Published in Image Processing On Line on 2023–11–09.
Submitted on 2022–09–12, accepted on 2023–08–03.
ISSN 2105–1232 © 2023 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.2023.420

A Signal-dependent Video Noise Estimator Via Inter-frame Signal Suppression

Yanhao Li, Marina Gardella, Quentin Bammey, Tina Nikoukhah, Rafael Grompone von Gioi, Miguel Colom, Jean-Michel Morel

Université Paris-Saclay, ENS Paris-Saclay, CNRS, Centre Borelli, Gif-sur-Yvette, France
{yanhao.li, marina.gardella, quentin.bammey, tina.nikoukhah, grompone, mcolomba, morel}@ens-paris-saclay.fr

Communicated by Pablo Arias   *Demo edited by* Yanhao Li

Abstract

We propose a block-based signal-dependent noise estimation method on videos, that leverages inter-frame redundancy to separate noise from signal. Block matching is applied to find block pairs between two consecutive frames with similar signal. Then the Ponomarenko et al. method is extended to video by sorting pairs by their low-frequency energy and estimating noise in the high frequencies. Experiments on a real dataset of drone videos show its performance for different parameter settings and different noise levels. Two extensions of the proposed method using subpixel matching and for multiscale noise estimation are respectively analyzed.

Source Code

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

Keywords: noise estimation; image processing; video processing; noise level function

1 Introduction

Noise estimation is a key preliminary step for various applications in image and video processing. An accurate noise level estimate can significantly boost the performance of downstream applications such as video denoising [2], forgery detection [10], camera identification, camera characterization, and video quality assessment. Most noise estimation methods focus on single images [24, 17, 3, 18, 4, 16, 19, 15]. These methods can be applied to each frame for video noise estimation, but video temporal redundancy is not used. To this aim, we propose an extension of the Ponomarenko et al. method [18, 5] to videos. Since two consecutive frames mostly contain the same signal up to a local motion, noise can be separated by eliminating the underlying scene content within a flexible frame-to-frame difference.

[1] https://doi.org/10.5201/ipol.2023.420

2 Related Work

A significant part of the literature on noise estimation focuses on single-image estimation. This is generally done by finding homogeneous regions where noise is predominant. Noise in these regions is estimated in either the spatial or frequency domain.

There are several approaches to identify homogeneous regions. Tai et al. [24] use a Sobel operator to discard edges, then apply a Laplacian operator to estimate noise. In [17], a hybrid discrete wavelet transform is used for edge-region removal. Colom et al. [3] select a small percentile of the image's blocks with the lowest standard deviations in the high frequencies. The bias induced by this selection criterion is then corrected. Instead, in [18, 4] a small percentile of blocks with the lowest low-frequency variances are selected. Noise is then estimated on the high-frequencies. Mohan et al. [16] first perform intra-image patch matching, then estimate noise in the discrete cosine transform (DCT) domain. Other methods use principal component analysis (PCA) to find homogeneous patches. Pyatykh et al. [19] select patches with similar structure and small variance using PCA, then estimate the noise from the smallest eigenvalues. Similarly, Liu et al. [15] present a PCA-based method for selecting low-rank patches with the smallest high-frequency energy based on their gradients.

The availability of numerous samples in video can improve estimation. Rakhshanfar et al. [20] estimate a noise level function (NLF) by selecting and clustering homogeneous frame regions. Inter-frame analysis then stabilizes temporal noise variations. Buades et al. [2] use [4] to estimate an NLF using simultaneously all frames for the block selection step. Temporal information is further employed to improve accuracy and robustness. In [28], the ideas presented in [1] are extended to the differential image obtained from two consecutive frames. The authors construct a homogeneity measure using high-pass operators along several directions. Then, the variance of the most homogeneous blocks yields the final estimation. Motion estimation is also used to better handle scene changes from one frame to another. Yin et al. [27] estimate the noise in the residual of motion estimation on half the matched blocks, to avoid overestimation due to mismatches. Xiao et al. [25] present an algorithm to suppress video motion by inter-frame block matching, and they estimate the noise level within the inter-frame difference with PCA.

Several approaches jointly exploit the spatial and temporal domains. Ghazal et al. [11] sort 3D cubes by their responses to directional Laplacian operators. The homogeneous ones are used to estimate the noise level in each direction. The mean of these estimations yields a final estimate. An improved version of this method [12] instead uses Laplacian of Gaussian operators and a median estimator. Zlokolica et al. [29] employ spatio-temporal gradients of image sequence content with wavelet transform analysis to determine the noise variation. Spatio-temporal gradients are also used in [26] to select homogeneous cuboids, followed by adaptive noise estimation. Izadi et al. [13] address inter-frame correlated noise and compute the noise level by a combination of spatial and temporal variance estimations.

3 Proposed Method

The extension of the Ponomarenko et al. method [5, 4] (referred to as the Ponomarenko method below) is designed to estimate an NLF from a single image. We extend this method to leverage the temporal redundancy for video noise estimation. Like in the Ponomarenko method, the blocks used for estimation are selected based on their low frequencies, and noise is then estimated in their high frequencies. The key difference is that instead of using the raw blocks in a single frame, we utilize the residual of matched blocks (or difference blocks) between two consecutive frames to estimate the noise.

The proposed algorithm first divides each frame into blocks of size $w \times w$ and matches blocks in pairs of consecutive frames. The matched pairs are then classified into bins according to their intensities. The residual between each block pair is computed; the DCT is then applied to calculate the frequency coefficients of each difference block. Finally, in a similar way to [18, 4], we select the difference blocks with the lowest energy at low frequencies and estimate the noise in the high frequencies of those blocks. The overall pipeline is shown in Figure 1. A preliminary version of this method was described in [14].

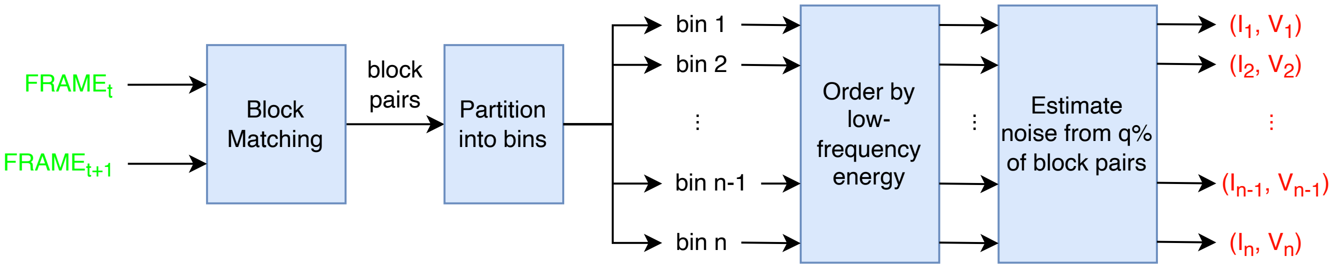


Figure 1: The pipeline of the proposed video noise estimation from two consecutive frames.

3.1 Signal-dependent Noise

In raw data of digital image sensing, the noise is signal-dependent and is modeled by the Poisson-Gaussian model [8]: $\tilde{I}(\mathbf{i}) = I(\mathbf{i}) + \eta_p(I(\mathbf{i})) + \eta_g(\mathbf{i})$, where $\mathbf{i}$ is the pixel position, $\tilde{I}(\mathbf{i})$ is the observed noisy signal, $I(\mathbf{i})$ is the unknown true signal, $\eta_p(I(\mathbf{i}))$ is the signal-dependent error due to the photon-counting process, namely the shot noise, and $\eta_g(\mathbf{i})$ is the signal-independent error mainly including the electric and thermal noise. $\eta_p(I(\mathbf{i}))$ and $\eta_g(\mathbf{i})$ are two independent noise components respectively characterized by *Poissonian* and *Gaussian* distributions,

$$\chi\left(I(\mathbf{i}) + \eta_p(I(\mathbf{i}))\right) \sim \mathcal{P}(\chi I(\mathbf{i})), \tag{1}$$

$$\eta_g(\mathbf{i}) \sim \mathcal{N}(0, \sigma^2), \tag{2}$$

where $\chi > 0$ and $\sigma \geq 0$ are two scalar constants, and $\mathcal{P}$ and $\mathcal{N}$ denote the *Poissonian* and *Gaussian* distribution respectively.

To further simplify our noise estimation problem, the Poisson distribution is approximated by the Gaussian distribution: $\mathcal{P}(\lambda) \approx \mathcal{N}(\lambda, \lambda)$, which is valid when λ is large enough. For digital image sensing this is the case where an image is taken in a good condition of exposure with sufficient photons collected on each pixel. Thus the Poisson noise can be considered as a heteroscedastic Gaussian noise

$$\chi\left(I(\mathbf{i}) + \eta_p(I(\mathbf{i}))\right) \sim \mathcal{P}(\chi I(\mathbf{i})) \approx \mathcal{N}(\chi I(\mathbf{i}), \chi I(\mathbf{i})) \;\Rightarrow\; \eta_p(I(\mathbf{i})) \sim \mathcal{N}(0, \frac{1}{\chi} I(\mathbf{i})). \tag{3}$$

Finally the signal-dependent noise model is rewritten as

$$\tilde{I}(\mathbf{i}) = I(\mathbf{i}) + (\frac{1}{\chi} I(\mathbf{i}) + \sigma)\eta(\mathbf{i}), \tag{4}$$

with $\eta(\mathbf{i}) \sim \mathcal{N}(0, 1)$ the independent standard Gaussian noise. The noise is thus an additive Gaussian noise whose variance is an affine function of the true signal. In this paper, we are going to estimate the relationship between the true signal intensity $I(\mathbf{i})$ and the noise variance $\mathrm{Var}(\tilde{I}(\mathbf{i}))$, expressed by a noise level function (NLF) $g : I(\mathbf{i}) \mapsto \mathrm{Var}(\tilde{I}(\mathbf{i})) = \alpha + \beta I(\mathbf{i})$ with $\alpha = \sigma^2$ and $\beta = \frac{1}{\chi^2}$ two constant scalar parameters.

3.2 Block Matching

Each frame is divided into overlapping blocks of size $w \times w$ with stride 1. Let $\{\tilde{\mathbf{U}}_k^t : k = 1, \ldots, K\}$ denote the set of blocks extracted from frame t. The purpose of block matching is to find, for each block $\tilde{\mathbf{U}}_k^t$ in frame t, a corresponding block $\tilde{\mathbf{U}}_{k'}^{t+1}$ in frame $t+1$ having similar signal, so that the noise can be estimated from their difference without interference of the scene signal. Indeed, if $\tilde{\mathbf{U}}_k^t = \mathbf{U}_k^t + \mathbf{n}_k^t$ with $\mathbf{U}_k^t$ the true signal and $\mathbf{n}_k^t$ the noise, then their difference has twice the noise

$$\tilde{\mathbf{U}}_k^t - \tilde{\mathbf{U}}_{k'}^{t+1} = (\mathbf{U}_k^t - \mathbf{U}_{k'}^{t+1}) + (\mathbf{n}_k^t - \mathbf{n}_{k'}^{t+1}) \approx \mathbf{n}_k^t - \mathbf{n}_{k'}^{t+1} \tag{5}$$

where the true signals of the two blocks $\mathbf{U}_k^t$ and $\mathbf{U}_{k'}^{t+1}$ are similar and the difference cancels them out.

The usual way to determine k' is to search the block among the candidate blocks in frame $t+1$ that minimizes the similarity distance within a search window. However, using the same $w \times w$ block for both matching and noise estimation would lead to an underestimation of the noise in the block difference. Indeed, if several candidate blocks contain the same signal as $\tilde{\mathbf{U}}_k^t$, then the block matching will try to find the block that also minimizes the noise difference. This over-fitting hazard is common in flat or smooth areas where neighboring blocks have the same signal. To avoid this underestimation, we separate the pixels used for block matching from those used for noise estimation. We only use the surrounding pixels of the block as matching area S to find the best-matching block (see Figure 2).

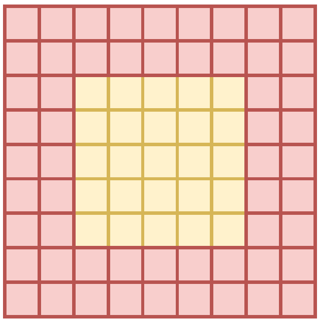


Figure 2: Surrounding pixels (in red) as matching area S, and central block (in yellow) of size $w \times w$ for noise estimation.

According to our observations preprocessing the images by applying some blur can enhance the robustness of block matching to noise, thus a simple 5×5 Gaussian blur with standard deviation $\sigma = 1$ is applied to both images before matching. Note $\mathbf{R}^t(\tilde{\mathbf{U}}_k^t)$ as the surrounding ring of pixels in the *blurred* frame t of a block $\tilde{\mathbf{U}}_k^t$. Using the sum of squared differences (SSD) as the similarity metric, the best matched block $\tilde{\mathbf{U}}_{k'}^{t+1}$ of the queried block $\tilde{\mathbf{U}}_k^t$ is then determined by

$$k' = \underset{c \in \text{candidates}}{\arg\min} \ \text{SSD}\left(\mathbf{R}^t(\tilde{\mathbf{U}}_k^t), \mathbf{R}^{t+1}(\tilde{\mathbf{U}}_c^{t+1})\right) := m(k). \tag{6}$$

where $m : k \mapsto k'$ maps the block index in frame t to the index of its matched block in frame $t+1$, and the candidate blocks $\{\tilde{\mathbf{U}}_c^{t+1}\}$ are those no more than s pixels away from $\tilde{\mathbf{U}}_k^t$ on both the x-axis and the y-axis. The search is discarded for the blocks $\tilde{\mathbf{U}}_k^t$ too close to the border of the image which lack some candidates within the search range $[-s, +s]$. The matching step results in a set of matched block pairs $\left\{\left(\tilde{\mathbf{U}}_k^t, \tilde{\mathbf{U}}_{m(k)}^{t+1}\right)\right\} := \left\{\left(\tilde{B}_k^t, \tilde{B}_k^{t+1}\right)\right\}$ with now $\left(\tilde{B}_k^t, \tilde{B}_k^{t+1}\right)$ to note the matched block pairs between two frames.

Given that noise is clipped in saturated pixels, the noise in the saturated zone does not fit the affine noise model. Here our purpose is to estimate the noise model of the main zone of luminosity where the noise is not clipped, hence we discard the block pairs that have at least one saturated pixel before grouping them into bins. Since the saturated pixel value can be different depending on the camera manufacturers, we determine the saturated value M by looking at the maximum value of the two frames at time t and $t+1$. The block pairs with at least one pixel equal to M will be removed from the matched block pair candidates. Finally, the block matching step outputs a set of matched block pairs $\mathbf{B}_{\text{match}} = \left\{ \left(\tilde{B}_k^t, \tilde{B}_k^{t+1} \right), k = 1, \ldots, K' \mid \mathtt{max}(\tilde{B}_k^t) < M \wedge \mathtt{max}(\tilde{B}_k^{t+1}) < M \right\}$ where $\tilde{B}_k^t$ and $\tilde{B}_k^{t+1}$ are two matched *central* blocks of size $w \times w$ (the block in yellow in Figure 2).

Besides, the image integral technique [7] is employed to further speed up the aggregation of pixel-wise distances for block matching, which will be detailed in Algorithm 1 and Algorithm 2.

3.3 Block Pair Partitioning

Since the noise is signal-dependent, we group the block pairs by their intensities and estimate the noise levels accordingly. The block pairs in a bin have similar true signal intensities, thus their noises have similar variances and can be approximated by additive white Gaussian noise (AWGN). Each bin of block pairs will then result in one intensity and one noise variance that together form a control point of the noise level function.

With the output matched blocks $\mathbf{B}_{\text{match}} = \left\{ \left(\tilde{B}_k^t, \tilde{B}_k^{t+1} \right), k = 1, \ldots, K' \right\}$ from the previous block matching step, we now partition them into b bins of the same size depending on the mean intensities of the block pairs. The block pairs need to be sorted by their mean intensities: $\mathbf{B}_{\text{sort}} = \left\{ \left(\tilde{B}_{(k)}^t, \tilde{B}_{(k)}^{t+1} \right), k = 1, \ldots, K' \right\}$ where $_{(\cdot)}$ is to note the indices of the sorted block pairs

$$\mathtt{mean}(\tilde{B}_{(l)}^t) + \mathtt{mean}(\tilde{B}_{(l)}^{t+1}) \leq \mathtt{mean}(\tilde{B}_{(l+1)}^t) + \mathtt{mean}(\tilde{B}_{(l+1)}^{t+1}) \quad \forall l = 1, \ldots, K'-1, \tag{7}$$

And the n-th bin of block pairs will be formed by

$$\mathbf{B}_n = \left\{ \left(\tilde{B}_{(k)}^t, \tilde{B}_{(k)}^{t+1} \right), k = \lfloor \frac{K'}{b} \rfloor \times (n-1) + 1, \ldots, \lfloor \frac{K'}{b} \rfloor \times n \right\} \subset \mathbf{B}_{\text{sort}}. \tag{8}$$

The implementation of this step is detailed in Algorithm 3.

3.4 Block Selection by Low-frequency Energy

To simplify the notation, the block pairs in the n-th bin in Equation (8) are now reindexed to $\mathbf{B}_n = \left\{ \left(\tilde{B}_{(k)}^t, \tilde{B}_{(k)}^{t+1} \right), k = 1, \ldots, \lfloor \frac{K'}{b} \rfloor \right\}$. Before estimating the noise level from the block pairs in $\mathbf{B}_n$, we further select the difference blocks whose signals are well removed by subtraction. Indeed, block matching cannot output perfectly-matched pairs due to the presence of noise, the matching precision limit, and changes in the scene content over time.

The idea is to select a small quantile q of block pairs for each bin whose differences contain the least signal. Contrary to the noise that exists equally in all the frequencies, the signal mainly exists in the low frequencies and has a fast frequency decay. With the assumption inherited from [18] that visual signals have larger low-frequency components than high-frequency components, we select the block pairs whose differences have the lowest energies in low frequencies, then the high-frequency signal residuals of the selected block pairs are also considered to be the smallest. The DCT-II is performed on the difference blocks to get the transformed blocks $\{D_k = \mathrm{DCT}(\tilde{B}_{(k)}^t - \tilde{B}_{(k)}^{t+1}) \in \mathbb{R}^{w \times w}\}$, where $\mathrm{DCT}\colon \mathbb{R}^{w \times w} \to \mathbb{R}^{w \times w}$ is defined by

$$\forall (i,j) \in [\![0, w-1]\!]^2, \;\; \mathrm{DCT}(x)_{i,j} = X_{i,j} = \alpha_{i,j} \sum_{p=0}^{w-1} \sum_{q=0}^{w-1} x_{p,q} \cos\left[\frac{\pi}{w} \left(p + \frac{1}{2} \right) i \right] \cos\left[\frac{\pi}{w} \left(q + \frac{1}{2} \right) j \right],$$

with the coefficients $\alpha_{i,j}$ defined by

$$\alpha_{i,j} = \begin{cases} \frac{1}{w} & \text{if } (i,j) = (0,0) \\ \frac{\sqrt{2}}{w} & \text{if } i = 0 \text{ and } j > 0, \text{or } j = 0 \text{ and } i > 0 \\ \frac{2}{w} & \text{otherwise.} \end{cases} \tag{9}$$

Then each transformed block D_k is divided into low- and high-frequency components by introducing a threshold T: $D_k(i,j)$ is considered as low frequency if $\|(i,j)\|_1 \leq T$ and as high frequency otherwise. Here (i,j) is the 2-dimensional DCT coefficient entry and $\|(i,j)\|_1 := i + j$ is the l_1 norm. The low-frequency energy of D_k is computed by

$$V_k^L = \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} \left[D_k\left(i,j\right)\right]^2 \mathbb{1}_{\{\|(i,j)\|_1 \leq T\}}, \tag{10}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. A quantile q of block pairs with the lowest low-frequency energies are selected as signal-free blocks, noted as $\mathbf{B}'_n = \{(\tilde{B}^t_{[k]}, \tilde{B}^{t+1}_{[k]}), k = 1, \ldots, \lfloor q|\mathbf{B}_n|\rfloor\} \subset \mathbf{B}_n$ with $_{[\cdot]}$ indicating the block indices sorted by increasing values of V_k^L.

3.5 Noise Estimation from High Frequencies

The noise variance of the n-th bin is estimated from the high frequencies of the previously selected block pairs whose high-frequency signal residuals are considered to be the smallest. The DCT coefficient blocks of the inter-block differences of the selected block pairs $\mathbf{B}'_n$ are first computed: $\mathbf{D}'_n = \{D_{[k]} = \text{DCT}(\tilde{B}^t_{[k]} - \tilde{B}^{t+1}_{[k]}), k = 1, \ldots, \lfloor q|\mathbf{B}_n|\rfloor\}$. Then the variance of each high-frequency coefficient is calculated across the signal-free blocks by

$$V_n^H(i,j) = \frac{1}{|\mathbf{D}'_n|} \sum_{k=1}^{|\mathbf{D}'_n|} \left[D_{[k]}, (i,j)\right]^2 \tag{11}$$

with $\|(i,j)\|_1 > T$. The noise variance of the bin is estimated by

$$V_n = \frac{1}{2}\text{median}(\{V_n^H(i,j), \|(i,j)\|_1 > T\}). \tag{12}$$

Here the median value is halved since the noise of a block difference has twice the original variance.

To get the noise curve, the mean intensity I_n of all the selected blocks in $\mathbf{B}'_n$ is also calculated

$$I_n = \frac{1}{|\mathbf{B}'_n|} \sum_{k=1}^{|\mathbf{B}'_n|} \frac{1}{2}\big(\text{mean}(\tilde{B}^t_{[k]}) + \text{mean}(\tilde{B}^{t+1}_{[k]})\big), \tag{13}$$

Finally from b bins we get b (intensity, variance) pairs $\{(I_n, V_n), n = 1, \ldots, b\}$ as the estimated noise curve.

4 Fusion of Multiple Noise Curves

The method described above is designed for one pair of successive frames, while for a video one has access to multiple frame pairs for noise estimation. Here we introduce a simple fusion technique to extract one noise curve from multiple noise curves by median estimation.

For each pair of successive frames, the aforementioned method outputs a noise curve with b control points. Then, a set of T curves are collected from a video sequence of $T + 1$ frames. The t-th curve is denoted by $\{(I_n^{(t)}, V_n^{(t)}), n = 1, \ldots, b\}$, where the intensities $\{I_n^{(t)}\}$ are in the increasing order for each curve. The fused curve will also have b control points, noted as $\{(\tilde{I}_n, \tilde{V}_n), n = 1, \ldots, b\}$. The intensity value of each control point is calculated from the N intensity values at the same position

$$\tilde{I}_n = \text{median}(\{I_n^{(t)}, t = 1, \dots, T\}), \tag{14}$$

Before computing the noise variances, each of the separate discrete noise curve is extended to a continuous piece-wise linear curve, where the values between two control points are determined by linear interpolation. The t-th continuous noise curve is defined by $f^{(t)} : [I_1^{(t)}, I_b^{(t)}] \to \mathbb{R}_+$. The noise variance value related to $\tilde{I}_n$ is

$$\tilde{V}_n = \text{median}(\{f^{(t)}(\tilde{I}_n), t = 1, \dots, T\}) \tag{15}$$

Note that $\tilde{I}_n$ might be out of the input range of $f^{(t)}$ for some curves, thus only the valid values of $\{f^{(t)}(\tilde{I}_n), t = 1, \dots, T\}$ will be taken into account by the median operation. For multi-channel videos, one noise curve is first obtained for each channel and each frame pair, and the fusion technique is applied independently to each channel to output one fused curve per channel.

The implementation of the fusion technique is detailed in Algorithm 8.

5 Detailed Implementation

The block matching described in Section 3.2 is implemented by Algorithm 1. For each block in shape $w \times w$ in the reference image (frame t), it searches for the best block in the moving image (frame $t + 1$) within a search range s that has the minimum matching cost for the surrounding ring of the block. The thickness of the ring is th.

The image integral technique is applied to efficiently compute the sums of pixels in the overlapping $w \times w$ blocks of an image of size $H \times W$. The implementation is shown in Algorithm 2.

The block pairs are partitioned according to their intensities, as described in Section 3.3 and implemented by Algorithm 3.

The step of selecting the block pairs with the lowest low-frequency energies in their differences is shown in Algorithm 4.

The noise variance is estimated from the high frequencies of the selected blocks, which is described in Section 3.5, and is implemented by Algorithm 5.

The low-frequency energy of a DCT block is computed in Algorithm 6.

The main function of the whole pipeline is shown in Algorithm 7.

The implementation for the fusion of multiple noise curves described in Section 4 is as shown in Algorithm 8.

Algorithm 1: Compute the blocks in the moving image that match the blocks in the reference image by matching their surrounding rings (**pixelMatch**)

```
Input img_ref: reference image of size $H \times W$
Input img_mov: moving image of size $H \times W$
Input w: block size
Input th: thickness of surrounding ring
Input s: search range
Output $\boldsymbol{B}_{ref}$: list of matched blocks in reference image
Output $\boldsymbol{B}_{mov}$: list of matched blocks in moving image

img_blur_ref ← GaussianBlur(img_ref)
img_blur_mov ← GaussianBlur(img_mov)
/* Prepare costs of block matching for offsets in a searching window */
offsets ← $\{-s, \ldots, s\} \times \{-s, \ldots, s\}$
costs_offsets ← empty map of size $(2s + 1, 2s + 1)$ that maps offsets to 2D arrays
outer_sz ← $2 \times th + w$                        # outer size of the surrounding ring
$\boldsymbol{B}_{ref}$ ← $\emptyset$                # array to store the matched blocks in img_ref
$\boldsymbol{B}_{mov}$ ← $\emptyset$                # array to store the matched blocks in img_mov
for off ∈ offsets do
    img_diff ← image of size $(H - 2s) \times (W - 2s)$
    for $\boldsymbol{p}$ ∈ img_diff do
        img_diff($\boldsymbol{p}$) ← $(\text{img\_blur\_ref}(\boldsymbol{p} + s) - \text{img\_blur\_ref}(\boldsymbol{p} + s + \text{off}))^2$
    costs_outer_blocks ← Convolve2dSum(img_diff, outer_sz)
                                        # Image of costs of outer blocks, see Algorithm 2
    costs_inner_blocks ← Convolve2dSum(img_diff, w)
                                        # Image of costs of inner blocks, see Algorithm 2
    costs_inner_blocks ← cropped costs_inner_blocks by rectangle
      $[1 + th, H - 2s - w + 1 - th] \times [1 + th, W - 2s - w + 1 - th]$   # Crop the inner cost image
      to fit the outer cost image
    costs_offsets(off) ← costs_outer_blocks − costs_inner_blocks
max_val ← max(max(img_ref), max(img_ref))
/* Select the best blocks in img_mov with the smallest costs */
for $\boldsymbol{p}$ ← $\{1, \ldots, H - 2s - \text{outer\_sz} + 1\} \times \{1, \ldots, W - 2s - \text{outer\_sz} + 1\}$ do
    cost_best ← $+\infty$
    off_best ← $(0, 0)$
    for off ∈ offsets do
        cost ← costs_offsets(off)($\boldsymbol{p}$)
        if cost < cost_best then
            off_best ← off
            cost_best ← cost
    $B_r$ ← w × w block of img_ref with top-left pixel at $(\boldsymbol{p} + s + th)$      # queried central block
    $B_m$ ← w × w block of img_mov with top-left pixel at $(\boldsymbol{p} + \text{off\_best} + s + th)$
                                                                   # matched central block
    if $max(B_r)$ < max_val and $max(B_m)$ < max_val then
        append $B_r$ to $\boldsymbol{B}_{ref}$
        append $B_m$ to $\boldsymbol{B}_{mov}$
return $\boldsymbol{B}_{ref}$, $\boldsymbol{B}_{mov}$
```

Algorithm 2: Compute the sums of pixels of overlapping blocks (**convolve2dSum**)

```
Input img: image of size $H \times W$
Input $w$: block size
Output img_sum: image of size $(H - w + 1) \times (W - w + 1)$

img_tmp ← image of size $H \times W$
for $i \leftarrow 1$ to $H$ do
    /* Integral along rows */
    img_tmp$(i, 1)$ ← img$(i, 1)$
    for $j \leftarrow 2$ to $W$ do
        img_tmp$(i, j)$ ← img_tmp$(i, j - 1)$ + img$(i, j)$
    /* Sums of w pixels along rows */
    img_sum$(i, 1)$ ← img_tmp$(i, w)$
    for $j \leftarrow 2$ to $W - w + 1$ do
        img_sum$(i, j)$ ← img_tmp$(i, j + w - 1)$ − img_tmp$(i, j - 1)$
for $j \leftarrow 1$ to $W$ do
    /* Integral along columns */
    img_tmp$(1, j)$ ← img_sum$(1, j)$
    for $i \leftarrow 2$ to $H$ do
        img_tmp$(i, j)$ ← img_tmp$(i - 1, j)$ + img_sum$(i, j)$
    /* Sums of $w$ pixels along rows */
    img_sum$(1, j)$ ← img_tmp$(w, j)$
    for $i \leftarrow 2$ to $H - w + 1$ do
        img_sum$(i, j)$ ← img_tmp$(i + w - 1, j)$ − img_tmp$(i - 1, j)$
return img_sum
```

Algorithm 3: Partition the block pairs into bins according to their mean intensities (**partition**)

```
Input $\boldsymbol{B}_{ref}$: a list of $N$ matched blocks in reference image
Input $\boldsymbol{B}_{mov}$: a list of $N$ matched blocks in moving image
Input $b$: number of bins
Output $\boldsymbol{R}_{ref}$: a list of $b$ bins, each bin contains $\lfloor \frac{N}{b} \rfloor$ blocks in reference image that have similar
        intensities
Output $\boldsymbol{R}_{mov}$: a list of $b$ bins, each bin contains $\lfloor \frac{N}{b} \rfloor$ block in moving image that have similar
        intensities
$\boldsymbol{L} \leftarrow$ a list of $N$ scalars                    # mean intensities of the block pairs
for $i \leftarrow 1$ to $N$ do
    $B_r \leftarrow \boldsymbol{B}_{ref}(i)$
    $B_m \leftarrow \boldsymbol{B}_{mov}(i)$
    $\boldsymbol{L}(i) \leftarrow (\texttt{mean}(B_r) + \texttt{mean}(B_m))/2$    # mean intensity of the two matched blocks
$\boldsymbol{I} \leftarrow \text{argsort}(\boldsymbol{L})$          # indices that sort $\boldsymbol{L}$ in the increasing order
$\boldsymbol{R}_{ref} \leftarrow \emptyset$
$\boldsymbol{R}_{mov} \leftarrow \emptyset$
idx $\leftarrow 1$
for $i \leftarrow 1$ to $b$ do
    bin_ref $\leftarrow \emptyset$                                  # a bin for blocks in reference image
    bin_mov $\leftarrow \emptyset$                                  # a bin for blocks in moving image
    for $j \leftarrow 1$ to $\lfloor \frac{N}{b} \rfloor$ do
        append $\boldsymbol{B}_{ref}(\boldsymbol{I}(\text{idx}))$ to bin_ref
        append $\boldsymbol{B}_{mov}(\boldsymbol{I}(\text{idx}))$ to bin_mov
        idx $\leftarrow$ idx $+ 1$
    append bin_ref to $\boldsymbol{R}_{ref}$
    append bin_mov to $\boldsymbol{R}_{mov}$
return $\boldsymbol{R}_{ref}, \boldsymbol{R}_{mov}$
```

Algorithm 4: Select a quantile of block pairs whose difference have the least low-frequency energies (**selectBlockPairs**)

```
Input $\boldsymbol{B}_{ref}$: list of $N$ matched blocks in the reference frame
Input $\boldsymbol{B}_{mov}$: list of $N$ matched blocks in the moving frame
Input $q$: quantile of block pairs that have the lowest inter-block energies at low frequencies
Input $T$: threshold separating low and high frequencies
Output $\boldsymbol{B}'_{ref}$: list of $q$ quantile of blocks in the reference frame
Output $\boldsymbol{B}'_{mov}$: list of $q$ quantile of blocks in the moving frame

$\boldsymbol{E} \leftarrow \emptyset$                      # a list of low-frequency energies of block differences
for $i \leftarrow 1$ to $N$ do
    $B_r \leftarrow \boldsymbol{B}_{ref}(i)$
    $B_m \leftarrow \boldsymbol{B}_{mov}(i)$
    $D \leftarrow$ DCT-II$(B_r - B_m)$           # compute the DCT of the difference of two blocks
    $e \leftarrow$ computeLowFreqEnergy$(D, T)$                              # see Algorithm 6
    append e to $\boldsymbol{E}$
$\boldsymbol{I} \leftarrow \text{argsort}(\boldsymbol{E})$                 # indices that sort $\boldsymbol{E}$ in the increasing order
$\boldsymbol{B}'_{ref} \leftarrow \emptyset; \boldsymbol{B}'_{mov} \leftarrow \emptyset;$
for $i \leftarrow 1$ to $\lfloor |\boldsymbol{I}| \times q \rfloor$ do
    append $\boldsymbol{B}_{ref}(\boldsymbol{I}(i))$ to $\boldsymbol{B}'_{ref}$
    append $\boldsymbol{B}_{mov}(\boldsymbol{I}(i))$ to $\boldsymbol{B}'_{mov}$
return $\boldsymbol{B}'_{ref}, \boldsymbol{B}'_{mov}$
```

Algorithm 5: Compute the noise variance from the high frequencies of the inter-block residuals (**computeVarianceFromPairs**)

```
Input $\boldsymbol{B}_{ref}$: list of $N$ matched blocks of size $w \times w$ in the reference frame
Input $\boldsymbol{B}_{mov}$: list of $N$ matched blocks of size $w \times w$ in the moving frame
Input $T$: threshold for separating the entries for low- and high-frequency DCT coefficients
Output variance: estimated noise variance from the block pairs

$\boldsymbol{D}_H \leftarrow \emptyset$                                   # list of high-frequency DCT blocks
num_high $\leftarrow 0$                           # number of high frequencies of a DCT block
for $i \leftarrow 1$ to $N$ do
    $U_r \leftarrow \boldsymbol{B}_{ref}(i);\ U_m \leftarrow \boldsymbol{B}_{mov}(i);$                # retrieve two matched blocks
    $D \leftarrow$ DCT-II$(U_r - U_m)$                           # DCT of the inter-block difference
    $D_H \leftarrow \emptyset$                                 # block of high-frequency coefficients
    for $x \leftarrow 0$ to $w - 1$ do
        for $y \leftarrow 0$ to $w - 1$ do
            if $x + y > T$ then append $D(x, y)$ to $D_H$
    append $D_H$ to $\boldsymbol{D}_H$
    num_high $\leftarrow |D_H|$          # needed only once for subsequent variance computation
$\boldsymbol{V}_H \leftarrow \emptyset$                                              # High-frequency variances
for $j \leftarrow 1$ to num_high do
    $\boldsymbol{V}_H(j) \leftarrow \frac{1}{N} \sum_{D_H \in \boldsymbol{D}_H} D_H(j)^2$
variance $\leftarrow$ median$(\boldsymbol{V}_H)$
return variance
```

Algorithm 6: Compute the low-frequency energy of a DCT block (**computeLowFreqEnergy**)

```
Input $D$: a DCT block of size $w \times w$
Input $T$: threshold for separating the entries for low- and high-frequency DCT coefficients
Output $e$: low-frequency energy

$e \leftarrow 0$
for $i \leftarrow 1$ to $w$ do
    for $j \leftarrow 1$ to $w$ do
        if $i + j \leq T$ then $e \leftarrow e + D(i,j)^2$
return $e$
```

Algorithm 7: Estimate noise curve from two successive images

```
Input img_ref: reference image of size $H \times W$
Input img_mov: moving image of size $H \times W$
Input $w$: block size
Input $T$: frequency separator
Input $q$: quantile of blocks used for estimation
Input $th$: thickness of compared area
Input $s$: search range
Input $b$: number of bins
Output intensities: intensities of the noise curve
Output variances: noise variances of the noise curve

$\boldsymbol{B}_{ref}, \boldsymbol{B}_{mov} \leftarrow$ pixelMatch(img_ref, img_mov, $w$, $th$, $s$)    # get $N$ matched pairs of blocks from two images, see Algorithm 1
$\boldsymbol{R}_{ref}, \boldsymbol{R}_{mov} \leftarrow$ partition($\boldsymbol{B}_{ref}$, $\boldsymbol{B}_{mov}$, $b$)    # get $b$ bins of block pairs, see Algorithm 3
intensities $\leftarrow \emptyset$
variances $\leftarrow \emptyset$
for $i \leftarrow 1$ to $b$ do
    $\boldsymbol{B}_{ref} \leftarrow \boldsymbol{R}_{ref}(i)$
    $\boldsymbol{B}_{mov} \leftarrow \boldsymbol{R}_{mov}(i)$
    $\boldsymbol{B}'_{ref}, \boldsymbol{B}'_{mov} \leftarrow$ selectBlockPairs($\boldsymbol{B}_{ref}, \boldsymbol{B}_{mov}, q, T$)    # select a small quantile of blocks with the lowest low-frequency energies, see Algorithm 4
    intensity $\leftarrow \frac{1}{|\boldsymbol{B}'_{ref}|+|\boldsymbol{B}'_{mov}|} \times (\sum_{U \in \boldsymbol{B}'_{ref}} \mathtt{mean}(U) + \sum_{U \in \boldsymbol{B}'_{mov}} \mathtt{mean}(U))$    # mean intensity of all the selected block pairs
    variance $\leftarrow$ computeVarianceFromPairs($\boldsymbol{B}'_{ref}, \boldsymbol{B}'_{mov}, T$)    # see Algorithm 5
    append intensity to intensities
    append variance to variances
return intensities, variances
```

Algorithm 8: Compute a median curve from a set of individual curves, described in Section 4 (**computeMedianCurve**)

```
Input in_curves: a list of curves $\{\{(I_i^{(n)}, V_i^{(n)}), i = 1, \dots, b\}, n = 1, \dots, N\}$, where $b$ is the number
                 of bins and $N$ is the number of curves
Output out_curve: fused curve by median method, $\{(\tilde{I}_i, \tilde{V}_i), i = 1, \dots, b\}$

out_curve ← $\emptyset$
for $i \leftarrow 1$ to $b$ do
    $I \leftarrow$ median($\{I_i^{(n)}, n = 1, \dots, N\}$)            # median intensity for the $i$-th bin
    $\boldsymbol{V}_{bin} \leftarrow \emptyset$                        # individual variances for the $i$-th bin
    for $n \leftarrow 1$ to $N$ do
        if $I \notin [I_1^{(n)}, I_b^{(n)}]$ then
            continue                                                   # discard the variance if $I$ is out of range
        find the $j$-th interval $[I_j^{(n)}, I_{j+1}^{(n)}]$ such that $I \in [I_j^{(n)}, I_{j+1}^{(n)}]$
        $V \leftarrow \frac{V_{j+1}^{(n)} \times (I - I_j^{(n)}) + V_j^{(n)} \times (I_{j+1}^{(n)} - I)}{I_{j+1}^{(n)} - I_j^{(n)}}$   # linear interpolation
        append $V$ to $\boldsymbol{V}_{bin}$                           # median variance for the $i$-th bin
    $V \leftarrow$ median($\boldsymbol{V}_{bin}$)
    out_curve ← out_curve $\cup$ $(I, V)$
    append $(I, V)$ to out_curve
return out_curve
```

6 Experiments

The proposed method was evaluated on a dataset of synthetic drone videos whose previews are shown in Figure 3. The videos had originally a resolution of 2160 × 3840 px. The red channels were extracted and blurred with a Gaussian kernel with parameter σ for anti-aliasing, then downsampled by a factor 4 to 540 × 960 px. It is mentioned in [6] that a Gaussian blur with $\sigma = 0.8\sqrt{s^2 - 1}$ before subsampling a natural well sampled image by factor s ensures that the subsampled image is also well sampled, thus $\sigma = 3.1$ in our case. This led to nearly noiseless videos that were used as reference.

Intensity-dependent simulated noise was then added to the frames with an NLF of the form $g(I) = \alpha + \beta \times I$ (see Section 3.1) as ground truth for evaluation. We used three noise models, $(\alpha, \beta) \in \{(0.2, 0.2), (0.8, 0.8), (3.2, 3.2)\}$ to simulate weak, medium and strong noises. For each noise model, a sequence of 20 noisy frames was generated from successive frames of a video.

The default parameters on this dataset were: block size for noise estimation $w = 20$, threshold for separating low and high frequencies $T = 21$, thickness of the surrounding area for block matching 3, number of bins $b = 16$, quantile of blocks with lowest low-frequency energy $q = 5\%$, and search window $[-s, s] \times [-s, s]$ for block matching with $s = 5$px. These settings were chosen based on the balance of the method's performances for all video sequences and all noise levels. The impacts of the parameters will be further discussed.

The mean relative error (MRE) was used to measure the accuracy of an estimated noise curve,

$$\text{MRE} = \frac{1}{N} \sum_{n=1}^{N} \frac{|V_n - g(I_n)|}{g(I_n)}, \tag{16}$$

with $\{V_n\}$ the noise variances, $\{I_n\}$ the intensities and N the number of discrete estimates of the curve. The reason for using the relative error instead of the absolute error of the whole noise curve

Figure 3: The 16 videos of the drone video dataset.

was to make a fair comparison of the noise estimates on the whole curve. Indeed, the error of an estimated variance is approximately proportional to the expected value of the estimated variance, thus the relative error is independent of the ground truth variance. For each video, the average MRE of noise curves over all the frame pairs ("w/o fusion") and the MRE of the fused noise curve by the median technique described in Section 4 ("with fusion") were calculated, with results shown in Table 1.

video sequence	$\alpha = 0.2$ $\beta = 0.2$		$\alpha = 0.8$ $\beta = 0.8$		$\alpha = 3.2$ $\beta = 3.2$	
	w/o fusion	with fusion	w/o fusion	with fusion	w/o fusion	with fusion
01	3.4	3.0 (0.4)	2.2	1.7 (0.5)	1.5	0.8 (0.7)
02	4.2	2.9 (1.3)	2.8	2.6 (0.2)	1.7	0.9 (0.8)
03	3.1	2.8 (0.3)	2.1	1.5 (0.6)	1.5	0.9 (0.6)
04	3.7	3.0 (0.7)	2.5	2.1 (0.4)	1.6	0.9 (0.7)
05	5.5	5.5 (0.0)	2.4	2.1 (0.3)	1.9	1.4 (0.5)
06	6.3	6.3 (0.0)	3.1	3.1 (0.0)	2.1	1.5 (0.6)
07	7.1	7.1 (0.0)	4.0	3.9 (0.1)	1.9	1.4 (0.5)
08	5.0	4.8 (0.2)	2.7	2.3 (0.4)	2.0	1.5 (0.5)
09	3.8	3.4 (0.4)	2.2	1.5 (0.6)	1.7	1.0 (0.7)
10	3.1	2.3 (0.8)	2.0	1.4 (0.6)	1.7	1.1 (0.6)
11	5.7	5.7 (0.0)	3.4	3.2 (0.2)	2.5	2.2 (0.3)
12	3.3	2.6 (0.7)	2.1	1.1 (1.0)	1.6	0.9 (0.7)
13	5.4	5.2 (0.2)	3.2	2.9 (0.3)	1.9	1.6 (0.5)
14	4.2	4.1 (0.1)	2.1	1.5 (0.6)	1.8	1.0 (0.8)
15	10.7	10.4 (0.3)	4.1	3.6 (0.5)	2.0	1.1 (0.9)
16	2.8	2.4 (0.4)	2.0	1.5 (0.5)	1.5	0.6 (0.9)
Average	**4.9**	**4.5 (0.4)**	**2.7**	**2.2 (0.5)**	**1.8**	**1.2 (0.6)**

Table 1: Noise estimation performance of the proposed method on the 16 videos. "w/o fusion": the average MREs (unit: %) of noise curves over all the frame pairs. "with fusion": the MREs of the fused noise curves by the median technique. The reduced errors by the fusion technique are indicated beside the MREs of fused curves.

As can be seen, the errors are larger for weak noise and smaller for strong noise, because the errors are mainly caused by the signal residuals in the block differences which only depend on the true signals but not on the noise level. As the noise level increases, the absolute estimation errors also increase because of the worse block matching, but increase more slowly than the noise variance. Thus the relative errors are smaller in the case of stronger noise. Also, the median fusion technique slightly boosts the accuracy on the basis of the estimation from frame pairs.

Figure 4 shows a good example of noise estimation on two frames from sequence 01, while the estimation errors are relatively high for some video sequences such as sequence 07 (Figure 5) and 15 (Figure 6) which have more textures than other sequences and have more high-frequency components of true signals. Even though our algorithm can suppress the textured signals by inter-block subtraction, in the case of imperfect block matching the signal residuals are represented in high frequencies and can be considered as noise by our method, resulting in the over-estimated noise level. In practice, it is recommended to use a smaller quantile of block pairs to filter out high-frequency residuals. Indeed, using a small quantile can significantly reduce the estimation errors of these two sequences for weak and medium noises, as is shown in Table 2. From Figure 7 we can see that the over-estimation of noise is alleviated for two frames in sequence 15 with smaller quantile. Note that a large quantile is recommended in the case of strong noise, which will be detailed in Section 6.3.

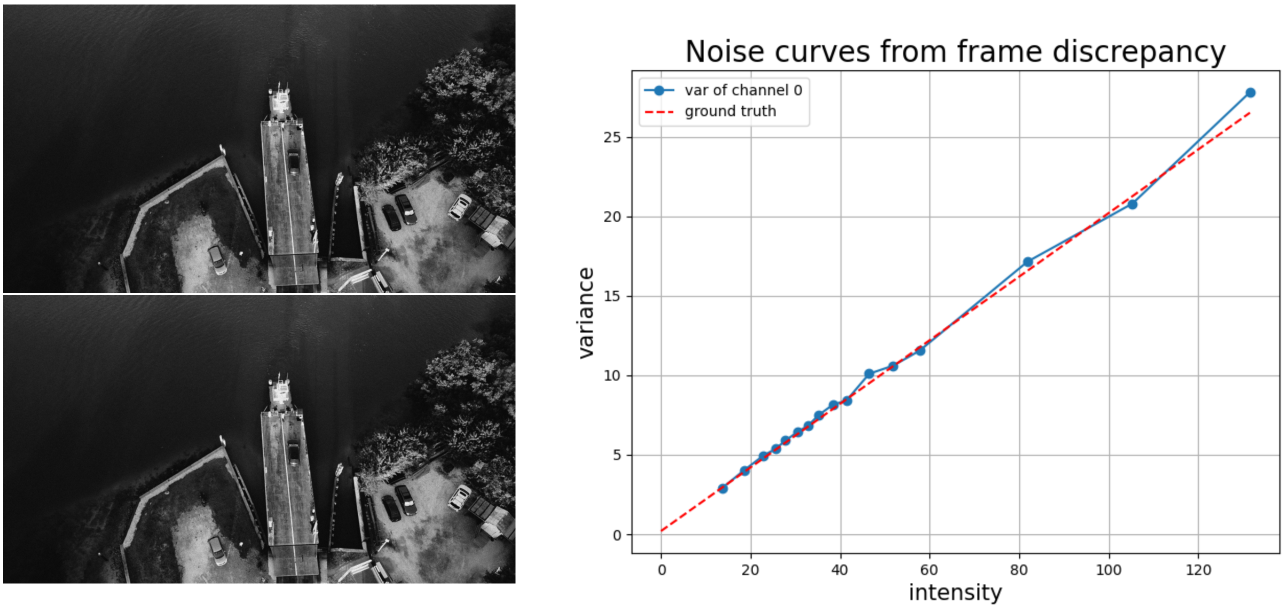


Figure 4: Estimated noise curve from two successive frames of sequence 01. 1st column: input frame t and frame $t+1$; 2nd column: estimated noise curve (in blue) and ground truth noise noise curve (in red). The simulated noise model is $g(I) = 0.2 \times I + 0.2$.

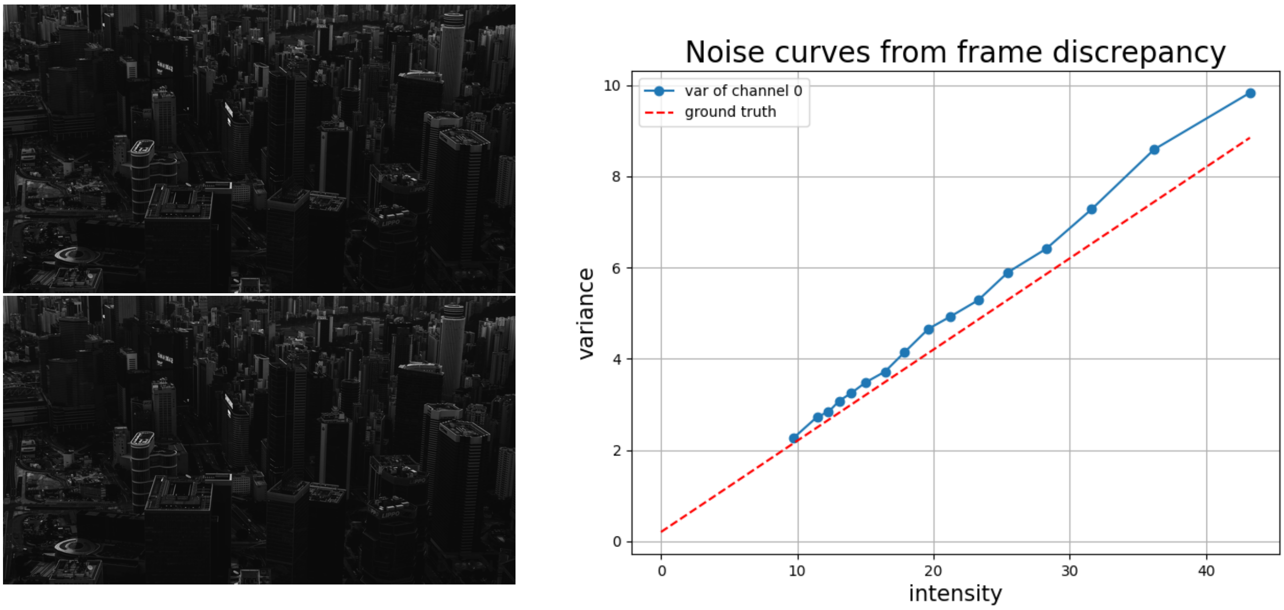


Figure 5: Estimated noise curve from two successive frames of sequence 07. 1st column: input frame t and frame $t+1$; 2nd column: estimated noise curve (in blue) and ground truth noise noise curve (in red). The simulated noise model is $g(I) = 0.2 \times I + 0.2$.

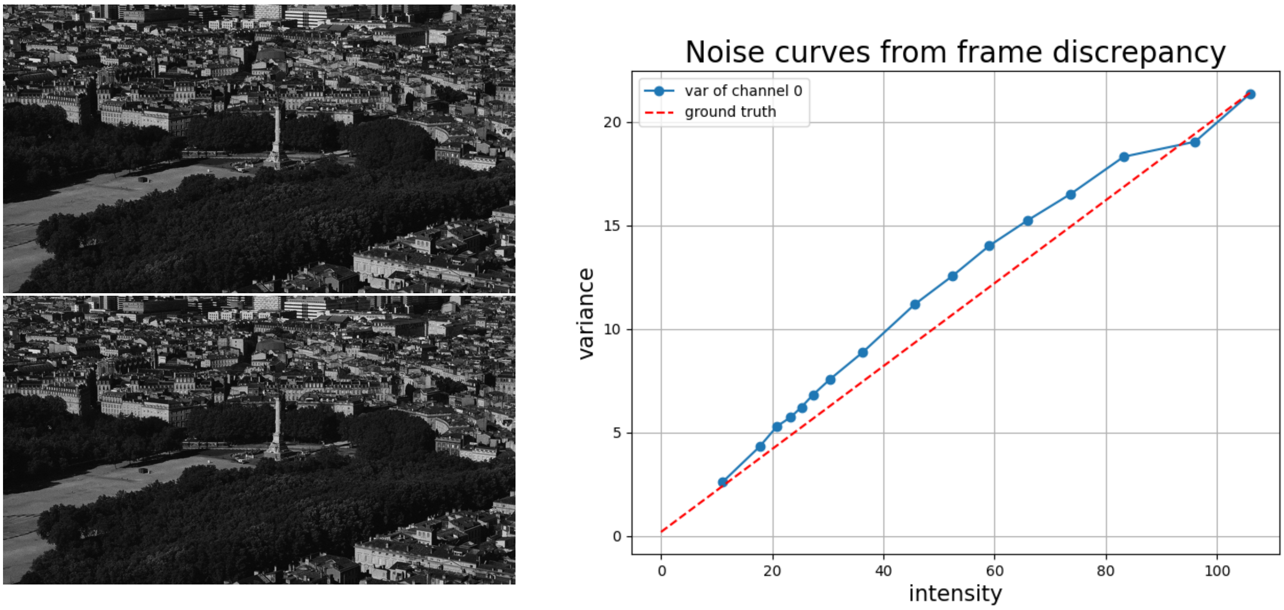


Figure 6: Estimated noise curve from two successive frames of sequence 15. 1st column: input frame t and frame $t+1$; 2nd column: estimated noise curve (in blue) and ground truth noise noise curve (in red). The simulated noise model is $g(I) = 0.2 \times I + 0.2$.

Sequence	q	$\alpha = 0.2$ $\beta = 0.2$	$\alpha = 0.8$ $\beta = 0.8$	$\alpha = 3.2$ $\beta = 3.2$
07	1%	5.4	3.4	3.0
07	5%	7.1	4.0	1.9
15	1%	7.5	3.7	2.8
15	5%	10.4	4.1	2.0

Table 2: The MREs (unit: %) of sequence 07 and 15 under two different values of quantile q. Using a smaller quantile q boosts the estimation accuracy for textured video sequences such as 07 and 15 in the case of weak noise. When strong noise appears, a larger quantile is recommended.

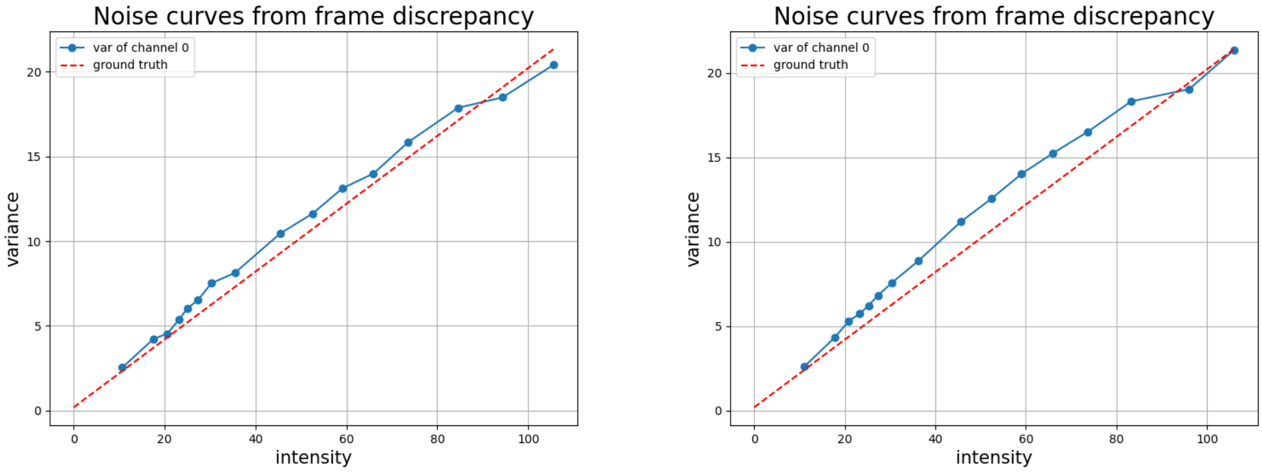


Figure 7: Estimated noise curves of two successive frames from sequence 15 at low noise level ($\alpha = \beta = 0.2$), using two different quantile values $q = 1\%$ (left) and $q = 5\%$ (right).

6.1 Impact of the Frequency Separator T

The frequency separator T, which classifies a DCT coefficient as low or high frequency, affects the performance of the proposed method. The method's estimation errors with the same block size $w = 20$ and different frequency separators T are shown in Table 3.

T	$\alpha = 0.2$ $\beta = 0.2$	$\alpha = 0.8$ $\beta = 0.8$	$\alpha = 3.2$ $\beta = 3.2$
17	5.7	3.0	**1.7**
19	5.2	2.8	1.7
21	4.9	2.7	1.8
22	4.7	**2.7**	1.9
24	4.6	2.7	2.1
26	**4.5**	2.8	2.3
28	4.6	3.1	2.7
30	4.9	3.5	3.2
32	5.7	4.2	4.0
34	7.1	5.6	5.2

Table 3: Mean relative errors (unit: %) of the proposed method with different frequency separators T and the same block size $w = 20$. Other parameters are the same as previous ones.

It is observed that the optimal value of T is an intermediate value. This is because a smaller value of T will pass fewer low frequencies for evaluating the amount of signal residual in a block, as a result the block pair selection is worse and there may be signal residual remaining in the high frequencies. On the other hand, a larger value of T will force more low frequencies for block pair selection which suppresses well the noise residual, but estimating noise variance from a limited number of high frequencies can result in higher error.

Moreover, the optimal value of the frequency separator T varies depending on the noise level. In general, a lower T is suitable for stronger noise. When the noise level is higher, the inherent estimation error of a noise variance is also larger given a fixed number of high-frequency coefficients as samples, and the error due to the signal residual is relatively smaller. A smaller value of T is then helpful to estimate noise variance from more high frequencies, reducing the inherent estimation error. In contrast, when the noise level is low, a larger value of T is preferable.

6.2 Impact of the Block Size w

The method's performance with different settings of block sizes w is also evaluated. Since the frequency separator T is related to the block size and affects the performance, only the best performances with the optimal value of T for each block size and each noise level are shown in Table 4. It can be observed that in general, a larger block size improves the noise estimation from frame pairs. Indeed, larger blocks contain enough low-frequency coefficients to evaluate and suppress signal residual, while there are also enough high-frequency coefficients to get an accurate estimate of noise variances.

w	$\alpha = 0.2$ $\beta = 0.2$	$\alpha = 0.8$ $\beta = 0.8$	$\alpha = 3.2$ $\beta = 3.2$
7	5.0	3.0	2.0
11	4.9	2.9	2.0
14	4.8	2.8	1.9
17	4.7	2.8	1.8
20	4.5	2.7	1.8
23	4.4	2.6	1.8
26	4.3	2.5	1.8
29	4.2	2.4	1.8
32	4.1	2.3	1.8
35	3.9	2.2	1.8
38	3.8	2.2	1.8
41	3.7	2.1	1.8
44	3.6	2.0	1.8
47	**3.6**	**2.0**	**1.8**

Table 4: Mean relative errors (unit: %) of the proposed method with different block sizes w. The optimal value of the frequency separator T is chosen for each block size and each noise level. Other parameters are the same as previous ones. As can be seen, the larger the block size, the lower the estimation errors.

One might think that smaller blocks are better to find easily block pairs with nearly identical signals, given that the video motion is not globally uniform and a finer granularity of chunk division is better for motion prediction. With bigger blocks, we risk signal residuals that affect the noise estimation. Even so, bigger blocks are still preferable. Indeed, most of the content of a video is background content that is still or uniformly-moving. Even with large blocks the method can find enough block pairs that contain little signal residuals. In short, the disadvantage of introducing signal residuals is relatively small compared to the advantage of having more frequency coefficients for block selection and noise estimation.

6.3 Impact of the Quantile of Block Pairs q

The estimation performances of the proposed method with different quantiles q of block pairs are shown in Table 5. The block size $w = 20$, frequency separator $T = 21$, and other parameters are the same as previous ones.

q	$\alpha = 0.2$ $\beta = 0.2$	$\alpha = 0.8$ $\beta = 0.8$	$\alpha = 3.2$ $\beta = 3.2$
0.5%	4.4	3.5	3.7
1%	**4.3**	3.0	2.9
2.5%	4.5	2.7	2.1
5%	4.9	**2.7**	1.8
7.5%	5.2	2.7	1.7
10%	5.4	2.8	**1.6**

Table 5: Mean relative errors (unit: %) of the proposed method with different quantiles of block pairs. Block size w is 20, frequency separator T is 21, and other parameters are the same as previous ones.

We can see the optimal quantile q is a compromising value: either too big or too small values degrade the performance. A small quantile of block pairs provides insufficient samples to estimate an accurate value of noise variance, and causes estimation uncertainty; on the other hand, keeping too many blocks will lead to retaining some block pairs whose differences still contain signal residuals, and leads to over-estimation of noise variance.

Furthermore, the optimal quantile of block pairs is higher for videos with a high noise level. As the noise level increases, the estimation error of the noise variance also increases, thus more samples are needed to reduce the estimation error. By increasing the quantile q in a certain range, the increase of error caused by the introduced signal residuals is smaller than the benefit of reducing the estimation error with the extra samples.

6.4 Comparison with the Ponomarenko's Method

Since our proposed method is derived from the Ponomarenko method [5, 4, 18] on single image, a comparison between them was conducted with different block sizes w. The other parameters were: threshold separator $T = w + 1$, quantile $q = 1\%$, search window parameter $s = 3$ and thickness of the surrounding ring for block matching $th = 3$. The results are shown in Table 6.

method	w	$\alpha = 0.2$ $\beta = 0.2$	$\alpha = 0.8$ $\beta = 0.8$	$\alpha = 3.2$ $\beta = 3.2$
Ponomarenko	5	5.3	3.2	3.2
	7	4.4	3.1	3.1
	14	7.1	3.6	3.8
	21	7.9	4.9	6.1
	32	9.3	6.1	8.5
our proposed	5	7.3	6	5.9
	7	5.8	4.3	4.1
	14	4.8	3.3	2.9
	21	4.6	3.4	3.5
	32	**4.1**	**2.3**	**1.8**

Table 6: Mean relative errors (unit: %) of the proposed method and the Ponomarenko method. Our method's optimal result (red) with $w = 32$ is better than that of the Ponomarenko method (blue) with $w = 7$.

It can be observed that the Ponomarenko method estimating noise curves on single images led to better estimation with small blocks, and performed worse as the block size increases, whereas our method performed better with larger blocks. The explanation is that the Ponomarenko method selects flat blocks for estimating noise, whose intensities are uniform on each block thus the noises are block-wise homoscedastic. The estimated noise variance on each block is perfectly related to its mean intensity. As for our method estimating noise from similar block pairs, the difference between two perfectly matched blocks contains only the noise, twice as much as the noise of each block, but the noise can be heteroscedastic if the intensities of each block are not spatially uniform, e.g. a highly textured block. The estimated variance of the heteroscedastic noise does not correspond perfectly to the mean intensity of these two textured blocks, leading to an additional estimation error. Nevertheless, using blocks of larger size can alleviate this error. Indeed, the optimal performance of our method with $w = 32$ is better than the best result of the Ponomarenko method with $w = 7$.

6.5 Computational Cost

The most time-consuming parts of the proposed method are the block matching (line 1 of Algorithm 7), DCT-II on blocks (line 8 and 10 of Algorithm 7) and the sorting of low-frequency energies (line 6 of Algorithm 3), while the computational times of the other steps are relatively negligible. Using the same notations of Algorithm 7, the time complexity is $\mathcal{O}(HWs^2)$ for block matching, $\mathcal{O}(HWw^2log(w))$ for DCT-II on all the overlapping blocks and $\mathcal{O}(HWlog(HW))$ for sorting low-frequency energies, thus the overall time complexity is $\mathcal{O}\left(HW\left(s^2 + w^2log(w) + log(HW)\right)\right)$. Note that the implementation of DCT-II[2] is accelerated by multi-threading. The total computing time on a machine equipped with 8 CPUs at 3.8 GHz on two 540 × 940 single-channel frames under the default parameter setting is 7.3 seconds.

7 Limitations

7.1 Parameter Tuning

As is mentioned in the experimental section, the method's performance is related to the parameter setting. In general, the optimal parameters of the frequency separator T, the block size w and the quantile of block pairs q depend on the noise level of the video, thus the best practice of parameter tuning will be first applying the proposed method with the default parameters to have a rough estimation of the noise level for choosing the optimal parameters, and then applying the method again with the optimal parameters.

The optimal parameters not only depend on the noise level of a video as mentioned above, but also depend on the content of a video. If a video has changing contents, we should rather take a smaller value of the quantile q so that the selected block pairs are less likely to have signal residuals. The strategy of parameter selection according to the signal amount is still an empirical process.

7.2 Over-estimation of Noise Curves

The main cause of the over-estimation of noise curves is the presence of remaining residuals between mismatched blocks. First, the choice of using only low frequencies for block matching makes the algorithm incapable of matching blocks with high-frequency signals (e.g. fine stripe textures), which results in high-frequency residuals in the block differences. Besides, when the video is not well sampled, the aliased signal presented in both low and high frequencies also leads to bad block

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.fft.dct.html

matching. Even in ideal conditions where the signal is well sampled and is present only in low frequencies, the changing contents in a video with moving scenes can cause mismatched block pairs. Slow motion (e.g. drone videos used above) still makes it possible to result in well matched blocks with same signals, while fast motion of a video (e.g. sport videos) would greatly change the signal from frame to frame and lead to unavailable block pairs for estimating noise. In addition, block matching is even harder when noise is present [23]. All these factors contribute to block mismatching, which leads to undesirable signal residuals and consequently the over-estimation of noise curves.

7.3 Under-estimation of Noise Curves

Besides the over-estimation, sometimes noise curves can have smaller estimated values than the ground truth. This is because selecting block pairs whose inter-block energies at low frequencies are low will also reduce the high frequencies for *non-uniform* blocks pairs.

Let us suppose all block pairs are perfectly matched, thus there is no signal residual remaining in the difference of two matched blocks. However, when the signals of two blocks present a gradation and are not spatially uniform (e.g. two textured blocks), the noise for each pixel entry has a different variance depending on the signal intensity of the pixel entry. The noise residual between two blocks is therefore heteroscedastic Gaussian noise. After DCT transformation on the heteroscedastic Gaussian noise, the high-frequency coefficients and the low-frequency coefficients are correlated. From our observations when processing the estimation on textured block pairs, the selection of block pairs with low energies of difference in low frequencies will also diminish the norms of the high frequencies, which leads to an under-estimation of noise from the high frequencies.

7.4 Unhandled Types of Noise

Temporally correlated noise cannot be correctly handled by our proposed method relying on the assumption that the noises in two successive frames are independent. One typical case is the compressed videos [22], where the inter-frame residuals are compressed and the noise is thus temporally correlated. Another special case is the fixed-pattern noise (FPN) which does not change from frame to frame at fixed positions. The FPN of matched blocks at the same positions can cancel out in their differences, let alone the fact that the FPN can be spatially correlated in some cameras such as CMOS cameras [9] and cannot be correctly estimated by our method.
Spatially correlated noise is colored noise and cannot be estimated by the proposed method. However, at a high scale the noise tends to white and is estimable again. This will be addressed by an extension of the proposed method to multiple scales, detailed in Section 8.2.

8 Two Simple Extensions of the Proposed Method

8.1 Subpixel Block Matching for Further Signal Suppression

The proposed method operates block matching at pixel scale, but the real motion of an object across two successive frames is subpixel. This results in a subpixel matching error and leaves the signal residuals in the block difference.

However, we found that subpixel block matching with image interpolation by Fourier Transform does not suppress the signal residuals. As was done for block matching at pixel scale, after subpixel interpolation we only used the surrounding pixels of a block to find the matched block, so that the noise in the central block was not affected by the matching. Although this approach can further reduce the matching error of the surrounding matching areas, it also creates ringing effects in highly contrasted areas. The ringing effects from the surrounding compared area of a block then propagate to the block itself. We therefore observed that the reduction of difference in the surrounding compared area did not result in the reduction of difference in the central block. What is worse, the periodic ringing effects are in high frequencies, which pollutes the high frequencies of the central block and finally degrades the noise estimation from high-frequency coefficients.

To illustrate the ringing effects and the noise estimation errors introduced by sub-pixel matching, we performed an additional experiment on two consecutive *noiseless* images I_1 (Figure 8) and I_2 between which the ground truth shift is known. In practice, a high resolution image I was blurred by Gaussian kernel ($\sigma = 3.1$) for anti-aliasing, and downscaled by factor 4 respectively with offsets $(0, 0)$ and $(1, 1)$ to obtain I_1 and I_2. Thus the ground truth shift between I_1 and I_2 was $(0.25, 0.25)$ pixel.

Note the shifting operator as $T : (I, dx, dy) \to I'$ such that $I(x, y) = I'(x + dx, y + dy)$ for all $(x, y) \in \Omega$ where Ω is the image domain. The shifting operator is realized by interpolation with FFT-shifting [21][3]. With the ground truth shift $(0.25, 0.25)$ px, the best shift at pixel scale should be $(0, 0)$ px. The difference between two images with the best matching at pixel scale $\Delta I = T(I_1, 0, 0) - I_2 = I_1 - I_2$ and with the best matching at subpixel scale $\Delta I^{sub} = T(I_1, 0.25, 0.25) - I_2$ are shown in Figure 8.

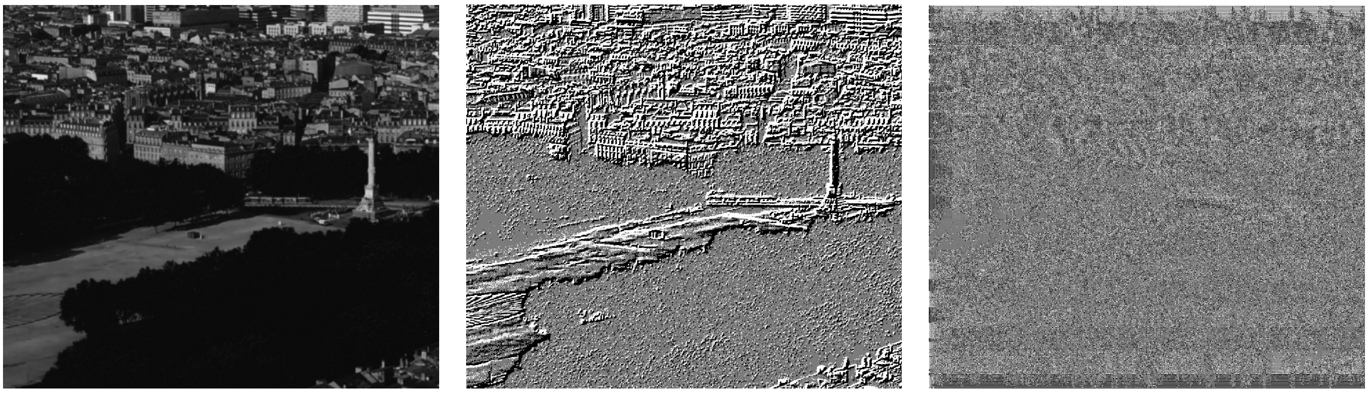


Figure 8: Left: noiseless image I_1, middle: image difference after pixel-scale matching: $\Delta I = T(I_1, 0, 0) - I_2 = I_1 - I_2$, right: image difference after subpixel matching $\Delta I^{sub} = T(I_1, 0.25, 0.25) - I_2$. Subpixel matching cancels out the main difference but leaves ringing patterns present in high frequencies.

As can be seen, the subpixel matching did cancel out most of the difference between two images, but left also ringing patterns in the image difference which correspond to the high frequencies.

To further assess the impact of ringing effects on the proposed noise estimation method, we computed the high-frequency energies of the differences between matched block pairs, and evaluated the change of the high-frequency energies with respect to the shift of matching. That is, we shifted I_1 by δs both along x axis and y axis to get $I_1^{\delta s} = T(I_1, \delta s, \delta s)$ with $\delta s \in [0, 0.4]$, divided the image difference $I_1^{\delta s} - I_2$ into overlapping 8×8 blocks $\mathcal{B} = \{\mathbf{B}_k\}$, and obtained the DCT transformed blocks $\tilde{D} = \{\mathbf{D}_k = \mathrm{DCT}(\mathbf{B}_k)\}$. With the similar definition of low-frequency energy in Section 3.4, the average low-frequency energy over all the blocks can be computed by

$$\bar{V}^L = \frac{1}{|\tilde{D}|} \sum_{\mathbf{D}_k \in \tilde{D}} \frac{\sum_{i=1}^{w} \sum_{j=1}^{w} \left[\mathbf{D}_k(i, j)\right]^2 \mathbb{1}_{\{\|(i,j)\|_1 \le T\}}}{\sum_{i=1}^{w} \sum_{j=1}^{w} \mathbb{1}_{\{\|(i,j)\|_1 \le T\}}}, \tag{17}$$

where the frequency threshold $T = 9$. Similarly, the average high-frequency energy over the blocks can be computed by

$$\bar{V}^H = \frac{1}{|\tilde{D}|} \sum_{\mathbf{D}_k \in \tilde{D}} \frac{\sum_{i=1}^{w} \sum_{j=1}^{w} \left[\mathbf{D}_k(i, j)\right]^2 \mathbb{1}_{\{\|(i,j)\|_1 > T\}}}{\sum_{i=1}^{w} \sum_{j=1}^{w} \mathbb{1}_{\{\|(i,j)\|_1 > T\}}}. \tag{18}$$

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.fourier_shift.html

The values of $\bar{V}^L$ and $\bar{V}^H$ for different shift values δs are shown in Table 7.

shift along x-axis and y-axis δs	0	0.05	0.10	0.15	0.20	0.25(δs^{GT})	0.30	0.35	0.40
avg. low-freq. energy $\bar{V}^L$	20.3	13.1	7.53	3.57	1.24	**0.552**	1.50	4.07	8.26
avg. high-freq. energy $\bar{V}^H$	0.293	0.249	0.214	0.191	0.179	**0.179**	0.191	0.214	0.247

Table 7: Average low-frequency energy and high-frequency energy over all the matched block differences between I_1 and I_2 for different shift values of δs. Note that the best matching corresponds to the shift value $\delta s^{GT} = 0.25$.

We can see that both the average low-frequency energy and average high-frequency energy were minimized at the ground truth shift $(0.25, 0.25)$, following that subpixel matching did reduce the inter-block differences both in low frequencies and in high frequencies. However, only a small quantile of block pairs whose low-frequency energies are the smallest are used for noise estimation, thus we have to look at the frequency energies of the small quantile of block differences. Note the set of the q-quantile of DCT blocks as $\tilde{D}_q \subset \tilde{D}$ whose low-frequency energies are the smallest. The average low-frequency energy $\bar{V}_q^L$ are computed similarly

$$\bar{V}_q^L = \frac{1}{|\tilde{D}_q|} \sum_{\mathbf{D}_k \in \tilde{D}_q} \frac{\sum_{i=1}^{w} \sum_{j=1}^{w} [\mathbf{D}_k(i,j)]^2 \mathbb{1}_{\{\|(i,j)\|_1 \leq T\}}}{\sum_{i=1}^{w} \sum_{j=1}^{w} \mathbb{1}_{\{\|(i,j)\|_1 \leq T\}}} \tag{19}$$

$$\bar{V}_q^H = \frac{1}{|\tilde{D}_q|} \sum_{\mathbf{D}_k \in \tilde{D}_q} \frac{\sum_{i=1}^{w} \sum_{j=1}^{w} [\mathbf{D}_k(i,j)]^2 \mathbb{1}_{\{\|(i,j)\|_1 > T\}}}{\sum_{i=1}^{w} \sum_{j=1}^{w} \mathbb{1}_{\{\|(i,j)\|_1 > T\}}}. \tag{20}$$

And the values of $\bar{V}_q^L$ and $\bar{V}_q^H$ for different shift values δs are shown in Table 8. As can be seen, the average low-frequency energy is minimized at δs close to its ground truth value. However, the average high-frequency energy is minimized at $\delta s = 0$, indicating that the shift at pixel scale $(0, 0)$ is better than at subpixel scale $(0.25, 0.25)$ here in order to suppress the high-frequency components of the inter-block differences. This is because with matching at pixel scale after selecting a small quantile of block differences the high-frequency residuals are already small enough. For these block differences if we further suppress the high-frequency residuals, the reduction of the original high-frequency residuals will be less significant thant the introduction of additional ringing artifacts mainly in the form of high frequencies. Hence, the high-frequency residuals of the small quantile of block differences are increased, which is contrary to our aim of suppressing high-frequency residuals.

shift along x-axis and y-axis δs	0	0.05	0.10	0.15	0.20	0.25(δs^{GT})	0.30	0.35	0.40
avg. low-freq. energy $\bar{V}_q^L$	0.092	0.082	0.076	0.072	**0.072**	0.074	0.081	0.091	0.103
avg. high-freq. energy $\bar{V}_q^H$	**0.077**	0.077	0.080	0.084	0.090	0.096	0.103	0.109	0.114

Table 8: Average low-frequency energy and high-frequency energy over a small quantile $q = 5\%$ of matched block differences between I_1 and I_2 whose low-frequency energies are the smallest, for different shift values of δs. Note that the best matching corresponds to the shift value $\delta s^{GT} = 0.25$.

Finally, we compared the noise estimation performance for pixel-scale matching and subpixel matching at precisions 0.5 px and 0.25 px. The other parameters were the same: block size $w = 8$, frequency threshold $T = 9$, quantile $q = 5\%$ and search window range $s = 5$. The comparison results are shown in Table 9. For low and medium noise levels, block matching at pixel scale gives better noise estimation results. For high noise level, subpixel matching results in better precision of noise estimation. This is because the block matching in images with stronger noise is more difficult and the correctly matched pairs are fewer. As can be seen in Table 7 and 8, only a small quantile of block pairs have very weak high-frequency residuals with pixel-scale matching, while the rest of block pairs have strong high-frequency residuals. When mismatching is present, some of the block pairs belonging to the small quantile in noiseless images that should have been matched well are now missing and are replaced with other block pairs having stronger high-frequency residuals, thus the high-frequency residuals in the selected quantile of block pairs become much higher. When the matching is at subpixel scale, the block pairs have globally weak high-frequency residuals. Even though some ideal block pairs that should have been matched are missing, the selected quantile including other pairs that are not perfectly matched in noiseless images still have relatively weak residuals. It is therefore better to enable subpixel matching for very noisy videos during the noise estimation.

matching	$\alpha = 0.2$	$\alpha = 0.8$	$\alpha = 3.2$
precision	$\beta = 0.2$	$\beta = 0.8$	$\alpha = 3.2$
1 px	5.0	3.0	2.4
$\frac{1}{4}$ px	6.0	3.8	2.1

Table 9: Mean relative errors (unit: %) of the proposed method with block matching at pixel scale (1 px) and at subpixel precision of $\frac{1}{4}$ px. The block size $w = 7$, the threshold separator $T = 8$ and the quantile of block pairs $q = 0.05$.

Besides the subpixel matching by Fourier Transform, other different subpixel matching methods such as spline interpolation and Lanczos interpolation could be explored in the future. However, no matter how precise the subpixel matching is, interpolation is always required to suppress the inter-block residual. For the noise estimation a preferable interpolator should be capable of preserving the property of the noise, and the interpolator by Fourier Transform is that preferable interpolator. Even though it also creates ringing effects and pollutes the noise in the high frequencies, it has the merit of keeping the white noise white, whereas other interpolators change the spectrum of the noise.

8.2 Multiscale Estimation for Colored Noise

Even though the noise of a raw video is white noise, the noise of a processed video has spatial correlation due to the operations in the video processing pipeline, such as demosaicing, white balance, color correction, etc. The demosaicing step operates pixel interpolation to reconstruct the missing colors of the pixels, which adds correlations between demosaiced values and results in different effects on noises of different frequencies. The compression step converts blocks into DCT coefficients and applies quantization to the DCT coefficients with different quantization factor at each coefficient entry, which has stronger reduction on the noise in high frequencies than in low frequencies. The pattern of correlated pixels in a processed image after JPEG compression is shown in Figure 9, where the pixels of a uniform pattern form small spots of about 3 pixels in diameter rather than forming pixel-size spots due to the interpolation of demosaicing, and where the pixels in a 8×8 block are similar due to the JPEG compression. The spatial correlation results in the colored noise whose spectrum energies are no longer homogeneous and makes our method fail. However, the spatial correlation of pixels usually exists within a small region, and two pixels far away from each other are still independent, so are their noises. One can expect the decrease of the spatial correlation by downscaling the images. Figure 10 shows the energy distribution of DCT coefficients of the noise of a flat image, at different steps of the processing pipeline and at different down-sampling scales. It can be seen that at a higher scale, the noise of the processed image becomes "whiter".

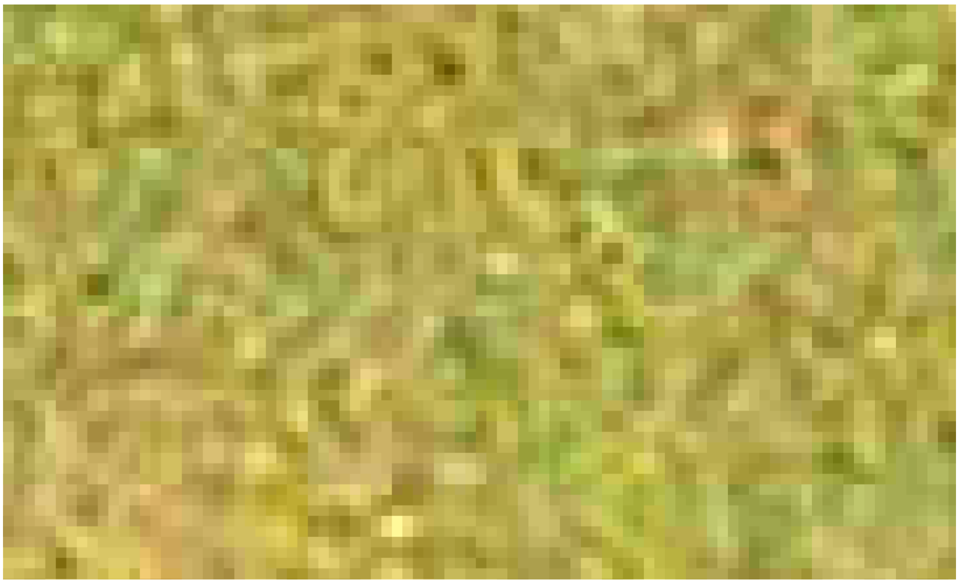


Figure 9: Noisy pattern of a flat zone in an image compressed by JPEG. Pixels of the same colors are usually in small groups rather than appearing individually due to the interpolation of demosaicing, and the pixels in a 8×8 block are similar due to the JPEG compression processed on 8×8 blocks.

This leads us to develop a simple extension of the method to roughly estimate the noise at multiple scales, as is done in [4]. At a sufficiently high scale, the noise of the images becomes approximately white and is estimable again by the proposed method. At a lower scale or the original scale, even though the noise is colored, we can make a rough assumption that the noise variances over the different frequencies are piece-wise constant, and estimate the noise from a small set of selected high frequencies to get a rough estimation of noise in the high-frequency band of the scale. The rough measurement of the noise levels at different scales can be seen as a characteristic of the colored noise.

When performing the noise estimation at a higher scale, the block matching is still performed at the original scale because matching at a higher scale will cause larger errors and leave higher signal residuals between matched blocks. The obtained matched blocks are then downscaled with the average filter, which ensures that the downscaled blocks are not aliased if the original images are well-sampled. A block $U \in \mathbb{R}^{fw \times fw}$ is downscaled by an integer factor f to obtain $U' \in \mathbb{R}^{w \times w}$ as follows

$$U'(i,j) = \frac{1}{f^2} \sum_{k=1}^{f} \sum_{l=1}^{f} U\left((i-1) \times f + k, (j-1) \times f + l\right). \tag{21}$$

With the downscaled block pairs, the rest of the steps operated on the block pairs are the same as in the original method for raw videos. At scale s, we downscale the blocks by factor $f = 2^s$, thus the block matching is processed with block size $2^s \times w$ in order to have the same size of blocks w during noise estimation at different scales. In practice, we process the noise estimations at 4 different scales. The detailed algorithm is described in Algorithm 12 with the flag of raw input images as *False*.

Besides, the noise estimation is less stable at higher scale because the signal residual in the high-frequency band at a higher scale is more significant than that in the same high-frequency band at a lower scale. To alleviate the impact of signal residual in high-frequency band at a high scale, we empirically multiply the quantile q by 0.7 at each downscaling such that fewer block pairs will be used for estimation and less signal residual in the high-frequency band will affect the noise estimation.

With this multi-scale estimation approach, at a specific intensity one can expect the reduction of

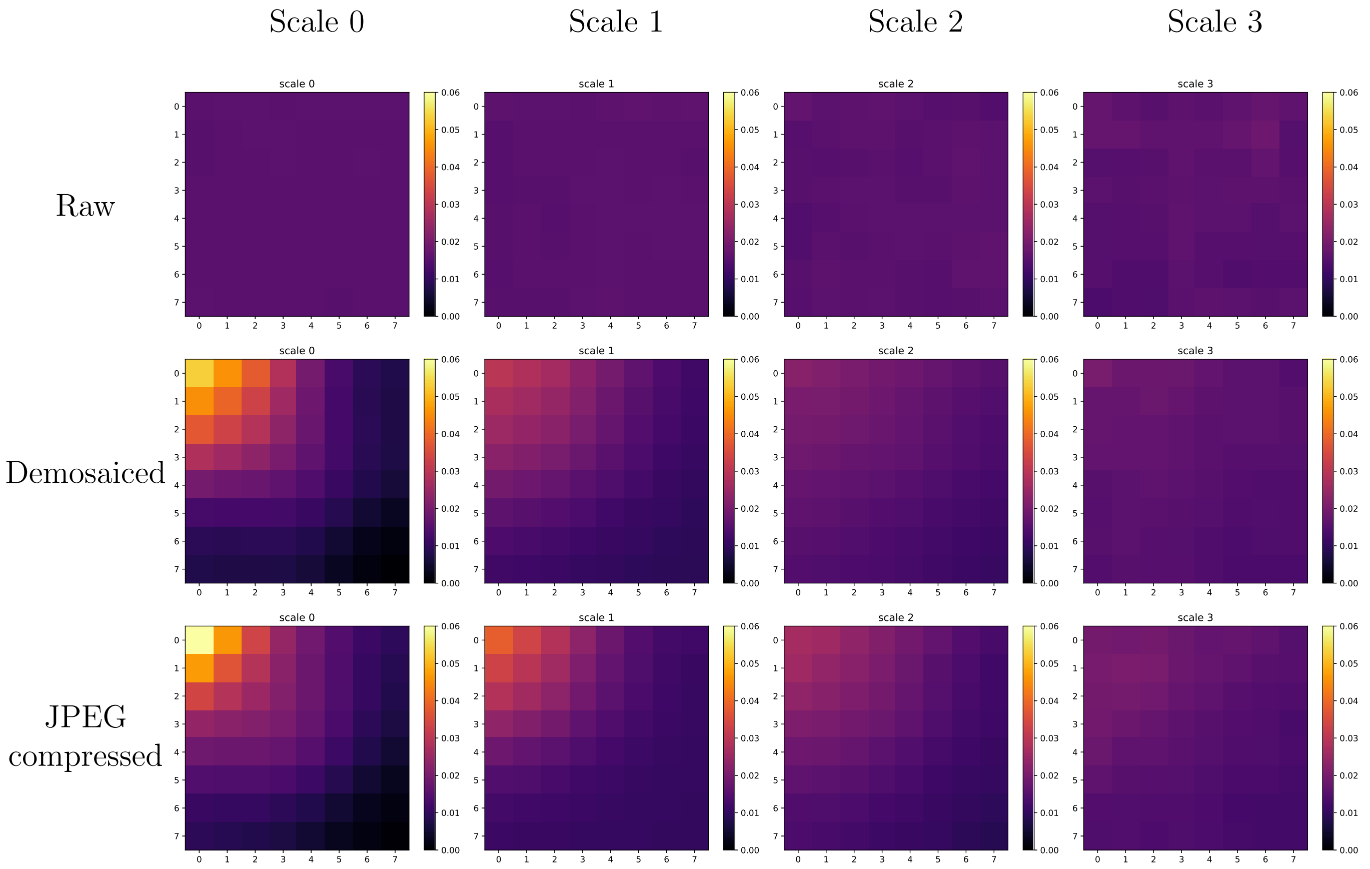
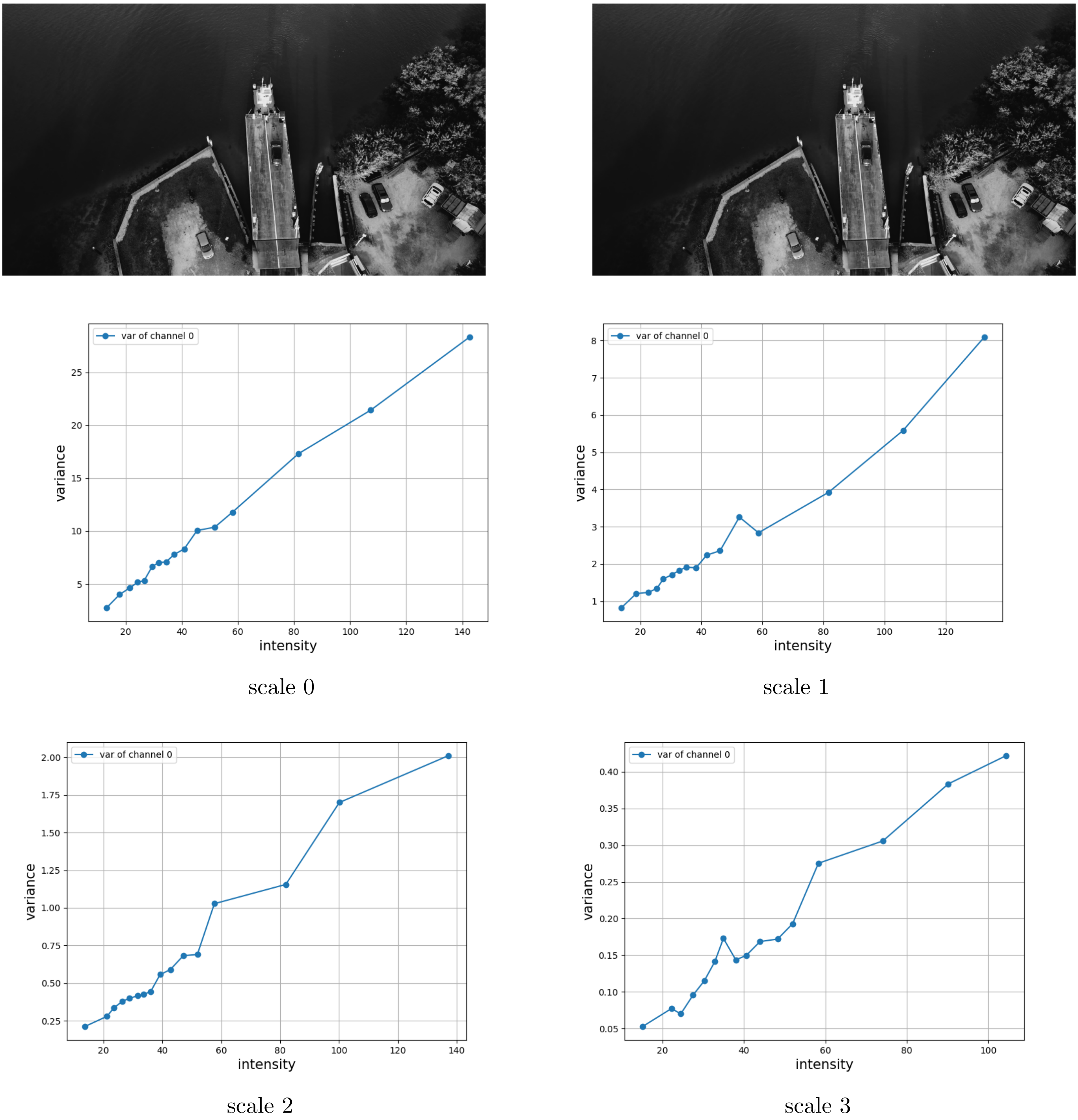


Figure 10: Distribution heatmap of noise energies on 8×8 DCT coefficients of a flat image with white noise. Each value of the 8×8 grid shows the ratio of the energy of the related DCT coefficient in the total noise energy. From top to bottom: noise energy distributions of the raw frames, the demosaiced frames and the frames compressed by JPEG. From left to right: noise energy disributions at different scales. An image is downscaled by factor 2^s at scale s with the average filter. Only the blue channel is shown here while the green and red channels give similar results. As the image is downscaled at a higher scale, the noise becomes "whiter".

the noise variance by a factor 4 at each increment of scale if the noise is white noise. Otherwise, by comparing the noise variance at scale s divided by 4 and the noise variance at scale $s+1$, one can assess the noise variation when switching from a high-frequency band to its adjacent low-frequency band. For instance, the noise variance at a certain intensity (e.g. intensity of 100) at a certain scale in Figure 11 is approximately 4 times as large as the noise variance at its adjacent higher scale, which means the noise is approximately white for each intensity. For two processed images taken in "burst mode" (see Figure 12), the noise variance at scale s (e.g. $s = 1$) is larger than one fourth of the noise variance at scale $s - 1$ at a specific intensity (e.g. intensity of 100 of green channel), meaning that the noise energy is larger in low frequencies than in high frequencies.

Note that this simple variant is still based on the assumption that the noises of two input images are independent, thus it does not fit the cases where the noises in two frames are dependent. For instance, two consecutive frames of a compressed video whose inter-frame residual is reduced by its video encoder (e.g. H.264 video compression [22]) have highly dependent noises. The images with temporally independent noises such as the images taken in "burst mode" are suitable input.



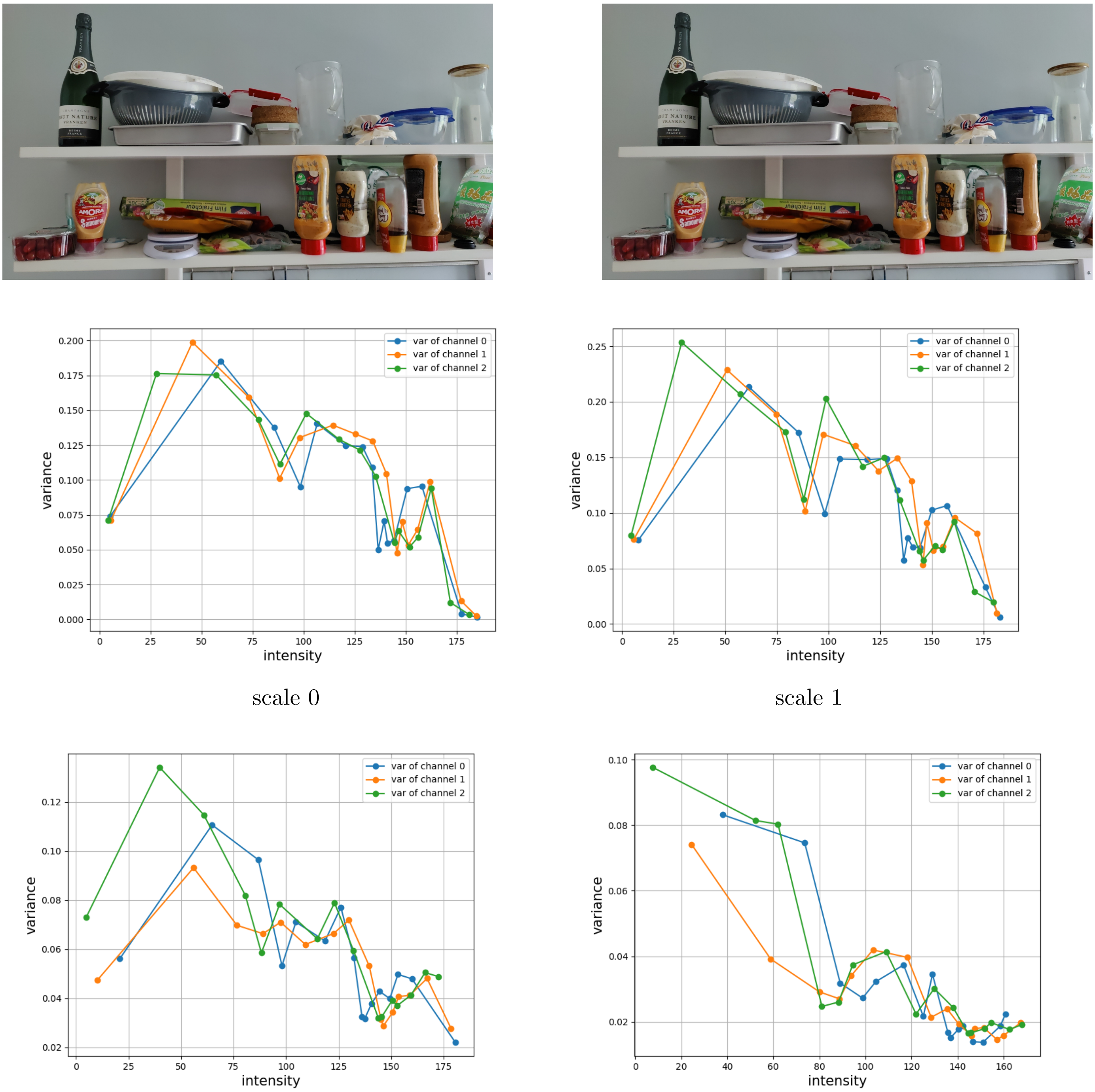
scale 0

scale 1

scale 2

scale 3

Figure 11: Estimated noise curves at different scales from two successive raw single-channel images. The noise variances at scale s are approximately 4 times as large as the ones at scale $s + 1$, indicating that the noise in the two images is approximately white for each specific intensity.



scale 0

scale 1

scale 2

scale 3

Figure 12: Estimated noise curves at different scales from two successive processed images taken in the "burst mode". If the noise were white, normally the noise variances at scale s would be 4 times as large as the noise variances at scale $s+1$, while in this example the noise variances at scale s are smaller than 4 times the ones at scale $s+1$, indicating that the noise in the two images is colored with higher noise energies at lower frequencies.

8.3 Implementation of the Two Variants

The two variants of the proposed method are implemented into a single program, with the subpixel block matching and the multiscale estimation as options. The image upsampling by FFT used for subpixel block matching is implemented in Algorithm 9.

Algorithm 9: Upsample an image by FFT zero padding (**upsample**)

```
Input img: image of size $H \times W$
Input $f$: upsampling factor, in integer
Output img_ups: upsampled image of size $fH \times fW$

$F \leftarrow$ fft(img)                                                # 2D FFT
$F \leftarrow$ fftshift($F$)  # shift the zero-frequency component to the center of the spectrum
$F_{\text{pad}} \leftarrow$ a zero matrix of size $fH \times fW$
for $(i, j) \in \{1, \dots, H\} \times \{1, \dots, W\}$ do
    $F_{\text{pad}}(\frac{f-1}{2} \times H + i, \frac{f-1}{2} \times W + j) \leftarrow F(i, j)$
img_ups $\leftarrow \Re$(ifft($F_{\text{pad}}$))          # inverse 2D FFT, only take the real part
return img_ups
```

The function processing the subpixel block matching is implemented in Algorithm 11.

Algorithm 10: Downscale a block by a given factor (**downscale**)

```
Input $B$: a block of size $w \times w$, $w$ is divisible by $f$
Input $f$: downscaling factor, in integer
Output $B'$: downscaled block

$B' \leftarrow$ a zero matrix of size $\frac{w}{f} \times \frac{w}{f}$
for $(i, j) \in \{1, \dots, \frac{w}{f}\} \times \{1, \dots, \frac{w}{f}\}$ do
    $B'(i, j) \leftarrow \frac{1}{f^2} \sum_{p=1}^{f} \sum_{q=1}^{f} B((i-1) \times f + p, (j-1) \times f + q)$
return $B'$
```

The downscaling of a block is simply an average filter applied to all the non-overlapping sub-blocks of size of the downscaling factor, as is shown in Algorithm 10. Note that the size of the input block must be divisible by the downscaling factor so that no pixels at the border will be left without being able to form a sub-block.

The main function of the extended method including subpixel block matching and multiscale noise estimation is described in Algorithm 12. It is worth noting that the subpixel matching (Algorithm 11) is computationally costly at a higher precision, whose time complexity is $O(2^s \times N)$ with the upsampling scale of s and N pre-matched block pairs. Therefore, it is better to pre-select the input block pairs before subpixel matching. Indeed only the best matched block pairs after subpixel matching whose low-frequency energies are the smallest will be used for noise estimation, empirically the low-frequency energies of these best pairs are also very small just after pixel-scale matching and before subpixel matching. With this consideration we can pre-filter the block pairs by selecting $3q$-quantile of the pairs whose low-frequency energies are the smallest before subpixel matching, and select again $\frac{1}{3}$-quantile of the finally matched pairs after subpixel matching in the same way. This is detailed at line 13 and 23 in Algorithm 12.

Algorithm 11: Compute the blocks in the moving image that match the blocks in the reference image by matching their surrounding rings at a subpixel precision. (**subpixelMatch**)

```
Input img_ref: reference image of size $H \times W$
Input img_mov: moving image of size $H \times W$
Input $\boldsymbol{B}_{ref}$: list of pre-matched blocks in reference image
Input $\boldsymbol{B}_{mov}$: list of pre-matched blocks in reference image
Input $w$: block size
Input $th$: thickness of surrounding ring
Input $s$: upsampling scale for subpixel matching
Output $\boldsymbol{B}'_{ref}$: list of subpixel-matched blocks in reference image
Output $\boldsymbol{B}'_{mov}$: list of subpixel-matched blocks in moving image
pos_ref $\leftarrow$ {the coordinate of the top-left pixel of $B_r$ in img_ref $\mid B_r \in \boldsymbol{B}_{ref}$}
pos_mov $\leftarrow$ {the coordinate of the top-left pixel of $B_r$ in img_mov $\mid B_r \in \boldsymbol{B}_{mov}$}
valid_indices $\leftarrow \emptyset$
for $i \leftarrow 1$ to #pos_mov do
    if pos_mov($i$) is at least 4 pixels away from the border of img_mov then
        append $i$ to valid_indices
pos_ref $\leftarrow$ pos_ref(valid_indices)
pos_mov $\leftarrow$ pos_mov(valid_indices)
$f \leftarrow 2^s$
img_blur_ref $\leftarrow$ GaussianBlur(img_ref)
img_blur_mov $\leftarrow$ GaussianBlur(img_mov)
img_blur_mov_up $\leftarrow$ upsample(img_blur_mov, $f$)                # see Algorithm 9
pos_mov_up $\leftarrow$ pos_mov $\times f$
$\Omega_{\text{offset}} \leftarrow \{1-f, \dots, f-1\} \times \{1-f, \dots, f-1\}$
for $n \leftarrow 1$ to #pos_ref do
    $(i_r, j_r) \leftarrow$ pos_ref($n$), $(i_m, j_m) \leftarrow$ pos_mov_up($n$)
    $R_r \leftarrow$ the surrounding ring of thickness $th$ in img_blur_ref, starting at $(i_r - th, j_r - th)$, whose
     inner block is of shape $w \times w$
    cost_best $\leftarrow +\infty$
    $(\delta i_{best}, \delta j_{best}) \leftarrow (0, 0)$
    for $(\delta i, \delta j) \in \Omega_{offset}$ do
        $R_m \leftarrow$ the “sparse ring” with pixels spaced by $f$ pixels along x-axis and y-axis in
         img_blur_mov, starting at $(i_m + \delta i - th \times f, j_m + \delta j - th \times f)$, whose inner block is of shape
         $fw \times fw$
        cost $\leftarrow$ SAD between $R_m$ and $R_r$
        if cost < cost_best then
            $(\delta i_{best}, \delta j_{best}) \leftarrow (\delta i, \delta j)$
            cost_best $\leftarrow$ cost
    pos_mov_up($n$) $\leftarrow$ pos_mov_up($n$) + $(\delta i_{best}, \delta j_{best})$
img_mov_up $\leftarrow$ upsample(img_mov, $f$)                          # see Algorithm 9
$\boldsymbol{B}'_{ref} \leftarrow \emptyset$, $\boldsymbol{B}'_{mov} \leftarrow \emptyset$
for $n \leftarrow 1$ to #pos_ref do
    $(i_r, j_r) \leftarrow$ pos_ref($n$), $(i_m, j_m) \leftarrow$ pos_mov_up($n$)
    $B_r \leftarrow$ the block in img_ref starting at $(i_r, j_r)$ with $w \times w$ pixels
    append $B_r$ to $\boldsymbol{B}'_{ref}$
    $B_m \leftarrow$ the “sparse” block in img_mov_up starting at $(i_m, j_m)$ with $w \times w$ pixels spaced by $f$
     pixels along x-axis and y-axis
    append $B_m$ to $\boldsymbol{B}'_{mov}$
return $\boldsymbol{B}'_{ref}$, $\boldsymbol{B}'_{mov}$
```

Algorithm 12: Estimate noise curve at from two successive images. Subpixel matching and multi-scale noise estimation are available as options.

```
Input img_ref: reference image of size $H \times W$
Input img_mov: moving image of size $H \times W$
Input $w$: block size
Input $T$: frequency separator
Input $q$: quantile of blocks used for estimation
Input $th$: thickness of compared area
Input $s$: search range
Input $b$: number of bins
Input f_us: upscaling factor for subpixel matching, the matching precision is $1/2^{\text{f\_us}}$
Input is_raw: a boolean indicating if the input images are raw images
Output intensities: intensities of the noise curve
Output variances: noise variances of the noise curve

if is_raw then num_scale $\leftarrow 1$
else num_scale $\leftarrow 4$
intensities $\leftarrow$ zero matrix of size num_scale $\times b$
variances $\leftarrow$ zero matrix of size num_scale $\times b$
for scale $\leftarrow 0$ to num_scale $- 1$ do
    f_ds $\leftarrow 2^{\text{scale}}$
    $w\_match \leftarrow w \times$ f_ds    # use larger size for block matching
    $\boldsymbol{B}_{ref}, \boldsymbol{B}_{mov} \leftarrow$ pixelMatch(img_ref, img_mov, $w\_match$, $th$, $s$)    # get $N$ matched pairs of blocks from two images, see Algorithm 1
    $\boldsymbol{R}_{ref}, \boldsymbol{R}_{mov} \leftarrow$ partition($\boldsymbol{B}_{ref}$, $\boldsymbol{B}_{mov}$, $b$)  # get $b$ bins of block pairs, see Algorithm 3
    $\boldsymbol{B}'_{ref,all} \leftarrow \emptyset$, $\boldsymbol{B}'_{mov,all} \leftarrow \emptyset$
    for $i \leftarrow 1$ to $b$ do
        $\boldsymbol{B}_{ref} \leftarrow \boldsymbol{R}_{ref}(i)$, $\boldsymbol{B}_{mov} \leftarrow \boldsymbol{R}_{mov}(i)$
        $\boldsymbol{B}'_{ref}, \boldsymbol{B}'_{mov} \leftarrow$ selectBlockPairs($\boldsymbol{B}_{ref}, \boldsymbol{B}_{mov}, 3 \times q \times 0.7^{\text{scale}}, T$)    # pre-select a $3q$-quantile of the matched blocks for subsequent subpixel matching, see Algorithm 4
        $\boldsymbol{B}'_{ref,all} \leftarrow \boldsymbol{B}'_{ref,all} \cup \boldsymbol{B}'_{ref}$
        $\boldsymbol{B}'_{mov,all} \leftarrow \boldsymbol{B}'_{mov,all} \cup \boldsymbol{B}'_{mov}$
    if f_us $> 0$ then
        $\boldsymbol{B}'_{ref,all}, \boldsymbol{B}'_{mov,all} \leftarrow$ subpixelMatch($\boldsymbol{B}'_{ref,all}, \boldsymbol{B}'_{mov,all}, w\_match, th,$ f_us)    # Algorithm 11
    $\boldsymbol{B}'_{ref,all} \leftarrow \{$downscale($B_r$, f_ds) $\mid B_r \in \boldsymbol{B}'_{ref,all}\}$    # see Algorithm 10
    $\boldsymbol{B}'_{mov,all} \leftarrow \{$downscale($B_m$, f_ds) $\mid B_m \in \boldsymbol{B}'_{mov,all}\}$
    for $i \leftarrow 1$ to $b$ do
        $\boldsymbol{B}'_{ref} \leftarrow \{\boldsymbol{B}'_{ref,all}(j) \mid j = (i-1) \times \lfloor \frac{\#\boldsymbol{B}'_{ref,all}}{b} \rfloor + 1, \dots, i \times \lfloor \frac{\#\boldsymbol{B}'_{ref,all}}{b} \rfloor\}$
        $\boldsymbol{B}'_{mov} \leftarrow \{\boldsymbol{B}'_{mov,all}(j) \mid j = (i-1) \times \lfloor \frac{\#\boldsymbol{B}'_{mov,all}}{b} \rfloor + 1, \dots, i \times \lfloor \frac{\#\boldsymbol{B}'_{mov,all}}{b} \rfloor\}$
        $\boldsymbol{B}''_{ref}, \boldsymbol{B}''_{mov} \leftarrow$ selectBlockPairs($\boldsymbol{B}'_{ref}, \boldsymbol{B}'_{mov}, \frac{1}{3}, T$)    # see Algorithm 4
        intensity $\leftarrow \frac{1}{|\boldsymbol{B}''_{ref}| + |\boldsymbol{B}''_{mov}|} \times (\sum_{U \in \boldsymbol{B}''_{ref}} \text{mean}(U) + \sum_{U \in \boldsymbol{B}''_{mov}} \text{mean}(U))$
        variance $\leftarrow$ computeVarianceFromPairs($\boldsymbol{B}''_{ref}, \boldsymbol{B}''_{mov}, T$)    # see Algorithm 5
        intensities(scale, $i$) $\leftarrow$ intensity
        variances(scale, $i$) $\leftarrow$ variance
return intensities, variances
```

9 Conclusion

In the proposed method, the temporal redundancy of video frames is employed to suppress the signal for better noise estimation of a raw video. More specifically, block matching is used to suppress the signal in inter-block differences, combined with the Ponomarenko principle of sorting blocks by their low-frequency energies and estimating noise in the DCT high frequencies of difference blocks. The strategy of separating the areas for matching and noise estimation of a block is proposed to avoid the under-estimation of noise caused by the minimization of inter-block difference of the block matching. A fusion method is also proposed to further improve the noise curve estimation from a set of separate noise curves. Experiments show the improvement of our method upon the Ponomarenko method when using larger block size. Different effects of the parameter setting are analyzed, and the selection of the optimal parameter depends on the noise levels of the video, thus one can first use the default parameters to estimate the noise level to determine the optimal parameters, then apply again the estimator with the optimal parameters to obtain more precise noise curves. Besides, the limitations of over-estimation and under-estimation of noise levels are discussed. An extension incorporating subpixel matching with interpolation by Fourier Transform is studied and shows its effectiveness only in the case of strong noise. And finally an extension of multiscale noise estimation is presented as a tool to estimate colored noise in a video.

Acknowledgement

This work was supported by grants from ANR (APATE, ANR-22-CE39-0016), Horizon Europe VERA.AI (No. 101070093), Rgion le-de-France and UDOPIA (ANR-20-THIA-0013). Centre Borelli is also a member of Universit Paris Cit, SSA and INSERM.

Image Credits

MilesDeep. *I was reported to the FAA (what went wrong)* [Video]. YouTube[4].

Drone Snap. *Hong Kong - by drone [4K]* [Video]. YouTube[5].

Travel Drone. *Bordeaux, France by Drone 4K - Aerial Drone Journey* [Video]. YouTube[6].

images from the authors.

References

[1] A. Amer and E. Dubois, *Fast and Reliable Structure-Oriented Video Noise Estimation*, IEEE Transactions on Circuits and Systems for Video Technology, 15 (2005), pp. 113–118, https://doi.org/10.1109/TCSVT.2004.837017.

[4] https://www.youtube.com/watch?v=C474T9PaGQA

[5] https://www.youtube.com/watch?v=dRHVb4ybua0

[6] https://www.youtube.com/watch?v=_RlvEPNgDWo

[2] A. Buades and J. L. Lisani, *Denoising of Noisy and Compressed Video Sequences*, in International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, (VISIGRAPP 2017), INSTICC, SciTePress, 2017, pp. 150–157, https://doi.org/10.5220/0006101501500157.

[3] M. Colom and A. Buades, *Analysis and Extension of the Percentile Method, Estimating a Noise Curve from a Single Image*, Image Processing On Line, 3 (2013), pp. 332–359. https://doi.org/10.5201/ipol.2013.90.

[4] ——, *Analysis and Extension of the Ponomarenko et Al. Method, Estimating a Noise Curve from a Single Image*, Image Processing On Line, 3 (2013), pp. 173–197. https://doi.org/10.5201/ipol.2013.45.

[5] M. Colom, A. Buades, and J.-M. Morel, *Nonparametric Noise Estimation Method for Raw Images*, J. Opt. Soc. Am. A, 31 (2014), pp. 863–871, https://doi.org/10.1364/JOSAA.31.000863.

[6] T. Ehret, A. Davy, M. Delbracio, and J.-M. Morel, *How to Reduce Anomaly Detection in Images to Anomaly Detection in Noise*, Image Processing On Line, 9 (2019), pp. 391–412. https://doi.org/10.5201/ipol.2019.263.

[7] G. Facciolo, N. Limare, and E. Meinhardt-Llopis, *Integral Images for Block Matching*, Image Processing On Line, 4 (2014), pp. 344–369. https://doi.org/10.5201/ipol.2014.57.

[8] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, *Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data*, IEEE Transactions on Image Processing, 17 (2008), pp. 1737–1754, https://doi.org/10.1109/TIP.2008.2001399.

[9] A. E. Gamal, B. A. Fowler, H. Min, and X. Liu, *Modeling and Estimation of FPN Components in CMOS Image Sensors*, in Solid State Sensor Arrays: Development and Applications II, M. M. Blouke, ed., vol. 3301, International Society for Optics and Photonics, SPIE, 1998, pp. 168 – 177, https://doi.org/10.1117/12.304560.

[10] M. Gardella, P. Mus, J.-M. Morel, and M. Colom, *Noisesniffer: a Fully Automatic Image Forgery Detector Based on Noise Analysis*, in IEEE International Workshop on Biometrics and Forensics (IWBF), 2021, pp. 1–6, https://doi.org/10.1109/IWBF50991.2021.9465095.

[11] M. Ghazal, A. Amer, and A. Ghrayeb, *Structure-Oriented Spatio-Temporal Video Noise Estimation*, in IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 2, 2006, pp. II–II, https://doi.org/10.1109/ICASSP.2006.1660475.

[12] ——, *A Real-Time Technique for Spatio-Temporal Video Noise Estimation*, IEEE Transactions on Circuits and Systems for Video Technology, 17 (2007), pp. 1690–1699, https://doi.org/10.1109/TCSVT.2007.903805.

[13] M. Izadi, N. Birkbeck, and B. Adsumilli, *Mutual Noise Estimation Algorithm for Video Denoising*, in IEEE International Conference on Image Processing (ICIP), 2019, pp. 2424–2428, https://doi.org/10.1109/ICIP.2019.8803260.

[14] Y. Li, M. Gardella, Q. Bammey, T. Nikoukhah, R. G. von Gioi, M. Colom, and J.-M. Morel, *Video Signal-Dependent Noise Estimation Via Inter-Frame Prediction*, in IEEE International Conference on Image Processing (ICIP), 2022, pp. 1406–1410, https://doi.org/10.1109/ICIP46576.2022.9897605.

[15] X. Liu, M. Tanaka, and M. Okutomi, *Single-Image Noise Level Estimation for Blind Denoising*, IEEE Transactions on Image Processing, 22 (2013), pp. 5226–5237, https://doi.org/10.1109/TIP.2013.2283400.

[16] S. B. Mohan, T. A. Raghavendiran, and R. Rajavel, *Patch Based Fast Noise Level Estimation Using DCT and Standard Deviation*, Cluster Computing, 22 (2019), pp. 14495–14504. https://doi.org/10.1007/s10586-018-2327-4.

[17] V. A. Pimpalkhute, R. Page, A. Kothari, K. M. Bhurchandi, and V. M. Kamble, *Digital Image Noise Estimation Using DWT Coefficients*, IEEE Transactions on Image Processing, 30 (2021), pp. 1962–1972, https://doi.org/10.1109/TIP.2021.3049961.

[18] N. N. Ponomarenko, V. V. Lukin, M. S. Zriakhov, A. Kaarna, and J. Astola, *An Automatic Approach to Lossy Compression of AVIRIS Images*, in IEEE International Geoscience and Remote Sensing Symposium, 2007, pp. 472–475, https://doi.org/10.1109/IGARSS.2007.4422833.

[19] S. Pyatykh, J. Hesser, and L. Zheng, *Image Noise Level Estimation by Principal Component Analysis*, IEEE Transactions on Image Processing, 22 (2013), pp. 687–699, https://doi.org/10.1109/TIP.2012.2221728.

[20] M. Rakhshanfar and M. A. Amer, *Estimation of Gaussian, Poissonian-Gaussian, and Processed Visual Noise and its Level Function*, IEEE Transactions on Image Processing, 25 (2016), pp. 4172–4185, https://doi.org/10.1109/TIP.2016.2588320.

[21] B. S. Reddy and B. N. Chatterji, *An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration*, IEEE Transactions on Image Processing, 5 (1996), pp. 1266–1271, https://doi.org/10.1109/83.506761.

[22] I. E. Richardson, *The H. 264 Advanced Video Compression Standard*, John Wiley & Sons, 2011. ISBN 9780470989418.

[23] N. Sabater, J.-M. Morel, and A. Almansa, *How Accurate Can Block Matches Be in Stereo Vision?*, SIAM Journal on Imaging Sciences, 4 (2011), pp. 472–500, https://doi.org/10.1137/100797849.

[24] S.-C. Tai and S.-M. Yang, *A Fast Method for Image Noise Estimation Using Laplacian Operator and Adaptive Edge Detection*, in International Symposium on Communications, Control and Signal Processing, 2008, pp. 1077–1081, https://doi.org/10.1109/ISCCSP.2008.4537384.

[25] J. Xiao, H. Tian, Y. Zhang, Y. Zhou, and J. Lei, *Blind Video Denoising Via Texture-Aware Noise Estimation*, Computer Vision and Image Understanding, 169 (2018), pp. 1–13, https://doi.org/10.1016/j.cviu.2017.11.012.

[26] S.-M. Yang and S.-C. Tai, *A Fast and Reliable Algorithm for Video Noise Estimation Based on Spatio-Temporal Sobel Gradients*, in International Conference on Electrical, Control and Computer Engineering (InECCE), 2011, pp. 191–195, https://doi.org/10.1109/INECCE.2011.5953874.

[27] W. Yin and H. Zhao, *A Novel Method of Video Noise Estimation Based on Motion Estimation*, in IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC), 2012, pp. 295–299, https://doi.org/10.1109/ICSPCC.2012.6335636.

[28] L. Zhu and P. Xu, *Differential Video Noise Estimation*, in International Conference on Communication Technology, 2006, pp. 1–4, https://doi.org/10.1109/ICCT.2006.341753.

[29] V. Zlokolica, A. Pizurica, and W. Philips, *Noise Estimation for Video Processing Based on Spatio-Temporal Gradients*, IEEE Signal Processing Letters, 13 (2006), pp. 337–340, https://doi.org/10.1109/LSP.2006.870481.