IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2016–12–29.
Submitted on 2014–09–16, accepted on 2016–12–13.
ISSN 2105–1232 © 2016 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.2016.124

Analysis and Extension of the PCA Method, Estimating a Noise Curve from a Single Image

Miguel Colom, Antoni Buades

Universitat de les Illes Balears, Spain
(colom@cmla.ens-cachan.fr)
(toni.buades@uib.es)

Abstract

In the article 'Image Noise Level Estimation by Principal Component Analysis', S. Pyatykh, J. Hesser, and L. Zheng propose a new method to estimate the variance of the noise in an image from the eigenvalues of the covariance matrix of the overlapping blocks of the noisy image. Instead of using all the patches of the noisy image, the authors propose an iterative strategy to adaptively choose the optimal set containing the patches with lowest variance. Although the method measures uniform Gaussian noise, it can be easily adapted to deal with signal-dependent noise, which is realistic with the Poisson noise model obtained by a CMOS or CCD device in a digital camera.

Source Code

The C++ implementation of the PCA noise estimator is the one which has been peer reviewed and accepted by IPOL. The source code, the code documentation, and the online demo are available in the IPOL web part of this article[1]. The files which were peer-reviewed are: `algo.cpp`, `curve_filter.cpp`, `PCANoiseLevelEstimator.cpp` and their associated `.h` headers.

Keywords: noise estimation; PCA; signal-dependent noise; blind noise estimation

1 Introduction

Nowadays almost all images and movies are obtained using CMOS or CCD detectors. At the first step of the camera processing chain (acquisition of the raw image) the value observed at each pixel is a Poisson random variable whose mean is the ideal image. The difference between the observed image and the ideal image is called "shot noise".

This shot noise is of course undesirable and usually one wants to remove it. If several samples of noisy images from different realizations of the random process are available, a naive but effective solution to reduce the noise would be to simply average the images. Indeed, this principle of aggregation is used by many denoising methods. If just a single noisy sample is available (say, the

[1] https://doi.org/10.5201/ipol.2016.124

noisy image itself), there exist other denoising methods which exploit the self-similarity property of natural images to perform denoising by aggregation of similar patches.

All denoising methods need to know about the noise model corrupting the image. For example, if the noise follows a Poisson distribution, the parameter λ of the distribution would be needed. The process of obtaining the underlying noise model or its parameters (typically, its variance) using just a single noisy image is known as *blind noise estimation*.

Many methods found in the literature focus on homoscedastic white Gaussian noise [3, 2, 9, 8, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22]. That is, the noise follows a normal distribution with an unknown fixed variance. Note that this very simple noise model is not realistic in principle since photon noise depends on the intensity of the signal. However we apply a technique [7] which allows to adapt most of the block-based homoscedastic noise estimator to measure signal-dependent noise (see Section 6.2).

This article discusses and evaluates a noise estimation method based on PCA [18], modified to measure signal-dependent noise. The plan of the paper is the following: Section 2 defines precisely the notation and mathematical operators used in the article, Section 3 presents the original homoscedastic noise estimation method, Section 4 presents and discusses the implementation of the homoscedastic method.

Section 6 presents the extensions proposed to the original homoscedastic method: extension to signal-dependent noise and computing the optimal number of block groups depending on the intensity, a filtering algorithm to reduce the overestimation of the variance because of the presence of textures, and a method to discard saturated pixels when estimating the noise.

Section 7 contains the evaluation of the method, with three kinds of tests: homoscedastic Gaussian noise, noise from real raw images, and tests of multiscale coherence. Section 8 completes the analysis of the method by calculating the complexity of all the algorithms. Finally, Section 9 presents the online demo, which allow the users to test the method with their own data and reproduce the results presented in this article.

2 Notation and Terminology

This section prepares the detailed description of the noise estimation algorithm given in Section 4 by fixing its notation and terminology.

- $\mathbf{x}$: the discrete noise-free image of size $N_x \times N_y$ pixels.
- $\mathbf{n}$: a discrete additive white Gaussian noise of variance σ^2 and zero mean of size $N_x \times N_y$ pixels.
- $\mathbf{y} = \mathbf{x} + \mathbf{n}$: the corrupted image resulting from adding the noise $\mathbf{n}$ to the noise-free image $\mathbf{x}$.
- $M = M_1 \times M_2$: the size in pixels of the overlapping blocks; $M_1 = M_2 = 5 \Rightarrow M = 25$.
- $Q(p)$: the p-quantile of the list of the variances of the overlapping blocks in $\mathbf{y}$.
- $N = (N_x - M_1 + 1) \times (N_y - M_2 + 1)$: the number of overlapping blocks.
- $\mathbf{X}$: matrix of samples of a random vector made from realizations $\mathbf{x}_i$, $i \in \{1, \dots, N\}$. It can be written in matrix notation as

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1M} \\ x_{21} & x_{22} & \dots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{NM} \end{pmatrix},$$

where each row of the matrix contains a different block, that is, a realization $\mathbf{x}_i$ with $i \in \{1, \dots, N\}$.

- Var: operator which gives the variance of a random variable.
- $\mathbb{E}$: operator which gives the expectation of a random variable.
- $\mathbf{Y}$: matrix of samples of a random vector made from realizations $\mathbf{y}_i$, $i \in \{1, \dots, N\}$.
- $\mathbf{Y}_j$: samples of $\mathbf{Y}$ inside the j-th disjoint group of samples. The samples are classified according to their mean intensity and we shall call *bin* each of these divisions. The samples in $\mathbf{Y}_j$ are sorted according to their variance, that is, $\mathbf{Y}_j = \{\mathbf{y}_i \in \mathbf{Y} : Q(1-(j+1)\Delta p) < \text{Var}[\mathbf{y}_i] \leq Q(1-j\Delta p)\}$.
- $\mathbf{N}$: matrix of samples of a random vector made from realizations $\mathbf{n}_i$, $i \in \{1, \dots, N\}$ such that $\mathbf{N} \sim \mathcal{N}_M(0, \sigma^2\mathbf{I})$ and $\text{cov}(\mathbf{X}, \mathbf{Y}) = 0$, where $\mathcal{N}$ represents the normal distribution and $\mathbf{I}$ is the identity matrix.
- $S_{\mathbf{X}}$ and $S_{\mathbf{Y}}$: the empirical covariance matrices obtained from $\mathbf{X}$ and $\mathbf{Y}$, respectively.
- $\lambda_{\mathbf{X},1} \geq \lambda_{\mathbf{X},2} \geq \dots \lambda_{\mathbf{X},M}$: the eigenvalues of $S_{\mathbf{X}}$.
- $\lambda_{\mathbf{Y},1} \geq \lambda_{\mathbf{Y},2} \geq \dots \lambda_{\mathbf{Y},M}$: the eigenvalues of $S_{\mathbf{Y}}$.
- $|.|$: the cardinality of a set.

3 Principal Components in Natural Images [18]

The authors of [18] claim that information in a patch from a natural image can be expressed with fewer values than the number of pixels in the patch. Formally, any patch lies in a sub-space $V_{M-m} \subset \mathbb{R}^M$ whose dimension $M - m$ is smaller than M, the number of pixels of the patch.

Assumption 1. *Let $\boldsymbol{X}$ be the matrix of samples made of N rows, each describing a noise free patch of size M. We assume that all realizations $\boldsymbol{x}_i$ lie in a sub-space $V_{M-m} \subset \mathbb{R}^M$ whose dimension $M - m$ is smaller that the dimension M of the $\boldsymbol{x}_i$. Therefore, $\boldsymbol{x}_i$ have zero variance along any direction orthogonal to V_{M-m}. That is, the eigenvalues (sorted in decreasing order) are*

$$\lambda_{\boldsymbol{X},M-m+1} = \dots = \lambda_{\boldsymbol{X},M} = 0. \tag{1}$$

A necessary condition to fulfill this Assumption 1 is that

$$\mathbb{E}\left[\left|\lambda_{\mathbf{Y},i} - \sigma^2\right|\right] \leq T\sigma^2/\sqrt{N}, \quad i = M - m + 1, \dots, M, \tag{2}$$

where $\lambda_{\mathbf{Y},i}$ denote the eigenvalues of the covariance matrix $S_{\mathbf{Y}}$, N the number of rows in $\mathbf{Y}$ and T is a positive constant. By assuming the previous result the authors show that for noisy patch observations $\mathbf{Y}$ the following results holds.

Theorem 1. *Let $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{n}$, where $\boldsymbol{n}$ is additive white Gaussian noise of variance σ^2. If Assumption 1 holds and M is large enough, the variance of noise can be estimated from $\lambda_{\boldsymbol{Y},M}$. Indeed, taking $N \to \infty$ and the M-th eigenvalue $\lambda_{\boldsymbol{Y},M}$ leads to*

$$\lim_{N\to\infty} \mathbb{E}\left[\left|\lambda_{\boldsymbol{Y},M} - \sigma^2\right|\right] = 0. \tag{3}$$

This result provides a way to estimate the variance of the noise from the M-th eigenvalue $\lambda_{\mathbf{Y},M}$ when the number of blocks N is large enough.

If we write the index of the eigenvalue as $i = M - m + 1$ (that is, $m = 1$ corresponds to the smallest eigenvalue and $m = M$ the greatest) the necessary condition to estimate the variance is

$$\lambda_{\mathbf{Y},M-m+1} - \lambda_{\mathbf{Y},M} < T\sigma^2/\sqrt{N}. \tag{4}$$

Assuming that M is large enough, Equation (3) states that $\lambda_{\mathbf{Y},M} \approx \sigma^2$ and therefore Equation (4) bounds the error of the estimated variance when using the eigenvalue at $M - m + 1$.

The authors fix empirically the parameter $m = 7$ for blocks of $M_1 \times M_2 = 5 \times 5$ pixels and $T = 49$ (see table II in [18], page 6). To ensure that most of the patches belong to a V_{M-m} subspace, the algorithm chooses a subset of patches defined as those with smallest variance. This means that the authors choose to use the variance of the blocks in $\mathbf{y}$ as a measure of the distance of a block $\mathbf{y}_i$ to $V_{M-m}, i \in \{1, \ldots, N\}$. A simple strategy to select the subset of blocks $B(p)$ for which Equation (4) is verified consists in taking a small enough p-quantile $Q(p)$ of the list of variances of the noisy blocks and therefore to set

$$B(p) := \{\mathbf{y}_i : \mathrm{Var}\,[\mathbf{y}_i] \leq Q(p),\ i \in \{1, \ldots, N\}\}. \tag{5}$$

In practice, the authors propose an iterative strategy which reduces the quantile p while Equation (4) is not satisfied.

4 Algorithm

The noise variance estimation algorithm (Algorithm 1) first obtains an upper bound of the variance of the noise as $\sigma^2_{\mathrm{ub}} = 3.1\, Q(0.0005, \mathbf{y})$, as given in [18]. This upper bound is also used as an initial value for the iterative procedure which refines the estimated variance, implemented in the function *GetNextEstimate* (see Algorithm 1). This procedure is iterated until the absolute difference between the new variance and the previous one is negligible.

To test inequality (4) the function *GetNextEstimate* uses a noise variance initialization. The eigenvalues of the set of patches are computed and checked to see if they satisfy Equation (4) with $\sigma = \sigma_{est}$. The number of patches used for this estimation is reduced by decreasing p for the set of blocks $B(p)$ while inequality (4) does not hold. The iteration begins with $p = 1$ and p is decremented by $\Delta p = 0.05$ after each iteration. This is also the parameter used in [18].

5 Optimizing the PCA Computation

Computing the eigenvectors and the corresponding eigenvalues of the sample covariance matrix is a time consuming operation which is called several times in function *GetNextEstimate*. The PCA is computed on the set $B(p)$ for different values of p. It is possible to use the nested structure of these sets to avoid recomputing the covariance matrix at each step.

The (biased) sample covariance matrix can be written as

$$\frac{1}{|B(p)|}\left(\sum_{\mathbf{y}_i \in B(p)} \mathbf{y}_i \mathbf{y}_i^T - \frac{1}{|B(p)|}\sum_{\mathbf{y}_i \in B(p)} \mathbf{y}_i \sum_{\mathbf{y}_i \in B(p)} \mathbf{y}_i^T\right). \tag{6}$$

The number of operations needed to compute directly this matrix is proportional to $|B(p)|\, M^2$. The sets $B(p)$ in Equation (5) satisfy

$$B(1) \supset B(1 - \Delta p) \supset B(1 - 2\Delta p) \supset B(1 - 3\Delta p) \supset \ldots$$

Algorithm 1 PCA noise estimation.

PCA noise estimation - Compute noise variance.
Input $\mathbf{y}$: list of blocks of the noisy image.
Output $\sigma^2_{\mathbf{est}}$: variance of the noise.

$i_{\max} = 10$
$\varepsilon = 1e-6$

$\sigma^2_{\text{ub}} = 3.1\, Q(0.0005, \mathbf{y})$ ▷ Function $Q(p, \mathbf{y})$ gives the p-quantile of $\mathbf{y}$
$\sigma^2_{\text{est}} = \sigma^2_{\text{ub}}$
for i $= 1 \ldots i_{\max}$ **do**
  $\sigma^2_{\text{next}} = \text{GetNextEstimate}(\mathbf{y}, \sigma^2_{\text{est}}, \sigma^2_{\text{ub}})$
  if $|\sigma^2_{\text{est}} - \sigma^2_{\text{next}}| < \varepsilon$ **then**
    return σ^2_{est}
  end if
  $\sigma^2_{\text{est}} = \sigma^2_{\text{next}}$
end for
return σ^2_{est}

GetNextEstimate - Refine current variance estimation
Input $\mathbf{y}$: list of blocks of the noisy image.
Input $\sigma^2_{\mathbf{est}}$: current value of the estimated noise variance.
Input $\sigma^2_{\mathbf{ub}}$: upper bound of the noise variance.
Output $\sigma^2_{\mathbf{next}}$: next approximation of the estimated noise variance.

$p_{\min} = 0.005$
$\Delta p = 0.05$
$M = M_1 \times M_2 = 5 \times 5 = 25$
$m = 7$
$T = 49$

$p = 1$
$\sigma^2_{\text{next}} = 0$
exit = False
while $p \geq p_{\min} \wedge \neg$exit **do**
  $\lambda_{\mathbf{Y},i} = \text{ApplyPCA}(B(p))$ ▷ $\lambda_{\mathbf{Y},i}$ is the i-th eigenvalue of $S_{\mathbf{Y}}$.
  $\sigma^2_{\text{next}} = \lambda_{\mathbf{Y},M}$
  exit $= (\sigma^2_{\text{next}} < 1e-6)$ ▷ Stop iterating if variance is already too low
  if ¬exit **then**
    if $\left(\lambda_{\mathbf{Y},M-m+1} - \lambda_{\mathbf{Y},M} < \frac{T\sigma^2_{\text{est}}}{\sqrt{|B(p)|}}\right) \wedge (\sigma^2_{\text{next}} < \sigma^2_{\text{ub}})$ **then**
      return σ^2_{next}
    end if
    $p = p - \Delta p$
  end if
end while
if $\sigma^2_{\text{next}} > \sigma^2_{\text{ub}}$ **then**
  $\sigma^2_{\text{next}} = \sigma^2_{\text{ub}}$
end if
return σ^2_{next}

Therefore, the set $B(1-j\Delta p)$ can be written as $B(1-j\Delta p) = B(1-(j+1)\Delta p) \cup \mathbf{Y}_j$ where $\mathbf{Y}_j := \{\mathbf{y}_i : Q(1-(j+1)\Delta p) < \mathrm{Var}\,[\mathbf{y}_i] \leq Q(1-j\Delta p)\}$.
Since for disjoint sets X_1 and X_2 one has $C_{X_1 \cup X_2} = C_{X_1} + C_{X_2}$, then

$$C_{B(1-j\Delta p)} = C_{B(1-(j+1)\Delta p)} + C_Y.$$

Let $C_X := \sum_{\mathbf{y}_i \in X} \mathbf{y}_i \mathbf{y}_i^T$ and $\mathbf{c}_X := \sum_{\mathbf{y}_i \in X} \mathbf{y}_i$. With this notation the expression (6) can be written as

$$\frac{1}{|B(p)|}\left(C_{B(p)} - \frac{1}{|B(p)|}\mathbf{c}_{B(p)}\mathbf{c}_{B(p)}^T\right). \tag{7}$$

This strategy permits to use the covariance matrices of the previous iteration, but not the eigenvalue decomposition. With expression (7) the number of operations needed is proportional to M^2.

6 Extensions to the Original Method

The original noise estimation method based on PCA by Pyatykh et al. [18] can be improved in order to use it with real images and not only with simulated homoscedastic noise. We propose the following chain:

1. discard saturated pixels, to avoid the possibility that particular isolated pixels produce an incorrectly interpolated noise curve;
2. extend the method to measure signal-dependent noise. Note that at the very first step of the camera processing chain (raw image acquisition) the noise is already signal-dependent (Poisson distributed);
3. filter the obtained signal-dependent noise curve, to avoid the overestimation effect which occurs when all the samples in a particular bin come from a texture.

6.1 Discarding Saturated Pixels

When the number of photons arriving to the CCD or CMOS device during the exposure time is too high, its output saturates. Consequently, when the pixels are completely saturated the measured variance is zero. If saturated pixels are taken into account when measuring the noise, the noise curve is no longer reliable. Since the intensity of the saturated pixels is much higher than the intensity of most of the pixels in the image, the noise curve shows typically a large gap between normal non-saturated control points and the first saturated control point. To obtain intermediate values inside the gap, any algorithm using the noise curve will need to interpolate the standard deviation values, giving a large set of interpolated wrong values. Therefore, it is better to avoid taking into account saturated pixels in the noise estimation, to prevent saturated control points in the noise curve.

The strategy used to discard saturated pixels is to avoid considering blocks which contain a subgroup of four-connected pixels with exactly the same intensity, in any of the channels. This is useful not only to discard saturated pixels, but also to avoid processing blocks whose pixels have suffered other types of alterations that can be detected by finding these particular blocks. For example, JPEG encoding with a high compression ratio sets to zero the value of high-frequency coefficients in most of the 8×8 blocks in the image. Among other undesired effects like low frequency artifacts and blocking patterns, it can also create smooth zones where the standard deviation of the noise falls to zero. In natural images that have not been deeply compressed, the probability of finding a set of four connected pixels sharing exactly the same value is very low, because of noise and textures. The pseudocode and the details on what is exactly the four-connection of the pixels is given in Algorithm 2.

Algorithm 2 Detection of completely saturated pixels.

SAT_DETECTION - Creates a mask of valid pixels. **Input I**: input image. **Input** N_x: width of **I**. **Input** N_y: height of **I**. **Input** w: block side. **Input** num_channels: number of channels of **I**. **Output** mask: mask of VALID/INVALID pixels.

$\varepsilon = 10^{-3}$
for $i = 0 \dots N_x - 1$ **do**
  for $j = 0 \dots N_y - 1$ **do** ▷ Check if the pixel is not too close to the image boundary
    if $i < N_x - w \wedge j < Ny - w$ **then**
      for c = 0, . . . , num_channels - 1 **do**
        u = **I**.get_channel(c) ▷ Look if the 2×2 block is constant
        pixel_status = (INVALID if c == 0 else mask[x,y]) ▷ Try to validate pixel
        if $|u[i,j]-u[i+1,j]| > \varepsilon \vee |u[i+1,j]-u[i,j+1]| > \varepsilon \vee |u[i,j+1]-u[i+1,j+1]| > \varepsilon$ **then**
          pixel_status = VALID
        end if
        mask[i, j] = pixel_status
      end for
    else
      mask[i, j] = INVALID
    end if
  end for
end for

6.2 Extension to Signal-Dependent Noise

Most noise estimation methods found in the literature assume that the noise in the image is additive, signal-independent, and Gaussian. Note that in this context *uniform* means that the variance of the Gaussian noise is fixed and does not depend on the intensity of the pixels of the noiseless image. This assumption is not realistic because of the quantum nature of light itself and the way a CMOS or CCD detector responds to light. It is well-known that the emission of photons by a body follows a Poisson distribution. This distribution can be approximated by a Gaussian distribution when the number of photons is large enough. For very dark regions of the image this assumption does not hold. We consider an image pixel $\tilde{\mathbf{U}}(x, y)$ as a Poisson variable with variance and mean $\mathbf{x}(x, y)$. The Poisson noise has therefore a standard deviation of $\sqrt{\mathbf{x}(x, y)}$. (An image is nothing but a noise whose mean would be the ideal image.) This noise adds up to a thermal noise and to an electronic noise which are approximately additive and white. On a motionless scene with constant lighting, the expected value $\mathbf{x}$ can be approximated by simply accumulating photons for a long exposure time, and then by taking the temporal average of this photon count. Any noise estimation algorithm assuming that the noise is uniform is unrealistic. Fortunately, most block-based methods are easily adapted to signal-dependent noise.

Applying a variance stabilizing transform (VST) to the image in order to turn the noise into homoscedastic is only possible when the noise model is known. For example, if the model is Poisson there exists closed formulas for the VST [1, 14]. However, if the noise model is not known computing the VST implies knowledge of the noise curve of the noise, which converts the issue in a chicken-egg problem.

For a signal-dependent noise, a "noise curve" (also known as a "noise level function" or NLF in the literature) must be established [7]. This noise curve associates with each image value $\mathbf{x}(x, y)$ an estimation of the standard deviation of the noise associated with this value. Thus, for each block

in the image, the mean values of the block are computed to estimate the noiseless signal $\mathbf{x}$. The means are classified into a disjoint union of variable intervals or bins, in such a way that each interval contains a large enough number of elements. These measurements allow for the construction of a list of block variances whose corresponding means belong to the given bin. Section 6.2.1 discusses the procedure used to obtain the optimal number of samples/bin when adapting the PCA method to signal-dependent noise.

Therefore, it is possible to apply the PCA noise estimation algorithm to each set of blocks associated with a given bin. In this way, one obtains an estimation of the noise for the intensities that belong to the considered bin. Since there is no gap between bins and a particular block can only belong to an unique bin, is it possible to deduce by interpolation a curve that relates the means of the blocks with their standard deviations, hence obtaining a signal-dependent *noise curve*. The intensity associated to each bin is given by the mean intensity of the blocks whose variance is under the p-quantile. The algorithmic description of the function building this histogram of block means can be found in Algorithm 3. This algorithm works as follows:

1. It takes as input the number of bins that will be used ("bins" variable), the input data (the variances of the blocks, "data" variable), the associated intensities of the input data (the means of the blocks, "datal" variable) and the total number of samples ("N" variable). The algorithm stores in the variable "samples_per_bin" the integer part of N/bins. In general, samples_per_bin = 112000 samples/bin. Since the last bin contains the remaining samples, it may contain less than samples_per_bin samples.

2. It returns for each bin b its intensity bounds ("limits_begin[b]" and "limits_end[b]" variables), the list of variances that belong to bin b ("data_bins[b]" variable) and the list of intensities (block means) that belong to bin b ("datal_bins[b]" variable).

3. For each bin b, the algorithm fills the data_bins[b] and datal_bins[b] buffers with the variances and intensities of the blocks, sorted by their mean.

4. The lower and upper intensity bounds of the current bin b are stored into the variables limits_begin[b] and limits_end[b]. Then, the next bin is processed.

6.2.1 Obtaining the Optimal Number of samples/bin

To be able to measure signal-dependent noise it suffices to classify all blocks of the image according to their mean intensity and to apply the homoscedastic noise estimator method to each block that belongs to a given bin. This is a general procedure which can be applied to most of the block-based noise estimation methods [7], taking into account that the required number of samples per bin will be different for each particular noise estimator.

To find the number of samples/bin that minimizes the RMSE of the obtained curve with the PCA method compared to the ground-truth curve, we added signal-dependent simulated $\mathcal{N}(\mu = 0, \sigma^2 = 1 + 2\mathbf{x})$ noise on a set of noise-free images (Figure 1) of 1080×808 pixels each. The parameters of the noise are chosen in order that the constant noise (of variance 1) is small with respect to the photonic noise (variance $2\mathbf{x}$). As explained in Section 7.1, these images contain a negligible amount of noise, but they will be assumed noise-free in practice. The mean SNR of these images can be calculated as

$$\mathrm{SNR} = 10 \log_{10} \frac{1}{x} \int_0^{\max x} \frac{x}{\sqrt{1 + 2x}} \, \mathrm{d}x \quad (\mathrm{dB}).$$

Since the maximum intensity for each pixel is 255, the mean SNR is 8.75 dB.

Algorithm 3 Algorithm classifying blocks by their means.

CLASSIFY_BY_MEAN - Classifies the elements in the input list into bins according to their mean. **Input bins**: number of bins. **Input data**: list of input data elements. **Input N**: number of elements/bin. **Input datal**: list of means of the input elements. **Output limits_begin**[b]: the lower intensity bound for bin b. **Output limits_end**[b]: the upper intensity bound for bin b. **Output data_bins**[b]: list of elements in bin b. **Output datal_bins**[b]: list of means of the elements at bin b. Each bin contains similar number of elements, with the exception of the last.

```
samples_per_bin = floor(N / bins)                          ▷ Compute number of samples per bin
if N mod bins != 0 then
    samples_per_bin++
end if
limits_begin, limits_end, num_elements, data_bins, datal_bins = zeros(bins)
buffer, bufferl = array(N)
indices = argsort(datal, N)                                ▷ Sort data by datal
elements_count = 0
bin = 0
for idx = 0 . . . N − 1 do
    buffer[elements_count] = data[indices[idx]]            ▷ Keep loading . . .
    bufferl[elements_count] = datal[indices[idx]]
    elements_count = elements_count + 1
    if (idx == N-1) ∨ (elements_count == samples_per_bin) then   ▷ Copy bin ← buffer
        copy(data_bins[bin], buffer, elements_count)
        copy(datal_bins[bin], bufferl, elements_count)
    end if
    limits_begin[bin] = bufferl[0]                         ▷ Write bin metadata
    limits_end[bin] = bufferl[elements_count-1]
    num_elements[bin] = elements_count
                                                           ▷ Prepare for next bin
    elements_count = 0
    idx = idx + 1
end for
limits_end[bins-1] = max_datal
```

Then the averaged RMSE for all the images in the set (Figure 2) was computed. The minimum is attained when using 5 bins. However, since the error made when using 8 bins is not much worse than the error using 5 bins and a noise curve with 8 control points is more informative than the noise curve with 5 bins, we decided to use 8 bins for an image of 1080×808 pixels. Therefore, the number of samples/bin is $\frac{1080 \times 808}{8} = 109080$. Lastly, we fixed a minimum of 112000 samples/bin. Empirically, this value gives accurate results for most natural images.

In other methods [6, 5] the minimum number of samples/bin obtained with this procedure is smaller. The PCA methods needs twice the number of samples in order to obtain a similar performance in terms of PSNR.

6.3 Filtering the Noise Curve

It may happen that all blocks assigned to a particular bin with Algorithm 3 belong to a highly textured area of the image. In this case, it is not possible for the presented algorithm to separate

Figure 1: Set of noise-free images (actually, assumed to contain a very low and negligible noise, as explained in Section 7.1) used to determine empirically the optimal number of samples/bin needed by the algorithm. Each image is 1080×808 pixels. The parameters of the noise are chosen in order that the constant noise (of variance 1) is small with respect to the photonic noise (variance $2\mathbf{x}$).

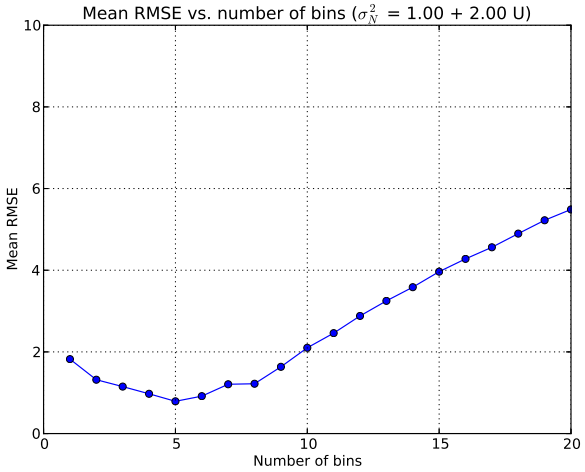

Figure 2: Mean RMSE of the estimation of the signal-dependent simulated $\mathcal{N}(\mu = 0, \sigma^2 = 1 + 2\mathbf{x})$ noise for all the noise-free (with negligible noise) test images in Figure 1. The images are encoded in 8 bits and the mean SNR is 8.75 dB

properly the noise from the signal. In this situation, most of the blocks show a high variance which is mostly explained by the geometry of the image and not by the noise. Thus, the estimated variance would be over the real variance of the noise. A way to overcome this problem consists in filtering the noise curve. Indeed, if the i-th bin contains most of its samples coming from textures but not the $(i-1)$-th and then $(i+1)$-th bins, it is possible to filter the peak which appears in the noise curve at bin i.

Given the i-th control point of the noise curve $(\hat{\mu}_i, \hat{\sigma}_i)$, the filtering algorithm considers the closed intensity interval $[\hat{\mu}_i - D, \hat{\mu}_i + D]$. We use the *hat* notation when we refer to *estimated values* of variance (or standard deviation) and mean intensity. For each intensity μ inside the interval (assuming that $\mu \in [\hat{\mu}_i - D, \ldots, \hat{\mu}_i + D]$ is incremented with step 0.05), the interpolated standard deviation that corresponds to each intensity μ is obtained. To avoid an excessive smoothing caused by interpolation, if $\hat{\mu}_i - D < \hat{\mu}_0$, then we use $\hat{\mu}_b - \hat{\mu}_0$ instead of the radius D. Also, if $\hat{\mu}_i + D > \hat{\mu}_{B-1}$

(where B is the number of bins), instead of D, the value $\hat{\mu}_{B-1} - \hat{\mu}_b$ is used.

Since each estimated $\hat{\sigma}_i$ can be seen as a perturbation around the true standard deviation, averaging the noise curve within the interval $[\hat{\mu}_i - D, \hat{\mu}_i + D]$ for each control point attenuates the perturbations and puts them closer to the ground-truth. These perturbations occur since the variance is measured with a block of $M_1 \times M_2$ pixels and therefore the obtained values of estimation can be modeled with a χ^2_M distribution of M (the number of pixels in a block) degrees of freedom, which has its own variance and zero mean. A large value of M implies less perturbations in the measure of the variance of the noise, but it also increases the probability of capturing zones of the image containing textures. Even if these perturbations can be attenuated, it still may happen that a control point corresponds to a peak, thus causing a overestimation peak in the noise curve. In that case, our strategy is to compute the average standard deviation within interval $[\hat{\mu}_i - D, \hat{\mu}_i + D]$ and to substitute the standard deviation $\hat{\sigma}_i$ by that average only if it is *lower* than the average of the intensities in the interval. In short, we only allow the control point of the curve to move down, never up.

The filtering procedure is iterated five times. In the first three iterations the control points are allowed to move up and down, thus canceling the perturbations around the ideal value. In the next two iterations the control points are only allowed to go down, to attenuate the overestimation caused by textures. The simple strategy presented here performs adequately for most natural images and in general not more than five filtering iterations are needed to get a reliable estimation of the noise. Applying more than five iterations does not improve the results significantly and for certain images it could produce noise curves that are excessively smooth. A radius $D = 7$ is recommended. The pseudo-code of the filtering is detailed in Algorithm 6. It uses Algorithm 5 to interpolate the standard deviation corresponding to a given intensity. Algorithm 5 uses Algorithm 4 to get the corresponding standard deviation by a simple affine transformation.

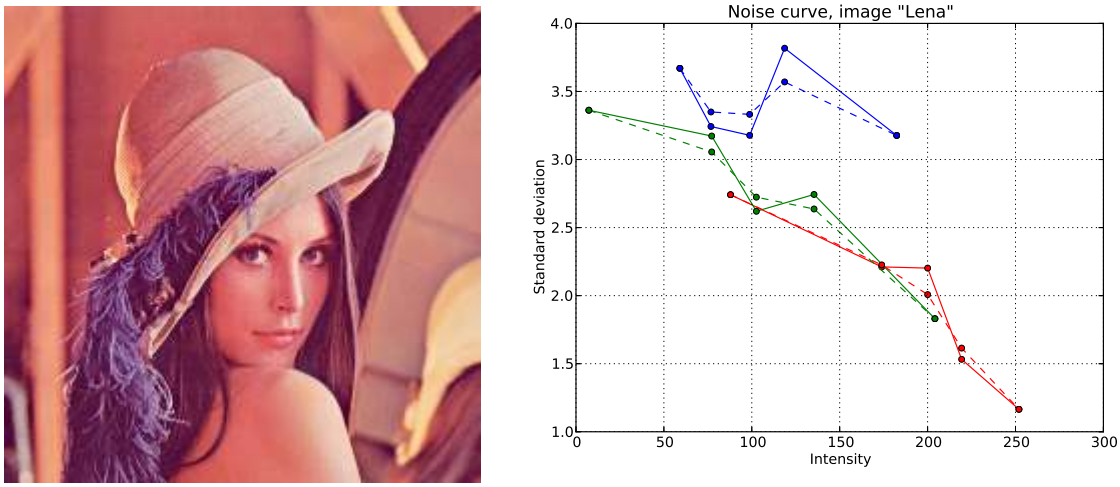

Figure 3: Left: test image *Lena* used to compare the noise curves with and without filtering. Right: noise curve for *Lena*. The color of the curves corresponds to the color channels of the image. The non-filtered curve is drawn with solid lines and the filtered curve (five iterations) with dashed lines, using $D = 7$, $p = 0.5\%$, $M_1 = M_2 = 5$ and 5 bins. Note that the peak in the blue channel is corrected after the filtering. The image has size 512×512 and thus the optimal number of bins would be two. However, using two bins is not enough and it is preferable to have more control points to have information on the noise for several intensities, even if the accuracy at each bin is not optimal.

Figure 3 shows the noise curve for the test image *Lena*. The non-filtered curve is drawn with solid lines and the filtered curve (five iterations) with dashed lines, using $D = 7$, $p = 0.5\%$, $M_1 = M_2 = 5$

and 5 bins. Note that the peak in the blue channel has decreased. The image has size 512×512 and thus the optimal number of bins would be two. However, using two bins is not enough and it is preferable to have more control points to have information on the noise for several intensities, even if the accuracy at each bin is not optimal.

Algorithm 4 Obtains a corresponding standard deviation by an affine transformation.

AFFINE. **Input** (μ_c, σ_c): current control point. **Input** (μ_e, σ_e): endpoint control point. **Input** μ: intensity of the control points whose standard deviation is wanted. **Output** σ: standard deviation attributed to the intensity μ.

$\varepsilon = 10^{-6}$
if $|\mu_c - \mu_e| < \varepsilon$ **then** ▷ Avoid dividing by zero
    $s = 0$
else
    $s = \frac{\sigma_c - \sigma_e}{\mu_c - \mu_e}$
end if
$\sigma = (\mu - \mu_e)s + \sigma_e$

Algorithm 5 Interpolates an affine standard deviation using the given reference control points.

INTERPOLATION. **Input** (μ_c, σ_c): known control points. **Input** μ: the intensity of the point whose interpolated standard deviation is wanted. **Output** σ: the interpolated standard deviation of the point whose intensity is μ.

▷ Find the nearest control point
$i = \operatorname{argmin}_i (\mu_c[i] - \mu|)$ ▷ Use the smallest i if several give the minimum
$m = \mu_c[i]$
if $\mu < m$ **then** ▷ on the right of μ
    if $i = 0$ **then** ▷ Treat boundary
        $i = 1$
        $m = \mu_c[i]$
    end if
    $m_1 = \mu_c[i-1]$
    $m_2 = m$
    $s_1 = \sigma_c[i-1]$
    $s_2 = \sigma_c[i]$
else ▷ on the left of μ
    $N = \text{len}(\mu_c)$ ▷ Compute the length of the array
    if $i \geq N - 1$ **then** ▷ Treat boundary
        $i = N - 2$
        $m = \mu_c[i]$
    end if
    $m_1 = m$
    $m_2 = \mu_c[i+1]$
    $s_1 = \sigma_c[i]$
    $s_2 = \sigma_c[i+1]$
end if
return AFFINE$(m_1, s_1, m_2, s_2, \mu)$

Algorithm 6 Filters a noise curve.

FILTER_CURVE Input (μ_c, σ_c): list of control points to be filtered. **Input** D: radius. **Input** allow_up: allows the points to go up and down. Otherwise, they are only allowed to go down. **Output** σ_c^o: list of filtered standard deviations

B = len(μ_c) ▷ Compute the length of the array
$\sigma_c^o \leftarrow \emptyset$
for $b = 0, \ldots, B-1$ **do**
    mu_current, std_current = $\mu_c[b], \sigma_c[b]$
    left = mu_current − D
    right = mu_current + D
    ▷ Adjust radius for the points near the boundary
    if left < $\mu_c[0]$ **then**
        dist = $\mu_c[b] - \mu_c[0]$
        left = mu_current - dist
        right = mu_current + dist
    else
        if right > $\mu_c[B-1]$ **then**
            dist = $\mu_c[B-1] - \mu_c[b]$
            left = mu_current - dist
            right = mu_current + dist
        end if
    end if
    ▷ Add the interpolated control points inside the interval [left, right]
    sum_window = 0
    L = 0
    for μ = left ... right (with step $\Delta = 0.05$) **do**
        sum_window += INTERPOLATION(μ_c, σ_c, μ)
        L += 1
    end for
    std_new = sum_window / L ▷ Compute mean
    if allow_up **then**
        std_filtered = std_new
    else
        std_filtered = std_new **if** std_new < std_current **else** std_current
    end if
    $\sigma_c^o \leftarrow$ std_filtered
end for
return σ_c^o

7 Evaluation of the Method

To evaluate the accuracy of the method, three kind of tests were performed.

- Tests for simulated uniform Gaussian noise using the images of Figure 4. In this case we have taken seven bins to classify the blocks according to their means (see Section 6.2). The reason to use seven bins and not two (the optimal value) is that a noise curve with only two bins is not informative at all.

- Tests for a set of real raw images (see Figure 5). The procedure explained in Section 6.2 was used to get a noise curve. The results were compared to the ground-truth noise curve of the camera.

- Test for multiscale coherence. The standard deviation of a Gaussian white noise is divided by two when the image is down-scaled. By *down-scaling* the image we mean a sub-sampling of the image where each block of four pixels is substituted by their mean. This test checks if the measured noise is divided by two at each image down-scaling.

7.1 Evaluation with Simulated Uniform Noise

In this experiment, uniform noise was simulated and then added to a set of ten noise-free (actually, with negligible noise) images. Since the noise model is perfectly known *a priori*, its noise level curve can be used as a ground truth. One can therefore compute the RMSE of the standard deviation estimations. Figure 4 shows a set of 704×469 pixels noise-free images that were used in this test. To get the noise-free images, we have applied the following procedure. The pictures were taken with a Canon EOS 30D reflex camera of scenes under good lighting conditions and with a low ISO level. To further reduce the noise level, the average of each block of 5×5 pixels was computed, reducing therefore the noise by a factor of 5. Since the images are RGB, the mean of the three channels was computed, reducing the noise by a further $\sqrt{3}$ factor. Therefore the standard deviation of the noise was reduced by a $5\sqrt{3} \simeq 8.66$ factor. Lastly, the images (which already had a good SNR before being processed), can be assumed noise-free.

Figure 4: Set of noise-free images used to test the noise estimation algorithm with uniform noise. Actually, they contain a negligible amount of noise, as explained in Section 7.1. From left to right and from top to bottom: bag, building1, computer, dice, flowers2, hose, leaves, lawn, stairs and traffic. The size of each image is 704×469 pixels.

To measure the error made when estimating the standard deviation σ of the simulated noise of bin b in image i, the RMSE for all the bins was used. This RMSE is denoted by $E^{(1)}_{i,\sigma}$ and it is defined

as

$$E_{i,\sigma}^{(1)} := \sqrt{\frac{1}{|B|}\sum_{b=1}^{|B|} |\hat{\sigma}_{i,b} - \sigma|^2},$$

where i is the image index ($1 \leq i \leq |I|$ and $|I|$ is the number of images), $|B|$ is the number of bins, b is the index of the bin ($1 \leq b \leq |B|$), σ is the standard deviation of the simulated noise, and $\hat{\sigma}_{i,b}$ is the estimated noise for image i at bin b. Table 1 gives the obtained $E_{i,\sigma}^{(1)}$ for each image i and each σ of the simulated noise. To test the response of the algorithm to a pure white noise image we used a constant image *flat image* where all the pixels have intensity 127. Three bins are used since each is image is 704×469 pixels and at least 112000 samples/bin are needed for an accurate estimation. Note that the PCA method is able to give an accurate estimation of the noise even if $\sigma = 1$ when the image has large flat zones (building1, computer, dice, flowers2, traffic), but fails to give a good estimation for very textured images (bag, hose, leaves, lawn). For very high noise levels ($\sigma = 50$, $\sigma = 80$) the estimation is inaccurate. Surprisingly, the method behaves better for small levels of noise. The result obtained with the constant image is worse when the algorithm is forced to use several bins. The reason is that in such a constant image a single bin should be used. In this extreme case, noise will make that the empirical mean of the noise blocks is different from the noise-free constant image, and the classification will be biased. The blocks in the bins will have a similar mean because of the noise, not because of the underlying signal.

The last row is the average RMSE obtained for a given σ and all the images. It is denoted by $E_{\sigma}^{(2)}$ and defined as

$$E_{\sigma}^{(2)} = \sqrt{\frac{1}{|B||I|}\sum_{i=1}^{|I|}\sum_{b=1}^{|B|} |\hat{\sigma}_{i,b} - \sigma|^2} = \sqrt{\frac{1}{|I|}\sum_{i=1}^{|I|} \left(E_{i,\sigma}^{(1)}\right)^2}.$$

For completeness, the results corresponding to the estimation using just a single bin are shown in Table 2, although this model of signal-independent noise using a single bin is not realistic at all. Table 3 gives the obtained $E_{\sigma}^{(2)}$ RMSE depending on the number of the iterations of the noise curve filter (see Section 6.3). Using one filtering iteration improves slightly the accuracy whereas more than five is useless.

7.2 Evaluation Comparing the Noise Curve of the Raw Image with the Ground Truth

In this evaluation, each noise curve obtained with the proposed algorithm for the raw images in Figure 5 (12 bits/channel, ISO 1600, t=1/30 s) was compared to the corresponding "ground truth" of the camera. The ground truth was obtained by computing, for each pixel, the standard deviation of a large burst [4] of fixed snapshots of the same calibration pattern (Figure 6).

To obtain the ground truth of the camera, we fixed the ISO sensitivity (in this case at ISO 1600) and used four exposure times, $t \in \{1/30s, 1/250s, 1/400s, 1/640s\}$. For each exposure time about two hundred pictures of the calibration pattern were taken (see Figure 6). After cropping the area of the image that does not contain the calibration pattern, the final size of the raw image was 1352×1952. Since each 2×2 block of the CFA[2] contains one sample of the red channel, two samples of the green channel and one sample of the blue channel, it is possible to discard one of the two green channels and combine the rest of the channels to obtain an RGB image of size 676×976 pixels. The position of the camera was fixed when taking the snapshots of the image. We assume that the lighting is

[2]Color Filter Array.

Image / $E^{(1)}_{i,\sigma}$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$	$\sigma = 80$
bag	0.65	0.49	0.29	0.27	0.40	6.04	22.98
building1	0.18	0.10	0.21	0.46	0.65	2.17	2.38
computer	0.14	0.11	0.17	0.37	0.67	1.10	3.43
dice	0.13	0.06	0.09	0.30	0.59	1.81	3.77
flowers2	0.17	0.10	0.07	0.22	0.81	2.78	9.28
hose	0.76	0.66	0.35	0.28	0.25	9.61	26.78
lawn	0.87	0.62	0.46	0.57	0.32	13.71	29.49
leaves	1.16	0.71	0.35	0.31	0.40	3.67	17.83
stairs	0.55	0.55	0.50	0.53	1.01	17.15	33.67
traffic	0.14	0.23	0.34	0.59	0.59	0.58	3.36
Flat image	0.58	1.18	2.90	5.85	11.59	28.92	46.80
$E^{(2)}_{\sigma}$	**0.49**	**0.44**	**0.52**	**0.89**	**1.57**	**7.96**	**18.16**

Table 1: This table shows the PCA $E^{(1)}_{i,\sigma}$ average RMSE after adding simulated noise to the set of noise-free images (Figure 4 shows the images, actually with negligible noise) with several values of standard deviation σ. The last row is the $E^{(2)}_{\sigma}$ RMSE using the estimated $\hat{\sigma}_{i,b}$ of all the images. Note that the PCA method is able to give an accurate estimation of the noise even when $\sigma = 1$ when the image has large flat zones (building1, computer, dice, flowers2, traffic), but fails to give a good estimation for very textured images (bag, hose, leaves, lawn). For very high levels of noise ($\sigma = 50$, $\sigma = 80$) the estimation is inaccurate. Surprisingly, the method behaves better for small levels of noise. Three bins are used.

Image / $E^{(1)}_{i,\sigma}$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$	$\sigma = 80$
bag	0.55	0.40	0.21	0.08	0.35	0.15	1.27
building1	0.17	0.08	0.08	0.12	0.18	1.48	1.83
computer	0.11	0.05	0.12	0.28	0.25	0.69	3.03
dice	0.11	0.07	0.07	0.26	0.52	1.71	3.76
flowers2	0.14	0.09	0.02	0.08	0.54	1.87	2.76
hose	0.32	0.25	0.08	0.20	0.18	0.91	2.83
lawn	0.33	0.12	0.15	0.32	0.38	1.97	1.65
leaves	1.02	0.62	0.06	0.18	0.18	0.29	2.34
stairs	0.38	0.24	0.24	0.44	0.29	1.25	2.61
traffic	0.25	0.28	0.17	0.33	0.29	0.03	2.99
Flat image	0.04	0.08	0.20	0.40	0.85	2.47	3.22
$E^{(2)}_{\sigma}$	**0.31**	**0.21**	**0.13**	**0.25**	**0.37**	**1.17**	**2.57**

Table 2: This table shows the $E^{(1)}_{1,\sigma}$ RMSE after adding simulated noise to the set of noise-free images (Figure 4, with negligible noise) with several values of standard deviation σ. The last row is the $E^{(2)}_{\sigma}$ RMSE using the estimated $\hat{\sigma}_{1,b}$ of all the images. A single bin is used.

constant during the acquisition of the burst. Therefore, the variance of every given pixel can only be explained by the noise. Therefore, it is possible to create an association mean→standard deviation (the ground truth) for the camera noise curve, given the ISO and exposure time. The noise curves corresponding to the same calibration pattern but with different exposure times share the same noise model and actually they overlap. Therefore, we use from short to large exposure times in order to

Image / $E_\sigma^{(2)}$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$	$\sigma = 80$
No filtering	0.49	0.44	0.52	0.89	1.57	7.96	18.16
1 iteration	0.49	0.42	0.48	0.82	1.46	7.22	16.90
2 iterations	0.49	0.41	0.46	0.78	1.39	6.66	15.93
3 iterations	0.49	0.41	0.45	0.76	1.35	6.23	15.17
4 iterations	0.48	0.40	0.45	0.75	1.34	6.25	15.19
5 iterations	0.48	0.40	0.45	0.75	1.34	6.26	15.20
6 iterations	0.48	0.40	0.45	0.74	1.33	6.27	15.21
7 iterations	0.47	0.40	0.45	0.74	1.33	6.28	15.23

Table 3: This table gives the obtained $E_\sigma^{(2)}$ RMSE values depending on the number of iterations of the noise curve filtering and the standard deviation of the noise (see Section 6.3). Three bins are used. Five iterations is the recommended value, since using more iterations does not improve the result significantly and it could smooth too much the noise curves for certain images.

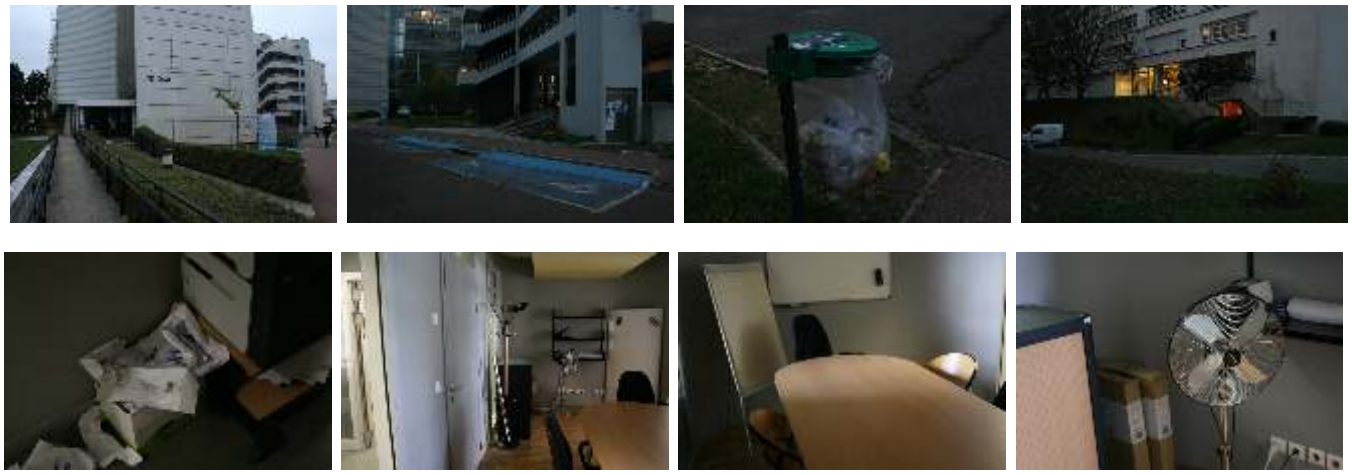

Figure 5: Set of raw images used to test the noise estimation algorithm using 25 bins and without any noise curve filtering. The images are 3500 × 3200 pixels, raw 12 bits/channel, ISO 1600 and exposure time t=1/30 s. From left to right and top to bottom: images 1, 2, 3, 4, 5, 6, 7, and 8.

Figure 6: One of the pictures of the calibration pattern used to build the ground truth noise curve of the camera.

cover a wide intensity range and later we combine these overlapping curves to obtain a single ground truth curve which depends only on the ISO, as shown in Figure 7.

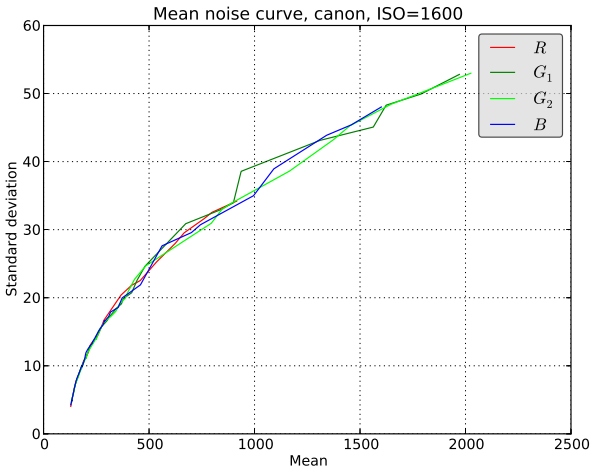

Figure 7: Ground truth of the Canon EOS 30D camera with ISO=1600.

Given an estimated (respectively a ground truth) noise curve A (respectively G) of a test image i, its control points are the pairs $(\hat{\mu}_{A,i,b}, \hat{\sigma}_{A,i,b}) \in A$ (respectively $(\mu_{G,v}, \sigma_{G,v}) \in G$) where $\hat{\mu}_{A,i,b}$ is the mean intensity for bin b and image i in A and $\hat{\sigma}_{A,i,b}$ is the corresponding standard deviation value for bin b and image i in A. The abscissas of the noise curve A and those in G do not necessarily coincide; that is, $\hat{\mu}_{A,i,b} \neq \mu_{G,v}$ for most (b, v) pairs. To solve this problem, instead of using G, a new ground truth curve $\tilde{G}_i$ is used. This $\tilde{G}_i$ curve has the same means $\hat{\mu}_{A,i,b}$ as A (and therefore the same number of bins), and its standard deviation values are obtained by a simple proportionality rule. Therefore, the control points in the new curve $\tilde{G}_i$ are $(\hat{\mu}_{A,i,b}, \tilde{\sigma}_{G_i,i,b}) = \left(\hat{\mu}_{A,i,b}, \frac{\sigma_{G,v+1}-\sigma_{G,v}}{\mu_{G,v+1}-\mu_{G,v}}(\hat{\mu}_{A,i,b} - \mu_{G,v+1}) + \sigma_{G,v}\right)$ where v is the index of the bin in the curve G such that $\mu_{G,v} \leq \hat{\mu}_{A,i,b} < \mu_{G,v+1}$ (see Figure 8).

The error between the ground truth noise curve G and the test noise curve A for image i and bin b is defined as

$$E^{(3)}_{G,A,i,b} := |\tilde{\sigma}_{G_i,i,b} - \hat{\sigma}_{A,i,b}| = \left| \frac{(\sigma_{G,v+1} - \sigma_{G,v})(\hat{\mu}_{A,i,b} - \mu_{G,v+1})}{\mu_{G,v+1} - \mu_{G,v}} + \sigma_{G,v} - \hat{\sigma}_{A,i,b} \right|.$$

Table 4 shows that that error is worse for those images that are highly textured and better for those that contain large flat zones.

Img. 1	Img. 2	Img. 3	Img. 4	Img. 5	Img. 6	Img. 7	Img. 8
0.984	1.107	3.577	1.640	2.103	1.010	0.788	1.381

Table 4: Values of $E^{(3)}_{G,A,i,b}$ measuring the error between the noise curve A obtained for each test image i (Figure 5) and the ground truth curve G with ISO 1600. The error is worse for those images that are highly textured (see image 3) and better for those that contain large flat zones (see images 6 and 7).

To test the average behavior of the algorithm in all the bins of any test image, we define a mean error function $E^{(4)}_{G,A,b}$ as the mean of the $E^{(3)}_{G,A,i,b}$ values over the $|I|$ images for each of the $|B|$ bins, that is,

$$E^{(4)}_{G,A,b} = \frac{1}{|I|} \sum_{i=1}^{|I|} E^{(3)}_{G,A,i,b}.$$

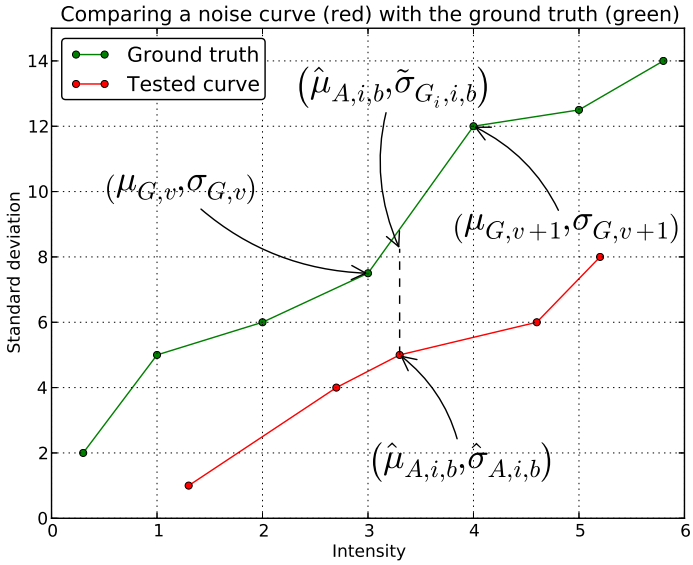

Figure 8: Checking a noise curve A (red) against the ground truth G (green), where i is the index of the image, b is the index of the bin, $(\hat{\mu}_{A,i,b}, \hat{\sigma}_{A,i,b})$ are the control points of the noise curve A, $(\mu_{G,v}, \sigma_{G,v})$ are the control points of G, and $\tilde{\sigma}_{G_i,i,b}$ is the standard deviation value projected from A on G.

Figure 9 shows the error $E^{(4)}_{G,A,b}$ for all the 25 bins of the test images in Figure 5 for the first green channel.

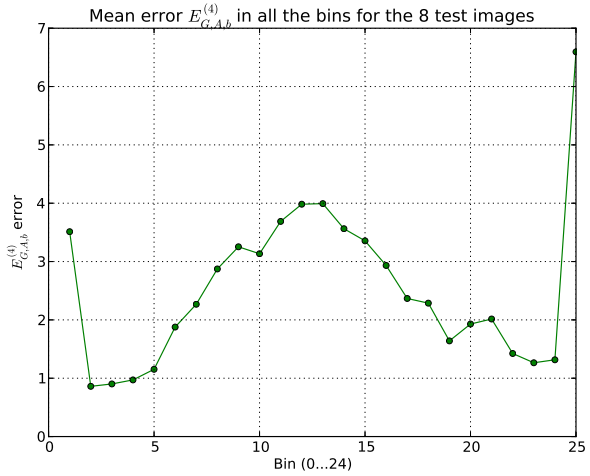

Figure 9: Mean error $E^{(4)}_{G,A,b}$ for the 25 bins of all the eight tests images of Figure 5.

7.3 Evaluation of the Multiscale Coherence of the Result

Consider the down-scaling operator $\mathbf{S}$ that tessellates the image into 2×2 pixels blocks, and replaces each block by a pixel having the mean of the four previous pixels as new value. If $\tilde{\mathbf{x}}$ is an image whose pixels are i.i.d. realizations of a Gaussian random variable with standard deviation σ, then $\mathbf{S}(\tilde{\mathbf{x}})$ has standard deviation $\frac{\sigma}{2}$.

This test aims at testing if the estimated standard deviation divides by two each time the image is down scaled by a factor two.

$$\text{Set:}\quad E^{(5)}_{A_0,A_k,i,b} = \left|\frac{\tilde{\sigma}_{A_0,i,b}}{\hat{\sigma}_{A_k,i,b}} - 2^k\right| = \left|\frac{(\hat{\sigma}_{A_0,i,v+1} - \hat{\sigma}_{A_0,i,v})(\hat{\mu}_{A_k,i,b} - \hat{\mu}_{A_0,i,v+1})}{\hat{\sigma}_{A_k,i,b}(\hat{\mu}_{A_0,i,v+1} - \hat{\mu}_{A_0,i,v})} + \frac{\hat{\sigma}_{A_0,i,v}}{\hat{\sigma}_{A_k,i,b}} - 2^k\right|,$$

where

- A_k is the noise curve corresponding to the i-th input image (denoted as $\mathbf{x}$) after applying the down-scaling operator k times. For example, if $k = 2$ then A corresponds to the curve of the noise estimation of $\mathbf{SS}\tilde{\mathbf{x}}$.
- i is the image index, for the raw images in Figure 5, $1 \leq i \leq |I|$, where $|I|$ is the number of images. $|I| = 8$ images were used.
- b is the bin index, $1 \leq b \leq |B_k|$ where $|B_k|$ is the number of bins of the noise curve at scale k. For the test images $|B_0| = 49$, $|B_1| = 12$ and $|B_2| = 3$ bins are used.
- v is the index of the bin in the curve A_0 such that $\hat{\mu}_{A_0,i,v} \leq \hat{\mu}_{A_k,i,b} < \hat{\mu}_{A_0,i,v+1}$ (see Figure 10).

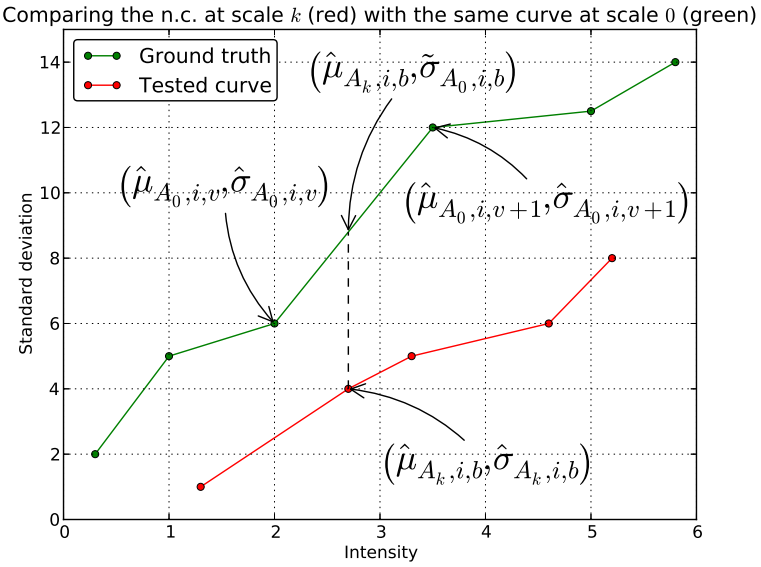

Figure 10: Checking a noise curve A_k at scale k (red) against the noise curve A_0 of the same image at scale 0 (green), where i is the index of the image, b is the index of the bin, $(\hat{\mu}_{A_k,i,b}, \hat{\sigma}_{A_k,i,b})$ are the control points of the noise curve of the sub-scaled image, $(\hat{\mu}_{A_0,i,v}, \hat{\sigma}_{A_0,i,v})$ are the control points of the noise curve of the image at scale 0, and $\tilde{\sigma}_{A_0,i,b}$ is the standard deviation value projected from A_k onto A_0.

Remark: since the noise should be divided by two when the operator $\mathbf{S}$ is applied and an ideal noise estimator is used, the relation $\frac{\tilde{\sigma}_{A_0,i,b}}{\hat{\sigma}_{A_k,i,b}}$ between the standard deviation estimations at scale 0 and scale k (applying k times $\mathbf{S}$) should be equal to 2^k. The error $E^{(5)}_{A_0,A_k,i,b}$ measures, for the tested noise estimator, the absolute deviation from the ideal value 2^k at each bin. To get a mean estimation of the error $E^{(5)}_{A_0,A_k,i,b}$ for all the test images and bins in Figure 5, we define another error function as

$$E^{(6)}_{A_0,A_k} := \frac{1}{|I||B_k|}\sum_{i=1}^{|I|}\sum_{b=1}^{|B_k|} E^{(5)}_{A_0,A_k,i,b}.$$

A method has a perfect scale coherence if $E^{(6)}$ is zero. Table 5 gives the obtained mean down-scale error $E^{(6)}_{A_0,A_k}$ for the raw images in Figure 5 depending on the scale k. The measurements are done for one of the two green channels of the raw images. The results are significantly worse than those given by the Ponomarenko et al. [6] and the Percentile [5] methods. The reason is that the PCA method needs 112000 samples/bin while Ponomarenko et al. and the Percentile need 42000 samples/bin (see Section 6.2). When the image is down-scaled it becomes too small for the PCA method to get an accurate estimation, since the minimum number of samples/bin is not attained and only one bin is used for the third and fourth scales.

k	1	2	3
$E^{(6)}_{A_0,A_k}$	0.627	2.407	4.997

Table 5: Evaluation $E^{(6)}_{A_0,A_k}$ for the raw images in Figure 5 depending on the scale k. The measurements are done for one of the two green channels of the raw images.

8 Complexity Analysis of the Algorithms

In function *GetNextEstimate* (in Algorithm 1) the most expensive part is the *ApplyPCA* function that is iterated several times until convergence is reached. The *ApplyPCA* function consists of:

- Computing the sample covariance matrix (Equation (6)). If computed directly, it comprises about $|B(p)|M^2$ operations. Note that we divide by $|B(p)|$ instead of $|B(p)|-1$ and thus (6) is a *biased* estimator of the variance. Since the number of samples is finite, it should be better to use an unbiased estimator, but we do not want to change the original method and thus we keep the same formulas and code used in [18].

- Computing the eigenvalues of the sample covariance matrix. Without any optimization, the number of operations is proportional to M^3, where we recall that M is the number of pixels in each patch. Again, since M is fixed, the cost of this step is constant, $O(1)$.

With the optimization explained in [18], the (biased) sample covariance matrix can be expressed as

$$\frac{1}{|B(p)|}\left(C_{B(p)} - \frac{1}{|B(p)|}\mathbf{c}_{B(p)}\mathbf{c}^T_{B(p)}\right),$$

where we recall that $C_X := \sum_{\mathbf{y}_i \in X} \mathbf{y}_i\mathbf{y}_i^T$ and $\mathbf{c}_X := \sum_{\mathbf{y}_i \in X} \mathbf{y}_i$.

Since the optimized algorithm first pre-computes the matrix $C_{B(1-j\Delta p)}$ and the vector $\mathbf{c}_{B(1-j\Delta p)}$ for $j = n-1, \ldots, 0$, if the worst case is considered ($j = 0$), then $B(1-j\Delta p) = B(1)$, that is, all the N blocks in the images would be used. The matrices and vectors are computed at most $\frac{1}{\Delta p} = \frac{1}{0.05} = 20$ times. To get the subsets of $B(1 - j\Delta p)$ according to j it suffices to call just once the *argsort* function, implemented with the Quicksort algorithm, that has a complexity $O(N \log N)$. Assuming that the matrices $C_{B(p)}$ and vectors $\mathbf{c}_{B(p)}$ and computed 20 times in the worst case scenario, i.e. $p = 1$, we have that $C_{B(p)} = C_{B(1)} = \sum_{\mathbf{y}_i \in B(1)} \mathbf{y}_i\mathbf{y}_i^T$ is executed with complexity $O(|B(1)|) = O(N)$. Similarly, we obtain that $C_{B(1)}$ is computed in $O(N)$. In summary, the computation of the sample covariance matrix consists in the execution of the *argsort* operation with complexity $N \log N$ and then looping at most 20 times through a loop that executes operations with complexity $O(N)$. Therefore, the complexity of function *GetNextEstimate* (in Algorithm 1) is $N \log N$, being N the number of overlapping $M \times M$ blocks in the image.

Algorithm 3 first executes the *argsort* (implemented with the Quicksort algorithm) operation with complexity $O(N \log N)$. The loop simply copies pixel values, and has linear complexity $O(N)$. Therefore, Algorithm 3 is executed with complexity $O(N \log N)$. Algorithm 4, is simply an *if* comparison and simple arithmetic operations. Therefore, Algorithm 4 is executed in constant time $O(1)$. Algorithm 6 loops over the number of bins of the noise curve and inside the loop Algorithm 4 is called. Since Algorithm 4 is executed in constant time $O(1)$, Algorithm 6 has a linear complexity $O(|B|)$, where $|B|$ is the number of bins. Algorithm 2 loops over all possible pixels in the image (with the exception of the boundary of the image). The loop iterates through all the channels of the image and looks for groups of four connected pixels. Therefore, the inner loop is executed in linear time with the number of channels, $O(\text{num\_channels})$. Since the number of channels is small, it can actually be considered executed in constant time $O(1)$ once the number of channels has been fixed. The complexity of Algorithm 2 is given by its main loop, that is executed in linear time with complexity $O(N)$.

9 Online Demo

An online demo is available for this algorithm in the IPOL web part of this article[3]. The users can upload any image to measure its noise. The demo also offers several types of pre-uploaded images to test the algorithm:

- Raw images obtained by splitting the raw channels R, G_1, G_2, B and leaving out the G_2 channel. Then, an RGB image is formed by using the R, G_1, B channels. Since in the raw image no gamma correction has been done yet, the values of the image are multiplied by 32 to increase their dynamics and screen visibility. The colors of these images are not quite adapted to human visualization, because no white balance has been applied to them.
- The JPEG versions of the same raw images, as they are encoded by the camera.
- Various JPEG images.
- High SNR raw images, down-scaled by a factor eight with their color channels averaged, so that they are nearly noiseless. In the demo they are referred to as "no noise" images.

Once an image has been chosen the following parameters can be configured:

- **Treatment of groups (2×2) of equal pixels**. It allows to choose between ignoring the blocks that contain a group of four equal pixels in any channel (default), or using all the blocks (see Section 6.1) unconditionally.
- **Curve filter iterations**. It indicates the number of filtering iterations that are applied to filter the noise curve (see Section 6.3). Default: five iterations.
- **Number of bins**. It is the number of bins in the noise curve (see Section 6.2). If it is set to *automatic selection*, each bin will contain approximately 112000 samples/bin.
- **A and B noise parameters**. Add a simulated noise with variance $A + B\mathbf{x}$ to the input image. If $A = B = 0$ no noise will be added. If $B = 0$ uniform noise with variance A will be added. Default: $A = B = 0$.

Note: since normally the last bin contains less samples than the other bins, to prevent a bad estimation with the chosen number of bins, the demo estimates the noise with an extra bin, which is discarded at the end.

[3] https://doi.org/10.5201/ipol.2016.124

9.1 Subtraction of the Quantization Noise

In the online demo, all the images are encoded using 8 bits/pixel/channel. An error is added when the continuous value is encoded (truncated) using an integer. For example, if the real value of the intensity is 123.45, it would be encoded as 123.00, or if the real value is 123.78, it would be encoded as 124.00. This error can be understood as a noise whose distribution is uniform. This quantization error over the noise being estimated must be subtracted, since it represents an added extra noise which is introduced when encoding the image. Thus, this correction is only applied to the demo, not in the noise estimator code. Indeed, the variance of a uniform random variable is $\sigma_q^2 = \int_{-\frac{1}{2}}^{\frac{1}{2}} (x - \bar{x})^2 \, \mathrm{d}x = \int_{-\frac{1}{2}}^{\frac{1}{2}} x^2 \, \mathrm{d}x = \left[\frac{x^3}{3}\right]_{-\frac{1}{2}}^{\frac{1}{2}} = \frac{1}{12}$. This is the variance of the quantization error that must be subtracted at each scale. The standard deviation of the noise is computed at each bin as the square root of the noise variance computed directly by the algorithm minus the variance of the (independent) quantization error. At each scale k the variance is divided by 4^k and thus the corrected standard deviation of the noise given by the demo is

$$\tilde{\sigma}_k = \sqrt{\hat{\sigma}_k^2 - \frac{\sigma_q^2}{4^k}} = \sqrt{\hat{\sigma}_k^2 - \frac{1}{4^k 12}}.$$

9.2 Example: Raw Image IMG_0177 (ISO 1250, t=1/30 s)

The results of this example can be reproduced by estimating the noise of the raw image without adding any noise (set $A = B = 0$) and choosing the raw image IMG_0177. The rest of the parameters are the default parameters of the demo. Figure 11 shows the input noisy image. Figure 12 shows the

Figure 11: Input noisy raw image IMG_0177.

noise estimated in the two first scales of the signal-dependent noise in IMG_0177 image. Left: scale S_0 (original image). Right: scale S_1. Note that, as expected, the noise standard deviation is divided by approximately two when down-scaling the image by two.

Acknowledgments

Work partially supported by DxO-Labs, the Centre National d'Etudes Spatiales (CNES, MISS Project), the European Research Council (Advanced Grant Twelve Labours), and the Office of Naval Research (Grant N00014-97-1-0839).

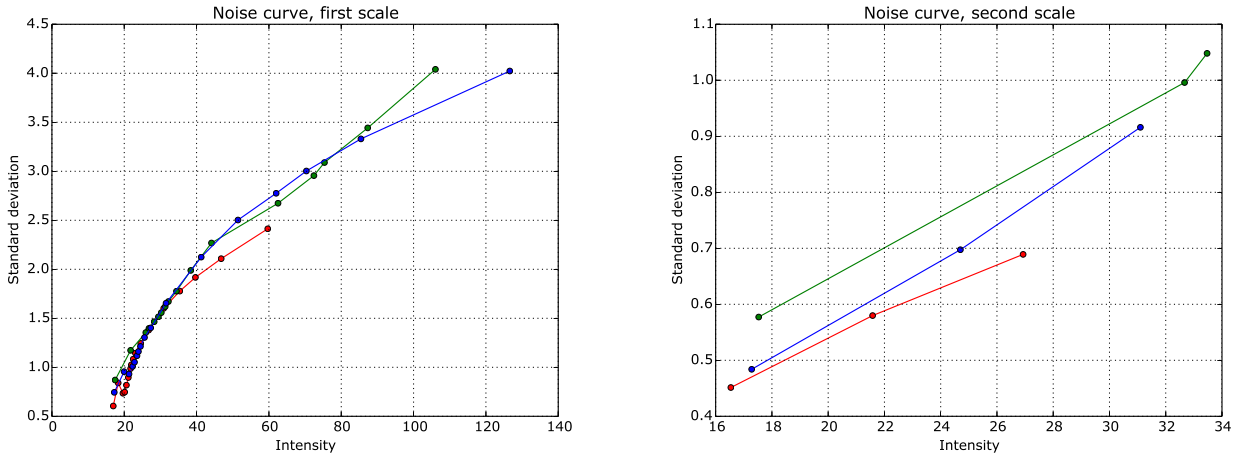

Figure 12: The noise estimated for the two first scales of the noisy raw image IMG_0177. Left: scale S_0 (first scale, original image). Right: scale S_1.

Image Credits

 Test image obtained from the USC-SIPI Image Database[4].
Rest of images, CC-BY by Miguel Colom.

References

[1] F. J. Anscombe, *The transformation of Poisson, binomial and negative-binomial data*, Biometrika, 35 (1948), pp. 246–254. https://doi.org/10.2307/2332343.

[2] A. Buades, B. Coll, and J.M. Morel, *A review of image denoising algorithms, with a new one*, Multiscale Modeling and Simulation, 4 (2006), pp. 490–530. https://doi.org/10.1137/040616024.

[3] A. Buades, B. Coll, and J.M. Morel, *Non-Local Means Denoising*, Image Processing On Line, 1 (2011). https://doi.org/10.5201/ipol.2011.bcm_nlm.

[4] A. Buades, Y. Lou, J.M. Morel, and Z. Tang, *A note on multi-image denoising*, in Proceedings of the International Workshop on Local and Non-Local Approximation in Image Processing, 2009, pp. 1–15. https://doi.org/10.1109/LNLA.2009.5278408.

[5] 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.

[6] ——, *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.

[4] http://sipi.usc.edu/database/database.php?volume=misc&image=12

[7] M. Colom, A. Buades, and J.M. Morel, *Nonparametric noise estimation method for raw images*, Journal of the Optical Society of America A, 31 (2014). https://doi.org/10.1364/JOSAA.31.000863.

[8] A. Danielyan and A. Foi, *Noise variance estimation in nonlocal transform domain*, in Proceedings of IEEE International Workshop on Local and Non-Local Approximation in Image Processing (LNLA), 2009, pp. 41–45. https://doi.org/10.1109/LNLA.2009.5278404.

[9] C.A. Deledalle, J. Salmon, and A. Dalalyan, *Image denoising with patch based PCA: local versus global*, in Proceedings of the British Machine Vision Conference, BMVA Press, 2011, pp. 25.1–25.10. https://doi.org/10.5244/C.25.25.

[10] J. Immerkaer, *Fast noise variance estimation*, Computer Vision and Image Understanding, 64 (1996), pp. 300–302. https://doi.org/10.1006/cviu.1996.0060.

[11] 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.

[12] M. Lebrun and A. Leclaire, *An Implementation and Detailed Analysis of the K-SVD Image Denoising Algorithm*, Image Processing On Line, 2 (2012), pp. 96–133. https://doi.org/10.5201/ipol.2012.llm-ksvd.

[13] J.S. Lee, *Refined filtering of image noise using local statistics*, Computer Graphics and Image Processing, 15 (1981), pp. 380–389. https://doi.org/10.1016/S0146-664X(81)80018-4.

[14] M. Makitalo and A. Foi, *Optimal inversion of the Anscombe transformation in low-count Poisson image denoising*, IEEE Transactions on Image Processing, 20 (2011), pp. 99–109. https://doi.org/10.1109/TIP.2010.2056693.

[15] P. Meer, J.M. Jolion, and A. Rosenfeld, *A fast parallel algorithm for blind estimation of noise variance*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (1990), pp. 216–223. https://doi.org/10.1109/34.44408.

[16] S.I. Olsen, *Estimation of noise in images: an evaluation*, Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing, 55 (1993), pp. 319–323. https://doi.org/10.1006/cgip.1993.1022.

[17] N.N. Ponomarenko, V.V. Lukin, S.K. Abramov, K.O. Egiazarian, and J.T. Astola, *Blind evaluation of additive noise variance in textured images by nonlinear processing of block DCT coefficients*, in Proceedings of the International Society for Optics and Photonics. Electronic Imaging, Image Processing: Algorithms and Systems II, vol. 5014, 2003, pp. 178–189. https://doi.org/10.1117/12.477717.

[18] S. Pyatykh, J. Hesser, and L. Zheng, *Image Noise Level Estimation by Principal Component Analysis*, IEEE Transactions on Image Processing, (2012), pp. 687–699. https://doi.org/10.1109/TIP.2012.2221728.

[19] T. Rabie, *Robust estimation approach for blind denoising*, IEEE Transactions on Image Processing, 14 (2005), pp. 1755–1765. https://doi.org/10.1109/TIP.2005.857276.

[20] B. Rajaei, *An Analysis and Improvement of the BLS-GSM Denoising Method*, Image Processing On Line, 4 (2014), pp. 44–70. https://doi.org/10.5201/ipol.2014.86.

[21] K. Rank, M. Lendl, and R. Unbehauen, *Estimation of image noise variance*, in Vision, Image and Signal Processing, vol. 146, 1999, pp. 80–84. https://doi.org/10.1049/ip-vis:19990238.

[22] G. Yu and G. Sapiro, *DCT Image Denoising: a Simple and Effective Image Denoising Algorithm*, Image Processing On Line, 1 (2011). https://doi.org/10.5201/ipol.2011.ys-dct.