Published in Image Processing On Line on 2021–07–27.
Submitted on 2021–05–13, accepted on 2021–07–01.
ISSN 2105–1232 © 2021 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.2021.358

A Mathematical Analysis and Implementation of Residual Interpolation Demosaicking Algorithms

Qiyu Jin[1], Yu Guo[1], Jean-Michel Morel[2], Gabriele Facciolo[2]

[1] Inner Mongolia University ({qyjin2015, yuguomath}@aliyun.com)
[2] Centre Borelli, ENS Paris-Saclay, CNRS ({jean-michel.morel, gabriele.facciolo}@ens-paris-saclay.fr)

Communicated by Luis Álvarez *Demo edited by* Gabriele Facciolo and Qiyu Jin

Abstract

Demosaicking is the process of reconstructing the full color image from its mosaic version on a Bayer pattern. It is an integral part of the image processing pipeline for single sensor digital color cameras. Demosaicking algorithms based on residual interpolation are interesting because they produce competitive results with a low computational complexity. In this article, we provide an analysis and careful implementation of the most relevant residual based demosaicking algorithms. Our contribution is twofold. First, we present an analysis of the mathematical principles of demosaicking algorithms from the Hamilton-Adams interpolation to the recent "adaptive residual interpolation". Our analysis untangles the relations of these algorithms and how each is improving on the preceding ones. Lastly, we provide a comparison between most recent state of the art methods on several image data sets and discuss their performances.

Keywords: demosaicking; residual interpolation; Bayer color filter array; guided filter

1 Introduction

Color images require at least three color components per pixel, so acquiring a color digital camera would require at least three different sensors for every pixel, each of them capturing information about a particular light wavelength. A first technical solution is to place three spectral sensor arrays on a plane. Then the light entering the camera is split and projected on each spectral sensor array. Unfortunately, this solution is very expensive. Furthermore it requires an accurate registration between the different acquired spectral images. Most digital cameras use a simpler color filter array (CFA) to capture the color information of a scene. At each pixel in the array, only one color component (R, G or B) is measured. The most common type of CFA is the Bayer array [5], shown in Figure 1, in which two out of four pixels measure the green component, one measures red and one blue.

Since only one color value is measured at each pixel, the resulting image is a scalar mosaic. The process of completing the missing red, green and blue values at each pixel is called demosaicking.

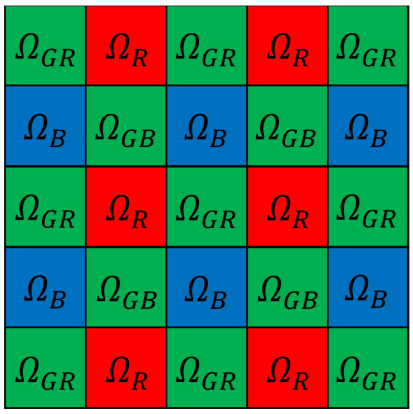


Figure 1: The RGB Bayer color filter array, CFA, which is used by most cameras.

The simplest demosaicking method consists in performing a bilinear interpolation of the known neighboring pixels, however this results in serious zipper effects. In order to smooth the edges of the interpolated image, Laroche and Prescott [24] introduced a gradient based interpolation. The gradient values of horizontal and vertical directions are compared with each other in order to select one of the directions as the preferred orientation for the interpolation of the green image and then the green image is interpolated. After that, the interpolation of the red and blue images is guided by the interpolated green image. Hence, the most important step is to estimate the missing green values. Hamilton and Adams (HA) [15, 1] improved the gradient based interpolation of the green channel by taking into account the second order derivatives estimated from the red and blue channels, thus, taking advantage of the inter-channel correlation. At each pixel, color interpolation is carried out in an adaptive direction given by the estimated gradient. Direction adaptive filtering is the most popular approach to color demosaicking for producing competitive results. Other direction adaptive filterings [8, 31] have been proposed by more complicated gradient-based demosaicking schemes. However, the gradient estimate is not robust for images whose sampling rate is below the Nyquist frequency. This is the main cause of color artifacts in demosaicked images.

In order to overcome the limit of Nyquist frequency, the correlation between the color channels is exploited. The three color channels of a natural image are highly correlated, which amounts to assuming that the difference between the R, G and B channels is smooth. Then Zhang and Wu [41] proposed to estimate the missing green samples in both horizontal and vertical directions by a directional linear minimum mean square-error estimation (DLMMSE) technique. This method combines horizontal and vertical color differences to the final difference and then adds them to the available (red/blue) target pixel to get an estimated green value. Following the idea of [41], Pekkucuksen and Altunbasak [30] decoupled the north-south and east-west directions from each other and then computed four directional color differences, i.e. for the north, south, east and west directions. Then the final difference to estimate the missing green value is obtained by combining the four directional color differences. Kiku at al. [20] then proposed to use a guided filter upsampling [16] and residual interpolation technique in the framework of [30] in order to smooth the edges. Their method is called residual interpolation (RI). Iterative versions of this algorithm were proposed in [38, 39]. In 2014, Kiku at al. [21] proposed minimized-Laplacian residual interpolation (MLRI), which estimated the tentative pixel values by minimizing the cost function on the Laplacian of the image using a guided filter. The weighted minimized-Laplacian residual interpolation (MLRI+wei) [22] introduces a weighted average of linear coefficients of the guided filter. Jaiswal et al. [18] and Wu et al. [36, 37] proposed to use an RI-based algorithm to improve the demosaicking performance. Finally the Adaptive Residual Interpolation (ARI) [28, 27] adaptively combines RI and MLRI, and selects

a suitable iteration number at each pixel. All the above mentioned algorithms interpolate the green image first, and then estimate the red and blue images by using information of the interpolated green image.

Besides the local interpolation based algorithms mentioned above, many more sophisticated techniques have been introduced: the "non-local" algorithms based on the grouping of similar patches [26, 6, 7, 42, 10, 11], wavelet-based algorithms [25, 2, 40], dictionary learning based algorithms [17, 4, 26]. Finally the recent deep learning methods [14, 34, 32, 33, 23] work very well for natural images.

In this paper, we shall first analyze in depth a seed method, the Hamilton-Adams interpolation (HA) [15, 1], then its generalizations, namely the gradient based threshold free method (GBTF) [30], RI [20], MLRI [21], MLRI+wei [22] and ARI [28, 27]. Our goal is to establish their relation and underlying mathematical models. The first key contribution of our research is to display the mathematical principles of these algorithms and to analyze the successive improvements from HA to ARI. Our second contribution is the comparison of these algorithms with the most recent and sophisticated state of the art demosaicking algorithms, to put in evidence their strengths and weaknesses.

This paper is organized as follows. Section 2 is dedicated to notation. Section 3 analyzes the Hamilton-Adams (HA) interpolation method. Section 4 investigates GBTF, the Gradient based threshold free interpolation method. Residual based algorithms (RI, MLRI and MLRI+wei) are studied together and different guided filters are introduced in Section 5. Section 6 is dedicated to the most advanced residual interpolation algorithm, ARI. In Section 7 we perform a comprehensive comparison of the performance of all algorithms for both visual and quantitative criteria.

2 Mosaicking and Notation

Definition 1. *(Arithmetic Operators) Given two scalar matrices $A(i,j)$ and $B(i,j)$ with the same dimensions, the operation symbol $.*$ denotes their element-wise multiplication. Thus setting $C = A.*B$, then for each element $C(i,j)$ of matrix C, $C(i,j) = A(i,j)B(i,j)$. It is natural to note $A.*A$ by A^2. Similarly the operation symbol $./$ denotes the element-wise right division, i.e. if $C = A./B$, then for each element $C(i,j)$ of matrix C, $C(i,j) = A(i,j)/B(i,j)$. We also write $C = |A|$ for $C(i,j) = |A(i,j)|$.*

In a single-sensor camera equipped with a color filter array (CFA) [5], only one pixel value among the three RGB values is recorded at each pixel. Consider a CFA block as shown in Figure 1, the CFA model can be obtained by masking the RGB image

$$\begin{bmatrix} M_R .* R(i,j) \\ (M_{GR} + M_{GB}) .* G(i,j) \\ M_B .* B(i,j) \end{bmatrix},$$

where $(R,G,B)(i,j)$, $(i,j) \in \Omega = [1,2,\cdots,N_1] \times [1,2,\cdots,N_2]$ is a complete RGB color image, R, G, B representing the red, green, and blue channel respectively and M_R, M_{GR}, M_{GB}, M_B are the CFA masks.

In detail, the mask matrices are written as

$$M_R(i,j) = \begin{cases} 1, & \text{if } (i,j) \in \Omega_R; \\ 0, & \text{if } (i,j) \notin \Omega_R, \end{cases}$$

$$M_{GR}(i,j) = \begin{cases} 1, & \text{if } (i,j) \in \Omega_{GR}; \\ 0, & \text{if } (i,j) \notin \Omega_{GR}, \end{cases}$$

$$M_{GB}(i,j) = \begin{cases} 1, & \text{if } (i,j) \in \Omega_{GB}; \\ 0, & \text{if } (i,j) \notin \Omega_{GB}, \end{cases}$$

$$M_B(i,j) = \begin{cases} 1, & \text{if } (i,j) \in \Omega_B; \\ 0, & \text{if } (i,j) \notin \Omega_B, \end{cases}$$

where $\Omega_R, \Omega_{GR}, \Omega_{GB}, \Omega_B \subseteq \Omega$ are disjoint sets of pixels which record red, green, and blue values respectively (see Figure 1) and satisfy $\Omega_R \cup \Omega_{GR} \cup \Omega_{GB} \cup \Omega_B = \Omega$. The green set of pixels Ω_G is combined with Ω_{GR} and Ω_{GB} as shown in Figure 1, i.e. $\Omega_{GR} \cup \Omega_{GB} = \Omega_G$ and $\Omega_{GR} \cap \Omega_{GB} = \emptyset$. For convenience, we introduce a green mask, and a non-green mask as follows

$$M_G = M_{GR} + M_{GB}, \quad \text{and} \quad M_{IG} = M_R + M_B.$$

Hence, putting all mosaicked color values in one matrix leads to the complete mosaicked image

$$Q = M_R. * R + M_{GR}. * G + M_{GB}. * G + M_B. * B, \tag{1}$$

which is equivalent to

$$Q(i,j) = \begin{cases} R(i,j), & \text{if } (i,j) \in \Omega_R, \\ G(i,j), & \text{if } (i,j) \in \Omega_G, \\ B(i,j), & \text{if } (i,j) \in \Omega_B. \end{cases}$$

2.1 Tensor Notation, Filters

Given an R, G, B image, we now set as a convenient notation,

$$\Phi = \begin{bmatrix} \Phi[1] \\ \Phi[2] \\ \Phi[3] \\ \Phi[4] \end{bmatrix} = \begin{bmatrix} M_R. * R \\ M_{GR}. * G \\ M_{GB}. * G \\ M_B. * B \end{bmatrix},$$

i.e. $\Phi[1] = M_R. * R$, $\Phi[2] = M_{GR}. * G$, $\Phi[3] = M_{GB}. * G$ and $\Phi[4] = M_B. * B$. Notice that for each pixel (i,j) at least three of these values are zero. Every element $\Phi[k,i,j]$ in the tensor Φ is given by

$$\Phi[k,i,j] = \Phi[k](i,j), \quad k \in \{1,2,3,4\} \quad \text{and} \quad (i,j) \in \Omega.$$

Two special permutation transformations $\mathrm{T_H}$ and $\mathrm{T_V}$ of the tensor Φ are defined by

$$\mathrm{T_H}(\Phi) = \mathrm{T_H}\left(\begin{bmatrix} \Phi[1] \\ \Phi[2] \\ \Phi[3] \\ \Phi[4] \end{bmatrix} \right) = \begin{bmatrix} \Phi[2] \\ \Phi[1] \\ \Phi[4] \\ \Phi[3] \end{bmatrix}, \tag{2}$$

and

$$\mathrm{T_V}(\Phi) = \mathrm{T_V}\left(\begin{bmatrix} \Phi[1] \\ \Phi[2] \\ \Phi[3] \\ \Phi[4] \end{bmatrix} \right) = \begin{bmatrix} \Phi[3] \\ \Phi[4] \\ \Phi[1] \\ \Phi[2] \end{bmatrix}. \tag{3}$$

We also define the mask

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}[1] \\ \mathbf{M}[2] \\ \mathbf{M}[3] \\ \mathbf{M}[4] \end{bmatrix} = \begin{bmatrix} M_R \\ M_{GR} \\ M_{GB} \\ M_B \end{bmatrix}.$$

i.e. $\mathbf{M}[1] = M_R$, $\mathbf{M}[2] = M_{GR}$, $\mathbf{M}[3] = M_{GB}$ and $\mathbf{M}[4] = M_B$.

A square window of size $d \times d$ centered at $(i,j) \in \Omega$ will be denoted by

$$\mathcal{N}_{i,j}^d = \{(s,t) : ||(i,j) - (s,t)||_\infty \leq d_s\},$$

where $d_s = \frac{d-1}{2}$ is a positive integer. For each $(i,j) \in \Omega$, the matrix

$$Q(\mathcal{N}_{i,j}^d) = (Q(s,t))_{(s,t)\in\mathcal{N}_{i,j}^d},$$

formed by the values $Q(s,t)$ of the image at pixels $(s,t) \in \mathcal{N}_{i,j}^d$, will be called data patch or similarity patch centered at (i,j). For example,

$$Q(\mathcal{N}_{i,j}^3) = \begin{bmatrix} Q(i-1,j-1) & Q(i,j-1) & Q(i+1,j-1) \\ Q(i-1,j) & Q(i,j) & Q(i+1,j) \\ Q(i-1,j+1) & Q(i,j+1) & Q(i+1,j+1) \end{bmatrix}.$$

The horizontal set of pixels of size $d \times 1$ centered at $(i,j) \in \Omega$ and the vertical set of pixels of size $1 \times d$ centered at $(i,j) \in \Omega$ are denoted by

$$\mathcal{H}_{i,j}^d = \{(s,j) : |s-i| \leq d_s\} \quad \text{and} \quad \mathcal{V}_{i,j}^d = \{(i,t) : |t-j| \leq d_s\},$$

respectively, and the vectors

$$Q(\mathcal{H}_{i,j}^d) = (Q(s,t))_{(s,t)\in\mathcal{H}_{i,j}^d} \quad \text{and} \quad Q(\mathcal{V}_{i,j}^d) = (Q(s,t))_{(s,t)\in\mathcal{V}_{i,j}^d}, \tag{4}$$

will be called horizontal data vector and vertical data vector. For instance

$$Q(\mathcal{H}_{i,j}^5) = \begin{bmatrix} Q(i-2,j) & Q(i-1,j) & Q(i,j) & Q(i+1,j) & Q(i+2,j) \end{bmatrix},$$

and

$$Q(\mathcal{V}_{i,j}^5) = \begin{bmatrix} Q(i,j-2) & Q(i,j-1) & Q(i,j) & Q(i,j+1) & Q(i,j+2) \end{bmatrix}^T.$$

Here we introduce notation for two local mean functions: the box mean

$$\overline{I} = f_m(I, d_s), \tag{5}$$

which is defined by

$$\overline{I}(i,j) = \frac{1}{(2d_s+1)^2} \sum_{(s,t)\in\mathcal{N}_{i,j}^d} I(s,t),$$

and the weighted mean

$$\overline{I} = f_{wm}(I, w, d_s), \tag{6}$$

which is defined by

$$\overline{I}(i,j) = \frac{\sum_{(s,t)\in\mathcal{N}_{i,j}^d} w(s,t)I(s,t)}{\sum_{(s,t)\in\mathcal{N}_{i,j}^d} w(s,t)}.$$

The scalar products of vectors or matrices A and B is denoted by $< A, B >$, for example,

$$\left\langle \begin{bmatrix} a_{1,1} & a_{2,1} \\ a_{1,2} & a_{2,2} \end{bmatrix}, \begin{bmatrix} b_{1,1} & b_{2,1} \\ b_{1,2} & b_{2,2} \end{bmatrix} \right\rangle = a_{1,1}b_{1,1} + a_{1,2}b_{1,2} + a_{2,1}b_{2,1} + a_{2,2}b_{2,2}. \tag{7}$$

The cross-correlation (or just correlation) between a $(2n_1+1)\times(2n_2+1)$ kernel A and an $N_1\times N_2$ image B is defined by

$$A\otimes B(i,j)=\sum_{s=-n_1}^{n_1}\sum_{t=-n_2}^{n_2}A(s,t)B(\varsigma(i,s),\tau(j,t)),$$

where

$$\varsigma(i,s)=\begin{cases} i+s, & \text{if } 1\le i+s\le N_1;\\ 1-(i+s), & \text{if } i+s<1;\\ 2N_1-(i+s), & \text{if } i+s>N_1,\end{cases}$$

and

$$\tau(j,t)=\begin{cases} j+t, & \text{if } 1\le j+t\le N_2;\\ 1-(j+t), & \text{if } j+t<1;\\ 2N_2-(j+t), & \text{if } j+t>N_2.\end{cases}$$

According to this definition, this correlation operation extends the image B by replication (by the value of the closest pixel) at the boundary. For convenience, we also define the correlation operation between a matrix and a tensor

$$A\otimes\Phi=A\otimes\begin{bmatrix}\Phi[1]\\ \Phi[2]\\ \Phi[3]\\ \Phi[4]\end{bmatrix}=\begin{bmatrix}A\otimes\Phi[1]\\ A\otimes\Phi[2]\\ A\otimes\Phi[3]\\ A\otimes\Phi[4]\end{bmatrix}.$$

3 The Hamilton-Adams Interpolation

Directional filtering is the most popular approach to color demosaicking. The best known directional interpolation scheme is perhaps the Laplacian filter proposed by Hamilton and Adams (HA) [15, 1]. Their key assumption is that in the "red" lines of the CFA, the second horizontal derivatives of R and G are nearly equal, and the same assumption for the "blue" lines links the horizontal derivatives of G and B. Similarly, the second vertical derivatives of R and G are nearly equal on the "red" columns of the CFA, and the analogue assumption for G and B on "blue" columns. This amounts to write for example

$$\frac{\partial^2 G(i,j)}{\partial i^2}\simeq\frac{\partial^2 R(i,j)}{\partial i^2},\quad \frac{\partial^2 G(i,j)}{\partial j^2}\simeq\frac{\partial^2 R(i,j)}{\partial j^2}.$$

To understand how such assumptions are used, let us state and explain the main formulas of the HA algorithm. Applying a second order Taylor formula to the green pixels yields

$$G(i-1,j)=G(i,j)-\frac{\partial G}{\partial i}(i,j)+\frac{1}{2}\frac{\partial^2 G}{\partial i^2}(i,j)+o(1),$$

and

$$G(i+1,j)=G(i,j)+\frac{\partial G}{\partial i}(i,j)+\frac{1}{2}\frac{\partial^2 G}{\partial i^2}(i,j)+o(1).$$

(Here "1" is assumed small, which makes sense if we consider that this is a pixel unit.) Summing the last two relations yields

$$\frac{G(i-1,j)+G(i+1,j)}{2}\simeq G(i,j)+\frac{1}{2}\frac{\partial^2 G}{\partial i^2}(i,j).$$

Hence,

$$G(i,j) \simeq \frac{G(i-1,j)+G(i+1,j)}{2} - \frac{1}{2}\frac{\partial^2 G}{\partial i^2}(i,j). \tag{8}$$

The HA method then takes the assumption that $\frac{\partial^2 G}{\partial i^2}(i,j) = \frac{\partial^2 R}{\partial i^2}(i,j)$ and it approximates the red derivative by the difference,

$$\frac{\partial^2 R}{\partial i^2}(i,j) \simeq \frac{R(i-2,j)-2R(i,j)+R(i+2,j)}{4}. \tag{9}$$

It therefore follows from (8) and (9) that

$$G(i,j) \simeq \frac{G(i-1,j)+G(i+1,j)}{2} - \frac{R(i-2,j)-2R(i,j)+R(i+2,j)}{4}.$$

To summarize, all of the HA formulas will be based on applying a second order discrete Taylor formula to all channels R, G, B and transferring known discrete second derivatives from R and B to G and conversely from G to R and B.

3.1 HA Green Channel Interpolation

Through the above analysis, the green HA horizontal interpolation at the red pixels $(i,j) \in \Omega_R$ can be expressed as

$$\begin{aligned} \widetilde{G}_H(i,j) &= \frac{G(i-1,j)+G(i+1,j)}{2} - \frac{R(i-2,j)-2R(i,j)+R(i+2,j)}{4} \\ &= \frac{1}{2} < G(\mathcal{H}^3_{i,j}), K_H > - \frac{1}{4} < R(\mathcal{H}^5_{i,j}), \widetilde{\Delta}_H >, \quad (i,j) \in \Omega_R. \end{aligned} \tag{10}$$

where $G(\mathcal{H}^3_{i,j})$ is defined by (4) with G instead of Q, the scalar product $< \cdot, \cdot >$ is defined by (7), the vector K_H is the horizontal interpolation kernel

$$K_H = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \tag{11}$$

and $\widetilde{\Delta}_H$ corresponds to the discrete horizontal Laplacian operator,

$$\widetilde{\Delta}_H = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \end{bmatrix}. \tag{12}$$

By replacing the operator $< \cdot, \cdot >$ with cross-correlation operations $\otimes$ we obtain the equivalent formulation

$$M_R . * \widetilde{G}_H = \frac{1}{2}(M_{GR} . * G) \otimes K_H - \frac{1}{4}(M_R . * R) \otimes \widetilde{\Delta}_H. \tag{13}$$

Similarly the vertical interpolations at the red pixels $(i,j) \in \Omega_R$ are given by

$$\widetilde{G}_V(i,j) = \frac{1}{2} < G(\mathcal{V}^3_{i,j}), K_V > - \frac{1}{4} < R(\mathcal{V}^5_{i,j}), \widetilde{\Delta}_V >, \quad (i,j) \in \Omega_R, \tag{14}$$

where $G(\mathcal{V}^3_{i,j})$ is defined by (4) with G instant of Q, and

$$K_V = K_H^T, \quad \text{and} \quad \widetilde{\Delta}_V = \widetilde{\Delta}_H^T. \tag{15}$$

The correlation-based version of the formula (14) is

$$M_R . * \widetilde{G}_V = \frac{1}{2}(M_{GB} . * G) \otimes K_V - \frac{1}{4}(M_R . * R) \otimes \widetilde{\Delta}_V.$$

In the same way, the interpolation of blue pixels set is given as

$$M_B. * \widetilde{G}_H = \frac{1}{2}(M_{GB}. * G) \otimes K_H - \frac{1}{4}(M_B. * B) \otimes \widetilde{\Delta}_H, \tag{16}$$

and

$$M_B. * \widetilde{G}_V = \frac{1}{2}(M_{GR}. * G) \otimes K_V - \frac{1}{4}(M_B. * B) \otimes \widetilde{\Delta}_V.$$

We deduce the green interpolation at pixels set $\Omega_R \cup \Omega_B$ as follows

$$\begin{aligned}
M_{IG}. * \widetilde{G}_H &= M_R. * \widetilde{G}_H + M_B. * \widetilde{G}_H \\
&= \frac{1}{2}(M_{GR}. * G) \otimes K_H - \frac{1}{4}(M_R. * R) \otimes \widetilde{\Delta}_H \\
&\quad + \frac{1}{2}(M_{GB}. * G) \otimes K_H - \frac{1}{4}(M_B. * B) \otimes \widetilde{\Delta}_H \\
&= \frac{1}{2}(M_G. * G) \otimes K_H - \frac{1}{4}(M_R. * R + M_B. * B) \otimes \widetilde{\Delta}_H \\
&= \frac{1}{2}(M_G. * Q) \otimes K_H - \frac{1}{4}[(M_R + M_B). * Q] \otimes \widetilde{\Delta}_H \\
&= \frac{1}{2}(M_G. * Q) \otimes K_H - \frac{1}{4}(M_{IG}. * Q) \otimes \widetilde{\Delta}_H \\
&= M_{IG}. * (\frac{1}{2}Q \otimes K_H - \frac{1}{4}Q \otimes \widetilde{\Delta}_H).
\end{aligned}$$

The definition of M_{IG} leads to the first equation. The second equation means that the green horizontal interpolation interpolates the green pixels values, uses the red pixels values by formula (13) and the blue pixels values by formula (16). We get the fourth equation by using the definition of Q given by (1): it is easy to get $M_G. * Q = M_G. * G$ and $(M_R + M_B). * Q = M_R. * R + M_B. * B$. The sixth equation is valid because $[\frac{1}{2}(M_G. * Q) \otimes K_H](i,j) = 0$ for all $(i,j) \in \Omega_G$. This yields the horizontal green interpolation

$$M_{IG}. * \widetilde{G}_H = M_{IG}. * (\frac{1}{2}Q \otimes K_H - \frac{1}{4}Q \otimes \widetilde{\Delta}_H). \tag{17}$$

In the same way, we obtain the vertical green interpolation at pixels set $\Omega_R \cup \Omega_B$,

$$M_{IG}. * \widetilde{G}_V = M_{IG}. * (\frac{1}{2}Q \otimes K_V - \frac{1}{4}Q \otimes \widetilde{\Delta}_V). \tag{18}$$

The sums of the absolute value of the first order partial derivatives and the absolute value of the second order partial derivatives

$$\left|\frac{\partial G(i,j)}{\partial i}\right| + \left|\frac{\partial^2 G(i,j)}{\partial i^2}\right|,$$

and

$$\left|\frac{\partial G(i,j)}{\partial j}\right| + \left|\frac{\partial^2 G(i,j)}{\partial j^2}\right|,$$

are used to estimate how flat G is in the i direction (horizontal direction) and j direction (vertical direction). For that the following classifiers are defined

$$CL_H(i,j) = \begin{cases} |< G(\mathcal{H}^3_{i,j}), D_H >| + |< R(\mathcal{H}^5_{i,j}), \widetilde{\Delta}_H >|, & \text{if } (i,j) \in \Omega_R, \\ |< G(\mathcal{H}^3_{i,j}), D_H >| + |< B(\mathcal{H}^5_{i,j}), \widetilde{\Delta}_H >|, & \text{if } (i,j) \in \Omega_B, \end{cases} \tag{19}$$

$$CL_V(i,j) = \begin{cases} |< G(\mathcal{V}^3_{i,j}), D_V >| + |< R(\mathcal{V}^5_{i,j}), \widetilde{\Delta}_V >|, & \text{if } (i,j) \in \Omega_R, \\ |< G(\mathcal{V}^3_{i,j}), D_V >| + |< B(\mathcal{V}^5_{i,j}), \widetilde{\Delta}_V >|, & \text{if } (i,j) \in \Omega_B. \end{cases} \tag{20}$$

with

$$D_H = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad D_V = D_H^T = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T. \tag{21}$$

Expressing Equation (19) using cross-correlations we obtain

$$\begin{aligned} M_{IG}. * CL_H &= M_R. * (|M_G. * G \otimes D_H| + |M_R. * R \otimes \widetilde{\Delta}_H|) \\ &\quad + M_B. * (|M_G. * G \otimes D_H| + |M_B. * B \otimes \widetilde{\Delta}_H|) \\ &= M_R. * (|M_G. * Q \otimes D_H| + |M_R. * Q \otimes \widetilde{\Delta}_H|) \\ &\quad + M_B. * (|M_G. * Q \otimes D_H| + |M_B. * Q \otimes \widetilde{\Delta}_H|) \\ &= (M_R + M_B). * (|M_G. * Q \otimes D_H| + |(M_R + M_B). * Q \otimes \widetilde{\Delta}_H|) \\ &= (M_R + M_B). * (|Q \otimes D_H| + |Q \otimes \widetilde{\Delta}_H|). \end{aligned}$$

In the above derivation the first step combines the red and blue classifiers, then the definition of Q is applied. Lastly, the mask $M_R + M_B$ and definition of D_H and $\widetilde{\Delta}_H$ imply that we can remove M_G from $|M_G. * Q \otimes D_H|$ and $M_R + M_B$ from $|(M_R + M_B). * Q \otimes \widetilde{\Delta}_H|$ yielding the result.

A similar derivation yields the expression for the vertical classifiers from (20)

$$M_{IG}. * CL_V = M_{IG}. * (|Q \otimes D_V| + |Q \otimes \widetilde{\Delta}_V|).$$

The magnitudes CL_H and CL_V measure how flat the image is in the horizontal or vertical direction. These classifiers are composed of Laplacian second-order terms for the red/blue data and gradients for the green data. When $CL_H(i,j) < CL_V(i,j)$ it means that the horizontal direction is more flat than the vertical direction, hence the horizontal interpolation $\widetilde{G}_H$ is better than the vertical interpolation $\widetilde{G}_V$ at pixel (i,j). Otherwise, the vertical interpolation $\widetilde{G}_V$ is better than the horizontal interpolation $\widetilde{G}_H$ at the pixel (i,j). When $CL_H(i,j) = CL_V(i,j)$ the horizontal direction is as flat as the vertical direction, in which case the mean of $\widetilde{G}_H$ and $\widetilde{G}_V$ at pixel (i,j) is taken. In summary the HA green image interpolation is obtained by

$$\widehat{G}(i,j) = \begin{cases} \widetilde{G}_H(i,j), & \text{if } (i,j) \in \Omega_R \cup \Omega_B \text{ and } CL_H(i,j) < CL_V(i,j), \\ \widetilde{G}_V(i,j), & \text{if } (i,j) \in \Omega_R \cup \Omega_B \text{ and } CL_H(i,j) > CL_V(i,j), \\ \frac{\widetilde{G}_H(i,j) + \widetilde{G}_V(i,j)}{2}, & \text{if } (i,j) \in \Omega_R \cup \Omega_B \text{ and } CL_H(i,j) = CL_V(i,j), \\ G(i,j), & \text{if } (i,j) \in \Omega_G. \end{cases}$$

The pseudo-code describing the green interpolation can be found in Algorithm 1.

3.2 HA Red and Blue Channel Interpolation

Once the green samples have been filled, the red and blue samples are interpolated in a similar way using the $1-$D Laplacian of the green channel as reference. The pseudo-code describing the red interpolation is given in Algorithm 2. The HA red interpolation for pixels $(i,j) \in \Omega_{GR}$ uses the horizontal interpolation

$$M_{GR}. * \widetilde{R}_H(i,j) = M_{GR}. * (\frac{1}{2}(M_R. * R) \otimes K_H - \frac{1}{4}\widehat{G} \otimes \Delta_H),$$

and for pixels $(i,j) \in \Omega_{GB}$ the analogous vertical interpolation is used,

$$M_{GB}. * \widetilde{R}_V(i,j) = M_{GB}. * (\frac{1}{2}(M_R. * R) \otimes K_V - \frac{1}{4}\widehat{G} \otimes \Delta_V),$$

Algorithm 1 Pseudo-code for HA Green Interpolation

Input: Mosaicked image $\Phi = [M_R .* R, M_{GR} .* G, M_{GB} .* G, M_B .* B]^T$,
Mask $\mathbf{M} = [M_R, M_{GR}, M_{GB}, M_B]^T$
Output: interpolated green $\widehat{G}$

function HA_G($\Phi, \mathbf{M}$)
  $K_H \leftarrow [1, 0, 1]$, $K_V \leftarrow K_H^T$
  $\widetilde{\Delta}_H \leftarrow [1, 0, -2, 0, 1]$, $\widetilde{\Delta}_V \leftarrow \widetilde{\Delta}_H^T$
  $D_H \leftarrow [-1, 0, 1]$, $D_V \leftarrow D_H^T$
  $Q \leftarrow M_R .* R + M_{GR} .* G + M_{GB} .* G + M_B .* B$
  for $I \in \{H, V\}$ **do**
    $\widetilde{G}_I \leftarrow Q \otimes (\frac{1}{2} K_I - \frac{1}{4} \widetilde{\Delta}_I)$
    $CL_I \leftarrow |Q \otimes D_I| + |Q \otimes \widetilde{\Delta}_I|$
  end for
  $\widehat{G} \leftarrow \widetilde{G}_H$ **when** $CL_V > CL_H$;
    $\widetilde{G}_V$ **when** $CL_V < CL_H$;
    $(\widetilde{G}_H + \widetilde{G}_V)/2$ **otherwise**
  $\widehat{G} \leftarrow (M_R + M_B) .* \widehat{G} + M_{GR} .* G + M_{GB} .* G$
end function

where the horizontal and vertical discrete Laplacian vectors are defined by

$$\Delta_H = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad \text{and} \quad \Delta_V = \Delta_H^T = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}^T. \tag{22}$$

The red value at blue pixels $(i, j) \in \Omega_B$ is interpolated similarly to the green interpolation by substituting the diagonal and anti-diagonal directions to the horizontal and vertical directions that where used for the green interpolation. The diagonal direction interpolation is defined by

$$\begin{aligned} \widetilde{R}_P(i,j) &= \frac{R(i-1,j-1) + R(i+1,j+1)}{2} - \frac{\widehat{G}(i-1,j-1) - 2\widehat{G}(i,j) + \widehat{G}(i+1,j+1)}{4} \\ &= \frac{1}{2} < R(\mathcal{N}_{i,j}^3), K_P > + \frac{1}{4} < \widehat{G}(\mathcal{N}_{i,j}^3), \Delta_P >, \end{aligned} \tag{23}$$

where K_P and Δ_P are defined by

$$K_P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \Delta_P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{24}$$

The correlation expression corresponding to Formula (23) is

$$\begin{aligned} M_B .* \widetilde{R}_P &= \frac{1}{2}(M_R .* R) \otimes K_P - \frac{1}{4}\widehat{G} \otimes \Delta_P \\ &= M_B .* (\frac{1}{2} R \otimes K_P - \frac{1}{4}\widehat{G} \otimes \Delta_P). \end{aligned}$$

In the same way, the anti-diagonal direction interpolation is given by

$$M_B .* \widetilde{R}_N = M_B .* (\frac{1}{2} R \otimes K_N - \frac{1}{4}\widehat{G} \otimes \Delta_N),$$

where

$$K_N = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \Delta_N = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \tag{25}$$

The classifiers for the positive and negative directions are obtained in the same way as the classifiers for green interpolation

$$M_B. * CL_P = M_B. * (|R \otimes D_P| + |\widehat{G} \otimes \Delta_P|),$$

and

$$M_B. * CL_N = M_B. * (|R \otimes D_N| + |\widehat{G} \otimes \Delta_N|),$$

where

$$D_P = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad D_N = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \tag{26}$$

With the same arguments as for the green interpolation, the red interpolation at blue pixels $\widehat{R}$ gets $\widetilde{R}_P$ when $CL_N > CL_P$, $\widetilde{R}_N$ when $CL_N < CL_P$, and $(\widetilde{G}_P + \widetilde{G}_N)/2$ otherwise. The red estimate is given by

$$\widehat{R} = M_B. * \widehat{R}^B + M_{GR}. * \widetilde{R}_H + M_{GB}. * \widetilde{R}_V + M_R. * R.$$

The blue interpolation is obtained in the same way as the red interpolation.

Algorithm 2 Pseudo-code for HA Red Interpolation

Input: Mosaicked image $\Phi = [M_R. * R, M_{GR}. * G, M_{GB}. * G, M_B. * B]^T$,
Mask $\mathbf{M} = [M_R, M_{GR}, M_{GB}, M_B]^T$,
green interpolation $\widehat{G}$
Output: Interpolated red $\widehat{R}$

function HA_RB($\Phi, \mathbf{M}, \widehat{G}$)
  $K_H \leftarrow [1, 0, 1]$, $K_V \leftarrow K_H^T$
  K_N and $K_P \leftarrow$ given by Equations (24) and (25)
  Δ_N and $\Delta_P \leftarrow$ given by Equations (24) and (25)
  Δ_H and $\Delta_V \leftarrow$ given by Equation (22)
  D_N and $D_P \leftarrow$ given by Equation (26)
  for $I \in \{H, V, N, P\}$ **do**
    $\widetilde{R}_I \leftarrow \frac{1}{2} R \otimes K_I - \frac{1}{4} \widehat{G} \otimes \Delta_I$ // HA interpolation for four directions //
    if $I \in \{N, P\}$ **then**
      $CL_I \leftarrow |R \otimes D_I| + |\widehat{G} \otimes \Delta_I|$ // classifiers of positive and negative directions //
    end if
  end for
  // determines the red value for the blue pixel //
  $M_B. * \widehat{R} \leftarrow \widetilde{R}_N$ **when** $CL_P > CL_N$;
  $\widetilde{R}_P$ **when** $CL_P < CL_N$;
  $(\widetilde{R}_N + \widetilde{R}_P)/2$ **otherwise**
  $\widehat{R} \leftarrow M_B. * \widehat{R} + M_{GR}. * \widetilde{R}_H + M_{GB}. * \widetilde{R}_V + M_R. * R$
end function

4 Gradient Based Threshold Free Interpolation

The HA method just uses the horizontal or vertical direction interpolations, but for a given direction, the conditions might be different for pixels falling to the opposite sides of the target pixel especially near edges or in textured regions as illustrated in Figure 2. In order to improve the results in

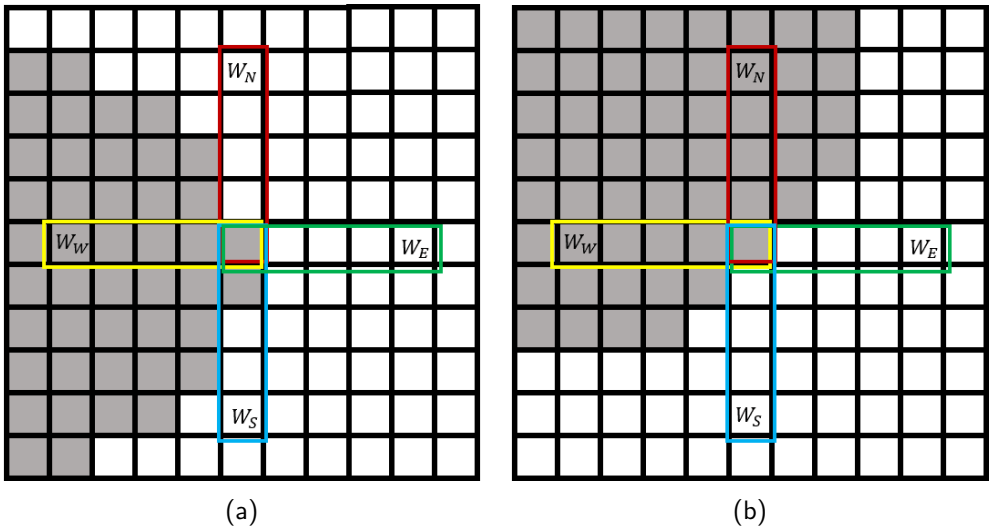


Figure 2: Illustration of the adaptive GBTF interpolation: (a) for the central pixel to be interpolated neither the horizontal nor the vertical interpolations are good. GBTF gives a larger weight value to the west direction, and smaller weights to all other directions. (b): the central pixel cannot be interpolated reliably in the horizontal or vertical direction. GBTF gives larger weights to the west and north interpolations.

these cases Pekkucuksen and Altunbasak [30] proposed to decouple the interpolation in north-south and east-west directions, to consider them separately, and then to combine the estimations from every direction. This eliminates the need for thresholds. Instead of making a hard decision on the direction of the edge they proposed a Gradient Based Threshold Free (GBTF) algorithm, which again assumes that the difference between two colors has a negligible Laplacian in either horizontal or vertical direction. Figure 2 (a) illustrates the advantage of this approach: the central pixel to be interpolated is located near an edge in such a way that neither horizontal nor vertical interpolations are good. One sees that the west direction interpolation is much better than other directions. The GBTF method gives a larger weight value to this direction and smaller weights to the other directions. In the case proposed by Figure 2 (b), we cannot interpolate reliably the central pixel in the horizontal or vertical direction. It is clear that the west and north interpolations are better than the east and south, so the method gives larger weights to the west and north interpolations. The method also first interpolates the G pixel values, then the R and B pixel values are interpolated by using the color difference interpolation.

4.1 GBTF Green Channel Interpolation

The GBTF [30] interpolation process of the G pixel values consists of four steps that are outlined in Figure 3 (see Algorithm 3).

Step (i) The Hamilton-Adams (HA) interpolation [15] is applied in the horizontal and vertical directions to estimate the G values at the R and B pixels and the R and B values at the G pixels. As a result, the horizontally and vertically interpolated R, G, and B pixel values are generated. From Section 3.1, we get $M_{IG}. * \widetilde{G}_H$ and $M_{IG}. * \widetilde{G}_V$ (see Equations (17) and (18)). According to the derivation process of Equation (17), we have

$$M_{GR}. * \widetilde{R}_H = M_{GR}. * [Q \otimes (\frac{1}{2}K_H - \frac{1}{4}\widetilde{\Delta}_H)],$$

$$M_{GB}. * \widetilde{B}_H = M_{GB}. * [Q \otimes (\frac{1}{2}K_H - \frac{1}{4}\widetilde{\Delta}_H)],$$

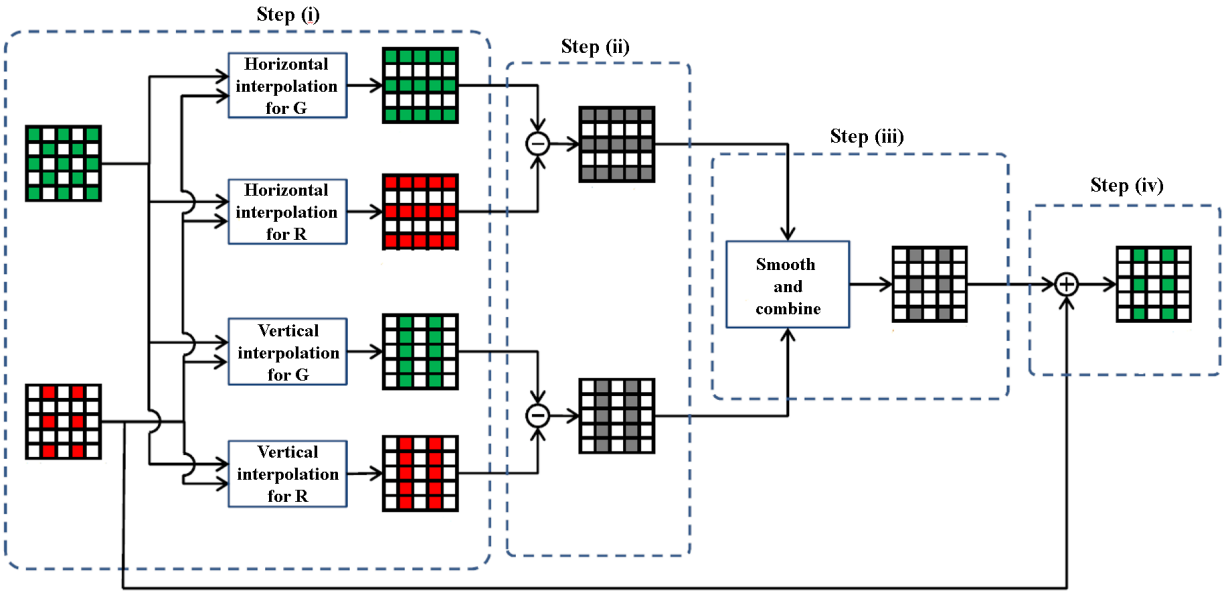


Figure 3: Outline of the proposed G pixel value interpolation at the R pixels which is common to four methods: GBTF, RI, MLRI and MLRI+wei. The type of interpolation used at Step (i) of the pipelines determines the kind of algorithm: HA interpolation $\mapsto$ GBTF, GF interpolation $\mapsto$ RI, MLGF interpolation $\mapsto$ MLRI, and Weighted-MLGF interpolation $\mapsto$ MLRI+wei.

$$M_{GB}.*\widetilde{R}_V = M_{GR}.*[Q\otimes(\frac{1}{2}K_V - \frac{1}{4}\widetilde{\Delta}_V)],$$

and

$$M_{GR}.*\widetilde{B}_V = M_{GR}.*[Q\otimes(\frac{1}{2}K_V - \frac{1}{4}\widetilde{\Delta}_V)].$$

Setting

$$\widetilde{Q}_H = \frac{1}{2}Q\otimes K_H - \frac{1}{4}Q\otimes\widetilde{\Delta}_H,$$
$$\widetilde{Q}_V = \frac{1}{2}Q\otimes K_V - \frac{1}{4}Q\otimes\widetilde{\Delta}_V,$$

then we have

$$\begin{array}{llll} M_{IG}.*\widetilde{G}_H = M_{IG}.*\widetilde{Q}_H, & M_{GR}.*\widetilde{R}_H = M_{GR}.*\widetilde{Q}_H, & \text{and} & M_{GB}.*\widetilde{B}_H = M_{GB}.*\widetilde{Q}_H, \\ M_{IG}.*\widetilde{G}_V = M_{IG}.*\widetilde{Q}_V, & M_{GB}.*\widetilde{R}_V = M_{GB}.*\widetilde{Q}_V, & \text{and} & M_{GR}.*\widetilde{B}_V = M_{GR}.*\widetilde{Q}_V. \end{array} \tag{27}$$

Step (ii) The horizontal difference $(G-R)$ is computed for each pixel as

$$\mathfrak{R}_H^{g,r}(i,j) = \begin{cases} \widetilde{G}_H(i,j) - R(i,j), & \text{if } (i,j)\in\Omega_R, \\ G(i,j) - \widetilde{R}_H(i,j), & \text{if } (i,j)\in\Omega_{GR}, \\ 0, & \text{if } (i,j)\in\Omega_{GB}\cup\Omega_B, \end{cases}$$

which is equivalent to

$$\mathfrak{R}_H^{g,r} = M_R.*(\widetilde{G}_H - R) + M_{GR}.*(G - \widetilde{R}_H), \tag{28}$$

while the horizontal difference $(G-B)$ is

$$\mathfrak{R}_H^{g,b} = M_B.*(\widetilde{G}_H - B) + M_{GB}.*(G - \widetilde{B}_H). \tag{29}$$

Combining (28), (29) and (27), we get

$$\begin{aligned}
\mathfrak{R}_H &= \mathfrak{R}_H^{g,r} + \mathfrak{R}_H^{g,b} \\
&= M_R .* (\widetilde{G}_H - R) + M_{GR} .* (G - \widetilde{R}_H) \\
&\quad + M_B .* (\widetilde{G}_H - B) + M_{GB} .* (G - \widetilde{B}_H) \\
&= (M_R + M_B) .* \widetilde{G}_H + (M_{GR} + M_{GB}) .* G \\
&\quad - (M_R .* R + M_{GR} .* \widetilde{R}_H + M_B .* B + M_{GB} .* \widetilde{B}_H) \\
&= M_{IG} .* \widetilde{G}_H + M_G .* G \\
&\quad - (M_R .* R + M_{GR} .* \widetilde{R}_H + M_B .* B + M_{GB} .* \widetilde{B}_H) \\
&= M_{IG} .* \widetilde{Q}_H + M_G .* Q \\
&\quad - (M_R .* R + M_{GR} .* \widetilde{Q}_H + M_B .* B + M_{GB} .* \widetilde{Q}_H) \\
&= M_{IG} .* \widetilde{Q}_H + M_G .* Q - (M_G .* \widetilde{Q}_H + M_{IG} .* Q) \\
&= (M_{IG} - M_G) .* \widetilde{Q}_H + (M_G - M_{IG}) .* Q,
\end{aligned}$$

i.e.

$$\mathfrak{R}_H = (M_{IG} - M_G) .* (\widetilde{Q}_H - Q).$$

In the same way for the vertical directions, we get

$$\mathfrak{R}_V = (M_{IG} - M_G) .* (\widetilde{Q}_V - Q).$$

Step (iii) The horizontal and vertical color differences are smoothed and then combined into the final color difference estimate. The final difference estimate for the target pixel $(i, j) \in \Omega_{GR}$ is computed as

$$\begin{aligned}
\mathfrak{R}(i,j) = \{ & W_N < f_N, \mathfrak{R}_V(\mathcal{V}_{i,j}^9) > + \\
& W_S < f_S, \mathfrak{R}_V(\mathcal{V}_{i,j}^9) > + \\
& W_E < \mathfrak{R}_H(\mathcal{H}_{i,j}^9), f_E > + \\
& W_W < \mathfrak{R}_H(\mathcal{H}_{i,j}^9), f_W > \} / W_T,
\end{aligned} \tag{30}$$

where

$$W_T = W_N + W_S + W_E + W_W,$$

$$\begin{aligned}
f_E &= \begin{bmatrix} 0 & 0 & 0 & 0 & 26 & 24 & 21 & 17 & 12 \end{bmatrix} / 100, \\
f_W &= \begin{bmatrix} 12 & 17 & 21 & 24 & 26 & 0 & 0 & 0 & 0 \end{bmatrix} / 100, \\
f_S &= \begin{bmatrix} 0 & 0 & 0 & 0 & 26 & 24 & 21 & 17 & 12 \end{bmatrix}^T / 100, \\
f_N &= \begin{bmatrix} 12 & 17 & 21 & 24 & 26 & 0 & 0 & 0 & 0 \end{bmatrix}^T / 100,
\end{aligned} \tag{31}$$

and $< \cdot, \cdot >$ denotes scalar products of vectors defined by (7). The weight for each direction (W_N, W_S, W_E, W_W) is computed using color difference gradients in horizontal and vertical directions as

$$W_E = 1/\max\{\left(< K_G^5, \mathfrak{R}_H^D(\mathcal{N}_{i+1,j}^5) >\right)^2, 1\},$$

$$W_W = 1/\max\{\left(< K_G^5, \mathfrak{R}_H^D(\mathcal{N}_{i-1,j}^5) >\right)^2, 1\},$$

$$W_N = 1/\max\{\left(< K_G^5, \mathfrak{R}_V^D(\mathcal{N}_{i,j-1}^5) >\right)^2, 1\},$$

and

$$W_S = 1/\max\{\left(< K_G^5, \mathfrak{R}_V^D(\mathcal{N}_{i,j+1}^5) >\right)^2, 1\},$$

where K_G^5 is a 5×5 Gaussian kernel matrix, i.e.

$$K_G^5 = \begin{bmatrix} 0.0232 & 0.0338 & 0.0383 & 0.0338 & 0.0232 \\ 0.0338 & 0.0492 & 0.0558 & 0.0492 & 0.0338 \\ 0.0383 & 0.0558 & 0.0632 & 0.0558 & 0.0383 \\ 0.0338 & 0.0492 & 0.0558 & 0.0492 & 0.0338 \\ 0.0232 & 0.0338 & 0.0383 & 0.0338 & 0.0232 \end{bmatrix}. \tag{32}$$

The directional gradients are computed as

$$\mathfrak{R}_H^D(i,j) \;=\; | < \mathfrak{R}_H(\mathcal{H}_{i,j}^3), D_H > | \tag{33}$$

and

$$\mathfrak{R}_V^D(i,j) \;=\; | < \mathfrak{R}_V(\mathcal{V}_{i,j}^3), D_V > |, \tag{34}$$

where the kernels for the horizontal and vertical derivatives, D_H and D_V, are defined by (21). Here there is a difference between the algorithm in paper [30] and the reference code provided by the authors. In the code, (33) and (34) are replaced by

$$\mathfrak{R}_H^D(i,j) = | < \mathfrak{R}_H(\mathcal{H}_{i,j-1}^3), D_H > | + | < \mathfrak{R}_H(\mathcal{H}_{i,j}^3), D_H > | + | < \mathfrak{R}_H(\mathcal{H}_{i,j+1}^3), D_H > |, \tag{35}$$

and

$$\mathfrak{R}_V^D(i,j) = | < \mathfrak{R}_V(\mathcal{V}_{i-1,j}^3), D_V > | + | < \mathfrak{R}_V(\mathcal{V}_{i,j}^3), D_V > | + | < \mathfrak{R}_V(\mathcal{V}_{i+1,j}^3), D_V > |, \tag{36}$$

to calculate $\mathfrak{R}_H^D(i,j)$ and $\mathfrak{R}_V^D(i,j)$.

Step (iv) The G pixel values at the R and B pixels are interpolated by adding the observed R or B pixel values to the final color difference estimates

$$\widehat{G}(i,j) = \begin{cases} R(i,j) + \mathfrak{R}(i,j), & \text{if } (i,j) \in \Omega_R, \\ G(i,j), & \text{if } (i,j) \in \Omega_G, \\ B(i,j) + \mathfrak{R}(i,j), & \text{if } (i,j) \in \Omega_B. \end{cases}$$

i.e.

$$\widehat{G} = M_R. * (\mathfrak{R} + R) + M_G. * G + M_B. * (\mathfrak{R} + B).$$

4.2 GBTF Red and Blue Channel Interpolation

Red pixel values at blue locations are interpolated using the following filter

$$\widehat{R}_0(i,j) = \begin{cases} \widehat{G}(i,j) - < \mathfrak{R}(\mathcal{N}_{i,j}^7), K_{rb} >, & \text{if } (i,j) \in \Omega_B, \\ R(i,j), & \text{if } (i,j) \in \Omega_R, \\ 0, & \text{if } (i,j) \in \Omega_G, \end{cases} \tag{37}$$

where the second-order polynomial interpolation filter K_{rb} (see Appendix of Paliy et al. [29]) was used, namely

$$K_{rb} = \begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 10 & 0 & 10 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 10 & 0 & 10 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 \end{bmatrix} \times \frac{1}{32}, \tag{38}$$

Algorithm 3 Pseudo-code for GBTF Green Interpolation

Input: Mosaicked image $\Phi = [M_R .* R, M_{GR} .* G, M_{GB} .* G, M_B .* B]^T$,
Mask $\mathbf{M} = [M_R, M_{GR}, M_{GR}, M_B]^T$
Output: interpolated green $\widehat{G}, \mathfrak{R}$

function GBTF_G($\Phi, \mathbf{M}$)
// Initialization //
$K_H \leftarrow [1,0,1]$, $K_V \leftarrow K_H^T$
$A_H \leftarrow [1,1,1]$, $A_V \leftarrow A_H^T$
$\widetilde{\Delta}_H \leftarrow [1,0-2,0,1]$, $\widetilde{\Delta}_V \leftarrow \widetilde{\Delta}_H^T$
$D_H \leftarrow [-1,0,1]$, $D_V \leftarrow D_H^T$
$F \leftarrow K_G^5$ defined by (32)
$K_E \leftarrow [0,0,1]$, $K_W \leftarrow [1,0,0]$, $K_S \leftarrow [0,0,1]^T$, $K_N \leftarrow [1,0,0]^T$,
$f_E, f_W, f_S, f_N \leftarrow$ defined by (31)
$Q \leftarrow M_R .* R + M_{GR} .* G + M_{GB} .* G + M_B .* B$
// Main loop: H means horizontal interpolation and V vertical interpolation //
for $I \in \{H, V\}$ **do**
// first step: compute HA interpolation//
$\widetilde{Q}_I \leftarrow Q \otimes (\frac{1}{2}K_I - \frac{1}{4}\widetilde{\Delta}_I)$
// Second step: compute the color difference//
$\mathfrak{R}_I \leftarrow (M_R + M_B - M_{GR} - M_{GB})(\widetilde{Q}_I - Q)$
// Third step: weighted combination of horizontal and vertical color difference//
$\mathfrak{R}_I^D \leftarrow |D_I \otimes \mathfrak{R}_I| \otimes A_I^T$
$\overline{\mathfrak{R}}_I^D \leftarrow F \otimes \mathfrak{R}_I$
end for
$W_E \leftarrow \frac{1}{\left(K_E \otimes \overline{\mathfrak{R}}_H^D\right)^2 + \epsilon}$, $W_W \leftarrow \frac{1}{\left(K_W \otimes \overline{\mathfrak{R}}_H^D\right)^2 + \epsilon}$, $W_S \leftarrow \frac{1}{\left(K_S \otimes \overline{\mathfrak{R}}_V^D\right)^2 + \epsilon}$, $W_N \leftarrow \frac{1}{\left(K_N \otimes \overline{\mathfrak{R}}_V^D\right)^2 + \epsilon}$
$W \leftarrow W_E + W_W + W_S + W_N$
$\mathfrak{R} \leftarrow (W_N .* (f_N \otimes \mathfrak{R}_V) + W_S .* (f_S \otimes \mathfrak{R}_V) + W_W .* (f_W \otimes \mathfrak{R}_H) + W_E .* (f_E \otimes \mathfrak{R}_H))./W$
// Fourth step: final result//
$\widehat{G} \leftarrow M_R .* (\mathfrak{R} + R) + (M_{GR} + M_{GB}) .* G + M_B .* (\mathfrak{R} + B)$
end function

and $< \cdot, \cdot >$ denotes scalar products of matrices. For red pixels at green locations, we use bilinear interpolation of the closest four neighbors. The immediate vertical neighbors of a green pixel are red or blue pixels. The red values at a green pixels are interpolated as follows: for $(i,j) \in \Omega_G$

$$\widehat{R}(i,j) = \begin{cases} G(i,j) - < [\widehat{G} - \widehat{R}_0](\mathcal{N}_{i,j}^3), K_A >, & \text{if } (i,j) \in \Omega_G, \\ \widehat{R}_0(i,j), & \text{if } (i,j) \in \Omega_R \cup \Omega_B, \end{cases} \tag{39}$$

where

$$K_A = \begin{bmatrix} 0 & 1/4 & 0 \\ 1/4 & 0 & 1/4 \\ 0 & 1/4 & 0 \end{bmatrix}. \tag{40}$$

The outline of the GBTF red interpolation and pseudo code is given in Figure 4 (a) and Algorithm 4 respectively. The blue interpolated image $\widehat{B}$ is computed in the same manner.

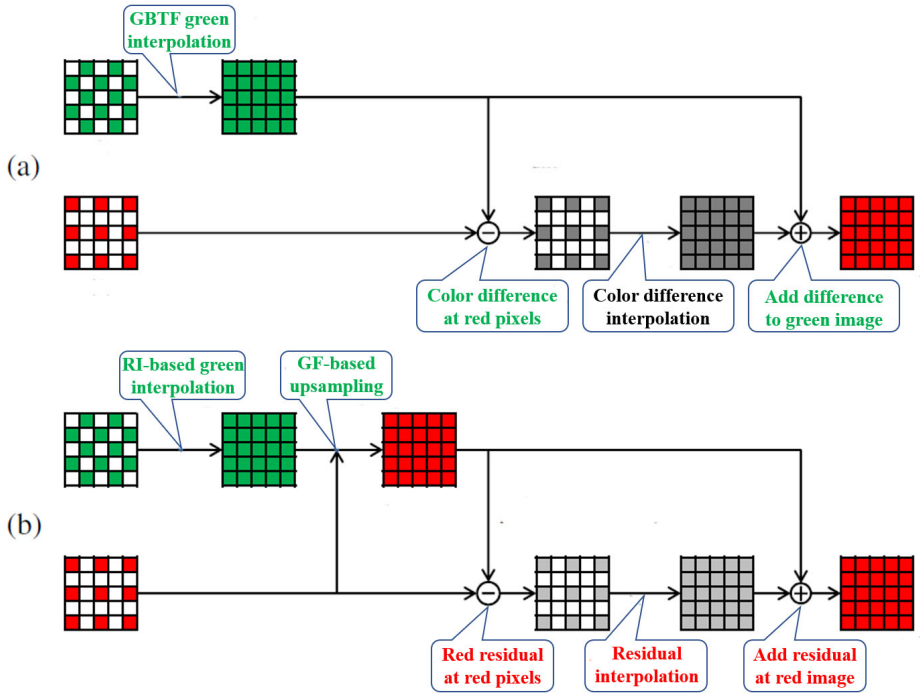


Figure 4: The interpolation of the red pixel values (a) by using GBTF, and (b) by using RI-based interpolation: RI, MLRI and MLRI+wei. The only two differences between RI, MLRI and MLRI+wei are the first and second steps: RI (RI green interpolation and GF upsampling), MLRI (MLRI green interpolation and MLGF upsampling), MLRI+wei (MLRI+wei green interpolation and weighted-MLGF upsampling).

Algorithm 4 Pseudo-code for GBTF Red Interpolation

Input: Mosaicked image $\Phi = [M_R .* R, M_{GR} .* G, M_{GB} .* G, M_B .* B]^T$,
Mask $\mathbf{M} = [M_R, M_{GR}, M_{GR}, M_B]^T$,
green interpolation $\widehat{G}$,
color difference $\mathfrak{R}$
Output: Interpolated red $\widehat{R}$ and blue $\widehat{B}$

function GBTF_RB($\Phi, \mathbf{M}, \widehat{G}, \mathfrak{R}$)
  $K_{rb} \leftarrow$ defined by Equation (38)
  $K_A \leftarrow$ defined by Equation (40)
  $\widehat{R}_0 \leftarrow M_B .* (\widehat{G} - \mathfrak{R} \otimes K_{rb}) + M_R .* R$ (Equation (37))
  $\widehat{R} \leftarrow (M_R + M_B) .* \widehat{R}_0 + (M_{GR} + M_{GB}) .* (G - [\widehat{G} - \widehat{R}_0] \otimes K_A)$ (Equation (39))
end function

5 Residual Interpolation

Residual interpolation (RI) was introduced by Kiku et al. [20]. Similarly to GBTF, RI first interpolates G, then R and B. For the green interpolation, RI computes the horizontal and vertical interpolations of R, G and B images by a Guided Filter (GF) [16], and then obtains the tentative estimates by improving the interpolation results with a residual technique instead of the HA interpolation. Once the tentative estimates are computed, it obtains the final green image estimate as a weighted average of the tentative estimates in four directions using the same method as GBTF (see Section 4.1). The extended RI versions such as the minimized-Laplacian residual interpolation (MLRI) [21] and the weighted minimized-Laplacian residual interpolation (MLRI+wei) [22] involve a modification of the guided filter algorithm which minimizes the Laplacian energy. The modified

version of GF is called Minimized-Laplacian Guided Filter (MLGF).

The guided filter is the key to RI algorithms, and it is combined with a residual technique to improve the tentative estimates for the interpolation. The GF and its modified version MLGF will be described in Appendix A.

5.1 RI Green Channel Interpolation

In order to improve the green image interpolation, Kiku et al. proposed in [20] a residual interpolation using GF in the first step of GBTF, but without changing the other steps of the method. Hence the outline of the green interpolation of GBTF and RI is the same except for the first step. Therefore, in this section we will only discuss the first step of the green channel interpolation which is divided into five parts (see Figure 3).

1. Firstly, in order to have a guide image for GF, the horizontal and vertical interpolations of the mosaicked image Φ are computed as

$$\Phi_H = \frac{1}{2}K_H \otimes \Phi = \begin{bmatrix} \frac{1}{2}K_H \otimes \Phi[1] \\ \frac{1}{2}K_H \otimes \Phi[2] \\ \frac{1}{2}K_H \otimes \Phi[2] \\ \frac{1}{2}K_H \otimes \Phi[4] \end{bmatrix},$$

and

$$\Phi_V = \frac{1}{2}K_V \otimes \Phi = \begin{bmatrix} \frac{1}{2}K_V \otimes \Phi[1] \\ \frac{1}{2}K_V \otimes \Phi[2] \\ \frac{1}{2}K_V \otimes \Phi[2] \\ \frac{1}{2}K_V \otimes \Phi[4] \end{bmatrix},$$

with $\Phi[1] = M_R .* R$, $\Phi[2] = M_{GR} .* G$, $\Phi[3] = M_{GB} .* G$ and $\Phi[4] = M_B .* B$.

2. Next, taking Φ_H and Φ_V as guide images for the GF algorithm (see Algorithm 11), the horizontal and vertical tentative estimates $\overline{\Phi}_H$ and $\overline{\Phi}_V$ are computed, i.e.

$$\overline{\Phi}_H = \mathrm{GF}(\mathrm{T_H}(\Phi_H), \Phi_H, M, \widetilde{\Delta}_H, S_H, J, Weight, \epsilon),$$

and

$$\overline{\Phi}_V = \mathrm{GF}(\mathrm{T_V}(\Phi_V), \Phi_V, M, \widetilde{\Delta}_V, S_V, J, Weight, \epsilon).$$

Here, setting $J = MLGF$ corresponds to using MLGF, otherwise it is GF; and when $Weight = True$ the weighted GF/MLGF algorithm is used.

3. The residual values between the original value and tentative estimate on the mask $\mathbf{M}$ are then computed i.e.

$$\mathfrak{R}\overline{\Phi}_H = \mathbf{M} .* (\Phi - \overline{\Phi}_H) = \begin{bmatrix} \mathbf{M}[1] .* (\Phi[1] - \overline{\Phi}_H[1]) \\ \mathbf{M}[2] .* (\Phi[2] - \overline{\Phi}_H[2]) \\ \mathbf{M}[3] .* (\Phi[3] - \overline{\Phi}_H[3]) \\ \mathbf{M}[4] .* (\Phi[4] - \overline{\Phi}_H[4]) \end{bmatrix},$$

and

$$\mathfrak{R}\overline{\Phi}_V = \mathbf{M} .* (\Phi - \overline{\Phi}_V) = \begin{bmatrix} \mathbf{M}[1] .* (\Phi[1] - \overline{\Phi}_V[1]) \\ \mathbf{M}[2] .* (\Phi[2] - \overline{\Phi}_V[2]) \\ \mathbf{M}[3] .* (\Phi[3] - \overline{\Phi}_V[3]) \\ \mathbf{M}[4] .* (\Phi[4] - \overline{\Phi}_V[4]) \end{bmatrix},$$

with $\mathbf{M}[1] = M_R$, $\mathbf{M}[2] = M_{GR}$, $\mathbf{M}[3] = M_{GB}$ and $\mathbf{M}[4] = M_B$.

4. After that, the residuals are interpolated, yielding

$$\mathfrak{R}\overline{\Phi}_H = \frac{1}{2}K_H \otimes \mathfrak{R}\overline{\Phi}_H = \begin{bmatrix} \frac{1}{2}K_H \otimes \mathfrak{R}\overline{\Phi}_H[1] \\ \frac{1}{2}K_H \otimes \mathfrak{R}\overline{\Phi}_H[2] \\ \frac{1}{2}K_H \otimes \mathfrak{R}\overline{\Phi}_H[3] \\ \frac{1}{2}K_H \otimes \mathfrak{R}\overline{\Phi}_H[4] \end{bmatrix},$$

and

$$\mathfrak{R}\overline{\Phi}_V = \frac{1}{2}K_V \otimes \mathfrak{R}\overline{\Phi}_V = \begin{bmatrix} \frac{1}{2}K_V \otimes \mathfrak{R}\overline{\Phi}_V[1] \\ \frac{1}{2}K_V \otimes \mathfrak{R}\overline{\Phi}_V[2] \\ \frac{1}{2}K_V \otimes \mathfrak{R}\overline{\Phi}_V[3] \\ \frac{1}{2}K_V \otimes \mathfrak{R}\overline{\Phi}_V[4] \end{bmatrix}.$$

5. Finally, the interpolated residual image is added to the tentative estimate to obtain the interpolated image

$$\widetilde{\Phi}_H = \mathrm{T_H}(\mathbf{M}).*(\overline{\Phi}_H + \mathfrak{R}\Phi_H) = \begin{bmatrix} \mathbf{M}[2].*(\overline{\Phi}_H + \mathfrak{R}\Phi_H)[1] \\ \mathbf{M}[1].*(\overline{\Phi}_H + \mathfrak{R}\Phi_H)[2] \\ \mathbf{M}[4].*(\overline{\Phi}_H + \mathfrak{R}\Phi_H)[3] \\ \mathbf{M}[3].*(\overline{\Phi}_H + \mathfrak{R}\Phi_H)[4] \end{bmatrix},$$

and

$$\widetilde{\Phi}_V = \mathrm{T_V}(\mathbf{M}).*(\overline{\Phi}_V + \mathfrak{R}\Phi_V) = \begin{bmatrix} \mathbf{M}[3].*(\overline{\Phi}_V + \mathfrak{R}\Phi_V)[1] \\ \mathbf{M}[4].*(\overline{\Phi}_V + \mathfrak{R}\Phi_V)[2] \\ \mathbf{M}[1].*(\overline{\Phi}_V + \mathfrak{R}\Phi_V)[3] \\ \mathbf{M}[2].*(\overline{\Phi}_V + \mathfrak{R}\Phi_V)[4] \end{bmatrix}.$$

The fact that the residual GF tentative estimate is more precise than the HA interpolate is the main contribution of the methods based on residual interpolation. After modifying the first step, Steps (ii)–(iv) are the same as for the GBTF green interpolation. The pseudo-code for all the variants of the RI algorithm is given in Algorithm 5. The only difference between the four types of residual interpolation resides in their use of the guided filter. Hence, setting the parameters for the guided filter determines the different kinds of residual interpolation. These parameter settings are:

- RI: $J = RI$, $Weight = False$;
- RI+wei: $J = RI$, $Weight = True$;
- MLRI: $J = MLRI$, $Weight = False$;
- MLRI+wei: $J = MLRI$, $Weight = True$.

5.2 RI Red and Blue Channel Interpolation

After the G image is interpolated, the GBTF algorithm uses the color difference interpolation for estimating the R and B pixel values as described in Section 4.2. Here, the color difference interpolation is replaced by the residual interpolation as illustrated in Figure 4 (b). We just discuss the red interpolation as the blue interpolation is analogous. The pseudo code is given in Algorithm 6.

The tentative estimate images $\overline{R}$ are obtained by the GF using the green image $\widehat{G}$ as guide, i.e.

$$\overline{R} = \mathrm{GF}(\widehat{G}, M_R.*R, M_R, \widetilde{\Delta}, S, J, Weight, \epsilon),$$

Algorithm 5 Pseudo-code for RI/MLRI green interpolation

Input: Mosaicked image $\Phi = [M_R .* R, M_{GR} .* G, M_{GB} .* G, M_B .* B]^T$,
Mask $\mathbf{M} = [M_R, M_{GR}, M_{GR}, M_B]^T$,
case J: case for algorithm choices $J \in \{RI, MLRI\}$,
case $Weight$: case for weights $Weight \in \{True, False\}$,
regularization ϵ,
directional weight smoothing σ (default 1)

Output: interpolated green $\widehat{G}$

function RI($\Phi, \mathbf{M}, J, Weight, \sigma$)

// Initialization //

$K_H \leftarrow [1, 0, 1]$, $K_V \leftarrow K_H^T$

$\widetilde{\Delta}_H \leftarrow [1, 0-2, 0, 1]$, $\widetilde{\Delta}_V \leftarrow \widetilde{\Delta}_H^T$

$D_H \leftarrow [-1, 0, 1]$, $D_V \leftarrow D_H^T$

$F \leftarrow K_G^9(\sigma)$ // 9×9 Gaussian kernel of standard deviation σ similar to definition (32) //

$S_H = S_V \leftarrow [3, 3]$ **if** $J \neq RI$; **otherwise** $S_H \leftarrow [5, 0]$, $S_V \leftarrow [0, 5]$

$K_E \leftarrow [0, 0, 1]$, $K_W \leftarrow [1, 0, 0]$, $K_S \leftarrow [0, 0, 1]^T$, $K_N \leftarrow [1, 0, 0]^T$ // 5-tap with RI i.e. $K_E \leftarrow [0, 0, 0, 0, 1]$//

// Main loop: H means horizontal interpolation and V vertical interpolation //

for $I \in \{H, V\}$ **do**

// first step: compute the residual GF tentative estimate with five parts//

$\Phi_I \leftarrow \frac{1}{2} K_I \otimes \Phi + \Phi$ // Part 1: horizontal and vertical interpolations //

//Notes for GF algorithm: If $J = MLGF$, it is MLGF, else GF. If $Weight = True$, it is weighed GF/MLGF, else GF/MLGF//

$\overline{\Phi}_I \leftarrow \mathrm{GF}(\mathrm{T_I}(\Phi_I), \Phi_I, M_I, \widetilde{\Delta}_I, S_I, J, Weight, \epsilon)$ //Part 2: generate GF estimate of all channels //

$\mathfrak{R}\overline{\Phi}_I \leftarrow \mathbf{M} .* (\Phi - \overline{\Phi}_I)$ //Part 3: compute the residual values under mask $\mathbf{M}$ //

$\mathfrak{R}\overline{\Phi}_I \leftarrow \frac{1}{2} K_I \otimes \mathfrak{R}\overline{\Phi}_I$ //Part 4: interpolate the residual values //

$\widetilde{\Phi}_I \leftarrow \mathrm{T_I}(\mathbf{M}) .* (\overline{\Phi}_I + \mathfrak{R}\Phi_I)$ //Part 5: add the residual value to GF estimate //

// Second step: compute the color difference//

$\mathfrak{R}\widetilde{\Phi}_I \leftarrow \mathbf{M} .* (\Phi - \mathrm{T_I}(\widetilde{\Phi}_I))$

$\mathfrak{R}_I \leftarrow -\mathfrak{R}\widetilde{\Phi}_I[1] + \mathfrak{R}\widetilde{\Phi}_I[2] + \mathfrak{R}\widetilde{\Phi}_I[3] - \mathfrak{R}\widetilde{\Phi}_I[4]$

// Third step: weighted combine horizontal and vertical color difference//

$\mathfrak{R}_I^D \leftarrow |D_I \otimes \mathfrak{R}_I|$

$\overline{\mathfrak{R}}_I^D \leftarrow F \otimes \mathfrak{R}_I$

end for

$W_E \leftarrow \frac{1}{\left(K_E \otimes \overline{\mathfrak{R}}_H^D\right)^2 + \epsilon}$, $W_W \leftarrow \frac{1}{\left(K_W \otimes \overline{\mathfrak{R}}_H^D\right)^2 + \epsilon}$, $W_S \leftarrow \frac{1}{\left(K_S \otimes \overline{\mathfrak{R}}_V^D\right)^2 + \epsilon}$, $W_N \leftarrow \frac{1}{\left(K_N \otimes \overline{\mathfrak{R}}_V^D\right)^2 + \epsilon}$

$W \leftarrow W_E + W_W + W_S + W_N$

$\mathfrak{R} \leftarrow (W_N .* (f_N \otimes \mathfrak{R}_V) + W_S .* (f_S \otimes \mathfrak{R}_V) + W_W .* (f_W \otimes \mathfrak{R}_H) + W_E .* (f_E \otimes \mathfrak{R}_H)) ./ W$

// Fourth step: final result//

$\widehat{G} \leftarrow \mathbf{M}[1] .* (\mathfrak{R} + \Phi[1]) + \Phi[2] + \Phi[3] + \mathbf{M}[4] .* (\mathfrak{R} + \Phi[4])$

end function

where $\widetilde{\Delta}$ is the sparse Laplacian operator given by

$$\widetilde{\Delta} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -4 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \tag{41}$$

then the residual images are computed and interpolated with

$$\mathfrak{R}\overline{R} = KH \otimes [M_R. * (R - \overline{R})],$$

where

$$KH = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}. \tag{42}$$

Finally, the red interpolated image is obtained by the formula

$$\widehat{R} = \overline{R} + \mathfrak{R}\overline{R}.$$

As before, the only difference between the four kinds of red residual interpolation resides in the use of different kinds of guided filter; the parameters are selected in the same way as for the green residual interpolation.

A note about Line 8 of Algorithm 6: for $(i,j) \in \Omega_R$

$$\begin{aligned} \widehat{R}(i,j) &= \overline{R}(i,j) + < [M_R. * (R - \overline{R})](\mathcal{N}^3_{i,j}), KH > \\ &= \overline{R}(i,j) + < [M_R. * R](\mathcal{N}^3_{i,j}), KH > - < [M_R. * \overline{R}](\mathcal{N}^3_{i,j}), KH > \\ &= \overline{R}(i,j) + R(i,j) - \overline{R}(i,j) \\ &= R(i,j). \end{aligned}$$

The reason of the fourth step in the derivation is that

$$[M_R. * \overline{R}](\mathcal{N}^3_{i,j}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \overline{R}(i,j) & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad (i,j) \in \Omega_R$$

and

$$\left\langle \begin{bmatrix} 0 & 0 & 0 \\ 0 & \overline{R}(i,j) & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} \right\rangle = \overline{R}(i,j).$$

6 Adaptive Residual Interpolation

Adaptive residual interpolation (ARI) [28, 27] improves existing RI-based algorithms by iterating the interpolation step, selecting a suitable iteration number at each pixel and combining RI and MLRI. The algorithm [28] first interpolates the green image and then uses the interpolated green image to interpolate the red and blue images. The algorithm described here applies the improved ARI processing to the red and blue bands, not only the green band as in [28].

Algorithm 6 Pseudo-code for RI/MLRI Red Interpolation

Input: Mosaicked image $M_R.*R$, Mask M_R, green interpolation $\widehat{G}$, case J, case $Weight$
Output: Interpolated red $\widehat{R}$ and blue $\widehat{B}$

function RI_RB($\Phi, \mathbf{M}, \widehat{G}, J, Weight$)
  $KH \leftarrow$ defined by Equation (42)
  $\widetilde{\Delta} \leftarrow$ defined by Equation (41)
  $S \leftarrow [5, 5]$
  //Here if J = MLGF, it is MLGF, else it is GF //
  $\overline{R} \leftarrow \mathrm{GF}(\widehat{G}, M_R.*R, M_R, \widetilde{\Delta}, S, J, Weight, \epsilon)$ //generate GF estimate of red and blue channels //
  $\mathfrak{R}\overline{R} \leftarrow M_R.*(R - \overline{R})$
  $\mathfrak{R}\overline{R} \leftarrow KH \otimes \mathfrak{R}\overline{R}$
  $\widehat{R} \leftarrow \overline{R} + \mathfrak{R}\overline{R}$
end function

6.1 ARI Green Channel Interpolation

The ARI green interpolation algorithm consists of three steps.

1. Four types of green channel interpolation are performed by RI and MLRI in the horizontal and vertical directions respectively.

2. For each interpolation, a suitable iteration number is selected at each pixel according to the smoothness of the pixel.

3. All four interpolated results are combined by a weighted average at each pixel to produce the final G interpolation.

The sequence of the above steps is called ARI green interpolation and its pseudo-code is given in Algorithm 7. It adaptively combines RI and MLRI and selects the number of iterations for each pixel. The overall flow of the horizontal green interpolation at R lines is illustrated in Figure 5. The details of the three steps are shown below.

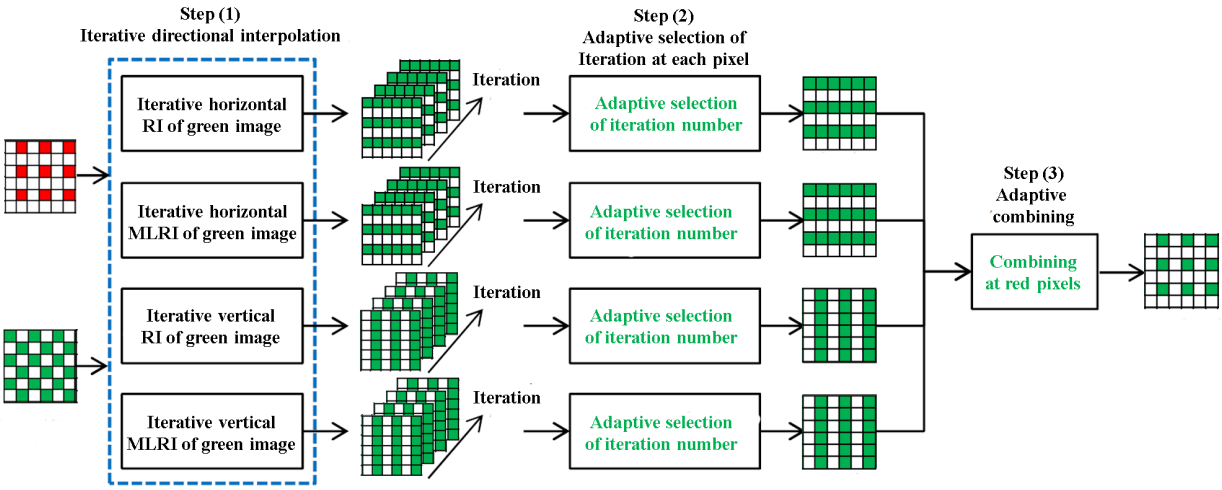


Figure 5: ARI green interpolation at red pixels. The adaptive selection of iteration number is defined by (46).

6.1.1 Step 1: Iterative RI and MLRI Estimates

In the first step, RI and MLRI are applied for iterative directional interpolation. We illustrate the flow of the horizontal green interpolation at the R pixels (See Figure 6).

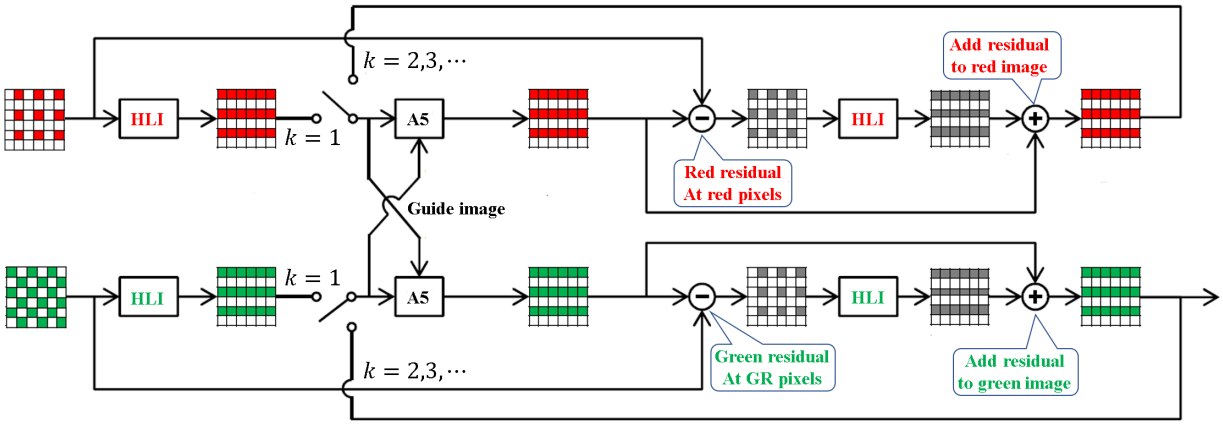


Figure 6: Flow of the iterative horizontal green interpolation at the R pixels by RI or MLRI. A5 means Algorithm 5 and the parameter J determines RI or MLRI algorithm.

First, the mosaicked images are interpolated horizontally and vertically to obtain initial interpolation images and masks as

$$\Phi_I = \frac{1}{2}K_I \otimes \Phi + \Phi = \begin{bmatrix} \frac{1}{2}K_I \otimes \Phi[1] + \Phi[1] \\ \frac{1}{2}K_I \otimes \Phi[2] + \Phi[2] \\ \frac{1}{2}K_I \otimes \Phi[2] + \Phi[3] \\ \frac{1}{2}K_I \otimes \Phi[4] + \Phi[4] \end{bmatrix},$$

$$\mathbf{M}_{I,J} = \frac{1}{2}K_I \otimes \mathbf{M} + \mathbf{M} = \begin{bmatrix} \frac{1}{2}K_I \otimes \mathbf{M}[1] + \mathbf{M}[1] \\ \frac{1}{2}K_I \otimes \mathbf{M}[2] + \mathbf{M}[2] \\ \frac{1}{2}K_I \otimes \mathbf{M}[2] + \mathbf{M}[3] \\ \frac{1}{2}K_I \otimes \mathbf{M}[4] + \mathbf{M}[4] \end{bmatrix},$$

where $I \in \{H, V\}$ and K_I is given by (11) with $I = H$ and (15) with $I = V$. We initialize the estimate $\Phi^0_{I,J}$, $\widehat{G}^0_{I,J}$, $W^0_{I,J}$ as

$$\Phi^0_{I,J} = \Phi_I,$$

$$\widehat{G}^0_{I,J} = \Phi[2]_{I,J} + \Phi[3]_{I,J},$$

and

$$W^0_{I,J}(i,j) = 10^{32},$$

where $I \in \{H, V\}$ and $J \in \{RI, MLRI\}$.

Next, the RI and MLRI tentative estimates are computed by GF (Algorithm 11)

$$\overline{\Phi}^k_{I,J} = \mathrm{GF}(\mathrm{T_I}(\breve{\Phi}^{k-1}_{I,J}), \breve{\Phi}^{k-1}_{I,J}, \mathbf{M}_{I,J}, \widetilde{\Delta}_I, S^{k-1}_{I,J}, J, Weight, \epsilon),$$

where $I \in \{H, V\}$ and $J \in \{RI, MLRI\}$, $\mathrm{T_I}$ is defined by (2) with $I = H$ or (3) with $I = V$, $M_{I,J} = \frac{1}{2}K_I \otimes \mathbf{M}$, $\widetilde{\Delta}_I$ is defined by (12) with $I = H$ or (15) with $I = V$, $S_{I,J}$ is the window

parameters of GF, $Weight \in \{0,1\}$ and ϵ is a constant. For example, if $I = H$ and $J = RI$, then

$$\begin{aligned}\overline{\Phi}^k_{I,J} &= \overline{\Phi}^k_{H,RI} = \mathrm{GF}(\mathrm{T_H}(\breve{\Phi}^{k-1}_{H,RI}), \breve{\Phi}^{k-1}_{H,RI}, M_{H,RI}, \widetilde{\Delta}_H, S^{k-1}_{H,RI}, RI, Weight, \epsilon) \\ &= \mathrm{GF}\left(\begin{bmatrix}\breve{\Phi}^{k-1}_{H,RI}[2]\\ \breve{\Phi}^{k-1}_{H,RI}[1]\\ \breve{\Phi}^{k-1}_{H,RI}[4]\\ \breve{\Phi}^{k-1}_{H,RI}[3]\end{bmatrix}, \begin{bmatrix}\breve{\Phi}^{k-1}_{H,RI}[1]\\ \breve{\Phi}^{k-1}_{H,RI}[2]\\ \breve{\Phi}^{k-1}_{H,RI}[3]\\ \breve{\Phi}^{k-1}_{H,RI}[4]\end{bmatrix}, \begin{bmatrix}M_{H,RI}[1]\\ M_{H,RI}[2]\\ M_{H,RI}[3]\\ M_{H,RI}[4]\end{bmatrix}, \widetilde{\Delta}_H, S^{k-1}_{H,RI}, RI, Weight, \epsilon\right).\end{aligned}$$

$\overline{\Phi}^k_{H,RI}$ means the kth iteration horizontal RI tentative estimate.

After that, the residuals are computed as

$$\mathbf{M}.*\mathfrak{R}\overline{\Phi}^k_{I,J} = \mathbf{M}.*(\Phi - \overline{\Phi}^k_{I,J}) = \begin{bmatrix}\mathbf{M}[1].*(\Phi[1]-\overline{\Phi}^k_{I,J}[1])\\ \mathbf{M}[2].*(\Phi[2]-\overline{\Phi}^k_{I,J}[2])\\ \mathbf{M}[3].*(\Phi[3]-\overline{\Phi}^k_{I,J}[3])\\ \mathbf{M}[4].*(\Phi[4]-\overline{\Phi}^k_{I,J}[4])\end{bmatrix},$$

and then interpolated as

$$\mathfrak{R}\overline{\Phi}^k_{I,J} = \frac{1}{2}K_I \otimes (\mathbf{M}.*\mathfrak{R}\overline{\Phi}^k_{I,J}) = \begin{bmatrix}\frac{1}{2}K_I \otimes \mathfrak{R}\overline{\Phi}^k_{I,J}[1]\\ \frac{1}{2}K_I \otimes \mathfrak{R}\overline{\Phi}^k_{I,J}[2]\\ \frac{1}{2}K_I \otimes \mathfrak{R}\overline{\Phi}^k_{I,J}[3]\\ \frac{1}{2}K_I \otimes \mathfrak{R}\overline{\Phi}^k_{I,J}[4]\end{bmatrix}.$$

Here, for convenience, we do not change the mathematical notation for residuals and interpolated residuals. In a word, $\mathfrak{R}\overline{\Phi}^k_{I,J} = \frac{1}{2}K_I \otimes [\mathbf{M}.*(\Phi - \overline{\Phi}^k_{I,J})]$.

Finally, to obtain the k-th horizontally interpolated results, the interpolated residuals are added to the tentative estimates, i.e.

$$\breve{\Phi}^k_{I,J} = \overline{\Phi}^k_{I,J} + \mathfrak{R}\overline{\Phi}^k_{I,J} = \begin{bmatrix}\overline{\Phi}^k_{I,J}[1] + \mathfrak{R}\overline{\Phi}^k_{I,J}[1]\\ \overline{\Phi}^k_{I,J}[2] + \mathfrak{R}\overline{\Phi}^k_{I,J}[2]\\ \overline{\Phi}^k_{I,J}[3] + \mathfrak{R}\overline{\Phi}^k_{I,J}[3]\\ \overline{\Phi}^k_{I,J}[4] + \mathfrak{R}\overline{\Phi}^k_{I,J}[4]\end{bmatrix}.$$

In the end, the value of the mask $\mathbf{M}$ pixels of $\breve{\Phi}^k_{I,J}$ is replaced by original one, i.e.

$$\breve{\Phi}^k_{I,J} = \mathbf{M}.*\Phi + \mathrm{T_I}(\mathbf{M}).*\breve{\Phi}^k_{I,J},$$

then the G interpolation is given as

$$G^k_{I,J} = \breve{\Phi}^k_{I,J}[2] + \breve{\Phi}^k_{I,J}[3].$$

6.1.2 Step 2: Adaptive Selection of Iteration Number

In this step, for each channel interpolation, a suitable iteration number is adaptively selected at each pixel by a criterion. The criterion is defined in a pixel-by-pixel manner based on the following residuals.

$$\mathfrak{R}\Phi^k_{I,J} = \mathbf{M}.*(\breve{\Phi}^{k-1}_{I,J} - \overline{\Phi}^k_{I,J}), \tag{43}$$

where $\mathfrak{R}\Phi^k_{I,J}$ represents the difference between the k-th tentative estimates and the previous interpolation results. These differences assess how effectively the tentative estimates converge at the k-th iteration. First the directional gradients are computed as

$$\mathfrak{R}^D\Phi^k_{I,J} = \mathbf{M}. * (|D \otimes \mathfrak{R}\Phi^k_{I,J}|), \tag{44}$$

then the criterion value for a pixel (i, j) at the k-th iteration is defined based on the magnitude and the smoothness of $\mathfrak{R}\Phi^k_{I,J}$ as

$$W^k_{I,J} = (< \mathfrak{R}^k_{I,J}(\mathcal{N}^d_{i,j}), K^5_G >)^2 . * < \mathfrak{R}^{D,k}_{I,J}, K^5_G >, \tag{45}$$

where

$$\mathfrak{R}^k_{I,J} = \mathfrak{R}\Phi^k_{I,J}[1] + \mathfrak{R}\Phi^k_{I,J}[2] + \mathfrak{R}\Phi^k_{I,J}[3] + \mathfrak{R}\Phi^k_{I,J}[4],$$
$$\mathfrak{R}^{D,k}_{I,J} = \mathfrak{R}^D\Phi^k_{I,J}[1] + \mathfrak{R}^D\Phi^k_{I,J}[2] + \mathfrak{R}^D\Phi^k_{I,J}[3] + \mathfrak{R}^D\Phi^k_{I,J}[4].$$

and K^5_G represents a spatial Gaussian kernel given by (32). The above criterion value becomes small if the residuals are small and smooth. Equations (43)–(45) are combined in an algorithm which is called Residual Interpolation Weights (RIW) (see Algorithm 8).

Based on criterion values corresponding to each pixel, the suitable iteration number $K_{I,J}(i, j)$ is adaptively selected at each pixel (i, j),

$$K_{I,J}(i, j) = \arg\min_k W^k_{I,J}(i, j). \tag{46}$$

The parameter for the maximum iteration number is empirically set to $itMax = 11$. But note that the value of $K_{I,J}(i, j)$ is not needed; the important thing is to find the green image estimate at the $K_{I,J}(i, j)$-th iteration, so one can proceed as follows

$$\widehat{G}^k_{I,J}(i, j) = \begin{cases} G^k_{I,J}(i, j), & \text{if } W^k_{I,J}(i, j) < \widehat{W}^{k-1}_{I,J}(i, j); \\ \widehat{G}^{k-1}_{I,J}(i, j), & \text{if } W^k_{I,J}(i, j) \geq \widehat{W}^{k-1}_{I,J}(i, j), \end{cases}$$

and

$$\widehat{W}^k_{I,J}(i, j) = \begin{cases} W^k_{I,J}(i, j), & \text{if } W^k_{I,J}(i, j) < \widehat{W}^{k-1}_{I,J}(i, j); \\ \widehat{W}^{k-1}_{I,J}(i, j), & \text{if } W^k_{I,J}(i, j) \geq \widehat{W}^{k-1}_{I,J}(i, j). \end{cases}$$

This means that if the criterion of the kth iteration $W^k_{I,J}(i, j)$ is smaller than the one of the $k-1$th iteration $\widehat{W}^{k-1}_{I,J}(i, j)$, one updates the tentative estimate $\widehat{G}_{I,J}(i, j)$ and the criterion $\widehat{W}_{I,J}(i, j)$, otherwise they are kept unchanged. In this way, the minimum criterion $\widehat{W}_{I,J}(i, j)$ and the estimate $\widehat{G}_{I,J}(i, j)$ at the iteration of minimum criterion $\widehat{W}_{I,J}(i, j)$ are maintained at each pixel (i, j) and for each kind of interpolation.

6.1.3 Step 3: Weighted Combination of Four Interpolated Results

In this step, two directional interpolation results of RI and MLRI are combined by the weighted averaging as

$$\widehat{G}(i, j) = \frac{\sum_{J\in\{RI,MLRI\}} \sum_{I\in\{H,V\}} \widehat{G}^{\text{itMax}}_{I,J}(i, j)/W^{\text{itMax}}_{I,J}(i, j)}{\sum_{J\in\{RI,MLRI\}} \sum_{I\in\{P,N\}} 1./W^{\text{itMax}}_{I,J}(i, j)},$$

where $\widehat{G}^{\text{itMax}}_{I,J}$ and $W^{\text{itMax}}_{I,J}(i, j)$ $J \in \{RI, MLRI\}$ and $I \in \{H, V\}$ represent the horizontal or the vertical direction interpolation and criterion of RI or MLRI respectively.

Algorithm 7 Pseudo-code for ARI green interpolation

Input: Mosaicked image Φ, Mask $\mathbf{M}$, itMax, regularization ϵ
Output: interpolated green $\widehat{G}$

function ARIG($\Phi, \mathbf{M}, \epsilon$)
  $K_H \leftarrow [1,0,1]$, $K_V \leftarrow K_H^T$
  $\widetilde{\Delta}_H \leftarrow [-1,0,2,0,-1]$, $\widetilde{\Delta}_V \leftarrow \widetilde{\Delta}_H^T$
  $D_H \leftarrow [-1,0,1]$, $D_V \leftarrow D_H^T$
  $\{S_{H,R}, S_{V,R}, S_{H,L}, S_{V,L}\} \leftarrow \{[2,1],[1,2],[4,0],[0,4]\}$// GF initial support sizes//
  // prepare interpolation images //
  for $I \in \{H,V\}$ **do**
    for $J \in \{RI, MLRI\}$ **do**
      $\Phi_{I,J} \leftarrow \frac{1}{2}K_I \otimes \Phi + \Phi$
      $\mathbf{M}_{I,J} \leftarrow \frac{1}{2}K_I \otimes \mathbf{M} + \mathbf{M}$
      $\widehat{G}_{I,J} \leftarrow \Phi[2]_{I,J} + \Phi[3]_{I,J}$
      $W_{I,J} \leftarrow W_{I,J}(i,j) = 10^{32}$
    end for
  end for
  // main iteration //
  for k from 1 to itMax **do**
    for $I \in \{H,V\}$ **do**
      for $J \in \{RI, MLRI\}$ **do**
        $\overline{\Phi}_{I,J}^k \leftarrow \mathrm{GF}(\mathrm{T_I}(\breve{\Phi}_{I,J}^{k-1}), \breve{\Phi}_{I,J}^{k-1}, M_{I,J}, \widetilde{\Delta}_I, S_{I,J}, J, Weight, \epsilon)$ // GF interp. of all channels //
        $\mathfrak{R}\overline{\Phi}_{I,J}^k \leftarrow \mathbf{M}. * (\Phi - \overline{\Phi}_{I,J}^k)$ // residual value at mask $\mathbf{M}$ pixels //
        $\mathfrak{R}\overline{\Phi}_{I,J}^k \leftarrow \frac{1}{2}K_I \otimes \mathfrak{R}\overline{\Phi}_{I,J}^k$ // interpolate masked pixels //
        $\breve{\Phi}_{I,J}^k \leftarrow \overline{\Phi}_{I,J}^k + \mathfrak{R}\overline{\Phi}_{I,J}^k$ // residual interpolation estimate //
        $W_{I,J}^k \leftarrow \mathrm{RIW}(\mathbf{M}_{I,J}, \breve{\Phi}_{I,J}^{k-1}, \overline{\Phi}_{I,J}^k, D_I)$ // compute the weights by Algorithm 9 //
        // updating //
        $\breve{\Phi}_{I,J}^k \leftarrow \mathbf{M}. * \Phi + \mathrm{T_I}(\mathbf{M}). * \breve{\Phi}_{I,J}^k$ // keep " original " pixels of Φ //
        $G_{I,J}^k \leftarrow \breve{\Phi}^k[2]_{I,J} + \breve{\Phi}^k[3]_{I,J}$
        $\widehat{G}_{I,J}^k \leftarrow \widehat{G}_{I,J}^{k-1}$ **when** $W_{I,J}^k(i,j) \geq \widehat{W}_{I,J}^{k-1}(i,j)$,
          $G_{I,J}^k$ **otherwise**
        $\widehat{W}_{I,J}^k \leftarrow \widehat{W}_{I,J}^{k-1}$ **when** $W_{I,J}^k(i,j) \geq \widehat{W}_{I,J}^{k-1}(i,j)$,
          $W_{I,J}^k$ **otherwise**
        $S_{I,J}^k \leftarrow S_{I,J}^{k-1} + [1,1]$ // increase GF interpolation support size //
      end for
    end for
  end for
  $\widehat{G} \leftarrow \left(\sum_{J\in\{RI,MLRI\}} \sum_{I\in\{H,V\}} \widehat{G}_{I,J}^{\mathrm{itMax}}./\widehat{W}_{I,J}^{\mathrm{itMax}}\right)./\left(\sum_{J\in\{RI,MLRI\}} \sum_{I\in\{H,V\}} 1./\widehat{W}_{I,J}^{\mathrm{itMax}}\right)$ //weighted mean of all estimates //
  $\widehat{G} \leftarrow \Phi[2] + \Phi[3] + (\mathbf{M}[1] + \mathbf{M}[4]). * \widehat{G}$ // keep " original " value at green pixels//
end function

Algorithm 8 Pseudo-code for Residual Interpolation Weights

Input: $\mathbf{M}, \Phi, \overline{\Phi}, D$
Output: W
function RIW$(\mathbf{M}, \Phi, \overline{\Phi}, D)$
  $K_G^5 \leftarrow$ defined by (32)
  $K \leftarrow$ the number of matrices in the tensor Φ
  $\mathfrak{R}\Phi \leftarrow \mathbf{M}. * (\Phi - \overline{\Phi})$ // residual value at mask **M** pixels //
  $\mathfrak{R}^D\Phi \leftarrow \mathbf{M}. * (|D \otimes \mathfrak{R}\Phi|)$ //absolute Differential of residual value at mask **M** pixels //
  $\mathfrak{R} \leftarrow \sum_{k=1}^{K} |\mathfrak{R}\Phi[k]|$ // sum of K channels of absolute residual value //
  $\mathfrak{R}^D \leftarrow \sum_{k=1}^{K} \mathfrak{R}^D\Phi[k]$ // sum of K channels of absolute Differential residual value //
  $\overline{\mathfrak{R}} \leftarrow K_G^5 \otimes \mathfrak{R}$ // Gaussian smooth of absolute residual value//
  $\overline{\mathfrak{R}^D} \leftarrow K_G^5 \otimes \mathfrak{R}^D$ // Gaussian smooth of absolute residual value//
  $W = \left[\overline{\mathfrak{R}}. * \overline{\mathfrak{R}}\right] . * \overline{\mathfrak{R}^D}$
end function

6.2 ARI Red and Blue Channel Interpolation

The ARI red and blue channel interpolation is divided into two steps:

1. The R values at the B pixels are interpolated using ARI along the diagonal directions. The B values at the R pixels are handled in the same manner.

2. The R values at the G pixels are interpolated using ARI along the horizontal and vertical directions.

The interpolated $\widehat{G}$ image is fixed throughout the process and is used as the guide image in the GF algorithm. The flow of ARI red interpolation is illustrated in Figure 7 and blue interpolation is done in the same manner. In the experiments, the maximum iteration number is set to 2. The Steps 1 and 2 of the red channel interpolation are given by Algorithms 9 and 10, respectively. The two steps are almost the same as for the green interpolation. Therefore, in the next subsection, we just give the details of Step 1.

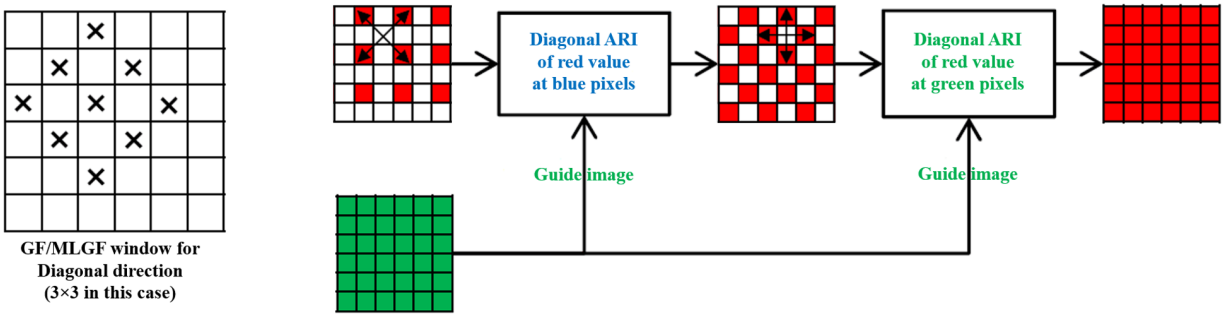


Figure 7: Flow of the ARI red interpolation. A window of GF for the diagonal direction is shown on the left.

6.2.1 Step 1: Diagonal Directions Interpolations

The initial value is computed as

$$R_{I,J}^0 = \frac{1}{2} K_I \otimes (M_R. * R) + M_R. * R,$$

Algorithm 9 Pseudo-code for ARI red interpolations 1

Input: mosaicked image Φ, Mask $\mathbf{M}$, green estimate $\widehat{G}$, itMax, regularization ϵ
Output: Interpolated red $\widetilde{R}$

function ARI_RB1($\Phi, \widehat{G}, \mathbf{M}, \epsilon$)
  K_P and $K_N \leftarrow$ defined by (24) and (25) respectively
  $\widetilde{\Delta}_P$ and $\widetilde{\Delta}_N \leftarrow$ defined by (24) and (25) respectively
  $M_{IG} \leftarrow M_R + M_B$
  $\{S_{1,R}, S_{2,R}, S_{1,L}, S_{2,L}\} \leftarrow \{[2,2],[2,2],[2,0],[0,2]\}$
  for $I \in \{P, N\}$ **do**
    for $J \in \{R, L\}$ **do**
      $R_{I,J} \leftarrow \frac{1}{2}K_I \otimes (M_R. * R) + M_R. * R$
      $\widehat{W}_{I,J} \leftarrow W_{I,J}(i,j) = 10^{32}$
    end for
  end for
  while k from 1 to itMax **do**
    for $I \in \{P, N\}$ **do**
      for $J \in \{R, L\}$ **do**
        $\overline{R}^k_{I,J} \leftarrow \mathrm{GF}(\widehat{G}, \breve{R}^{k-1}_{I,J}, M_{IG}, \widetilde{\Delta}_I, S^{k-1}_{I,J}, J, Weight, \epsilon)$.
        $\mathfrak{R}\overline{R}^k_{I,J} \leftarrow M_R. * (M_R. * (R - \overline{R}^k_{I,J}))$
        $\mathfrak{R}\overline{R}^k_{I,J} \leftarrow \frac{1}{2}K_I \otimes \mathfrak{R}\overline{R}^k_{I,J}$
        $\widehat{R}^k_{I,J} \leftarrow M_B. * (\overline{R}^k_{I,J} + \mathfrak{R}\overline{R}^k_{I,J})$
        $W^k_{I,J} \leftarrow \mathrm{RIW}(M_{IG}, \breve{R}^{k-1}_{I,J}, \overline{R}^k_{I,J}, D_I)$ // compute the weights by Algorithm 9 // *// updating //*
        $\widehat{R}^k_{I,J} \leftarrow M_R. * R + M_B. * \widehat{R}^k_{I,J}$ // keep "original" pixels of $M_R. * R$ //
        $\breve{R}^k_{I,J}(i,j) \leftarrow \breve{R}^{k-1}_{I,J}(i,j)$ **when** $W^k_{I,J}(i,j) \geq \widehat{W}^{k-1}_{I,J}(i,j)$,
                $\widehat{R}^k_{I,J}(i,j)$ **otherwise**
        $\widehat{W}^k_{I,J}(i,j) \leftarrow \widehat{W}^{k-1}_{I,J}(i,j)$ **when** $W^k_{I,J}(i,j) \geq \widehat{W}^{k-1}_{I,J}(i,j)$,
                $W^k_{I,J}(i,j)$ **otherwise**
        $S^k_{I,J} \leftarrow S^{k-1}_{I,J} + [1,1]$ // increase GF interpolation support size //
      end for
    end for
  end while
  $\breve{R} \leftarrow \left(\sum_{J\in\{RI,MLRI\}} \sum_{I\in\{P,N\}} \breve{R}^{\mathrm{itMax}}_{I,J} / \widehat{W}^{\mathrm{itMax}}_{I,J} \right) / \left(\sum_{J\in\{RI,MLRI\}} \sum_{I\in\{P,N\}} 1 / \widehat{W}^{\mathrm{itMax}}_{I,J} \right)$
  $\widetilde{R} \leftarrow M_R. * R + M_B. * \breve{R}$
end function

Algorithm 10 Pseudo-code for ARI Red interpolations 2

Input: Mask $\mathbf{M}$, Interpolated images $\widehat{G}$, $\widetilde{R}$, itMax, regularization ϵ
Output: Interpolated red $\widehat{R}$

function ARI_RB2($\Phi, \widehat{G}, \mathbf{M}, \epsilon$)
  $K_H \leftarrow [1, 0, 1]$, $K_V \leftarrow K_H^T$
  $\widetilde{\Delta}_H \leftarrow [-1, 02, 0, -1]$, $\widetilde{\Delta}_V \leftarrow \widetilde{\Delta}_H^T$
  $\{S_{H,R}, S_{V,R}, S_{H,L}, S_{V,L}\} \leftarrow \{[2,2], [2,2], [2,0], [0,2]\}$
  $M_{IG} \leftarrow M[1] + M[4]$
  for $I \in \{H, V\}$ **do**
    for $J \in \{R, L\}$ **do**
      $\widetilde{R}_{I,J} \leftarrow \frac{1}{2} K_I \otimes \widetilde{R} + \widetilde{R}$
      $\widehat{W}_{I,J} = \{\widehat{W}_{I,J}[1], \widehat{W}_{I,J}[2]\} \leftarrow \widehat{W}_{I,J}[k,i,j] = 10^{32}$
    end for
  end for
  while k from 1 to itMax **do**
    for $I \in \{H, V\}$ **do**
      for $J \in \{R, L\}$ **do**
        $\overline{R}^k_{I,J} \leftarrow \mathrm{GF}(\widehat{G}, \breve{R}^{k-1}_{I,J}, M_{IG}, \widetilde{\Delta}_I, S^{k-1}_{I,J}, J, Weight, \epsilon)$
        $\mathfrak{R}\overline{R}^k_{I,J} \leftarrow M_{IG}. * (\widetilde{R} - \overline{R}^k_{I,J})$
        $\mathfrak{R}\overline{R}^k_{I,J} \leftarrow \frac{1}{2} K_I \otimes \mathfrak{R}\overline{R}^k_{I,J}$
        $\widehat{R}^k_{I,J} \leftarrow M_{IG}. * (\overline{R}^k_{I,J} + \mathfrak{R}\overline{R}^k_{I,J})$
        $W^k_{I,J} \leftarrow \mathrm{RIW}(M_{IG}, \breve{R}^{k-1}_{I,J}, \overline{R}^k_{I,J}, D_I)$ // Computed residual wights // *// updating //*
        $\widehat{R}^k_{I,J} \leftarrow M_{IG}. * \widetilde{R} + M_G. * \widehat{R}^k_{I,J}$ // keep "original" pixels of $\widetilde{R}$ //
        $\breve{R}^k_{I,J} \leftarrow \breve{R}^{k-1}_{I,J}$ **when** $W^k_{I,J}(i,j) \geq \widehat{W}^k_{I,J}(i,j)$,
          $\widehat{R}^k_{I,J}$ **otherwise**
        $\widehat{W}^k_{I,J} \leftarrow \widehat{W}^{k-1}_{I,J}$ **when** $W^k_{I,J}(i,j) \geq \widehat{W}^{k-1}_{I,J}(i,j)$,
          $W^k_{I,J}$ **otherwise**
        $S^k_{I,J} \leftarrow S^{k-1}_{I,J} + [1,1]$ // increase GF interpolation support size //
      end for
    end for
  end while
  $\breve{R} \leftarrow \left(\sum_{J \in \{RI, MLRI\}} \sum_{I \in \{H,V\}} \breve{R}^{\mathrm{itMax}}_{I,J} / \widehat{W}^{\mathrm{itMax}}_{I,J} \right) / \left(\sum_{J \in \{RI, MLRI\}} \sum_{I \in \{H,V\}} 1 / \widehat{W}^{\mathrm{itMax}}_{I,J} \right)$
  $\widehat{R} \leftarrow M_{IG}. * \widetilde{R} + M_G. * \breve{R}$
end function

and the initial weights as

$$W^0_{I,J}(i,j) = 10^{32},$$

where $I \in \{P, N\}$ and $J \in \{RI, MLRI\}$, K_P and K_N are defined by (24) and (25) respectively, $k \in \{1, 2\}$, $(i, j) \in \Omega$.

In the iteration, the RI and MLRI tentative estimates are computed by GF (Algorithm 11) with $\widehat{G}$ as guide image, i.e.

$$\overline{R}^k_{I,J} = \mathrm{GF}(\widehat{G}, \breve{R}^{k-1}_{I,J}, M_{IG}, \widetilde{\Delta}_I, S^{k-1}_{I,J}, J, Weight, \epsilon).$$

Next, the residuals are computed and interpolated

$$\mathfrak{R}\overline{R}^k_{I,J} = \frac{1}{2} K_I \otimes (M_{R.} * (R - \overline{R}^k_{I,J})).$$

Lastly, the interpolated residuals are added to the tentative estimates

$$\widehat{R}^k_{I,J} = M_{B.} * (\overline{R}^k_{I,J} + \mathfrak{R}\overline{R}^k_{I,J}).$$

The criterion $W^k_{I,J}$ is computed as $\mathrm{RIW}(M_{IG}, \breve{R}^{k-1}_{I,J}, \overline{R}^k_{I,J}, D_I)$. The results are then updated as

$$\widehat{R}^k_{I,J} = M_{R.} * R + M_{B.} * \widehat{R}^k_{I,J},$$

where

$$\breve{R}^k_{I,J}(i,j) = \begin{cases} \widehat{R}^k_{I,J}(i,j), & \text{if } W^k_{I,J}(i,j) < \widehat{W}^{k-1}_{I,J}(i,j); \\ \breve{R}^{k-1}_{I,J}(i,j), & \text{if } W^k_{I,J}(i,j) \geq \widehat{W}^{k-1}_{I,J}(i,j), \end{cases}$$

and

$$\widehat{W}^k_{I,J}(i,j) = \begin{cases} W^k_{I,J}(i,j), & \text{if } W^k_{I,J}(i,j) < \widehat{W}^{k-1}_{I,J}(i,j); \\ \widehat{W}^{k-1}_{I,J}(i,j), & \text{if } W^k_{I,J}(i,j) \geq \widehat{W}^{k-1}_{I,J}(i,j), \end{cases}$$

At the end of the iterations, the two directional interpolation results of RI and MLRI are combined by the weighted average

$$\breve{R}(i,j) = \frac{\sum\limits_{J \in \{RI, MLRI\}} \sum\limits_{I \in \{H,V\}} \breve{R}^{\mathrm{itMax}}_{I,J}(i,j) / \widehat{W}^{\mathrm{itMax}}_{I,J}(i,j)}{\sum\limits_{J \in \{RI, MLRI\}} \sum\limits_{I \in \{H,V\}} 1./\widehat{W}^{\mathrm{itMax}}_{I,J}(i,j)},$$

where $\breve{R}^{\mathrm{itMax}}_{I,J}$ and $\widehat{W}^{\mathrm{itMax}}_{I,J}(i,j)$, $J \in \{RI, MLRI\}$ and $I \in \{P, N\}$ represent the horizontal or the vertical direction interpolation and criterion of RI or MLRI respectively. The interpolated results are given by

$$\widetilde{R} = M_{R.} * R + M_{B.} * \breve{R}.$$

7 Experimental Results

7.1 Image Datasets and Peak Signal-to-noise Ratio

To evaluate the demosaicking algorithms which we analyzed in the paper, we implemented these algorithms using python and used four full color image datasets: the Kodak image dataset[1], the

[1]Kodak lossless true color image suite, http://r0k.us/graphics/kodak/

McMaster image dataset [41], the Microsoft Demosaicing Canon Dataset [19] and the Microsoft Demosaicing Panasonic Dataset [19]. The Kodak dataset consists of 24 full-color noiseless images and each image is either 768×512 or 512×768 in size. It was downsampled from 3072×2048 or 2048×3072 high-resolution images. The McMaster image dataset consists of 18 images all of size 500×500, cropped from original high-resolution images (size: 2310×1814). The Kodak image dataset and the McMaster image dataset are widely used as standard datasets for demosaicking and many other color image processing fields. The Microsoft Demosaicing Dataset Canon contains 57 images and Microsoft Demosaicing Dataset Panasonic includes 500 images and all images have size 210×318. The raw images were taken by two different cameras: a Canon EOS 550D and a Panasonic Lumix DMC-LX3.

The peak signal-to-noise ratio (PSNR) or color peak signal-to-noise ratio (CPSNR) [3] is preferred as a logarithmic measure of the algorithmic performance. The CPSNR is defined as

$$\mathrm{CPSNR} = 10 \log_{10} \frac{255^2}{\sum_{X=R,G,B} \mathrm{MSE}(X)/3},$$

with

$$\mathrm{MSE}(X) = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} (\widehat{X}(i,j) - X(i,j))^2,$$

where X is the ground truth image and $\widehat{X}$ the estimated image. The larger the CPSNR value between the original images and their respective estimated versions, the better the algorithm.

7.2 Comparison

To better analyze the six algorithms described in this paper, we compared them with the state-of-the-art algorithms LSSC [26], RCNN [34] and JCNN [14], which are all learning-based methods. All four image datasets were treated with the nine demosaicking algorithms. Six interpolation-based algorithms which are analyzed in this paper HA [15, 1], GBTF [30], RI [20], MLRI [21], MLRI+wei [22], ARI [28, 27], and three learning-based algorithms LSSC [26], RCNN [34], JCNN [14]. The experiments with LSSC were done using the binaries provided by the authors[2], while the experiments with RCNN and JCNN were reproduced using the implementations in the IPOL publication [12]. In order to test the robustness of these algorithms, we also added white Gaussian noise with several levels to the mosaicked images. The standard deviation σ measuring the noise level was fixed to 1, 3, and 10.

Tables 1–3 show the average PSNR values on the image datasets of various algorithms for the green band, the red band (the blue being similar), and the CPSNR of the full color images. The best values of all methods are marked in **bold** font and the best values of all interpolation-based methods are marked in red. They demonstrate similar behaviour for the distribution of PSNRs of R, G, B and for the CPSNR.

We can see that among interpolation-based methods, ARI is always the best. For demosaicking without noise, ARI is slightly better than LSSC, but not as good as deep learning methods like RCNN and JCNN. We can also observe that the interpolation-based algorithms are robust to noise. The higher the noise level, the smaller the gap between deep learning and interpolation based methods. With larger noise, ARI obtains the best average PSNR for the four image datasets for the methods without denoising function. The tables also show that GBTF, RI, MLRI and MLRI+wei all improve over the HA interpolation, and that the difference between them is actually very small. Their average difference in CPSNR for the four image datasets is less than 0.25db. In the case of noisy mosaicked images, we found that the joint denoising and demosaicking method (JCNN) is not superior to the

[2] https://lear.inrialpes.fr/people/mairal/resources/demosaicking_ICCV09.tar.gz

Table 1: Comparison of the demosaicking performance of nine algorithms, measured in average PSNR(db) for the standard Kodak, IMAX, MSR_canon and MSR_panasonic: Green band.

Algorithm	HA	GBTF	RI	MLRI	MLRI +wei	ARI	LSSC	RCNN	JCNN
Kodak ($\sigma = 0$)	38.28	41.19	40.95	40.81	41.35	42.36	44.30	44.75	**45.03**
Kodak ($\sigma = 1$)	37.76	40.44	40.18	40.06	40.53	41.38	42.92	**43.26**	42.05
Kodak ($\sigma = 3$)	35.34	37.22	36.98	36.93	37.20	37.59	38.12	38.31	**39.27**
Kodak ($\sigma = 10$)	28.16	29.15	29.09	29.07	29.17	29.22	28.92	28.60	**33.74**
Imax ($\sigma = 0$)	38.05	37.84	39.95	39.98	40.16	40.63	38.80	42.04	**42.12**
Imax ($\sigma = 1$)	37.56	37.45	39.33	39.36	39.53	39.93	38.27	**41.06**	40.03
Imax ($\sigma = 3$)	35.25	35.41	36.54	36.58	36.70	36.88	35.72	37.27	**38.41**
Imax ($\sigma = 10$)	28.34	28.88	29.20	29.21	29.28	29.34	28.34	28.68	**33.99**
Canon ($\sigma = 0$)	40.18	43.36	42.79	42.77	43.00	44.80	44.77	46.73	**46.73**
Canon ($\sigma = 1$)	39.20	41.94	41.40	41.41	41.60	42.82	43.17	**44.44**	43.70
Canon ($\sigma = 3$)	36.02	37.83	37.50	37.53	37.66	38.16	38.15	38.70	**40.88**
Canon ($\sigma = 10$)	28.88	29.82	29.81	29.84	29.92	30.02	29.40	29.19	**35.58**
Panasonic ($\sigma = 0$)	37.71	40.64	40.16	40.13	40.39	41.95	42.25	44.21	**44.25**
Panasonic ($\sigma = 1$)	37.11	39.78	39.32	39.31	39.54	40.77	41.22	**42.69**	41.90
Panasonic ($\sigma = 3$)	34.76	36.69	36.34	36.35	36.52	37.09	37.24	37.94	**39.40**
Panasonic ($\sigma = 10$)	28.20	29.23	29.14	29.15	29.24	29.33	28.87	28.72	**34.18**
Average ($\sigma = 0$)	38.55	40.76	40.96	40.92	41.23	42.44	42.53	44.43	**44.53**
Average ($\sigma = 1$)	37.91	39.90	40.06	40.04	40.30	41.23	41.40	**42.86**	41.92
Average ($\sigma = 3$)	35.34	36.79	36.84	36.85	37.02	37.43	37.31	38.05	**39.49**
Average ($\sigma = 10$)	28.39	29.27	29.31	29.32	29.40	29.48	28.88	28.80	**34.37**

demosaicking methods when the noise level $\sigma \leq 1$, while for $\sigma = 3$, and 10, the advantage of JCNN is obvious.

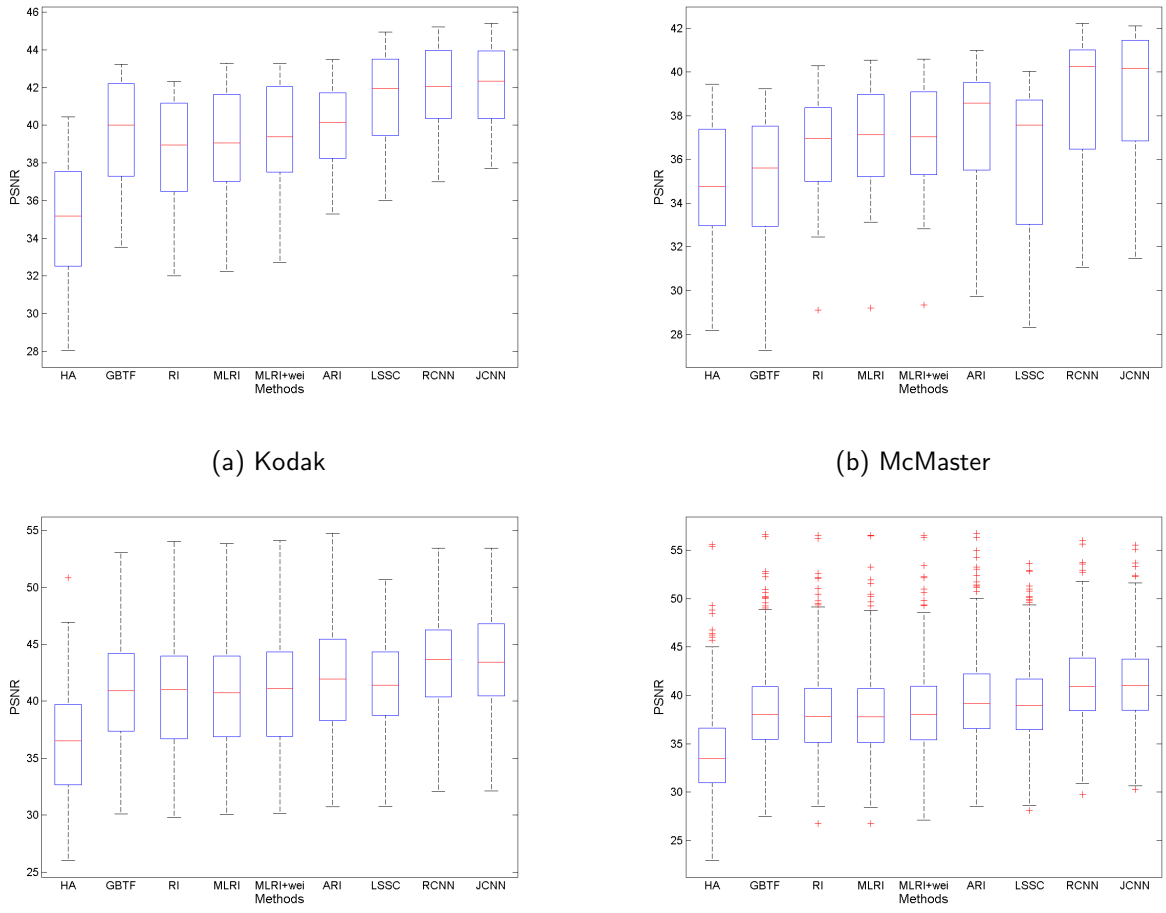
Table 2: Comparison of the demosaicking performance of nine algorithms, measured in average PSNR(db) for the standard Kodak, IMAX, MSR_canon and MSR_panasonic: Red band.

Algorithm	HA	GBTF	RI	MLRI	MLRI +wei	ARI	LSSC	RCNN	JCNN
Kodak ($\sigma = 0$)	33.86	38.98	37.91	38.30	38.72	39.23	40.52	41.31	**41.38**
Kodak ($\sigma = 1$)	33.69	38.44	37.48	37.82	38.19	38.68	39.90	**40.53**	39.49
Kodak ($\sigma = 3$)	32.59	35.80	35.27	35.38	35.58	36.06	36.70	37.05	**37.65**
Kodak ($\sigma = 10$)	27.54	28.22	28.43	28.01	27.99	28.83	28.51	28.32	**33.00**
Imax ($\sigma = 0$)	34.51	34.58	36.08	36.56	36.60	37.42	36.01	39.09	**39.34**
Imax ($\sigma = 1$)	34.32	34.36	35.75	36.23	36.25	37.00	35.73	**38.55**	37.53
Imax ($\sigma = 3$)	33.12	33.04	34.03	34.35	34.36	34.97	34.04	35.89	**36.50**
Imax ($\sigma = 10$)	27.89	27.57	28.18	27.93	27.87	28.75	28.21	28.23	**32.92**
Canon ($\sigma = 0$)	35.49	40.31	40.13	40.09	40.29	41.37	40.47	42.51	**42.66**
Canon ($\sigma = 1$)	35.16	39.39	39.17	39.14	39.30	40.16	39.91	**41.39**	40.51
Canon ($\sigma = 3$)	33.58	36.18	36.02	35.90	35.99	36.64	36.67	37.33	**38.70**
Canon ($\sigma = 10$)	28.31	28.84	29.11	28.55	28.50	29.67	29.63	28.85	**34.35**
Panasonic ($\sigma = 0$)	32.75	37.31	37.12	37.12	37.33	38.38	37.89	39.93	**39.97**
Panasonic ($\sigma = 1$)	32.54	36.75	36.58	36.58	36.75	37.68	37.49	**39.20**	38.28
Panasonic ($\sigma = 3$)	31.51	34.52	34.41	34.36	34.49	35.13	35.10	36.11	**36.82**
Panasonic ($\sigma = 10$)	27.29	28.12	28.30	27.94	27.93	28.72	28.64	28.28	**32.84**
Average ($\sigma = 0$)	34.15	37.79	37.81	38.02	38.23	39.10	38.72	40.71	**40.84**
Average ($\sigma = 1$)	33.93	37.23	37.24	37.44	37.62	38.38	38.26	**39.92**	38.95
Average ($\sigma = 3$)	32.70	34.88	34.93	35.00	35.10	35.70	35.63	36.60	**37.42**
Average ($\sigma = 10$)	27.76	28.19	28.50	28.11	28.07	28.99	28.75	28.42	**33.28**

Table 3: Comparison of the demosaicking performance of nine algorithms, measured in average CPSNR(db) for the standard Kodak, IMAX, MSR_canon and MSR_panasonic for the three color bands.

Algorithm	HA	GBTF	RI	MLRI	MLRI +wei	ARI	LSSC	RCNN	JCNN
Kodak ($\sigma = 0$)	34.84	39.58	38.58	38.91	39.32	39.90	41.44	41.97	**42.10**
Kodak ($\sigma = 1$)	34.61	38.99	38.11	38.38	38.75	39.30	40.69	**41.09**	40.00
Kodak ($\sigma = 3$)	33.27	36.20	35.75	35.80	36.02	36.49	37.15	37.37	**38.02**
Kodak ($\sigma = 10$)	27.71	28.5056	28.68	28.35	28.37	29.00	28.74	28.47	**33.26**
Imax ($\sigma = 0$)	34.76	34.80	36.48	36.74	36.82	37.57	36.15	38.94	**39.13**
Imax ($\sigma = 1$)	34.54	34.58	36.16	36.41	36.49	37.17	35.86	**38.44**	37.65
Imax ($\sigma = 3$)	33.28	33.27	34.47	34.59	34.65	35.20	34.18	35.91	**36.60**
Imax ($\sigma = 10$)	27.99	27.91	28.56	28.33	28.31	28.97	28.26	28.44	**33.07**
Canon ($\sigma = 0$)	36.45	40.99	40.84	40.80	41.02	42.27	41.51	43.55	**43.61**
Canon ($\sigma = 1$)	36.01	39.97	39.78	39.75	39.93	40.93	40.72	**42.19**	41.34
Canon ($\sigma = 3$)	34.14	36.59	36.47	36.37	36.47	37.14	37.07	37.78	**39.36**
Canon ($\sigma = 10$)	28.48	29.16	29.36	28.96	28.94	29.81	29.61	29.09	**34.83**
Panasonic ($\sigma = 0$)	34.07	38.37	38.18	38.17	38.39	39.55	39.25	41.25	**41.28**
Panasonic ($\sigma = 1$)	33.79	37.74	37.57	37.56	37.74	38.75	38.71	**40.35**	39.48
Panasonic ($\sigma = 3$)	32.50	35.26	35.16	35.11	35.25	35.90	35.90	36.83	**37.79**
Panasonic ($\sigma = 10$)	27.63	28.51	28.65	28.38	28.40	29.01	28.83	28.58	**33.48**
Average ($\sigma = 0$)	35.03	38.44	38.52	38.65	38.89	39.82	39.59	41.43	**41.53**
Average ($\sigma = 1$)	34.74	37.82	37.90	38.02	38.23	39.04	39.00	**40.52**	39.61
Average ($\sigma = 3$)	33.30	35.33	35.46	35.47	35.60	36.18	36.08	36.97	**37.94**
Average ($\sigma = 10$)	27.95	28.52	28.81	28.50	28.50	29.20	28.86	28.64	**33.66**

Beside the average PSNR, we also considered the variance of the PSNR for the image dataset. This variance somehow evaluates the robustness of each algorithm. In Figures 8–10, one can observe that the variance of ARI is smaller for Kodak without noise and that there is no significant difference between all these algorithms for the McMaster, Canon and Panasonic image datasets. The variance of RCNN always is smallest for all image datasets with $\sigma = 3$ and 10. Among all interpolation-based methods, ARI always has a small variance, but GBTF, RI, MLRI and MLRI+wei show almost identical performance. JCNN is a method for denoising and demosaicking and performs best when $\sigma \geq 3$.



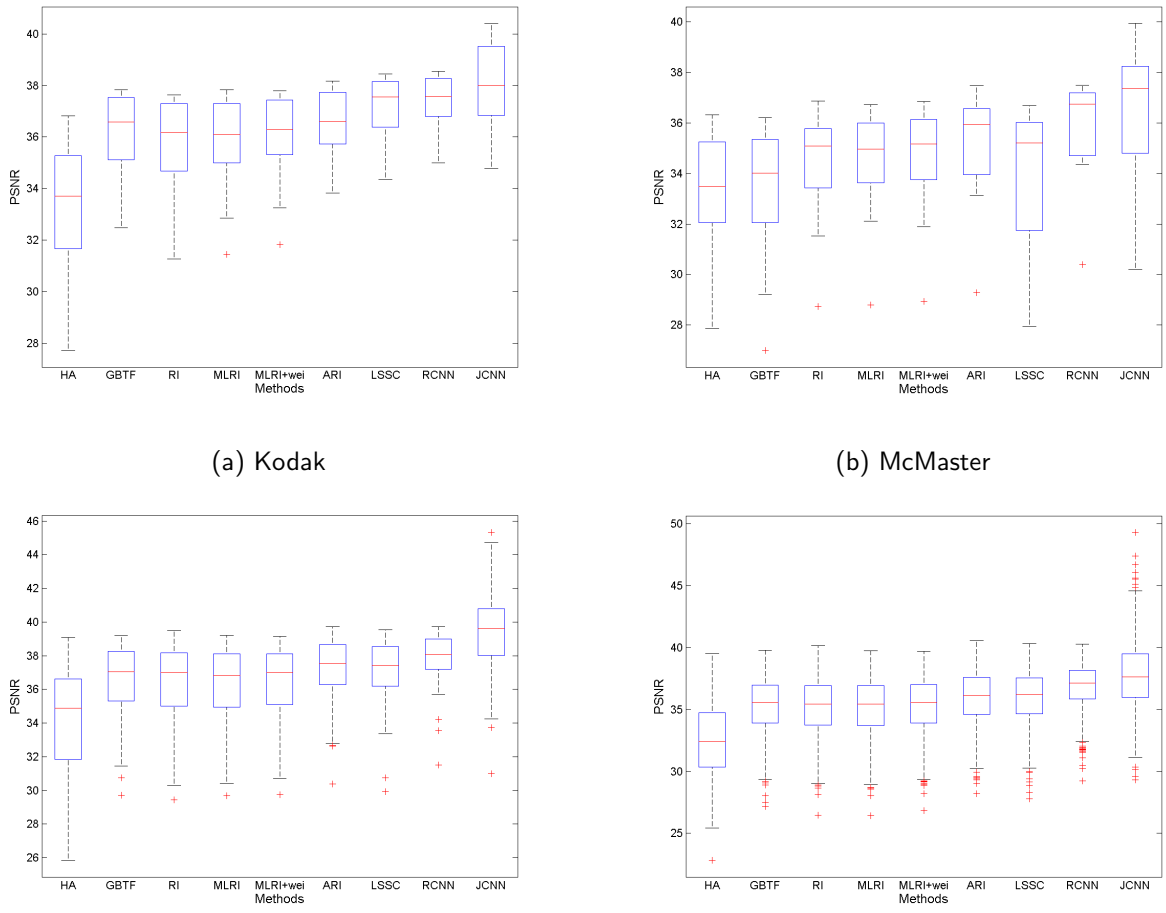
(a) Kodak

(b) McMaster

(c) Canon

(d) Panasonic

Figure 8: CPSNR performance for the standard Kodak, McMaster, Canon and Panasonic datasets without noise.



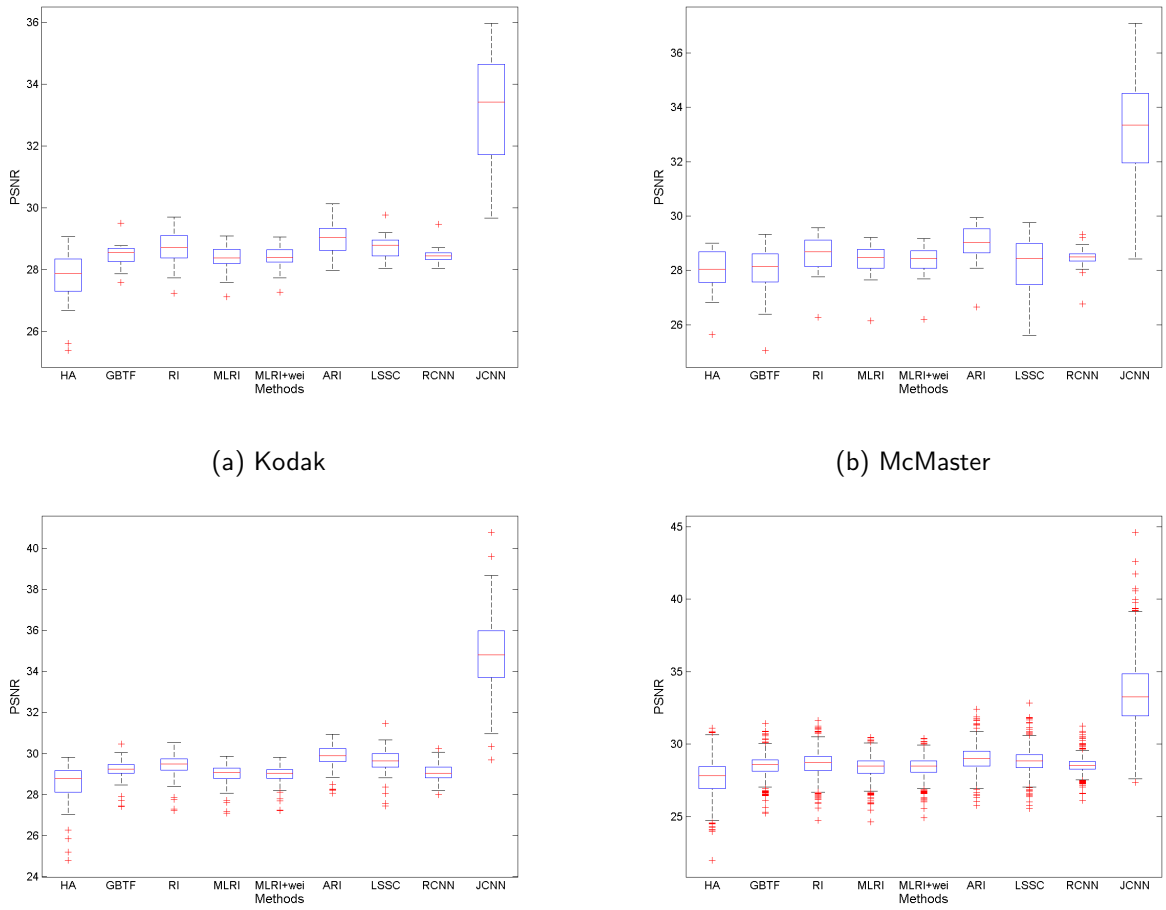
(a) Kodak

(b) McMaster

(c) Canon

(d) Panasonic

Figure 9: CPSNR performance for the standard Kodak, McMaster, Canon and Panasonic datasets with noise ($\sigma = 3$).



(a) Kodak

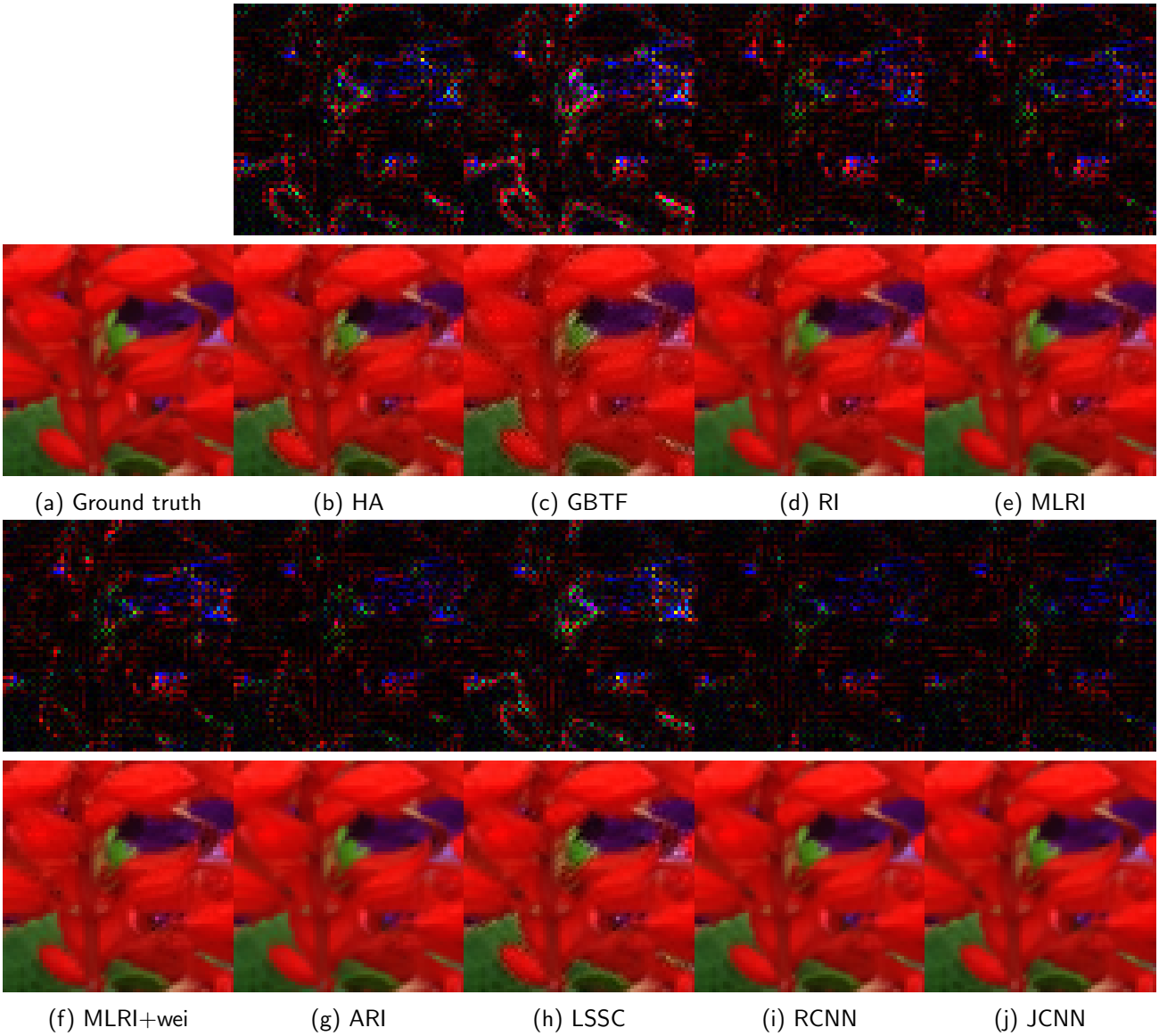
(b) McMaster

(c) Canon

(d) Panasonic

Figure 10: PSNR performance for the standard Kodak, McMaster, Canon and Panasonic datasets with noise ($\sigma = 10$).

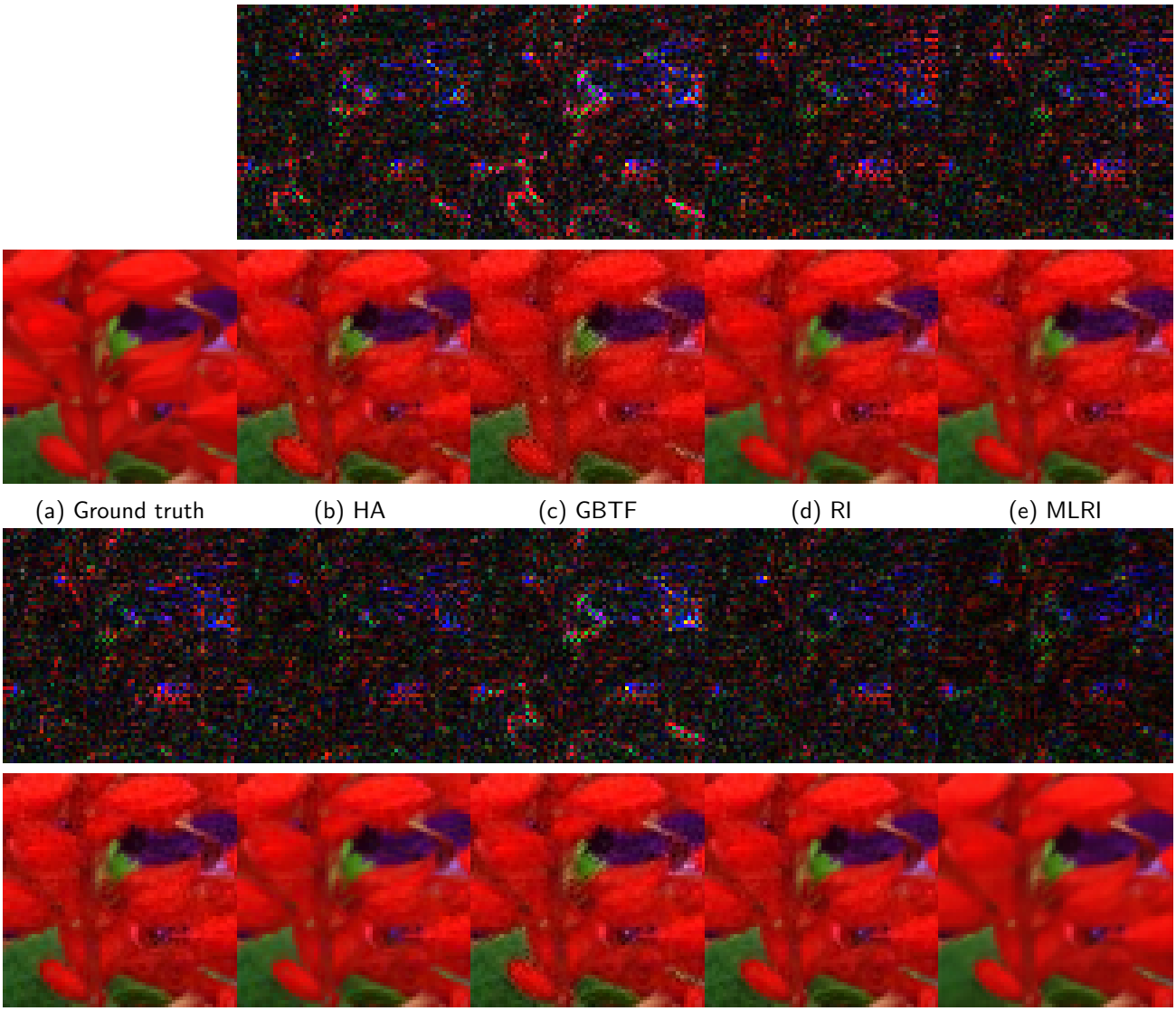
Figures 11–14 visually compare the demosaicked results of the nine algorithms. The figures show the restored images and the error images (the difference between restored and ground truth image). From Figure 11, one can see that HA, GBTF, LSSC have strong zipper artifacts and RI, MLRI and MLRI+wei have slight zipper artifacts. The most perceptually pleasing results are from the ARI, RCNN and JCNN results. Figure 12 shows the image 17 from the McMaster dataset with noise $\sigma = 5$, there we see that a lot of noise still remains in the demosaicked results of HA, GBTF, RI, MLRI, MLRI+wei and LSSC, while ARI, RCNN and JCNN effectively remove noise during the demosaicking process. No significant color distortions are observed in this image for any algorithm. However, on image 01 of the Kodak dataset the interpolation-based algorithms introduce strong color distortions in the demosaicked results (see Figures 13 and 14 (b), (c),(d),(e),(f),(g)). Minor color distortions can be found in the output of LSSC (see Figure 13 and 14 (h)). Perceptually good results can be seen in the results of RCNN and JCNN. The interpolation-based algorithms do not work well on detailed structures.



(a) Ground truth (b) HA (c) GBTF (d) RI (e) MLRI

(f) MLRI+wei (g) ARI (h) LSSC (i) RCNN (j) JCNN

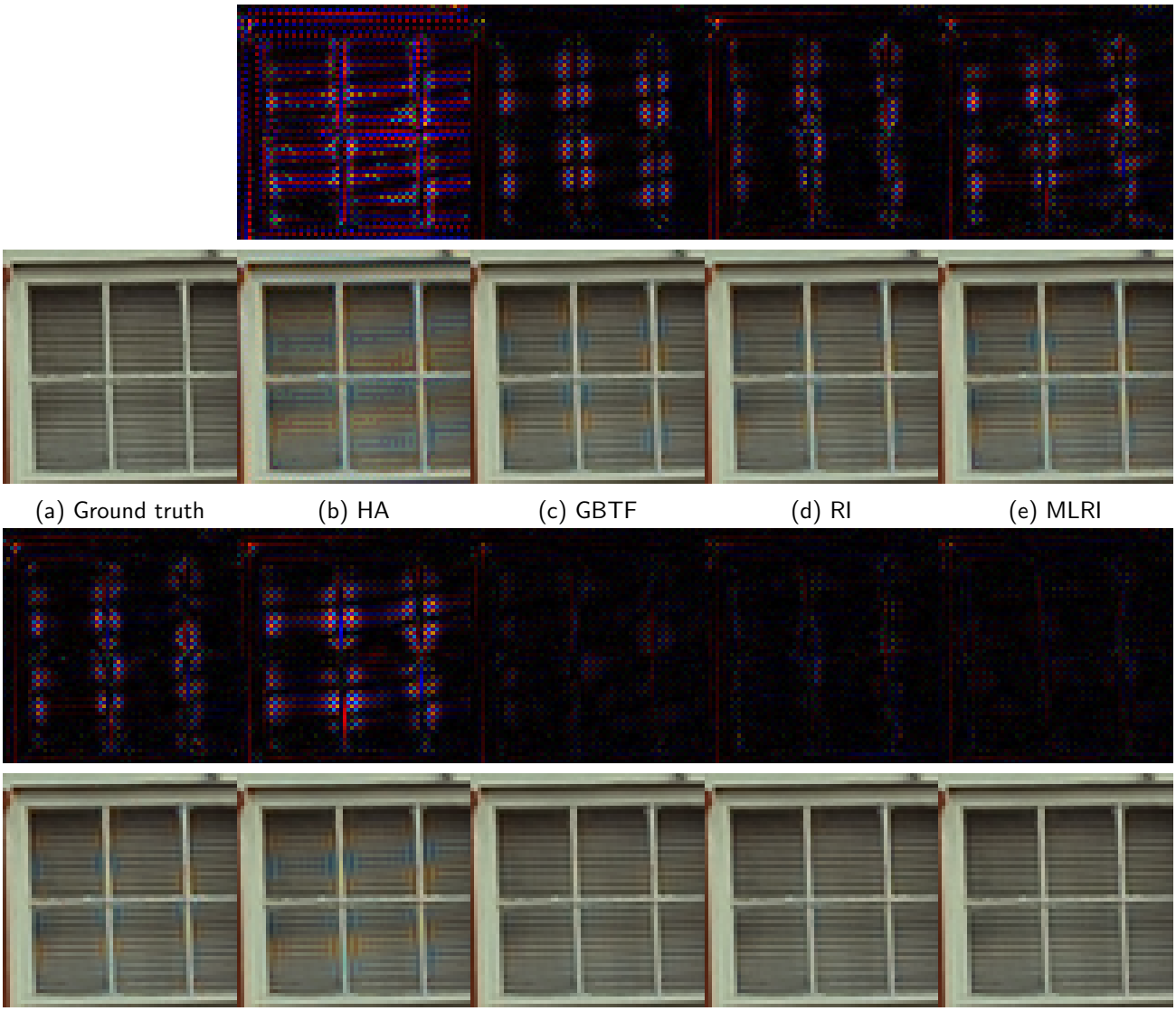
Figure 11: Visual comparison of demosaicked images of different algorithms for McMaster 17 without noise ($\sigma = 0$).



(a) Ground truth (b) HA (c) GBTF (d) RI (e) MLRI

(f) MLRI+wei (g) ARI (h) LSSC (i) RCNN (j) JCNN

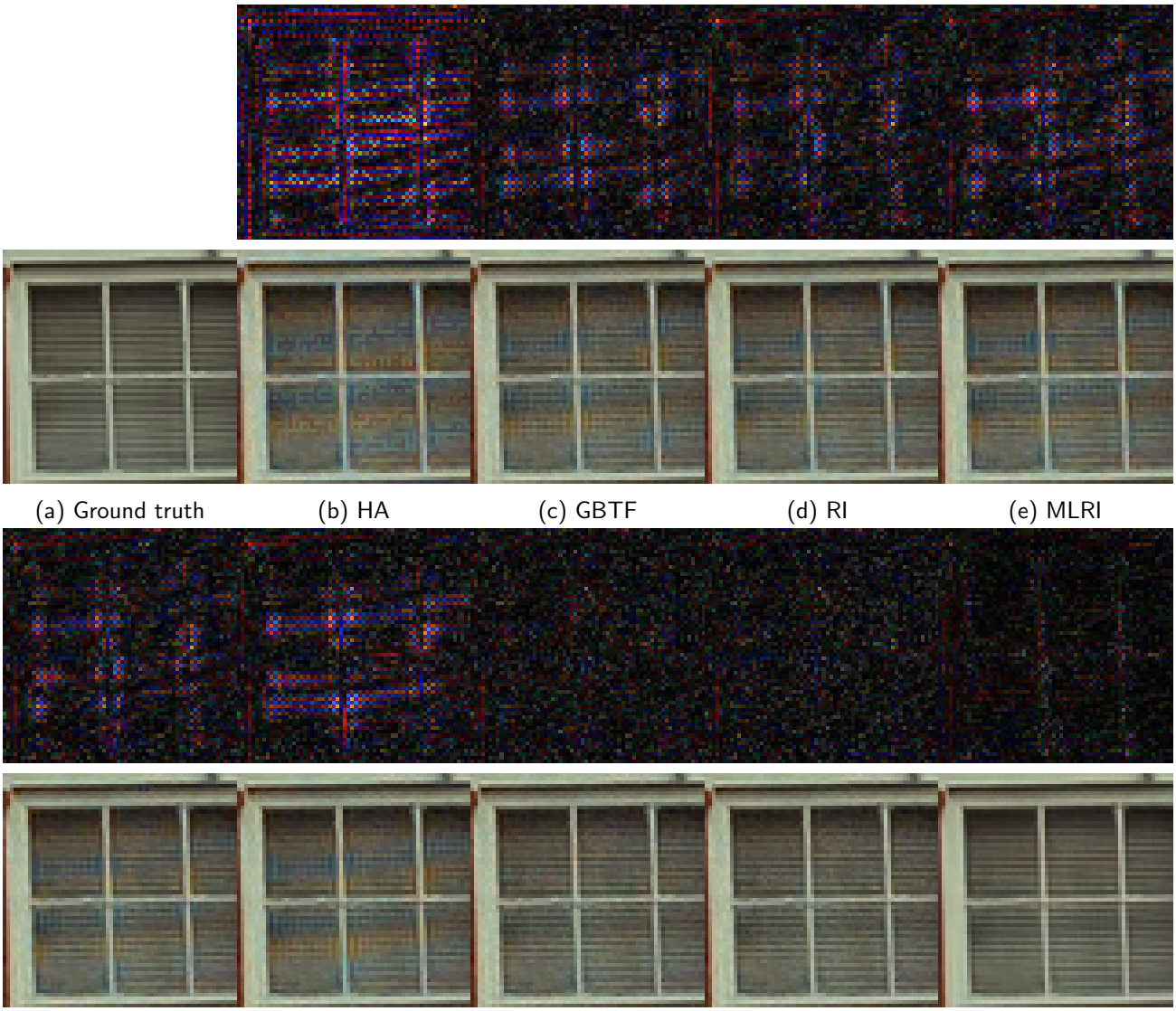
Figure 12: Visual comparison of demosaicked images of different algorithms for McMaster 17 with noise ($\sigma = 5$).



(a) Ground truth (b) HA (c) GBTF (d) RI (e) MLRI

(f) MLRI+wei (g) ARI (h) LSSC (i) RCNN (j) JCNN

Figure 13: Visual comparison of demosaicked images of different algorithms for Kodak 01 without noise.



(a) Ground truth (b) HA (c) GBTF (d) RI (e) MLRI

(f) MLRI+wei (g) ARI (h) LSSC (i) RCNN (j) JCNN

Figure 14: Visual comparison of demosaicked images of different algorithms for Kodak 01 with noise ($\sigma = 5$).

The running times of all algorithms for a 500×500 image are compared in Table 4. HA, GBTF, RI, MLRI, MLRI+wei are very fast with running times of less than one second. Because of the iterations, ARI is much slower than the other interpolation-based algorithms. The running time of the deep learning-based methods RCNN and JCNN is halfway between non-iterative interpolation methods and iterative interpolation methods. LSSC is the slowest amongst the nine methods as it takes 734 seconds to process a single 500×500 image!

Table 4: Running time of all algorithms for a 500×500 image using a Core(TM) i7-6820HQ CPU 2.70 GHz PC.

Algorithm	HA	GBTF	RI	MLRI	MLRI +wei	ARI	LSSC	RCNN	JCNN
Time (s)	0.3015	0.3402	0.4410	0.6100	0.8578	17.1670	734.3666	5.2174	2.9735

For each algorithm, we observed the following:

- The HA algorithm, which was developed in 1997, is very simple but works well and is fast.

- GBTF is a more elaborated version of HA, and runs as fast as HA. The PSNR of the demosaicked image of the GBTF algorithm is significantly higher than that of HA. For the images with much texture such as image Kodak 01, the visual quality has significantly improved, but at the edge of small objects, zipper artifacts are stronger than with HA.

- RI, MLRI and MLRI+wei have almost the same PSNR values and variances for all test image datasets. The computational complexity of RI, MLRI, MLRI+wei are respectively 0.4410, 0.6100 and 0.8578 seconds to process a 500×500 image, so they are quite fast. Their PSNR values are almost the same as for GBTF, but their visual quality is better than GBTF because they reduce the zipper artifacts.

- ARI performs the best amongst the interpolation-based algorithms in terms of both PSNR value and visual quality, and is robust to noise in images. For mosaicked images with noise $\sigma = 10$ and above, it performs even better than deep learning based algorithms. Its disadvantage is its computational complexity. It is much slower than the other interpolation-based methods. It takes more than 17 seconds for processing a 500×500 image while other interpolation-based algorithms just used less than 1 second.

- LSSC is a dictionary learning algorithm. It learns parameters for every image and then it involves a huge computational burden. For example, it takes 734 seconds to process a 500×500 image. The PSNR value of LSSC is almost the same as the one of ARI. In terms of visual quality, LSSC performs better in textured areas but ARI succeeds on the edges of small objects.

- RCNN is a deep learning based method. It wins for both PSNR and visual quality, except when the mosaicked images have noise exceeding $\sigma = 10$.

- JCNN is a deep learning based method for joint denoising and demosaicking, but the authors provide two networks, one trained to deal with noisy mosaicked images and the other for noise free mosaicked images. When noise level $\sigma \geq 3$, this algorithm outperforms the rest.

8 Conclusion

In this paper, we analyzed arguably the best important interpolation-based handcrafted demosaicking algorithms. From HA to ARI, we investigated their mathematics principles and clarified the conceptual progress that leads from each method to the next more sophisticated one. We managed a huge simplification of the description of these methods, and delivered compact but complete pseudo-code for each of them. We also discussed these algorithms in terms of PSNR, output visual quality, and computational complexity, and compared them with learning-based algorithms. The interpolation-based algorithms which are simple and run very fast, might provide useful ideas to improve and accelerate deep learning-based demosaicking algorithm.

The demo applies all the methods presented here to a same input image. It allows to select the Bayer pattern and the noise level. (The default value 1 for the smoothing parameter σ is good most of the time, however a larger value can perform better on images with saturated color such as those from the Kodak dataset).

Appendix A The Guided Filter

The Guided Filter (GF) [16] is a powerful edge-preserving filter that can be used as an alternative to the well-known bilateral filter [35]. Because of its effectiveness, GF is widely used in computer vision and image processing and it was recently included in the official MATLAB and OpenCV libraries.

GF involves a guidance image I, an input image p to be filtered, and an output image q. Perhaps the most important aspect of the guided filter is the local linear relation that is established between the guidance image I and the output image q in a window $\mathcal{N}^d_{s,t}$. We use the notation $q^{raw}_{s,t}$ (which depends on the window $\mathcal{N}^d_{s,t}$, therefore when (s,t) changes the value of $q^{raw}_{s,t}$ will be different) to denote the first step of the guided filter. At this step, and in each window, the output of the guided filter is a linear transformation of the guide. For each window $\mathcal{N}^d_{s,t}$, we have

$$q^{raw}_{s,t}(i,j) = a(s,t)I(i,j) + b(s,t), \quad \forall (i,j) \in \mathcal{N}^d_{s,t}, \tag{47}$$

where $(a(s,t), b(s,t))$ are some linear coefficients assumed to be constant in $\mathcal{N}^d_{s,t}$. This local linear model ensures that $q^{raw}_{s,t}$ has an edge only if I has one, because

$$\nabla q^{raw}_{s,t}(i,j) = a(s,t)\nabla I(i,j), \quad \forall (i,j) \in \mathcal{N}^d_{s,t}.$$

In $\mathcal{N}^d_{s,t}$, $(s,t) \in \Omega$, the raw guided filter is the result of fitting the linear model (47) to the input image p by minimizing the cost function

$$E(a(s,t), b(s,t)) = \sum_{(i,j)\in\mathcal{N}^d_{s,t}} \left((a(s,t)I(i,j) + b(s,t) - p(i,j))^2 + \epsilon a^2(s,t) \right), \tag{48}$$

where ϵ is a regularization parameter penalizing large values of $a(s,t)$. The underlying model is a decomposition $p(i,j) = q^{raw}_{s,t}(i,j) + n(i,j)$ where $n(i,j)$ is a component such as noise or texture that we want to separate from the base $q^{raw}_{s,t}(i,j)$. The minimization of the energy (48) amounts to minimizing the difference between p and the base $q^{raw}_{s,t}(i,j)$. Moreover, the parameter ϵ penalizes large values of the coefficient a, and thus helps removing the small variations in p. Model (48) is the linear ridge regression model [9, 13]. The linear coefficient $b(s,t)$ is obtained by minimizing the energy (48),

$$\frac{\partial E(a(s,t), b(s,t))}{\partial b(s,t)} = \sum_{(i,j)\in\mathcal{N}^d_{s,t}} (2(a(s,t)I(i,j) + b(s,t) - p(i,j))) = 0,$$

i.e.

$$b(s,t)\sum_{(i,j)\in\mathcal{N}_{s,t}^d}1=\sum_{(i,j)\in\mathcal{N}_{s,t}^d}p(i,j)-a(s,t)\sum_{(i,j)\in\mathcal{N}_{s,t}^d}I(i,j).$$

The equation above implies

$$b(s,t)=\overline{p}(s,t)-a(s,t)\overline{I}(s,t), \tag{49}$$

where $\overline{I}(s,t)$ is the mean of I in $\mathcal{N}_{s,t}^d$, i.e.

$$\overline{I}(s,t)=\frac{1}{(2d_s+1)^2}\sum_{(i,j)\in\mathcal{N}_{s,t}^d}I(i,j), \tag{50}$$

and $\overline{p}(s,t)$ is the mean of p in $\mathcal{N}_{s,t}^d$ given by

$$\overline{p}(s,t)=\frac{1}{(2d_s+1)^2}\sum_{(i,j)\in\mathcal{N}_{s,t}^d}p(i,j). \tag{51}$$

By the same argument, we have

$$\frac{\partial E(a(s,t),b(s,t))}{\partial a(s,t)}=\sum_{(i,j)\in\mathcal{N}_{s,t}^d}(2I(i,j)(a(s,t)I(i,j)+b(s,t)-p(i,j))+2\epsilon a(s,t))=0. \tag{52}$$

Substituting (49) into (52) leads to

$$a(s,t)=\frac{\frac{1}{(2d_s+1)^2}\sum_{(i,j)\in\mathcal{N}_{s,t}^d}I(i,j)p(i,j)-\overline{I}(s,t)\overline{p}(s,t)}{\sigma^2(s,t)+\epsilon}. \tag{53}$$

Here, $\sigma^2(s,t)$ is the variance of I in $\mathcal{N}_{s,t}^d$, i.e.

$$\sigma^2(s,t)=\frac{1}{(2d_s+1)^2}\sum_{(i,j)\in\mathcal{N}_{s,t}^d}(I(i,j)-\mu(s,t))^2.$$

Once the linear coefficients $(a(s,t),b(s,t))$ have been obtained, the output $q_{s,t}^{raw}(i,j)$ can be computed by (47). Interestingly, the numerator in Equation (53) is the empirical covariance between the input image p and the guide I and σ^2 is the empirical variance of I. Thus $a(s,t)$ and $b(s,t)$ can be expressed as

$$a(s,t)=\frac{\mathrm{Cov}\{I,p\}(s,t)}{\mathrm{Var}\{I\}(s,t)+\epsilon},$$

and

$$b(s,t)=\mathrm{Mean}\{p\}(s,t)-a(s,t)\mathrm{Mean}\{I\}(s,t),$$

where Mean denotes the mean in the window $\mathcal{N}_{s,t}^d$.

However, a pixel (i,j) is involved in all the overlapping windows $\mathcal{N}_{s,t}^d$ containing it. Thus the value of $q_{s,t}^{raw}(i,j)$ in (47) varies when computed in different windows $\mathcal{N}_{s,t}^d$. A simple strategy is to average all the possible values of $q_{s,t}^{raw}(i,j)$ with $(i,j)\in\mathcal{N}_{s,t}^d$. Since the window $\mathcal{N}_{s,t}^d$ contains (i,j) if only if $(s,t)\in\mathcal{N}_{i,j}^d$, the guided filter is given by

$$q(i,j)=\frac{1}{(2d_s+1)^2}\sum_{(s,t)\in\mathcal{N}_{i,j}^d}q_{s,t}^{raw}(i,j),\quad\forall(i,j)\in\Omega. \tag{54}$$

Thus, after computing $(a(s,t), b(s,t))$ for all windows $\mathcal{N}^d_{s,t}$ in the image, the filter (54) becomes

$$q(i,j) = \frac{1}{(2d_s+1)^2} \sum_{(s,t)\in\mathcal{N}^d_{i,j}} \left(a(s,t)I(i,j) + b(s,t)\right), \quad \forall (i,j) \in \Omega. \tag{55}$$

Due to the symmetry of the box window, the linear coefficients can be averaged instead, the definition of the guided filter (55) is rewritten as

$$q(i,j) = \overline{a}(i,j)I(i,j) + \overline{b}(i,j), \quad \forall (i,j) \in \Omega, \tag{56}$$

with

$$\overline{a}(i,j) = \frac{1}{(2d_s+1)^2} \sum_{(s,t)\in\mathcal{N}^d_{i,j}} a(s,t), \tag{57}$$

$$\overline{b}(i,j) = \frac{1}{(2d_s+1)^2} \sum_{(s,t)\in\mathcal{N}^d_{i,j}} b(s,t), \tag{58}$$

where (57) and (58) are the average coefficients of all windows overlapping (i,j).

Considering the modification introduced by (56), $q(i,j)$ is no longer a scaling of $I(i,j)$ in $\mathcal{N}^d_{i,j}$, because the linear coefficients $(\overline{a}(i,j), \overline{b}(i,j))$ vary spatially. But as $(\overline{a}(i,j), \overline{b}(i,j))$ are the output of a mean filter, their gradients can be expected to be much smaller than the gradient of I near strong edges. Thus, we still expect that $\nabla q(i,j) \simeq \overline{a}(i,j)\nabla I(i,j)$, meaning that abrupt intensity changes in I are mostly preserved in q. The pseudo-code of GF is given in Algorithm 11.

A.1 The Minimized-Laplacian Guided Filter

Kiku et al. introduced in [22] MLGF for demosaicking. Instead of obtaining the linear coefficients $(a(s,t), b(s,t))$ by minimizing the cost function (48), the linear coefficient $a(s,t)$ is computed by minimizing the following cost function on the Laplacian of the image

$$\begin{aligned} E(a(s,t)) &= \sum_{(i,j)\in\mathcal{N}^d_{s,t}} \left(\left(\Delta(a(s,t)I(i,j) + b(s,t) - p(i,j))\right)^2 + \epsilon a^2(s,t) \right) \\ &= \sum_{(i,j)\in\mathcal{N}^d_{s,t}} \left(\left(a(s,t)\Delta I(i,j) - \Delta p(i,j)\right)^2 + \epsilon a^2(s,t) \right), \end{aligned} \tag{59}$$

where Δ denotes the discrete Laplacian operator and M is a mask. Minimizing (59) leads to

$$\frac{\mathrm{d}E(a(s,t))}{\mathrm{d}a(s,t)} = \sum_{(i,j)\in\mathcal{N}^d_{s,t}} \left(2\Delta I(i,j)(a(s,t)\Delta I(i,j) - \Delta p(i,j) + 2\epsilon a(s,t))\right) = 0.$$

Then we obtain

$$a(s,t) = \frac{\overline{\Delta p \Delta I}(s,t)}{\overline{\Delta I^2}(s,t) + \epsilon}, \tag{60}$$

where

$$\overline{\Delta p \Delta I}(s,t) = \frac{\sum_{(i,j)\in\mathcal{N}^d_{s,t}} \left(\Delta I(i,j)\Delta p(i,j)\right)}{\sum_{(i,j)\in\mathcal{N}^d_{s,t}} M(i,j)}$$

and

$$\overline{\Delta I^2}(s,t) = \frac{\sum_{(i,j)\in\mathcal{N}^d_{s,t}} M(i,j)\Delta I^2(i,j)}{\sum_{(i,j)\in\mathcal{N}^d_{s,t}} M(i,j)}.$$

Algorithm 11 Pseudo-code for Guided Filter

Input: guidance image I, filtering input image p, Mask matrix M, Laplacian map Δ, radius S, case J: case for algorithm choices $J \in \{RI, MLRI\}$, xcase $Weight$: case for weights $Weight \in \{True, False\}$, regularization ϵ

Output: Estimate q

1: **function** GF$(I, p, M, \Delta, S, J, Weight, \epsilon)$

// Initialization //

2: $\overline{I} \leftarrow f_{wm}(I. * M, M, d_s)$ // $\overline{I}$ is defined by Equation (50) and f_{wm} denotes the weighted box mean (6)//

3: $\overline{p} \leftarrow f_{wm}(p. * M, M, d_s)$ // $\overline{p}$ is defined by Equation (51) //

// Compute the parameters a and b//

4: **if** $J = MLGF$ **then** //If $J = MLGF$, it is MLGF, else GF//

5: $\overline{\Delta p \Delta I} \leftarrow f_{wm}(\Delta I. * \Delta p. * M, M, d_s)$

6: $\overline{\Delta I^2} \leftarrow f_{wm}(\Delta I. * \Delta I. * M, M, d_s)$

7: $a \leftarrow \overline{\Delta p \Delta I}./(\overline{\Delta I^2} + \epsilon)$ // The parameter a of MLGF is defined by Equation (60) //

8: **else**

9: $\overline{pI} \leftarrow f_{wm}(I. * p. * M, M, d_s)$

10: $Cov(IP) \leftarrow \overline{pI} - \overline{I}. * \overline{p}$

11: $Var(I) \leftarrow \overline{I^2} - \overline{I}. * \overline{I}$

12: $a \leftarrow Cov(IP)./(Var(I) + \epsilon)$ // The parameter a of GF is defined by Equation (53) //

13: **end if**

14: $b \leftarrow \overline{p} - a. * \overline{I}$

// Compute the average or weighted average of a and b//

15: **if** $Weight = True$ **then**//If $Weight = True$, it is weighed GF/MLGF, else GF/MLGF//

16: $W \leftarrow 1./\left(f_{wm}\left(M. * (p. * M - a. * I. * M - b)^2, M, d_s\right)\right)$ // The weights defined by Equation (61) //

17: $\overline{a} \leftarrow f_{wm}(a, W, d_s)$ // Weighted average $\overline{a}$ defined by Equation (62) //

18: $\overline{b} \leftarrow f_{wm}(b, W, d_s)$ // Weighted average $\overline{b}$ defined by Equation (62) //

19: **else**// The case is the guide filter //

20: $\overline{a} \leftarrow f_m(a, d_s)$ // Box average $\overline{a}$ defined by Equation (57) //

21: $\overline{b} \leftarrow f_m(b, d_s)$ // Box average $\overline{b}$ defined by Equation (58) //

22: **end if**

// Compute the final GF estimate //

23: $q \leftarrow \overline{a}. * I + \overline{b}$ // The final GF estimate defined by Equation (56) //

24: **return** $result$

25: **end function**

Note that the parameter $b(s,t)$ has disappeared in (59) as it is constant with respect to (i,j). After computing $a(s,t)$, we obtain the linear coefficient $b(s,t)$ by (49). If $a(i,j)$ is given by (60), the formula becomes the MLGF estimate. The difference between GF and MLGF is just the different definition of the parameter $a(i,j)$. It therefore is easy to implement both algorithms in a single algorithm (see Algorithm 11). If $a(i,j)$ is computed by Algorithm 11 lines 5-7, then we have MLGF, otherwise we have GF.

A.2 The Weighed Guided Filter

Kiku et al. [22] consider that for a pixel (i,j), the smaller the square error $(a(s,t)I(i,j) + b(s,t) - p(i,j))^2$, the larger the weight value. This is reasonable, because when the error is small, the estimate

$q_{s,t}^{raw}(i,j)$ is good. Therefore the authors introduced the weight

$$W(s,t) = \left(\frac{\sum_{(i,j)\in\mathcal{N}_{s,t}^d} (M(i,j)(a(s,t)I(i,j) + b(s,t) - p(i,j))^2)}{\sum_{(i,j)\in\mathcal{N}_{s,t}^d} M(i,j)} \right)^{-1}. \tag{61}$$

Hence in [22] Kiku et al. proposed their Weighted Guided Filter by using the parameters $\overline{a}(i,j)$ and $\overline{b}(i,j)$ of (56) which are given by

$$\overline{a}(i,j) = \frac{\sum_{(s,t)\in\mathcal{N}_{i,j}^d} W(s,t)a(s,t)}{\sum_{(s,t)\in\mathcal{N}_{i,j}^d} W(s,t)} \quad and \quad \overline{b}(i,j) = \frac{\sum_{(s,t)\in\mathcal{N}_{i,j}^d} W(s,t)b(s,t)}{\sum_{(s,t)\in\mathcal{N}_{i,j}^d} W(s,t)}. \tag{62}$$

We incorporate the weights in the GF code (see Algorithm 11 lines 15-18). The difference between the four types of guided filters is small, so we have decided to present the four guided filter variants in a single Algorithm 11. Different parameters determine different types of guided filter. The corresponding parameters of Algorithm 11 are selected as follows:

- Guided filter: $J = RI$, $Weight = False$;
- Weighted guided filter: $J = RI$, $Weight = True$;
- Minimizing-Laplacian guided filter: $J = MLRI$, $Weight = False$;
- Weighted minimizing-Laplacian guided filter: $J = MLRI$, $Weight = True$.

Acknowledgments

Jin has been supported by the National Natural Science Foundation of China (Grants No. 61661039, No.61661040, No. 61661038), and Jin was also supported by the China Scholarship Council for a one year visiting at École Normale Supérieure Paris-Saclay (No. 201806810001).

Image Credits

 R. Franzen, Kodak lossless true color image suite[3]

 L. Zhang and X. Wu, McMaster image dataset[4]

References

[1] J. E. Adams, *Design of practical color filter array interpolation algorithms for digital cameras .2*, in Proceedings of the IEEE International Conference on Image Processing, vol. 1, 1998, pp. 488–492. http://dx.doi.org/10.1109/ICIP.1998.723540.

[2] J. Aelterman, B. Goossens, J. De Vylder, A. Pižurica, and W. Philips, *Computationally efficient locally adaptive demosaicing of color filter array images using the dual-tree complex wavelet packet transform*, PloS one, 8 (2013), p. e61846. http://dx.doi.org/10.1371/journal.pone.0061846.

[3] http://r0k.us/graphics/kodak/

[4] http://r0k.us/graphics/kodak

[3] D. Alleysson, S. Susstrunk, and J. Hérault, *Linear demosaicing inspired by the human visual system*, IEEE Transactions on Image Processing, 14 (2005), pp. 439–449. http://dx.doi.org/10.1109/TIP.2004.841200.

[4] C. Bai, J. Li, and Z. Lin, *Demosaicking based on channel-correlation adaptive dictionary learning*, Journal of Electronic Imaging, 27 (2018), pp. 043–047. http://dx.doi.org/10.1117/1.JEI.27.4.043047.

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

[6] A. Buades, B. Coll, J. M. Morel, and C. Sbert, *Self-similarity driven color demosaicking*, IEEE Transactions on Image Processing, 18 (2009), pp. 1192–1202. http://dx.doi.org/10.1109/TIP.2009.2017171.

[7] ——, *Self-similarity driven demosaicking*, Image Processing On Line, 1 (2011), pp. 51–56. http://dx.doi.org/info:doi/10.5201/ipol.2011.bcms-ssdd.

[8] E. Chang, S. Cheung, and D. Y. Pan, *Color filter array recovery using a threshold-based variable number of gradients*, in Proceedings SPIE 3650, Sensors, Cameras, and Applications for Digital Photography, 1999. https://doi.org/10.1117/12.342861.

[9] N. R. Draper and H. Smith, *An introduction to nonlinear estimation*, Applied Regression Analysis, (1998), pp. 505–565. https://doi.org/10.1002/9781118625590.ch24.

[10] J. Duran and A. Buades, *Self-similarity and spectral correlation adaptive algorithm for color demosaicking*, IEEE Transactions on Image Processing, 23 (2014), pp. 4031–4040. http://dx.doi.org/10.1109/TIP.2014.2341928.

[11] ——, *A demosaicking algorithm with adaptive inter-channel correlation*, Image Processing On Line, 5 (2015), pp. 311–327. http://dx.doi.org/info:doi/10.5201/ipol.2015.145.

[12] T. Ehret and G. 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.

[13] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1, Springer Series in Statistics New York, 2001.

[14] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, *Deep joint demosaicking and denoising*, ACM Transactions on Graphics, 35 (2016), pp. 191:1–12. http://dx.doi.org/10.1145/2980179.2982399.

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

[16] K. He, J. Sun, and X. Tang, *Guided image filtering*, IEEE Transactions on Pattern Analysis & Machine Intelligence, (2013), pp. 1397–1409. http://dx.doi.org/10.1109/TPAMI.2012.213.

[17] K. Hua, S. C. Hidayati, F. He, C. Wei, and Y. F. Wang, *Context-aware joint dictionary learning for color image demosaicking*, Journal of Visual Communication and Image Representation, 38 (2016), pp. 230–245. https://doi.org/10.1016/j.jvcir.2016.03.004.

[18] S. P. Jaiswal, O. C. Au, V. Jakhetiya, Y. Yuan, and H. Yang, *Exploitation of inter-color correlation for color image demosaicking*, in Proceedings of the International Conference on Image Processing, 2014, pp. 1812–1816. http://dx.doi.org/10.1109/ICIP.2014.7025363.

[19] D. Khashabi, S. Nowozin, J. Jancsary, and A. W. Fitzgibbon, *Joint demosaicing and denoising via learned nonparametric random fields*, IEEE Transactions on Image Processing, 23 (2014), pp. 4968–4981. https://doi.org/10.1109/TIP.2014.2359774.

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

[21] ——, *Minimized-Laplacian residual interpolation for color image demosaicking*, in Proceedings of the Digital Photography X, vol. 9023, 2014, p. 90230L. http://www.ok.sc.e.titech.ac.jp/res/DM/MLRI.pdf.

[22] ——, *Beyond color difference: Residual interpolation for color image demosaicking*, IEEE Transactions on Image Processing, 25 (2016), pp. 1288–1300. https://doi.org/10.1109/TIP.2016.2518082.

[23] F. Kokkinos and S. Lefkimmiatis, *Iterative joint image demosaicking and denoising using a residual denoising network*, IEEE Transactions on Image Processing, 28 (2019), pp. 4177–4188. http://dx.doi.org/10.1109/TIP.2019.2905991.

[24] C. A. Laroche and M. A. Prescott, *Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients*, Dec. 13 1994. US Patent 5,373,322.

[25] Y. M. Lu, M. Karzand, and M. Vetterli, *Demosaicking by alternating projections: theory and fast one-step implementation*, IEEE Transactions on Image Processing, 19 (2010), pp. 2085–2098. http://dx.doi.org/10.1109/TIP.2010.2045710.

[26] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman, *Non-local sparse models for image restoration*, in Proceedings of the IEEE International Conference on Computer Vision, 2009, pp. 2272–2279. http://dx.doi.org/10.1109/ICCV.2009.5459452.

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

[28] ——, *Adaptive residual interpolation for color and multispectral image demosaicking*, Sensors, 17 (2017), p. 2787. http://dx.doi.org/10.3390/s17122787.

[29] 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, 17 (2007), pp. 105–122. https://doi.org/10.1002/ima.20109.

[30] I. Pekkucuksen and Y. Altunbasak, *Gradient based threshold free color filter array interpolation*, in Proceedings of the IEEE International Conference on Image Processing, 2010, pp. 137–140. http://dx.doi.org/10.1109/ICIP.2010.5654327.

[31] R. Ramanath and W. E. Snyder, *Adaptive demosaicking*, Journal of Electronic Imaging, 12 (2003), pp. 633–642. https://doi.org/10.1117/1.1606459.

[32] K. Satya and T. Jayachandra, *Deep learning approach for image denoising and image demosaicing*, International Journal of Computer Applications, 168 (2017), pp. 18–26. http://dx.doi.org/10.5120/ijca2017914500.

[33] D. S. Tan, W. Chen, and K. Hua, *Deep demosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks*, IEEE Transactions on Image Processing, 27 (2018), pp. 2408–2419. http://dx.doi.org/10.1109/TIP.2018.2803341.

[34] R. Tan, K. Zhang, W. Zuo, and L. Zhang, *Color image demosaicking via deep residual learning*, in Proceedings of the IEEE International Conference on Multimedia and Expo, 2017, pp. 793–798. http://www4.comp.polyu.edu.hk/~cslzhang/paper/CNNCDM_ICME2017.pdf.

[35] C. Tomasi and R. Manduchi, *Bilateral filtering for gray and color images*, in Proceedings of the IEEE International Conference on Computer Vision, 1998, pp. 839–846. http://dx.doi.org/10.1109/ICCV.1998.710815.

[36] J. Wu, R. Timofte, and L. Van Gool, *Efficient regression priors for post-processing demosaiced images*, in Proceedings of the IEEE International Conference on Image Processing, 2015, pp. 3495–3499. http://dx.doi.org/10.1109/ICIP.2015.7351454.

[37] ——, *Demosaicing based on directional difference regression and efficient regression priors*, IEEE Transactions on Image Processing, 25 (2016), pp. 3862–3874. http://dx.doi.org/10.1109/TIP.2016.2574984.

[38] W. Ye and K. Ma, *Image demosaicing by using iterative residual interpolation*, in Proceedings of the IEEE International Conference on Image Processing, 2014, pp. 1862–1866. http://dx.doi.org/10.1109/ICIP.2014.7025373.

[39] ——, *Color image demosaicing using iterative residual interpolation*, IEEE Transactions on Image Processing, 24 (2015), pp. 5879–5891. http://dx.doi.org/10.1109/TIP.2015.2482899.

[40] J. Zhang, A. Sheng, and K. Hirakawa, *A wavelet-GSM approach to demosaicking*, IEEE Signal Processing Letters, 25 (2018), pp. 778–782. http://dx.doi.org/10.1109/LSP.2018.2822802.

[41] L. Zhang and X. Wu, *Color demosaicking via directional linear minimum mean square-error estimation*, IEEE Transactions on Image Processing, 14 (2005), pp. 2167–2178. http://dx.doi.org/10.1109/TIP.2005.857260.

[42] 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), pp. 023016:1–16. http://dx.doi.org/10.1117/1.3600632.