Published in Image Processing On Line on 2026–07–00.
Submitted on 2025–12–17, accepted on 2026–06–18.
ISSN 2105–1232 © 2026 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2026.662

Photo Response Non-Uniformity Extraction and Detection

Benjamin Loison[1,2], Quentin Bammey[3,4], Rafael Grompone von Gioi[2], Pablo Musé[2], Marina Gardella[2]

[1]AMIAD, Pôle recherche, France
[2]ENS Paris-Saclay, Université Paris-Saclay, Centre Borelli, CNRS, 91190 Gif-sur-Yvette, France
[3]LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[4]EPFL, Image and Visual Representation Lab, Suisse
benjamin.loison@polytechnique.edu

Communicated by T. Ehret and Y. Li *Demo edited by* R. Grompone, Q. Bammey, P. Musé and B. Loison

Abstract

Identifying the specific device model or instance that captured a given image is crucial in various forensic applications. In this context, the Photo Response Non-Uniformity (PRNU) plays a central role. The PRNU is a component of the image noise caused by the manufacturing process of digital cameras. Each pixel sensor responds slightly differently to the same light stimulus, under- or over-estimating the incoming signal in a way that depends on subtle, fixed imperfections unique to that sensor. This pixel-level variation creates a characteristic pattern that acts as a fingerprint of the camera. Because this fingerprint is intrinsic to the physical sensor, it remains consistent across different images taken with the same device. Furthermore, even among cameras of the same model, which share identical hardware and image processing pipelines, the PRNU pattern is distinct. The ability to extract and compare these fingerprints enables the attribution of an image to a specific device. The goal of this article is to provide a unified and transparent implementation of PRNU-based source camera identification methods, including several state-of-the-art approaches from the literature. We aim to systematically compare these methods, with particular emphasis on reproducibility, clarity of implementation, and ease of experimentation. Our objective is to deepen the understanding of the underlying assumptions, design choices, and processing steps involved in PRNU estimation and matching.

Source Code

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

Keywords: image forensics; photo response non-uniformity; source camera identification

[1]https://doi.org/10.5201/ipol.2026.662

1 Introduction

Providing information about the device that captured an image, such as the device model or the specific device instance, is important for various forensic applications. While cameras aim to capture scenes as faithfully as possible, this work focuses on image noise: the discrepancy between the real-world scene and the image produced by the camera. In particular, we concentrate on the Photo Response Non-Uniformity (PRNU), a deterministic component of the image noise that arises from slight manufacturing defects in the image sensor. Each pixel exhibits a small, consistent multiplicative gain error, causing it to slightly over-estimate or under-estimate the intensity of incoming light. These gain variations are independent across pixels, remain stable over time and across images, and act as a unique fingerprint of the sensor that captured the photo [14, 26].

The uniqueness and stability of the PRNU enable one to attribute an image to a specific camera, distinguishing even among different instances of the same camera model, even though these instances share identical hardware and processing pipelines. The primary application of PRNU analysis is source camera attribution in image forensics [26, 5]. Beyond that, PRNU analysis is also employed in image and video forgery detection [20, 28], user authentication [32], scanner identification [12, 18], synthetic images detection [27], and cross-platform matching of user profiles [3].

To estimate a given camera's PRNU, the standard procedure, depicted in Figure 1, is to extract and combine noise residuals from multiple images taken with this camera. Section 3.1 details how noise residuals can be extracted using a denoiser. Section 3.2 highlights how combining them helps suppress the random components of noise and reveals the fixed pattern that is common to all images from that camera. While this residual combining provides a first PRNU estimate, it is often polluted by artifacts caused not by the sensors, but by the Image Signal Processor (ISP) of that camera or by the software used to process the image. These artifacts, called *non-unique artifacts*, need to be removed from the fingerprint estimate, so as to avoid misattributing an image to another instance of the same camera. Section 3.3 showcases different processing techniques which can be used to remove or suppress the non-unique artifacts from the PRNU estimate.

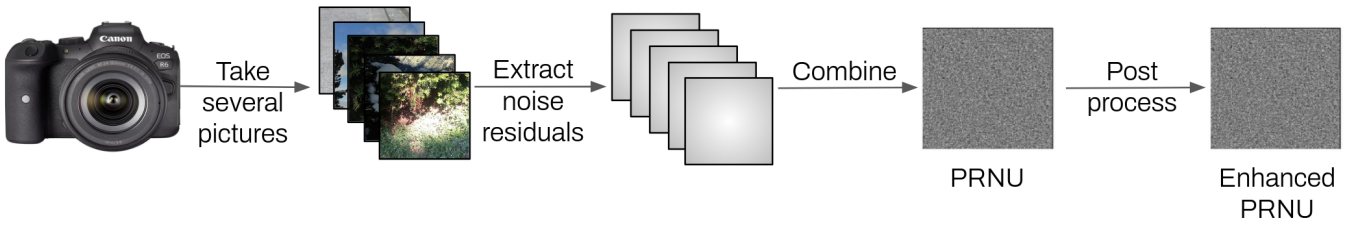


Figure 1: Typical PRNU estimation procedure.

Once the PRNU has been estimated, source camera attribution can be formulated as a statistical hypothesis testing against the absence of PRNU in a query image, typically, as described in Section 4, using correlation-based statistics: a high correlation enables matching an image and its camera.

In this article, we aim to provide an analysis and a unified implementation of several state-of-the-art PRNU-based source camera identification methods, compare their performance, and emphasize reproducibility and clarity of implementation. This also enables a better understanding of the underlying assumptions and the individual steps involved in PRNU extraction, refinement, and detection.

2 Sensor Noise and PRNU

The concept of noise in an image stems from the acquisition process. The first step in acquiring a raw image is to count the number of incident photons during exposure time using a sensor. This

process is inherently noisy. The *noise* in an image thus corresponds to the difference between the ideal image representing the actual scene and the observed one. Image noise is typically categorized into two broad classes:

Random noise. On the one hand, the so-called *random noise* causes variations in pixel values above and below the actual image intensity. These variations differ randomly across repeated captures of the same scene and do not depend on the specific location of a pixel. Many sources of noise belong to this family: *shot noise*, which is due to the physical nature of the light; *dark current fluctuations*, which are present even in the absence of light; and thermal and electronic readout noise. Under typical capturing conditions, *shot noise* is dominant. Random noise is typically assumed to have zero mean, meaning it does not bias pixel values. As a result, when multiple images of the same scene are averaged, this noise tends to cancel out.

Pattern noise. On the other hand, *pattern noise* [15], also known as *non-uniformities*, refers to deterministic variations that are consistent across images captured by the same sensor. This kind of noise depends on the position of each pixel within the image and can take different forms: an additional count of photons in the dark, called *dark signal non-uniformity* (DSNU), a periodic noise due to electromagnetic interference in the camera, and pixel-specific deviations in sensitivity due to manufacturing imperfections, called *photo response non-uniformity* (PRNU). Pattern noise is by nature localized and, unlike random noise, it persists and accumulates in repeated captures.

To formalize the interaction of these noise components in the image formation process, we adopt the sensor output model introduced by Chen et al. [5], which describes the signal before demosaicking for a given color channel. In this model, the incident light intensity at each pixel is first corrupted by the camera's Photo-Response Non-Uniformity (PRNU), a multiplicative noise term. Random additive noise is then introduced. The resulting signal is scaled by the channel-specific gain, subject to a gamma correction, and finally distorted by quantization noise. Figure 2 illustrates this simplified processing chain. In matrix form, the sensor output model can be written as

$$\mathbf{I} = g^\gamma \cdot [(\mathbf{1} + \mathbf{K}_a)\mathbf{Y} + \mathbf{\Lambda}]^\gamma + \mathbf{\Theta}_q, \tag{1}$$

where $\mathbf{Y}$ is the matrix of incident light intensities at each pixel, g is the color channel gain used for white balancing, γ is the gamma correction factor (typically $\gamma \approx 0.45$), $\mathbf{K}_a$ is the PRNU factor, modeled as a zero-mean Gaussian multiplicative noise, $\mathbf{\Lambda}$ aggregates the additive random noise sources discussed above, and $\mathbf{\Theta}_q$ is the quantization noise. Throughout this article, bold symbols denote matrices while scalar quantities appear in regular font. All operations are understood to be element-wise.

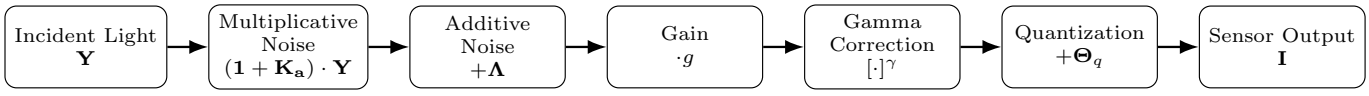


Figure 2: Simplified sensor output model. The incident light $\mathbf{Y}$ is first multiplied by the device-specific PRNU factor $\mathbf{K}_a$, then affected by additive noise $\mathbf{\Lambda}$. The signal is amplified by a gain factor g, transformed by a power-law function $[\cdot]^\gamma$ known as gamma correction, and finally quantized, introducing the quantization noise $\mathbf{\Theta}_q$.

Factoring out the scene light intensity $\mathbf{Y}$ in the first term, Equation (1) can be rewritten as

$$\mathbf{I} = (g\mathbf{Y})^\gamma \cdot \left[\mathbf{1} + \left(\mathbf{K}_a + \frac{\mathbf{\Lambda}}{\mathbf{Y}}\right)\right]^\gamma + \mathbf{\Theta}_q. \tag{2}$$

Assuming that the PRNU factor and the additive noise are small, we can simplify this expression by applying a first-order Maclaurin expansion of $(1+x)^\gamma = 1 + \gamma x + O(x^2)$

$$\mathbf{I} \approx (g\mathbf{Y})^\gamma \cdot \left(1 + \gamma \mathbf{K}_a + \gamma \frac{\mathbf{\Lambda}}{\mathbf{Y}}\right) + \mathbf{\Theta}_q. \tag{3}$$

To further simplify, we absorb the gamma correction factor into the PRNU factor $\mathbf{K}_a$, thus instead of $\gamma \mathbf{K}_a$ we write $\mathbf{K}$ and introduce $\mathbf{I}^{(0)} = (g\mathbf{Y})^\gamma$ and $\mathbf{\Theta} = \gamma \mathbf{I}^{(0)} \frac{\mathbf{\Lambda}}{\mathbf{Y}} + \mathbf{\Theta}_q$, being respectively the sensor output signal in the absence of noise and the sum of multiple independent noise components

$$\mathbf{I} \approx \mathbf{I}^{(0)} + \mathbf{I}^{(0)}\mathbf{K} + \mathbf{\Theta}. \tag{4}$$

This model captures the key sources of variability in sensor output and serves as the basis for PRNU-based source attribution and forensic analysis.

3 PRNU Extraction

In this section we describe the different steps involved in extracting a camera PRNU from multiple images, as illustrated in Figure 1. The first step involves denoising the images to obtain their noise residuals. The second step consists of combining these residuals to form an aggregate estimate. Finally, an optional post-processing step can be applied to refine the resulting *raw* PRNU estimate.

3.1 Noise Residuals Extraction

The first step in PRNU estimation involves extracting noise residuals from a set of images using denoising algorithms. These algorithms aim to suppress the scene content in each image and isolate the underlying noise components, ideally extracting the signal $\mathbf{I}^{(0)}\mathbf{K} + \mathbf{\Theta}$, where $\mathbf{I}^{(0)}$ represents the original scene noise-independent signal, $\mathbf{K}$ the PRNU, and $\mathbf{\Theta}$ the remaining random noise.

While various denoisers can be employed for this task, none is perfect. In practice, scene content often remains partially preserved, resulting in residuals that still contain traces of underlying image structures. This phenomenon, known as scene leakage, can significantly affect the quality of the PRNU estimates. In certain cases, such as when processing flat-field images, i.e., photos of uniformly lit textureless scenes, it is possible to mitigate scene leakage. Section 3.1.3 explores this particular case in more detail, providing a practical solution to reduce the influence of residual scene content in such controlled scenarios.

The remainder of this subsection presents several denoising algorithms commonly used for noise residual extraction in PRNU estimation.

3.1.1 Wavelet-based Filter

To estimate the noise component of an image, the denoising method proposed by Mihcak et al. [19] employs a discrete wavelet transform (DWT) based on Daubechies wavelets [9]. The Daubechies wavelet considered here has 8 taps and is known as the db4 wavelet. While the limited number of taps imposes some constraints on the precision of signal representation, db4 filters provide a good trade-off between detail preservation and computational efficiency.

The wavelet decomposition separates the image into two types of coefficients: approximation and detail coefficients. The approximation coefficients, denoted as $\mathbf{a}$, represent a low-frequency, down-sampled version of the image that captures general shapes and smooth variations. In contrast, the detail coefficients capture the high-frequency components. These detail coefficients are further

subdivided into three orientations: horizontal details, denoted as **h**; vertical details, denoted as **v**; and diagonal details, denoted as **d**.

After the first decomposition, the approximation coefficients (which represent a smoothed, low-frequency version of the original image) are again subjected to the same wavelet filtering and down-sampling process. This process is repeated multiple times, each time increasing the frequency resolution of the transform, allowing the analysis of signal components at different scales. At each level, the spatial resolution of the approximation signal is reduced, but the ability to distinguish different frequency bands improves. In the context of this method, the decomposition is carried out up to four levels. This multiscale decomposition structure, including the recursive splitting of the approximation band, is illustrated in Figure 3, where at each level we introduce one approximation subband and three detail subbands: horizontal, vertical, and diagonal.

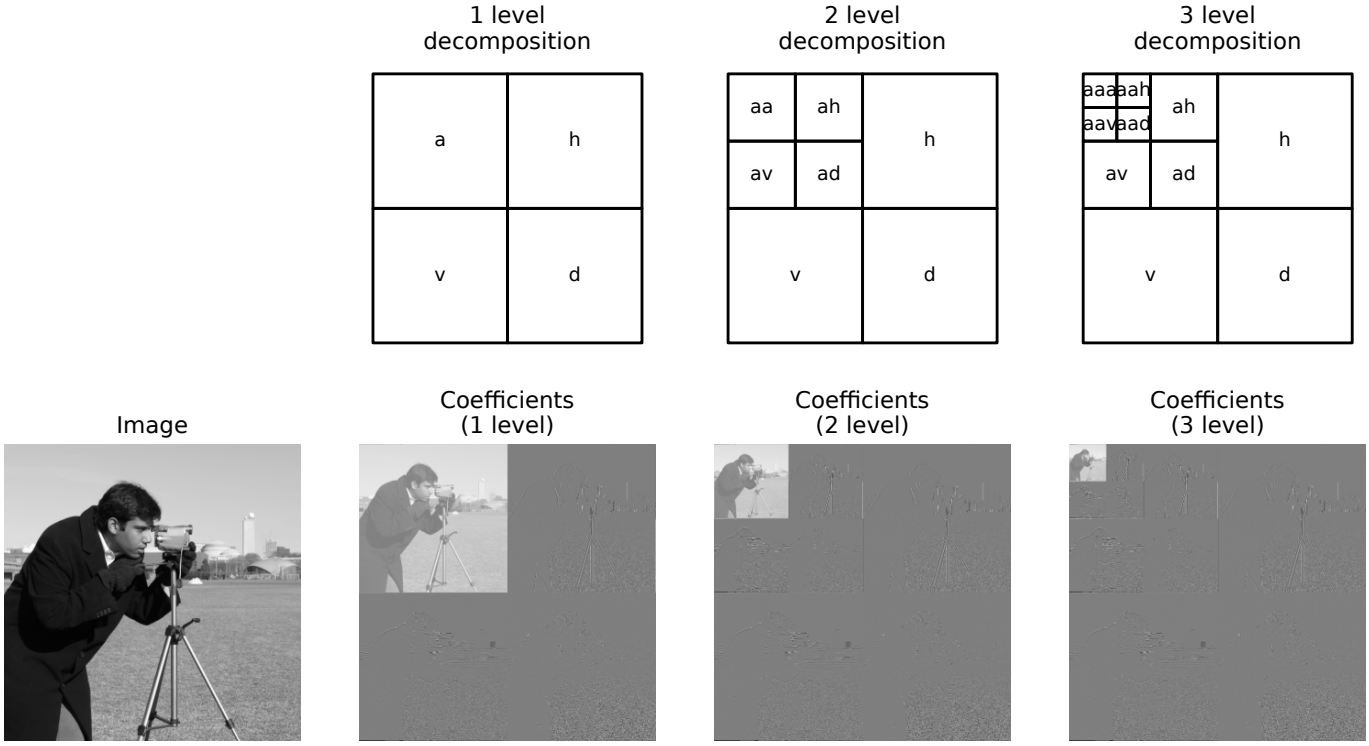


Figure 3: The original image (left) is progressively decomposed into multiple subbands using the Discrete Wavelet Transform (DWT) [24]. At level 1, the image is split into four components: approximation (**a**), horizontal detail (**h**), vertical detail (**v**), and diagonal detail (**d**). At level 2, the approximation subband **a** is further decomposed into the new approximation **aa**, the horizontal detail **ah**, the vertical detail **av**, and the diagonal detail **ad**, while **h**, **v**, and **d** remain from level 1. At level 3, the process is repeated on the **aa** subband, producing **aaa**, **aah**, **aav**, and **aad**. This recursive decomposition allows for multiscale analysis, capturing both coarse and fine details of the image structure.

To estimate the image noise, it is not sufficient to simply set to zero the wavelet approximation coefficients and invert the transform. While this operation removes the coarse content, the detail coefficients at each decomposition level still contain significant contributions from scene structures such as edges, textures, and patterns. As a result, the inverse transform would yield a signal that remains contaminated by structured content and does not represent pure noise.

To more effectively isolate the noise, it is necessary to further process the detail coefficients. To do so, a Wiener filter is applied independently to each detail subband. For each coefficient (i, j) in a given detail subband $\mathbf{w} \in \{\mathbf{h}, \mathbf{v}, \mathbf{d}\}$, the denoised coefficient is computed as

$$\mathbf{w}_{\text{denoised}}(i,j) = \mathbf{w}(i,j)\,\frac{\hat{\boldsymbol{\sigma}}^2(i,j)}{\hat{\boldsymbol{\sigma}}^2(i,j) + \sigma^2}, \tag{5}$$

where σ^2 is the global noise variance, which is left as a parameter of the method, and $\hat{\boldsymbol{\sigma}}^2(i,j)$ denotes the local variance of the wavelet coefficients of the "original" noise-free image, estimated as

$$\hat{\boldsymbol{\sigma}}^2(i,j) = \min_{q\in\{1,\dots,4\}} \left(\max\left(0, \frac{1}{(2q+1)^2} \sum_{m=i-q}^{i+q} \sum_{n=j-q}^{j+q} \mathbf{w}^2(m,n) - \sigma^2 \right) \right). \tag{6}$$

The adaptive window size q allows the method to focus on narrow regions around edges, where local variations are significant. In regions where the local variance $\hat{\boldsymbol{\sigma}}^2(i,j)$ is substantial, the Wiener filter applies minimal smoothing. Conversely, in areas where the local variance is low, the Wiener filter increases its smoothing effect. The denoised image $\mathbf{I}_{\text{denoised}}$ is reconstructed by applying the inverse wavelet transform to the denoised wavelet coefficients. The noise residual $\mathbf{R}$ is then computed as $\mathbf{R} = \mathbf{I} - \mathbf{I}_{\text{denoised}}$.

Algorithms 1 and 2 respectively implement the described wavelet-based denoiser and its adaptive Wiener filter. The Python implementation we used is the `noise_extract` function of Bondi et al. [4] matching Algorithm 1.

Algorithm 1: Wavelet-based denoiser

function *wavelet_based_denoiser*($\boldsymbol{I}, \sigma^2$)
- **Input** $\boldsymbol{I}$**:** grayscale image
- **Input** σ^2**:** global noise variance
- **Output** $\boldsymbol{\nu}$**:** estimated noise of the image $\boldsymbol{I}$
- // $\mathcal{W}$ is the 2D wavelet transform from pywavelet's [24] wavedec2 function, with 8-taps Daubechies wavelets ('db4') at decomposition level 4. It returns the approximation coefficients $\boldsymbol{A}$ and the tuple of horizontal, vertical and diagonal detail coefficients $(\boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D})$, each of which being a list of 4 wavelet matrices.
- $\boldsymbol{A}, (\boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D}) \coloneqq \mathcal{W}(\boldsymbol{I})$
- $\boldsymbol{A} \coloneqq 0$ // Set level-4 approximation coefficients to 0.
- **for** $\boldsymbol{L} \in \{\boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D}\}$ **do**
  - **for** $\boldsymbol{W} \in \boldsymbol{L}$ **do**
    - // extract_noise_with_adaptive_wiener_filter is described in Algorithm 2.
    - $\boldsymbol{W} \coloneqq$ `extract_noise_with_adaptive_wiener_filter`$(\boldsymbol{W}, \sigma^2)$
- // $\mathcal{W}^{-1}$ is the inverse wavelet transform, from pywavelet's waverec2 function, with the same parameters as $\mathcal{W}$ above.
- $\boldsymbol{\nu} \coloneqq \mathcal{W}^{-1}(\boldsymbol{A}, (\boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D}))$ // With $\boldsymbol{A}$ set to zeroes and $(\boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D})$ as processed above.
- **return** $\boldsymbol{\nu}$

3.1.2 Block-matching and 3D (BM3D)

Block-Matching and 3D Filtering (BM3D), introduced by Dabov et al. [7], consists of two successive stages that follow a similar sequence of operations. In the first stage, the algorithm searches a predefined spatial neighborhood for patches similar to a reference patch and groups them into a 3D stack. A 3D linear transform is then applied, the transform coefficients are shrunk (typically via hard thresholding), and the estimate is reconstructed through the inverse 3D transform. The resulting patches are finally combined through weighted averaging of their overlapping regions. The

Algorithm 2: Extract noise with adaptive Wiener filter

function *extract_noise_with_adaptive_wiener_filter*($\boldsymbol{W}, \sigma^2$)

// To extract $\boldsymbol{W}$'s noise, for each wavelet coefficient the minimum average variance over different centered window sizes is returned.

Input $\boldsymbol{W}$**:** 2D wavelet coefficients matrix

Input σ^2**:** global noise variance

Output $\boldsymbol{D}$**:** filtered wavelet coefficients

$\hat{\boldsymbol{\sigma}}^{\boldsymbol{2}} \coloneqq \min_{sz \in \{3,5,7,9\}} \left(\max(0, \texttt{uniform\_filter}(\boldsymbol{W}^2, sz) - \sigma^2)\right)$

$\boldsymbol{D} \coloneqq \dfrac{\sigma^2}{\sigma^2 + \hat{\boldsymbol{\sigma}}^{\boldsymbol{2}}} \boldsymbol{W}$

return $\boldsymbol{D}$

second stage mirrors this procedure with two key differences. Patch distances are computed using the filtered patches obtained in the first stage, and the 3D groups – although still formed from the original image samples – are now processed using Wiener filtering rather than simple thresholding. The complete workflow is summarized in Figure 4.

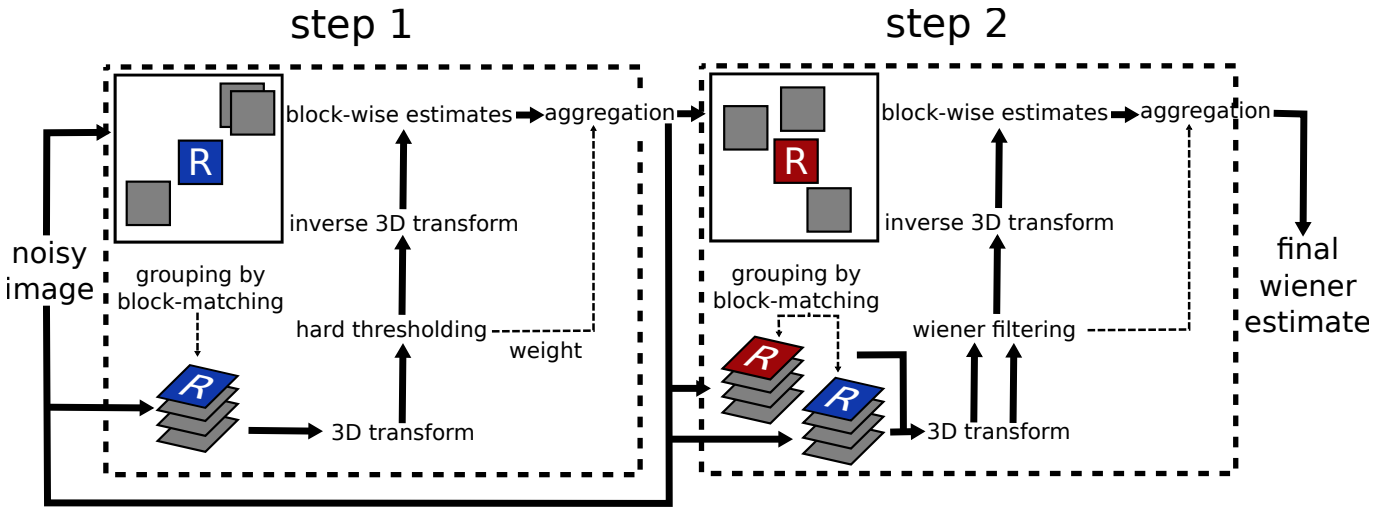
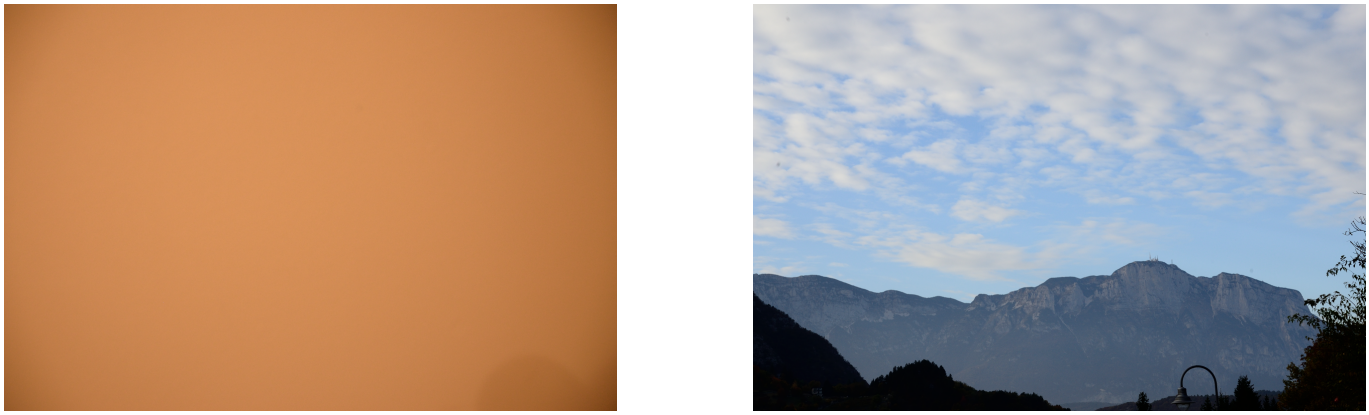


Figure 4: Scheme of the BM3D algorithm [23].

Our BM3D denoiser builds upon the implementation of Lebrun [23], incorporating updates available at https://github.com/gfacciol/bm3d/tree/fab1293534c0a0823081cc9d2c67923909b8c591.

3.1.3 Mean High-pass Filter of Normalized Flat-field Photos

In forensic applications, law enforcement agencies often aim to determine whether a given image was taken by a specific camera. This involves comparing the residue extracted from the image with the PRNU of the suspect camera. When the camera is available, the optimal strategy for estimating its PRNU is to capture a set of flat-field images: photos taken under controlled conditions with uniform, evenly distributed lighting. Indeed, as previously discussed, scene content can leak into the estimated noise residuals, thereby contaminating the PRNU signal. Figure 5 depicts such a flat-field image (left) in contrast to a non-flat-field one (right). This clean reference enhances the accuracy of source camera attribution, which is critical when the outcome may be used as evidence in a judicial context.



(a) Flat-field photo

(b) Non-flat-field photo

Figure 5: Example of flat-field (left) and non-flat-field (right) photos from the RAISE dataset [8]. Flat-field images have uniform lighting and minimal texture, making them suitable for reducing scene leakage in PRNU estimation.

When considering multiple flat-field photos taken by a given camera, a trivial denoiser can be employed to extract the random noise from these images. Specifically, to ensure compatibility with different flat-field scenes, we compute the pixel-wise average of the normalized flat-field images. To obtain a normalized flat-field image, its illumination intensity is divided by its maximum value. Averaging these normalized images suppresses independent random noise while preserving both the fixed PRNU multiplicative factor and the average of normalized flat-field scenes. To isolate the PRNU, we apply a Gaussian filter to the normalized averaged image, producing a smooth approximation of the averaged normalized scene illumination. The estimated PRNU is then obtained by subtracting this filtered version from the averaged image. This procedure, described in Algorithm 3, relies on the assumption that the average flat-field scene is sufficiently smooth to be captured by the low-pass filter and, therefore, should only be used when dealing with flat-field images.

Algorithm 3: Camera PRNU estimation from flat-field photos

function *estimate_camera_prnu_from_flat_field_photos*($(\boldsymbol{I}_l)_{l\in[\![1,L]\!]}$, σ)

Input $(\boldsymbol{I}_l)_{l\in[\![1,L]\!]}$**:** L same camera grayscale flat-field images

Input σ**:** standard deviation for Gaussian kernel to separate the PRNU and the average flat-field scene

Output $\hat{\mathbf{K}}_{\text{FFP}}$**:** estimated PRNU of the camera

$\bar{\boldsymbol{I}} \coloneqq \frac{1}{L}\sum_{l=1}^{L} \frac{\boldsymbol{I}_l}{\max(\boldsymbol{I}_l)}$

$\hat{\mathbf{K}}_{\text{FFP}} \coloneqq \bar{\boldsymbol{I}} - \mathcal{G}(\bar{\boldsymbol{I}}, \sigma)$

return $\hat{\boldsymbol{K}}_{\text{FFP}}$

Alternatively, in theory, this denoiser can also be used with a sufficiently large number of photos from various non-flat-field scenes, as long as the average of these photos is uniform.

3.1.4 Other Denoisers

In this subsection we succinctly describe widely studied well-known image denoisers that have been also considered in PRNU extraction.

The Context-Adaptive Interpolator (CAI) proposed by Kang et al. [16] locally averages a given pixel depending on whether the eight neighbouring pixels belong to a smooth or edge region. A mean filter is used in the smooth regions. In edge regions the prediction is along the edge. In

other regions a median filter is used. A spatial Wiener filter is subsequently applied to eliminate the image scene leakage due to the local average.

The 2-Pixel Approach introduced by Al-Ani [2] estimates $\mathbf{K}(x,y)\mathbf{I}^{(0)}(x,y)$ by leveraging the high sensor pixel density, that is for two close pixels $\mathbf{I}(x,y)$ and $\mathbf{I}(m,n)$: $\mathbf{I}^{(0)}(x,y) \approx \mathbf{I}^{(0)}(m,n)$. One of the two pixels is chosen such that its PRNU estimate sign is the opposite of the other. The rough PRNU estimate sign is obtained by any basic filter, such as the median on multiple photos. $\mathbf{K}(x,y)\mathbf{I}^{(0)}(x,y)$ is then estimated as $(\mathbf{I}(x,y) - \mathbf{I}(m,n))/2$.

The Adaptive Spatial (AS) Filtering defined by Cooper [6] is a spatially varying filter that first applies an adaptive Wiener filter directly in the spatial domain, then applies two 2×2 cascaded median filters. It has been found that median filtering has the potential to increase the correlation magnitude for matching data and, at the same time, reduce the correlation bias for non-matching data. These enhancements are considered to result in a reduction of the effects of JPEG compression.

The Perona-Malik diffusion (PMD) filter [30] was used by Houten et al. [33] for PRNU estimation by reducing image noise without removing parts of the image content, such as edges and other details, that are important for interpreting the image. This filter uses anisotropic diffusion, which generates a parameterized discrete sequence of increasingly blurred images based on a diffusion process. For the denoised output, the authors utilize the image produced at the third iteration.

The Total Variation (TV) filter introduced by Rudin et al. [31] was considered for the PRNU estimation by Gisolf et al. [10]. This filter preserves edges while smoothing away noise in flat regions, even at low signal-to-noise ratios. It reduces the total variation of the image, while preserving edges with only a minimal increase in the mean squared error. To this aim, this filter minimizes $|\nabla \mathbf{I}| + (\mathbf{I}_{\text{denoised}} - \mathbf{I})^2$ using a single gradient-descent step.

3.2 Combining Noise Residuals

Once noise residuals are extracted from a set of L images, the next step in PRNU estimation is to combine them in a way that reinforces the sensor-specific pattern while suppressing random noise. Since the PRNU is a fixed noise component inherent to the camera sensor, it is present across all images captured with that device, whereas random noise varies. In the following, we describe the strategies used to combine noise residuals and estimate the PRNU.

3.2.1 Basic Averaging

To estimate the PRNU of a given camera, the simplest method to combine the noise residuals $\mathbf{R}_1, \ldots, \mathbf{R}_L$ extracted from L images is to compute a simple average

$$\hat{\mathbf{K}}_{\text{BA}} = \frac{1}{L} \sum_{l=1}^{L} \mathbf{R}_l. \tag{7}$$

Note that, according to the model derived in Section 2 and summarized in Equation (4), $\hat{\mathbf{K}}_{\text{BA}}$ actually estimates $\frac{1}{L}\sum_{l=1}^{L} \mathbf{I}_l^{(0)}\mathbf{K}$, instead of $\mathbf{K}$. While the camera fingerprint $\hat{\mathbf{K}}_{\text{BA}}$ is specific to the set of scenes associated to $\left(\mathbf{I}_l^{(0)}\right)_{l\in[\![1,L]\!]}$, in the forensic fingerprint comparison, detailed in Section 4, tested and reference camera fingerprints are usually reliably computed from different sets of scenes, as shown in Section 5.

This approach builds on the fact that the PRNU is a consistent signal present in all images captured by the same sensor, whereas random noise varies and has zero mean. By averaging multiple noise residuals, random noise tends to cancel out, while the PRNU component, being constant, is reinforced. More precisely, averaging L images, results in dividing the standard deviation of the

noise by $\sqrt{L}$. This method is computationally efficient and widely used as a baseline in the literature. However, its effectiveness relies on the quality of the extracted residuals. In particular, if the residuals contain strong scene leakage, the averaging process may fail to isolate the true PRNU.

The basic averaging implementation we used is described in Algorithm 4.

Algorithm 4: Basic averaging

function *basic_averaging*$\big((\boldsymbol{R}_l)_{l\in[\![1,L]\!]}\big)$
  Input $\boldsymbol{R}_l$**:** noise extracted from L grayscale images
  Output $\hat{\mathbf{K}}_{\mathrm{BA}}$**:** estimated PRNU of the camera
  $\hat{\mathbf{K}}_{\mathrm{BA}} \coloneqq \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{R}_l$
  return $\hat{\boldsymbol{K}}_{\mathrm{BA}}$

3.2.2 Weighted Averaging

If the noise variance is constant across all images, simple averaging is theoretically optimal. However, when the noise level differs from image to image, performing a weighted average, as proposed by Lawgaly et al. [22], provides the minimum-mean-square-error estimator. The weighted estimate is computed as

$$\hat{\mathbf{K}}_{\mathrm{WA}} = \sum_{l\in[\![1..L]\!]} w_l \mathbf{R}_l, \tag{8}$$

where w_l is the weight assigned to the l-th image, defined as

$$w_l = \frac{1/\sigma_l^2}{\sum_{s=1}^{L} 1/\sigma_s^2}, \tag{9}$$

with σ_l^2 denoting the noise variance associated with the residual $\mathbf{R}_l$; the denominator runs over all images and uses their corresponding variances σ_s^2.

For $l \in [\![1..L]\!]$, the variance σ_l^2 can be estimated according to Laciar et al. [21] as

$$\hat{\sigma}_l^2 = \frac{\sum_{(x,y)} (\hat{\mathbf{n}}_l(x,y) - \bar{\mathbf{n}}_l)^2}{N}, \tag{10}$$

where N is the total number of pixels, $\bar{\mathbf{n}}_l$ is the mean of the random noise in $\mathbf{R}_l$ and $\hat{\mathbf{n}}_l$ is computed as

$$\hat{\mathbf{n}}_l = \mathbf{R}_l - \hat{\mathbf{K}}_{\mathrm{BA}}, \tag{11}$$

with $\hat{\mathbf{K}}_{\mathrm{BA}}$ denoting the simple (basic) average of all residuals.

3.2.3 Maximum Likelihood Estimator (MLE)

If we assume that the random noise is homoscedastic, that is, the random noise has the same finite variance at every pixel, then the maximum likelihood estimator is [5]

$$\hat{\mathbf{K}}_{\mathrm{MLE}} = \frac{\sum_{l=1}^{L} \mathbf{R}_l \mathbf{I}_{\mathrm{denoised}_l}}{\sum_{l=1}^{L} (\mathbf{I}_{\mathrm{denoised}_l})^2}, \tag{12}$$

where $\mathbf{R}_1, \ldots, \mathbf{R}_L$ are the noise residuals extracted from the input images and $\mathbf{I}_{\mathrm{denoised}_1}, \ldots, \mathbf{I}_{\mathrm{denoised}_L}$ are their denoised versions.

This estimator suggests that, to optimally estimate the PRNU, image luminance should be maximized without reaching saturation. This is indeed expected, as the PRNU is a multiplicative noise component whose strength increases with signal intensity. At the same time, the assumption of noise homoscedasticity can be satisfied when using uniformly illuminated images, such as flat-field captures. Nevertheless, the estimator has also been shown to perform reliably on natural scenes.

The maximum likelihood estimator implementation we used is described in Algorithm 5.

Algorithm 5: Maximum likelihood estimator

function *maximum_likelihood_estimator*$((\boldsymbol{I}_{denoised_l})_{l\in[\![1,L]\!]}, (\boldsymbol{R}_l)_{l\in[\![1,L]\!]})$

Input $(\boldsymbol{I}_{denoised_l})_{l\in[\![1,L]\!]}$**:** L same camera grayscale denoised images

Input $(\boldsymbol{R}_l)_{l\in[\![1,L]\!]}$**:** extracted noise residuals to obtain the provided $(\boldsymbol{I}_{denoised_l})_{l\in[\![1,L]\!]}$

Output $\hat{\mathbf{K}}_{\text{MLE}}$**:** maximum likelihood estimator

$$\hat{\mathbf{K}}_{\text{MLE}} \coloneqq \frac{\sum_{l=1}^{L} \mathbf{R}_l \mathbf{I}_{\text{denoised}_l}}{\sum_{l=1}^{L} \left(\mathbf{I}_{\text{denoised}_l}\right)^2}$$

return $\hat{\boldsymbol{K}}_{\text{MLE}}$

3.2.4 Phase-only Operation

Kang et al. [17] propose a phase-only operation, leveraging the assumption that the PRNU is white noise and hence has a flat frequency spectrum. The phase-only operation consists of whitening each image noise residual, then combining these whitened noise residuals

$$\hat{\mathbf{K}}_{\text{PO}} = \text{real}\left[\mathcal{F}^{-1}\left(\frac{\sum_{l=1}^{L} \mathbf{Ph}_l}{L}\right)\right], \tag{13}$$

where real$[\cdot]$ denotes the real-part operator, $\mathcal{F}^{-1}(\cdot)$ is the inverse of the Fourier transform and $\mathbf{Ph}_l$ denotes the phase component of the l-th noise residual $\mathbf{R}_l$, defined as

$$\mathbf{Ph}_l = \frac{\mathcal{F}(\mathbf{R}_l)}{|\mathcal{F}(\mathbf{R}_l)|}, \tag{14}$$

where $\mathcal{F}(\cdot)$ denotes the Fourier transform and $|\cdot|$ denotes the magnitude.

The phase-only operation implementation we used is described in Algorithm 6.

3.3 PRNU Post-processing

While the previous sections addressed the extraction of noise residuals from images and their combination, it is important to note that these residuals contain not only the unique PRNU pattern of the camera sensor, but also various non-unique artifacts. Non-unique artifacts are usually due to the image processing pipeline whose steps introduce structured patterns that are systematically present in every image produced by the camera. Because these processing operations are shared across many devices, the resulting artifacts are not tied to a specific sensor and may appear in images from multiple cameras of the same model, or even across different models. As non-unique artifacts get undesirably enhanced during the PRNU estimation process, it is necessary to apply post-processing techniques to the *raw* PRNU estimate to discard such artifacts and isolate the true PRNU – the component that uniquely identifies a camera.

Algorithm 6: Phase-only operation

function *phase_only_operation*$\left((\boldsymbol{R}_l)_{l\in[\![1,L]\!]}\right)$

Input $(\boldsymbol{R}_l)_{l\in[\![1,L]\!]}$**:** noise extracted from L grayscale images

Output $\hat{\mathbf{K}}_{\mathrm{PO}}$**:** estimated PRNU of the camera

// $\mathcal{F}$ (respectively $\mathcal{F}^{-1}$) is the (respectively inverse) Fourier transform, from numpy's fft2 (respectively ifft2) function.

for l **from** *1* **to** L **do**

$$\boldsymbol{Ph}_l \coloneqq \frac{\mathcal{F}(\boldsymbol{R}_l)}{|\mathcal{F}(\boldsymbol{R}_l)|}$$

$$\hat{\mathbf{K}}_{\mathrm{PO}} \coloneqq \texttt{real}\left[\mathcal{F}^{-1}\left(\frac{\sum_{l=1}^{L}\boldsymbol{Ph}_l}{L}\right)\right]$$

return $\hat{\boldsymbol{K}}_{\mathrm{PO}}$

3.3.1 Zero-mean and Wiener Filtering

The approach introduced by Chen et al. [5] consists of two main steps. The first step applies a zero-mean operation ZM along the rows and columns, targeting artifacts introduced by color interpolation and row/column-wise processing in imaging sensors and circuits. Specifically, for each pixel, the mean value of its column is subtracted, followed by the subtraction of the mean of its row (or vice versa).

The second step applies a 3×3 Wiener filter W in the frequency domain, aimed at attenuating structured spectral components. The variance used by the filter is computed as the variance of the magnitude of the Fourier transform $|\mathcal{F}(\hat{\mathbf{K}}_{ZM})|$. The Wiener filter is then applied to the Fourier transform of the zero-meaned estimate. Because W suppresses spectral components with strong local structure, this step reduces residual periodic and low-frequency artifacts that do not originate from the true PRNU.

Combining both steps, the final post-processed PRNU is obtained as

$$\hat{\mathbf{K}}_{WF} = \mathcal{F}^{-1}[\mathcal{F}(\hat{\mathbf{K}}_{ZM}) - W(\mathcal{F}(\hat{\mathbf{K}}_{ZM}))]. \tag{15}$$

The implementations of the zero-mean operator and the optional subsequent Wiener filter are described in Algorithm 7 and Algorithm 8, respectively.

3.3.2 Sensor Pattern Noise Enhancer Models

To further reduce scene leakage in the noise residual, [25] introduces a set of nonlinear reweighting functions applied to the wavelet coefficients of the residual. The underlying assumption is that the wavelet coefficients with large magnitude are more likely to originate from scene structures than from the weak PRNU component, and should therefore be attenuated. Conversely, coefficients with small magnitude are considered more reliable PRNU estimates and should be preserved.

Let R_w denote a wavelet coefficient. The method proposes the following parametric models for coefficient reweighting, controlled by a parameter α

Algorithm 7: Set PRNU estimation means of rows and columns to 0

function *set_PRNU_estimation_means_of_rows_and_columns_to_0($\hat{\boldsymbol{K}}$)*

Input $\hat{\mathbf{K}}$: PRNU estimate of size $X \times Y$

Output $\hat{\mathbf{K}}_{ZM}$: $\hat{\mathbf{K}}$ whose row and column means are set to zero

for x **from** *1* **to** X **do**

$\bar{y} \coloneqq \dfrac{\sum_{y=1}^{Y} \hat{\mathbf{K}}[x, y]}{Y}$

$\hat{\mathbf{K}}[x, 1:Y] \mathrel{-}= \bar{y}$

for y **from** *1* **to** Y **do**

$\bar{x} \coloneqq \dfrac{\sum_{x=1}^{X} \hat{\mathbf{K}}[x, y]}{X}$

$\hat{\mathbf{K}}[1:X, y] \mathrel{-}= \bar{x}$

$\hat{\mathbf{K}}_{ZM} \coloneqq \hat{\mathbf{K}}$

return $\hat{\boldsymbol{K}}_{ZM}$

Algorithm 8: Wiener filter zero-mean PRNU estimation

function *wiener_filter_zero_mean_PRNU_estimation($\hat{\boldsymbol{K}}_{ZM}$)*

Input $\hat{\mathbf{K}}_{ZM}$: zero-mean PRNU estimation

Output $\hat{\mathbf{K}}_{WF}$: Wiener filtered $\hat{\mathbf{K}}_{ZM}$ in the Fourier domain

`// wiener is the Wiener filter from scipy, with window size` 3×3 `and variance being the variance of the magnitude of the Fourier transform, that is` $|\mathcal{F}(\hat{\mathbf{K}}_{ZM})|$.

$\hat{\mathbf{K}}_{WF} \coloneqq \texttt{real}(\mathcal{F}^{-1}(\mathcal{F}(\hat{\mathbf{K}}_{ZM}) - \texttt{wiener}(\mathcal{F}(\hat{\mathbf{K}}_{ZM}))))$

return $\hat{\boldsymbol{K}}_{WF}$

$$R_{M_1} = \begin{cases} 1 - e^{-R_w}, & 0 \le R_w \le \alpha, \\ (1 - e^{-\alpha}) e^{\alpha - R_w}, & R_w > \alpha, \\ -1 + e^{R_w}, & -\alpha \le R_w < 0, \\ (-1 + e^{-\alpha}) e^{\alpha + R_w}, & R_w < -\alpha. \end{cases} \tag{16}$$

$$R_{M_2} = \begin{cases} e^{-0.5 R_w^2/\alpha^2}, & R_w \ge 0, \\ -e^{-0.5 R_w^2/\alpha^2}, & R_w < 0. \end{cases} \tag{17}$$

In both models, coefficients close to zero retain values close to their original magnitude, while larger coefficients are progressively suppressed. After applying one of the above mappings to all wavelet coefficients, the enhanced noise residual is obtained via inverse wavelet reconstruction.

These enhancer models are mainly designed for non-flat-field images, where scene structures significantly contaminate the noise residual. For flat-field images, where scene content is negligible, the method provides limited improvement.

4 PRNU Detection

The final objective is to determine whether a suspect image $\mathbf{I}$ was captured by a specific camera. This amounts to testing whether the camera's estimated PRNU pattern $\hat{\mathbf{K}}$ is present in $\mathbf{I}$.

Following the classical formulation by Goljan [13], this task is posed as a binary hypothesis test. Let $\mathbf{R}$ denote the noise residual extracted from $\mathbf{I}$. Under the noise-only hypothesis H_0, the residual consists solely of non-PRNU noise components, modeled collectively as $\mathbf{\Theta}$. Under the PRNU-presence hypothesis H_1, the residual contains both noise and the multiplicative PRNU contribution, yielding

$$\mathrm{H}_0 : \quad \mathbf{R} = \mathbf{\Theta}, \qquad\qquad \mathrm{H}_1 : \quad \mathbf{R} = \mathbf{I}^{(0)}\,\hat{\mathbf{K}} + \mathbf{\Theta},$$

where $\mathbf{I}^{(0)}$ denotes the denoised version of $\mathbf{I}$.

Assuming that $\mathbf{\Theta}$ are Gaussian, independent and identically distributed random variables with unknown variance, then, as detailed in *Appendix*, the optimal detector is the normalized correlation

$$\rho = \mathrm{corr}_{\mathrm{Pearson}}\left(\mathbf{I}^{(0)}\hat{\mathbf{K}}, \mathbf{R}\right). \tag{18}$$

Given a decision threshold ρ_{th}, we decide that the estimated PRNU $\hat{\mathbf{K}}$ is present in the image under inspection $\mathbf{I}$ – and therefore that $\mathbf{I}$ originates from the corresponding camera – if and only if $\rho > \rho_{\mathrm{th}}$. This binary decision framework naturally leads to two types of errors. A *false alarm* occurs when H_0 is true, but we decide H_1, meaning that the camera estimated PRNU is incorrectly declared present in the image. Conversely, a *false rejection* occurs when H_1 is true, but we decide H_0, meaning that the estimated PRNU of the camera is incorrectly declared missing in the image.

In what follows, we review the correlation metrics that have been proposed in the literature as test statistics, namely the Pearson correlation and the Peak-to-Correlation Energy (PCE). Although PRNU detection involves evaluating the correlation between $\mathbf{I}^{(0)}\hat{\mathbf{K}}$ and a noise residual $\mathbf{R}$, it is standard to present these metrics in a more general setting where two PRNU estimates are compared. This removes the modulation by $\mathbf{I}^{(0)}$ and makes the presentation clearer.

4.1 Pearson Correlation

The pioneering work of Lukas et al. [26] adopted the Pearson correlation as the test statistic for PRNU detection. Under the assumption that the noise components $\mathbf{\Theta}$ are Gaussian, independent and identically distributed random variables, this choice is theoretically justified. In fact, in this setting the Pearson correlation coincides with the likelihood-ratio test statistic, which according to the Neyman-Pearson lemma [29], yields the most powerful test.

The Pearson correlation of two PRNU estimates $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{K}}_1$ is defined as the ratio between their covariance and the product of their standard deviations:

$$\mathrm{corr}_{\mathrm{Pearson}} = \frac{\sum_{i=1}^{n}\left(\hat{\mathbf{K}}_0(i) - \overline{\hat{\mathbf{K}}_0}\right)\left(\hat{\mathbf{K}}_1(i) - \overline{\hat{\mathbf{K}}_1}\right)}{\sqrt{\sum_{i=1}^{n}\left(\hat{\mathbf{K}}_0(i) - \overline{\hat{\mathbf{K}}_0}\right)^2}\sqrt{\sum_{i=1}^{n}\left(\hat{\mathbf{K}}_1(i) - \overline{\hat{\mathbf{K}}_1}\right)^2}}, \tag{19}$$

with $\overline{\hat{\mathbf{K}}}$ the mean of $\hat{\mathbf{K}}$ and n the number of pixels of each estimated PRNU. Note that for the computation of the Pearson correlation, both PRNU estimates are reshaped into one-dimensional vectors (using the same ordering of pixels). This is necessary because the Pearson correlation is defined for pairs of 1-D signals.

In our implementation, the Pearson correlation is computed using the `pearsonr` function of the SciPy library.

4.2 Peak-to-Correlation Energy

While the Pearson correlation is the optimal detector under the assumption that the noise components $\mathbf{\Theta}$ are Gaussian, independent and identically distributed random variables with unknown variance, this assumption is rarely satisfied in practice. Modern imaging pipelines introduce shared processing artifacts that appear systematically across different cameras. These shared components introduce a common signal in the estimated PRNUs, increasing their correlation even under H_0, ultimately triggering false alarms. The Peak-to-Correlation Energy (PCE), introduced by Goljan [13], was specifically designed to be robust against such spurious periodic correlations.

The PCE is defined as the ratio between the squared correlation peak – located at zero shift – and the average squared correlation computed over all non-zero shifts. For simplicity, here it is assumed that each estimated PRNU has zero mean, which can be enforced by subtracting its empirical average from every pixel. Formally, for two estimated PRNUs $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{K}}_1$, the PCE is

$$\mathrm{PCE}(\hat{\mathbf{K}}_0, \hat{\mathbf{K}}_1) = \frac{\mathbf{C}^2_{\hat{\mathbf{K}}_0,\hat{\mathbf{K}}_1}(0,0)}{\frac{1}{MN-|\mathbf{A}|}\sum\limits_{(x,y)\notin\mathbf{A}} \mathbf{C}^2_{\hat{\mathbf{K}}_0,\hat{\mathbf{K}}_1}(x,y)}, \tag{20}$$

where M and N are the height and width of each estimated PRNU, $\mathbf{A}$ is a small exclusion region around $(0,\,0)$, and $\mathbf{C}^2_{\hat{\mathbf{K}}_0,\hat{\mathbf{K}}_1}(x,y)$ the unnormalized circular cross-correlation defined as

$$\mathbf{C}_{\hat{\mathbf{K}}_0,\hat{\mathbf{K}}_1}(x,y) = \frac{1}{MN}\sum_{(m,n)} \hat{\mathbf{K}}_0(m,n)\hat{\mathbf{K}}_1(m+x \bmod M, n+y \bmod N). \tag{21}$$

Note that $\mathbf{C}_{\hat{\mathbf{K}}_0,\hat{\mathbf{K}}_1}(x,y)$ corresponds to the Pearson correlation between $\hat{\mathbf{K}}_0$ and a (x,y)-shifted version of $\hat{\mathbf{K}}_1$ but without the normalization by the standard deviations. This omission is intentional: since the PCE is a ratio of these quantities, the normalization would cancel out and therefore does not need to be computed explicitly.

The PCE normalizes the squared correlation peak by the average squared correlation obtained at all shifted positions outside a small neighborhood around the origin. As a result, the PCE becomes large only when the correlation at perfect alignment stands out significantly above the correlation observed at non-aligned shifts. Conversely, periodic or structured artifacts tend to produce elevated correlations at multiple shifts, reducing the overall PCE and avoiding false alarms.

Algorithm 9: Circular cross-correlation

function *circular_cross-correlation($\hat{\boldsymbol{K}}_0$, $\hat{\boldsymbol{K}}_1$)*
- **Input $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{K}}_1$:** zero-mean PRNU estimates obtained using Algorithm 7
- **Output $\mathbf{corr_{CC}}$:** circular cross-correlation matrix between $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{K}}_1$
- $\hat{\mathbf{K}}_1 \coloneqq \texttt{horizontal\_and\_vertical\_flip}(\hat{\mathbf{K}}_1)$
- $\mathbf{corr_{CC}} \coloneqq \texttt{real}(\mathcal{F}^{-1}(\mathcal{F}(\hat{\mathbf{K}}_0)\mathcal{F}(\hat{\mathbf{K}}_1)))$
- **return $\mathbf{corr_{CC}}$**

Algorithm 9 implements the circular cross-correlation relying on the convolution theorem. Since the Fourier transform of a convolution equals the pointwise product of the Fourier transforms, and since the cross-correlation of two real 2D signals is equivalent to the convolution of one signal with a flipped version of the other, circular cross-correlation can be computed efficiently in the frequency domain. With this formulation, $\mathrm{corr}_{\mathrm{CC}}[-1,-1]$ is the maximum of $\mathrm{corr}_{\mathrm{CC}}$. This fact is used in Algorithm 10, which implements the computations of the Peak-to-Correlation energy. It computes

Algorithm 10: Peak-to-Correlation Energy

function *peak_to_correlation_energy*($\mathbf{corr_{CC}}, \mathcal{A}_{sz}$)

Input $\mathbf{corr_{CC}}$**:** circular cross-correlation matrix of size X and Y between $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{K}}_1$

Input $\mathcal{A}_{sz}$**:** size of square around (0, 0) to set to 0 (default is 5)

Output PCE**:** Peak-to-Correlation Energy ratio between $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{K}}_1$

$\mathbf{corr_{CC}}_{-1,-1} \coloneqq \mathbf{corr_{CC}}[-1, -1]$

// Set to 0 a square of size $\mathcal{A}_{sz} \times \mathcal{A}_{sz}$ around (0, 0) in $\mathbf{corr_{CC}}$

$\mathbf{corr_{CC}}\left[-\frac{\mathcal{A}_{sz}}{2} : \frac{\mathcal{A}_{sz}}{2}, -\frac{\mathcal{A}_{sz}}{2} : \frac{\mathcal{A}_{sz}}{2}\right] \coloneqq 0$

$\text{PCE} \coloneqq \frac{\mathbf{corr_{CC}}^2_{-1,-1}}{\frac{1}{X*Y-\mathcal{A}_{sz}^2} * \sum \mathbf{corr_{CC}}^2}$

return PCE

the PCE by contrasting this peak with the average correlation energy at all non-zero shifts outside the exclusion region $\mathbf{A}$, chosen as a square centered at $(0, 0)$ of size 5×5 by default.

While the PCE captures the strength of the correlation peak relative to its shifted versions, it discards the sign of the correlation. This is undesirable in PRNU analysis, since a valid PRNU match should yield a positive correlation peak. To address this, the Signed PCE (sPCE) is defined by restoring the sign of the zero-shift correlation peak into the PCE value. In practice, the sPCE simply multiplies the PCE by the sign of the peak correlation, as described in Algorithm 11.

Algorithm 11: Signed Peak-to-Correlation Energy

function *signed_peak_to_correlation_energy*($\mathbf{corr_{CC}}$)

Input $\mathbf{corr_{CC}}$**:** circular cross-correlation matrix between $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{K}}_1$

Output sPCE**:** signed Peak-to-Correlation Energy ratio between $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{K}}_1$

$\text{sPCE} \coloneqq \texttt{sign}(\mathbf{corr_{CC}}[-1, -1])\text{PCE}(\mathbf{corr_{CC}})$

return sPCE

5 Experiments

In this section, we evaluate the presented methods on a controlled and reproducible experimental setup. We describe the dataset preparation, the testing protocol, the evaluation metrics, and the baseline configuration against which variants are compared. Finally, we report quantitative results and analyze the impact of each modification to the baseline.

Dataset. We use the Dresden dataset [11][2], including 8,650 photos from 33 camera instances representing 14 camera models from 10 brands, and follow the protocol of Al-Ani et al. [1]. For each camera instance, we extract a centered 768×768 pixel region from every available image. Each such crop is then uniformly partitioned into a 12×12 grid of non-overlapping patches of size 64×64 pixels. In the remainder of this section, we refer to these 64×64 patches simply as *crops*. Note that, even if the crops come from the same camera, crops taken from different spatial locations contain a different PRNU and therefore are considered different instances.

[2] https://www.kaggle.com/datasets/micscodes/dresden-image-database

Testing protocol. The PRNU is estimated using the residuals from L crops, considering only the green color channel. Among the various color channels, the green one has a higher density of sensors, hence in demosaicked photos the interpolation for this color channel is minimized. Whenever specified, post-processing is also applied. For the matching test, we consider crops from the same camera instance and the same crop location. For the mismatching test, we consider crops from the same camera instance but different crop locations. Note that pairs from different camera instances are not included in this mismatching category.

Evaluation metrics. For each camera instance, we compute two ROC-based metrics: $\mathcal{P}$ and $\mathcal{R}$.

- $\mathcal{P}$: This metric reports the true positive rate when the false positive rate (FPR), i.e. the false alarm rate, is fixed to 10^{-3}. The higher, the better, since a large $\mathcal{P}$ means the method correctly identifies matching pairs even under a very stringent false alarm constraint.
- $\mathcal{R}$: This metric reports the equal error rate, i.e., the operating point where FPR equals the false negative rate (FNR). The lower, the better as a lower $\mathcal{R}$ indicates better separation between intra- and interclass correlations.

To express improvements relative to the baseline configuration, we introduce

$$\grave{\mathcal{P}} = \mathcal{P} - \mathcal{P}_{\texttt{baseline}}, \qquad \grave{\mathcal{R}} = \mathcal{R}_{\texttt{baseline}} - \mathcal{R}. \tag{22}$$

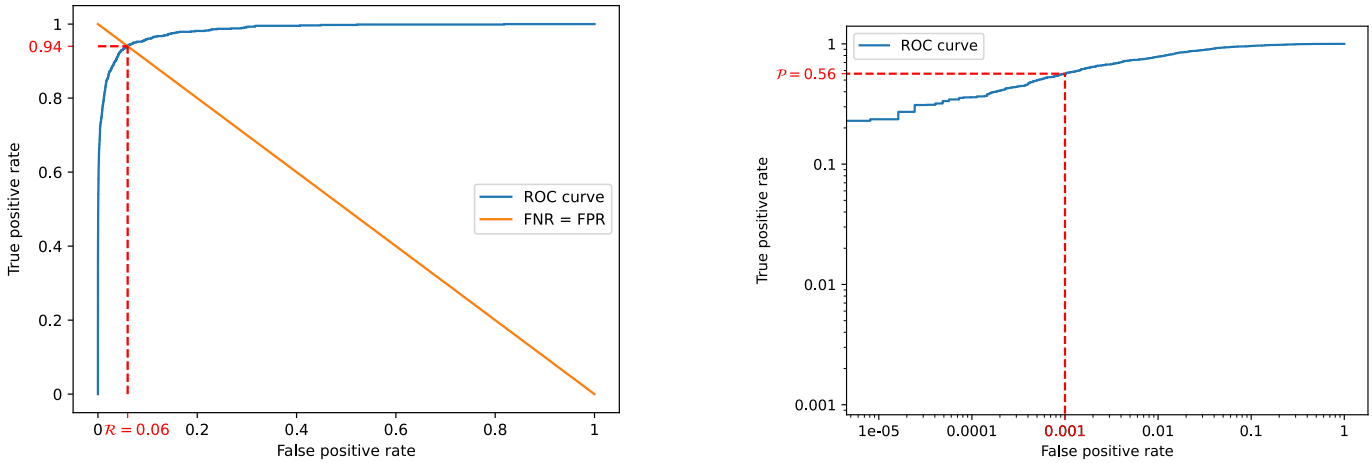
Since $\mathcal{P}$ and $\mathcal{R}$ are reported as percentages, the quantities $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ are expressed in percentage points (%pt).

ROC-based metrics should be interpreted primarily as indicators of separability: they measure how well a method can separate positives from negatives, regardless of the specific decision threshold. However, practical forensic systems must ultimately choose a decision threshold. In practice, a method is more reliable when the threshold inferred from representative data remains stable across operational scenarios. This stability requirement is not captured directly by ROC curves but is essential for real-world deployment. Therefore, while we report ROC-based metrics for comparability, they should be read with caution: high separability does not automatically guarantee stable or operationally robust performance.

In all experiments we report $\bar{\mathcal{P}}$, the average of $\mathcal{P}$ across all camera instances, and $\bar{\mathcal{R}}$ the average of $\mathcal{R}$ across all camera instances. In addition, whenever available, we also report the average results from Al-Ani et al. [1]. Small differences in our results with respect to those are explained by the fact that the mentioned article considers, in addition to the Dresden dataset, photos from five camera models that were unavailable to us.

Baseline. Unless stated otherwise, our baseline uses $L = 50$, the wavelet denoiser of Section 3.1.1, the basic averaging scheme of Section 3.2.1, and the Pearson correlation of Section 4.1. Figure 6 shows an example ROC curve obtained under this configuration. Our baseline results (Table 1) achieve $\bar{\mathcal{P}} = 74.9\%$ and $\bar{\mathcal{R}} = 4.8\%$. Although each camera model contributes at least 200 images, providing statistically meaningful estimates, we observe significant performance variability across camera models: some cameras are perfectly identified ($\mathcal{P} = 100\%$, $\mathcal{R} = 0\%$), while others suffer substantial degradation (e.g., $\mathcal{P} = 27.4\%$, $\mathcal{R} = 18.3\%$ for the Fujifilm FinePix J50).

Evaluated variants. To assess the limitations of the baseline and identify potential improvements, we evaluate variants of each processing step and parameter. Concerning the PRNU extraction, Section 5.1.1, focusing on noise residuals extraction, experimentally assesses the influence of the



(a) Linear scale.

(b) Logarithmic scale.

Figure 6: ROC curve for the baseline experiment with the first Praktica DCZ 5.9 camera instance.

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
Baseline	$\mathcal{P}$	47.6	100	61.8	56.8	94.2	94.4	100	99.3	99.4	100	100	27.4	81.4	78.4	**74.9**	66.7
	$\mathcal{R}$	8.5	0	9.5	5.8	1.3	1.2	0	0.4	0.2	0	0	18.3	3.1	2.6	**4.8**	7.6
	AUC	0.97	1	0.96	0.98	1	1	1	1	1	1	1	0.90	0.99	1	0.98	N/A

Table 1: Performances $\mathcal{P}$ and $\mathcal{R}$ in % for the baseline. AUC denotes the area under the ROC curve.

estimated noise standard deviation, the presence of saturated pixels, and the block-matching and 3D denoiser. Section 5.1.2, considering the combination of noise residuals, empirically evaluates the impact of the number of photos and the resolution of crops, and the effect of the phase-only operation and of the Maximum Likelihood Estimator. Section 5.1.3 experimentally assesses the Zero-mean and Wiener filtering PRNU post-processing. Concerning the PRNU detection, Section 5.2 empirically evaluates the Peak-to-Correlation Energy.

Note that Section 5.1 does not consider the denoiser for flat-field photos described in Section 3.1.3. Unlike the other PRNU extraction and detection operations, this noise residual extraction method requires flat-field images of the same scene. Consequently, we could not evaluate it, as the Dresden dataset does not contain such images.

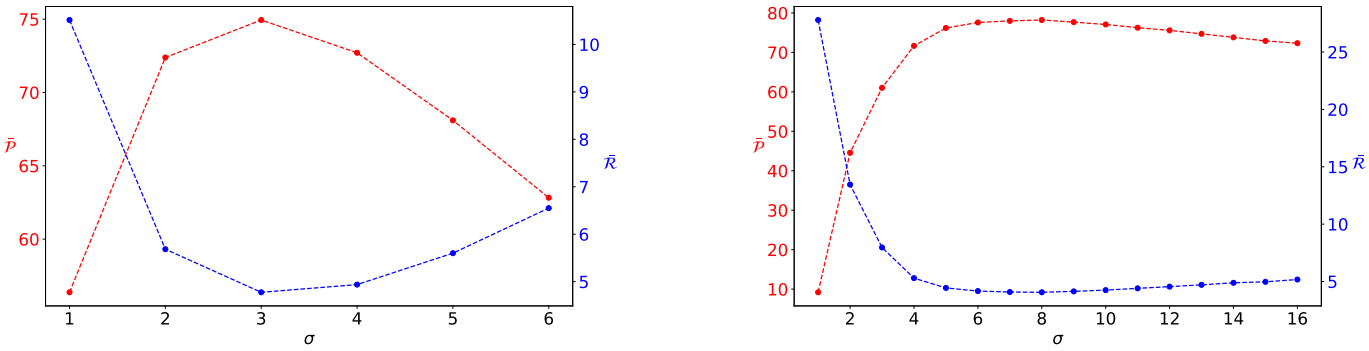
5.1 PRNU Extraction

5.1.1 Noise Residuals Extraction

Influence of σ on PRNU extraction. σ is the estimated noise standard deviation used in the denoisers. If the estimated noise standard deviation is lower than the actual one, then not enough noise is removed from the image. Conversely, if the estimated noise standard deviation is higher than the actual one, then all the noise and some scene content is removed from the image. In the following we consider a fixed σ, independently of the considered images.

Figure 7a shows that the best performance is obtained for $\sigma = 3$. Hence, we consider $\sigma = 3$ for

the wavelet denoiser in the following. While Al-Ani et al. [1] mention that, for the wavelet denoiser, setting σ between 2 and 5 has little impact on the performances, Figure 7a shows that in this interval $\bar{\mathcal{P}}$ varies from 68.1% to 74.9% and $\bar{\mathcal{R}}$ varies from 4.8% to 5.6%.



(a) Wavelet denoiser performances versus σ.

(b) BM3D denoiser performances versus σ.

Figure 7: Performances of denoisers versus σ.

The performance of the wavelet denoiser as a function of σ is explained by the fact that σ denotes the global noise standard deviation. Hence, the performance peaks around an optimal σ specific to the data and the denoiser; the further the considered σ shifts from this value, the worse the performance becomes. Note that the optimal σ does not correspond to the actual global noise standard deviation. Instead, this optimal value emerges because the subsequent steps – noise residual combination, PRNU post-processing, and PRNU detection – leverage noise artifacts specific to the chosen denoiser.

Influence of saturated pixels in the extraction of the PRNU. Pixels reaching saturation have to be considered with caution. For such pixels there is an ambiguity whether the PRNU is present. These pixels may be white because of the associated camera pixel sensors reaching saturation or because of camera high dynamic range (HDR) processing. In the considered image crops, 0.5% of pixels are white. To avoid such ambiguity, a possibility is to not consider saturated pixels, that is, noise residual pixels associated to saturated pixels are forced to be zero. However, Table 2 shows that when considering unsaturated pixels the performances of $\mathcal{P}$ and $\mathcal{R}$ vary by at most 0.4 %pt across the considered camera models. Note that these performance changes are small, possibly because of the small portion of saturated pixels. As there is no change in average, for the sake of simplicity we have not excluded saturated pixels in the remaining of this article.

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
Unsaturated	$\mathcal{P}$	47.7	100	62	56.8	94.2	94.3	100	99.3	99.4	100	100	27	81.3	78.2	74.9	N/A
	$\grave{\mathcal{P}}$	0.1	0	0.2	0	0	-0.1	0	0	0	0	0	-0.4	-0.1	-0.2	0	N/A
	$\mathcal{R}$	8.4	0	9.5	5.9	1.3	1.3	0	0.4	0.2	0	0	18.3	3.1	2.7	4.8	N/A
	$\grave{\mathcal{R}}$	0.1	0	0	-0.1	0	-0.1	0	0	0	0	0	0	0	-0.1	0	N/A

Table 2: Performances $\mathcal{P}$ and $\mathcal{R}$ in % and $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ in %pt when considering unsaturated pixels.

Block-matching and 3D (BM3D). Instead of the wavelet denoiser, here we consider the block-matching and 3D (BM3D) denoiser. Similarly to the analysis of the performance of the wavelet denoiser as a function of σ above, Figure 7b shows that the best performance is achieved for $\sigma = 8$, therefore we used this value in our experiment.

Concerning the computation time, note that to denoise the 144 64×64 crops of each photo in parallel on a 32-core AMD EPYC 9354 processor, BM3D takes about 10 seconds, while the wavelet-based denoiser takes about 5 ms. Despite being more computationally demanding than the wavelet-based denoiser, BM3D outperforms the baseline denoiser by reaching $\bar{\mathcal{P}} = 78.2\%$ and $\bar{\mathcal{R}} = 4.1\%$ that is $\grave{\mathcal{P}} = 3.3\%$pt and $\grave{\mathcal{R}} = 0.7\%$pt (see Table 3).

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
BM3D	$\mathcal{P}$	67.2	99.8	62.6	60.2	98.5	93.2	100	99.6	100	100	99.9	38.5	72.8	84.3	78.2	**82.4**
	$\grave{\mathcal{P}}$	19.6	-0.2	0.8	3.4	4.3	-1.2	0	0.3	0.6	0	-0.1	11.1	-8.6	5.9	3.3	15.7
	$\mathcal{R}$	4.8	0.2	10	5.9	0.5	1.4	0	0.3	0	0	0.1	14	4.4	2.4	**4.1**	5.1
	$\grave{\mathcal{R}}$	3.7	-0.2	-0.5	-0.1	0.8	-0.2	0	0.1	0.2	0	-0.1	4.3	-1.3	0.2	0.7	2.5

Table 3: Performances $\mathcal{P}$ and $\mathcal{R}$ in % and $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ in %pt for block-matching and 3D.

5.1.2 Combining Noise Residuals

Influence of the number of photos considered to extract the PRNU. Based on experiments with $L = 100$ and $L = 150$, summarized in Table 4, we observe that, as expected, the performance improves as the number of considered crops increases. Indeed, for almost all camera models, the method is able to leverage the additional crops to achieve perfect classification.

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
$L = 100$	$\mathcal{P}$	94.8	100	89.1	99.4	100	100	100	100	100	100	100	82.6	99.8	100	97	N/A
	$\grave{\mathcal{P}}$	47.2	0	27.3	42.6	5.8	5.6	0	0.7	0.6	0	0	55.2	18.4	21.6	22.1	N/A
	$\mathcal{R}$	1	0	1.3	0.2	0	0	0	0	0	0	0	3.2	0.1	0	0.5	N/A
	$\grave{\mathcal{R}}$	7.5	0	8.2	5.6	1.3	1.2	0	0.4	0.2	0	0	15.1	3	2.6	4.3	N/A
$L = 150$	$\mathcal{P}$	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	N/A
	$\grave{\mathcal{P}}$	52.4	0	38.2	43.2	5.8	5.6	0	0.7	0.6	0	0	72.6	18.6	21.6	25.1	N/A
	$\mathcal{R}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	N/A
	$\grave{\mathcal{R}}$	8.5	0	9.5	5.8	1.3	1.2	0	0.4	0.2	0	0	18.3	3.1	2.6	4.8	N/A

Table 4: Performances $\mathcal{P}$ and $\mathcal{R}$ in % and $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ in %pt for various numbers of crops.

Influence of the resolution of the crops in the extraction of the PRNU. Based on experiments with 32×32 and 128×128 crops, summarized in Table 5, we observe that, as expected, the performance degrades as the size of the crop decreases. Unlike the trend observed with the number of crops, we notice that for certain models, such as the Panasonic DMC-FZ50, the method is unable to leverage the larger crop resolution to achieve perfect classification. This occurs despite this model having a baseline performance similar to the Praktica DCZ 5.9, which does not face this limitation.

Table 5 also shows reproduced experiments from Al-Ani et al. [1] with $L = 100$ and $L = 150$[3]. These hybrid experiments do not independently evaluate the dependence on the number of crops and their sizes, unlike the experiments discussed in the previous two sections. These hybrid experiments yield similar insights to our independent approach by avoiding performance saturation due to the higher number of crops by considering lower resolution crops.

Table 5 illustrates how much more effectively the PRNU is estimated with more crops than with higher resolution crops. With 4 times more pixels, a single crop of 960×960 instead of 50 crops of 64×64, reaches $\bar{\mathcal{P}} = 21.2\%$ and $\bar{\mathcal{R}} = 25.9\%$, that is a $\grave{\mathcal{P}}$ = -53.7%pt and $\grave{\mathcal{R}}$ = -21.1%pt. In other words, the estimated PRNU of a single high resolution crop can still be leveraged for forensic tasks with a false positive rate of 10^{-3}. However, having only a single small 64×64 crop leads to ineffective performances with $\bar{\mathcal{P}} = 0.2\%$ and $\bar{\mathcal{R}} = 47.2\%$ for this false positive rate.

Phase-only operation. Our experiment, using the phase-only operation instead of the basic averaging, does not support the claims by Al-Ani et al. [1]. Indeed, as shown in Table 6, we obtained $\bar{\mathcal{P}} = 66\%$, that is a 8.9%pt $\bar{\mathcal{P}}$ decrease from the baseline, and $\bar{\mathcal{R}} = 6.4\%$, just a 1.6%pt $\bar{\mathcal{R}}$ increase.

Maximum Likelihood Estimator (MLE). Combining noise residuals with the maximum likelihood estimator, instead of the basic averaging, outperforms the baseline configuration, as can be seen in Table 7. We achieve $\bar{\mathcal{P}} = 80.7\%$ and $\bar{\mathcal{R}} = 4.2\%$, that is $\grave{\mathcal{P}} = 5.8\%$pt and $\grave{\mathcal{R}} = 0.6\%$pt.

5.1.3 PRNU Post-processing

Zero-mean and Wiener filtering. Adding the straightforward zero-mean PRNU post-processing improves performance relative to the baseline, as shown in Table 8. We achieve $\bar{\mathcal{P}} = 79.6\%$ and $\bar{\mathcal{R}} = 3.7\%$, that is $\grave{\mathcal{P}} = 4.7\%$pt and $\grave{\mathcal{R}} = 1.1\%$pt.

If we also apply the Wiener filter in the Fourier domain after the zero-mean operation, compared to only applying the zero-mean post-processing, we improve the performance and achieve $\bar{\mathcal{P}} = 80.4\%$, that is a 5.5%pt decrease in $\bar{\mathcal{P}}$, and $\bar{\mathcal{R}} = 3.4\%$, which is a 1.4%pt increase in $\bar{\mathcal{R}}$.

5.2 PRNU Detection

Peak-to-Correlation Energy. Computing the correlation with the Signed Peak-to-Correlation Energy (sPCE), instead of the Pearson correlation, does not outperform the baseline for $\sigma = 3$, as shown in Table 9. We achieve $\bar{\mathcal{P}} = 71.8\%$ and $\bar{\mathcal{R}} = 4.8\%$, that is $\grave{\mathcal{P}} = -3.1\%$pt and $\grave{\mathcal{R}} = 0\%$pt. For $\sigma = 5$, sPCE outperforms the baseline correlation with $\grave{\mathcal{P}} = 5\%$pt and $\grave{\mathcal{R}} = 0.6\%$pt. For $\sigma = 3$ the performance decrease is probably due to the fact that σ has been optimized for the baseline correlation, not for sPCE. Considering the best σ values for sPCE and the Pearson correlation, Figure 8 shows that sPCE achieves for $\sigma = 4$ a lower $\bar{\mathcal{R}}$ than the Pearson correlation, and the Pearson correlation achieves for $\sigma = 3$ a higher $\bar{\mathcal{P}}$ than sPCE. In consequence, there is a trade-off between $\bar{\mathcal{P}}$ and $\bar{\mathcal{R}}$ depending on whether the correlation metric used is the Pearson correlation or sPCE. Table 9 also shows that, as expected, sPCE and PCE lead to almost identical performances.

[3]In both cases, only camera instances with more than 300 photos are considered.

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
$L = 50$ 32×32 crops	$\mathcal{P}$	5.3	76.5	20.8	23.3	30.2	49.6	88.1	62.6	60.9	84.7	88.9	7	25.8	19	34.2	N/A
	$\grave{\mathcal{P}}$	-42.3	-23.5	-41	-33.5	-64	-44.8	-11.9	-36.7	-38.5	-15.3	-11.1	-20.4	-55.6	-59.4	-40.7	N/A
	$\mathcal{R}$	25.8	3.1	18.2	11.6	11.3	9	2.1	6.2	4.4	2.1	1.6	26.6	15.7	15.3	14	N/A
	$\grave{\mathcal{R}}$	-17.3	-3.1	-8.7	-5.8	-10	-7.8	-2.1	-5.8	-4.2	-2.1	-1.6	-8.3	-12.6	-12.7	-9.2	N/A
$L = 50$ 128×128 crops	$\mathcal{P}$	96.4	100	65.9	69.7	100	99.9	100	100	100	100	100	41.6	99.9	99.8	89.4	N/A
	$\grave{\mathcal{P}}$	68.9	0.1	27.3	6.2	17.6	7	0	2.1	1.6	0	0	12	25	38.1	21.3	N/A
	$\mathcal{R}$	0.7	0	9.4	5.6	0	0.1	0	0	0	0	0	17.2	0.1	0.1	2.8	N/A
	$\grave{\mathcal{R}}$	11.7	0.1	4.4	-1.4	3.1	1.3	0	0.6	0.4	0	0	-2.1	4.1	4.7	2.8	N/A
$L = 100^4$ 32×32 crops	$\mathcal{P}$	N/A	N/A	67.4	78.5	N/A	N/A	100	N/A	97.9	N/A	N/A	N/A	72	N/A	**77.1**	69.1
	$\grave{\mathcal{P}}$	N/A	N/A	5.6	21.7	N/A	N/A	0	N/A	-1.5	N/A	N/A	N/A	-9.4	N/A	2.2	2.4
	$\mathcal{R}$	N/A	N/A	4.7	3	N/A	N/A	0	N/A	0.3	N/A	N/A	N/A	4.2	N/A	**3.4**	6.5
	$\grave{\mathcal{R}}$	N/A	N/A	4.8	2.8	N/A	N/A	0	N/A	-0.1	N/A	N/A	N/A	-1.1	N/A	**1.4**	1.1
$L = 150^4$ 32×32 crops	$\mathcal{P}$	N/A	N/A	92.7	100	N/A	N/A	100	N/A	100	N/A	N/A	N/A	92.3	N/A	**94.7**	79
	$\grave{\mathcal{P}}$	N/A	N/A	30.9	43.2	N/A	N/A	0	N/A	0.6	N/A	N/A	N/A	10.9	N/A	**19.8**	12.3
	$\mathcal{R}$	N/A	N/A	0.7	0	N/A	N/A	0	N/A	0	N/A	N/A	N/A	1.1	N/A	**0.7**	4.8
	$\grave{\mathcal{R}}$	N/A	N/A	8.8	5.8	N/A	N/A	0	N/A	0.2	N/A	N/A	N/A	2	N/A	**4.1**	2.8
$L = 1$ 64×64 crops	$\mathcal{P}$	0.2	0.3	0.2	0.2	0.3	0.3	0.3	0.2	0.3	0.3	0.3	0.2	0.2	0.2	0.2	N/A
	$\grave{\mathcal{P}}$	-47.4	-99.7	-61.6	-56.6	-93.9	-94.1	-99.7	-99.1	-99.1	-99.7	-99.7	-27.2	-81.2	-78.2	-74.7	N/A
	$\mathcal{R}$	48.7	45.6	47.5	47.1	47.5	46.5	45	46	45.5	45.2	44.6	48.4	48.1	47.6	47.2	N/A
	$\grave{\mathcal{R}}$	-40.2	-45.6	-38	-41.3	-46.2	-45.3	-45	-45.6	-45.3	-45.2	-44.6	-30.1	-45	-45	-42.4	N/A
$L = 1^5$	$\mathcal{P}$	12.8	57.6	13.7	1.3	37.6	30	48.9	53	31.2	14.2	12.1	4.8	24.8	15.5	21.2	N/A
	$\grave{\mathcal{P}}$	-34.8	-42.4	-48.1	-55.5	-56.6	-64.4	-51.1	-46.3	-68.2	-85.8	-87.9	-22.6	-56.6	-62.9	-53.7	N/A
	$\mathcal{R}$	28	11	29.3	36.9	14.6	21.4	15	16	13	22.8	22.3	44.1	25	28.6	25.9	N/A
	$\grave{\mathcal{R}}$	-19.5	-11	-19.8	-31.1	-13.3	-20.2	-15	-15.6	-12.8	-22.8	-22.3	-25.8	-21.9	-26	-21.1	N/A

[4] Result from [1]. Only camera instances with more than 300 photos are considered.
[5] Only 80 photos by camera instance are considered and 2×2, instead of 12×12, 960×960 crops.

Table 5: Performances $\mathcal{P}$ and $\mathcal{R}$ in % and $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ in %pt for various resolutions of crops.

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
Phase-only	$\mathcal{P}$	24.7	99.7	73.8	62.2	93.3	89	99.9	93.6	95	99.9	99.9	25	47.5	66	66	69.2
	$\grave{\mathcal{P}}$	-22.9	-0.3	12	5.4	-0.9	-5.4	-0.1	-5.7	-4.4	-0.1	-0.1	-2.4	-33.9	-12.4	-8.9	2.5
	$\mathcal{R}$	13.8	0.1	6.8	4.7	1.3	2.3	0.1	1.2	1	0.1	0.1	18.5	9	4.6	6.4	7.59
	$\grave{\mathcal{R}}$	-5.3	-0.1	2.7	1.1	0	-1.1	-0.1	-0.8	-0.8	-0.1	-0.1	-0.2	-5.9	-2	-1.6	0.01

Table 6: Performances $\mathcal{P}$ and $\mathcal{R}$ in % and $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ in %pt for phase-only operation.

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
MLE	$\mathcal{P}$	59.9	100	58.3	70.1	98.8	99.2	100	99.9	97	100	100	26.3	91.7	89.6	**80.7**	70.6
	$\grave{\mathcal{P}}$	12.3	0	-3.5	13.3	4.6	4.8	0	0.6	-2.4	0	0	-1.1	10.3	11.2	**5.8**	3.9
	$\mathcal{R}$	7	0	10.6	4.7	0.4	0.4	0	0.1	0.7	0	0	19.5	1.7	1.7	**4.2**	6.2
	$\grave{\mathcal{R}}$	1.5	0	-1.1	1.1	0.9	0.8	0	0.3	-0.5	0	0	-1.2	1.4	0.9	0.6	1.4

Table 7: Performances $\mathcal{P}$ and $\mathcal{R}$ in % and $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ in %pt for maximum likelihood estimator.

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
Zero-mean	$\mathcal{P}$	50.7	100	81.4	70.2	97.1	99.7	100	99	98.1	100	100	34.8	81.4	83.8	79.6	N/A
	$\grave{\mathcal{P}}$	3.1	0	19.6	13.4	2.9	5.3	0	-0.3	-1.3	0	0	7.4	0	5.4	4.7	N/A
	$\mathcal{R}$	8.4	0.1	4	4.1	0.8	0.2	0	0.4	0.6	0	0	14	3.2	2.3	3.7	N/A
	$\grave{\mathcal{R}}$	0.1	-0.1	5.5	1.7	0.5	1	0	0	-0.4	0	0	4.3	-0.1	0.3	1.1	N/A
ZM and WF	$\mathcal{P}$	49.7	100	79.8	69.9	97	99.7	100	99	99.5	100	100	46.6	80.8	84.2	**80.4**	73.3
	$\grave{\mathcal{P}}$	2.1	0	18	13.1	2.8	5.3	0	-0.3	0.1	0	0	19.2	-0.6	5.8	5.5	6.6
	$\mathcal{R}$	8.5	0.1	4.4	4.1	0.7	0.2	0	0.4	0.2	0	0	10.6	3.2	2.2	**3.4**	6.7
	$\grave{\mathcal{R}}$	0	-0.1	5.1	1.7	0.6	1	0	0	0	0	0	7.7	-0.1	0.4	1.4	0.9

Table 8: Performances $\mathcal{P}$ and $\mathcal{R}$ in % and $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ in %pt for zero-mean and Wiener filter.

	Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
	Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
PCE	$\mathcal{P}$	41.4	100	67.9	63.5	91.8	96.5	100	99.2	98.2	77	68.1	26.3	81.2	72.8	71.8	N/A
	$\grave{\mathcal{P}}$	-6.2	0	6.1	6.7	-2.4	2.1	0	-0.1	-1.2	-23	-31.9	-1.1	-0.2	-5.6	-3.1	N/A
	$\mathcal{R}$	9.2	0.1	5.5	4.4	1	0.6	0	0.4	0.5	2.5	3.2	18.7	3.5	3.5	4.8	N/A
	$\grave{\mathcal{R}}$	-0.7	-0.1	4	1.4	0.3	0.6	0	0	-0.3	-2.5	-3.2	-0.4	-0.4	-0.9	0	N/A
sPCE	$\mathcal{P}$	41.4	100	67.9	63.5	92.6	96.5	100	99.2	98.2	77	68.1	26.3	82.4	72.8	72.1	N/A
	$\grave{\mathcal{P}}$	-6.2	0	6.1	6.7	-1.6	2.1	0	-0.1	-1.2	-23	-31.9	-1.1	1	-5.6	-2.8	N/A
	$\mathcal{R}$	9.2	0.1	5.5	4.4	0.9	0.6	0	0.4	0.5	2.5	3.2	18.7	3	3.5	4.8	N/A
	$\grave{\mathcal{R}}$	-0.7	-0.1	4	1.4	0.4	0.6	0	0	-0.3	-2.5	-3.2	-0.4	0.1	-0.9	0	N/A

Table 9: Performances $\mathcal{P}$ and $\mathcal{R}$ in % and $\grave{\mathcal{P}}$ and $\grave{\mathcal{R}}$ in %pt for PCE and sPCE.

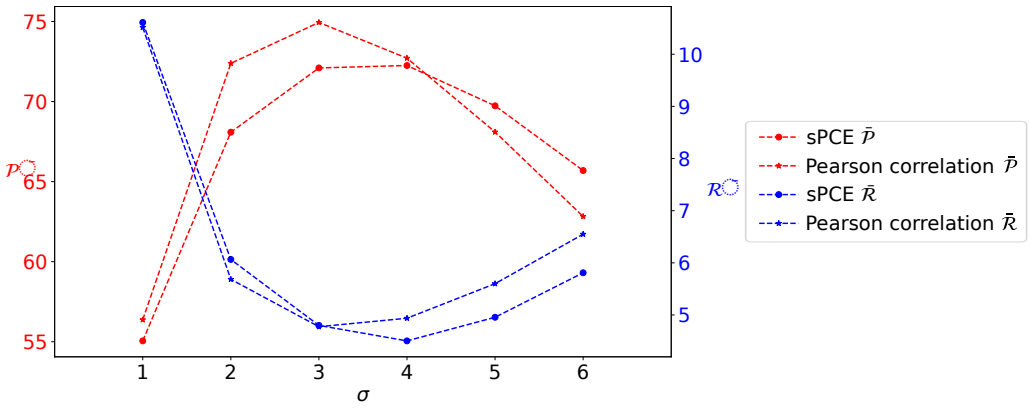
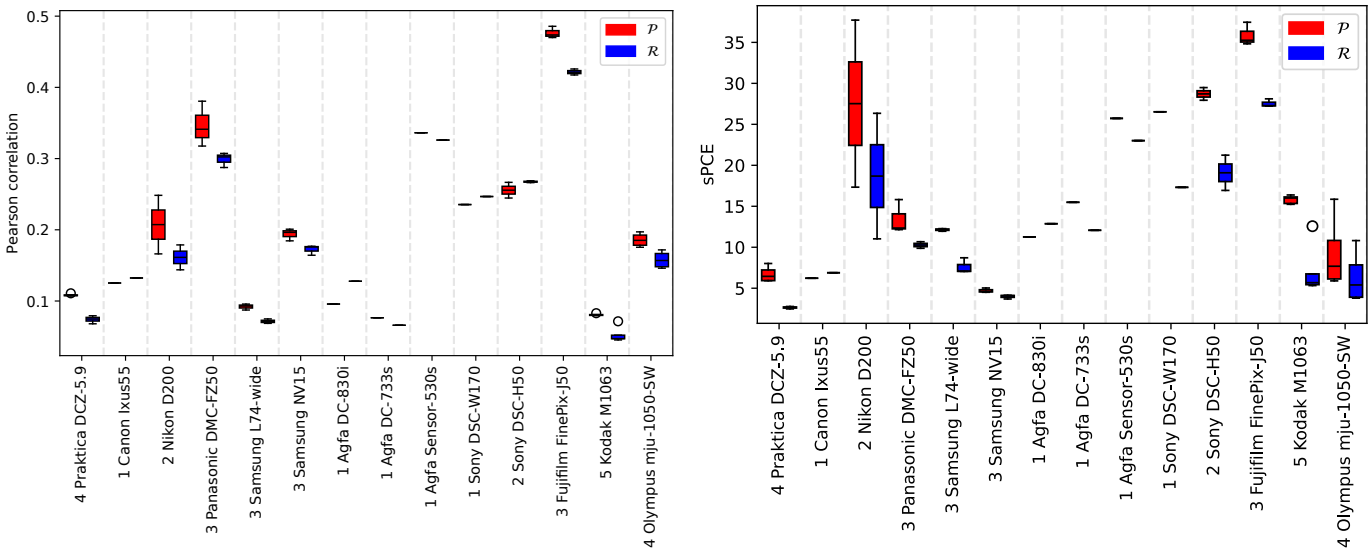


Figure 8: sPCE and Pearson correlation performances versus σ.

Decision threshold stability. Figure 9 completes the discussion in Section 5 by showing the Pearson correlation and sPCE thresholds inferred from the $\mathcal{P}$ and $\mathcal{R}$ computations for the baseline. In other words, the $\mathcal{P}$ and $\mathcal{R}$ correlation thresholds are obtained, respectively, for a false positive rate of 10^{-3}, and for the false positive rate being equal to the false negative rate. For camera models with multiple instances, we notice that the correlation thresholds are significantly stable, except for the Nikon D200 and the Panasonic DMC-FZ50 for the Pearson correlation, and the Nikon D200 and the Olympus mju-1050-SW for the sPCE. To ensure a false positive rate of 10^{-3} for the evaluated cameras, an sPCE threshold of 38 can be used. As a comparison, to guarantee a false positive below 10^{-5}, we found an experimental sPCE threshold of 60. This aligns with the threshold of 60 reported by Goljan et al. [14] for a false positive rate of 10^{-6}.

As shown in Table 10, substantial true positive rates are maintained even when strictly disallowing false alarms (i.e., enforcing a false positive rate below 10^{-5}). Under these conditions, we observe an average of $\bar{\mathcal{P}} = 45.7\%$, that is $\grave{\mathcal{P}} = -26.1\%$pt. The reference to compute this $\grave{\mathcal{P}}$ is sPCE and a false positive rate fixed to 10^{-3}.

Following Section 5.1.2, let us evaluate, for a given PRNU and random noise zero-mean Gaussian distributions, the effect of the number of pixels, either by changing the crop size or varying the number of crops. In the first case, Pearson, PCE and sPCE give, on average, the same correlation when



(a) Pearson correlation thresholds inferred from $\mathcal{P}$ and $\mathcal{R}$ computations.

(b) sPCE thresholds inferred from $\mathcal{P}$ and $\mathcal{R}$ computations.

Figure 9: Pearson correlation and sPCE thresholds inferred from $\mathcal{P}$ and $\mathcal{R}$ computations.

Brand	Praktica	Canon	Nikon	Panasonic	Samsung		Agfa			Sony		Fujifilm	Kodak	Olympus	Average	Average [1]
Model	DCZ 5.9	Ixus55	D200	DMC-FZ50	L74 wide	NV15	DC-830i	DC-733s	Sensor-530s	DSC-W170	DSC-H50	FinePix J50	M1063	mju 1050 SW	Average	Average [1]
$\mathcal{P}$	13.6	99.5	33.7	33.6	53	79.1	99.2	91.5	87	50.7	45.2	6.3	52.4	29.4	45.7	N/A
$\grave{\mathcal{P}}$	-27.8	-0.5	-34.2	-29.9	-38.8	-17.4	-0.8	-7.7	-11.2	-26.3	-22.9	-20	-28.8	-43.4	-26.1	N/A

Table 10: Performances $\mathcal{P}$ in % and $\grave{\mathcal{P}}$ in %pt with the sPCE and disallowing any false alarm. $\grave{\mathcal{P}}$ is computed with sPCE and the false positive rate fixed to 10^{-3} as reference.

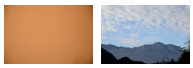
different sizes of crops are considered. The higher the crop resolution, the less noisy the computed correlation is, while the PRNU and random noise standard deviations at each pixel remain identical. In the second case, the more crops are used to estimate the PRNU, the better the PRNU is estimated, as the random noise converges to zero. As an example, ignoring the photo scene and considering an arbitrarily large number of pixels: if these pixels form a single, arbitrarily high-resolution crop per camera, then the expected Pearson correlation is strictly below 1. Conversely, if the same number of pixels is distributed across an arbitrarily large number of distinct crops, then the expected Pearson correlation is 1. As a result, the correlation threshold depends on the number of crops for a given false positive rate. More precisely, for a PRNU estimated with L crops and a false positive rate τ – and hence a correlation threshold ρ_{th} – then estimating the PRNU with more than L crops and using the correlation threshold ρ_{th} leads to a false positive rate lower than τ. However, if the PRNU is estimated with fewer than L crops, then the correlation threshold ρ_{th} leads to a false positive rate strictly higher than τ.

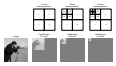
6 Conclusions

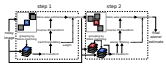
In this paper, we presented a unified and transparent implementation of several state-of-the-art PRNU estimation, refinement, and detection methods. We evaluated these techniques on a com-

prehensive dataset comprising 8,650 images from 33 camera instances, spanning 14 models and 10 brands. Our experimental metrics demonstrate that, for the evaluated models, these methods can reliably attribute photos to their specific source camera instances. Furthermore, for a given device, they successfully localize cropped regions to their original coordinates within a photo, all while maintaining a strictly low false attribution rate.

Image Credits

 RAISE Nikon D7000 and D90 cameras [8].

 Multilevel Discrete Wavelet Transform [24].

 BM3D scheme [23].

Library Versions

PyWavelets: 1.9.0 [24]
SciPy: 1.16.1 [34]

For more details, see the implementation file `requirements.txt` in the source code available from the web page of this article[4].

Appendix

Optimal Detector for Gaussian i.i.d. Random Variables Θ

As mentioned in Section 4, assuming that $\mathbf{\Theta}$ are Gaussian, independent and identically distributed random variables with unknown variance, then the optimal detector is the normalized correlation

$$\rho = \mathrm{corr}_{\mathrm{Pearson}}\left(\mathbf{I}^{(0)}\hat{\mathbf{K}}, \mathbf{R}\right). \tag{23}$$

Indeed, the likelihood functions under H_0 and H_1 are respectively

$$p\left(\mathbf{R}|\mathrm{H}_0, \sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left(-\frac{||\mathbf{R}||^2}{2\sigma^2}\right), \tag{24}$$

$$p\left(\mathbf{R}|\mathrm{H}_1, \sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left(-\frac{\left\|\mathbf{R} - \mathbf{I}^{(0)}\hat{\mathbf{K}}\right\|^2}{2\sigma^2}\right), \tag{25}$$

with N the number of image pixels. As σ^2 is unknown, we maximize each likelihood over σ^2. The maxima are respectively reached for $\frac{1}{N}||\mathbf{R}||^2$ and $\frac{1}{N}\left\|\mathbf{R} - \mathbf{I}^{(0)}\hat{\mathbf{K}}\right\|^2$. Plugging these maxima σ^2 in the

[4] https://doi.org/10.5201/ipol.2026.662

likelihood functions, we obtain the likelihood-ratio test variant

$$\Lambda(\mathbf{R}) = \frac{p\left(\mathbf{R}|\mathrm{H}_0, \frac{1}{N}||\mathbf{R}||^2\right)}{p\left(\mathbf{R}|\mathrm{H}_1, \frac{1}{N}\left\|\mathbf{R} - \mathbf{I}^{(0)}\hat{\mathbf{K}}\right\|\right)}, \tag{26}$$

$$\Lambda(\mathbf{R}) = \frac{||\mathbf{R}||^2}{||\mathbf{R}||^2 + \left\|\mathbf{I}^{(0)}\hat{\mathbf{K}}\right\|^2 - 2\mathbf{R}^{\mathrm{T}}\left(\mathbf{I}^{(0)}\hat{\mathbf{K}}\right)}. \tag{27}$$

Since for a given test, $||\mathbf{R}||^2$ and $\left\|\mathbf{I}^{(0)}\hat{\mathbf{K}}\right\|^2$ are constant, this ratio increases monotonically with $\mathbf{R}^{\mathrm{T}}\left(\mathbf{I}^{(0)}\hat{\mathbf{K}}\right)$. To avoid that the decision threshold depends on the unknown σ^2, we normalize this term as

$$\frac{\mathbf{R}^{\mathrm{T}}\left(\mathbf{I}^{(0)}\hat{\mathbf{K}}\right)}{||\mathbf{R}||\left\|\mathbf{I}^{(0)}\hat{\mathbf{K}}\right\|}, \tag{28}$$

which is the normalized correlation.

References

[1] M. Al-Ani and F. Khelifi, *On the SPN Estimation in Image Forensics: A Systematic Empirical Evaluation*, IEEE Transactions on Information Forensics and Security, 12 (2017), pp. 1067–1081, https://doi.org/10.1109/TIFS.2016.2640938.

[2] M. Al-Ani, F. Khelifi, A. Lawgaly, and A. Bouridane, *A Novel Image Filtering Approach for Sensor Fingerprint Estimation in Source Camera Identification*, in IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2015, pp. 1–5, https://doi.org/10.1109/AVSS.2015.7301808.

[3] F. Bertini, R. Sharma, A. Iannì, and D. Montesi, *Profile Resolution Across Multilayer Networks Through Smartphone Camera Fingerprint*, in International Database Engineering & Applications Symposium, Association for Computing Machinery, 2015, p. 23–32, https://doi.org/10.1145/2790755.2790765.

[4] L. Bondi and N. Bonettini, *Polimi-Ispl/prnu-Python: V.1.2*. Zenodo, 2019, https://doi.org/10.5281/zenodo.2554965.

[5] M. Chen, J. Fridrich, M. Goljan, and J. Lukas, *Determining Image Origin and Integrity Using Sensor Noise*, IEEE Transactions on Information Forensics and Security, 3 (2008), pp. 74–90, https://doi.org/10.1109/TIFS.2007.916285.

[6] A. J. Cooper, *Improved Photo Response Non-Uniformity (PRNU) Based Source Camera Identification*, Forensic Science International, 226 (2013), pp. 132–141, https://doi.org/https://doi.org/10.1016/j.forsciint.2012.12.018.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering*, IEEE Transactions on Image Processing, 16 (2007), pp. 2080–2095, https://doi.org/10.1109/TIP.2007.901238.

[8] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, *RAISE: a Raw Images Dataset for Digital Image Forensics*, in ACM Multimedia Systems Conference, 2015, p. 219–224, https://doi.org/10.1145/2713168.2713194.

[9] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, 1992, https://doi.org/10.1137/1.9781611970104.

[10] F. Gisolf, A. Malgoezar, T. Baar, and Z. Geradts, *Improving Source Camera Identification Using a Simplified Total Variation Based Noise Removal Algorithm*, Digital Investigation, 10 (2013), pp. 207–214, https://doi.org/10.1016/j.diin.2013.08.002.

[11] T. Gloe and R. Böhme, *The 'Dresden Image Database' for Benchmarking Digital Image Forensics*, in ACM Symposium on Applied Computing, Association for Computing Machinery, 2010, p. 1584–1590, https://doi.org/10.1145/1774088.1774427.

[12] T. Gloe, E. Franz, and A. Winkler, *Forensics for Flatbed Scanners*, Proceedings of SPIE, 6505 (2007), https://doi.org/10.1117/12.704165.

[13] M. Goljan, *Digital Camera Identification from Images-Estimating False Acceptance Probability*, International Workshop on Digital Watermarking, (2008), pp. 454–468, https://doi.org/10.1007/978-3-642-04438-0_38.

[14] M. Goljan, J. Fridrich, and T. Filler, *Large Scale Test of Sensor Fingerprint Camera Identification*, in Media Forensics and Security, vol. 7254, International Society for Optics and Photonics, SPIE, 2009, p. 72540I, https://doi.org/10.1117/12.805701.

[15] B. Jähne, *EMVA 1288 Standard for Machine Vision: Objective Specification of Vital Camera Data*, Optik & Photonik, 5 (2010), pp. 53–54, https://doi.org/10.1002/opph.201190082.

[16] X. Kang, J. Chen, K. Lin, and P. Anjie, *A Context-Adaptive SPN Predictor for Trustworthy Source Camera Identification*, EURASIP Journal on Image and Video Processing, (2014), p. 19, https://doi.org/10.1186/1687-5281-2014-19.

[17] X. Kang, Y. Li, Z. Qu, and J. Huang, *Enhancing ROC Performance of Trustworthy Camera Source Identification*, Proceedings of SPIE, 7880 (2011), https://doi.org/10.1117/12.872595.

[18] N. Khanna, A. K. Mikkilineni, and E. J. Delp, *Scanner Identification Using Feature-Based Processing and Analysis*, IEEE Transactions on Information Forensics and Security, 4 (2009), pp. 123–139, https://doi.org/10.1109/TIFS.2008.2009604.

[19] M. Kivanc Mihcak, I. Kozintsev, and K. Ramchandran, *Spatially Adaptive Statistical Modeling of Wavelet Image Coefficients and its Application to Denoising*, in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1999, p. 3253–3256, https://doi.org/10.1109/ICASSP.1999.757535.

[20] P. Korus and J. Huang, *Multi-Scale Analysis Strategies in PRNU-Based Tampering Localization*, IEEE Transactions on Information Forensics and Security, 12 (2016), pp. 809–824, https://doi.org/10.1109/TIFS.2016.2636089.

[21] E. Laciar and R. Jane, *An Improved Weighted Signal Averaging Method for High-Resolution ECG Signals*, in Computers in Cardiology, 2001, pp. 69–72, https://doi.org/10.1109/CIC.2001.977593.

[22] A. Lawgaly, F. Khelifi, and A. Bouridane, *Weighted Averaging-Based Sensor Pattern Noise Estimation for Source Camera Identification*, in IEEE International Conference on Image Processing (ICIP), 2014, pp. 5357–5361, https://doi.org/10.1109/ICIP.2014.7026084.

[23] M. Lebrun, *An Analysis and Implementation of the BM3D Image Denoising Method*, Image Processing On Line, 2 (2012), pp. 175–213. https://doi.org/10.5201/ipol.2012.l-bm3d.

[24] G. R. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and A. O'Leary, *PyWavelets: A Python Package for Wavelet Analysis*, Journal of Open Source Software, 4 (2019), p. 1237, https://doi.org/10.21105/joss.01237.

[25] C.-T. Li, *Source Camera Identification Using Enhanced Sensor Pattern Noise*, IEEE Transactions on Information Forensics and Security, 5 (2010), pp. 280–287, https://doi.org/10.1109/TIFS.2010.2046268.

[26] J. Lukas, J. Fridrich, and M. Goljan, *Digital Camera Identification from Sensor Pattern Noise*, IEEE Transactions on Information Forensics and Security, 1 (2006), pp. 205–214, https://doi.org/10.1109/TIFS.2006.873602.

[27] L. Min, F. Peng, and Y. Zhu, *Identifying Natural Images and Computer Generated Graphics Based on Binary Similarity Measures of PRNU*, Multimedia Tools and Applications, 78 (2019), https://doi.org/10.1007/s11042-017-5101-3.

[28] N. Mondaini, R. Caldelli, A. Piva, M. Barni, and V. Cappellini, *Detection of Malevolent Changes in Digital Video for Forensic Applications*, in Security, Steganography, and Watermarking of Multimedia Contents IX, vol. 6505, International Society for Optics and Photonics, SPIE, 2007, pp. 300 – 311, https://doi.org/10.1117/12.704924.

[29] J. Neyman and E. S. Pearson, *IX. On the Problem of the Most Efficient Tests of Statistical Hypotheses*, Philosophical Transactions of the Royal Society of London, Series A: Containing Papers of a Mathematical or Physical Character, 231 (1933), pp. 289–337, https://doi.org/10.1098/rsta.1933.0009.

[30] P. Perona and J. Malik, *Scale-Space and Edge Detection Using Anisotropic Diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (1990), pp. 629–639, https://doi.org/10.1109/34.56205.

[31] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear Total Variation Based Noise Removal Algorithms*, Physica D: Nonlinear Phenomena, 60 (1992), pp. 259–268, https://doi.org/10.1016/0167-2789(92)90242-F.

[32] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, *User Authentication Via PRNU-Based Physical Unclonable Functions*, IEEE Transactions on Information Forensics and Security, 12 (2017), pp. 1941–1956, https://doi.org/10.1109/TIFS.2017.2697402.

[33] W. van Houten and Z. Geradts, *Using Anisotropic Diffusion for Efficient Extraction of Sensor Noise in Camera Identification*, Journal of Forensic Sciences, 57 (2012), pp. 521–527, https://doi.org/10.1111/j.1556-4029.2012.02057.x.

[34] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272, https://doi.org/10.1038/s41592-019-0686-2.