IMAGE DEMOSAICKING VIA CHROMINANCE IMAGES WITH PARALLEL CONVOLUTIONAL NEURAL NETWORKS

Takuro Yamaguchi, Masaaki Ikehara

EEE Dept., Keio Univ., Yokohama, Kanagawa 223-8522, Japan
{yamaguchi,ikehara}@tkhm.elec.keio.ac.jp

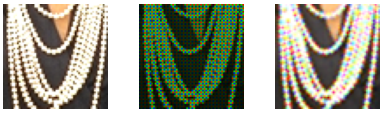
ABSTRACT

Many conventional demosaicking methods are based on hand-crafted filters. However, the filters yield false colors in salient regions like edges and textures. For acquisition of high quality images, we focus on neural networks. Neural networks lead to high accuracy in many fields. However, there are few methods in demosaicking field. For adaptation to demosaicking, we consider not only network's architecture but also the input. In this research, we utilize a Bayer image as input of our networks. However, different filter is needed in estimation at different color pixels, for example, missing red value at green pixel and that at blue pixel. Therefore, we prepare four networks with downsampling operators classified by color patterns in Bayer images. This downsampling operator not only identifies the color pattern but also reduces the calculation cost in each network due to reduction of the size of feature maps. Besides, preparation of multi-networks instead of a deep single-network is suitable for today's parallel computing. Moreover, we utilize not missing color images but chrominance images as output. Compared to results with missing color images as output, the results with chrominance images obtains higher accuracy. Experimental results show our CNN-based approach produces high quality restored images.

Index Terms — Demosaicking, Convolutional Neural Network, Multi-network, Parallel Computing

1. INTRODUCTION

Digital color images are widely used in today's world. The color images used in our daily life are full-color images. Each pixel of full-color images has the three color values: red, green and blue. There are two types of digital cameras to obtain a digital color image. The digital cameras which can take full-color images are three-plate cameras. Three-plate cameras have three color sensers corresponding to the three colors. However, three-plate cameras are expensive and large equipments. Therefore, single-plate cameras are in general use instead. Single-plate cameras have a color filter array (CFA) and the output color images have one color value in each pixel. Hence, restoration of CFA images taken by single-plate cameras to the full-color counterparts is needed and it called demosaicking. Especially, Bayer pattern [1] is one of the most popular CFA and many demosaicking methods are proposed. The simple demosaicking way is image interpolation-based methods [2–6]. Bicubic [2] is a famous interpolation algorithm to upscale grayscale images and it can be applied for demosaicking by processing each color independently. However, its outputs contain gaps in edges among the colors and the gaps cause false colors shown in Fig. 1. The false color is the main problem in demosaicking. To solve the problem, some methods utilize continuity of color difference or guided upsampling [3–6]. These methods obtain much better accuracy than the methods processing each color independently. However, they are based on hand-crafted filters to interpolate and cause some errors in some high frequency regions such as textures.



(a) Full-color (b) Bayer (c) Estimated

Fig. 1. False color in estimated image. (c) is estimated image by Bicubic [2] from (b).

The methods which have better accuracy are learning-based methods [7–10]. Learning-based ways gather a lot of CFA images and the full-color counterparts and train model parameters to make full-color images from CFA images. Learning-based methods can overcome interpolation-based methods in accuracy, however, their accuracy depends on training datasets and their models.

In this paper, we propose a new learning-based method utilizing Convolution Neural networks (CNN). Neural networks gather a lot of attention in many image processing fields and produces better results than conventional methods. For example, SRCNN [11] produce high quality upsampled images in gray scale image upsampling. For adaptation to demosaicking of CNN, our proposal has three contributions. First, we use a Bayer image as input without any interpolation. Second, we prepare networks for four color patterns. Third, we utilize not missing color images but chrominance images as output. These points lead to acquirement of better results compared to conventional demosaicking methods and experimental results show high accuracy of the proposed method. Moreover, our networks are suitable for parallel computing. We adopt downsample operators to identify the color patterns. The size of feature maps in our networks is reduced after the operators. Besides, our network has independent parts which can parallel processing.

2. RELATED WORKS

Compared to super-resolution, CNN-based demosaicking methods is fewer. Especially, most methods use a pre-estimated full-color image by a conventional demosaicking method as input. For example, R. Tan et al. [9] adapt a pre-estimated image by Bilinear as the input. This method is divided into two parts. The first part focus on estimation of green pixels, which are contained the most in Bayer images. The second part is to estimated full-color images by updating the output of the first part. They use residual maps as outputs of the both parts. In [12], D. S. Tan et al. also adapt a pre-estimated image by a conventional demosaicking method as the input and the residual map between the input and the full-color counterpart. Moreover,

(a) R (b) B (c) G1 (d) G2

Fig. 2. Color pattern in Bayer image classified by the centered pixel in a patch

they show results with inputs pre-estimated by different conventional demosaicking methods. This results show the methods used in pre-estimation have great influence for the network's outputs. Gharbi et al. [10] use not any pre-estimation methods but downsampling Bayer image to pick up pixels in each color as the input. Before the last hidden layer, they process upsampling for outputs of the previous layer and add upscaled RGB frames to the input channels.

Hence, we should consider how we select the input for adaptation of the neural network for demosaicking. In our method, we select the Bayer image as it is for input. Although this seems a simple idea, it is difficult for image demosaicking. In Bayer images, neighboring pixels have different colors and the pixel values have no continuity. However, as conventional methods utilize color difference or guided upsampling, the pixels have some relationship. Therefore, we suppose to take its full advantage and decide the Bayer image as it is for input.

3. PROPOSED METHOD

In Bayer images, each pixel has a color value, red or blue or green. Moreover, green pixels are divided into two patterns by the color of neighboring pixels. Namely, we can divide pixels in a Bayer image to the four patterns as shown in Fig. 2. For adaptation of Bayer images as input, we pay attention to the patterns. Although CNN can approximate various transformation, it is a local processing such as local filtering. We think different patterns need different filters. Therefore, we decide to prepare four networks corresponding to the four patterns. For acquirement of higher accuracy, we consider not only the input but also the output. Demosaicking is to estimate the missing two colors at each pixel. However, we find to use chrominance images instead of full-color images as the output leads our network to higher accuracy. Moreover, estimation of each chrominance images also needs different filters. Therefore, we prepare parts corresponding to each chrominance images in our network architecture.

We show the architecture of our network in Fig. 3. Each part of our network architecture is a simple linear one except for adoption of a downsample process. In this section, we first introduce how to calculate missing color values from the chrominance images obtained by our networks. Then, we introduce the our network's architecture.

3.1. Image Demosaicking via Chrominance Images

Chrominance images Cb, Cr are widely used in image compression. In image compression, chrominance images are downsampled to reduce image capacity because our eyes more insensitive to the change of chrominance image than luminance ones. We think this property is useful in demosaicking and estimation errors in chrominance outputs affect less influence than that in a RGB outputs. Therefore, we adopt chrominance images as output of our networks to improve the accuracy. However, the loss of the chrominance images at green pixels estimated by our network is larger than that of at red and blue pixels. Hence, we newly define a chrominance image Cg. First of

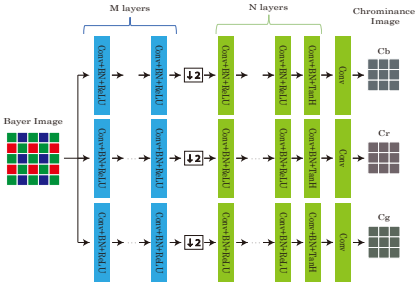


Fig. 3. Outline of our network for each color pattern shown in Fig. 2. We prepare additional three networks which have almost same architecture corresponding to other color patterns. The difference of these networks is where coordinates we remain in downsample.

all, the chrominance images Cb, Cr has the property as follows.

$$\mathrm{R} = \mathrm{Y} + 1.4020\mathrm{Cr}, \quad \mathrm{B} = \mathrm{Y} + 1.7720\mathrm{Cb} \tag{1}$$

We define Cg with same property as follows.

$$\mathrm{G} = \mathrm{Y} + a\mathrm{Cg} = 0.2990\mathrm{R} + 0.5870\mathrm{G} + 0.1140\mathrm{B} + a\mathrm{Cg} \tag{2}$$

$$\mathrm{Cg} = b\mathrm{R} + 0.5\mathrm{G} + c\mathrm{B} \tag{3}$$

By (2) and (3), we obtain the following identical equation.

$$\mathrm{G} = (0.2990 + ab)\mathrm{R} + (0.5 + a)\mathrm{G} + (0.1140 + c)\mathrm{B} \tag{4}$$

By the identical equation, we get (a, b, c) and

$$\mathrm{Cg} = -0.3620\mathrm{R} + 0.5\mathrm{G} - 0.1380\mathrm{B} \tag{5}$$

Hence, luminance image Y, and chrominance images Cb, Cr and Cg are calculated from their counterpart of a full-color RGB image as follows.

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{Cb} \\ \mathrm{Cr} \\ \mathrm{Cg} \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \\ -0.3620 & 0.5 & -0.1380 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix}. \tag{6}$$

For image demosaicking, we calculate missing color values in each pixel by estimated chrominance images. When we calculate X value at Y pixel (X and Y are R(red), G(green) or B(blue)), we utilize the chrominance values Cx and Cy. For example, we calculate the missing G value $\mathrm{G_R}$ at a R pixel by Cr and Cg. Namely,

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{Cr} \\ \mathrm{Cg} \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ 0.5 & -0.4187 & -0.0813 \\ -0.3620 & 0.5 & -0.1380 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G_R} \\ \mathrm{B} \end{bmatrix}. \tag{7}$$

By solving the simultaneous equation, we get the following equations.

$$\mathrm{G_R} = \mathrm{R} + 0.8260\mathrm{Cg} - 1.4020\mathrm{Cr} \tag{8}$$

In the same way, we estimate the missing values as follows. .

$$\mathrm{B_R} = \mathrm{R} + 1.7720\mathrm{Cb} - 1.4020\mathrm{Cr} \tag{9}$$

$$\mathrm{R_G} = \mathrm{G} + 1.4020\mathrm{Cr} - 0.8260\mathrm{Cg} \tag{10}$$

$$\mathrm{B_G} = \mathrm{G} + 1.7720\mathrm{Cb} - 0.8260\mathrm{Cg} \tag{11}$$

$$\mathrm{R_B} = \mathrm{B} + 1.4020\mathrm{Cr} - 1.7720\mathrm{Cb} \tag{12}$$

$$\mathrm{G_B} = \mathrm{B} + 0.8260\mathrm{Cg} - 1.7720\mathrm{Cb} \tag{13}$$

where $\mathrm{X_Y}$ is the missing X value at a Y pixel.

3.2. Network Architecture

Our network is composed three independent parts to estimate Cb,Cr and Cg. Each part has $(M+N+1)$ convolutional layers and one downsampling operator. Every convolutional layer except the final layer is composed of convolution operator with an activation function and batch normalization operator as follows.

$$L_t(c) = f\left(g\left(b_t(c) + \sum_{c'=1}^{m_{t-1}} w_t(c,c') * L_{t-1}(c')\right)\right) \tag{14}$$

where $L_t(c)$ is the c-th channel in the t-th layer, $w_t(c,c')$ and $b_t(c)$ is its two-dimensional convolution filter and scalar bias, $f(\cdot)$ is the activation function, $g(\cdot)$ is the batch normalization operator, and m_{t-1} is the number of channel in the $(t-1)$-th layer. In the former M layers, we process this operation with 5×5 sized filters for every pixel in the input image with ReLU operator $f(\cdot) = \max(0, \cdot)$ as the activation function. In the latter N layers, we process this operation with 3×3 sized filters for only the coordinates of pixels in a color pattern. The activation function of the N layers but the last is ReLU operator and that of the last is hyperbolic tangent function. The connection of the M layers and the N layers is the following downsample operator that downsamples each feature map by a factor of 2 in vertical and horizontal dimensions.

$$\boldsymbol{L}'_M(i,j) = \boldsymbol{L}_M(2i - x_n, 2j - y_n) \tag{15}$$

where $\boldsymbol{L}'_M$ is the output of this operator and it is also the input of the $M+1$ layer and $(x_n, y_n) \in \{(0,0),(1,0),(0,1),(1,1)\}$ select each pattern shown in Fig. 2. This process picks up the coordinates of pixels in a color pattern. After the $M+N$ layers, we obtain the final output $\boldsymbol{y}$ by the following convolutional operation with 5×5 sized filters.

$$\boldsymbol{y} = \left(b + \sum_{c'=1}^{m_{M+N}} w_{c'} * L_{M+N}(c')\right) \tag{16}$$

where b is the scalar bias for the final layer and $w_{c'}$ is a two-dimensional convolution filter. In our network, the output $\boldsymbol{y}$ corresponds to Cb,Cr or Cg.

Finally, we prepare four networks corresponding to the color patterns. These networks have the same architecture expressed in this section but different parameters. These four networks are independent and possible to parallel processing. Moreover, our network has the independent parts to estimate Cb,Cr and Cg, which is also possible to parallel processing. Therefore, our method is very suitable for parallel computing.

3.3. Training Procedure

We express the learnable parameters of each part to estimate Cb, Cr and Cg image as $\boldsymbol{\Theta}_n^{\mathrm{Cb}}, \boldsymbol{\Theta}_n^{\mathrm{Cr}}, \boldsymbol{\Theta}_n^{\mathrm{Cg}}$ in the i-th pattern of the centered pixels in Fig. 2, respectively. These parameters are set as follows.

$$\boldsymbol{\Theta}_n^{\mathrm{Cb}} = \underset{\boldsymbol{\Theta}_{\mathrm{Cb}}}{\operatorname{argmin}} \left\| \boldsymbol{y}_n^{\mathrm{Cb}} - \mathcal{F}_n(\boldsymbol{x}_i; \Theta_{\mathrm{Cb}}) \right\| \tag{17}$$

$$\boldsymbol{\Theta}_n^{\mathrm{Cr}} = \underset{\boldsymbol{\Theta}_{\mathrm{Cr}}}{\operatorname{argmin}} \left\| \boldsymbol{y}_n^{\mathrm{Cr}} - \mathcal{F}_n(\boldsymbol{x}_i; \Theta_{\mathrm{Cr}}) \right\| \tag{18}$$

$$\boldsymbol{\Theta}_n^{\mathrm{Cg}} = \underset{\boldsymbol{\Theta}_{\mathrm{Cg}}}{\operatorname{argmin}} \left\| \boldsymbol{y}_n^{\mathrm{Cg}} - \mathcal{F}_n(\boldsymbol{x}_i; \Theta_{\mathrm{Cg}}) \right\|. \tag{19}$$

where $\mathcal{F}_n(\cdot)$ is the network for the centered pixel pattern. We represent $\boldsymbol{y}^{\mathrm{Cb}}, \boldsymbol{y}^{\mathrm{Cr}}, \boldsymbol{y}_i^{\mathrm{Cg}}$ are the Cb,Cr and Cg image transformed from the full-color version of the input bayer image $\boldsymbol{x}$, respectively. $\boldsymbol{y}_n^{\mathrm{Cb}}, \boldsymbol{y}_n^{\mathrm{Cr}}, \boldsymbol{y}_n^{\mathrm{Cg}}$ is a downsampled one to pick up the centered pixel pattern in the same manner as (15). Namely,

$$\boldsymbol{y}_n^{\mathrm{Cb}}(i,j) = \boldsymbol{y}^{\mathrm{Cb}}(2i - x_n, 2j - y_n) \tag{20}$$

$$\boldsymbol{y}_n^{\mathrm{Cr}}(i,j) = \boldsymbol{y}^{\mathrm{Cr}}(2i - x_n, 2j - y_n) \tag{21}$$

$$\boldsymbol{y}_n^{\mathrm{Cg}}(i,j) = \boldsymbol{y}^{\mathrm{Cg}}(2i - x_n, 2j - y_n) \tag{22}$$

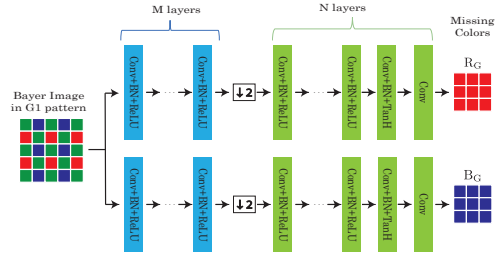


Fig. 4. Outline of our network for G1 color pattern shown in Fig. 2 in "Method 1". We prepare additional three networks which have almost same architecture for the other patterns. The difference of "Method 1" and "Method 2" shown in Fig. 3 is only the outputs.

Fig. 5. 24 images sized 768×512 in Kodak database.

4. EXPERIMENTAL RESULTS

To assess our algorithm, we compare our method with some conventional methods. In this test, there are eight comparative methods [2–6, 8, 10]. Bicubic [2] is a famous image interpolation algorithm for gray scale images and we process it for each color space independently. GBTF [3] utilizes color differences between R and G or B and G. MLRI [4],FDRI [5],ARI [6] utilize residual interpolation. DDR and FR are proposed in [8]. These are learning-based methods. As an example of neural network based methods, we compare Gharbi's method [10]. Learning based methods are depended on the training datasets. Fortunately, we get authors' weights and codes for these learning based methods. Moreover, to assess the learning by chrominance images, we show two proposal methods as "Method 1" and "Method 2". The difference of these methods are only the output. Method 1 uses missing color images as output and we show the outline in Fig. 4. Method 2 is the proposed method mentioned in this paper.

As testing datasets, we use 24 color images sized 768×512 in Kodak datasets shown in Fig. 5 and 18 color images sized 500×500 in McMaster datasets shown in Fig. 6. As training datasets of our networks, we use the Waterloo Exploration database (WED) [13] and the Food-101 datasets [14]. The WED has 4744 images and we utilize them. The Food-101 datasets has 101 food categories and we pick up 50 images from each categories randomly. Eventually, we use 10244 color images in training. In this test, we use same hyperparameters for both proposed methods. We set the number of hidden layers of the former layers of our network M to 2 and that of the latter layers N to 10 and the number of each channel m_t set to

Table 1: Comparison on restoration results in PSNR[dB]. **BOLD** is the highest result of the eight methods.

PSNR	Kodak				McM				Kodak+McM			
	R	G	B	CPSNR	R	G	B	CPSNR	R	G	B	CPSNR
Bicubic [2]	29.33	33.50	29.23	30.28	32.40	36.21	31.60	32.90	30.65	34.66	30.24	31.40
GBTF [3]	39.68	43.34	40.01	40.62	33.98	37.34	33.07	34.38	37.24	40.77	37.04	37.95
MLRI [4]	38.62	41.18	38.48	39.21	36.72	40.23	35.59	36.91	37.80	40.77	37.24	38.23
FDRI [5]	37.31	39.12	37.15	37.74	36.92	40.17	35.54	36.99	37.14	39.57	36.46	37.42
ARI [6]	39.35	42.43	39.14	40.00	37.47	40.68	36.22	37.61	38.54	41.68	37.89	38.97
DDR [8]	40.18	43.93	40.39	41.10	37.12	40.35	35.64	37.17	38.87	42.39	38.35	39.42
FR [8]	40.19	43.85	40.34	41.07	37.50	41.01	35.82	37.49	39.04	42.63	38.41	39.54
Gharbi's [10]	40.09	42.98	39.47	40.56	37.72	40.48	**36.54**	37.88	39.07	41.91	38.22	39.41
Method 1	39.79	44.25	39.87	40.69	37.88	41.19	36.29	37.90	38.97	42.94	38.34	39.50
Method 2	**41.29**	**44.79**	**40.73**	**41.88**	**38.53**	**41.30**	36.40	**38.18**	**40.11**	**43.30**	**38.88**	**40.30**

128 at the former and 100 at the latter. Namely, the number of layers in the whole network is 13. We use Adam solver [15] to optimize the network parameters and set the number of iteration to 200,000.

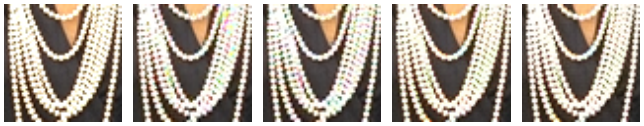
We show the average PSNR in every RGB image and CPSNR in Table 1. Compared to conventional methods, Method 2 has the highest PSNR in almost all colors and CPSNR in both datasets. On the other hand, Method 1 has equivalent PSNR to conventional learning-based methods. This is the effect of chrominance learning. We show some restoration outputs in Fig. 7 and Fig. 8 as comparison with conventional methods. In Fig. 7, there is a texture region. In this region, conventional methods cause false colors but our methods can reduce them. In Fig. 8, there is an edge and some high frequency components. In this region, conventional methods cause not only false colors but also blurs. Our methods keep the high frequency components without false colors. In subjective evaluation, we cannot find some difference between the outputs of Method 1 and Method 2. In other words, using our CNN networks for image demosaicking can suppress false colors. Moreover, chrominance learning make natural color values and Method 2 has more PSNR than Method 1. Compared with Gharbi's method, Method 2 improves at least 1dB from [9] in every color and CPSNR with Kodak dataset. As the comparison with Gharbi's method, we show results for two images in Fig. 9 and Fig. 10. Fig. 9 is prone to moire. In this area, both [10] and our methods can obtain restored image without moire. Fig. 10 has visual difference between [10] and our method. This is the effect for adoption of Bayer images without any interpolation and preparation for every color pattern.

5. CONCLUSION

In this paper, we propose a CNN-based image demosaicking method. In this research, we consider not only architecture but also the input and the output. There are three contributions in the new algorithm. First, we use the Bayer image as input without any interpolation. Second, we prepare networks corresponding to four color patterns. Third, we use chrominance images as output and calculate missing R,G or B values by known color value and the estimated chrominance images. By these points, our method obtains much higher quality demosaicking compared to conventional methods. Compared with another neural network based method, our method can higher accuracy and restore natural images. Moreover, the downsampling operators to pick up the pixels in a color pattern reduce the size of feature maps. It leads to reduction of the calculation cost in each network. These networks are independent and each network has independent parts which is also possible to parallel processing. Hence, our method has not only high accuracy but also high suitability for parallel computing.

This algorithm is useful for adaptation for demosaicking of CNN.

Fig. 6. 18 images sized 500×500 in McMaster database.



(a) original (b) ARI [6] (c) FR [8] (d) Method 1 (e) Method 2

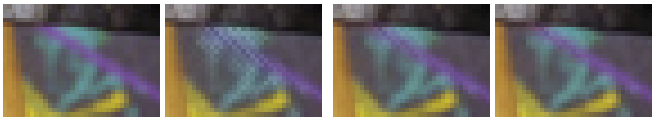
Fig. 7. The parts of result images with conventional methods from an image in the Kodak database.

(a) original (b) ARI [6] (c) FR [8] (d) Method 1 (e) Method 2

Fig. 8. The parts of result images with conventional methods from an image in the McMaster database.

(a) Original (b) Gharbi [10] (c) Method 1 (d) Method 2

Fig. 9. Comparison of our methods and the neural network based method [10] with an image in the Kodak database.



(a) Original (b) Gharbi [10] (c) Method 1 (d) Method 2

Fig. 10. Comparison of our methods and the neural network based method [10] with an image in the McMaster database.

6. REFERENCES

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

[2] R. Keys, "Cubic convolution interpolation for digital image processing," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 29, no. 6, pp. 1153–1160, Dec 1981.

[3] I. Pekkucuksen and Y. Altunbasak, "Gradient based threshold free color filter array interpolation," in *2010 IEEE International Conference on Image Processing*, Sept 2010, pp. 137–140.

[4] Daisuke Kiku, Yusuke Monno, Masayuki Tanaka, and Masatoshi Okutomi, "Minimized-laplacian residual interpolation for color image demosaicking," in *Digital Photography X*. International Society for Optics and Photonics, 2014, vol. 9023, p. 90230L.

[5] Y. Kim and J. Jeong, "Four-direction residual interpolation for demosaicking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 5, pp. 881–890, May 2016.

[6] Yusuke Monno, Daisuke Kiku, Masayuki Tanaka, and Masatoshi Okutomi, "Adaptive residual interpolation for color and multispectral image demosaicking," *Sensors*, vol. 17, no. 12, 2017.

[7] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *2009 IEEE 12th International Conference on Computer Vision*, Sept 2009, pp. 2272–2279.

[8] J. Wu, R. Timofte, and L. Van Gool, "Demosaicing based on directional difference regression and efficient regression priors," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3862–3874, Aug 2016.

[9] Runjie Tan, Kai Zhang, Wangmeng Zuo, and Lei Zhang, "Color image demosaicking via deep residual learning," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2017, pp. 793–798.

[10] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand, "Deep joint demosaicking and denoising," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 191, 2016.

[11] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Learning a deep convolutional network for image super-resolution," in *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Eds., Cham, 2014, pp. 184–199, Springer International Publishing.

[12] D. S. Tan, W. Chen, and K. Hua, "Deepdemosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2408–2419, May 2018.

[13] Kede Ma, Zhengfang Duanmu, Qingbo Wu, Zhou Wang, Hongwei Yong, Hongliang Li, and Lei Zhang, "Waterloo Exploration Database: New challenges for image quality assessment models," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 1004–1016, Feb. 2017.

[14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[15] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.