Published in Image Processing On Line on 2026–06–00.
Submitted on 2024–03–12, accepted on 2026–05–28.
ISSN 2105–1232 © 2026 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.2026.533

Fixed Pattern Noise Reduction: Optimization-Based and Temporal High Pass Filter Methods

Hortensia Barral[1], Pablo Arias[2], Axel Davy[1]

[1]Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, France
[1]Universitat Pompeu Fabra, Dept. of Information and Communication Technologies, Spain
{hortensia.barral,axel.davy}@ens-paris-saclay.fr pablo.arias@upf.edu

Communicated by Marina Gardella *Demo edited by* Hortensia Barral

Abstract

Fixed pattern noise (FPN) is a temporally coherent noise present on video due to the non-uniform response of the sensors. It is a common problem for infrared videos and can degrade the quality of the observation. In this work we study and compare two types of FPN removal methods: temporal high pass filter and optimization-based methods. Both are recursive real-time methods that perform very few operations per pixel. We show that the temporal high pass filter can be obtained as a particular case of optimization-based methods. We implement and compare several variants that have been proposed in the literature.

Source Code

The reviewed source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README.md` file of the archive.

Keywords: video denoising; fixed pattern noise; non-uniformity correction; optimization-based methods

1 Introduction

The goal of video restoration is to recover the original video from a degraded observation. There are different types of degradation depending on the acquisition system, but in almost all realistic cases, noise is part of the degradation. Noise can be caused by several reasons. Fixed pattern noise (FPN) is a very specific kind of noise that is more prominent in infrared videos.

This type of noise is called *fixed pattern* noise because the noise is theoretically the same for every frame in the video, i.e. it is constant in time. In practice, the FPN might change slowly over time

[1]https://doi.org/10.5201/ipol.2026.533

but can be considered constant over a short period of time. The following linear model is widely used [12, 22, 8, 15, 19, 16, 6, 7, 18, 4, 10, 5] to describe noisy images with FPN

$$y(n) = a \otimes x(n) + b, \tag{1}$$

where $\otimes$ is the element-wise product, $x(n)$ is the clean frame at time n, $y(n)$ is the observed frame at time n, a and b are respectively the FPN gain and offset coefficients. For a pixel (i, j) we thus have that

$$y(n)_{i,j} = a_{i,j} x(n)_{i,j} + b_{i,j}. \tag{2}$$

In general, most FPN removal methods try to estimate correction coefficients in a recursive fashion [12, 22, 8, 15, 19, 16, 18, 5]. Let $G(n)$ and $O(n)$ be the gain and offset correction coefficients at time n, respectively. Then, the denoised image $\hat{x}(n)$ at time n is

$$\hat{x}(n) = G(n) \otimes y(n) + O(n). \tag{3}$$

The optimal correction coefficients are given by

$$G^*(n)_{i,j} = \frac{1}{a_{i,j}}, \tag{4}$$

$$O^*(n)_{i,j} = -b_{i,j} G^*(n)_{i,j} = -\frac{b_{i,j}}{a_{i,j}}. \tag{5}$$

The problem of removing FPN is called FPN removal (FPNR) or non-uniformity correction (NUC). FPNR methods can be divided into two main families: reference-based and scene-based. The reference-based methods [3] remove noise according to fixed correction coefficients that are calibrated offline for example by using a black body. However, the FPN changes slightly over time, which requires an update of the parameters. Because of this, most of the research is focused on scene-based methods, which try to estimate the FPN from the videos acquired during the normal camera operation, without requiring special calibration procedures.

We can distinguish several types of scene-based methods: those that work from image statistics [5], temporal high pass filters [13, 9, 2, 22, 21], registration methods [4, 10] and optimization-based methods [14, 12, 15, 8, 18, 16]. Recent works [7, 20, 6] are learning methods and mainly use convolutional neural networks (CNNs) that take as input a single noisy image.

In this work, we study several classical optimization-based methods and temporal high pass filter methods. We show that temporal high pass filter methods can be obtained as particular cases of optimization-based methods. We provide our own implementation for these methods and an online demo where they can be tested. Our code and the accompanying demo allow users to reproduce several variants of optimization-based methods [14, 19, 12, 8, 15] and temporal high pass filter [13, 9].

Section 2 reviews the state of the art and shows the link between the temporal high pass filter and optimization-based methods. In Section 3 we present the implementation and the pseudocodes. The influence of the parameters and both quantitative and qualitative results are presented in Section 4.

2 FPNR Methods

2.1 Temporal High Pass Filter Methods

FPN removal methods based on temporal high pass filter (THPF) work directly with the images and try to estimate correction coefficients in a recursive framework. The main assumption is that

temporal high frequency information belongs to the scene, while temporal low frequency information belongs to fixed pattern noise [13]. Therefore, by performing temporal high pass filtering, the noise can be removed. The second main assumption, introduced in [9], is that the content of the image is of low (spatial) frequency, thus the spatial high frequencies contain mainly noise (possibly both FPN and temporally varying noise). The second assumption was not part of the initial formulation of this family of methods [13], but was adopted as soon as it was introduced in [9]. From these two assumptions it follows that in a spatio-temporal frequency domain, the energy of the clean scene is concentrated in regions with either low spatial frequencies or high temporal frequencies. THPF methods work by letting these frequencies pass, and filtering out high spatial and low temporal frequencies. The high spatial frequencies can be estimated using linear filters [9] or non-linear filters such as a guided filter [2].

THPF methods consider only additive FPN b, i.e. for the multiplicative component we have $a = 1$ in Equation (1). They estimate b recursively as follows

$$\hat{b}(n) = \left(1 - \frac{1}{M}\right) \hat{b}(n-1) + \frac{1}{M} F_{HS}(n), \tag{6}$$

where M is a parameter, $\hat{b}(n)$ the estimated FPN at time n, and $F_{HS}(n)$ is the result of a spatial high pass filter applied to the noisy image $y(n)$. The value $\hat{b}(0)$ is initialized to 0.

The denoised image $\hat{x}(n)$ at time n is then

$$\hat{x}(n) = y(n) - \hat{b}(n). \tag{7}$$

Note that $\hat{b}$ is the result of a recursive linear filter. We can also write it as a convolution with a kernel with infinite support h_{LT}

$$\hat{b}(0) = 0, \tag{8}$$

$$\hat{b}(n) = \sum_{k=0}^{n} F_{HS}(k) h_{LT}(n-k), \quad n \geq 1, \tag{9}$$

with

$$\forall\ 0 \leq n,\ \ h_{LT}(n) = \frac{1}{M} \times \left(1 - \frac{1}{M}\right)^n, \tag{10}$$

where k is the number of considered frames and n is the total number of frames.

Figure 1 shows the value of the kernel $h_{LT}(n-k)$ with respect to the values of k varying between 0 and n, for different values of M used in the experiments below. The value of *sum* given in the legend of the figure, corresponds to the sum of the coefficients $\sum_{k=0}^{n} h_{LT}(n-k)$. Note that setting the total number of images as the value for M, which is the default value, leads to a sum of coefficient values of less than 1.

2.2 Optimization-based Methods

Optimization-based FPN removal methods estimate the correction coefficients via the minimization of an energy that is linked to the quality of the corrected frames. The energy depends on the current noisy frame, and thus it varies with time.

Correction coefficients are first initialized and then updated to minimize the energy. The minimization is performed by a single gradient descent step at each frame.

In [14, 19] the energy at time n is defined as the squared L_2 distance between a corrected image $\hat{x}(n)$ as defined in Equation (3) and an image $F_{LS}(n)$ obtained by applying a low pass spatial filter to $y(n)$

$$\mathcal{E}_n(G, O) = \|\hat{x}(n) - F_{LS}(n))\|_2^2 = \|G \otimes y(n) + O - F_{LS}(n)\|_2^2. \tag{11}$$

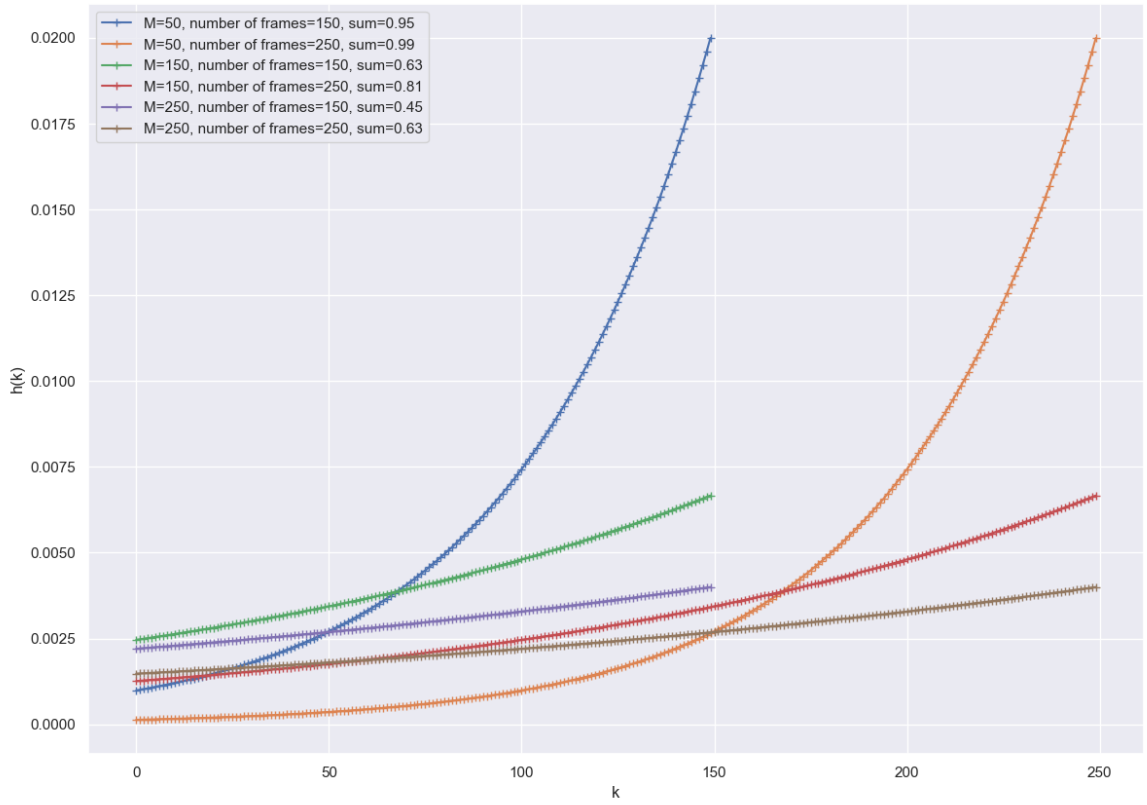


Figure 1: Weights of the temporal high pass filter as presented in Equation (10).

The spatial low pass filter can be an average filter [14, 19] or it can be a non-linear filter, such as a bilateral filter [12], for example. We denote the filtered image F_{LS} as the reference image. A gradient descent step then results in the following update

$$\begin{aligned} G(n+1)_{i,j} &= G(n)_{i,j} - 2\alpha(\hat{x}(n)_{i,j} - F_{LS}(n)_{i,j})y(n)_{i,j}, \\ O(n+1)_{i,j} &= O(n)_{i,j} - 2\alpha(\hat{x}(n)_{i,j} - F_{LS}(n)_{i,j}), \end{aligned} \tag{12}$$

where $\alpha > 0$ is the step in the gradient descent.

Thus, optimization-based FPNR methods define a temporal sequence of energies $\mathcal{E}_n$ as an attachment to the low pass filtered version of the current noisy frame. At time n, a new image $y(n)$ is acquired, and the method runs a single iteration of the gradient descent update from the energy defined at time n. This process results in a temporal smoothing of the correction coefficients $G(n)$ and $O(n)$. Figure 2 illustrates the iterative scheme for optimization-based methods.

2.3 Link Between these Two Families of Methods

THPF methods can actually be described as a subset of optimization-based methods. In order to show that, let us consider an optimization-based setup with additive FPN and reference images $(F_{LS}(n))_n$ obtained by filtering the images $(y(n))_n$ with a spatial low pass filter. In that case the

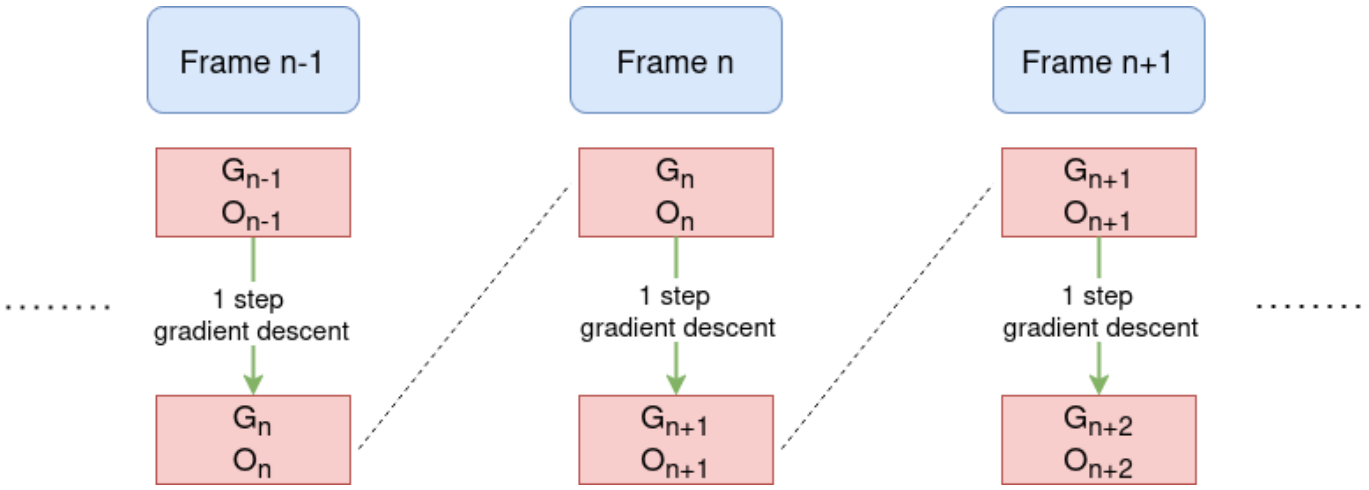


Figure 2: Iterative scheme for optimization-based methods.

energy in Equation (11) is simplified as follows

$$\mathcal{E}_n(O) = \|\hat{x}(n) - F_{LS}(n))\|_2^2 = \|y(n) + O - F_{LS}(n)\|_2^2 = \|O + \underbrace{y(n) - F_{LS}(n)}_{F_{HS}(n)}\|_2^2, \tag{13}$$

which shows that $-O$ should minimize the distances to the spatial denoising residuals $F_{HS}(n) = y(n) - F_{LS}(n)$. The update of O is given by Equation (12)

$$O(n+1) = O(n) - 2\alpha(\hat{x}(n) - F_{LS}(n)). \tag{14}$$

We can rewrite it as

$$O(n+1) = O(n) - 2\alpha(y(n) + O(n) - F_{LS}(n)) = (1-2\alpha)O(n) - 2\alpha(\underbrace{y(n) - F_{LS}(n)}_{F_{HS}(n)}) \tag{15}$$

$$\implies O(n+1) = (1-2\alpha)O(n) - 2\alpha F_{HS}(n), \tag{16}$$

where $F_{HS}(n)$ is the residual of $F_{LS}(n)$, which amounts to applying a spatial high pass filter to the images.

If we note $\frac{1}{M} = 2\alpha$, we have

$$O(n+1) = \left(1 - \frac{1}{M}\right) O(n) - \frac{1}{M} F_{HS}(n). \tag{17}$$

Here O is the correction coefficient for additive noise, which implies that the estimated additive FPN is $\hat{b}(n) = -O(n)$, thus

$$\hat{b}(n+1) = \left(1 - \frac{1}{M}\right) \hat{b}(n) + \frac{1}{M} F_{HS}(n), \tag{18}$$

which is almost Equation (6). Almost, because in Equation (6), the index of $\hat{b}$ is $(n-1)$ and not n. Thus, THPF methods are mathematically equivalent to an optimization-based method if the gain component of the FPN is not considered.

The reason for this equivalence is that optimization-based methods do not really optimize a (static) energy. They take steps towards the reference image, which changes each frame. This has a temporal smoothing effect on the correction coefficients O and G.

2.4 Discussion on the Design Choices

In the accompanying demo and in this article, we present and test optimization-based FPNR methods that use a filter [14, 19, 15, 12]. Several filters and step sizes have been proposed in the literature [14, 19, 15]. In this work, we implement and compare some of these choices.

2.4.1 Filter

As explained in Section 2.2, most optimization-based and THPF FPNR methods use a spatial filter [14, 19, 12, 15, 8, 13, 9] to compute the reference image in the optimization methods or the high pass spatial filter in the THPF methods.

No filter [13]. The first paper introducing the THPF method did not use any spatial filter [13]. The estimated additive noise was just a recursive average of the frames.

Average filter [14]. The first paper that proposed to use an optimization-based method for FPNR used an average filter [14]. The reference image F_{LS} in Equation (11) is usually defined as follows

$$F_{LS}(n) = y(n) * h_{LS},$$

where $*$ is the convolution operator and h_{LS} is the averaging filter. The averaging filter can be defined either by a $d \times d$ box filter or by the average of the four neighboring pixels of the pixel under consideration

$$h_{LS} = 1/4 \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \text{or} \quad h_{LS} = \frac{1}{d^2} \begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{pmatrix}. \tag{19}$$

This filter was introduced to reduce ghosting artifacts and increase convergence speed [9].

Threshold average filter [9]. The spatial low temporally high (SLTH) THPF method introduced the use of spatial filtering for THPF methods. The authors used a spatial low-pass filter [9], and since then, the use of such a filter has become standard in this family of methods [2, 22]. As explained in Section 2.1, the assumption is that the low spatial frequencies (LSFs) of the frames contain the clean signal, whereas the high spatial frequencies (HSFs) mainly contain noise. The authors of the SLTH THPF method used a thresholded average filter. They first applied an average filter, computed the residual, and then applied a threshold

$$y^{HSF}(n) = y(n) - y^{LSF}(n) = y(n) - y(n) * h_{LS}, \tag{20}$$

$$F_{HS}(n)_{i,j} = \begin{cases} y^{HSF}(n)_{i,j} & \text{if } |y^{HSF}(n)_{i,j}| < \tau, \\ 0 & \text{otherwise,} \end{cases} \tag{21}$$

where h_{LS} is a spatial low pass filter (in the original paper they used a 10×10 average filter) as in Equation (19) with $d = 10$ and τ is the threshold parameter.

The intuition behind this is that high frequencies with high values are mainly caused by the signal (and therefore should be discarded) while those high frequencies with low values are mainly due to noise (and those are the ones to keep).

Others. More recent approaches propose to replace the above filters by a bilateral filter [12, 8, 22] or a guided filter [15, 2]. The bilateral filter was introduced as a way of increasing convergence speed and reducing ghosting artifacts [22, 12], and is defined as

$$F(n)_{i,j} = \sum_{k,l} w(i,j,k,l) x(n)_{k,l}, \tag{22}$$

with

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_s^2} - \frac{\|x(n)_{i,j} - x(n)_{k,l}\|^2}{2\sigma_c^2}\right), \tag{23}$$

where σ_s and σ_c are smoothing parameters, and (i,j) and (k,l) are pixels.

2.4.2 Adaptive Step Size

Some optimization-based methods proposed to adapt the step size according to the images. The first paper [14] that introduced an optimization-based FPNR method used a constant step size $\alpha > 0$. Subsequent works have tried to improve the step size by taking into account the temporal and spatial correlation of the images.

Space dependent step size [19]. The authors of [19] proposed to use a step size that is dependent on the local spatial variance of the image

$$\alpha(n)_{i,j} = \frac{\alpha_0}{1 + \sigma^S(n)_{i,j}}, \tag{24}$$

where α_0 is the nominal step size value, a scale parameter of the form $\alpha_0 = 10^{-\beta}$ with $\beta \in \mathbb{N}$, $\sigma^S(n)_{i,j}$ is the spatial standard deviation of the pixels centered around pixel (i,j) in the image n

$$\sigma^S(n) = \sqrt{[y(n) \otimes y(n)] * h'_{LS} - (y(n) * h'_{LS}) \otimes (y(n) * h'_{LS})}, \tag{25}$$

where h'_{LS} is a low pass spatial filter and the square root is applied element-wise. Note that h'_{LS} can differ from the spatial filter h_{LS} used to compute the reference images.

The intuition behind this is that for a given pixel (i,j), if $\sigma^S(n)_{i,j}$ is small, it means that this part of the image is smooth and has no edges. In this case, the value of the spatially filtered reference image F_{LS} is a good approximation of the clean image, and a larger gradient descent step can be performed. On the contrary, a high spatial variance suggests edges or high frequency details that might not be removed from F_{LS}. Using a lower step size in those regions reduces image blur around the edges and in high frequency details.

Space and time dependent step size [15]. The authors of [15] added a temporal component to the space-dependent step size from [19]. In this case, the step size becomes

$$\alpha(n)_{i,j} = \frac{\alpha_0 \times \sigma^T(n)_{i,j}}{1 + \sigma^S(n)_{i,j}}, \tag{26}$$

where $\sigma^T(n)_{i,j}$ is the temporal standard deviation of the pixel (i,j) for frames n, $n-1$, …, $n-s_t+1$

$$\sigma^T(n)^2_{i,j} = \frac{1}{s_t - 1} \sum_{m=n-s_t+1}^{n} y(m)^2_{i,j} - \left(\frac{1}{s_t - 1} \sum_{m=n-s_t+1}^{n} y(m)_{i,j}\right)^2, \tag{27}$$

where s_t is the number of images used to compute the temporal standard deviation.

The idea behind this is that if the temporal standard deviation is large, it means there is a lot of motion between frames. This is beneficial for FPN estimation, as the clean signal component is of high temporal frequency and can be well separated from the FPN. In that case, the step size can be increased since the actual content of the image will not be interpreted as noise.

Note that this heuristic modification to the step size, adding a temporal component, can lead to divergence of the iterative scheme. To the best of our knowledge there is no theoretical convergence result for optimization-based methods. Results for convergence of optimization-based methods are usually empirical [12, 2, 8, 19]. For the particular case of the average filter and a constant step size, the authors of [14] claim that convergence can be theoretically proved using Lyapunov functions.

3 Implementation

3.1 Pseudocode

The pseudocodes for optimization-based and THPF methods are respectively given in Algorithm 1 and Algorithm 2. Algorithm 2 is written following [22]. There is a small difference in the ordering of the operations within the loop in the algorithms. The update of the parameters at time n is performed using the frame at time $n-1$ for the optimization-based methods and the frame at time n for the THPF methods. If this is ignored, Algorithm 2 is a particular case of Algorithm 1 with a specific choice of parameters, as shown in Section 2.3. Note that in the demo, methods have been implemented according to Algorithm 1 rather than Algorithm 2.

Algorithm 1: Optimization-based method

function *Optimization_FPNR*
- **Input** y**:** A list of N noisy images of size $H \times W$ with C channels
- **Output** $\hat{x}$**:** A list of N denoised images of size $H \times W$ with C channels
- **Param** α_0**:** nominal step size value

$O[1] \coloneqq \texttt{zeros}(H, W, C)$ `# initialize O with a matrix of zeros`
$G[1] \coloneqq \texttt{ones}(H, W, C)$ `# initialize G with a matrix of ones`
for n **from** *1* **to** $N-1$ **do**
- $\hat{x}[n] = G[n] \otimes y[n] + O[n]$
- $F_{LS}[n] \coloneqq \texttt{spatial-filter}(y[n])$
- $\alpha[n] = \texttt{return\_step\_size}(y, \alpha_0)$
- $G[n+1] = G[n] - 2\alpha[n](\hat{x}[n] - F_{LS}[n]) \otimes y[n]$
- $O[n+1] = O[n] - 2\alpha[n](\hat{x}[n] - F_{LS}[n])$

$\hat{x}[N] = G[N] \otimes y[N] + O[N]$
return $\hat{x}$

We implemented the average filter and, for the bilateral filter, we used the implementation from scikit-image [17].

3.2 Noise Modelling

We used the noise model from Equation (1) where a and b are the multiplicative and additive components of the FPN.

Our noise model considers both spatially independent FPN and spatially correlated FPN. Indeed, FPN can have both a structured component and an unstructured component, and it is important to

Algorithm 2: THPF

function *THPF_FPNR*
  Input y**:** A list of N noisy images of size $H \times W$ with C channels
  Output $\hat{x}$**:** A list of N denoised images of size $H \times W$ with C channels
  $\hat{b}[1] \coloneqq \texttt{zeros}(H, W)$
  `# initialize` $\hat{b}$ `with a matrix of zeros`
  $\hat{x}[1] \coloneqq y[1]$
  for n **from** *1* **to** $N-1$ **do**
    $F_{HS}[n+1] \coloneqq \texttt{spatial-high-pass-filter}(y[n+1])$
    $\hat{b}[n+1] = \left(1 - \frac{1}{M}\right)\hat{b}[n] + \frac{1}{M}F_{HS}[n+1]$
    $\hat{x}[n+1] = y[n+1] - \hat{b}[n+1]$
  return $\hat{x}$

consider both since the denoising algorithm can have different behaviors depending on the structure of the noise. For the structured component, we considered row and column noise. Specifically, we define the multiplicative and additive components as

$$a = a^w \otimes a^r \otimes a^c, \quad b = b^w + b^r + b^c, \tag{28}$$

with

$$\forall i, i', j, j' \;\; a^r_{i,j} = a^r_{i,j'}, \;\; a^c_{i,j} = a^c_{i',j}, \;\; b^r_{i,j} = b^r_{i,j'}, \;\; b^c_{i,j} = b^c_{i',j}, \tag{29}$$

where the superscripts w, r, and c denote the unstructured, row, and column components, respectively. Note that the row noise is constant across columns for a given row, and the column noise is constant across rows for a given column. The unstructured component, in contrast, is spatially independent. Finally, we assume that the three components are mutually independent.

These components can be modeled as Gaussian random variables. The unstructured terms are drawn independently for each pixel, with distributions $b^w \sim \mathcal{N}(0, \sigma_{b^w})$ and $a^w \sim \mathcal{N}(1, \sigma_{a^w})$. The row terms are constant along each row (independent across rows), with distributions $b^r \sim \mathcal{N}(0, \sigma_{b^r})$ and $a^r \sim \mathcal{N}(1, \sigma_{a^r})$. Similarly, the column terms are constant along each column (independent across columns), with distributions $b^c \sim \mathcal{N}(0, \sigma_{b^r})$ and $a^c \sim \mathcal{N}(1, \sigma_{a^r})$. Note that, for simplicity, we consider the same variance for the row and column noise components.

In the demo, the user can select the level of structured and unstructured fixed pattern noise, for both the additive and multiplicative components.

4 Influence of the Parameters on the Performance of the Algorithm

In this section we discuss the parameters of optimization-based methods. THPF methods are a particular case with a specific choice of parameters. This was proven in Section 2.3.

4.1 Datasets

To test the different approaches, we used infrared datasets [1, 11], and visible datasets from the derf's Test Media collection[2].

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

- The bus sequence from derf contains 150 visible images and has some motion.
- The flower sequence from derf contains 250 visible images and has little motion.
- The 8_selma sequence from ltir [1] contains 235 infrared images. The camera is fixed and films the scene with people walking.
- The 640_ataset [11] is a dataset of 1000 infrared images. This dataset does not contain videos, but we generate one by concatenating 150 images from the training dataset. Note that this "video" does not have any temporal consistency, the concatenated images have nothing in common. This situation is ideal for FPN removal. While temporal consistency of the signal is useful for removing non-fixed noise, it is harmful for removing FPN.

Since the method takes several frames to "warm-up", for the quantitative results, we report the PSNR of the last frame of each dataset and, for the visual results, the last frame of each dataset. Note also that some methods may require several hundred images for warm-up, or even more.

4.2 Step Size

The step size controls the speed at which the correction coefficients O and G are updated with each new frame. The step size can be constant, space dependent, or space and time dependent. The default option is set to constant.

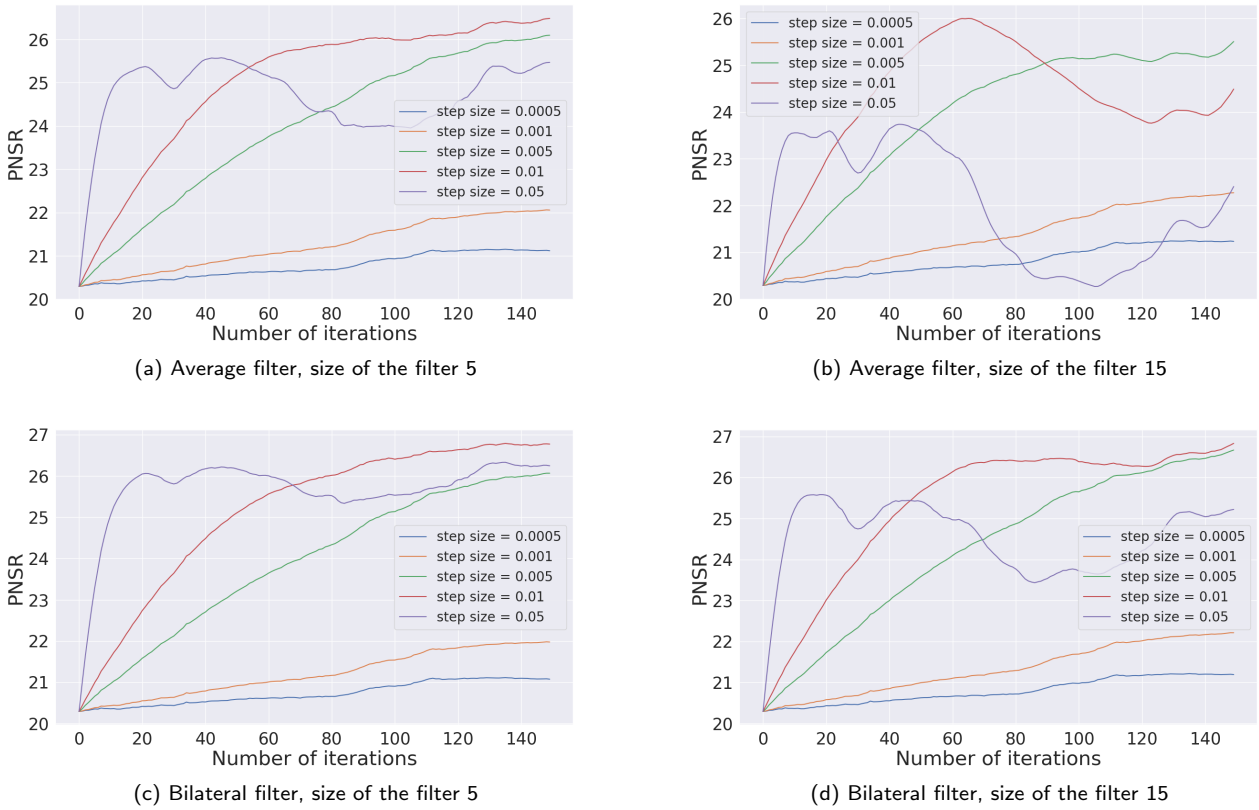
4.2.1 Constant Step Size

We start by discussing the effect of a constant step size α. In Figure 3 we observe the evolution of the PSNR obtained using Algorithm 1 with different step sizes, for denoising the sequence *bus* with additive FPN. In plots 3a and 3b we show results obtained with an average filter respectively with a filter size of 5 and 15, where we vary the step size. The same is done for the bilateral filter in plots 3c and 3d. In plots 4a and 4b we do the opposite: the step size is fixed and we compare the bilateral and average filter with different filter sizes.

The algorithm is initialized with $O = 0, G = 1$, meaning that initially no FPN is removed. Since one gradient descent step is performed per frame, several frames are required for the correction coefficients O, G to converge. We use "converge" in a loose sense, since the energy itself changes each frame. Due to this initial warm-up stage, the initial frames are severely under-denoised, resulting in a low PSNR. The convergence speed is controlled by the step size α. For large step sizes, the warm-up needs fewer iterations. On the contrary, if the step size is too small, the warm-up might need more iterations than the number of frames in the sequence. The plots in Figure 4 show that the warm-up is mostly determined by the step size, and not by the spatial filter.

For additive noise, based on the interpretation of optimization-based methods as temporal filters (see Section 2.3), we can also view the step size as controlling the cut-off frequency of a recursive temporal filter that estimates $O(n)$ from the temporal low frequencies of the sequence of spatial residuals $F_{HS}(n) = y(n) - F_{LS}(n)$. Smaller step sizes imply a lower cut-off frequency, i.e. $O(n)$ will be very smooth temporally, consisting of the low temporal frequencies of the spatial filtering residual F_{HS}. With larger step sizes, $O(n)$ will have higher temporal frequencies.

This explains why larger step sizes might lead to a worse steady-state PSNR, specially for extremely simple spatial filters: with larger step sizes, the estimated correction coefficients will be more impacted by signal components that are left in the spatial filtering residual $F_{HS}(n)$. The bilateral filter is more robust to this.



(a) Average filter, size of the filter 5

(b) Average filter, size of the filter 15

(c) Bilateral filter, size of the filter 5

(d) Bilateral filter, size of the filter 15

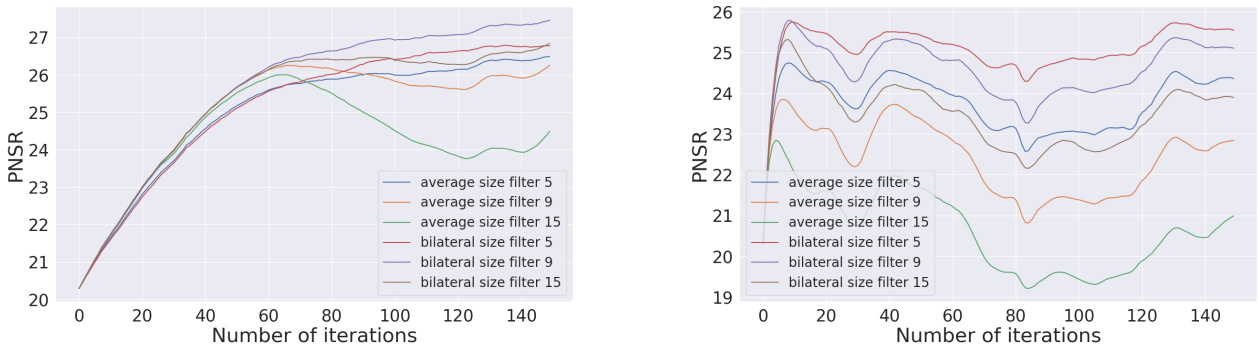
Figure 3: Plot of the evolution of the PSNR on the bus dataset with a constant step size. Additive FPN, $\sigma_{b^w} = \sigma_{b^r} = 15$, was added. (a) The size of the filter is fixed to 5 and several step size values are compared. Only the average filter is used. (b) Same as (a) with a filter size of 15. (c) The size of the filter is fixed to 5 and several step size values are compared. Only the bilateral filter is used. (d) Same as (c) with a filter size of 15.

4.2.2 Adaptive Step Sizes

Table 1 shows results for different types of step size: constant, space dependent and space and time dependent. Table 2 shows results for different adaptive step sizes and the median step size value for each sequence.

Space dependent step size. The space dependent step size requires setting one more parameter: the size of the filter used to compute the spatial standard deviation, which we denote by s_x. The default value is set to $s_x = 3$. Since the space dependent step size is always smaller than the nominal step size, it might be needed to increase the nominal step size. In practice, we have observed that the space dependent step size does not improve the results and may even deteriorate them (see Table 1). The value of s_x seems to have no influence on the performance. This is somehow implicit in the fact that the median step value is equal to the nominal step size regardless of the value of s_x (see Table 2). Even in Figure 5a, almost no difference can be seen between space dependent step size and the constant step size that has the same nominal value. We do not have a clear explanation for this. Our best guess is that since the images are relatively piece-wise constant, there is not that much variation, and so the spatial variance is close to 0 in most cases.

Space and time dependent step size. The space-and-time-dependent step size requires one additional parameter compared with the space-dependent step size: namely, the number of images, s_t, used to compute the temporal standard deviation. The default value is set to $s_t = 9$. The time dependent step size increases the value of the step size (see Table 2). In this case, the nominal step



(a) $\alpha = 0.01$ (b) $\alpha = 0.1$

Figure 4: Plot of the evolution of the PSNR on the bus dataset with a constant step size. Additive FPN, $\sigma_{b^w} = \sigma_{b^r} = 15$, was added. (a) The step size is fixed to $\alpha = 0.01$, and we compare average and bilateral filter with several filter sizes. (b) Same as (a) with a constant step size equal to $\alpha = 0.1$.

size value must be decreased, otherwise the algorithms may diverge (see Figure 5a). Based on our results, the space and time dependent step size does not improve the results in most cases. The time and space dependent step size can sometimes achieve better results than the constant one (see Table 1). However, such results can be obtained with a fine-tuned constant step size. Higher values for s_x, s_t tend to provide better results, but it is also more time-consuming. While using this adaptive step size does not seem to improve the final PSNR compared to a fine-tuned constant step size, it does provide a shorter warm-up time (see Figure 5b).

parameters			**PSNR**			
nominal step size (α_0)	s_x	s_t	**bus**	**flower**	**8_selma**	**640_ataset**
5×10^{-3}	3	-	25.97	24.92	27.51	27.20
equivalent	-	-	26.10	24.94	27.52	27.22
5×10^{-3}	15	-	25.97	24.92	27.51	27.20
equivalent	-	-	26.10	24.94	27.52	27.22
5×10^{-3}	3	9	22.84	21.14	24.44	27.95
equivalent	-	-	24.53	23.03	25.5	27.94
5×10^{-4}	3	9	26.36	24.86	20.63	28.18
equivalent	-	-	26.50	24.81	20.58	28.18
5×10^{-4}	3	25	26.39	24.76	20.99	28.17
equivalent	-	-	26.33	24.66	20.67	28.20

Table 1: Parameter comparison between adaptive step size and equivalent constant step size (median step size used) obtained with an average filter (for both the reference image and the spatial variance) and a simulated additive FPN, spatially structured and spatially independent with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0$. The filter kernel size is set to 5 for all experiments. The PSNR of the last frame of the sequence is shown.

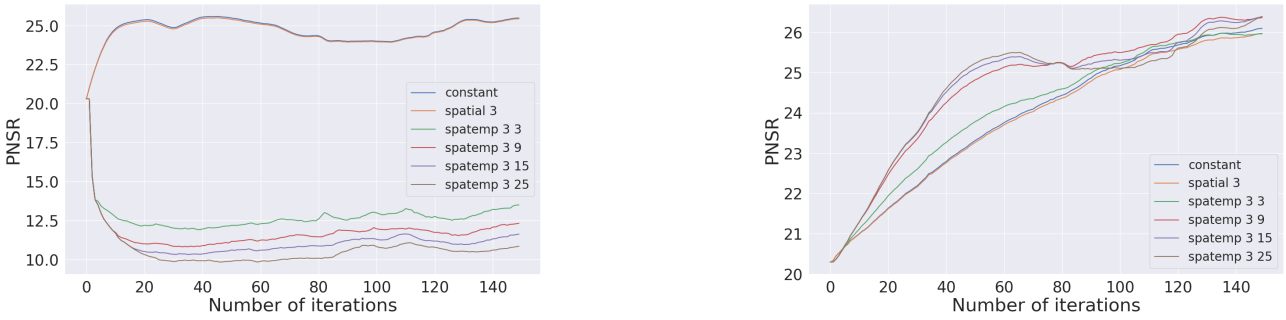
4.3 Filter

Both optimization-based and THPF methods use spatial low pass filters.

Filter size. This parameter is common for both types of filters present in the demo: the bilateral filter and the average filter. The default value is set to 5. Too high values tend to over-denoise and so over-smooth the video, and can also be much more time-consuming and resource intensive. The size of the filter also has an impact on the stationary behavior (see Figure 3a compared to 3b). A filter size of 15 provides better results than a filter size of 9 or 5 with the bilateral filter, whereas the

parameters			median step size value				type
nominal step size (α_0)	s_x	s_t	bus	flower	8_selma	640_ataset	
5×10^{-3}	3	-	5.00×10^{-3}	5.00×10^{-3}	5.00×10^{-3}	5.00×10^{-3}	spa
5×10^{-3}	15	-	5.00×10^{-3}	5.00×10^{-3}	5.00×10^{-3}	5.00×10^{-3}	spa
5×10^{-3}	3	9	9.15×10^{-2}	8.51×10^{-2}	2.08×10^{-3}	2.15×10^{-1}	spatemp
5×10^{-4}	3	9	9.15×10^{-3}	8.51×10^{-3}	2.08×10^{-4}	2.15×10^{-2}	spatemp
5×10^{-4}	3	25	1.54×10^{-2}	1.37×10^{-2}	2.32×10^{-4}	2.39×10^{-2}	spatemp

Table 2: Median value of the space, space and time dependent step size. s_x and s_t are respectively the size of the filter used to compute the spatial standard deviation and the number of images used to compute the temporal standard deviation. Type refers to the type of adaptive step size used, "spa" for space dependent and "spatemp" for space and time dependent step size.



(a) $\alpha = 0.05$ (b) Different

Figure 5: Plot of the evolution of the PSNR on the bus dataset with the average filter and a filter size of 5. Additive FPN, $\sigma_{b^w} = \sigma_{b^r} = 15$, was added. (a) Constant, space dependent and space and time dependent step sizes are compared with the same nominal step size value. (b) Constant, space dependent and space and time dependent step sizes are compared with different nominal step size values. For constant and space dependent step size $\alpha_0 = 0.05$, for the space and time dependent step size $\alpha_0 = 5 \times 10^{-4}$. Using a space dependent step size provides almost the same result as using a constant step size. Using a space and time dependent step size increases the value for the step size and so it is necessary to decrease its nominal value.

average filter achieves better results with a filter size of 5 or 9, except for the sequence where images have nothing in common (see Tables 3 and 4). Both the step size and the size of the filter have an impact on the stationary behavior.

parameters		PSNR			
size	step size	bus	flower	8_selma	640_ataset
5	5×10^{-3}	26.10	**24.94**	27.52	27.22
9	5×10^{-3}	**26.45**	24.62	**28.73**	28.85
15	5×10^{-3}	25.51	24.18	28.63	**29.96**
3	5×10^{-3}	24.85	24.34	25.63	25.43
5	5×10^{-2}	25.47	23.75	27.54	28.16
5	5×10^{-4}	21.12	21.88	21.54	21.18

Table 3: Comparison of the results obtained with different values of the filter size and the step size, obtained with an average filter and a simulated additive FPN , spatially structured and spatially independent with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0$. The PSNR of the last frame of the sequence is shown.

Filter type. In the demo, the user can choose between an average and a bilateral filter for the reference image. For the average filter, the average of the $d \times d$ pixels is used as defined in Equation (19). For the bilateral filter, σ_s is fixed to 25 and σ_c to the default value from the scikit implementation,

parameters		PSNR			
size	**step size**	**bus**	**flower**	**8_selma**	**640_ataset**
5	5×10^{-3}	26.07	25.54	25.75	25.84
9	5×10^{-3}	**26.84**	**25.80**	26.65	26.75
15	5×10^{-3}	26.67	25.36	**26.90**	27.19
3	5×10^{-3}	24.73	24.58	24.38	24.57
5	5×10^{-2}	26.25	24.80	26.33	**27.60**
5	5×10^{-4}	21.07	21.85	21.06	20.98

Table 4: Comparison of the results obtained with different values of the filter size and the step size, obtained with a bilateral filter and a simulated additive FPN, spatially structured and spatially independent with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0$. The PSNR of the last frame of the sequence is shown.

which is the standard deviation of the image. We compared the average and the bilateral filter. The bilateral filter achieved higher PSNR for the bus and flower datasets (see Table 6 for additive and multiplicative FPN, and Table 5 for additive FPN only).

	PSNR		
dataset	**Noisy**	**Average filter**	**Bilateral filter**
bus	20.10	26.45	**26.84**
flower	20.53	24.94	**25.80**
8_selma	19.87	**28.73**	26.90
640_ataset	20.11	**29.96**	27.60

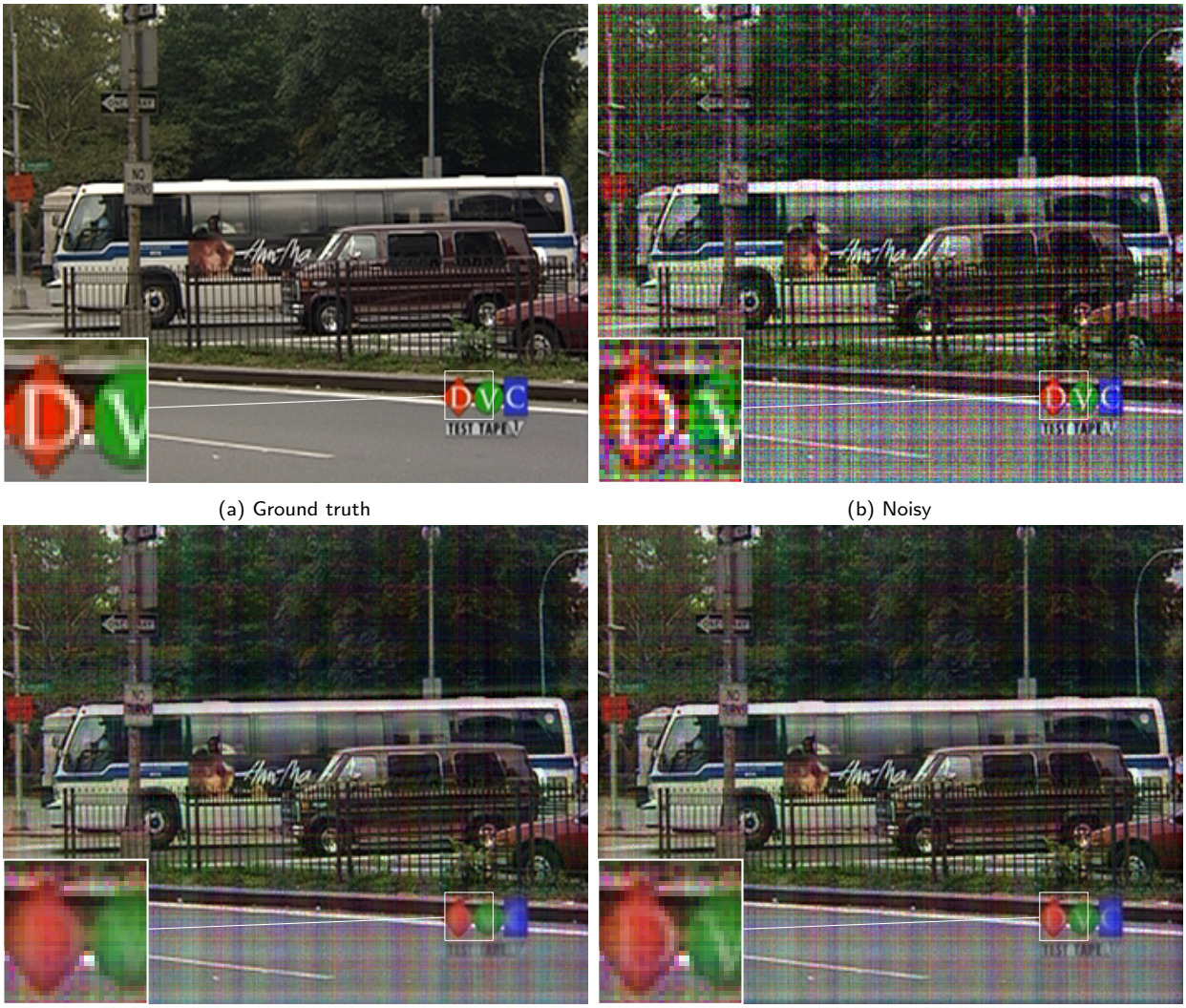
Table 5: Quantitative PSNR results obtained with simulated additive FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0$ and a fixed step size. The PSNR of the last frame is shown.

	PSNR		
dataset	**Noisy**	**Average filter**	**Bilateral filter**
bus	18.66	24.55	**24.71**
flower	17.18	21.57	**21.92**
8_selma	17.33	**27.68**	24.02
640_ataset	18.27	**26.59**	24.12

Table 6: Quantitative PSNR results obtained with simulated multiplicative and additive FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0.10$ and a fixed step size. The PSNR of the last frame is shown. The best results per dataset are highlighted. Note that the results are obtained in an oracular way: we chose the best parameters for each sequence as the ones that maximize the PSNR with respect to the ground truth.

The bilateral filter obtained better results overall, especially for the scene with more motion, for example in Figures 6 and 7. The visual results obtained with a bilateral filter are sharper than with an average filter, for example on the logo. The average filter provided better results for more "extreme" scenarios: a fixed scene and images having nothing in common. However when looking at the visual results, for more "extreme" scenarios the results obtained with the average filter have less structured noise than the results of the bilateral filter (see Figure 8 and 9).

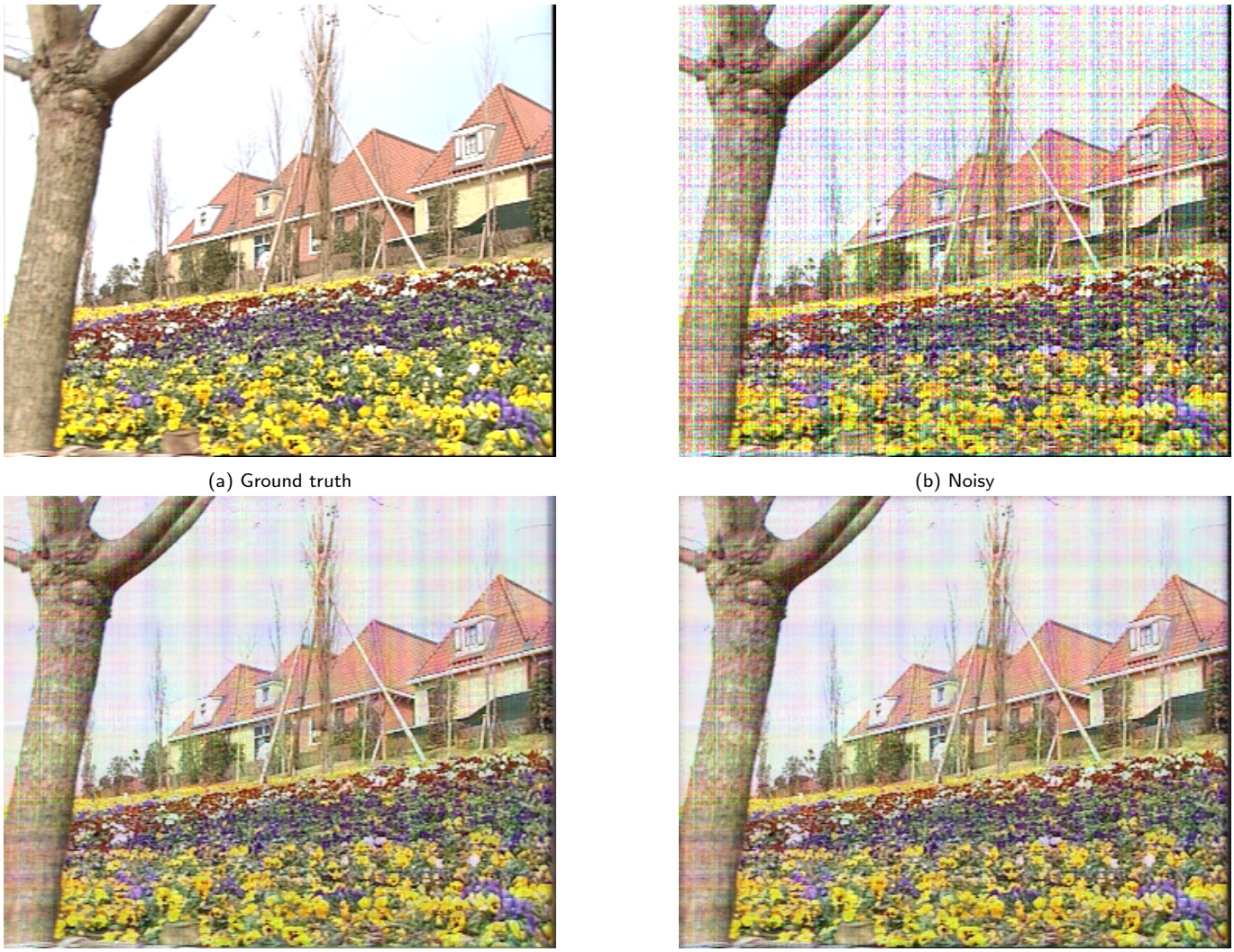
All methods still produce blurry results on the fixed scene (see Figure 8). In that case, it is very hard for the methods to distinguish between noise and actual content of the image, since both are fixed. The methods still achieve a reasonable PSNR by blurring the images. As the bilateral filter also takes pixel values into account, this method has more difficulty in removing spatially correlated noise compared to the average filter, as shown in the zoomed-in section of Figure 8.



(a) Ground truth (b) Noisy

(c) Average filter (d) Bilateral filter

Figure 6: Comparison of the different methods on the last image of the bus dataset that contains 150 frames and some motion. Simulated additive and multiplicative FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0.10$ was added to the frames. (a), (b), (c), (d) are respectively the ground truth clean image, the noisy image with simulated FPN, the image denoised by an average filter and the image denoised by a bilateral filter.



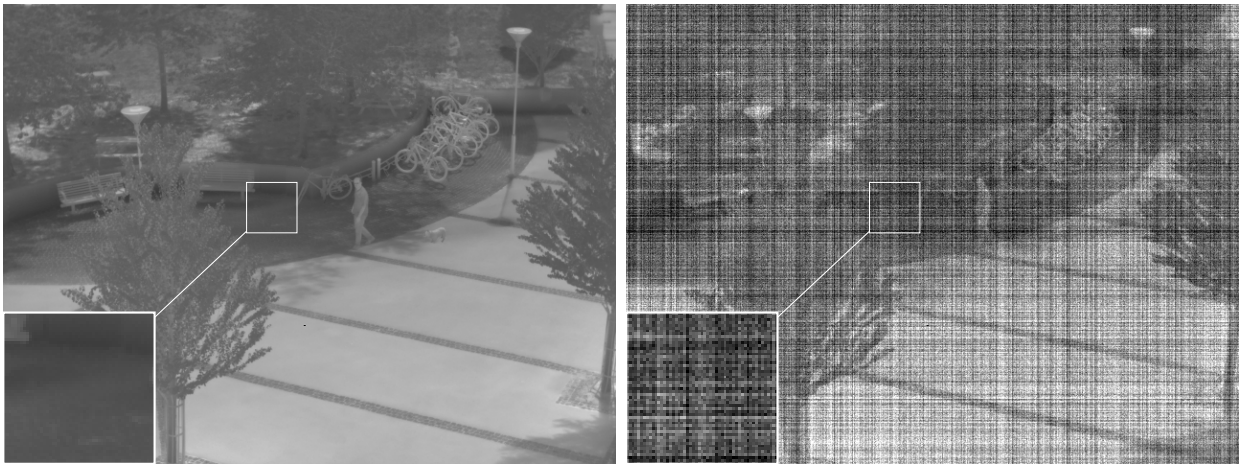
(a) Ground truth

(b) Noisy

(c) Average filter

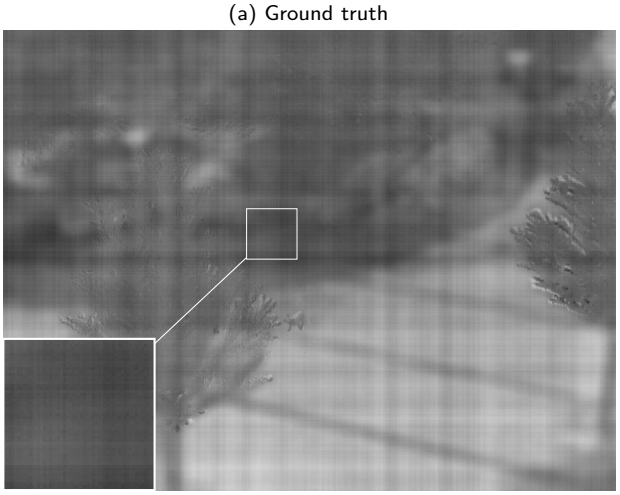
(d) Bilateral filter

Figure 7: Comparison of the different methods on the last image of the flower dataset that contains 250 frames and little motion. Simulated additive and multiplicative FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0.10$ was added to the frames. (a), (b), (c), (d) are respectively the ground truth clean image, the noisy image with simulated FPN, the image denoised by an average filter and the image denoised by a bilateral filter.

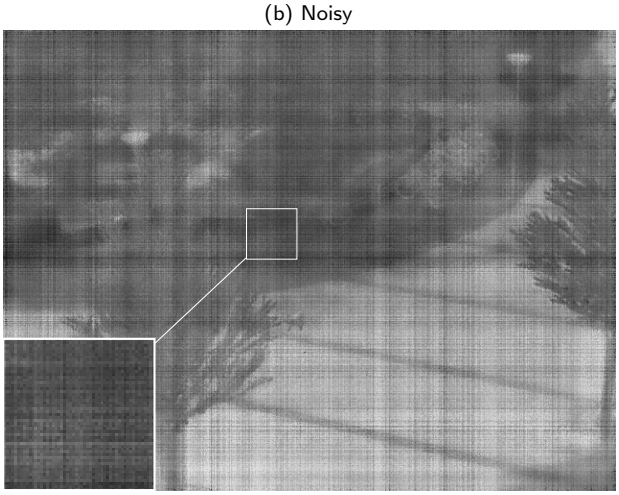


(a) Ground truth

(b) Noisy

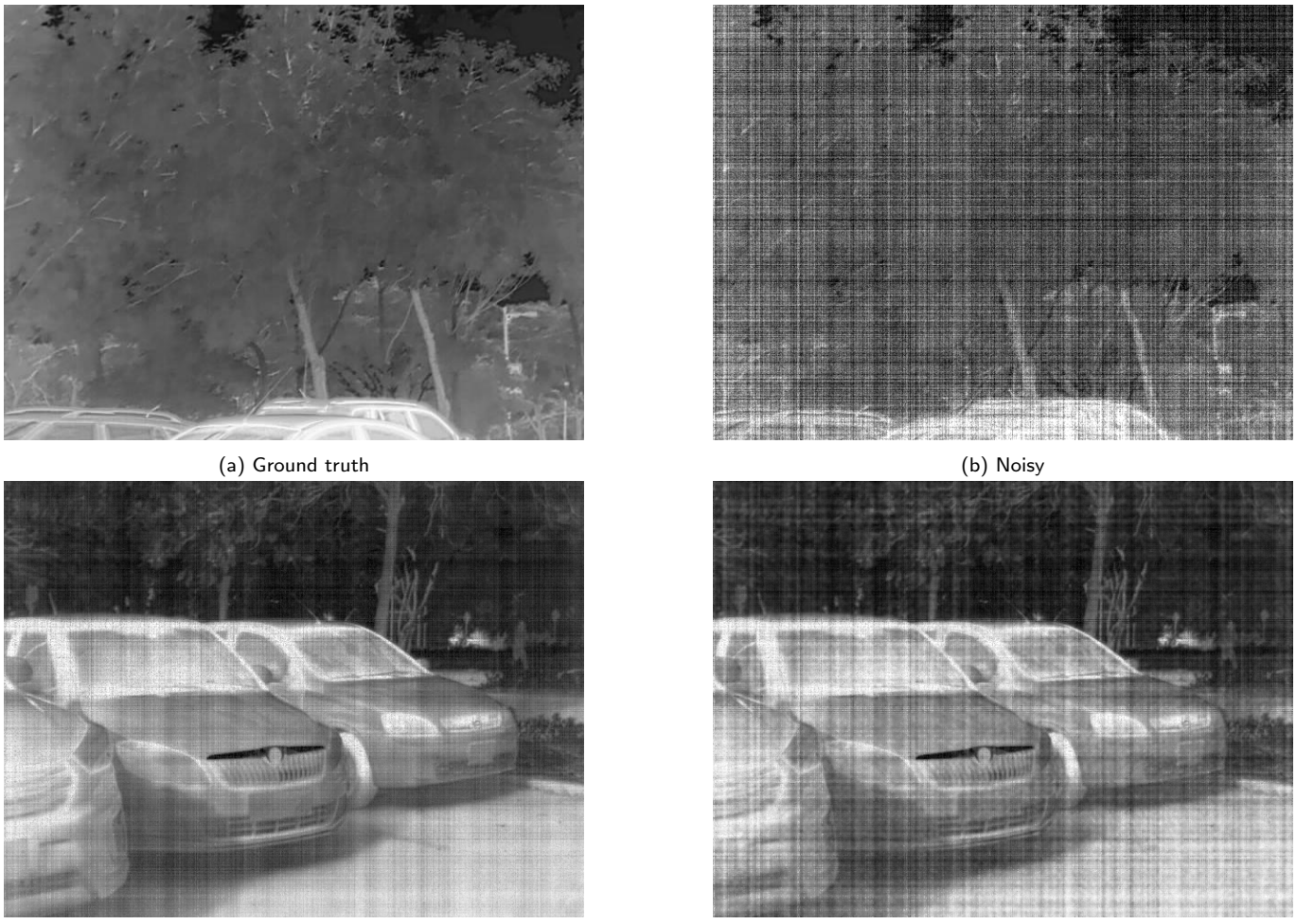


(c) Average filter



(d) Bilateral filter

Figure 8: Comparison of the different methods on the last image of the 8_selma dataset that contains 235 frames and no motion (it is a fixed scene). Simulated additive and multiplicative FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0.10$ was added to the frames. (a), (b), (c), (d) are respectively the ground truth clean image, the noisy image with simulated FPN, the image denoised by an average filter and the image denoised by a bilateral filter.



(a) Ground truth

(b) Noisy

(c) Average filter

(d) Bilateral filter

Figure 9: Comparison of the different methods on the last image of a sample of the 640_ataset dataset. The sample contains 150 images that have nothing in common. Simulated additive and multiplicative FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0.10$ was added to the frames. (a), (b), (c), (d) are respectively the ground truth clean image, the noisy image with simulated FPN, the image denoised by an average filter and the image denoised by a bilateral filter.

4.4 Quantitative Results: Additive FPN

For the experiments, we used $\sigma_{b^w} = \sigma_{b^r} = 15$ for the additive structured and unstructured components of the noise. We did not consider multiplicative noise in this section.

FPNR methods, with additive noise only, produced better results than FPNR methods that remove both additive and multiplicative noise (see Tables 5 and 6). This was expected as the total variance of the noise is lower since there is no signal dependent noise.

4.5 Quantitative and Qualitative Results: Multiplicative and Additive FPN

For the experiments, we used $\sigma_{a^w} = \sigma_{a^r} = 0.10$ for the multiplicative component, both structured and unstructured, and $\sigma_{b^w} = \sigma_{b^r} = 15$ for the additive, structured and unstructured, component of the noise.

Most of the observations for the multiplicative setup are the same as the ones for the additive one. The main difference between these two related to parameter tuning is the nominal step size value. Optimization-based methods consider the same value for both offset and gain parameters.

parameters				PSNR			
size	**step size**	s_x	s_t	**bus**	**flower**	**8_selma**	**640_ataset**
5	5×10^{-3}	-	-	24.16	21.70	22.79	22.80
9	5×10^{-3}	-	-	**24.71**	**21.92**	23.70	23.53
15	5×10^{-3}	-	-	24.49	21.65	**24.02**	23.85
3	5×10^{-3}	-	-	23.01	20.91	21.45	21.71
5	5×10^{-2}	-	-	24.35	20.69	23.09	**24.12**
5	5×10^{-4}	-	-	19.68	18.82	18.70	18.67
5	5×10^{-3}	3	-	24.16	21.70	22.79	22.80
5	5×10^{-3}	15	-	24.16	21.71	22.79	22.80
5	5×10^{-3}	3	9	22.62	18.58	23.03	23.91
5	5×10^{-4}	3	9	24.51	21.25	22.88	23.92
5	5×10^{-4}	3	25	24.63	21.29	22.76	23.92

Table 7: Comparison of the results obtained with different values of the size of the filter, the step size, s_x and s_t, obtained with a bilateral filter and a simulated multiplicative and additive FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0.10$. The PSNR of the last frame is shown. The best results per dataset are highlighted.

Convergence of correction coefficients to the FPN coefficients. If the denoising algorithms work correctly, correction coefficients should converge to the FPN coefficients according to Equation (4). Experiments showed that this does not happen for the offset coefficient as presented in Table 9. In this table, the Relative Root Mean Square Error (RRMSE) between the correction coefficients and FPN coefficients is reported

$$RRMSE(G) = \frac{\|G^* - G\|_2}{\|G^*\|_2}, \quad RRMSE(O) = \frac{\|O^* - O\|_2}{\|O^*\|_2}. \tag{30}$$

Even with the best parameters, the additive correction coefficient was far from the true value, and, in the best case had a RRMSE greater than 50%. The estimation of the multiplicative correction coefficient was better, reaching a RRMSE of less than 20% in the best case.

parameters				PSNR			
size	**step size**	s_x	s_t	**bus**	**flower**	**8_selma**	**640_ataset**
5	5×10^{-3}	-	-	24.29	21.51	25.47	24.37
9	5×10^{-3}	-	-	**24.55**	**21.57**	27.20	25.72
15	5×10^{-3}	-	-	23.69	21.35	**27.68**	**26.59**
3	5×10^{-3}	-	-	23.17	20.84	23.29	22.78
5	5×10^{-2}	-	-	23.59	20.03	25.47	25.19
5	5×10^{-4}	-	-	19.74	18.90	19.55	19.00
5	5×10^{-3}	3	-	24.28	21.51	25.47	24.37
5	5×10^{-3}	15	-	24.28	21.51	25.47	24.37
5	5×10^{-3}	3	9	21.02	17.61	25.34	24.75
5	5×10^{-4}	3	9	24.11	20.84	25.19	25.01
5	5×10^{-4}	3	25	24.24	20.93	25.07	25.01

Table 8: Comparison of the results obtained with different values of the size of the size of the filter, the step size, s_x and s_t, obtained with an average filter and a simulated multiplicative and additive FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0.10$. The PSNR of the last frame is shown. The best results per dataset are highlighted.

	Average filter		Bilateral filter	
Dataset	**gain**	**offset**	**gain**	**offset**
bus	0.1730	0.5991	**0.1724**	**0.5910**
flower	0.1622	0.8181	**0.1614**	**0.7718**
8_selma	**0.1638**	**0.7111**	0.1700	0.6831
640_ataset	**0.1642**	**0.6068**	0.1400	0.5786

Table 9: Comparison between the average filter and the bilateral filter on simulated multiplicative and additive FPN, spatially structured and spatially independent, with a standard deviation of $\sigma_{b^w} = \sigma_{b^r} = 15$, $\sigma_{a^w} = \sigma_{a^r} = 0.10$. The RRMSE between the estimated parameters coefficients and the optimal ones on the last frame is shown. The best results per dataset (according to the PSNR) are highlighted.

5 Conclusion

Temporal high pass filtering methods and optimization-based methods are simple techniques for FPN removal. While their results are not state-of-the-art, they require few operations per pixel and may be considered baseline methods. We provided implementations and tested some optimization-based and temporal high pass filter methods for fixed pattern noise removal. We provided insights into the parameters, demonstrated the limitations and advantages of these families of methods, and proved that temporal high pass filter methods are equivalent to optimization-based methods.

Acknowledgment

Project PID2024-162897NA-I00 funded by MICIU/AEI/10.13039/501100011033/ FEDER, UE. This work was partly funded by AID-DGA (Agence de l'Innovation de Défense à la Direction Générale de l'Armement — Ministère des Armées), and was performed using HPC resources from GENCI-IDRIS (grants 2023-AD011011801R3, 2023-AD011012453R2, 2023-AD011012458R2) and from the "Mésocentre" computing center of CentraleSupélec and ENS Paris-Saclay, supported by CNRS and Région Île-de-France (http://mesocentre.centralesupelec.fr/). Centre Borelli is also with Université Paris Cité, SSA and INSERM.

References

[1] A. Berg, J. Ahlberg, and M. Felsberg, *A Thermal Object Tracking Benchmark*, in IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2015, https://doi.org/10.1109/AVSS.2015.7301772.

[2] K. Cheng, H.-X. Zhou, S. Rong, H. Qin, R. Lai, D. Zhao, and Q. Zeng, *Temporal High-Pass Filter Nonuniformity Correction Algorithm Based on Guided Filter for IRFPA*, 2015, p. 96752S, https://doi.org/10.1117/12.2202781.

[3] A. Friedenberg and I. Goldblatt, *Nonuniformity Two-Point Linear Correction Errors in Infrared Focal Plane Arrays*, Optical Engineering, 37 (1998), pp. 1251 – 1253, https://doi.org/10.1117/1.601890.

[4] R. Hardie, M. Hayat, E. Armstrong, and B. Yasuda, *Scene-Based Nonuniformity Correction with Video Sequences and Registration*, Applied Optics, 39 (2000), pp. 1241–50, https://doi.org/10.1364/AO.39.001241.

[5] J. Harris and Y.-M. Chiang, *Nonuniformity Correction of Infrared Image Sequences Using the Constant-Statistics Constraint*, IEEE Transactions on Image Processing, 8 (1999), pp. 1148–1151, https://doi.org/10.1109/83.777098.

[6] Z. He, Y. Cao, Y. Dong, J. Yang, Y. Cao, and C.-L. Tisse, *Single-Image-Based Nonuniformity Correction of Uncooled Long-Wave Infrared Detectors: a Deep-Learning Approach*, Applied Optics, 57 (2018), pp. D155–D164, https://doi.org/10.1364/AO.57.00D155.

[7] X. Kuang, X. Sui, Q. Chen, and G. Gu, *Single Infrared Image Stripe Noise Removal Using Deep Convolutional Networks*, IEEE Photonics Journal, 9 (2017), pp. 1–13, https://doi.org/10.1109/JPHOT.2017.2717948.

[8] R. Lai, J. Guan, Y. Yang, and A. Xiong, *Spatiotemporal Adaptive Nonuniformity Correction Based on BTV Regularization*, IEEE Access, 7 (2019), pp. 753–762, https://doi.org/10.1109/ACCESS.2018.2885803.

[9] W. Qian, Q. Chen, and G. Gu, *Space Low-Pass and Temporal High-Pass Nonuniformity Correction Algorithm*, Optical Review, 17 (2010), pp. 24–29, https://doi.org/10.1007/s10043-010-0005-8.

[10] B. Ratliff, M. Hayat, and R. Hardie, *An Algebraic Algorithm for Nonuniformity Correction in Focal-Plane Arrays*, Journal of the Optical Society of America A, 19 (2002), pp. 1737–47, https://doi.org/10.1364/JOSAA.19.001737.

[11] R. E. Rivadeneira, A. D. Sappa, and B. X. Vintimilla, *Thermal Image Super-Resolution: a Novel Architecture and Dataset*, in International Conference on Computer Vision Theory and Applications, 2020, pp. 1–2, https://doi.org/10.5220/0009173601110119.

[12] A. Rossi, M. Diani, and G. Corsini, *Bilateral Filter-Based Adaptive Nonuniformity Correction for Infrared Focal-Plane Array Systems*, Optical Engineering, 49 (2010), https://doi.org/10.1117/1.3425660.

[13] D. A. Scribner, K. A. Sarkady, J. T. Caulfield, M. R. Kruer, G. Katz, C. J. Gridley, and C. Herman, *Nonuniformity Correction for Staring IR Focal Plane Arrays Using Scene-Based Techniques*, in Infrared Detectors and Focal Plane Arrays, vol. 1308, International

Society for Optics and Photonics, SPIE, 1990, pp. 224 – 233, https://doi.org/10.1117/12.21730.

[14] D. A. Scribner, K. A. Sarkady, M. R. Kruer, J. T. Caulfield, J. D. Hunt, and C. Herman, *Adaptive Nonuniformity Correction for IR Focal-Plane Arrays Using Neural Networks*, in Infrared Sensors: Detectors, Electronics, and Signal Processing, vol. 1541, International Society for Optics and Photonics, SPIE, 1991, pp. 100 – 109, https://doi.org/10.1117/12.49324.

[15] R. Sheng-Hui, Z. Hui-Xin, Q. Han-Lin, L. Rui, and Q. Kun, *Guided Filter and Adaptive Learning Rate Based Non-Uniformity Correction Algorithm for Infrared Focal Plane Array*, Infrared Physics & Technology, 76 (2016), pp. 691–697, https://doi.org/10.1016/j.infrared.2016.04.037.

[16] S. Torres and M. Hayat, *Kalman Filtering for Adaptive Nonuniformity Correction in Infrared Focal-Plane Arrays*, Journal of the Optical Society of America A, 20 (2003), pp. 470–80, https://doi.org/10.1364/JOSAA.20.000470.

[17] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, *Scikit-Image: Image Processing in Python*, PeerJ, 2 (2014), p. e453, https://doi.org/10.7717/peerj.453.

[18] E. Vera, P. Meza, and S. Torres, *Total Variation Approach for Adaptive Nonuniformity Correction in Focal-Plane Arrays*, Optics Letters, 36 (2011), pp. 172–4, https://doi.org/10.1364/OL.36.000172.

[19] E. Vera and S. Torres, *Fast Adaptive Nonuniformity Correction for Infrared Focal-Plane Array Detectors*, EURASIP Journal on Advances in Signal Processing, 2005 (2005), pp. 1994–2004, https://doi.org/10.1155/ASP.2005.1994.

[20] P. Xiao, Y. Guo, and P. Zhuang, *Removing Stripe Noise From Infrared Cloud Images Via Deep Convolutional Networks*, IEEE Photonics Journal, 10 (2018), pp. 1–1, https://doi.org/10.1109/JPHOT.2018.2854303.

[21] Y. Yuan, Q. Song, X. Guo, and Y. Wang, *A New Temporal High-Pass Adaptive Filter Nonuniformity Correction Based on Rolling Guidance Filter*, 2020, p. 112, https://doi.org/10.1117/12.2539305.

[22] C. Zuo, Q. Chen, G. Gu, and W. Qian, *New Temporal High-Pass Filter Nonuniformity Correction Based on Bilateral Filter*, Optical Review, 18 (2011), pp. 197–202, https://doi.org/10.1007/s10043-011-0042-y.