Published in Image Processing On Line on 2017–10–29.
Submitted on 2017–01–15, accepted on 2017–10–10.
ISSN 2105–1232 © 2017 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.2017.201

Multi-Scale DCT Denoising

Nicola Pierazzo, Jean-Michel Morel, and Gabriele Facciolo

CMLA, ENS Paris-Saclay, Cachan, France
{Nicola.Pierazzo,morel,facciolo}@cmla.ens-cachan.fr

This IPOL article is related to a companion publication in the SIAM Journal on Imaging Sciences:
Gabriele Facciolo, Nicola Pierazzo, Jean-Michel Morel, "Conservative Scale Recomposition for Multiscale Denoising (The Devil is in the High Frequency Detail)" *SIAM Journal on Imaging Sciences*, vol. 10, no. 3, pp. 1603–1626, 2017. https://doi.org/10.1137/17M1111826

Abstract

DCT denoising is a classic low complexity method built in the JPEG compression norm. Once made translation invariant, this algorithm was still proven to be competitive at the beginning of this century. Since then, it has been outperformed by patch based methods, which are far more complex. This paper proposes a two-step multi-scale version of the algorithm that boosts its performance and reduces its artifacts. The multi-scale strategy decomposes the image in a dyadic DCT pyramid, which keeps noise white at all scales. The single scale denoising is then applied to all scales, thus giving multiple denoised versions of the low frequency coefficients of the denoised image. A "multi-scale fusion" of these multiple estimates avoids the ringing artifacts resulting from the pyramid recomposition. The final algorithm attains a good PNSR and much improved visual image quality. It is shown to have a deficit of only 1dB with respect to state of the art algorithms, but its complexity is two orders of magnitude lower.

Source Code

The C++ source code, the code documentation, and the online demo are accessible at the IPOL web page of this article web site[1] Compilation and usage instruction are included in the `README.txt` file of the archive.

Keywords: multi-scale; image denoising; DCT denoising

[1] https://doi.org/10.5201/ipol.2017.201

1 Introduction

DCT denoising is a classic low complexity algorithm, that nonetheless was still proven to outperform more recent multi-scale wavelet denoising algorithms [14, 19]. A reference interpretation and implementation is proposed in [20], where the efficiency of the algorithm is boosted by its translation invariant implementation: all patches of given size (between 4×4 and 16×16) are denoised. Then the result of this denoising is aggregated by taking as final result at each pixel the average of all values given by all denoised patches containing this pixel. The recommended patch size in [20] is 16×16.

Since the processing of each pixel involves pixels at a distance smaller than 16, the low frequency noise is not handled. For severely noisy images, this residual low frequency noise becomes conspicuous, particularly in smooth or flat image regions. This points to the Achilles' heel structural drawback of this simple and powerful algorithm: it operates at a single scale and does not attack low frequencies.

In this paper we propose a two-step and multi-scale version of DCT denoising that keeps all features of its single scale single-step version, but improves notably its performance. Our method for transforming a single scale denoising algorithm into a multi-scale algorithm is detailed in a companion SIIMS paper [7], where the method is applied to six different denoising algorithms. Here we limit ourselves to a detailed description of the method applied to DCT denoising, and we deliver in that way a well performing algorithm with surprisingly low complexity.

Multi-scale principles for denoising have already been explored in the literature. Multi-scale image processing is justified by the classic assumption [9] that the statistics of natural images are invariant to a change of scale [12]. The scale invariance assumption is invoked by most multi-scale algorithms [16, 2, 22].

Yet existing multi-scale approaches either end up changing the noise structure at lower frequencies [2, 6], or require a non-standard denoising algorithm [17, 21]. An example of approach specific to a particular denoising algorithm is [15], which is a two-scale extension of EPLL [21]. It could nevertheless be understood as a general multi-scale framework applicable to any single scale variational method. Similar in that respect to DCT denoising, the wavelet-based denoising algorithms [4, 8, 16, 13] always perform some sort of transform thresholding. This entails annoying "ringing" or "butterfly" artifacts attributable to Gibbs effects caused by harsh frequency cut-offs. This remains true of sophisticated algorithms involving a wavelet multi-scale transform like [18], where the KSVD algorithm is applied on a wavelet image decomposition. There have also been attempts to post-process ringing artifacts by a variational method [5].

We shall use the multi-scale framework proposed in [7], that can be applied to any existing single-scale denoising algorithm. The framework is not computationally demanding as it uses a simple global DCT transform to extract successive subsampled images. The benefit of DCT subsampling is that it does not change the structure of white noise, which allows to use any denoising algorithm without adaptation. Each level of the resulting image pyramid is then denoised independently and the results combined. This is in contrast with multi-scale schemes that operate on the differences between consecutive scales. The proposed multi-scale fusion avoids the ringing artifacts indirectly caused by the denoising of the coarse scale images. It does so by dropping the high frequency components from the subsampled images of the pyramid.

Our second improvement of DCT denoising is based on a 2-step oracle method, which idea is to use a first denoised image as an oracle to estimate the empirical Wiener factors of the DCT coefficients in the second step. Note that a local DCT is used by the algorithm to denoise the image patches. The multi-scale scheme instead uses a global DCT transform. Iterating a denoising algorithm in order to use the result of the former step as oracle for the next step is also a classic tool that mitigates the ill-posed character of the denoising problem. We refer to [11] for a review and for the proposal of an

Algorithm 1: two-step DCT Denoising

```
Function DCTdenoising2step(Y, σ, s)
    input  : noisy image Y, noise level σ, and patch size s
    output: denoised image
    G ← DCTdenoisingHard(Y, σ, s)
    return DCTdenoisingWiener(Y, G, σ, s)
```

oracle method for DCT denoising.

Section 2 describes the single scale DCT denoising extended with an oracle step (also called Wiener step). Section 3 describes the DCT-based multi-scale framework and applies it to DCT denoising. Section 4 evaluates the PSNR gain obtained for each considered improvement, namely the Wiener step and the multiscale framework, depending on the patch size. It also illustrates on several images the visual quality gains of the final method. Implementation details are presented in Section 5 and Section 6 is a conclusion.

2 Oracle DCT Denoising

The simplest DCT denoising algorithm as described by Yu and Sapiro in [20] consists in a threshold of a patch-wise DCT of the image and aggregation of the resulting patches. Color images are first pre-processed to de-correlate the input colors. Color patches are then processed channel-wise but their adaptive aggregation weights are computed over all the channels. Algorithm 2 summarizes the method, and the details about the DCT transform convention used in it are recalled in Section 5.

Let us note that although thresholding the DCT of a single patch introduces ringing artifacts, it has been observed that the patch-wise aggregation limits them in the final result.

Oracle. The use of an oracle is an improvement of the DCT denoising algorithm which was mentioned in [11] but not integrated into [20]. A guide image (or oracle) is a clean but not completely restored result, which allows to estimate the spectra of the patches in the Wiener filtering step as shown in Algorithm 3. The resulting two-step DCT denoising algorithm is summarized in Algorithm 1, and its improvement can be corroborated in Figure 7 and Table 2.

Adaptive aggregation. Adaptive patch aggregation [11, 3] allows to further reduce the halo effects near the contrasted image edges (as shown in Figure 1) while keeping essentially the same PSNR. Since denoised patches representing a contrasted edge always end up containing some ringing, it is preferable to give more weight to other overlapping patches that do not contain the edge. Adaptive aggregation does this by effectively giving more weight to patches that have a sparser representation in the DCT domain. This permits to reduce the ringing effects near image edges. In the Appendix A we justify in detail our choice of aggregation weights.

For the hard thresholding pass of the algorithm the aggregation weights are set, as in [3], by counting the number N_P of nonzero DCT coefficients (excluding the zero frequency) in the patch after thresholding. These aggregation weights are then given by

$$(1 + N_P)^{-1}\,, \tag{1}$$

where the one is added to prevent the dividing by zero (but it is an arbitrary choice). Indeed, the number of non-zero coefficients will be small for the flat patches, compared to patches containing

Algorithm 2: DCT Denoising - Hard thresholding

1 **Function** DCTdenoisingHard(Y, σ, s)
  input : noisy image Y, noise level σ, and patch size s
  output: denoised image
2 $X, W \leftarrow 0$
3 $Y \leftarrow$ DecorrelateColors(Y)
4 **for each** *patch domain* $\Omega_{patch} \subset \Omega$ *of size* $s \times s$ **do** // Ω is the image support
5 $b_{tmp} \leftarrow 0$ // color patch temp variable
6 $N_P \leftarrow 0$
7 **for each** *color channel* c **do**
8 $\widehat{b} \leftarrow \mathrm{DCT}(\text{ExtractPatch}(Y, \Omega_{patch}, c))$ // uses DCT/IDCT defined in (10)-(11)
9 **for** $\omega \in (\{0, \cdots, s-1\} \times \{0, \cdots, s-1\})$ **do** // scan patch frequency domain
10 **if** $\omega \neq \vec{0}$ **then** // don't filter the zero frequency
11 **if** $|\widehat{b}(\omega)| < 3\sigma$ **then** $\widehat{b}(\omega) \leftarrow 0$
12 **else** $N_P \leftarrow N_P + 1$ // # of nonzero coefficients of $\widehat{b}$
13 $b_{tmp}[c] \leftarrow \mathrm{IDCT}(\widehat{b})$ // store channel c of color patch
14 $X(\Omega_{patch}) \leftarrow X(\Omega_{patch}) + b_{tmp} \cdot (1 + N_P)^{-1}$
15 $W(\Omega_{patch}) \leftarrow W(\Omega_{patch}) + (1 + N_P)^{-1}$ // Adaptive weights, see Section A
16 $X \leftarrow X/W$
17 **return** UndoDecorrelateColors(X)

edges. Therefore the flat patches will be privileged in the aggregation thus reducing the ringing introduced by the denoised edge patches.

The weights for the second step (lines 17 and 18 of Algorithm 3) are set to the squared $\ell 2$-norm of the Wiener coefficients ρ_P (defined in Algorithm 3, line 13) but excluding the zero frequency. To avoid dividing by zero we define the weights as

$$(1 + S_P)^{-1}. \tag{2}$$

This is similar to what is used in the BM3D Algorithm [3]. The justification for this choice of weights, as well as an empirical exploration of other weighting strategies, is presented in Appendix A.

For color images the aggregation weights of a patch are computed by counting N_P or S_P over all the channels.

Filtering of the zero frequency. Finally, a small, but important difference in Algorithms 2 and 3 with respect to the one described in [20] is that the zero frequency (which contains the mean of the patch) is not altered by the thresholding or shrinkage. The justification for this is that the quadratic mean is already the optimal estimator of the patch mean value, so thresholding or shrinking would just bias it toward zero. This is because the sparsity prior, applied to all the other oscillatory coefficients, does not apply on the mean value.

3 The Multi-scale Framework

Figure 2 shows the frequency distribution of the result of DCT denoising on an image composed only of white noise, together with the results for our multi-scale version of the very same algorithm.

Algorithm 3: DCT Denoising - Wiener

Function DCTdenoisingWiener(Y, G, σ, s)
  input : noisy image Y, guide image G, noise level σ, and patch size s
  output: denoised image
  $X, W \leftarrow 0$
  $Y \leftarrow$ DecorrelateColors(Y)
  $G \leftarrow$ DecorrelateColors(G)
  for each *patch domain* $\Omega_{patch} \subset \Omega$ *of size* $s \times s$ **do** // Ω is the image support
    $b_{tmp} \leftarrow 0$ // color patch temp variable
    $S_P \leftarrow 0$
    for each *color channel* c **do**
      $\widehat{b} \leftarrow \mathrm{DCT}(\text{ExtractPatch}(Y, \Omega_{patch}, c))$ // uses DCT/IDCT defined in (10)-(11)
      $\widehat{g} \leftarrow \mathrm{DCT}(\text{ExtractPatch}(G, \Omega_{patch}, c))$
      for $\omega \in (\{0, \cdots, s-1\} \times \{0, \cdots, s-1\})$ **do** // scan patch frequency domain
        if $\omega \neq \vec{0}$ **then** // don't filter the zero frequency
          $\rho_P(\omega) \leftarrow \left(\dfrac{|\widehat{g}(\omega)|^2}{|\widehat{g}(\omega)|^2 + \sigma^2} \right)$
          $\widehat{b}(\omega) \leftarrow \widehat{b}(\omega)\, \rho_P(\omega)$
          $S_P \leftarrow S_P + \rho_P(\omega)^2$ // squared $\ell 2$-norm of ρ_P excluding $\rho_P(\vec{0})$
      $b_{tmp}[c] \leftarrow \mathrm{IDCT}(\widehat{b})$ // store channel c of color patch
    $X(\Omega_{patch}) \leftarrow X(\Omega_{patch}) + b_{tmp} \cdot (1 + S_P)^{-1}$ // weight by non-divergent inverse of S_P
    $W(\Omega_{patch}) \leftarrow W(\Omega_{patch}) + (1 + S_P)^{-1}$
  $X \leftarrow X/W$
  return UndoDecorrelateColors(X)

Because of the limited size of the patches used by the denoising methods we note that they underperform on low frequencies. For instance, we see that the lowest frequency "visible" with a 4×4 patch is half the Nyquist rate (one fourth for 8×8 patches), as a result the lower frequencies are not well denoised.

Given a multi-scale image representation a straightforward way to improve the denoising performance is to apply the denoising algorithm at each scale, and then to recompose the image, always preferring the low frequency coefficients from the lower scales.

There are two restrictions to this. First, to be able to adapt trivially each original one-scale algorithm, we need every layer of the multiscale representation to keep a white Gaussian additive noise. Second, we need a practical and effective rule to recompose an output image from the denoised results at different scales. We now sketch the solution analyzed in detail in [7].

A way to address both requirements is to use a DCT Pyramid. The Discrete Cosine Transform, or DCT given in (3) is a real separable orthogonal transform. For 2-D signals, the isometric DCT can be computed by applying (3) to the rows and the columns (see Section 5). Its inverse is the

no aggregation weights (27.89 dB) aggregation weights (27.99 dB)

Figure 1: Detail of a result from MS DCT denoising with 8×8 patches computed without and with aggregation weights for a noise level $\sigma = 50$. Note the reduced oscillations in the sky.

IDCT (4). For $k = 0, \cdots, N-1$ and $j = 0, \cdots, N-1$,

$$Y_k = \alpha_k \, 2 \sum_{j=0}^{N-1} X_j \cos\left[\pi \left(j + \frac{1}{2}\right) \frac{k}{N}\right], \qquad \text{with} \quad \alpha_k = \begin{cases} \sqrt{1/(4N)}, & k = 0 \\ \sqrt{1/(2N)}, & k = 1 \ldots, N-1 \end{cases} \tag{3}$$

$$X_j = \beta_0 \, Y_0 + \sum_{k=1}^{N-1} \beta_k \, 2 Y_k \cos\left[\pi \left(j + \frac{1}{2}\right) \frac{k}{N}\right], \quad \text{with} \quad \beta_k = \begin{cases} \sqrt{1/N}, & k = 0 \\ \sqrt{1/(2N)}, & k = 1 \ldots, N-1. \end{cases} \tag{4}$$

The DCT of an image is displayed in Figure 2 with its low frequency coefficients in the upper-left corner. It transforms additive Gaussian white noise into additive Gaussian white noise. The DCT transform can be used to form a multi-scale representation of an image. The down-sampling of the image is simply done by extracting the low frequencies from the DCT transform of the image, and by computing the IDCT on just those frequencies. Each layer of the pyramid has half the width and half the height of the previous one.

Using (3) and (4) for this procedure guarantees that white Gaussian noise remains so under the DCT transform, so the noise model remains the same in every layer of the pyramid. A scaling factor is used (Algorithm 4 lines 14 and 25) to guarantee that the values of the image remain on the same range after resizing, which also implies that the standard deviation of the noise gets halved at each successive scale.

Thus, no particular adaptation of the initial single scale denoising algorithm is needed to denoise the coarse layers. Recomposing the pyramid is trivial, since it can be reduced to substituting the low frequencies of a layer with the frequencies of the coarser layer.

The drawback of this substitution is that, since each layer is essentially the result of the convolution of the previous one with a sinc-like function, ringing artifacts due to the Gibbs effect unavoidably appear in the coarse levels of the pyramid. But, this is not a problem for the pyramid representation in itself since these Gibbs artifacts are usually compensated during the recomposition by the complementary oscillations resulting from the high pass filtering of the higher resolution layers. The problem occurs because the high-frequency (and low amplitude) oscillations in the lower resolution levels of the pyramid are likely to be damaged or even removed by the denoising method. Thus, in a naïve recomposition the oscillations resulting from the high-pass will no longer be compensated, and the Gibbs artifacts become visible [7]. In essence the Gibbs artifacts in the lower resolution levels of the pyramid are crucial for recomposing an artifact-free pyramid, but they are removed by the denoising algorithms!

Algorithm 4: Pseudo-code for the Multi-Scale Framework.

Function MultiScale*(input, σ_{noise}, n_{scales}, f_{rec})*
  for $l \leftarrow n_{scales} - 1, \dots, 0$ **do**
    $layer \leftarrow$ ExtractScale$(input, l)$
    // Because of integer parts noise reduction may not be an exact power of 2,
    // hence the noise scaling ratio below uses the pixel count instead of 2^{-l}
    $tmp \leftarrow$ Denoise$(layer, \sigma_{noise}\sqrt{\frac{\text{Numpix}(layer)}{\text{Numpix}(input)}})$
    if $l == n_{scales} - 1$ **then** $result \leftarrow tmp$
    else $result \leftarrow$ MergeCoarse$(tmp, result, f_{rec})$
  return $result$

Function ExtractScale*(image, l)*
  $w, h \leftarrow$ Size$(image)$
  $w_{out} \leftarrow \lfloor w/2^l \rfloor$
  $h_{out} \leftarrow \lfloor h/2^l \rfloor$
  $freq \leftarrow$ DCT$(image)$ // uses DCT/IDCT defined in (10)-(11)
  $tmp \leftarrow$ Zeros(w_{out}, h_{out})
  $scaling \leftarrow \sqrt{\frac{w_{out} \cdot h_{out}}{w \cdot h}}$
  for $i \leftarrow 0, \dots, h_{out} - 1,\ j \leftarrow 0, \dots, w_{out} - 1$ **do**
    $tmp[i, j] \leftarrow freq[i, j] \cdot scaling$
  return IDCT(tmp)

Function MergeCoarse*(image, coarse, f_{rec})*
  // Note that the *coarse* and *input* images have different sizes
  $freq \leftarrow$ DCT$(image)$ // uses DCT/IDCT defined in (10)-(11)
  $tmp \leftarrow$ DCT$(coarse)$
  $w, h \leftarrow$ Size$(coarse)$
  $w_{out}, h_{out} \leftarrow$ Size$(image)$
  $w_{rec} \leftarrow w \cdot f_{rec}$
  $h_{rec} \leftarrow h \cdot f_{rec}$
  $scaling \leftarrow \sqrt{\frac{w_{out} \cdot h_{out}}{w \cdot h}}$
  for $i \leftarrow 0, \dots, h_{rec} - 1,\ j \leftarrow 0, \dots, w_{rec} - 1$ **do**
    $freq[i, j] \leftarrow tmp[i, j] \cdot scaling$
  return IDCT$(freq)$

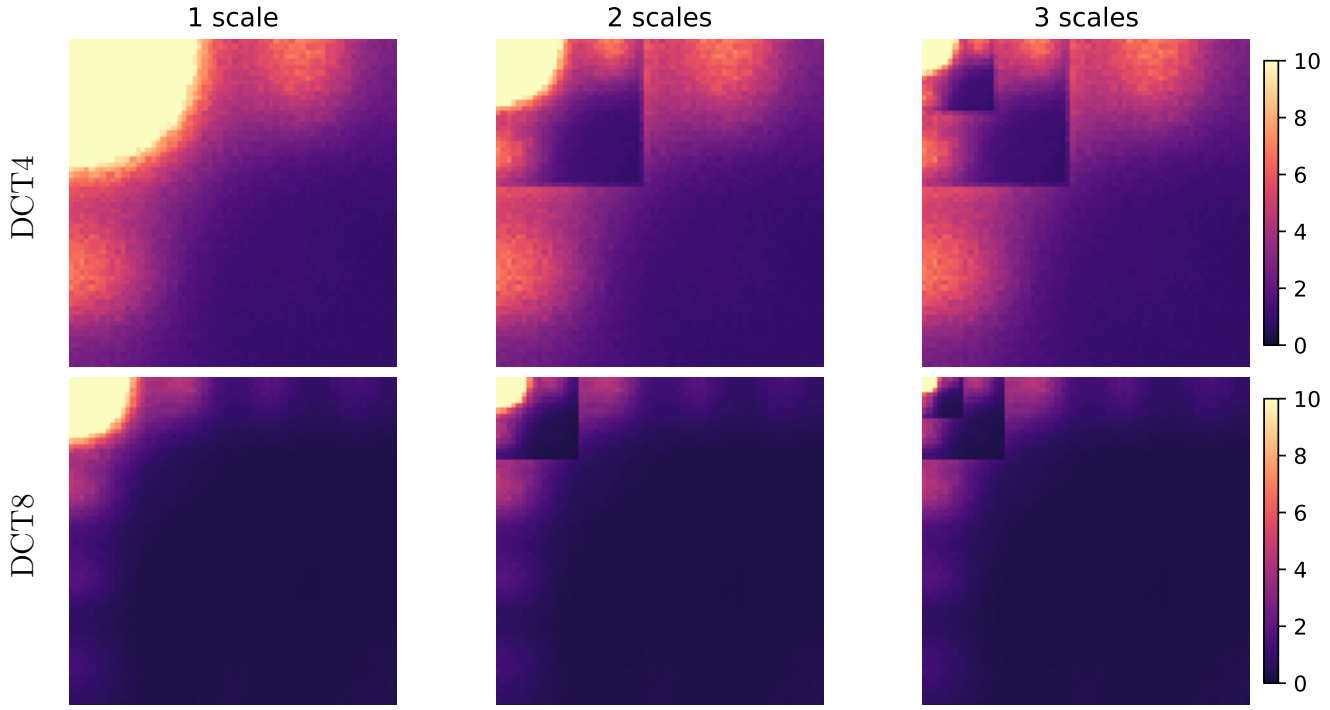

Figure 2: DCT transform amplitude of results of the multiscale DCT denoising algorithm applied to an image of pure white noise. The columns correspond to different numbers of scales. Notice the remaining low frequency noise in the upper left corner of the single scale results (first column). As expected, in the multi-scale results the residual noise is much lower. The results in the first row are computed with DCT denoising using 4×4 patches and using a recomposition factor $f_{rec} = 0.9$, while for the second row 8×8 patches are used with $f_{rec} = 0.5$.

In the solution proposed in [7] the original single scale algorithm is applied on the first level of the DCT image pyramid, and therefore to all frequencies. But it is also applied to the down-sampled images. Thus we get two different denoised estimates for the image low frequencies.

Avoiding Gibbs effects amounts to discard the higher frequencies of the denoised down-sampled image, and to replace them by the corresponding medium frequencies of the denoised upper layer. This effectively restores the frequency cut-off (and Gibbs effects), which are needed for an artifact-free recomposition of the pyramid. In short, we only keep the *lower frequencies* of the coarser layers (except of course for the highest level), as detailed in Algorithm 4. The recomposition factor parameter $f_{rec} \in (0, 1]$ controls the fraction of low frequencies at each scale being used in the recomposition. That is, setting $f_{rec} = 1$ keeps all the frequencies of each level.

The support function ExtractScale($image$, l) in Algorithm 4 is used to extract a specific level from the DCT pyramid. The level 0 is the input image itself, and every other level is half the size of the previous one. Conversely, the support function MergeCoarse($image$, $coarse$, f_{rec}) is used to join together two different levels. The low frequency coefficients of $image$ get replaced by the ones from $coarse$, in a ratio proportional to f_{rec}. Finally, MultiScale($input$, σ_{noise}, n_{scales}, f_{rec}) performs the whole denoising process, using the previous two functions. Here Denoise($image$, σ) is the denoising algorithm that is used with the framework.

Note that the multi-scale recomposition factor aims at reducing the Gibbs artifacts resulting from the global DCT subsampling (used for building the pyramid). While the adaptive aggregation weights reduce the Gibbs artifacts due to the patch-wise DCT denoising. Figure 3 illustrates the contributions of these modifications in the case of a synthetic image.

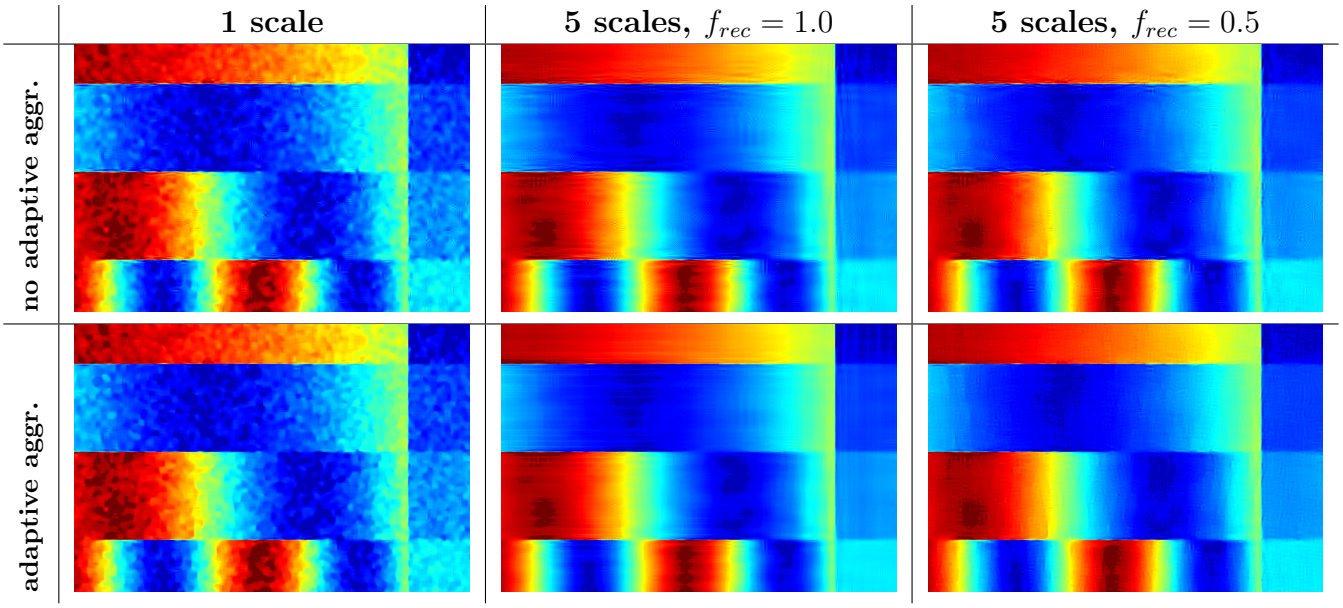

Figure 3: Gibbs artifacts reduction obtained by the multi-scale recomposition factor and the adaptive aggregation weights. The images show the results of the oracle DCT denoising with 8×8 patches applied to a synthetic image (not shown), for a noise with $\sigma = 70$. Images are monochrome but rendered with false colors to improve visualization. The recomposition factor $f_{rec} = 1.0$ implies that the high frequencies of lower scales are preserved, while $f_{rec} = 0.5$ (optimal value for this method according to Table 1) uses only the lower frequencies. Note how the horizontal and vertical streaks (ringing artifacts) visible in the results with $f_{rec} = 1.0$ (central column) are largely attenuated in the right-most column. (This is better seen in the electronic version.) Note that both the adaptive aggregation and the multiscale recomposition contribute to the reduction of the ringing artifacts.

4 Experiments

The first task is to fix the parameters of the multiscale framework, namely the fraction f_{rec} of low frequencies at each scale being used in the recomposition and the number of scales involved in the multiscale framework, both depending on the noise level. We show in Figure 4 the result for four noise levels, $\sigma = 10,\ 30,\ 50,\ 70,\ 90$. The images display the average PSNR gain obtained with different parameters of the multi-scale framework applied to the DCT denoising algorithm with different patch sizes. The integers on the left of each figure $(1, 2, \ldots, 5)$ represent the number of scales n_{scales} used in Algorithm 5. The value at the bottom is the fraction f_{rec} of low frequencies at each scale being used in the recomposition. These figures were obtained from experiments on a choice of noiseless test images displayed in Figure 5. The optimal parameters vary depending on the patch size and the noise level. This is particularly true for the results obtained using 16×16 patches, which can be worse than the single scale ones for many configurations of the Multi-Scale Framework (see the negative PSNR gains in Figure 4) Nevertheless, since the parameters are quite stable across noise levels we choose to fix them for all noise levels as shown in Table 1.

Table 1: Optimal parameters, found by the experiments shown in Figure 4, for the multiscale DCT denoising with different patch sizes.

Patch size w	**Scales** s	**Recomposition factor** f_{rec}
4×4	5	0.8
8×8	5	0.4
16×16	4	0.2

The PSNR comparative results are given in Table 2. The table includes the results obtained on

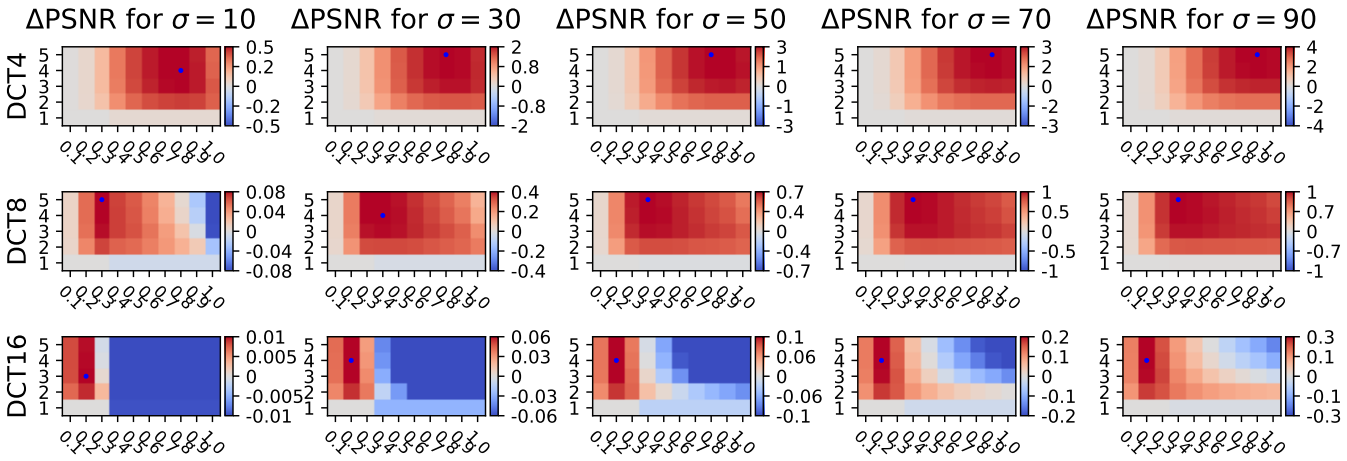

Figure 4: Average PSNR changes (in dB) obtained with different parameters of the Multi-Scale Framework applied to the DCT denoising algorithm for different noise levels. The integers on the left of each figure (1, 2, ..., 5) represent the number of scales n_{scales} used in Algorithm 5. The bottom row of each graphic corresponds to the single-scale algorithm for which ΔPSRN $= 0$. The value at the bottom is the fraction f_{rec} of low frequencies at each scale being used in the recomposition.

Algorithm 5: Multiscale DCT Denoising

Function MultiscaleDCT$(Y, \sigma, s, n_{scales}, f_{rec})$
  input : noisy image Y, noise level σ, patch size s,
    number of scales n_{scales}, and multiscale recomposition factor f_{rec}
  output: denoised image
  for $l \leftarrow n_{scales} - 1, \ldots, 0$ **do**
    $Y_l \leftarrow$ ExtractScale(Y, l)
    $X_l \leftarrow$ DCTdenoising2step$(Y_l, \sigma/2^l, s)$
    if $l == n_{scales} - 1$ **then** *combined* $\leftarrow X_l$
    else *combined* $\leftarrow$ MergeCoarse$(X_l, combined, f_{rec})$
  return *combined*

the training dataset of Figure 5 and on a set of test images with very low noise shown in Figure 6. A first observation is that the second step based on the oracle given by the first step improves the PSNR for larger patch sizes. The best result in terms of PSNR is obtained by the multiscale strategy with 8×8 patches, but the PSNR difference with respect to the single scale algorithm on large 16×16 patches seems almost negligible. Only for small or moderate patch sizes the multiscale strategy improves substantially the PSNR with respect to the 2-step single-scale DCT denoising. Yet the visual experiments that can be made with the on-line demo, which are exemplified in Figure 7, tell us another story. Indeed they show a significant quality gain by using a small (4×4) or moderate (8×8) patch size and with the multiscale strategy, particularly on flat or smooth image regions, where ringing and color spot artifacts can be conspicuous and annoying. In addition, the algorithm using the large 16×16 patches has about the same complexity as NL-Bayes, which is four times slower than using 8×8 patches with the multiscale strategy.

Figure 8 shows image details (taken from the set of test images in Figure 6) comparing the results of single- and multi-scale DCT denoising using 4×4 patches. Our choice of an important noise $\sigma = 40$ is made on purpose, as most state of the art algorithms start producing artifacts around this value. In all these examples, the multi-scale version shows a spectacular improvement over the single-scale version. One can observe a removal of spurious oscillations (ringing effects) in smooth regions (water, glass) and a significant gain in detail sharpness.

Figure 5: Images used to find the best parameters of the multi-scale algorithm. The size of each image is about 1.5 Megapixels.

Figure 6: Images with very low noise used to test the multi-scale algorithm. The size of each image is about 1.5 Megapixels.

Table 2: Average PSNR (in dB) of DCT denoising without oracle (1step), with oracle (2step), and with multiscale (algorithms 2, 1, 5), using patches of sizes 4×4, 8×8, and 16×16, with the corresponding optimal parameters. The experiments correspond to a noise with standard deviation $\sigma = 50$.

	Train Set (Fig. 5)		Test Set (Fig. 6)	
Algorithm	**PSNR** (dB)	**Gain wrt 1step**	**PSNR** (dB)	**Gain wrt 1step**
DCT4 1step	26.3	-	26.7	-
DCT4 2step	26.0	-0.3 ± 0.2	26.3	-0.3 ± 0.2
MS DCT4 1step	28.2	+1.9 ± 1.3	28.2	+1.5 ± 0.8
MS DCT4 2step	**28.5**	+2.2 ± 1.3	**28.5**	+1.8 ± 0.8
DCT8 1step	27.9	-	28.0	-
DCT8 2step	28.1	+0.2 ± 0.1	28.3	+0.3 ± 0.1
MS DCT8 1step	28.4	+0.5 ± 0.5	28.4	+0.4 ± 0.2
MS DCT8 2step	**28.8**	+0.9 ± 0.6	**28.9**	+0.8 ± 0.3
DCT16 1step	28.2	-	28.3	-
DCT16 2step	**28.6**	+0.3 ± 0.1	28.7	+0.4 ± 0.1
MS DCT16 1step	28.2	-0.0 ± 0.1	28.4	+0.1 ± 0.0
MS DCT16 2step	**28.6**	+0.4 ± 0.2	**28.8**	+0.5 ± 0.1

Figure 7: Results and PSNR of DCT denoising without oracle (1step), with oracle (2step), and with multiscale (algorithms 2, 1, 5), using patches of sizes 4×4, 8×8, and 16×16. For each patch size the optimal parameters from Table 1 were used.

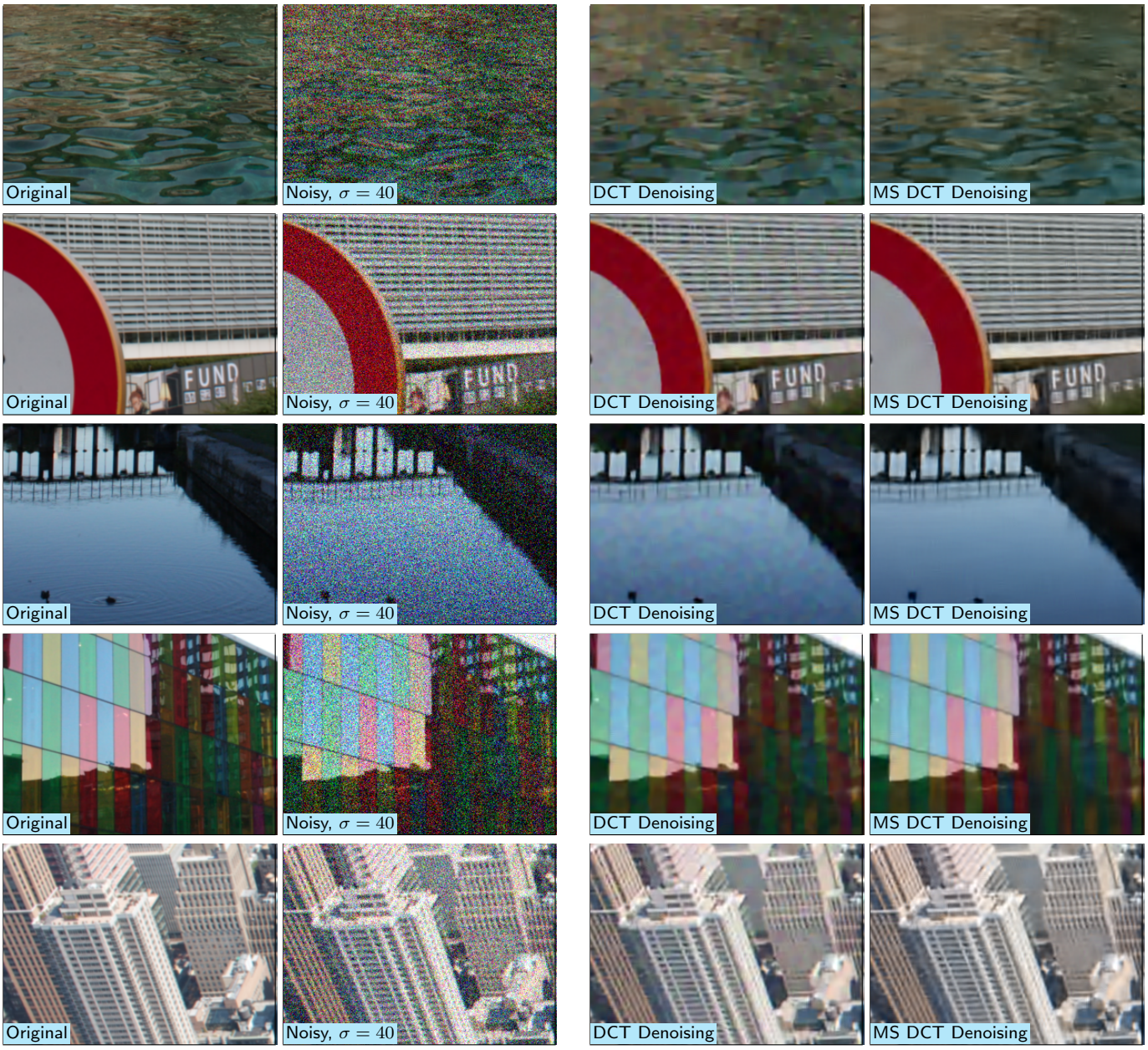

Figure 8: Results of single- and multi-scale DCT denoising with 8×8 patches applied to different images. In all cases one can observe a removal of spurious oscillations in smooth regions (water, glass) and a gain in detail sharpness.

5 Implementation Details

DCT transform using FFTW. Unlike the implementation of Yu and Sapiro [20] our implementation of the DCT denoising only uses a small amount of memory to process the image, as the extraction and processing of each patch is performed in a single loop, which can also be parallelized. Moreover, using the FFTW library to compute the DCT transforms further accelerates the process. The processing of large images is parallelized by splitting them into smaller tiles and processing each tile in parallel.

Isometric DCT transform. The type-II DCT transform implemented in the FFTW library and its inverse (type-III) are not isometric, so in order to implement the frequency domain denoising they must be normalized. The FFTW transforms (identified by the w superindex) compute for $k = 0, \cdots, N-1$

$$\mathrm{DCT}^{\mathrm{w}}(X)_k = 2\sum_{j=0}^{N-1} X_j \cos\left[\pi\left(j+\frac{1}{2}\right)\frac{k}{N}\right], \tag{5}$$

$$\mathrm{IDCT}^{\mathrm{w}}(Y)_k = Y_0 + 2\sum_{k=1}^{N-1} Y_k \cos\left[\pi\left(j+\frac{1}{2}\right)\frac{k}{N}\right], \tag{6}$$

which are unnormalized, hence $\mathrm{IDCT}^{\mathrm{w}}(\mathrm{DCT}^{\mathrm{w}}(X)) = 2N\,X$.
The isometric transforms $Y = \mathrm{DCT}(X)$ and $X = \mathrm{IDCT}(Y)$ that satisfy Parseval's equality $\sum_k |Y_k|^2 = \sum_j |X_j|^2$ are obtained as

$$Y_k = \mathrm{DCT}(X)_k = \alpha_k\ \mathrm{DCT}^{\mathrm{w}}(X)_k = \alpha_k\, 2\sum_{j=0}^{N-1} X_j \cos\left[\pi\left(j+\frac{1}{2}\right)\frac{k}{N}\right], \tag{7}$$

$$X_j = \mathrm{IDCT}(Y)_j = \mathrm{IDCT}^{\mathrm{w}}(\beta\cdot Y)_j = \beta_0\, Y_0 + \sum_{k=1}^{N-1} \beta_k\, 2Y_k \cos\left[\pi\left(j+\frac{1}{2}\right)\frac{k}{N}\right], \tag{8}$$

$$\text{with}\quad \alpha_k = \begin{cases} \sqrt{1/(4N)}, & k=0 \\ \sqrt{1/(2N)}, & k=1\ldots,N-1 \end{cases} \quad\text{and}\quad \beta_k = \begin{cases} \sqrt{1/N}, & k=0 \\ \sqrt{1/(2N)}, & k=1\ldots,N-1. \end{cases} \tag{9}$$

The normalization factors corresponding to the 2D-DCT of a $N \times M$ image are given by

$$Y_{k,m} = \alpha_k\,\alpha'_m\ \mathrm{DCT2D}^{\mathrm{w}}(X)_{k,m}, \tag{10}$$

$$X_{j,l} = \mathrm{IDCT2D}^{\mathrm{w}}(\widetilde{Y})_{j,l} \qquad \text{with}\quad \widetilde{Y}_{k,m} = \beta_k\,\beta'_m\,Y_{k,m}, \tag{11}$$

where α' and β' are defined as in Equation (9) but for the range $[0\ldots,M]$.

Color transform. Following [20] the orthogonal color transform implemented by the function DecorrelateColors is specified by Equation (12)

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1/\sqrt{6} & -2/\sqrt{6} & 1/\sqrt{6} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{12}$$

6 Conclusion

We made an attempt at refreshing an old school algorithm, DCT denoising. Three generic tools applicable to all denoising algorithms were listed in [11] and claimed to boost any denoising algorithm: these are 1) use a color transform before denoising, 2) use an oracle step, 3) apply aggregation of estimates (in other terms make the algorithm translation invariant).

A fourth generic tool, the multiscale operation, was proposed in [7]. The DCT denoising version proposed here benefits now from the four above listed generic tools. In particular, it extends the DCT denoising version [20] by complementing it with an oracle step and a multiscale structure. The improvement thus obtained is spectacular. First the visual aspect in smooth image parts is much improved by the elimination of most ringing effects. Second the new version increases significantly the PSNR performance of the algorithm and makes it close in performance to the best state of the art algorithms. There is still an observable gap of about 1dB with respect to these algorithms. Yet consider what state of the art algorithms do: instead of denoising individually each patch, as DCT denoising does, they group patches to take advantage of the so-called image self-similarity, and to denoise them jointly. This is the principle introduced in [1] and used in BM3D [3] and Non-local Bayes [10], for example. Thus, we may well deduce that this 1dB increment is attributable to the involvement of image self-similarities. On the other hand the complexity of patch-based algorithms is higher by about two orders of magnitude. Thus DCT denoising remains a valid competitor for low cost implementations.

A Adaptive Aggregation Weights

In adaptive aggregation, overlapping denoised patches are treated as independent estimations of each pixel. Under that assumption, if the variance of each estimator is known, then it is possible to derive optimal per-pixel aggregation weights that minimize the variance of the aggregated estimator.

Optimal aggregation weights under independence hypothesis. Let us assume that the estimates of overlapping patches (of size $s \times s$ pixels) are independent and denote them X_i with $i \in [1, \dots, s^2]$. Let us also assume that the variance σ_i^2 of each X_i is known. Then for a fixed image pixel p all the overlapping estimates $X_i(p)$ are independent random variables with known variance σ_i^2. We want to determine the optimal weights $\alpha_i \geq 0$ for the aggregate estimator $\sum_i \alpha_i X_i(p)$ such that its variance is minimum:

$$\arg \min_{\sum \alpha_i = 1} E\left[\left(\sum_i \alpha_i (X_i(p) - E[X_i(p)])\right)^2\right]. \tag{13}$$

In the following we shall denote $X_i(p)$ as X_i. Then, developing we have

$$\arg \min_{\sum \alpha_i = 1} E\left[\sum_i \alpha_i^2 \left(X_i - E[X_i]\right)^2\right] + \underbrace{E\left[\sum_{i \neq j} \alpha_i \alpha_j \left(X_i - E[X_i]\right)\left(X_j - E[X_j]\right)\right]}_{=0 \text{ independence}} = \tag{14}$$

$$\arg \min_{\sum \alpha_i = 1} \sum_i \alpha_i^2 \underbrace{E\left[\left(X_i - E[X_i]\right)^2\right]}_{\sigma_i^2} \tag{15}$$

Note that since α_i appear squared in the above equation, there is no need to enforce the positivity as we can always choose a non-negative weight. The above constrained optimization problem is solved

by Lagrange multipliers

$$L(\{\alpha\}_i, \lambda) = \sum_i \alpha_i^2 \sigma_i^2 - \lambda(\sum_i \alpha_i - 1) \tag{16}$$

yielding the condition

$$2\alpha_i \sigma_i^2 = \lambda, \quad \forall i. \tag{17}$$

By imposing $\sum_i \alpha_i = 1$ and replacing $\lambda = 2(\sum_i \sigma_i^{-2})^{-1}$ in (17) we get to the optimal weights

$$\alpha_i = \frac{\sigma_i^{-2}}{\sum_i \sigma_i^{-2}} \quad \forall i. \tag{18}$$

Estimator variance. The variance of a patch estimate can be due to the residual noise after filtering. Since the filtering is done in the frequency domain this leads to oscillatory effects in the result. For a patch that has undergone hard thresholding (Algorithm 2) if N_{P_i} denotes the number of nonzero DCT coefficients in the i-th patch after thresholding, then Parseval's formula yields an estimate of the variance due to the noise remaining in the patch as $\sigma^2 N_{P_i}$. Plugging this estimate in (18) we get the aggregation weights

$$\frac{N_{P_i}^{-1}}{\sum_i N_{P_i}^{-1}} \tag{19}$$

that are used in [3] and in Algorithm 2.

A similar reasoning for the Wiener filtering step (Algorithm 3) yields the aggregation weights

$$\frac{\|\rho_{P_i}\|^{-2}}{\sum_i \|\rho_{P_i}\|^{-2}}, \tag{20}$$

where ρ_{P_i} denotes the vector (of length s^2) formed with the Wiener coefficients ρ_P (line 13 of Algorithm 3) of the i-th patch. This justifies the choice of aggregation weights in the second step of BM3D [3]. Let us note that these aggregation weights based on the Wiener coefficients such as N_{P_i} or ρ_{P_i} favor patches with sparser representations in the DCT domain. Also note that the zero frequency of the DCT should not be considered in N_{P_i} or ρ_{P_i} since it does not add variance to the patch.

Practical aggregation weights. The previous derivations are based on the hypothesis of independence between estimates of overlapping patches, which is not true. For this reason in practice other weight choices could give better results. We experimented with different criteria for setting the estimator variance (in Equation (18)) for the first and second step of the DCT denoising algorithm. For the first step we concluded that setting the weights based on N_{P_i} as in (19) already yields the highest PSNR.

For the second step we observed that excluding the zero-frequency from the coefficient vector ρ_P yielded slightly higher PSNR, so we denote this modified vector without the zero-frequency by ρ_0. We also observed that patches with higher energy also tend to introduce more oscillatory effects. For this reason we also considered computing the aggregation weights using the DCT coefficients of the denoised patch itself but excluding the zero frequency, which we denote $\widehat{b}_0$. In addition, functions other than the squared $\ell 2$-norm are also explored.

In Figure 9 we summarize the results of various estimators derived from ρ_0 and $\widehat{b}_0$, and consider the choice between $\ell 2$ and $\ell 1$-norm, squared or not. For the evaluation we used the DCT8 and MS DCT8 algorithms with two noise levels $\sigma = 30$ and 50. We observe that, in terms of PSNR, the aggregation weights lead to worse results in the case of the non-multiscale algorithm.

From the graphs in Figure 9 we see that, for all the weighting strategies, at least one quartile of the tests result in a PSNR loss. These images are usually dominated by textures as illustrated

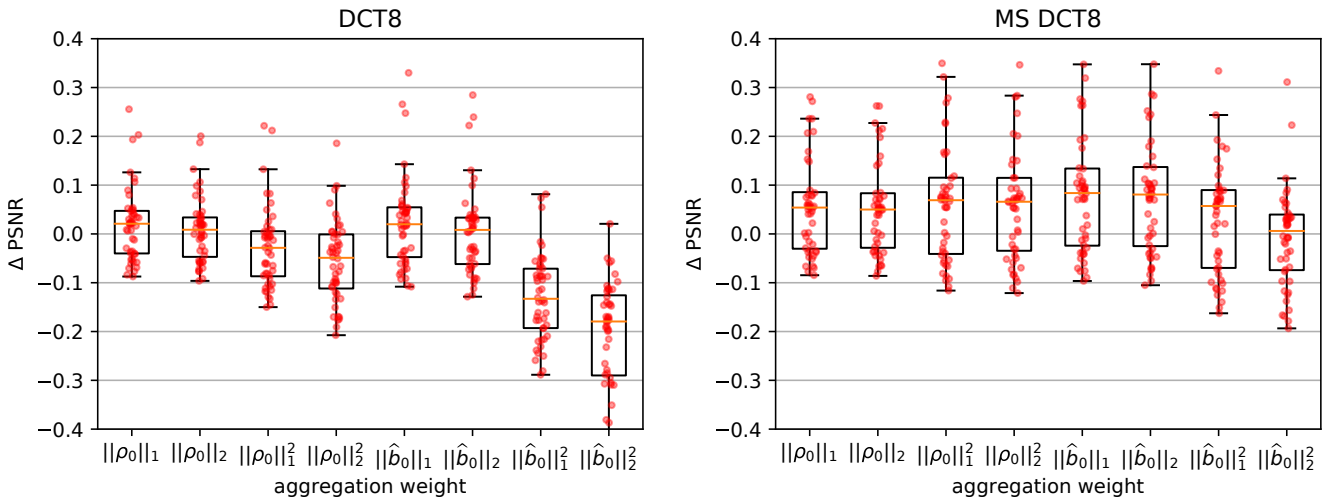

Figure 9: Effect of different aggregation weights on the DCT and MS DCT algorithms (with 8×8 patches) with noise $\sigma = 50$ and 30. The PSNR increments (red dots) are computed with respect to the output of the algorithm without aggregation weights for all the images of the training database (Figure 5). The black boxes extend from the lower to upper quartile values of the data and the orange line marks the mean.

in Figure 11. Figure 10 illustrates the higher quantile which corresponds to images with larger objects and flat areas. Visual inspection of the results obtained with different weighting strategies in Figures 9 and 11 reveal that the aggregation weights indeed reduce the ringing. But, in the end, there is little difference between the different weighting choices. So for the final algorithm we keep the aggregation weights based on the optimal weight derivation (20).

Acknowledgment

Work partly founded by BPIFrance and Région Ile de France in the framework of the FUI 18 Plein Phare project, Office of Naval research grant N00014-17-1-2552, ANR-DGA project ANR-12-ASTR-0035. The authors would also like to thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the final version of the paper.

Image Credits

 Miguel Colom CC-BY

 Jean-Michel Morel CC-BY

 Jean-Michel Morel CC-BY

 Nicola Pierazzo CC-BY

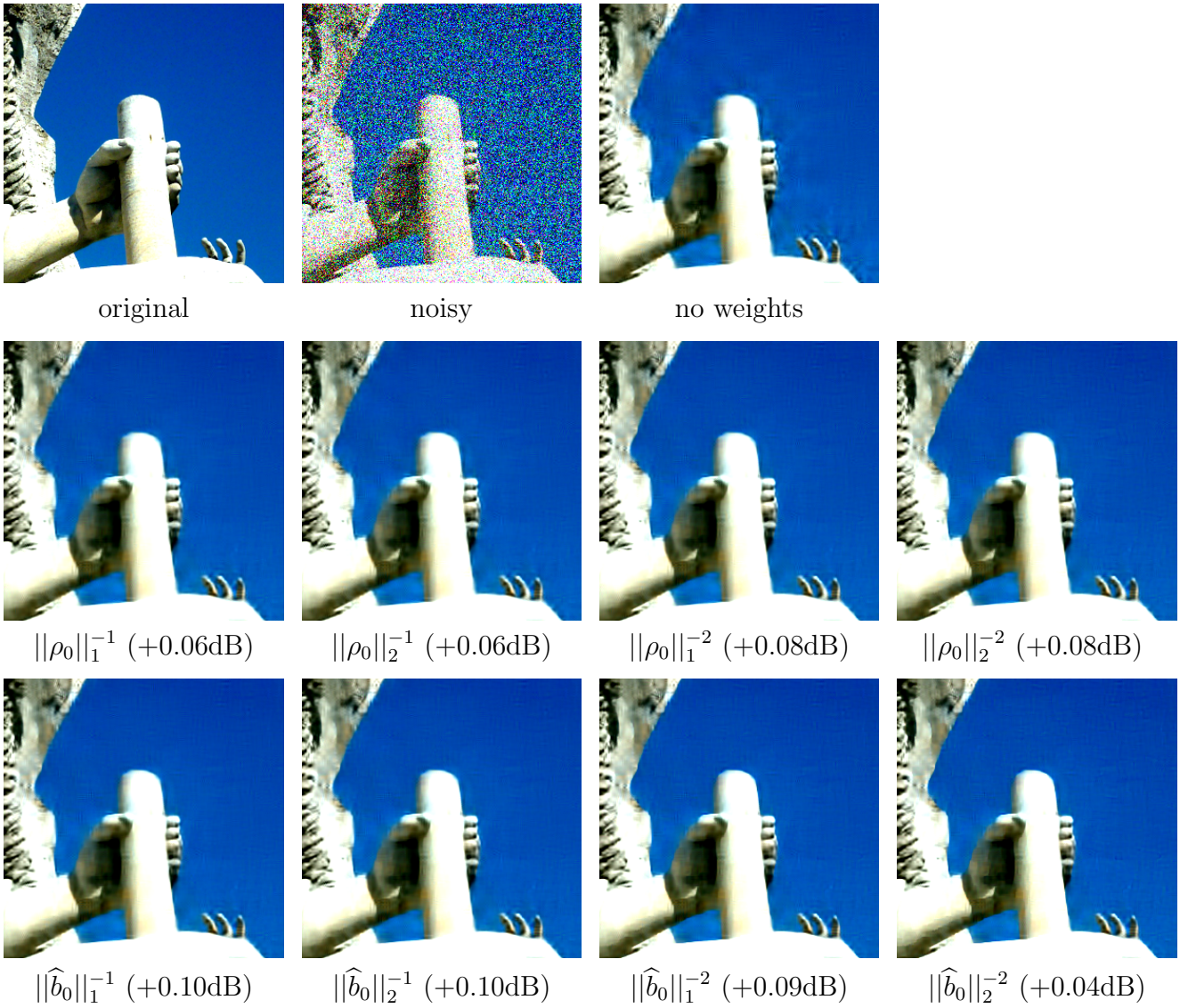

Figure 10: Effect of different aggregation weights on the MS DCT algorithm (with 8×8 patches) with noise $\sigma = 50$. This image illustrates a case in which aggregation weights yield an important PSNR increase, usually for images depicting large and sharp geometric structures. The PSNR increments are computed with respect to the output of the algorithm without aggregation weights. Visually all the aggregation weights produce similar results, improving notably with respect to the non-weighted aggregation. The contrast on these crops has been increased to highlight the ringing artifacts (better seen in the electronic version).

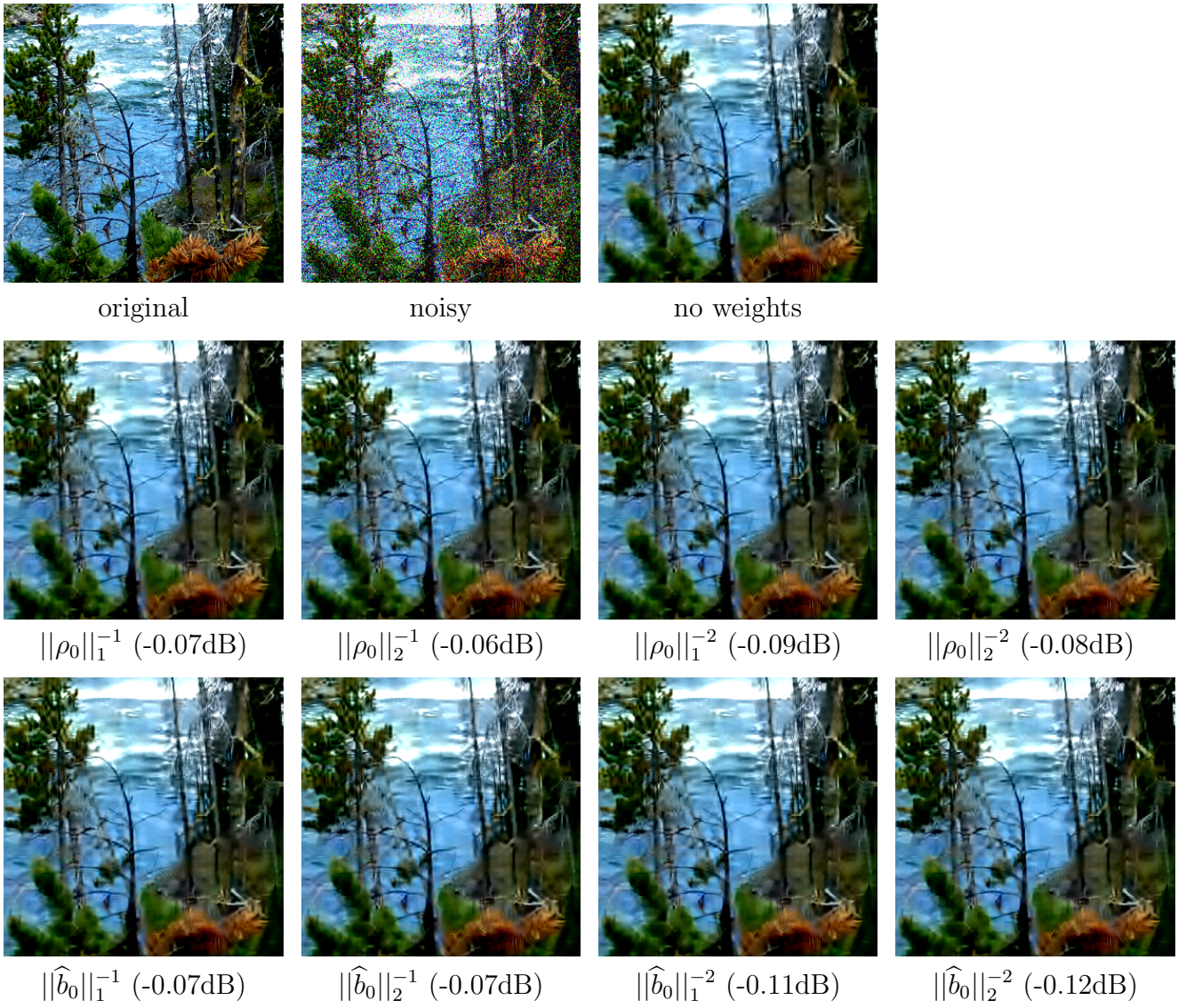

Figure 11: Effect of different aggregation weights on the MS DCT algorithm (with 8×8 patches) with noise $\sigma = 50$. This image illustrates a case in which aggregation weighting results in PSNR loss, usually in images dominated by textures. The PSNR increments are computed with respect to the output of the algorithm without aggregation weights. For this image all the aggregation weights produce similar results, which are also barely distinguishable from the non-weighted version. The contrast on these crops has been increased to highlight the artifacts (better seen in the electronic version).

References

[1] A. Buades, B. Coll, and J-M. Morel, *The staircasing effect in neighborhood filters and its solution*, IEEE Transactions on Image Processing, 15 (2006), pp. 1499–1505. https://doi.org/10.1109/TIP.2006.871137.

[2] H.C. Burger and S. Harmeling, *Improving denoising algorithms via a multi-scale meta-procedure*, in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 6835 LNCS, 2011, pp. 206–215. https://doi.org/10.1007/978-3-642-23123-0_21.

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

[4] D.L. Donoho and J.M. Johnstone, *Ideal spatial adaptation by wavelet shrinkage*, Biometrika, 81 (1994), pp. 425–455. https://doi.org/10.1093/biomet/81.3.425.

[5] S. Durand and J. Froment, *Artifact free signal denoising with wavelets*, in IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01), vol. 6, IEEE, 2001, pp. 3685–3688.

[6] F. Estrada, D. Fleet, and A. Jepson, *Stochastic Image Denoising*, Procedings of the British Machine Vision Conference 2009, (2009), pp. 117.1–117.11. https://doi.org/10.5244/C.23.117.

[7] G. Facciolo, N. Pierazzo, and J-M. Morel, *Conservative Scale Recomposition for Multi-scale Denoising (The Devil is in the High Frequency Detail)*, SIAM Journal on Imaging Sciences, 10 (2017), pp. 1603–1626. https://doi.org/10.1137/17M1111826.

[8] D. Gnanadurai and V. Sadasivam, *Image De-Noising Using Double Density Wavelet Transform Based Adaptive Thresholding Technique*, International Journal of Wavelets, Multiresolution and Information Processing, 03 (2005), pp. 141–152. https://doi.org/10.1142/S0219691305000701.

[9] J.. Huang and D. Mumford, *Statistics of natural images and models*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (1999), pp. 541–547. https://doi.org/10.1109/CVPR.1999.786990.

[10] M. Lebrun, A. Buades, and J-M. Morel, *Implementation of the Non-Local Bayes (NL-Bayes) Image Denoising Algorithm*, Ipol, 3 (2013), pp. 1–42. https://doi.org/10.5201/ipol.2013.16.

[11] M. Lebrun, M. Colom, A. Buades, and J-M. Morel, *Secrets of image denoising cuisine*, Acta Numerica, 21 (2012), pp. 475–576. https://doi.org/10.1017/S0962492912000062.

[12] A.B. Lee, D. Mumford, and J. Huang, *Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model*, International Journal of Computer Vision, 41 (2001), pp. 35–59. https://doi.org/10.1023/A:1011109015675.

[13] H-Q. Li, S-Q. Wang, and C-Z. Deng, *New Image Denoising Method Based Wavelet and Curvelet Transform*, WASE International Conference on Information Engineering, 1 (2009). https://doi.org/10.1109/ICIE.2009.228.

[14] R. Oktem, L. Yarovslavsky, and K. Egiazarian, *Signal and image denoising in transform domain and wavelet shrinkage: A comparative study*, in 9th European Signal Processing Conference (EUSIPCO 1998), Sept 1998, pp. 1–4.

[15] V. Papyan and M. Elad, *Multi-scale patch-based image restoration*, IEEE Transactions on Image Processing, 25 (2016), pp. 249–261. https://doi.org/10.1109/TIP.2015.2499698.

[16] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli, *Image denoising using scale mixtures of Gaussians in the wavelet domain*, IEEE Transactions on Image Processing, 12 (2003), pp. 1338–1351. https://doi.org/10.1109/TIP.2003.818640.

[17] U. Rajashekar and E.P. Simoncelli, *Multiscale Denoising of Photographic Images*, in The Essential Guide to Image Processing, 2009, pp. 241–261. https://doi.org/10.1016/B978-0-12-374457-9.00011-1.

[18] J. Sulam, B. Ophir, and M. Elad, *Image denoising through multi-scale learnt dictionaries*, in IEEE International Conference on Image Processing (ICIP), 2014, pp. 808–812. https://doi.org/10.1109/ICIP.2014.7025162.

[19] L.P. Yaroslavsky, K.O. Egiazarian, and J.T. Astola, *Transform domain image restoration methods: review, comparison, and interpretation*, in Photonics West 2001-Electronic Imaging, International Society for Optics and Photonics, 2001, pp. 155–169.

[20] G. Yu and G. Sapiro, *DCT image denoising: a simple and effective image denoising algorithm*, Image Processing On Line, (2011). https://doi.org/10.5201/ipol.2011.ys-dct.

[21] D. Zoran and Y. Weiss, *From learning models of natural image patches to whole image restoration*, in 2011 International Conference on Computer Vision, IEEE, 2011, pp. 479–486.

[22] ——, *Natural images, Gaussian mixtures and dead leaves*, in Advances in Neural Information Processing Systems, 2012, pp. 1736–1744.