A MULTILAYER NEURAL NETWORK FOR IMAGE DEMOSAICKING

Yi-Qing Wang

CMLA, Ecole Normale Supérieure de Cachan, France

ABSTRACT

The recent revival of interest in artificial neural networks has been fueled by their successful applications in various image processing and computer vision tasks. In this work, we make use of the rotational invariance of the natural image patch distribution and propose a 4×4 patch based multilayer neural network for image demosaicking. We show that it does surprisingly well compared to state-of-the-art approaches requiring much larger neighborhoods. An online demo can be found at `http://dev.ipol.im/~yiqing/ipol_demo/neuaick/`.

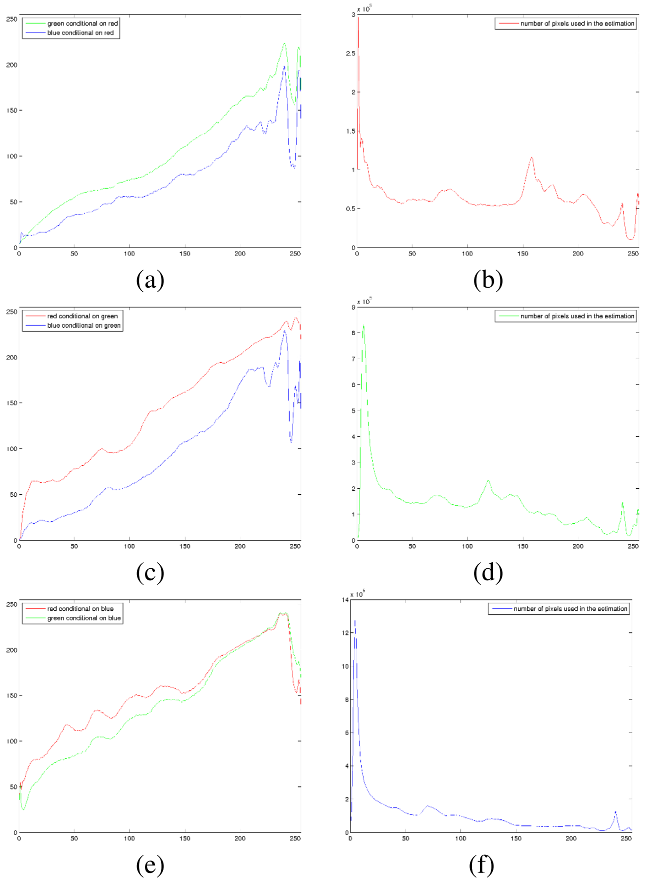
Index Terms— image demosaicking, multilayer neural network, distributional invariance

1. INTRODUCTION

Most digital cameras place a Bayer [1] color filter array (CFA) in front of the sensor to record just one color (R, G or B) at each pixel site. This results in a *mosaiced* image. The process of restoring the missing color information in such an image through cross-channel interpolation is termed *demosaicking*. A huge body of work has been done in this field [2] including two neural network inspired approaches [3, 4]. However, recent successful applications of multilayer neural networks in image processing [5] warrant a revisit of the subject.

A patch based demosaicking algorithm should approximate the expectation of the missing pixels conditional on the visible ones under some patch distribution. For an illustration, we start with pixels, or 1×1 patches, for which conditional expectations are simple to evaluate. For instance, to estimate the green channel given its red counterpart, it suffices to sample a large number of color pixels having the same red intensity and take the average of their green intensities (see Fig.1). Unsurprisingly, despite their accuracy, the obtained conditional expectations do not work well even when applied to an image taken from the same image set (see Fig.2). Because of the quasi-linear relationship in all three cases, here demosaicking tends to make a gray-looking image out of a mosaiced one. In addition, since the local geometry is not taken into account, the so-called zipper effect is pronounced.

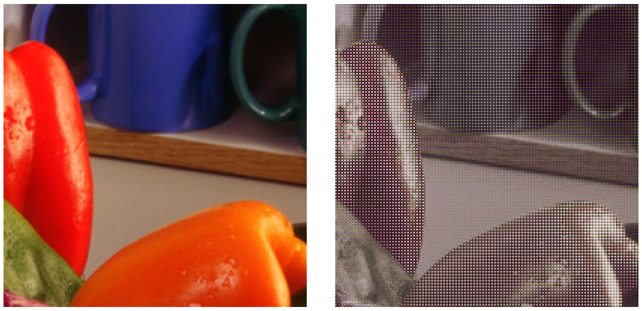
Therefore, the natural extension is to enlarge patches. Its benefit is obvious from the previous discussion, even though computing conditional expectations are no longer as easy, because the same approach quickly becomes intractable due to the curse of dimensionality. Fortunately, studies have shown that neural networks are universal approximators [7] and that deep networks tend to represent signals more efficiently [8]. Owing to the problem's non-convex nature in general, it was not clear as to how best to tap into this potential from a numerical standpoint. However, since 2006, major advances have been made in this direction [8]. Key to training a deep, or multilayer, neural network seems to be an autoencoder-enabled unsupervised learning process.



(a) (b) (c) (d) (e) (f)

Fig. 1. In the *McMaster dataset* [6]: conditional expectations of the other two channels given the (a) red (c) green (e) blue channel and their respective un-normalized prior distributions (b) red (d) green (f) blue.

This work was supported in part by CNES, ERC (AG Twelve Labours), ONR N00014-97-1-0839, DGA and FMHJ.



(b) original (c) demosaiced

Fig. 2. Demosaicking a *McMaster* subimage with pixel-wise conditional expectations computed from the same dataset.

2. TRAINING A DEEP NEURAL NETWORK

First, let us introduce the concept of autoencoder. An autoencoder is a neural network whose fan-in x and fan-out $\widehat{x}$ are vectors of the same dimension, related by a non-linear hidden layer of potentially different size

$$\begin{aligned}\widehat{x} &= Va + h \\ a &= \tanh(Wx + b) \end{aligned} \tag{1}$$

or

$$\begin{aligned}\widehat{x} &= \tanh(Va + h) \\ a &= \tanh(Wx + b) \end{aligned} \tag{2}$$

where $\tanh(\cdot)$ is understood to be applied element-wise. The filtering defined by (W, b) followed by the hidden layer transforms the fan-in x to the activation a, which is supposed to code x in such a way as to make the reconstruction $\widehat{x}$ close to x. Therefore, building an autoencoder can be seen as extracting the most statistically relevant features from the fan-in. Suppose that we have n input signals $(x_j)_{1\le j\le n} \in \mathbb{R}^{d\times n}$ which produce n activations $\mathfrak{a} := (a_j)_{1\le j\le n} \in \mathbb{R}^{\mathfrak{m}d\times n}$, the autoencoder's parameters are determined through minimizing the following objective

$$\frac{1}{nd}\sum_{j=1}^{n}\|\widehat{x}_j - x_j\|_2^2 + \frac{\alpha}{\mathfrak{m}d^2}(\|V\|_2^2 + \|W\|_2^2) + \beta\mathfrak{sp}(\mathfrak{a}, n, d, \rho) \tag{3}$$

with α, β, ρ three prefixed hyper-parameters. The last penalty, intended to induce an over-complete dictionary for sparse representation, is a function of average activation

$$\mathfrak{sp}(a_1, \cdots, a_n, n, d, \rho) = \frac{1}{\mathfrak{m}d}\sum_{k=1}^{\mathfrak{m}d} KL\Big(\rho\Big|\Big|\frac{1}{n}\sum_{j=1}^{n}|a_{jk}|\Big)$$

where

1. $\mathfrak{m}$ is a redundancy control. Together with the activation level $\rho \in (0, 1)$, it suggests that ideally, $\rho\mathfrak{m}d$ hidden nodes per patch are fully activated on average.
2. $KL(\cdot||\cdot)$ is the standard Kullback-Leibler divergence between two Bernoulli probabilities. Regardless of ρ, this divergence strongly penalises average activation near 0 and 1, either because the associated feature is irrelevant or because it has rather low information value. Yet it is still possible for a spurious feature to reach the prescribed activation level if its norm is big enough, hence the regularization term $\|W\|_2^2$.
3. The term $\|V\|_2^2$ is there to prevent over-fitting. Another concern is to prevent an autoencoder from learning an identity, which may come with a small W and a large V to exploit the quasi-linearity of $\tanh(\cdot)$ around zero. Thus the prior on V helps. See [9] for another solution.
4. The features recorded in W ought to stay away from 0 thanks to the sparsity constraint. If there are not enough distinct features to fill all the hidden nodes, almost identical features may result, which indicates a higher than necessary $\mathfrak{m}$ or poorly chosen hyper-parameters.

Turn now to our deep neural network. Given a $\kappa \times \kappa$ mosaiced patch v (hence $d = \kappa^2$), represented as a $\mathbb{R}^{\kappa^2}$-valued column vector, our network with two hidden and one linear output layers will make an educated guess $\widehat{u}$ as to what the missing pixels $u \in \mathbb{R}^{2\kappa^2}$ should look like according to

$$\widehat{u} = W_3[v^t, \tanh(W_2\tanh(W_1 v + b_1) + b_2)^t]^t + b_3.$$

Note that this architecture is meant to let the neural network focus on the non-linear part of the color interpolation.

To determine $(W_l, b_l)_{1\le l\le 3}$, a combination of supervised and unsupervised learning, as advocated in [8], was used. We first drew n random patches from some high quality color images. To each of these patches, the RGGB Bayer CFA was applied to separate the visible pixels v from the invisible ones u. An autoencoder was trained on v and the resultant filter (W, b) henceforth became (W_1, b_1). The activations from the first hidden layer were then computed, on which another autoencoder was trained to set up (W_2, b_2). Finally, the output linear layer was initialized by a linear regression to fit the teaching signals u. All the involved objectives (3) were optimized using L-BFGS [10]. Note that before starting training, it helps to zero-phase whiten v and u (see Algorithm 1).

Algorithm 1 Zero-phase whitening

```
Input: $n$ vectors $p_j \in \mathbb{R}^d$.
Output: the whitening operator $\mathcal{W}$.
Parameter: smoothing factor $\sigma_Z$.
Center the inputs $\tilde{p}_j = (p_j - \bar{p})/255$ with $\bar{p} = \frac{1}{n}\sum_{j=1}^{n} p_j$.
Run the principal component analysis (PCA) on $(\tilde{p}_j)_{1\le j\le n}$ to get their eigenvectors and eigenvalues $(\phi_i, \lambda_i)_{1\le i\le d}$.
$\mathcal{W}(p) = 255^{-1}\sum_{i=1}^{d}\sqrt{\frac{1}{\lambda_i+\sigma_Z}}\langle\phi_i, p - \tilde{p}\rangle\phi_i$ with $\langle\cdot,\cdot\rangle$ a standard scalar product.
```

Once fully configured, the deep network was fine-tuned with the backpropagation algorithm [11] on the same training data for several more rounds. Fresh data was then drawn in to run the stochastic gradient in the hope of further improving the network. In these last two stages, the network's generalization error was assessed on a distinct set of validation examples. Since we could not know in advance the hyper-parameters most conducive to good learning, an exhaustive exploration in the parameter space was carried out.

Trained to restore color removed by the RGGB CFA represented in Fig.3(a) for 2×2 patches, a network should be able to handle those masked by any of the other three CFAs as well because the patch distribution is rotation-invariant. The same argument remains true when extended to $2k\times2k$ patches with $k \geq 1$. In practice, it means that each missing color pixel can be estimated multiple times, thereby enhancing the algorithm's overall performance.

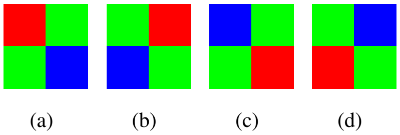

(a) (b) (c) (d)

Fig. 3. Four basic blocks of the Bayer pattern

3. EXPERIMENTS

In our experiments, the patches were 4×4 and whitened using Algorithm 1 with $\sigma_Z = 0.1$ for inputs and $\sigma_Z = 0$ for outputs. The first hidden layer containing 80 nodes was trained 400 rounds using an autoencoder of type (1) with $\alpha = 0.4$, $\beta = 5$, $\rho = 0.1$ on 10^5 examples. The second hidden layer also has 80 nodes, and was trained 400 rounds using an autoencoder of type (2) with hyper-parameters $\alpha = 0.4$, $\beta = 0.1$, $\rho = 0.05$. The final layer followed from

$$(W_3, b_3) = \operatorname*{argmin}_{(W,b)} \|\tilde{u} - W\bar{v} - b\|_2^2 + \lambda\|W\|_2^2$$

where $\bar{v}$ is the concatenation of the whitened input $\tilde{v}$ and their activations from the network's second hidden layer. $\tilde{u}$ is the whitened output. The decay parameter λ was set to 0.005.

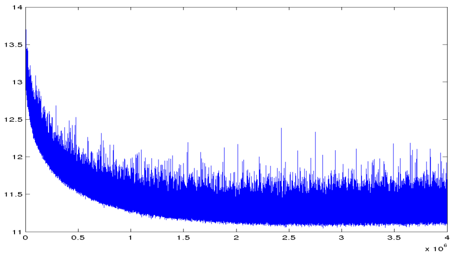
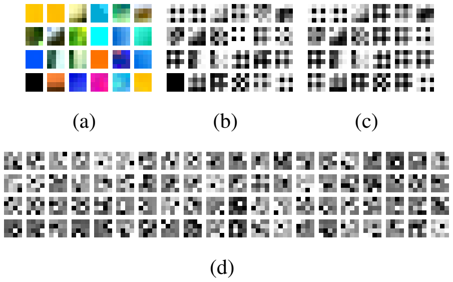


Fig. 4. Evolution of the validation RMSEs during the stochastic gradient descent. Though noisy, this RMSE path clearly demonstrates the value of stochastic gradient descent.

When tracking the neural network's performance (Fig.4) on a validation dataset of 10^4 patches, we ran 4×10^6 rounds of stochastic gradient descent with batch size 100 at a learning rate 0.001 to further drive down the objective

$$\|\tilde{u} - f(\tilde{v}, \theta)\|_2^2 + \lambda\|W_3\|_2^2,$$

where $f(\cdot, \theta)$ is the network defined by its parameter set θ.


(a) (b) (c)

(d)

Fig. 5. Examples of 5(a) color patches 5(b) RGGB CFA filtered patches and 5(c) zero-phase whitened patches used in the training. 5(d) All the 80 learned features stored in the first hidden layer of the trained network.

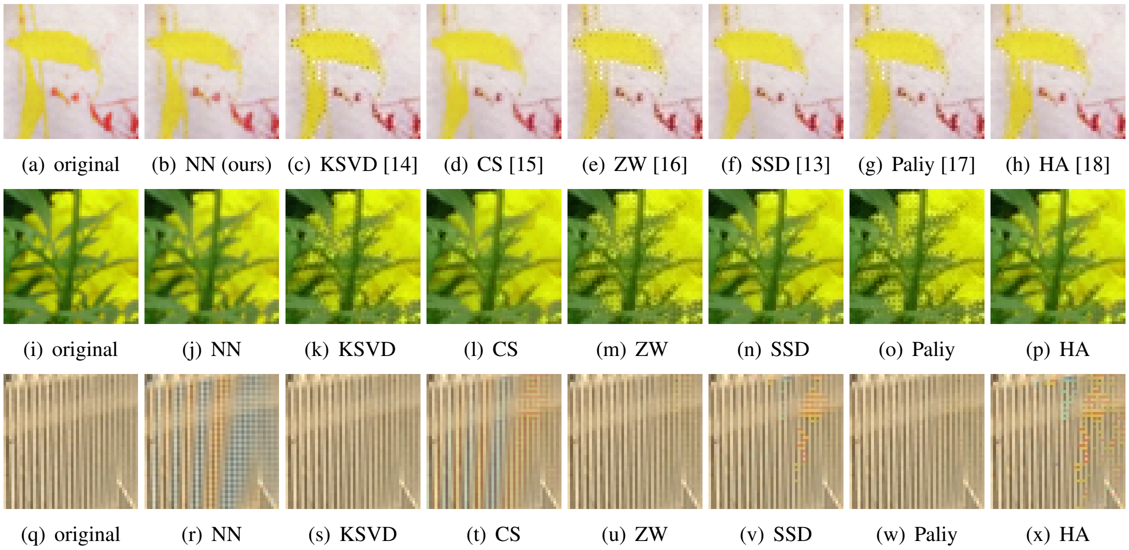
The training and validation examples were collected from two distinct sets, having respectively 2992 and 10 *Flickr* images. Having been demosaicked one way or another, these images were not fed directly to the neural network or it might wind up simply imitating other demosaicking algorithms. To guarantee their quality, we downsampled the images by 2 after convolving them with a Gaussian kernel of standard deviation equal to 1.2 [12].

We tested our neural network against some best performing methods available. The results, stated in both RMSE and the zipper effect as in [13], are reported in Tab.1 for the two datasets (Fig.6). As stated in [14], the last five *Kodak* images were used to tune KSVD. They were excluded for fairness.

Fig. 6. The *Kodak* and *McMaster* dataset

4. DISCUSSION

The experiments show that our tiny neural network compares favorably with these state-of-the-art algorithms. We have deliberately chosen images of vibrant colors to constitute our training set, which explains in part that the resultant network



(a) original (b) NN (ours) (c) KSVD [14] (d) CS [15] (e) ZW [16] (f) SSD [13] (g) Paliy [17] (h) HA [18]

(i) original (j) NN (k) KSVD (l) CS (m) ZW (n) SSD (o) Paliy (p) HA

(q) original (r) NN (s) KSVD (t) CS (u) ZW (v) SSD (w) Paliy (x) HA

Fig. 7. Comparison of the seven algorithms: our neural network does not do well when image patterns to recover oscillate much, in part because the chosen patch size 4×4 may be too small for it to detect high frequency variations. In contrast, it does interpolate well where color changes abruptly. Visually speaking, contour stencil [15] yields the closest numerical results. However, the neural network seems to be more accurate on blobs.

McMaster	**NN**	**KSVD**	**CS**	**SSD**	**HA**	**Paliy**	**ZW**
1	**8.52**	9.67	8.92	10.60	10.45	11.61	11.65
2	**4.66**	4.89	5.03	5.03	5.08	5.36	5.44
3	5.75	**5.54**	5.57	5.81	6.53	5.97	6.03
4	3.70	**3.49**	4.37	3.90	4.53	4.48	5.09
5	**4.84**	5.98	5.37	6.17	6.00	6.82	7.20
6	3.61	**3.36**	3.82	4.33	4.41	5.14	5.65
7	3.83	2.68	**2.50**	3.80	4.37	2.78	2.79
8	3.16	**2.96**	3.26	3.46	3.76	3.23	3.49
9	**3.57**	3.65	3.97	4.11	4.32	4.79	5.17
10	**3.05**	3.13	3.25	3.30	3.50	3.88	4.02
11	2.78	**2.74**	2.97	3.06	3.33	3.61	3.66
12	**2.96**	3.20	3.32	3.33	3.54	3.82	3.89
13	**2.42**	2.51	2.61	2.50	2.65	2.99	3.03
14	**2.96**	3.09	3.28	3.15	3.27	3.76	3.60
15	**2.89**	3.02	3.10	3.15	3.29	3.51	3.63
16	**4.82**	6.32	6.15	7.15	6.94	8.92	8.11
17	**5.44**	6.05	5.55	7.40	7.28	8.83	9.10
18	**4.38**	4.71	5.03	5.16	5.22	5.34	5.34
rmse avg.	**4.07**	4.27	4.33	4.74	4.91	5.26	5.38
zipper avg.	0.34	0.34	**0.30**	0.33	0.38	0.38	0.39

Kodak	**KSVD**	**Paliy**	**ZW**	**NN**	**SSD**	**CS**	**HA**
01	**2.23**	2.44	2.73	4.62	4.70	4.09	5.35
02	**2.19**	2.26	2.42	3.32	3.91	4.32	3.49
03	**1.54**	1.69	1.86	2.57	3.05	3.88	3.04
04	**1.98**	2.34	2.53	3.05	3.10	4.05	3.30
05	**2.64**	3.33	3.25	4.19	4.36	4.72	4.88
06	**2.13**	2.30	2.38	4.24	4.16	4.29	4.78
07	**1.64**	1.82	2.06	2.45	2.94	3.82	2.84
08	**3.33**	3.51	3.72	5.72	5.19	5.14	6.32
09	**1.62**	1.67	1.74	2.37	2.64	3.69	2.75
10	**1.78**	1.87	1.92	2.40	2.75	3.71	2.88
11	**2.16**	2.36	2.43	3.80	3.71	4.11	4.09
12	**1.48**	1.62	1.70	2.48	2.57	3.50	2.58
13	**3.74**	3.95	4.19	7.76	6.48	5.10	8.09
14	**2.81**	3.51	3.65	4.01	4.71	5.01	4.66
15	**2.25**	2.62	2.73	3.17	3.94	4.59	3.97
16	**1.44**	1.58	1.56	2.98	3.57	3.95	3.63
17	**1.95**	2.02	2.02	2.95	2.74	2.20	2.97
18	**3.17**	3.65	3.50	5.20	4.55	4.15	5.20
19	**1.95**	2.20	2.19	3.52	3.02	2.54	3.40
rmse avg.	**2.21**	2.46	2.55	3.72	3.79	4.04	4.11
zipper avg.	**0.22**	0.24	0.25	0.36	0.32	0.23	0.35

Table 1. The results from our neural network (NN), (the *Kodak*-tuned) KSVD [14], CS [15, 19], SSD [13, 20], HA [18], Paliy [17] and ZW [16, 21] on *McMaster* (top) and *Kodak* (bottom). The row-wise best results are in bold. The columns are ordered to reflect the performances in RMSE. We only show the average zipper effects for lack of space.

NN	**HA**	**CS**	**KSVD**	**ZW**	**Paliy**	**SSD**
4×4	5×5	5×5	8×8	9×9	12×12	15×15

Table 2. Comparison of the neighborhood size demanded by the seven algorithms. Note that because of their design, these algorithms do not necessarily use every pixel in a given neighborhood. Nonetheless, the neighborhood size remains a basic indicator of the associated algorithmic complexity.

works better on *McMaster* than on *Kodak*. Fig.7 shows that the network handles abrupt color transitions remarkably well. However, its also reveals the neural network's inability to recover high frequency patterns, which is likely to result from the network's diminutive input size (Tab.2).

Recent evidence [5] suggests that endowed with a higher capacity, a deep network should perform substantially better with larger patches, because more structural information can then be discovered. However, training a larger network incurs a higher computational cost. It took roughly three days on a 8-core Linux machine to train the current network.

As opposed to many noise sensitive algorithms, the neural network approach may be extended to noisy image demosaicking, which is certainly of a bigger practical interest because denoising and demosaicking could then be handled in one single procedure [22, 23].

5. REFERENCES

[1] B. E. Bayer, "Color imaging array," July 20 1976, US Patent 3,971,065.

[2] X. Li, B. Gunturk, and L. Zhang, “Image demosaicking: a systematic survey,” in *Visual Communications and Image Processing*, 2008, vol. 6822, pp. 68221J–68221J.

[3] O. Kapah and H. Z. Hel-Or, “Demosaicking using artificial neural networks,” in *Electronic Imaging*, 2000, pp. 112–120.

[4] J. Go, K. Sohn, and C. Lee, “Interpolation using neural networks for digital still cameras,” *IEEE Transactions on Consumer Electronics*, vol. 46, no. 3, pp. 610–616, 2000.

[5] H. C. Burger, C. J. Schuler, and S. Harmeling, “Image denoising: Can plain neural networks compete with BM3D?,” in *Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2392–2399.

[6] L. Zhang, X. Wu, A. Buades, and X. Li, “Color demosaicking by local directional interpolation and nonlocal adaptive thresholding,” *Journal of Electronic Imaging*, vol. 20, no. 2, pp. 023016–023016, 2011.

[7] A. R. Barron, “Approximation and estimation bounds for artificial neural networks,” *Machine Learning*, vol. 14, no. 1, pp. 115–133, 1994.

[8] Y. Bengio, “Learning deep architectures for AI,” *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[9] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[10] J. Nocedal, “Updating quasi-Newton matrices with limited storage,” *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[11] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, “Efficient backprop,” in *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 1998.

[12] J. M. Morel and G. Yu, “Is SIFT scale invariant?,” *Inverse Problems and Imaging*, vol. 5, no. 1, pp. 115–136, 2011.

[13] A. Buades, B. Coll, J. M. Morel, and C. Sbert, “Self-similarity driven color demosaicking,” *IEEE Transactions on Image Processing*, vol. 18, no. 6, pp. 1192–1202, 2009.

[14] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, “Non-local sparse models for image restoration,” in *International Conference on Computer Vision*. IEEE, 2009, pp. 2272–2279.

[15] P. Getreuer, “Contour stencils: Total variation along curves for adaptive image interpolation,” *SIAM Journal on Imaging Sciences*, vol. 4, no. 3, pp. 954–979, 2011.

[16] L. Zhang and X. Wu, “Color demosaicking via directional linear minimum mean square-error estimation,” *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2167–2178, 2005.

[17] D. Paliy, V. Katkovnik, R. Bilcu, S. Alenius, and K. Egiazarian, “Spatially adaptive color filter array interpolation for noiseless and noisy data,” *International Journal of Imaging Systems and Technology*, vol. 17, no. 3, pp. 105–122, 2007.

[18] J. F. Hamilton Jr and J. E. Adams Jr, “Adaptive color plan interpolation in single sensor color electronic camera,” May 13 1997, US Patent 5,629,734.

[19] P. Getreuer, “Image Demosaicking with Contour Stencils,” *Image Processing On Line*, vol. 2012, 2012.

[20] A. Buades, B. Coll, J. M. Morel, and C. Sbert, “Self-similarity Driven Demosaicking,” *Image Processing On Line*, vol. 2011, 2011.

[21] P. Getreuer, “Zhang-Wu Directional LMMSE Image Demosaicking,” *Image Processing On Line*, vol. 2011, 2011.

[22] K. Hirakawa and T. W. Parks, “Joint demosaicing and denoising,” *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2146–2157, 2006.

[23] P. Chatterjee, N. Joshi, S. B. Kang, and Y. Matsushita, “Noise suppression in low-light images through joint denoising and demosaicing,” in *Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 321–328.