IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2013–07–23.
Submitted on 2012–09–25, accepted on 2013–09–09.
ISSN 2105–1232 © 2013 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2013.45

Analysis and Extension of the Ponomarenko et al. Method, Estimating a Noise Curve from a Single Image

Miguel Colom[1], Antoni Buades[2]

[1] Universitat de les Illes Balears, Spain and CMLA, ENS Cachan, France (miguel.colom@cmla.ens-cachan.fr)
[2] Universitat de les Illes Balears, Spain (toni.buades@uib.es)

Abstract

In the article *An Automatic Approach to Lossy Compression of AVIRIS Images* N.N. Ponomarenko et al. propose a new method to specifically compress AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) images. As part of the compression algorithm, a noise estimation is performed with a proposed new algorithm based on the computation of the variance of overlapping 8×8 blocks. The noise is estimated on the high-frequency orthonormal DCT-II coefficients of the blocks. To avoid the effect of edges and textures, the blocks are sorted according to their energy measured on a set of low-frequency coefficients. The final noise estimation is obtained by computing the median of the variances measured on the high-frequency part of the spectrum of the blocks using only those whose energy (measured on the low-frequencies) is low. A small percentile of the total set of blocks (typically the 0.5%) is used to select those blocks with the lower energy at the low-frequencies. 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 CCD device in a digital camera.

Source Code

The C++ implementation of the Ponomarenko et al. noise estimator version 3.0 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 page of this article[1].

Keywords: noise, noise estimation, gaussian noise, DCT

1 Noise Estimation Method

1.1 Notation and Terminology

This section prepares the detailed description of the noise estimation algorithm given in section 1.2 by fixing its notation and terminology.

[1] http://dx.doi.org/10.5201/ipol.2013.45

- $\mathbf{U}$: the noiseless ideal image.
- $\tilde{\mathbf{U}}$: the discrete noisy image of $\mathbf{U}$.
- N_x, N_y: the width and height of $\tilde{\mathbf{U}}$ in pixels.
- $\tilde{\mathbf{U}}(x, y)$: the gray-level value of $\tilde{\mathbf{U}}$ at pixel (x, y), $x \in [0, N_x - 1]$ and $y \in [0, N_y - 1]$.
- $\mathbf{W}(x, y)$: a $w \times w$ pixels block in $\tilde{\mathbf{U}}$, $\mathbf{W}(x, y) = \{\tilde{\mathbf{U}}(x+i, y+j) : i \in [0, w-1], j \in [0, w-1], x \in [0, N_x - w + 1], y \in [0, N_y - w + 1]\}$.
- w: the side of the overlapping $w \times w$ pixels blocks $\mathbf{W}(x, y)$.
- M: the total number of overlapping blocks. $M = (N_x - w + 1)(N_y - w + 1)$.
- $\mathbf{D}_{x,y}$: the result of applying the orthonormal 2D DCT-II to a block $\mathbf{W}(x, y)$. Its coefficients are $\mathbf{D}_{x,y}(i, j)$ and the transform is defined as

$$\mathbf{D}_{x,y}(i,j) = Q_w(i)Q_w(j) \sum_{n_x=0}^{w-1} \sum_{n_y=0}^{w-1} \mathbf{W}(x+n_x, y+n_y) \cos\left[\frac{\pi}{w}\left(n_x + \frac{1}{2}\right) i\right] \cos\left[\frac{\pi}{w}\left(n_y + \frac{1}{2}\right) j\right],$$

  with $x \in [0, N_x - w - 1], y \in [0, N_y - w - 1], i \in [0, w - 1], j \in [0, w - 1]$ and $Q_N(k)$ is a normalization factor

$$Q_N(k) = \begin{cases} \frac{1}{\sqrt{N}}, & k = 0 \\ \sqrt{\frac{2}{N}}, & k \neq 0 \end{cases}.$$

  From now on the DCT operator will refer specifically to this definition of the orthonormal 2D DCT-II.

- $\mathbf{W}_m$: a $w \times w$ pixels block in $\tilde{\mathbf{U}}$ according to its index m in a list of overlapping blocks $\mathbf{W}_m = \mathbf{W}(x, y)$ where $y = \left\lfloor \frac{m}{N_x - w + 1} \right\rfloor$, $x = m - y(N_x - w + 1), m \in \{0, 1, \ldots, M - 1\}$.
- $\mathbf{D}_m(i, j) = \mathrm{DCT}(\mathbf{W}_m)$ where m is the index of the block.
- T: threshold used by the function $\delta(i, j)$ that labels the coefficients of the transformed blocks $\mathbf{D}_{x,y}(i, j)$ as *low-frequency* coefficients (see section 1.2.2 for more details).

1.2 The Algorithm

1.2.1 Step 1: Computing the Set of Transformed Blocks $\{\mathbf{D}_m(i, j)\}$

From an image $\tilde{\mathbf{U}}$ of width N_x and height N_y, corrupted with additive white Gaussian noise of variance σ^2, it is extracted a set of $M = (N_x - w + 1)(N_y - w + 1)$ (overlapping) $w \times w$ blocks $\{\mathbf{W}_m\}$, where m is the index of the block, $m \in \{0, 1, \ldots, M - 1\}$. Many noise estimation algorithms [2, 4, 6, 9] compute local estimates of the noise variance in small blocks that are used for a final statistical estimation (median, average, percentile, ...). Unlike other methods that pre-filter the image before extracting noise variance information from the blocks [5], the Ponomarenko et al. method measures the variance directly on $\mathbf{W}_m$. The DCT of each of these blocks is computed and gives the set $\{\mathbf{D}_m\}$ of transformed blocks. The DCT coefficients in each block are denoted by $\mathbf{D}_m(i, j)$ where m is the index of the block and $0 \leq i, j < w$ is the frequency pair associated to that coefficient.

1.2.2 Step 2: Defining a Function to Label the Low/High Frequency Coefficients

The algorithm labels coefficients of the transformed blocks as belonging to low or medium/high frequencies. A coefficient corresponds to a low frequency if and only if $\delta(i,j) = 1$. If not, it is labeled as belonging to the medium/high frequencies set, where δ is defined by

$$\delta(i,j) = \begin{cases} 1, & (i+j < T) \wedge (i+j \neq 0), \rightarrow \text{low frequencies} \\ 0, & (i+j \geq T) \vee (i+j = 0) \rightarrow \text{medium/high frequencies.} \end{cases}$$

where T is a given threshold, and $\wedge$ and $\vee$ stand for the *AND* and *OR* logical operators, namely. Note that this function does not label the mean of the block term $(i+j=0)$ as a low-frequency.

1.2.3 Step 3: Estimating the Block Empirical Variance only with the Low-Frequency Coefficients

Given the set of transformed blocks $\{\mathbf{D}_m\}$ with $m = 0, 1, \ldots, M-1$ the set of (empirical) variances associated to the low-frequency coefficients of the block m is defined as

$$\mathbf{V}_m^L = \frac{1}{\theta} \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} [\mathbf{D}_m(i,j)]^2 \, \delta(i,j),$$

where $\theta = \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} \delta(i,j)$ is the adequate normalization factor to get a mean.

1.2.4 Step 4: Computing the Empirical Variance of the High-Frequency Coefficients

The set of transformed blocks $\{\mathbf{D}_0, \ldots, \mathbf{D}_m, \ldots, \mathbf{D}_{M-1}\}$ is rewritten with respect to the corresponding value of $\mathbf{V}_m^L$ in ascending order. The block that gives the lowest low-frequency variance will be noted as $\mathbf{D}_{(0)}$, the next with the lowest low-frequency variance as $\mathbf{D}_{(1)}$ and so on. The block with the highest low-frequency variance is $\mathbf{D}_{(M-1)}$. Given the list of sorted blocks $\{\mathbf{D}_{(m)}\}$, the noise variance estimate associated with the high-frequency coefficient at (i,j) is defined by

$$\mathbf{V}^H(i,j) = \frac{1}{K} \sum_{k=0}^{K-1} \left[\mathbf{D}_{(k)}(i,j)\right]^2,$$

where $i+j \geq T$ and $K = \lfloor pM \rfloor, p < 1$ is the position of the p-quantile in the list $\{\mathbf{D}_{(m)}\}_{m \in [0, M-1]}$. Note that this empirical variance estimate is made with the list of the K transformed blocks whose empirical variance as measured in their low-frequencies is lowest. It is understood that these blocks are likely to contain only noise. Thus their high frequencies are good candidates to estimate the noise. Noise in high and low frequencies is uncorrelated and since most of the energy of the ideal image is concentrated in the low and medium frequency coefficients (because of the sparsity of most natural images), one can assume that $\mathbf{V}^H(i,j)$ gives an accurate estimation of the noise variance. However, if the image is highly textured, those high-frequency coefficients might give a variance that is explained by the textures of the image and not by the noise.

1.2.5 Step 5: Choosing the Best K and Obtaining the Final Noise Estimate

The final noise estimation is given by the median of the variance estimates $\mathbf{V}^H(i,j)$,

$$\hat{\sigma} := \sqrt{\text{median}_{i,j} \left(\{ \mathbf{V}^H(i,j) \mid i+j \geq T \} \right)}.$$

However, the values in the list $\{\mathbf{V}^H(i,j)\}$ depend on the value of the quantile K. Ponomarenko et al. [8] propose to use the following adaptive strategy to find out the best value for K:

1. Set $K = \sqrt{M}$. The original setting is $K = M/512$, because the algorithm is designed to work with AVIRIS images of size 512×677. In order to be able to use any size of image, we propose to set $K = \sqrt{M}$.
2. Compute an upper bound of noise variance as $A = 1.3\mathbf{V}^L_{K/2}$.
3. Determine a new $K = m_{\min}$, where $m_{\min}$ is the value of m that minimizes $|A - \mathbf{V}^L_m|$.
4. Repeat seven times the steps 2 and 3.
5. Set $A = A/5$.

Nevertheless, we found that fixing directly a small percentile equal to 0.5% of the set of variances gives more accurate and reliable results than the above procedure. This is the only place where our implementation differs from the original algorithm. The complete algorithmic description of the original method is summarized in algorithm 1. The modified version of the algorithm that uses a fixed percentile $p = 0.5\%$ instead of the iterations to find the value of K is given in algorithm 2.

For a review of several noise estimation methods we refer the reader to the *Secrets of image denoising cuisine* [3] and *Estimation of noise in images: an evaluation* [7] articles.

2 Extensions of the Original Method

2.1 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 it does not depend on the intensity of the pixels of the ideal image. This assumption is not realistic because of the physical nature of light and the way a CCD 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 $\mathbf{U}(x,y)$. The Poisson noise has therefore a standard deviation of $\sqrt{\mathbf{U}(x,y)}$. 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 $\mathbf{U}$ 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.

For a signal dependent noise, a "noise curve" must be established. This noise curve associates with each image value $\mathbf{U}(x,y)$ an estimation of the standard deviation of the noise associated with this value. Thus, for each block in the image, its mean must be computed and will give an estimation of a value in $\mathbf{U}$. The measurement of the variation of the block (for example, its variance) will also be stored. 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. For the Ponomarenko et al. algorithm, the chosen minimum was 42000 elements/bin. These measurements allow for the construction of a list of block variances whose corresponding means belong to the given bin. Therefore, it is possible

Algorithm 1 Pseudo-code for the Ponomarenko et al. noise estimation algorithm.

NOISE_ESTIMATION - Returns the standard deviation of the white Gaussian noise of the input image.

Input $\tilde{\mathbf{U}}$: noisy image.

Output $\tilde{\sigma}$: estimated standard deviation of its noise.

$w = 8$.

$T = 9$.

$N_x = \text{width}(\tilde{\mathbf{U}})$

$N_y = \text{height}(\tilde{\mathbf{U}})$

$M = (N_x - w + 1)(N_y - w + 1)$: number of (overlapping) blocks in $\tilde{\mathbf{U}}$.

$\mathbf{W} \leftarrow$ all M possible $w \times w$ (overlapping) blocks in $\tilde{\mathbf{U}}$.

$\mathbf{D} \leftarrow \text{DCT}(\mathbf{W})$. 2D orthonormal DCT-II of the $w \times w$ blocks in $\mathbf{W}$.

$\delta[i,j] = 1$ if $(i + j < T) \wedge (i + j \neq 0)$ else 0, $\forall (i,j) \in [0, w-1]^2$.

$\theta = \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} \delta[i,j]$.

$\mathbf{V}_m^L = \frac{1}{\theta} \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} [\mathbf{D}_m(i,j)]^2 \, \delta[i,j]$

$K = \sqrt{M}$.

for $n = 1 \ldots 7$ **do**

    $A = 1.3\mathbf{V}_{K/2}^L$.

    $K = \text{argmin}_m \left(|A - \mathbf{V}_m^L| \right)$.

end for

$K = K/5$.

$\mathbf{I} = \text{sort}_m(\mathbf{V}_m^L)$. $\mathbf{I}$ contains the sorting indices.

$\mathbf{V}^H[i,j] = \frac{1}{K} \sum_{k=0}^{K-1} \mathbf{D}^2_{\mathbf{I}[k]}(i,j)$, where $\mathbf{V}^H[i,j]$ is defined only for those $[i,j]$ such that $i + j \geq T$.

$\hat{\sigma} = \sqrt{\text{median}_{i,j} \left(\mathbf{V}^H(i,j) \right)}$

Algorithm 2 Pseudo-code for the Ponomarenko et al. noise estimation algorithm (using a fixed percentile).

NOISE_ESTIMATION - Returns the standard deviation of the white Gaussian noise of the input image.

Input $\tilde{\mathbf{U}}$: noisy image.

Output $\tilde{\sigma}$: estimated standard deviation of its noise.

1: $w = 8$.

2: $T = 9$.

3: $p = 0.005$.

4: $N_x = \text{width}(\tilde{\mathbf{U}})$

5: $N_y = \text{height}(\tilde{\mathbf{U}})$

6: $M = (N_x - w + 1)(N_y - w + 1)$: number of (overlapping) blocks in $\tilde{\mathbf{U}}$.

7: $\mathbf{W} \leftarrow$ all M possible $w \times w$ (overlapping) blocks in $\tilde{\mathbf{U}}$.

8: $\mathbf{D} \leftarrow \text{DCT}(\mathbf{W})$. 2D orthonormal DCT-II of the $w \times w$ blocks in $\mathbf{W}$.

9: $\delta[i,j] = 1$ if $(i + j < T) \wedge (i + j \neq 0)$ else 0, $\forall (i,j) \in [0, w-1]^2$.

10: $\theta = \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} \delta[i,j]$.

11: $K = pM$. Get a p-quantile of the list of variances. Typically $p = 0.005 \Rightarrow$ the 0.5%-percentile.

12: $\mathbf{V}_m^L = \frac{1}{\theta} \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} \left[\mathbf{D}_m(i,j)\right]^2 \delta[i,j]$.

13: $\mathbf{I} = \text{sort}_m(\mathbf{V}_m^L)$. $\mathbf{I}$ contains the sorting indices.

14: $\mathbf{V}^H[i,j] = \frac{1}{K} \sum_{k=0}^{K-1} \mathbf{D}^2_{\mathbf{I}[k]}(i,j)$, where $\mathbf{V}^H[i,j]$ is defined only for those $[i,j]$ such that $i + j \geq T$.

15: $\hat{\sigma} = \sqrt{\text{median}_{i,j}\left(\mathbf{V}^H(i,j)\right)}$

to apply the Ponomarenko noise estimation algorithm to each set of blocks associated with a given bin. In this way, an estimation of the noise for the intensities inside the limits of the bin is obtained. Because the set of bins is disjoint and there is no gap between bins, it is possible to deduce by interpolation a curve that relates the means of the blocks with their standard deviation, hence obtaining a signal-dependent *noise curve*. Note that if the number of bins is exactly one, the original signal-independent white Gaussian noise estimator [8] using a fixed percentile is obtained. To choose the intensity value associated with each bin, the median of the means of all blocks inside each bin is computed. 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 at the variable "samples_per_bin" the integer value of N/bins. In general, samples_per_bin = 42000 samples/bin. Since the last bin contain 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.

2.2 Filtering the Noise Curve

Optionally, the noise curve obtained on real images can be filtered. Indeed, it may present peaks when some given gray level interval contains mostly means of blocks belonging to a highly textured region. In this case, the measured block variance would be caused by the signal itself and not by the noise and the noise variance would be overestimated.

Given the i-th control point of the noise curve $(\hat{\mu}_i, \hat{\sigma}_i)$ a closed intensity interval centered at this bin is considered, that is, $[\hat{\mu}_i - D, \hat{\mu}_i + D]$. For each intensity μ inside the interval (assuming that μ starts at $\hat{\mu}_i - D$ and it is incremented with a step of 0.05 while it is less or equal to $\hat{\mu}_i + D$), it is obtained the interpolated standard deviation that corresponds to each intensity μ. In order to avoid an excessive interpolation, if $\hat{\mu}_i - D < \hat{\mu}_0$ for the i-th bin, then the diameter D is changed to the value $\hat{\mu}_b - \hat{\mu}_0$. In the same way, if $\hat{\mu}_i + D > \hat{\mu}_{B-1}$ (being B the number of bins), then the diameter D is changed to the value $\hat{\mu}_{B-1} - \hat{\mu}_b$. Since each $\hat{\mu}_i$ can be seen as an oscillation (given by the RMSE) around the ideal value, averaging the noise curve inside the interval $[\hat{\mu}_i - D, \hat{\mu}_i + D]$ for each control point attenuates the oscillations and puts them closer to the ground-truth. Once the oscillations have been attenuated, it might happen that a control point corresponds to a peak caused by a texture. In that case, the action taken is to compute the average inside the interval $[\hat{\mu}_i - D, \hat{\mu}_i + D]$ and to substitute the standard deviation $\hat{\mu}_i$ of the i-th control point by the average only if it is *lower* than the average of the intensities in the interval. In practice, the filtering procedure is iterated five times. In the first three iterations the control points are allowed to go up and down, thus canceling the oscillations around the ideal value. In the next two iterations the points are only allowed to go down, to attenuate the overestimation of the noise because of textures. The simple strategy presented here performs properly for most natural images and in general not more than five filtering

Algorithm 3 Algorithm classifying blocks by their means.

CLASSIFY_BY_MEAN - Splits the input elements into disjoint *bins* according to the mean of the elements trying that each bin has the same cardinality.
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 at bin b.
Output datal_bins[b]: list of means of the elements at bin b.

```
samples_per_bin = ⌊N/bins⌋
limits_begin = zeros(bins)
limits_end = zeros(bins)
num_elements = zeros(bins)
data_bins = array(bins)
datal_bins = zeros(bins)
buffer = array(N)
bufferl = zeros(N)
indices = argsort(datal, N)                          ▷ Sort data by datal
min_datal = datal[indices[0]]                        ▷ Min and max
max_datal = datal[indices[N-1]]
lim0 = min_datal
elements_count = 0
bin = 0
for idx = 0 ... N do
    if idx == N then
        finished_loading = true
    else
        lim1 = datal[indices[idx]]
        finished_loading = ¬ (bin == bins - 1) ∧ (elements_count ≥ samples_per_bin)
    end if
    if finished_loading then
        data_bins[bin] ← buffer
        datal_bins[bin] ← bufferl
        limits_begin[bin] = lim0          ▷ Update limits and number of elements of the bin
        limits_end[bin] = lim1
        num_elements[bin] = elements_count
        lim0 = lim1                                  ▷ Prepare for the next element
        bin = bin + 1
        elements_count = 0
    else
        buffer[elements_count] = data[indices[idx]]                ▷ Keep loading...
        bufferl[elements_count] = datal[indices[idx]]
        elements_count = elements_count + 1
    end if
end for
limits_end[bins-1] = max_datal
```

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 diameter $D = 7$ 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.

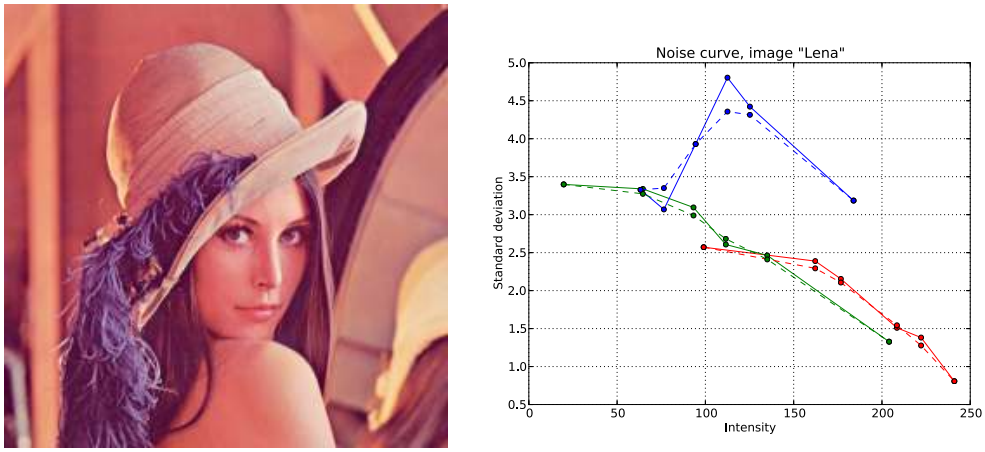


Figure 1: Left: test image *Lena* used to compare the noise curves with and without filtering. Right: noise curve for *Lena*. The non-filtered curve is drawn with solid lines and the filtered curve (five iterations) with dashed lines, using $D = 7$, $p = 0.5\%$, $w = 8$ and 6 bins. Note that the peak in the blue channel has decreased.

Figure 1 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 $D = 7$, $p = 0.5\%$, $w = 8$ and 6 bins. Note that the peak in the blue channel has decreased.

Algorithm 4 Obtain 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.

1: $\varepsilon = 10^{-6}$
2: **if** $|\mu_c - \mu_e| < \varepsilon$ **then**
3: $\quad s = 0$ ▷ Avoid dividing by zero
4: **else**
5: $\quad s = \frac{\sigma_c - \sigma_e}{\mu_c - \mu_e}$
6: **end if**
7: $\sigma = (\mu - \mu_e)s + \sigma_e$ **return** σ

2.3 Discarding Saturated Pixels

When the number of photons measured by the CCD during the exposure time is too high, its output may get saturated, and therefore underestimated. When the signal saturates the output of the CCD, the measured variance in the saturated areas of the image is zero. Figure 2 shows an image with

Algorithm 5 Interpolates an affine standard deviation from of the points of the given noise curve.

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.

$i = \text{argmin}_i\,(\mu_c[i] - \mu|)$ ▷ Find the nearest control point
$m = \mu_c[i]$
if $\mu < m$ **then** ▷ on the right of μ
  if $i = 0$ **then**
    $i = 1$ ▷ Treat boundary
    $m = \mu_c[i]$
  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
  $N = \text{len}(\mu_c)$
  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.

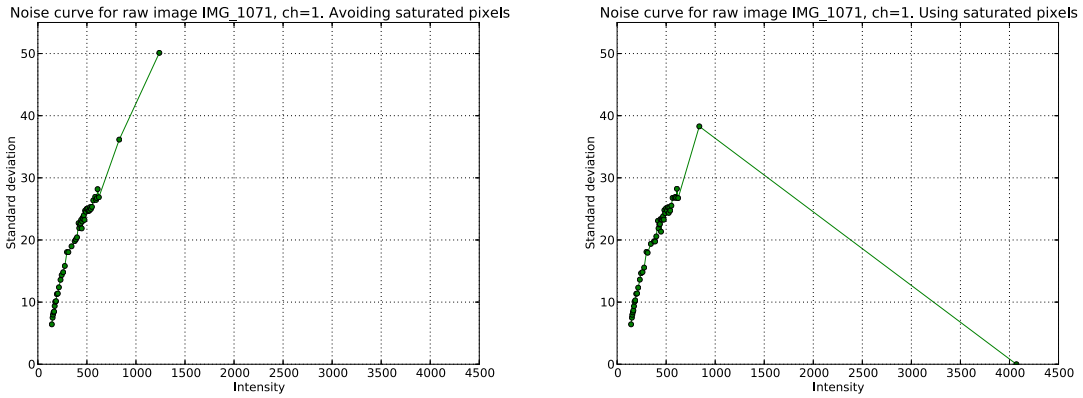
FILTER_CURVE
Input (μ_c, σ_c): list of control points to be filtered.
Input D: diameter.
Input allow_up: allow the points to go up and down. Otherwise, they are only allowed to go down.
Output σ_c^o: returned list filtered standard deviations

B = len(μ_c)
$\sigma_c^o \leftarrow \emptyset$
for $b = 0 \dots B-1$ **do**
    mu_current, std_current = $\mu_c[b], \sigma_c[b]$
    left = mu_current − D
    right = mu_current + D
    if left < $\mu_c[0]$ **then** ▷ Adjust the diameter for the points near the boundary
        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
    sum_window = 0 ▷ Add the interpolated control points inside the interval [left, right]
    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 /= L
    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

some saturated pixels. If the saturated pixels are taken into account when measuring the noise, the

Figure 2: Image with saturated pixels.

noise curve is no more reliable. Figure 3 shows a noise curve obtained when the saturated pixels are avoided in the noise estimation (left) and when they are used (right). In this estimation 8×8 blocks and 49 bins where used. Since the intensity of the saturated pixels is much higher than the intensity of most of the pixels in the image, there is usually a large intensity gap between the values of normal non saturated pixels and those saturated (from about 600 to approximately 4000 in figure 3) since the over-exposed pixels represent usually outlier values. Even if there are few pixels with an intensity ranging between 600 and 4000, the noise curve will interpolate the standard deviation between the curve value at mean 600 and the noise curve value at mean 4000. Of course, the information given by the noise curve inside this gap is not correct at all. Therefore, it is better to detect and remove the saturated points before the noise curve estimation. In general, the strategy used to discard



(a) Noise curve estimation avoiding the saturated pixels

(b) Noise curve estimation affected by the saturated pixels

Figure 3: Noise curve obtained when the saturated pixels are avoided in the noise estimation (a) and when they are taken into account (b). Using $w = 8$, $p = 0.5\%$ and 49 bins, blue channel.

saturated pixels is to avoid the blocks that contain a group of four connected exactly equal pixels, 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

special blocks. For example, JPEG encoding with a high compression factor sets to zero the value of the high-frequency coefficients in many of the 8×8 blocks. Among other undesired effects like lower frequency artifacts and blocking patterns, it can also create smooth zones, since high-frequency coefficients of the DCT of the block were set to zero by the JPEG encoder. 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 pseudo-code and the details on how the pixels should be connected can be found in algorithm 7.

Algorithm 7 Removal of equal pixels algorithm.

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

3 Evaluation of the Method

To evaluate the accuracy of the method, several kinds of tests were performed.

- Tests on simulated uniform Gaussian noise using the images of figure 4. In this case we have taken seven and also one bins to classify the blocks according to their means (see section 2.1).
- Tests on simulated signal-dependent Gaussian noise with variance $\sigma^2 = 5 + 0.3\tilde{\mathbf{U}}$ using the images shown in figure 4.
- Tests on a set of real raw images obtained by a Canon EOS 30D camera (see figure 5). The procedure explained in section 2.1 was used to get a noise curve. The results were compared to the ground-truth noise curve of the camera.

- Test on multi-scale 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.

3.1 Evaluation with Simulated Uniform Noise

In this experiment, uniform noise was simulated and then added to a set of ten noise-free images. Since the noise is perfectly known *a priori*, it 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. In order 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 reduce further 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 noise was reduced by a $5\sqrt{3} \simeq 8.66$ factor. Finally, the images, which already had a good SNR before being processed, can be considered noise-free.

Figure 4: Set of noise-free images used to test the noise estimation algorithm with uniform noise. From left to right and from top to bottom: bag, building1, computer, dice, flowers2, hose, lawn, leaves, stairs and traffic.

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

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

where $|I|$ is the number of images, i is the image index ($1 \leq i \leq |I|$), $|B|$ 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 the image i at the bin b. Table 1 shows the obtained $E^{(1)}_{i,\sigma}$ for each image i and each σ of the simulated noise. A new image is added to the set of noise-free images, the *flat image*, which is a constant image where all pixels have the value 127. This permits to test the response of the algorithm to a pure white noise image. It is apparent that the highly textured images create a significant error, particularly when little noise was added. Estimates of noise below 2 are therefore obviously clearly unreliable. All in all, the estimate is nevertheless quite reliable for values $\sigma > 5$. Seven bins are used. The last row is the RMSE obtained for a given σ and all the images. It is

denoted by $E_\sigma^{(2)}$ and defined as

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

Image / $E_{i,\sigma}^{(1)}$	$\sigma=1$	$\sigma=2$	$\sigma=5$	$\sigma=10$	$\sigma=20$	$\sigma=50$	$\sigma=80$
bag	0.86	0.48	0.28	0.59	0.45	1.10	1.05
building1	0.19	0.16	0.06	0.29	0.52	1.16	1.66
computer	0.20	0.10	0.19	0.21	0.48	2.26	1.82
dice	0.12	0.05	0.11	0.18	0.43	0.94	2.17
flowers2	0.19	0.08	0.13	0.30	0.50	1.60	0.89
hose	0.86	0.60	0.39	0.42	0.58	1.68	1.18
lawn	1.46	1.25	0.68	0.47	0.51	1.40	1.92
leaves	1.47	1.09	0.65	0.56	0.44	1.43	2.17
stairs	0.59	0.32	0.34	0.28	0.49	0.80	1.30
traffic	0.13	0.09	0.21	0.22	0.64	1.44	2.21
Flat image	0.02	0.03	0.05	0.17	0.37	1.34	1.23
$E_\sigma^{(2)}$	**0.56**	**0.39**	**0.28**	**0.34**	**0.49**	**1.38**	**1.60**

Table 1: This table shows the $E_{i,\sigma}^{(1)}$ RMSE after adding simulated noise to the set of noise-free images (figure 4) with several values of standard deviation σ. The last row is the $E_\sigma^{(2)}$ RMSE using the estimated $\hat{\sigma}_{i,b}$ of all the images. The percentile $p = 0.005$ and seven bins are used.

For completeness, the results corresponding to the estimation using just a single bin and with the iteration to fix the percentile K used by the original method (see algorithm 1) are shown in table 2, although this model of signal-independent noise using a single bin is not realistic at all.

Table 3 shows the obtained $E_\sigma^{(2)}$ RMSE depending on the number of the iterations of the noise curve filter (see section 2.2). Using five filtering iterations seems to be safe for any image with independence of the standard deviation of the noise, the kind of textures or the number and type of the edges the image may contain.

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

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

To get 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 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, one of the green channels can be discarded to get a single color pixel of each 2×2 block of the CFA, given an effective size of the color raw image of

[2]Color Filter Array.

Image / $E^{(1)}_{i,\sigma}$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$	$\sigma = 80$
bag	1.78	1.57	0.73	0.32	0.26	0.13	0.16
building1	0.16	0.09	0.02	0.10	0.07	0.04	0.30
computer	0.15	0.05	0.08	0.05	0.05	0.15	0.23
dice	0.11	0.06	0.01	0.03	0.07	0.51	0.29
flowers2	0.09	0.03	0.02	0.05	0.11	0.17	0.07
hose	0.75	0.36	0.24	0.17	0.06	0.44	0.22
lawn	1.86	0.42	0.75	0.27	0.37	0.09	0.00
leaves	3.04	1.30	0.55	0.67	0.44	0.41	0.05
stairs	1.06	0.85	0.45	0.23	0.14	0.41	0.27
traffic	0.10	0.07	0.08	0.13	0.11	0.47	0.27
Flat image	0.00	0.01	0.01	0.02	0.05	0.23	0.22
$E^{(2)}_{\sigma}$	**0.83**	**0.44**	**0.27**	**0.18**	**0.16**	**0.28**	**0.19**

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 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. It corresponds to the original signal-independent method, using a single bin and the iterations show in algorithm 1 to fix the percentile K.

Image / $E^{(2)}_{\sigma}$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$	$\sigma = 80$
No filtering	0.56	0.39	0.28	0.34	0.49	1.38	1.60
1 iteration	0.55	0.38	0.26	0.27	0.41	1.23	1.35
2 iterations	0.54	0.37	0.24	0.24	0.38	1.16	1.22
3 iterations	0.53	0.37	0.23	0.23	0.36	1.12	1.13
4 iterations	0.53	0.36	0.23	0.22	0.36	1.10	1.11
5 iterations	0.52	0.35	0.22	0.21	0.35	1.09	1.10
6 iterations	0.51	0.35	0.21	0.21	0.35	1.08	1.10
7 iterations	0.51	0.34	0.21	0.21	0.35	1.08	1.09

Table 3: This table shows the obtained $E^{(2)}_{\sigma}$ RMSE values depending on the number of iterations of the noise curve filtering and the standard deviation of the noise (see section 2.2). The percentile $p = 0.005$ and seven bins are used. Five iterations is the recommend value, since using more iterations does not improve the result significantly and it could soften too much the noise curves for certain images.

Figure 5: Set of raw images used to test the noise estimation algorithm using 8×8 blocks, percentile 0.5%, 49 bins and without any noise curve filtering. The images are raw 12 bits/channel, taken with a Canon EOS 30D camera, ISO 1600 and exposure time t=1/30s. From left to right and up to bottom: images 1, 2, 3, 4, 5, 6, 7, and 8.

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

676×976 pixels. Since the position of the camera was fixed when taking the snapshots of the image and assuming constant lighting, the variance along several samples coming from different images at the same pixel position could only be explained by the presence of noise. Therefore, it was possible to measure the mean of a block and the temporal standard deviation along all the snapshots to create an association mean→standard deviation, that is, a ground truth for camera noise curve, given the ISO and exposure times. Moreover, since the exposure time only affects the photon count and not the noise model, it was possible to overlap the noise curves for the four exposure times tested in a single ground truth noise curve depending only on the ISO parameter (see figure 7). Figure 8 shows

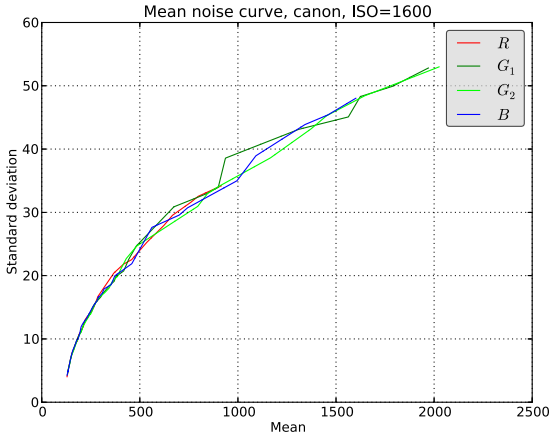


Figure 7: Ground truth of the Canon EOS 30D camera with ISO=1600.

an example of the noise curve obtained with this method. It matches with the ground truth quite accurately (figure 7).

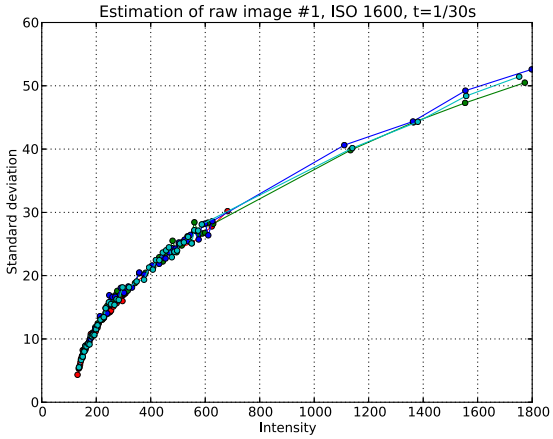


Figure 8: Noise curve obtained for the test image #1. Compare it with the ground truth noise curve in figure 7. Both curves match with small error.

Given an estimated noise curve A of a test image, its control points are the pairs $(\hat{\mu}_{A,i,b}, \hat{\sigma}_{A,i,b}) \in A$ 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 values for bin b and image i in A. In the same way, given a ground truth noise curve G, its control points are the pairs $(\mu_{G,v}, \sigma_{G,v}) \in G$. Unfortunately the means 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 9).

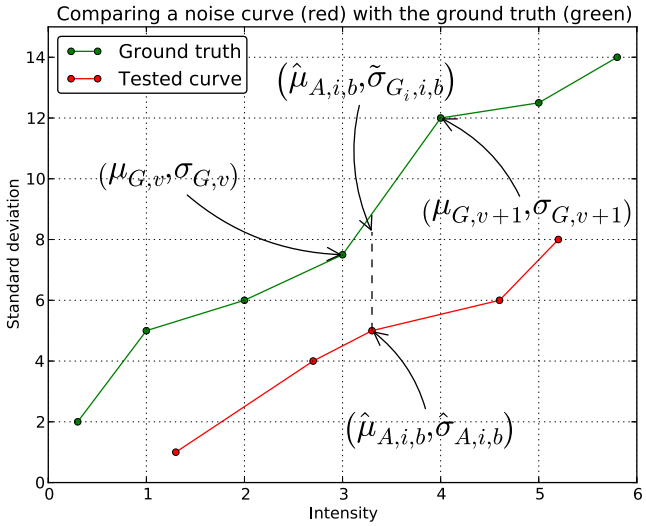


Figure 9: Checking a noise curve A (red) against the ground truth G (green), where i is the index of the image, b is the index of the bin, $(\hat{\mu}_{A,i,b}, \hat{\sigma}_{A,i,b})$ are the control points of the noise curve A, $(\mu_{G,v}, \sigma_{G,v})$ are the control points of G, and $\tilde{\sigma}_{G_i,i,b}$ is the standard deviation value projected from A into G.

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

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

The values of $E^{(3)}_{G,A,i,b}$ for each test image in figure 5 are shown in table 4.

Img. 1	Img. 2	Img. 3	Img. 4	Img. 5	Img. 6	Img. 7	Img. 8
0.802	0.381	0.372	0.307	0.437	0.694	0.436	0.580

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 for the Canon EOS 30D with ISO 1600. Note that the values of the intensities in a raw image are expressed in 12 bits and not with the usual 8 bits. Therefore, these errors should be divided by 16 in order to be compared with those obtained using 8 bits.

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,

$$E^{(4)}_{G,A,b} := \frac{1}{|I|}\sum_{i=0}^{|I|-1} E^{(3)}_{G,A,i,b}.$$

Figure 10 shows the error $E^{(4)}_{G,A,b}$ for all the 49 bins of the test images in figure 5 for the first green channel.

Figure 10: Mean error $E^{(4)}_{G,A,b}$ for the 49 bins of all the eight tests images in figure 5.

3.3 Evaluation of the Multi-scale 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 $\tilde{\mathbf{U}}$ is a discrete pure Gaussian noise image with standard deviation σ, then $\mathbf{S}(\tilde{\mathbf{U}})$ has standard deviation $\frac{\sigma}{2}$. Indeed, if a block $\mathbf{W}$ contains the pixels $\{u_1, u_2, u_3, u_4\}$ each one with variance σ^2, the variance of the mean of $\mathbf{W}$ is $\mathrm{Var}(\bar{\mathbf{W}}) = \mathrm{Var}\left(\frac{u_1+u_2+u_3+u_4}{4}\right) = \frac{1}{16}[\mathrm{Var}(u_1)+\mathrm{Var}(u_2)+\mathrm{Var}(u_3)+\mathrm{Var}(u_4)] = \frac{1}{16}[4\sigma^2] = \frac{\sigma^2}{4}$. Therefore, the standard deviation is $\mathrm{Std}(\bar{\mathbf{W}}) = \frac{\sigma}{2}$; the noise is divided by two. The objective of this test is to check if the noise estimation algorithm indeed divides the noise by two when the image is down-scaled several times.

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

where

- A_k is the noise curve corresponding to the input image i after applying the down-scaling operator k times. For example, if $k=2$ then A corresponds to the curve of the noise estimation of $\mathbf{SS}(\tilde{\mathbf{U}})$.
- i is the image index, for the raw images in figure 5, $1 \le i \le |I|$, where $|I|$ is the number of images. $|I| = 8$ images were used.
- b is the bin index, $1 \le b \le |B_k|$ 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 $\hat{\mu}_{A_0,i,v} \le \hat{\mu}_{A_k,i,b} < \hat{\mu}_{A_0,i,v+1}$ (see figure 11).

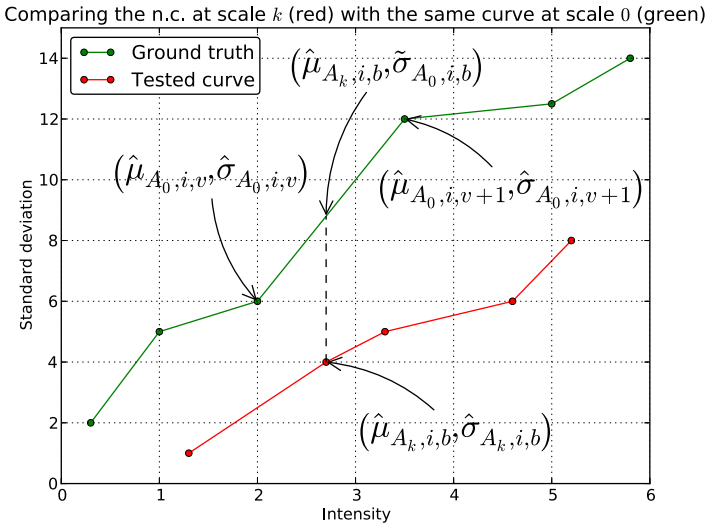


Figure 11: Checking a noise curve A_k at scale k (red) against the noise curve A_0 of the same image at scale 0 (green), where i is the index of the image, b is the index of the bin, $(\hat{\mu}_{A_k,i,b}, \hat{\sigma}_{A_k,i,b})$ are the control points of the noise curve of the sub-scaled image, $(\hat{\mu}_{A_0,i,v}, \hat{\sigma}_{A_0,i,v})$ 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 into 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 $\mathbf{D}$) 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 $E^{(5)}_{A_0,A_k,i,b}$ along all the test images and bins in figure 5, we define another error function as

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

Table 5 shows the obtained mean down-scale error $E^{(6)}_{A_0,A_k,b}$ for the raw images in figure 5 depending on the scale k. The measurements are done for one of the green channels of the raw images.

$\boldsymbol{k}$	1	2	3
$E^{(6)}_{A_0,A_k}$	0.195	0.738	1.684

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

4 Complexity Analysis of the Algorithms

The noise estimation procedure (algorithm 1) first computes the DCT of all the $w \times w$ blocks in the image. Since the DCT is computed using the FFT algorithm, that has a complexity $O(M \log M)$.

The loop that iterates seven times to get the optimal K executes the *argmin* operation that involves the Quick-sort algorithm and therefore it can be done with $O(n \log n)$ with $n = |\mathbf{V}_m^L|$. The computation of $\mathbf{V}_m^L = \frac{1}{\theta} \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} [\mathbf{D}_m(i,j)]^2 \, \delta(i,j)$ and $\mathbf{V}^H(i,j) = \frac{1}{K} \sum_{k=0}^{K-1} \left[\mathbf{D}_{(k)}(i,j)\right]^2$ have linear complexity $O(M)$ and $O(K)$, namely. Therefore, algorithm 1 has a linear complexity $O(M)$ since $M > K$. Algorithm 2 performs the same operations with the exception of substituting the seven iterations to get K with the product $k = pM$. Therefore, algorithm 2 has also linear complexity $O(M)$. Algorithm 3 first executes the *argsort* operation that can be executed with complexity $O(N \log N)$ using the Quick-sort algorithm. The loop that iterates $\text{idx} = 0 \ldots N$ just copies data in linear time $O(N)$. Therefore, algorithm 3 is executed with complexity $O(N \log N)$. Algorithm 4, is simply an conditional comparison and then 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 $O(B)$, being B the number of bins. Algorithm 7 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 fixed and smaller than M, the complexity of algorithm 7 is given by its main loop, that is executed in linear time with complexity $O(M)$.

5 Online Demo

An online demo is available for this algorithm in the IPOL web page 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 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:

Percentile. The possible values are 0.01%, 0.1%, 0.5% (default), 5%, 10%, 50%. It can be also configured to use the iterations of the original method in order to find the percentile K (see algorithm 1).

Block size. The possible choices are 3×3, 5×5, 7×7, 8×8 (default), 11×11, 15×15 and 21×21.

Mean of blocks computation. permits to choose how the intensity associated to each bin is calculated. The possible choices are the average of the mean value of the pixels of the blocks that belong to the bin, or the median of the pixels of the blocks that belong to the bin (default).

[3] http://dx.doi.org/10.5201/ipol.2013.45

Curve filter iterations. It indicates the number of filtering iterations that are applied to filter the noise curve (see section 2.2). Default: five iterations.

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 unconditionally (see section 2.3).

Number of bins. It is the number of bins in the noise curve (see section 2.1). The number of bins that are used depends on the size of the image when "automatic selection" is chosen. First, a nearest compatible size of the image is considered. For images whose size is S_0, S_1 or S_2, the number of bins is given by dividing the total number of pixels of the image by 42000. If the image is compatible with the size S_3, 4 bins are used. If the image is compatible with the size S_4 or if it is smaller, a single bin is used. Default: automatic selection.

A and B noise parameters. Add a simulated noise with variance $A + B\tilde{\mathbf{U}}$ is added 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: $A = B = 0$.

5.1 Subtraction of the Quantization Noise

In the online demo, all the images are encoded using 8 bits/pixel/channel. This adds a quantization error over the noise being estimated that must be subtracted. 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 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

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

5.2 Example: *traffic* Image

The results of this example can be reproduced by adding noise with parameters $A = 0$ and $B = 0.5$ to the *traffic* image. The rest of the parameters are the default parameters of the demo. Figure 12 shows the input noiseless image *traffic* before adding signal dependent noise with variance $\sigma^2 = 0.5\mathbf{U}$. Figure 13 shows the noise estimated for the three first scales of the signal-dependent noise with variance $\sigma^2 = 0.5\mathbf{U}$ added to the *traffic* image. Because the noise was added to a noise-free image, we can compute the RMSE for the different scales S_0, S_1 and S_2, and the corresponding errors are 0.15, 0.18 and 0.16, respectively. Note that, as expected, the noise standard deviation is divided by approximately two when down-scaling the image by the same ratio.

Acknowledgements

Research partially financed by the MISS project of Centre National d'Etudes Spatiales, the Office of Naval Research under grant N00014-97-1-0839, by the European Research Council, advanced grant

Figure 12: Noise free input image *traffic* before adding noise with variance $\sigma^2 = 0.5\mathbf{U}$.

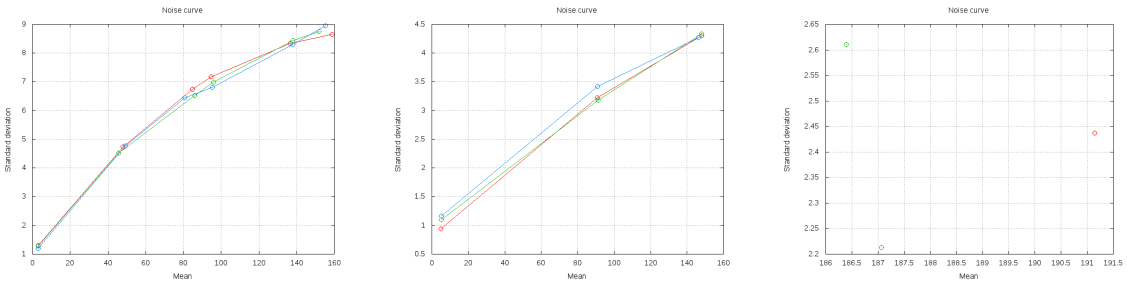
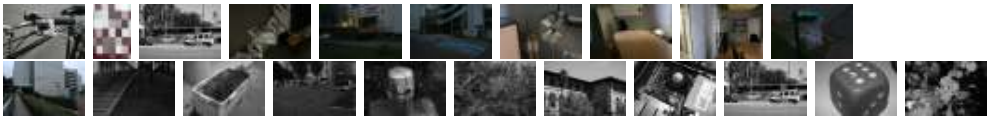


Figure 13: The noise estimated for the three first scales of the signal-dependent noise with variance $\sigma^2 = 0.5\mathbf{U}$ added to the *traffic* image. From left to right: scales S_0 (original), S_1 and S_2. Note that the noise standard deviation is approximately divided by two when down-scaling.

"Twelve labours" and the Spanish government under TIN2011-27539.

Image Credits

 Miguel Colom, CC-BY

 The USC-SIPI Image Database [10].

References

[1] 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*, pages 1–15. IEEE, 2009, http://dx.doi.org/10.1109/LNLA.2009.5278408.

[2] J. Immerkaer. Fast noise variance estimation. *Computer Vision and Image Understanding*, 64(2):300–302, 1996, http://dx.doi.org/10.1006/cviu.1996.0060.

[3] M. Lebrun, M. Colom, A. Buades, and J.M. Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21:475–576, 2012, http://dx.doi.org/10.1017/S0962492912000062.

[4] J.S. Lee and K. Hoppel. Noise modelling and estimation of remotely-sensed images. In *Proceedings of the International Geoscience and Remote Sensing Symposium*, volume 2, pages 1005–1008, 1989, http://dx.doi.org/10.1109/IGARSS.1989.579061.

[5] G.A. Mastin. Adaptive filters for digital image noise smoothing: An evaluation. *Computer Vision, Graphics, and Image Processing*, 31(1):103–121, 1985, http://dx.doi.org/10.1016/S0734-189X(85)80078-5.

[6] 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(2):216–223, 1990, http://dx.doi.org/10.1109/34.44408.

[7] S.I. Olsen. Estimation of noise in images: an evaluation. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 55(4):319–323, July 1993. http://dx.doi.org/10.1006/cgip.1993.1022.

[8] N.N. Ponomarenko, V.V. Lukin, M.S. Zriakhov, A. Kaarna, and J.T. Astola. An automatic approach to lossy compression of AVIRIS images. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 472–475, 2007, http://dx.doi.org/10.1109/IGARSS.2007.4422833.

[9] K. Rank, M. Lendl, and R. Unbehauen. Estimation of image noise variance. In *Vision, Image and Signal Processing*, volume 146, pages 80–84. IET, 1999, http://dx.doi.org/10.1049/ip-vis:19990238.

[10] Allan G Weber. The USC-SIPI image database version 5. *USC-SIPI Report*, 315:1–24, 1997.