Image demosaicing using Deep Image Prior

Peter Balušík
Dept. of Telecommunications, FEEC, Brno University of Technology
Brno, Czech Republic
230531@vut.cz

***Abstract*—The paper focuses on the problem of image demosaicing using the deep image prior. The deep image prior (DIP) is an uncommon concept that uses a generative neural network which, however, utilizes only the degraded image as the input for training. A novel method for image demosaicing is proposed, based on DIP, and it is compared with common demosaicing methods. In terms of the objective PSNR and SSIM values, the proposed method proved to be comparable with a widely used Malvar's demosaicing method. Nevertheless, subjectively, DIP produces demosaiced images comparable with the superior Menon's algorithm. Unfortunately, the proposed method turned out to be computationally immensely challenging.**

***Index Terms*—demosaicing, debayerization, color filter array, deep image prior**

I. Introduction

At present, most people own a mobile phone with a camera. Virtually everyone can say they have taken a photo at least once in their life. However, only few people understand how a digital color photo is acquired. To put it briefly, a photo comes into existence by light passing through the lens and projecting onto a flat image sensor with a *color filter*. After the image sensor has captured the different color components, *demosaicing* must begin. Demosaicing is a process that is usually built into the camera. It generates an image that can be viewed after taking a photo. If demosaicing was not present, the image would be obtained in the RAW format. A RAW image contains only the unprocessed information captured by the camera sensor.

The most common color filter used is the Bayer filter [1]. It is a color filter array (CFA) designed in such a way that each of the array cells allows solely the red, green, or blue (RGB) light component to pass. A usual configuration of such a filter can be seen in Fig. 1. It is made using a square pattern with two green elements, one blue and one red element. This pattern is then repeated to cover the whole sensor area. A commonly used arrangement is RG–GB (Fig. 1), but arrangements such as BG–GR, GR–BG, and GB–RG are also possible.

Naturally, a CFA can capture only one color value at each component. Demosaicing can be used to obtain full RGB values of each component. Therefore, demosaicing can be understood as a process of estimating the missing color values of a CFA. In other words, it is a special type of color interpolation. Many different demosaicing methods exist. One of the simplest and quickest methods is the bilinear interpolation. Other, more effective methods include more complex techniques such as Malvar's [2] and Menon's demosaicing methods [3].

This work was supported project No 23-07294S of the Czech Science Foundation (GAČR).

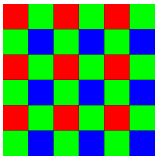

Fig. 1. Schema of Bayer filter with RG–GB arrangement.

Malvar's method uses the bilinear interpolation but corrects its result using the value of the gradient multiplied by a gain factor. The gradient is calculated using predefined filters which are described in [2]. The gradient correction results in sharper images. Additionally, it slightly reduces aliasing artifacts originating due to using the bilinear interpolation.

Menon's approach slightly differs from those mentioned above. It uses a directional interpolation of the green pixels along both horizontal and vertical directions and creates two corresponding "green" images. Following the interpolation, a choice is made between the two green images using two classifiers [3]. Subsequently, the red and blue components are interpolated. Besides information from the Bayer filter, the obtained green image and the two classifiers help with the reconstruction of these components. This results in more accurately reconstructed color channels with significantly less pronounced artifacts.

Deep neural networks (DNN) have also been exploited in demosaicing. Klatzer et al. [4] created a DNN that was able to learn its weights on the training image dataset such that the network performed joint demosaicing and denoising. Kurniawan et al. [5] utilizes the deep image prior (DIP) concept [6] as the present paper does, but it does so for random CFAs containing white elements in addition to RGB.

This paper focuses on an approach to demosaicing related to [5], but using the DIP idea in a much more straightforward way. We adopt the DIP network to solve the image demosaicing problem for its use with standard RGB CFA arrays, and perform comparisons of the restoration quality against the other standard methods.

II. Deep Image Prior

The Deep image prior [6] is a concept used for the reconstruction of degraded images. It restores the images using a generative convolutional neural network (CNN). The main difference from common reconstructive algorithms is that

DIP uses a network that is completely *untrained*. Given the structure of the network, its only input information is an image damaged or degraded in some way. The task of the network is to recover the image as well as possible. All of the weights of the network are randomly initialized. The weights of the network, serving as a parameterization of an image, are trained such that the network learns a conversion of a completely random initial image into the degraded image provided. The size of the network is usually great enough to allow such an unusual transformation after a sufficient number of learning epochs. The key observation of [6] is that along that learning path, a restored image appears. It is only necessary to *stop learning* at the right point. This way, DIP proves that the architecture of the neural network fundamentally influences the results (and needs to be taken into consideration in the future of deep learning). The DIP applied to image processing tasks typically utilizes the U-net-like "hourglass" architecture, also called encoder–decoder with skip connections.

Several applications of this neural network are described in [6]. These applications include denoising, super-resolution, inpainting, image enhancement, removal of JPEG-compressed artifacts. In some applications, the visual quality of the restored images is comparable with the state-of-the-art methods, which are *trained* in most cases. The downside of this concept is that the optimization process is very slow, compared with the other networks which—once learned—are fast.

A deep neural network used for image generation can be considered a parametric function $x = f_\theta(z)$ that maps a code vector z to an image x. In the case of DIP, the code vector z is randomly initialized but fixed in the course of the learning epochs. The parameter θ incorporates all the weights and biases present in the network.

Reconstruction tasks can be formulated as energy minimization problems:

$$x^* = \min_x E(x; x_0) + R(x), \tag{1}$$

where $E(x; x_0)$ is the data term that is chosen according to the required application, x_0 represents the damaged image and $R(x)$ is an explicit regularizer. The regularizer is not tied to particular applications; commonly it characterizes naturally looking images. But in the case of DIP, R is replaced by an implicit prior captured by the neural network itself:

$$\theta^* = \operatorname*{argmin}_\theta E(f_\theta(z); x_0), \tag{2}$$

where the minimizer θ^* can be found using standard optimization techniques. Once θ^* is obtained, the final recovered image can be reconstructed by the application of the network to the initial code, i.e. $x^* = f_{\theta^*}(z)$. As already mentioned, the ability of the network to (over)fit and thus result in the corrupted image itself (i.e. $x_0 = x^*$) requires stopping the training process after a proper number of epochs [6].

III. Method

As mentioned above, one of the applications of DIP in [6] is inpainting. Inpainting is a process where damaged or missing portions of a digital image are repaired (repainted). The authors of [6] propose the following approach to solve this problem. Let x_0 be the depleted image of size $H \times W$, and $m \in \{0, 1\}^{H \times W}$ the binary mask indicating missing pixels. Then the image can be restored using the following data term (after parameterization of the network):

$$E(f_\theta(z); x_0) = \|(f_\theta(z) - x_0) \odot m\|^2, \tag{3}$$

where $\odot$ is the Hadamard product. The inclusion of the data prior is crucial, given that the energy (data) term alone would not be sufficient to spread information about the color values to the missing sections. If the data prior was not present, the resulting image would simply not change after optimization over pixel values x. Again, DIP uses an implicit data prior that is given by the optimization of (3) with respect to parameters θ.

In this paper, an approach similar to inpainting is proposed for demosaicing. Instead of the binary mask that takes on the values 0 or 1, a Bayer mask involving RGB channels is used. This paper works with an RG–GB Bayer mask but other combinations would result only in the change of the mask. If the binary mask m is replaced by a Bayer mask M_{CFA}, the data term for demosaicing becomes

$$E(f_\theta(z); x_0) = \|(f_\theta(z) - x_0) \odot M_{\text{CFA}}\|^2. \tag{4}$$

Again, the data prior is present through the optimization of a data term similar to (3).

IV. Experiments and Results

A. Data set

All 24 images from the traditional Kodak image data set [7] were used in the experiments. The size of these images was 768×512 or 512×768, depending on the image orientation. The images come in the uncompressed PNG-24 file format with 8 bits per channel.

B. Artificial mosaicing

Real "RAW" images, in practice acquired by the camera sensor, had to be simulated. Therefore, an artificial bayerization, or mosaicing, was performed on the Kodak images. An RG–GB Bayer mask was used to create R, G, B undersampled images.

C. Parameters of the neural network

An encoder–decoder architecture similar to the U-net [8] with five downsampling and five upsampling convolutional layers was chosen for the demosaicing method. Even though skip connections led to a faster optimization, they also brought undesirable artifacts, at least with the parameter settings used in the experiments. The learning rate used was 0.001 and the network weights were randomly initialized with Gaussian noise. The counts of feature maps in both the upsampling and downsampling convolution layers were as follows: 64, 64, 128, 256, and 512. Compared with the inpainting experiment presented in [6], the number of feature maps was the most distinctive setting of the proposed network. The feature maps were obtained by a 3×3 convolution with zero padding,

followed by a LeakyReLU activation function. The Adam optimizer [9] was used as the optimization algorithm. The number of epochs was chosen to be 2000 since the highest PSNR values most frequently appeared around 2000 epochs. Also, the maximum SSIM values were observed close to 2000 epochs.

D. Evaluation

The proposed demosaicing method was compared with the bilinear interpolation, to Malvar's and Menon's methods, and to the `demosaic` function in Matlab (very similar to Malvar's demosaicing method). As the objective scores for the comparison, the peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM) [10] were used. It is worth noting that the higher the PSNR value, the better. The maximal SSIM value is 1. The PSNR and SSIM values were obtained using the `scikit-image` [11] Python package.

It is worth noting that the classic demosaicing methods mentioned above are purely deterministic, while randomness of several kinds affects the output of the DIP-based method: First, the code vector z is newly generated each time the optimization is run. Second, the network is initialized with random weights θ, meaning that every time the program runs, a different local minimum of the loss function can be found. Finally, the optimization algorithm utilizes the stochastic gradient descent [9], which may also lead to a perturbation in the final solution. As a consequence, the proposed method is not perfectly stable in terms of the PSNR/SSIM over multiple runs, since the results vary for each run of the program. However, the variation is not severe: Across the test images, the PSNR values of individual program runs differ by ± 0.3 on average. As for the SSIM values, the variation is negligible (± 0.002).

E. Findings

1) Comparison of PSNR and SSIM values: The proposed method proved to be on a par with the `demosaic` function in Matlab and close to Malvar's democaicing method, with an average PSNR of 33.7 dB and an average SSIM of 0.967 (see Table I). Furthermore, is was marginally better than bilinear interpolation. Menon's method proved to be the superior method among all methods. A comparison of PSNR and SSIM values can be seen in Fig. 2.

2) Subjective visual comparison: Visually, the proposed method performs significantly better than any other method except for Menon's demosaicing. Most images demosaiced by the proposed method are indistinguishable from those produced by Menon's method. A visual difference reveals itself rarely, for example, in the case of the image no. 19, which is shown in Fig. 3. No formal subjective test was performed to prove these impressions statistically, however.

3) Additional improvements: The value of the objective criteria typically slightly oscillates in the course of training epochs. Thus, the proposed method can be improved by averaging the images from the last few epochs. This improvement comes basically with no additional cost. In terms of the number of images averaged, 50 images proved to be a sensible

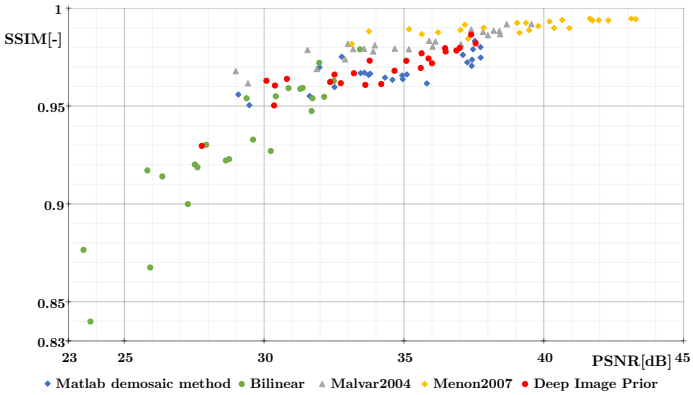

Fig. 2. A scatter plot of the PSNR and SSIM values of all images and methods. The vertical axis represents the SSIM values while the PSNR values are displayed horizontally.

choice with regard to objective quality. Averaging more than 50 images did not bring any significant improvement, doing so only (very slightly) increases the time of execution. The described development results in a PSNR improvement of 0.4 dB on average. The SSIM improves slightly too; however, the improvement is negligible (0.0019).

TABLE I
TOP BLOCK: AVERAGED RESULTS OF TRADITIONAL METHODS. BOTTOM BLOCK: AVERAGED RESULTS OF 10 RUNS OF THE PROPOSED ALGORITHM ON THE IMAGES FROM THE KODAK DATA SET.

	PSNR [dB]	SSIM [-]
Bilinear interpolation	29.1635	0.93098
Malvar's method	35.1601	0.98111
Menon's method	**39.0132**	**0.99049**
Matlab `demosaic` function	34.6454	0.96777
Value at 2000 epochs	33.7053	0.96685
Oracle (without averaging)	34.0019	0.96821
Oracle (with averaging)	34.1320	0.96875
Images 1999 & 2000 averaged	34.0039	0.96797
Images 1900 & 1901 averaged	34.0095	0.96780
Best run (without averaging)	33.9096	0.96845
Best run (with averaging)	**34.3156**	**0.97022**
Images 1991–2000 averaged	34.0862	0.96841
Images 1981–2000 averaged	34.0987	0.96849
Images 1971–2000 averaged	34.1064	0.96856
Images 1961–2000 averaged	34.1133	0.96861
Images 1951–2000 averaged	34.1177	0.96863
Images 1901–2000 averaged	34.1226	0.96861

4) Downsides: The crucial downside of the proposed method and the Deep Image Prior as a whole is that it is extremely computationally demanding compared with other methods. Averaged over all test images, the execution time of classic demosaicing methods is below one second, with the fastest being the bilinear interpolation at around 0.4 seconds. The proposed method takes, around 90 seconds on average. Unlike with the other methods, the time is heavily dependent on the computer hardware. The shortest reconstruction times are obtained when a GPU is utilized. Our results were acquired using the NVIDIA Tesla V100S PCIe 32 GB graphics card. It is important to consider that the times presented hold for

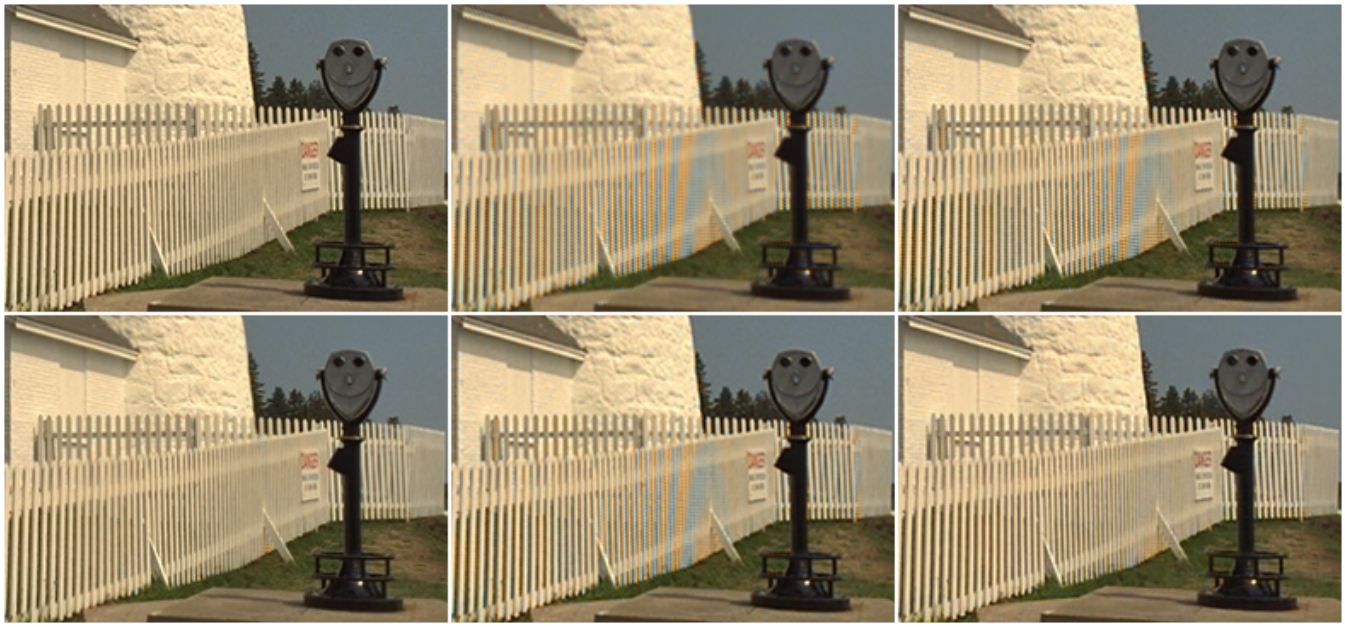

Fig. 3. Visual comparison of all methods on the fence from the Kodak image no. 19 – the colored stripes on the images represent color artifacts mentioned in the text. a) the ground truth; b) bilinear interpolation; c) Malvar's method; d) Menon's method; e) Matlab `demosaic`; f) the proposed method.

768×512 images, which are relatively small in the context of today's cameras. Larger images would require even more computation time.

5) Surprises: To show the maximal capabilities of the proposed method, an oracle approach was selected. In this case, the oracle approach means that the data is treated as if the ground truth image was known. As a result, the images with the maximum PSNR/SSIM value could be selected; in our case, the best image (in terms of PSNR/SSIM) from epochs 1900–2000 was chosen. The greatest surprise came with averaging the last few images. Averaging only the last two images proved to be on a par with the oracle approach, at least in terms of PSNR. Taking it even further, averaging the last 10 images came with better results than the oracle approach. The comparison of PSNR and SSIM values of different numbers of averaged images and other noteworthy statistics can be seen in Table I.

V. Conclusion

The paper proposed a demosaicing method based on the DIP concept. In terms of PSNR and SSIM values, it delivered results very similar to those of the classic Malvar's method. Visually, DIP did not surpass only Menon's method, whose results are comparable with DIP. The most surprising fact was that averaging the last few images of the optimization process brought very solid results, even when compared with the oracle approach. The resulting images from the Kodak data set are available at `https://github.com/sedemto/results_dip`. It is worth mentioning that even though DIP has shown very good visual results, it cannot be used in practice due to the fact that it is very computationally demanding.

Future development could involve the inclusion of an additional regularization of the network to tackle aliasing artifacts even more strongly. Also, the joint processing of the RGB channels by the DNN could be revised. Finally, in difference to most of other methods, the DIP-based demosaicing could be straightforwardly adapted to the case of a random CFA, where aliasing issues are automatically restrained.

Acknowledgment

This work was supervised by dr. Pavel Rajmic, Signal Processing Laboratory, Brno University of Technology.

References

[1] B. E. Bayer, "Color imaging array," US3971065A, Jul. 20, 1976.
[2] H. S. Malvar, L. He, and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," in *2004 IEEE Intl. Conference on Acoustics, Speech, and Signal Proc.*, May 2004, vol. 3.
[3] D. Menon, S. Andriani, and G. Calvagno, "Demosaicing With Directional Filtering and a posteriori Decision," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 132–141, Jan. 2007.
[4] T. Klatzer et al., "Learning joint demosaicing and denoising based on sequential energy minimization," IEEE *Intl. Conference on Computational Photography* (ICCP), USA, 2016
[5] E. Kurniawan, Y. Park, and S. Lee, "Noise-Resistant Demosaicing with Deep Image Prior Network and Random RGBW Color Filter Array," *Sensors* 22, no. 5: 1767, 2022
[6] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep Image Prior," *Intl. Journal of Computer Vision*, vol. 128, no. 7, pp. 1867–1888, Jul. 2020.
[7] M. Bakr, "Kodak Lossless True Color Image Suite." Oct. 14, 2022. Accessed: Mar. 08, 2023. [Online].
[8] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention 2015*, Oct. 2015, pp. 234–241.
[9] D. P. Kingma and L. J. Ba, "Adam: A Method for Stochastic Optimization," in *Intl. Conference on Learning Representations (ICLR)*, 2015.
[10] Z. Wang et al., "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Proc.*, vol. 13, no. 4, 2004.
[11] S. van der Walt et al., "scikit-image: Image processing in Python," *PeerJ*, vol. 2, p. e453, Jun. 2014.