IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2019–09–05.
Submitted on 2019–08–06, accepted on 2019–08–10.
ISSN 2105–1232 © 2019 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.2019.274

A Study of Two CNN Demosaicking Algorithms

Thibaud Ehret, Gabriele Facciolo

CMLA, CNRS, ENS Paris-Saclay, Université Paris-Saclay
thibaud.ehret@cmla.ens-cachan.fr

Communicated by Miguel Colom and Jean-Michel Morel   *Demo edited by* Thibaud Ehret

Abstract

Most cameras capture the information of only one color for a given pixel. This results in a mosaicked image that must be interpolated to get three colors at each pixel. The step going from a mosaicked image to a regular RGB image is called demosaicking. This paper studies two recent demosaicking methods based on convolutional neural networks that achieve artifact-free state-of-the-art results: Deep joint demosaicking and denoising by Gharbi et al. and Color image demosaicking via deep residual learning by Tan et al. We show that these methods beat by almost two decibels the best human-crafted methods, while being faster by one order of magnitude. This, arguably, seals the destiny of human-crafted methods on this subject.

Source Code

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

Keywords: demosaicking; neural network; CNN

1 Introduction

Most cameras capture only one color per pixel, this color being determined by a color filter array (CFA) located on top of the sensor. The most commonly used CFA is the so-called Bayer pattern, consisting of a regular subsampling of each color channel. This means that each pixel of the resulting raw image contains one third of the necessary information, and that the color channels are sampled on different grids. The problem of interpolating the missing color channels at each pixel, by using the information available in a neighborhood, is called demosaicking. It is a challenging ill-posed inverse problem.

The simplest demosaicking method is the independent bilinear interpolation of each channel. Yet recent methods are far more sophisticated. Getreuer [2] solves the demosaicking by using contours

[1] https://doi.org/10.5201/ipol.2019.274

Thibaud Ehret, Gabriele Facciolo, *A Study of Two CNN Demosaicking Algorithms*, Image Processing On Line, 9 (2019), pp. 220–230.
https://doi.org/10.5201/ipol.2019.274

of objects as guide. Mairal et al. [13] suggested combining group sparsity and dictionary learning to achieve optimal performance. In a series of papers, Kiku et al. [8], [9] and [15] proposed increasingly complex interpolation methods where they use the interpolated Green channel as a guide for the Red and Blue channels. These last four methods may be considered among the best ones designed by humans. A recently emerged trend consists in using convolutional neural networks for image processing and computer vision applications. See for example the early attempt in [19]. These methods have by now reached state-of-the-art results. This has added a new chapter to the success book of applications of neuronal networks in computer vision. They were already celebrated for applications such as classification [11] or object detection [16], but also for low-level vision applications as image denoising [20].

The rest of the paper is organized as follows: In Section 2 we present the method "Deep joint demosaicking and denoising" by Gharbi et al. [3], in Section 3 we present the method "Color image demosaicking via deep residual learning" by Tan et al. [18]. A quantitative and qualitative comparison with previous state-of-the-arts methods is made in Section 4.

2 Deep Joint Demosaicking and Denoising by Gharbi et al. [3]

2.1 Architecture

The network starts by downsampling the CFA image into a four-channel image that then goes through a serie of 14 Conv+bias layers with 64 features and 3 × 3 convolutions. A $15th$ layer of Conv+ Batch Normalization (BN)+ReLu produces 12 features with 3×3 convolutions. It is followed by an upsampling layer producing an RGB image of twice the width and twice the height. The CFA input is split into the respective channels and concatenated to this intermediate RGB image. This layer acts as a residual layer. After this concatenation, a Conv+bias layer is added before the final layer producing the RGB output. The architecture is depicted in Figure 1.

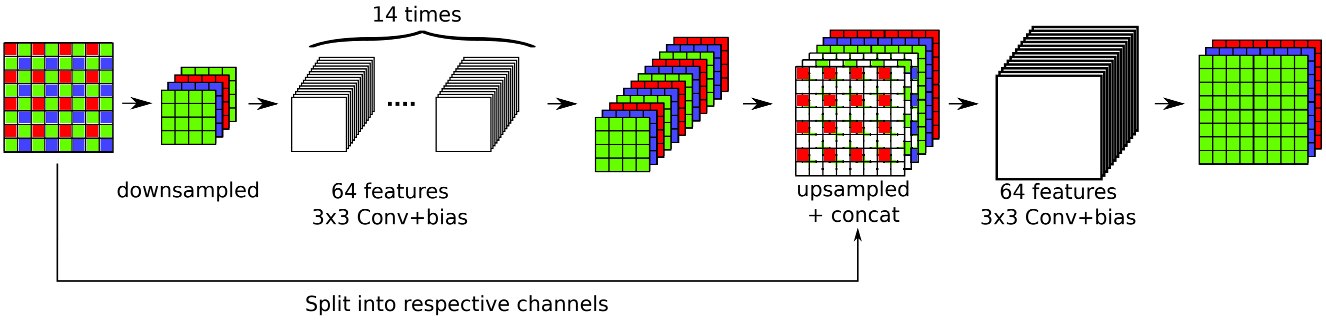

Figure 1: Architecture used for demosaicking by Gharbi et al. [3].

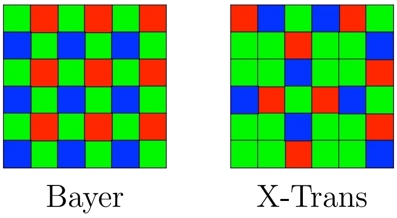

Figure 2: Mosaicking patterns: Bayer (2 × 2) and Fujifilm X-Trans (6 × 6).

The architecture is slightly modified when working with noisy or Fujifilm X-Trans data. For X-Trans data, the image is not downsampled. Indeed Gharbi et al. argue that, since the X-Trans pattern is 6×6 (as illustrated in Figure 2), the downsampling would be too aggressive. The layer acting as a residual layer is also removed. For noisy data, the noise level (noise standard deviation) σ is concatenated into the downsampling layer (using a channel with values all equal to σ). The rest of the network stays the same.

2.2 Training

Loss. A classic L_2 loss is used for training in all three cases (Bayer demosaicking, X-Trans demosaicking and joint demosaicking and denoising). All noise levels are trained at the same time for joint demosaicking and denoising.

Training dataset. A first pre-training was done using natural images. In particular 1.3 millions images from ImageNet [1] and 1 million images from MirFlickr [6] were chosen for this pre-training. The images chosen had a minimum size of $16M$ pixels and then were downsampled by a factor of 4. Data augmentation was performed by flipping, rotating and applying a 1 pixel-shift. CFA images were obtained by masking.

Gharbi et al. argue that classic metrics such as L_2 and PSNR are not much impacted by demosacking artifacts. Moreover, since difficult patches are rare, this means that it would be hard to train a network for demosaicking without artifacts unless the training set be biased towards the creation of such artifacts. For this reason, they created two datasets (one for each type of artifact, namely luminance and moiré) containing only difficult patches used to fine-tune the network. Using the pre-trained network, luminance artifacts are found using the HDR-VDP2 metric [14]. Moiré artifacts are found by looking at frequencies that have a much larger energy in Fourier for the demosaicked patch. Figure 3 shows examples of patches in these datasets.

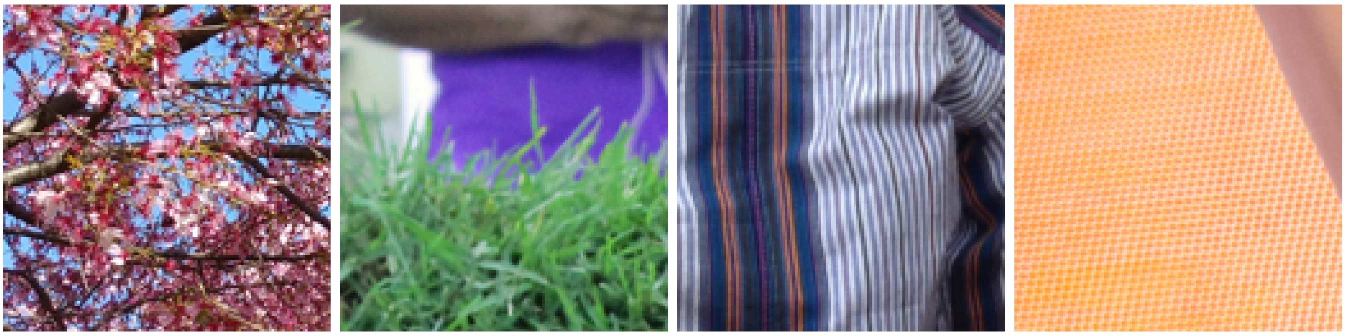

Figure 3: Example of images from the luminance dataset (two on the left) and the moiré dataset (two on the right).

Weights initialization. He et al. suggested in [5] a way to generalize the "*Xavier*" initialization initially proposed by Glorot and Bengio [4] to take into account ReLUs non-linearity. This allows for a good convergence even for very deep models. He et al. derive conditions on the weights of the kernels at each layer to avoid the problems of vanishing or exploding gradients and vanishing or exploding signal. It leads to initializing all kernels using a zero-mean Gaussian distribution with variance $\frac{2}{n}$ where n is the size of the kernel.

Training parameters. While most parameters were initialized using the method presented in the previous paragraph, biases were initialized to 0. The patch size for the training was set to 128×128 and the batch size to 64. The training was done using the Adam optimizer [10]. The learning rate was set to $1 \cdot 10^{-4}$ and an L_2 weight decay to $1 \cdot 10^{-8}$. The other parameters for the Adam optimizer were left to their default value. On top of the weight decay, there was a decrease of the learning rate of a factor 10 when the validation error stagnated for ten epochs. The model took about two to three weeks to train on a NVIDIA Titan X[2].

3 Color Image Demosaicking Via Deep Residual Learning by Tan et al. [18]

3.1 Architecture

The network starts with a bilinear interpolation of the CFA image into a first-estimate RGB image that then goes through a series of 6 convolutional layers (the three middle layers also have batch-normalization (BN) [7] and a ReLU) with 64 features and 3 × 3 convolutions (convolutions are padded to ensure that the output has the same size as the input). The bilinearly interpolated image is added to the result of these convolutional layers producing a first estimate RGB image. It then goes through a second series of 5 convolutions (the three middle layers also have BN+ReLU) before adding the first estimate to produce the final result. The architecture is depicted in Figure 4.

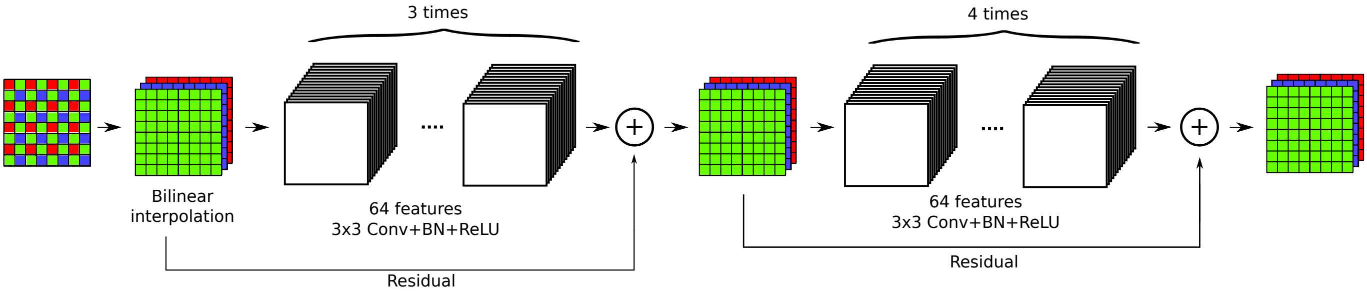

Figure 4: Architecture for demosaicking used by Tan et al. [18].

3.2 Training

Loss. The network can be decomposed into two steps: The first part (corresponding to the first residual section) estimates the green channel and the second estimates the other two channels (using the estimated green channel as a guide). For this reason the loss is split into two terms: the first term focuses only on the green channel and the second one takes into account the final result. For a training pair (x, y), where x is the CFA image and y the corresponding RGB image, and $\theta = (\theta_1, \theta_2)$ the parameters of the network (respectively the first and second part of the network for θ_1 and θ_2), the loss is defined as

$$\mathcal{L}(x, y, \theta = (\theta_1, \theta_2)) = \|y_G - \mathcal{N}_{\theta_1}(x)_G\|_2^2 + \|\mathcal{N}_\theta(x) - y\|_2^2, \tag{1}$$

where the G sub-index corresponds to extracting the green channel and $\mathcal{N}_{\theta_1}$ is the partial network up to the first residual section.

Training dataset. The dataset used for training is the Waterloo dataset presented in [12]. A sample of images is shown in Figure 5. It is made of 4744 images, out of which 100 are used for

[2] This description of the network training is based on the authors' description. We did not attempt to emulate this training. The neural network weights used here therefore are the original ones provided by the authors.

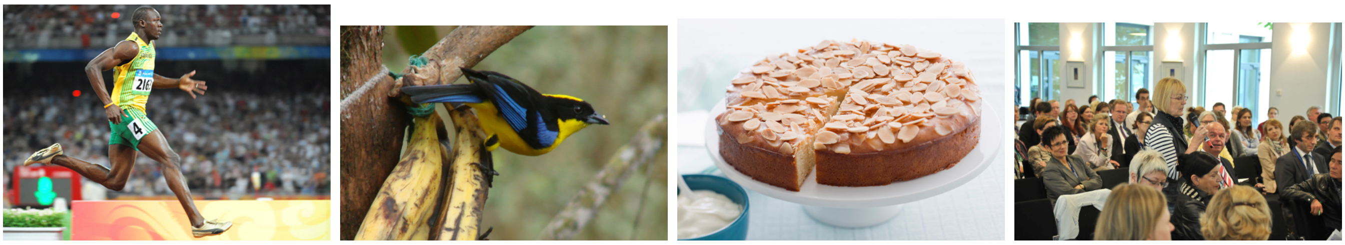

Figure 5: Example of images from the Waterloo dataset [12].

validation. Data augmentation was performed by flipping and rotating the images. CFA images were generated by appropriate masking. Finally, out of all training images, 3000 batches of 128 patches were extracted.

Training parameters. The network was initialized using the same methods than [3] presented in Section 2.2. The patch size for the training was set to 50×50 and the batch size to 128. The training was done using the Adam optimizer [10] with default parameters except for the learning rate. The learning rate was set to $2 \cdot 10^{-4}$ for the beginning of the training and $1 \cdot 10^{-4}$ for the rest of the training. Early stopping of the training was done as soon as the training error stagnated for more than three epochs. The model took about one day to train on a NVIDIA Titan X[3].

4 Quantitative and Qualitative Comparison

In this section we compare both quantitatively and qualitatively some of the most efficient demosaicking methods. We used the state-of-the-art methods LSSC [13], RI [8], ARI [15], MLRI [9] and Getreuer [2] that are compared in most recent papers and retain a low rank in most. Actually, ARI and LSSC are generally considered the best handcrafted methods, but they are also very complex. Table 1 presents the PSNR for all these methods. As one can see the two CNN-based methods outperform all previous demosaicking methods. The PSNRs were computed on the Kodak dataset comprised of 24 images, the McMaster dataset from Zhang et al. [21] comprised of 18 images and a subset of 14 images from the Flickr500 dataset from Syu et al. [17]. We also added the average PSNR on a dataset made of the images from all three datasets. In order to avoid boundary effects, the images were first padded (with a padding of 48 pixels) using symmetry before generating the CFA image. This padding was removed before computing PSNRs. Extra care was taken to make sure that the CFA images used for each method were the same. We also used the parameters recommended by the authors for each method. Table 2 shows that not only do these methods perform well in terms of PSNR but that they are also reasonably fast. The method [3] is written in Python and the methods [18, 13, 8, 9, 15] are written in Matlab. All computation times were computed on an Intel Core i7-7820HQ even for CNN methods. The CNN-based methods were tested without using a GPU. We used the weights provided by the authors for both CNN methods.

Figures 6, 7, 8 and 9 present visual examples of the different demosaicking methods. In particular Figure 7 shows that the method with the least moiré artifacts is the method that was trained specifically to avoid this artifact. Figure 8 shows that both CNN-based methods have no visible zipper effects.

Overall, the two reviewed CNN methods outperform both visually and in terms of PSNR all previous demosaicking methods while also being reasonably fast, even without a GPU.

[3] This description of the network training is based on the authors' description. We did not attempt to emulate this training. The neural network weights used here therefore are the original ones provided by the authors.

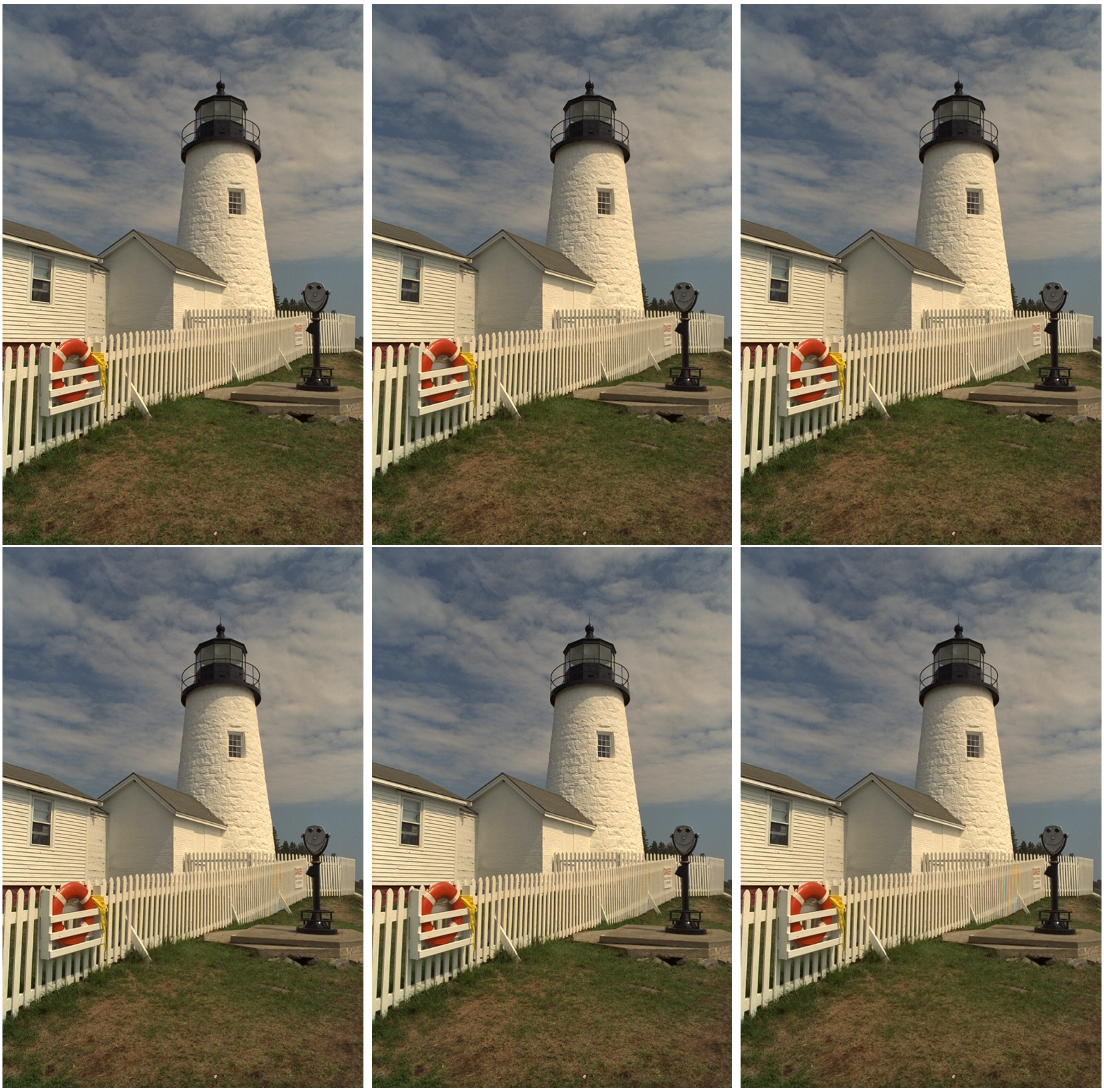

Figure 6: From top-left to bottom-right: Original, Gharbi et al. [3], Tan et al. [18], MLRI [9], ARI [15], Getreuer [2]. All demosaicking methods shown here perform well overall, with no visible strong artifacts. Yet a moiré artifact appears in the fence for some of the methods.

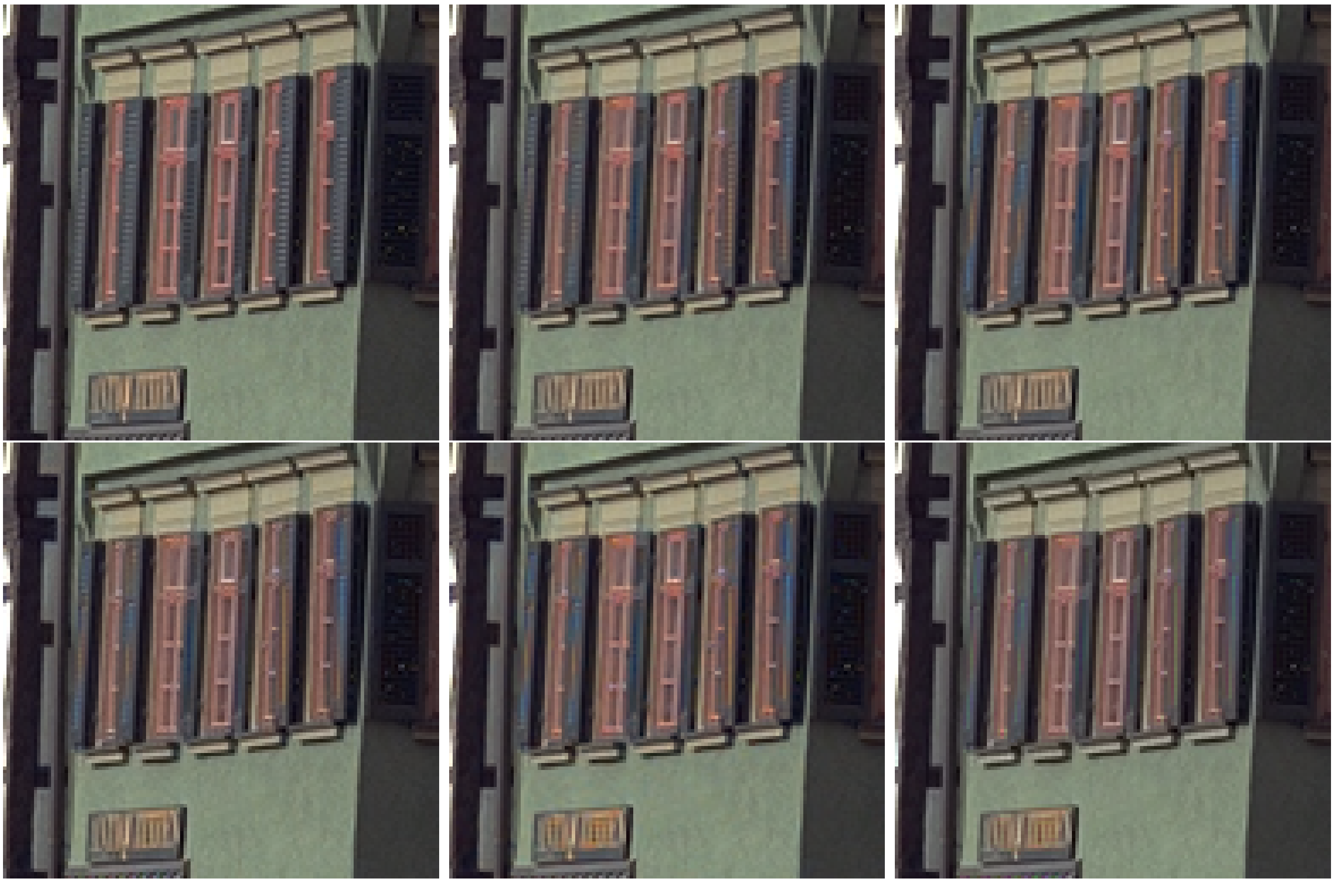

Figure 7: From top-left to bottom-right: Original, Gharbi et al. [3], Tan et al. [18], MLRI [9], ARI [15], Getreuer [2]. All demosaicking methods still present some moiré artifacts on the shutters. However, the Gharbi et al. method, which was trained specifically to avoid this artifact, has fewer than the other methods.

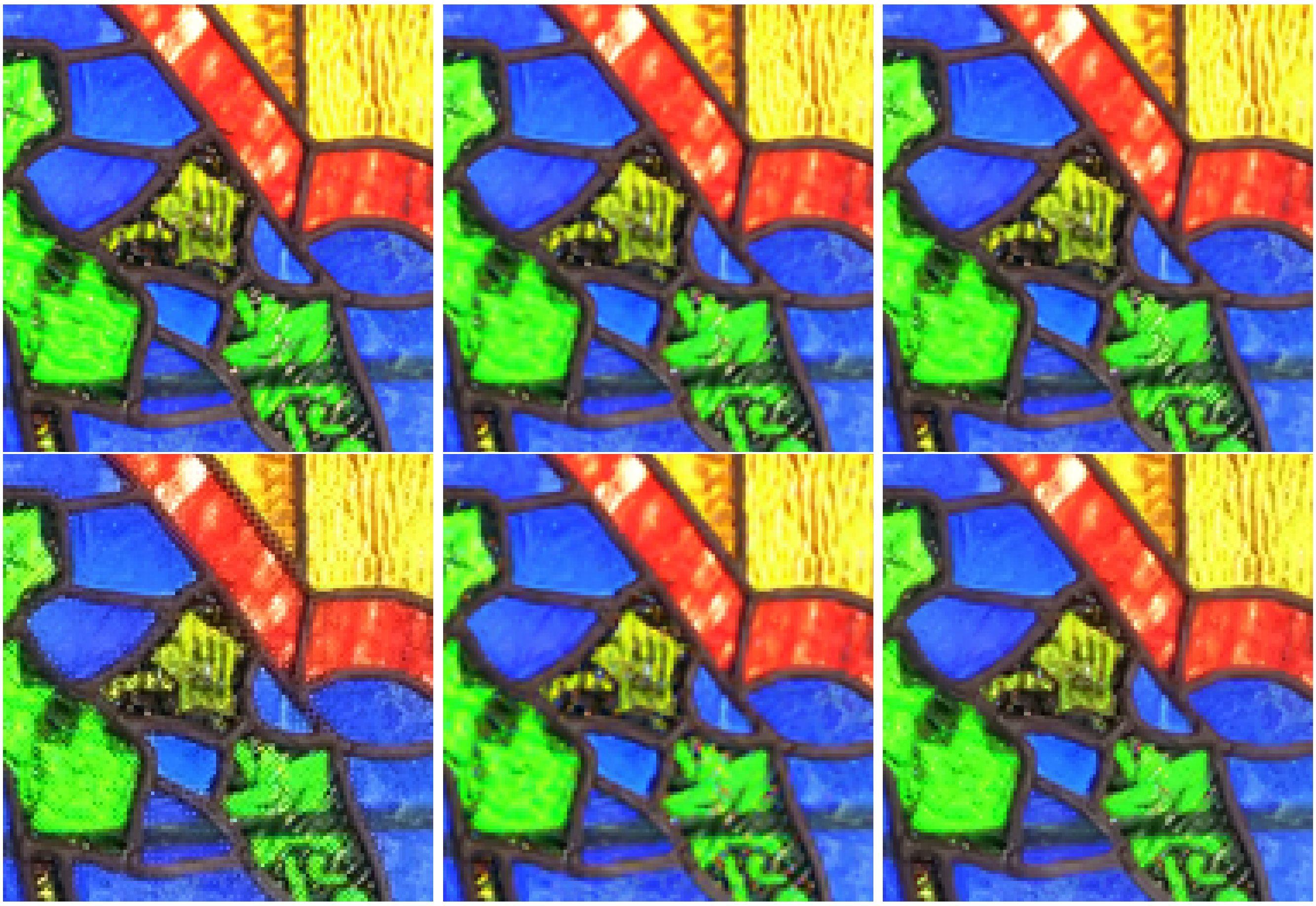

Figure 8: From top-left to bottom-right: Original, Gharbi et al. [3], Tan et al. [18], MLRI [9], ARI [15], Getreuer [2]. Most methods perform well regarding luminance artifacts. Nevertheless the Gharbi et al. method still performs best.

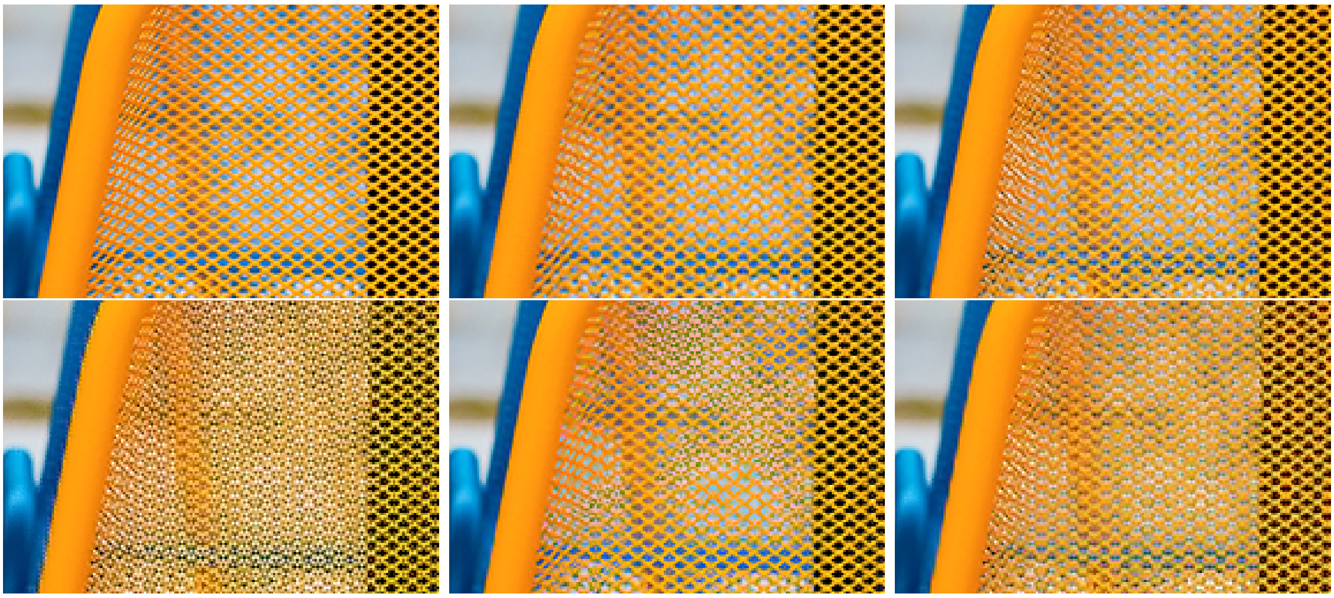

Figure 9: From top-left to bottom-right: Original, Gharbi et al. [3], Tan et al. [18], MLRI [9], ARI [15], Getreuer [2]. Only the Gharbi et al. method performs well visually on really difficult examples.

Method	Kodak	McMaster	Flickr500	All
Gharbi et al. [3]	41.82	**39.08**	**32.22**	38.54
Tan et al. [18]	**41.99**	39.02	32.08	**38.56**
LSSC [13]	41.44	36.21	29.86	36.86
RI [8]	39.77	34.76	29.46	35.58
ARI [15]	39.24	37.48	30.42	36.47
MLRI [9]	39.85	35.29	29.62	35.83
Getreuer [2]	39.30	35.95	29.36	35.74

Table 1: Quantitative comparison of different demosaicking methods. The table shows PSNRs on three datasets: the Kodak dataset, the McMaster dataset [21] and a subset of the Flickr500 dataset [17].

Method	Gharbi et al. [3]	Tan et al. [18]	LSSC [13]	RI [8]	ARI [15]	MLRI [9]	Getreuer [2]
Time	13.97	14.06	736.18	1.86	61.32	1.79	3.07

Table 2: Computation time of the compared methods. Computation time was computed for the lighthouse image of the Kodak dataset and is given in seconds.

4.1 Conclusion

We have seen that the CNN-based demosaicking methods beat by almost two decibels the best human-crafted methods, while being faster by one order of magnitude. To reach this performance, they did not rely on the clever human techniques established by the anterior state of the art, but simply applied rather standard CNN architectures. This success is explainable. The first reason is that human-crafted algorithms rely on the iteration of nonlinear local filters, the most complex one, ARI, having more than 20 such iterations. But deep convolution networks have the same structure and are in addition scalable in depth and number of filters, until they reach the best performance. Furthermore, human-crafted methods have been designed to avoid certain artifacts, probably to the cost of losing in PSNR. CNNs, on the contrary, can be taught to learn intricate casuistry by the variety of presented images, and to keep a memory of complex image statistics. Being trained to reach the best PSNR, they definitely reach that goal. Finally, the Gharbi et al. method succeeded in biasing the learning process so as to avoid completely the most annoying moiré and zipper effects, thus also achieving the most acute requirement of demosaicking methods, while still retaining the best PNSR! This performance arguably seals the destiny of human-crafted methods on this subject.

Acknowledgment

Work partly supported by IDEX Paris-Saclay IDI 2016, ANR-11-IDEX-0003-02, Office of Naval research grant N00014-17-1-2552, DGA Defals challenge n°ANR-16-DEFA-0004-01.

Image Credits

Figures 1, 4 are from the authors. Figure 3 is from the datasets created by Gharbi et al. for [3]. Figure 5 is from the Waterloo dataset [12]. Figures 6 and 7 are from the Kodak dataset. Figure 8 is from the McMaster dataset [21]. Figure 9 is from the Flickr500 dataset [17].

References

[1] J. Deng, W. Dong, R. Socher, L-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848.

[2] P. Getreuer, *Image Demosaicking with Contour Stencils*, Image Processing On Line, 2 (2012), pp. 22–34. https://doi.org/10.5201/ipol.2012.g-dwcs.

[3] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, *Deep joint demosaicking and denoising*, ACM Transactions on Graphics (TOG), 35 (2016), p. 191. https://doi.org/10.1145/2980179.2982399.

[4] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[5] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, in IEEE International Conference on Computer Vision, 2015, pp. 1026–1034. https://doi.org/10.1109/ICCV.2015.123.

[6] M.J. Huiskes and M.S. Lew, *The MIR flickr retrieval evaluation*, in First ACM International Conference on Multimedia Information Retrieval, 2008, pp. 39–43. https://doi.org/10.1145/1460096.1460104.

[7] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in 32nd International Conference on Machine Learning, vol. 37, 2015, pp. 448–456.

[8] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, *Residual interpolation for color image demosaicking*, in IEEE International Conference on Image Processing, 2013, pp. 2304–2308. https://doi.org/10.1109/ICIP.2013.6738475.

[9] ——, *Minimized-Laplacian residual interpolation for color image demosaicking*, in Digital Photography X, vol. 9023, International Society for Optics and Photonics, 2014, p. 90230L. https://doi.org/10.1117/12.2038425.

[10] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

[11] A. Krizhevsky, I. Sutskever, and G.E. Hinton, *Imagenet classification with deep convolutional neural networks*, in 25th International Conference on Neural Information Processing Systems, 2012, pp. 1097–1105.

[12] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, *Waterloo Exploration Database: New challenges for image quality assessment models*, IEEE Transactions on Image Processing, 26 (2017), pp. 1004–1016. https://doi.org/10.1109/TIP.2016.2631888.

[13] J. Mairal, F.R. Bach, J. Ponce, G. Sapiro, and A. Zisserman, *Non-local sparse models for image restoration*, in IEEE International Conference on Computer Vision, vol. 29, 2009, pp. 54–62. https://doi.org/10.1109/ICCV.2009.5459452.

[14] R. Mantiuk, K.J. Kim, A.G. Rempel, and W. Heidrich, *HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions*, in ACM Transactions on graphics (TOG), vol. 30, 2011, p. 40. https://doi.org/10.1145/2010324.1964935.

[15] Y. Monno, D. Kiku, M. Tanaka, and M. Okutomi, *Adaptive residual interpolation for color image demosaicking*, in IEEE International Conference on Image Processing, 2015, pp. 3861–3865. https://doi.org/10.1109/ICIP.2015.7351528.

[16] S. Ren, K. He, R. Girshick, and J. Sun, *Faster R-CNN: Towards real-time object detection with region proposal networks*, in 28th International Conference on Neural Information Processing Systems, 2015, pp. 91–99.

[17] N-S. Syu, Y-S. Chen, and Y-Y. Chuang, *Learning deep convolutional networks for demosaicing.* arXiv preprint arXiv:1802.03769, 2018.

[18] R. Tan, K.i Zhang, W. Zuo, and L. Zhang, *Color image demosaicking via deep residual learning*, in IEEE International Conference Multimedia and Expo (ICME), 2017.

[19] Y-Q. Wang and N. Limare, *A Fast C++ Implementation of Neural Network Backpropagation Training Algorithm: Application to Bayesian Optimal Image Demosaicing*, Image Processing On Line, 5 (2015), pp. 257–266. https://doi.org/10.5201/ipol.2015.137.

[20] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, *Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising*, IEEE Transactions on Image Processing, 26 (2017), pp. 3142–3155. https://doi.org/10.1109/TIP.2017.2662206.

[21] L. Zhang, X. Wu, A. Buades, and X. Li, *Color demosaicking by local directional interpolation and nonlocal adaptive thresholding*, Journal of Electronic imaging, 20 (2011), p. 023016. https://doi.org/10.1117/1.3600632.