in [15]. It consists of seven grayscale sequences with 100 960×540 frames from [39] used in [2]. Table 3 shows the quantitative results together with six state-of-the-art methods: VNLB [3], VNLnet [15], SPTWO [9], V-BM3D [13], V-BM4D [29] and FNLK [18]. For reasons of space we did not include other frame recursive methods in our comparison. An indirect comparison can be made through V-BM3D: for example the results reported in [1, 16] are on average around 2dB below V-BM3D and in [23] 1.6dB.

The results after the single filtering iteration are rather poor (more than 2dB below FNLK). This is in part because the parameters are set to optimize the output only after the second iteration. But it is also because of the approximations introduced with respect to FNLK. The results are greatly improved by the second iteration, up to 3dB for $\sigma = 40$. The reason for this might be that the transition variances are better estimated in the second iteration using the first iterate as guide. Our results after the two filtering iterations (BNLK) are on par with FNLK and V-BM3D. For $\sigma = 40$ we obtain a better performance than FNLK. In terms of SSIM, our results are slightly better than those of V-BM4D as well. More complex methods such as VNLB, SPTWO and VNLnet dominate in quantitative terms. This comes at the expense of a much higher complexity: they use a dozen of frames to estimate each output frame and take two to three minutes per-frame. Our non-optimized C implementation takes $16s$ but is amenable to parallelization in GPU, where it should reach real-time performances, although we have not tested this yet. A visual comparison shows that the recursive methods are able to recover many more details than the non-recursive ones (particularly for high levels of noise). However, the proposed BNLK method produces a sharper result than FNLK and is able to recover even more details. The smoother boosts the denoising performance and temporal consistency (see the supplementary material) and is a good option for off-line denoising.

7. Conclusions and perspectives

Recursive methods, and in particular frame recursive ones were until recently relegated from the state of the art in video denoising. This work demonstrates that good results can be achieved with fast and simple methods with low memory requirements, making them ideal for real-time processing. We are currently working on a GPU implementation of the proposed method.

The proposed method is based on a recursive Bayesian estimation framework. In addition to the filter and smoother presented in this work, other options are possible, such as a fixed lag smoother that estimates image t having observed a few future frames. We plan to explore such options.

Acknowledgments. Work partly financed by IDEX Paris-Saclay IDI 2016, ANR-11-IDEX-0003-02, Office of Naval research grant N00014-17-1-2552, DGA Astrid project «filmer la Terre» n^o ANR-17-ASTR-0013-01, MENRT.

References

[1] Redha A. Ali and Russell C. Hardie. Recursive non-local means filter for video denoising. *EURASIP Journal on Image and Video Processing*, 2017.

[2] Pablo Arias, Gabriele Facciolo, and Jean-Michel 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.

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

[4] Coloma Ballester, Lluis Garrido, Vanel Lazcano, and Vicent Caselles. A TV-L1 optical flow method with occlusion detection. In Axel Pinz, Thomas Pock, Horst Bischof, and Franz Leberl, editors, *Pattern Recognition*, pages 31–40, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[6] M. J. Black. *Robust Incremental Optical Flow*. PhD thesis, Yale University, Department of Computer Science, New Haven, CT, 1992. Research Report YALEU-DCS-RR-923.

[7] James C. Brailean, Richard P. Kleihorst, Serafim Efstratiadis, Aggelos K. Katsaggelos, and R.L. Lagendijk. Noise Reduction Filters for Dynamic Image Sequences: A Review. *Proceedings of the IEEE*, 83(9):1272–1292, 1995.

[8] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms , with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, 2006.

[9] Antoni Buades, Jose-Luis Lisani, and Marko Miladinović. Patch-based video denoising with optical flow estimation. *IEEE Transactions on Image Processing*, 25(6):2573–2586, June 2016.

[10] A. Bugeau, P. Gargallo, O. D'Hondt, A Hervieu, N. Papadakis, and V. Caselles. Coherent Background Video Inpainting through Kalman Smoothing along Trajectories. In *Modeling, and Visualization Workshop*, page 8, 2010.

[11] Priyam Chatterjee and Peyman Milanfar. Patch-based near-optimal image denoising. *IEEE Transactions on Image Processing*, 21(4):1635–1649, 2012.

[12] Xinyuan Chen, Li Song, and Xiaokang Yang. Deep RNNs for video denoising. In *Applications of Digital Image Processing*, 2016.

[13] Kostadin Dabov, Alessandro Foi, and Karen Egiazarian. Video denoising by sparse 3D transform-domain collaborative filtering. In *EUSIPCO*, 2007.

[14] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space. 2007.

[15] Axel Davy, Thibaud Ehret, Gabriele Facciolo, Jean-Michel Morel, and Pablo Arias. Non-local video denoising by CNN. *CoRR*, abs/1811.12758, 2018.

[16] J. Ehmann, L. Chu, S. Tsai, and C. Liang. Real-time video denoising on mobile phones. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 505–509, Oct 2018.

[17] Thibaud Ehret, Pablo Arias, and Jean-Michel Morel. Global patch search boosts video denoising. In *International Conference on Computer Vision Theory and Applications*, 2017.

[18] 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.

[19] M. Elad and A. Feuer. Super-resolution reconstruction of image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):817–834, Sep. 1999.

[20] Sina Farsiu, Michael Elad, and Peyman Milanfar. Video-to-video dynamic super-resolution for grayscale and color sequences. *EURASIP Journal on Advances in Signal Processing*, 2006(1):061859, Dec 2006.

[21] Clement Godard, Kevin Matzen, and Matt Uyttendaele. Deep burst denoising. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[22] O. G. Guleryuz. Weighted overcomplete denoising. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1992–1996 Vol.2, Nov 2003.

[23] Yubing Han and Rushan Chen. Efficient video denoising based on dynamic nonlocal means. *Image and Vision Computing*, 30(2):78 – 85, 2012.

[24] Antoine Houdard, Charles Bouveyron, and Julie Delon. High-Dimensional Mixture Models For Unsupervised Image Denoising (HDMI). Preprint, June 2017.

[25] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[26] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. A Nonlocal Bayesian Image Denoising Algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013.

[27] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm. *Image Processing On Line*, 3:1–42, 2013.

[28] Ce Liu and William T. Freeman. A high-quality video denoising algorithm based on reliable motion estimation. In *ECCV*, pages 706–719, 2010.

[29] Matteo Maggioni, Giacomo Boracchi, Alessandro Foi, and Karen Egiazarian. Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms. *IEEE Transactions on Image Processing*, 2012.

[30] Ben Mildenhall, Jonathan T. Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Rob Carroll. Burst denoising with kernel prediction networks. In *CVPR*, 2018.

[31] Sergio G. Pfleger, Patricia D. M. Plentz, Rodrigo C. O. Rocha, Alyson D. Pereira, and Márcio Castro. Real-time video denoising on multicores and gpus with kalman-based and bilateral filters fusion. *Journal of Real-Time Image Processing*, Feb 2017.

[32] H. E. Rauch, C. T. Striebel, and F. Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

[33] Alexandre Saint-Dizier, Julie Delon, and Charles Bouveyron. A unified view on patch aggregation. Preprint, Aug. 2018.

[34] Javier Sánchez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, 3:137–150, 2013.

[35] Venkataraman Santhanam, Vlad I. Morariu, and Larry S. Davis. Generalized deep image to image regression. *CoRR*, abs/1612.03268, 2016.

[36] Kalpana Seshadrinathan and Alan Conrad Bovik. Motion tuned spatio-temporal quality assessment of natural videos. *Trans. Img. Proc.*, 19(2):335–350, Feb. 2010.

[37] A. Singh. Incremental estimation of image-flow using a kalman filter. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 36–43, Oct 1991.

[38] Bihan Wen, Yanjun Li, Luke Pfister, and Yoram Bresler. Joint adaptive sparsity and low-rankness on the fly: an online tensor reconstruction scheme for video denoising. In *IEEE ICCV*, 2017.

[39] Xiph.org. Xiph.org Video Test Media (Derf's collection). `https://media.xiph.org/video/derf`. Accessed: 2019/04/14.

[40] Guoshen Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. *Image Processing, IEEE Transactions on*, 21(5):2481–2499, May 2012.

[41] C. Zach, T. Pock, and H. Bischof. *A Duality Based Approach for Realtime TV-L1 Optical Flow*, pages 214–223. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[42] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 7 2017.

[43] Kai Zhang, Wangmeng Zuo, and Lei Zhang. FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising. *CoRR*, abs/1710.0, 2017.

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

[45] Chenglin Zuo, Yu Liu, Xin Tan, Wei Wang, and Maojun Zhang. Video denoising based on a spatiotemporal kalman-bilateral mixture model. *The Scientific World Journal*, 2013, 2013.