# Optimizing the Data Adaptive Dual Domain Denoising Algorithm

Nicola Pierazzo[1(✉)], Jean-Michel Morel[1], and Gabriele Facciolo[1,2]

[1] CMLA, École Normale Supérieure de Cachan, Cachan, France
nicola.pierazzo@cmla.ens-cachan.fr
[2] IMAGINE/LIGM, École Nationale des Ponts et Chaussées, Champs-sur-Marne, France

**Abstract.** This paper presents two new strategies that greatly improve the execution time of the DA3D Algorithm, a new denoising algorithm with state-of-the-art results. First, the weight map used in DA3D is implemented as a quad-tree. This greatly reduces the time needed to search the minimum weight, greatly reducing the overall computation time. Second, a simple but effective tiling strategy is shown to work in order to allow the parallel execution of the algorithm. This allows the implementation of DA3D in a parallel architecture. Both these improvements do not affect the quality of the output.

**Keywords:** Image denoising · Quad-tree · Parallel processing

## 1 Introduction

Image denoising is one of the fundamental image restoration challenges [18]. It consists in estimating an unknown noiseless image $\mathbf{y}$ from a noisy observation $\mathbf{x}$. We consider the classic image degradation model

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \tag{1}$$

where the observation $\mathbf{x}$ is contaminated by an additive white Gaussian noise $\mathbf{n}$ of variance $\sigma^2$. All denoising methods assume some underlying image regularity. Depending on this assumption they can be divided, among others, into transform-domain and spatial-domain methods.

Transform domain methods work by shrinking (or thresholding) the coefficients of some transform domain [7,14,25]. The Wiener filter [28] is one of the first such methods operating on the Fourier transform. Donoho et al. [5] extended it to the wavelet domain.

Space-domain methods traditionally use a local notion of regularity with edge-preserving algorithms such as total variation [24], anisotropic diffusion [19], or the bilateral filter [27]. Nowadays however spatial-domain methods achieve remarkable results by exploiting the self-similarities of the image [1]. These patch-based methods are non-local as they denoise by averaging similar patches


A. Pardo and J. Kittler (Eds.): CIARP 2015, LNCS 9423, pp. 358–365, 2015.
DOI: 10.1007/978-3-319-25751-8_43

in the image. Patch-based denoising has developed into attempts to model the patch space of an image, or of a set of images. These techniques model the patch as sparse representations on dictionaries [4,6,15,16,29], using Gaussian Scale Mixtures models [23,29,30], or with non-parametric approaches by sampling from a huge database of patches [12,13,17,21].

Current state-of-the-art denoising methods such as BM3D [3] and NL-Bayes [11] take advantage of both space- and transform-domain approaches. They group similar image patches and jointly denoise them by collaborative filtering on a transformed domain. In addition, they proceed by applying two denoising stages, the second stage using the output of the first one as its guide.

Some recently proposed methods use the result of a different algorithm as their guide for a new denoising step. Combining for instance, nonlocal principles with spectral decomposition [26], or BM3D with neural networks [2]. This allows one to mix different denoising principles, thus yielding high quality results [2,26].

DDID [9] is an iterative algorithm that uses a guide image (from a previous iteration) to determine spatially uniform regions to which Fourier shrinkage could be applied without introducing ringing artifacts. Several methods [8,10,20] use a single step of DDID with a guide image produced by a different algorithm. This yields much better results than the original DDID. Unfortunately, DDID has a prohibitive computational cost, as it paradoxically denoises a large patch to recover a single pixel. Moreover, contrary to other methods, aggregation of these patches doesn't improve the results since it introduces blur.

More recently, Data-Adaptive Dual Domain Denoising (DA3D) [22] was presented to address those issues. By using just a small fraction of the patches, it avoids unnecessary computations in the uniform areas of the image. Moreover, DA3D uses a more complex estimation of the image shape to reduce staircasing artifacts. In order to choose the patches to process, DA3D keeps track of the partial aggregation weights, and iteratively selects the patch with the smaller weight. The search for this patch can be expensive, especially on large images. Moreover, since the choice of a patch depends on the previous ones, there is no straight-forward method to parallelize this algorithm.

**Contribution.** This paper proposes an effective method to accelerate the search step of DA3D, and an approach to run the algorithm in a multi-processor environment. With these, DA3D can run in reasonable time even on medium-large images, making it even more interesting for real-world applications.

In order to accelerate the search step, a quadtree is used to store and update the minimum value. This decreases the complexity of the search from $O(n)$ to $O(\log n)$, where n is the number of pixels in the image.

To allow the execution of the algorithm on a multi-core architecture, it is noted that the simple strategy of dividing the images in stripes is effective and yields the same results than the single-process version of the algorithm.

Section 2 recalls the DA3D algorithm. Section 3 tackles the problems of tracking the minimum weight and of parallel execution. Section 4 shows some results and section 5 presents the conclusions.

## 2 Data-Adaptive Dual Domain Denoising

This section describes the DA3D algorithm, as presented in [22].

To denoise the area around the pixel $p$ from the noisy image $\mathbf{y}$ DA3D extracts a $64\times64$ pixel block centered in $p$ (denoted $y$) and the corresponding block $g$ from the guide image $\mathbf{g}$. An affine model $P(q) = \langle \boldsymbol{\alpha}, q\rangle + \beta$ of the block is estimated computing a weighted least squares regression

$$\min_P \sum [y(q) - P(q)]^2 \cdot K_{reg}(q), \tag{2}$$

with the constraint $P(p) = g(p)$, where the sum is computed over the domain of $y$ and $K_{reg}$ is a bilateral weight function

$$K_{reg}(q) = \exp\left(-\frac{|g(q) - g(p)|^2}{\gamma_{rr}\sigma^2} - \frac{|q-p|^2}{2\sigma_{sr}^2}\right), \tag{3}$$

which selects the parts of the block that gets approximated by $P$. Once estimated, the local plane $P$ is subtracted from the patch, effectively removing shades and gradients.

The blocks are processed to eliminate discontinuities that may cause artifacts in the subsequent frequency-domain denoising. To that end, a bilateral weight function $k$ is derived from the guide $g$, to identify the pixels of the block belonging to the same object as the center $p$.

$$k(q) = \exp\left(-\frac{|g(q) - g(p)|^2}{\gamma_r\sigma^2}\right)\exp\left(-\frac{|q-p|^2}{2\sigma_s^2}\right). \tag{4}$$

The first term identifies the pixels belonging to the same structure as $p$, by selecting the ones with a similar color in the guide, while the second term removes the periodization discontinuities associated with the Fourier transform.

The weights in $k$ are then used to modify $y$ and $g$ in order to remove their discontinuities and to obtain $y_m$ and $g_m$ (see lines 12-13 of Table 1). In this way the "relevant" part of the blocks (similar to the central pixel) is retained by $k$, and its average value is assigned to the rest. The modified block $y_m$ is denoised by shrinkage of its Fourier coefficients using $g_m$ as an oracle (lines 14-18 of Table 1). Then the "modification" of the patches is reverted and the regression plane $P$ is added back to the block. The shrinkage assumes that the image $y$ contains additive white Gaussian noise.

For color images, $k$ is computed by using the Euclidean distance, while the shrinkage is done independently on each channel of the YUV color space.

Since the denoising remains valid for all pixels in the "relevant" part of the block, the processed blocks are aggregated to form the final result. The aggregation weights are the squares of the weights (4).

The image blocks to be processed are selected using a greedy approach. At each iteration a weight map $\mathbf{w}$ with the sum of the aggregation weights is updated. This weight map permits to identify the pixel in the image with the

**Table 1.** Pseudo-code for DA3D. Variables in **bold** denote whole images, while *italics* denote single blocks. Multiplication and division are pixel-wise.

**Input:** **y** (noisy image), **g** (guide)
**Output:** denoised image

1 $\mathbf{w} \leftarrow 0$
2 **out** $\leftarrow 0$
3 **while** $\min(\mathbf{w}) < \tau$ **do**
4 $p \leftarrow \arg\min(\mathbf{w})$
5 $y \leftarrow \textsc{ExtractPatch}(\mathbf{y}, p)$
6 $g \leftarrow \textsc{ExtractPatch}(\mathbf{g}, p)$
7 $K_{reg} \leftarrow \textsc{ComputeKreg}(g)$ // regression weight, eq. 3
8 $P \leftarrow \arg\min_P \sum [y(q) - P(q)]^2 \cdot K_{reg}(q)$ // regression plane, eq. 2
9 $y \leftarrow y - P$ // subtract plane from the block
10 $g \leftarrow g - P$ // and from the guide
11 $k \leftarrow \textsc{ComputeK}(g)$ // eq. 4
12 $y_m \leftarrow k \cdot y + (1 - k) \left( \frac{\sum k(l) y(l)}{\sum k(l)} \right)$
13 $g_m \leftarrow k \cdot g + (1 - k) \left( \frac{\sum k(l) g(l)}{\sum k(l)} \right)$
14 $Y \leftarrow \mathrm{DFT}(y_m)$
15 $G \leftarrow \mathrm{DFT}(g_m)$
16 $\sigma_f^2 \leftarrow \sigma^2 \sum k(q)^2$
17 $K \leftarrow \begin{cases} 1 & \text{if } f = 0 \\ \exp\left( -\frac{\gamma_f \sigma_f^2}{\lvert G(f) \rvert^2} \right) & \text{otherwise} \end{cases}$ // shrinkage
18 $x_m \leftarrow \mathrm{IDFT}(K \cdot Y)$
19 $x \leftarrow \left[ x_m - (1 - k) \left( \frac{\sum k(l) y(l)}{\sum k(l)} \right) \right] / k$ // revert line 12
20 $x \leftarrow x + P$ // add plane back to the block
21 $aggw \leftarrow k \cdot k$ // aggregation weight
22 $\mathbf{w} \leftarrow \textsc{AddPatchAt}(p, \mathbf{w}, aggw)$ // accumulate in the correct position
23 **out** $\leftarrow \textsc{AddPatchAt}(p, \mathbf{out}, aggw \cdot x)$
24 **return out**/**w**

lowest aggregation weight, which will be selected as the center of the next block to process (line 4 of Table 1). The process iterates until the total weight for each pixel becomes larger than a threshold $\tau$. The total number of processed blocks depends on the image complexity. The centers of the effectively processed blocks are concentrated on edges and details.

The parameters $\sigma_{sr}$, $\gamma_{rr}$, $\sigma_s$, $\gamma_r$, $\gamma_f$ and $\tau$ are specific of the algorithm, and $\sigma$ is the standard deviation of the noise.

## 3 Improvements

### 3.1 Tracking the Minimum Weight

In the implementation of DA3D the authors select the position with the lowest aggregation weight in **w** with a simple linear search. This approach shows its

limit when the size of the image increases. In fact, for every processed block, the computation needed to find it is of the order of $O(n)$, with $n$ the number of pixels of the image. Therefore, under the reasonable assumption that the number of processed blocks is a fraction of the total (from the original article, between 1% and 20%), and since the denoising of a block is performed in a bounded time, the complexity of the algorithm is $O(n^2)$. This is peculiar, because the algorithm appears local in its nature.

We propose to use a quad-tree to keep track of the minimum. The weight map $\mathbf{w}$ is in the leaves of the tree, and every node contains the minimum value of its four children. This can also be interpreted as a multi-scale version of $\mathbf{w}$, built using a *min* filter. The space complexity for this data structure is

$$\sum_{i=0}^{\log n} \frac{n}{4^i} \leq \frac{4}{3} n = O(n) \tag{5}$$

because every "layer" of the tree is 25% smaller than the previous one.

In order to retrieve the position of the minimum value, one has simply to traverse the tree from the root to the leaf, always choosing one of the children with the minimum value. This guarantees that the chosen pixel is a global minimum for $\mathbf{w}$, and has time complexity $O(\log n)$.

To update the tree, it suffices to update the appropriate leaves, and then recompute the minima in the upper nodes until the top. Since the aggregation is done one patch at a time, it is simple to calculate which nodes need to be updated, thus avoiding to recompute the values for areas in which $\mathbf{w}$ has not changed. The time complexity for this update is $O(k)$, where $k$ is the number of pixels of the patch that is aggregated. Since $k$ is constant, the aggregation does not increase the complexity of the algorithm.

One could be tempted to update the values one by one. Although this could be simpler to implement, it is slower, having a time complexity of $O(k \log n)$. Using this data structure instead, the total complexity of the algorithm becomes $O(n \log n)$, which allows to denoise bigger images.

### 3.2 Parallel Processing

Since DA3D selects the patches to denoise in a greedy fashion, it is impossible to know where the next patch will be prior to the aggregation step of the current one. This makes parallelization more complex than in other denoising algorithms.

In order to denoise a pixel p, the algorithm uses the other pixels inside a $(64 \times 64)$ window, all the pixels needed to denoise $p$ are at a distance of at most 32. This makes the algorithm *local*, and allows to solve the problem of parallelism by just splitting the image in tiles. Each tile can be denoised separately, and then the results can be combined together.

It is clear that the patches chosen in this way will not correspond exactly with the patches chosen without parallelism. The main difference can be an oversampling of the areas near the edges, since the weights from a neighboring tile are not taken into account. This could result in a slight overhead in the processing time.

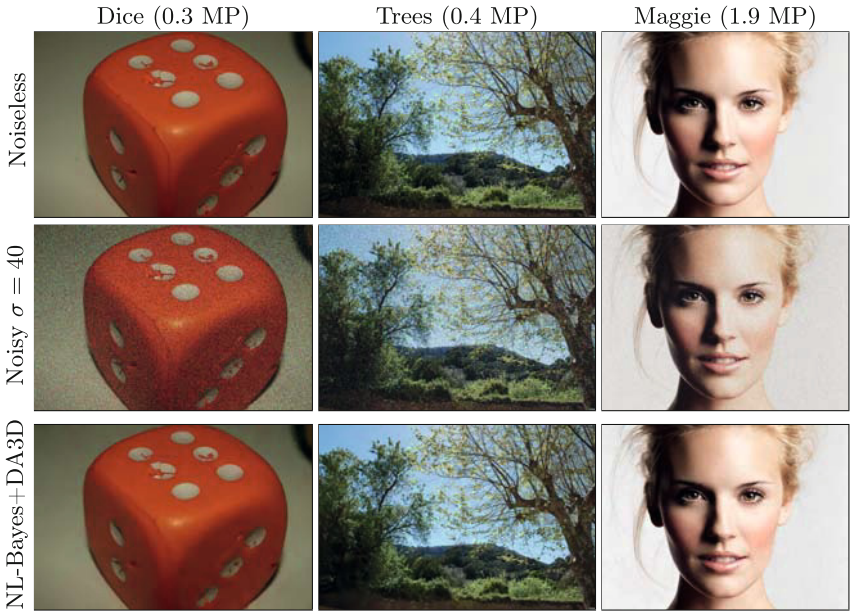


**Fig. 1.** Sample images, with noisy and denoised version. It is advised to zoom in on the digital version for more details. The results of the DA3D with the different improvements are visually identical.

However, the experiments show that the overhead is negligible, and the results of the simple and parallel versions of the algorithm are identical from a practical standpoint. With bigger images, the overlap area becomes smaller, therefore the factor of acceleration should become even closer to the number of processors.

## 4 Results

We tested the methods of Section 3 on the test images of Figure 1. We selected this set to be as varied as possible. Dice contains many smooth areas, Trees is mainly composed of texture and Maggie is a relatively large image, for which the improvement due to the quad-tree is more noticeable. On those images, we tested the DA3D algorithm with the suggested improvements, and we timed the execution on a dual-core laptop. The results are shown in Table 2.

The improvements do not change the output image significantly. The comparison of PSNR is shown in Table 3. Notice how the value of PSNR does not change among the different versions of the algorithm.

## 5 Conclusion

This paper presented two new strategies to improve the execution time of the DA3D Algorithm. First, a quad-tree structure was used as the weight map $\mathbf{w}$.

**Table 2.** Runtimes for different versions of DA3D. Baseline represents the time of the original version. Notice how the improvement of the quad-tree is more prominent on bigger images.

| Image | Baseline | Quad-Tree | Parallel (2 cpu) | Tree + Parallel |
|---|---|---|---|---|
| Dice | 2.12s | 1.51s | 1.12s | 0.92s |
| Trees | 19.10s | 12.78s | 10.68s | 8.56s |
| Maggie | 28.05s | 8.26s | 15.03 | 5.36 |

**Table 3.** PSNR values for different versions of DA3D, also compared with Non-local Bayes[11].

| Image | NL-Bayes | Baseline | Quad-Tree | Parallel | Tree + Par. |
|---|---|---|---|---|---|
| Dice | 36.17 dB | 37.90 dB | 37.90 dB | 37.90 dB | 37.90 dB |
| Trees | 23.49 dB | 23.71 dB | 23.71 dB | 23.71 dB | 23.71 dB |
| Maggie | 33.48 dB | 34.54 dB | 34.54 dB | 34.54 dB | 34.54 dB |

This greatly reduces the time needed to search for the minimum weight, without affecting the time needed to update the weight map in a significant way. Then, a tiling strategy is shown to work to allow the parallelization of the code with minimum overhead.

**Acknowledgments.** Work partly founded by Centre National d'Etudes Spatiales (MISS Project), European Research Council (advanced grant Twelve Labours), Office of Naval research (ONR grant N00014-14-1-0023), DGA Stéréo project, ANR-DGA (project ANR-12-ASTR-0035), FUI (project Plein Phare) and Institut Universitaire de France.

## References

1. Buades, A., Coll, B., Morel, J.M.: A review of image denoising algorithms, with a new one. SIAM Mult. Model. Simul. **4**(2) (2006)
2. Burger, H.C., Schuler, C., Harmeling, S.: Learning how to combine internal and external denoising methods. In: Weickert, J., Hein, M., Schiele, B. (eds.) GCPR 2013. LNCS, vol. 8142, pp. 121–130. Springer, Heidelberg (2013)
3. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3d transform-domain collaborative filtering. IEEE TIP **16**(82) (2007)
4. Dong, W., Shi, G., Li, X.: Nonlocal image restoration with bilateral variance estimation: A low-rank approach. IEEE TIP **22**(2) (2013)
5. Donoho, D.L., Johnstone, J.M.: Ideal spatial adaptation by wavelet shrinkage. Biometrika (1994)
6. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE TIP **15**(12) (2006)
7. Gnanadurai, D., Sadasivam, V.: Image denoising using double density wavelet transform based adaptive thresholding technique. IJWMIP **03**(01) (2005)
8. Knaus, C.: Dual-domain image denoising. Ph.D. thesis, Diss. Univ. Bern (2013)

9. Knaus, C., Zwicker, M.: Dual-domain image denoising. IEEE ICIP (2013)
10. Knaus, C., Zwicker, M.: Progressive image denoising. IEEE TIP **23**(7) (2014)
11. Lebrun, M., Buades, A., Morel, J.M.: Implementation of the "non-local bayes" (NL-bayes) image denoising algorithm. Image Processing On Line (2013)
12. Levin, A., Nadler, B.: Natural image denoising: Optimality and inherent bounds. IEEE CVPR (2011)
13. Levin, A., Nadler, B., Durand, F., Freeman, W.T.: Patch complexity, finite pixel correlations and optimal denoising. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part V. LNCS, vol. 7576, pp. 73–86. Springer, Heidelberg (2012)
14. Li, H.Q., Wang, S.Q., Deng, C.Z.: New image denoising method based wavelet and curvelet transform. WASE ICIE **1** (2009)
15. Mairal, J., Bach, F., Ponce, J., Sapiro, G., Zisserman, A.: Non-local sparse models for image restoration. IEEE ICCV (2009)
16. Mairal, J., Sapiro, G., Elad, M.: Learning multiscale sparse representations for image and video restoration. SIAM Mult. Model. Simul. **7**(1) (2008)
17. Mosseri, I., Zontak, M., Irani, M.: Combining the power of internal and external denoising. IEEE ICCP (2013)
18. Motwani, M.C., Gadiya, M.C., Motwani Jr., R.C., Harris, F.C.: Survey of image denoising techniques. GSPX (2004)
19. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE TPAMI **12**(7) (July 1990)
20. Pierazzo, N., Lebrun, M., Rais, M., Morel, J.M., Facciolo, G.: Non-local dual image denoising. IEEE ICIP (2014)
21. Pierazzo, N., Rais, M.: Boosting shotgun denoising by patch normalization. IEEE ICIP (2013)
22. Pierazzo, N., Rais, M., Morel, J.M., Facciolo, G.: DA3D: Fast and data adaptive dual domain denoising. IEEE ICIP (2015)
23. Portilla, J., Strela, V., Wainwright, M.J., Simoncelli, E.P.: Image denoising using scale mixtures of gaussians in the wavelet domain. IEEE TIP (2003)
24. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D **60** (1992)
25. Starck, J.L., Candès, E.J., Donoho, D.L.: The curvelet transform for image denoising. IEEE TIP **11**(6) (2002)
26. Talebi, H., Milanfar, P.: Global image denoising. IEEE TIP **23**(2) (2014)
27. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. IEEE ICCV (1998)
28. Wiener, N.: Extrapolation, Interpolation, and Smoothing of Stationary Time Series. The MIT Press (1964)
29. Yu, G., Sapiro, G., Mallat, S.: Image modeling and enhancement via structured sparse model selection. IEEE ICIP (2010)
30. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. IEEE ICCV (November 2011)