Iterative Demosaicking Accelerated: Theory and Fast Noniterative Implementations

Yue M. Lu, Mina Karzand, and Martin Vetterli

Audio-Visual Communications Laboratory
Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland

ABSTRACT

Color image demosaicking is a key process in the digital imaging pipeline. In this paper, we present a rigorous treatment of a classical demosaicking algorithm based on alternating projections (AP). Since its publication, the AP algorithm has been widely cited and served as a benchmark in a flurry of papers in the demosaicking literature. Despite its impressive performances, a relative weakness of the AP algorithm is its high computational complexity. In our work, we provide a rigorous analysis of the convergence of the AP algorithm based on the concept of contraction mapping. Furthermore, we propose an efficient noniterative implementation of the AP algorithm in the polyphase domain. Numerical experiments show that the proposed noniterative implementation achieves the same results obtained by the original AP algorithm at convergence, but is about an order of magnitude faster than the latter.

1. INTRODUCTION

Most digital cameras use a single monochromatic image sensor to capture the incoming light intensities. To acquire the tri-value color information, a color filter array (CFA) is placed above the image sensor, ensuring that each pixel in the array detects one of the primary (*e.g.* red, green, and blue) color signals. A key process in the digital imaging pipeline is thus to reconstruct full-resolution color signals from their CFA downsampled versions. The feasibility and quality of this interpolation process, often referred to as *demosaicking* (or demosaicing), largely depend on the existence of strong correlations between different color channels.

Since the birth of digital cameras in the early 1990s, numerous algorithms have been proposed for image demosaicking (see Refs. 1–3 for excellent reviews on existing approaches). In this paper, we study one classical algorithm proposed by Gunturk, Altunbasak, and Mersereau,[4] which is based on the concept of alternating projections (AP). Published in 2002, the AP algorithm has since become one of the representative and high quality methods on demosaicking, and has been widely cited and served as a benchmark in a flurry of more recent papers. Despite the impressive performances, a relative weakness of the AP algorithm is its high computational complexity: The full-resolution color images are reconstructed in an iterative fashion, with each iteration involving a two-dimensional (2-D) *nonsubsampled* wavelet decomposition and reconstruction of the entire image.

The purposes of this paper are two-fold. First, we provide a rigorous analysis of the convergence of the AP algorithm. In the original paper,[4] the authors attribute the convergent property to projection onto convex sets. In our analysis however, we maintain that a more accurate explanation should be based on the contraction mapping theorem (a.k.a. Banach fixed point theorem).[5] Due to the popularity of the AP algorithm in the demosaicking literature, we believe that such a careful treatment is worthwhile in its own right, even if only for pedagogical reasons.

In addition to providing a more rigorous foundation for the AP algorithm, our theoretical analysis also leads to the construction of a fast *noniterative* implementation of the AP algorithm, which is the second and more practical purpose of this work. Using the powerful tool of polyphase representation in multirate signal processing, we show that the iterative AP algorithm can be *equivalently* implemented as three simple linear filtering operations in the polyphase domain. The filter coefficients can be pre-computed and are completely determined by the wavelet filters used in the original algorithm. Numerical experiments have shown that the proposed noniterative

Further author information: (Send correspondence to Y. M. Lu.)
Y. M. Lu: yue.lu@epfl.ch; M. Karzand: mina.karzand@epfl.ch; M. Vetterli: martin.vetterli@epfl.ch

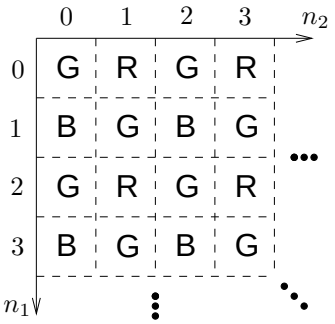


Figure 1. A portion of the Bayer color filter array. R, G, and B represent the red, green, and blue pixels, respectively.

implementation achieves the same results obtained by the original AP algorithm at convergence, but is about an order of magnitude faster than the latter.

The rest of this paper is organized as follows. We have a brief overview of the original AP algorithm in Section 2, which sets the ground for all subsequent discussions. We derive in Section 3 a compact representation of the AP algorithm in the polyphase domain, and provide a rigorous analysis of its convergence based on the contraction mapping theorem. Section 4 presents a fast noniterative implementation of the AP algorithm. We conclude the paper in Section 5. Due to space limitations, we only present the proofs for the most important results in this paper, and leave the proofs for all other results to Ref. 6.

Notation: Throughout the paper, we regard grayscale images $x[\boldsymbol{n}]$, with $\boldsymbol{n} \stackrel{\text{def}}{=} (n_1, n_2)$, as vectors in $\ell^2(\mathbb{Z}^2)$, *i.e.*, the linear space of square-summable sequences. The 2-D Z-transform of $x[\boldsymbol{n}]$ is defined as

$$X(\boldsymbol{z}) = \sum_{\boldsymbol{n} \in \mathbb{Z}^2} x[\boldsymbol{n}]\, \boldsymbol{z}^{-\boldsymbol{n}},$$

where $\boldsymbol{z}^{-\boldsymbol{n}} \stackrel{\text{def}}{=} z_1^{-n_1} z_2^{-n_2}$. Specializing to the unit sphere, we have $\boldsymbol{z} = e^{j\boldsymbol{\omega}} \stackrel{\text{def}}{=} (e^{j\omega_1}, e^{j\omega_2})$, and hence $X(e^{j\boldsymbol{\omega}})$ represents the Fourier transform of $x[\boldsymbol{n}]$. Finally, we denote by $\mathbb{1}_{\mathcal{D}}(\boldsymbol{\omega})$ the indicator function of a domain $\mathcal{D}$, *i.e.*, $\mathbb{1}_{\mathcal{D}}(\boldsymbol{\omega}) = 1$ if $\boldsymbol{\omega} \in \mathcal{D}$ and $\mathbb{1}_{\mathcal{D}}(\boldsymbol{\omega}) = 0$ otherwise.

2. BACKGROUND AND PROBLEM STATEMENT

In this section, we first briefly overview the original AP demosaicking algorithm,[4] and then explicitly state the key problems that we want to address in this work.

2.1 Demosaicking by Alternating Projections

Figure 1 shows the sampling pattern of the widely used Bayer CFA.[7] Let $s[\boldsymbol{n}]$ denote the raw sensor measurements obtained from the CFA, and let $r[\boldsymbol{n}], g[\boldsymbol{n}], b[\boldsymbol{n}]$ represent the full-resolution red, green, and blue color channels, respectively. The goal of the AP demosaicking algorithm is to estimate these full-resolution color images from $s[\boldsymbol{n}]$.

Similar to many other demosaicking schemes in the literature, the AP algorithm starts by obtaining an estimate of the full-resolution green channel, which is then used in the subsequent estimation of the missing red and blue pixels. This two-step strategy is justified by the fact that the green channel in the Bayer CFA has twice the sampling density as that of the red or blue channels (see Figure 1), and hence is relatively easier to reconstruct. In the AP algorithm, the green channel is initially estimated by an edge-directed interpolation scheme (such as the one proposed in Ref. 8), followed by a subband-based update step. We omit further descriptions of this update step, whose details can be found in Ref. 4.

In this work, our focus is on how the AP algorithm iteratively interpolates the missing red and blue pixels, with alternations between two constraint sets. For simplicity, we only concentrate on the case of red pixels throughout the paper. The processing for the blue pixels can be easily inferred by symmetry.

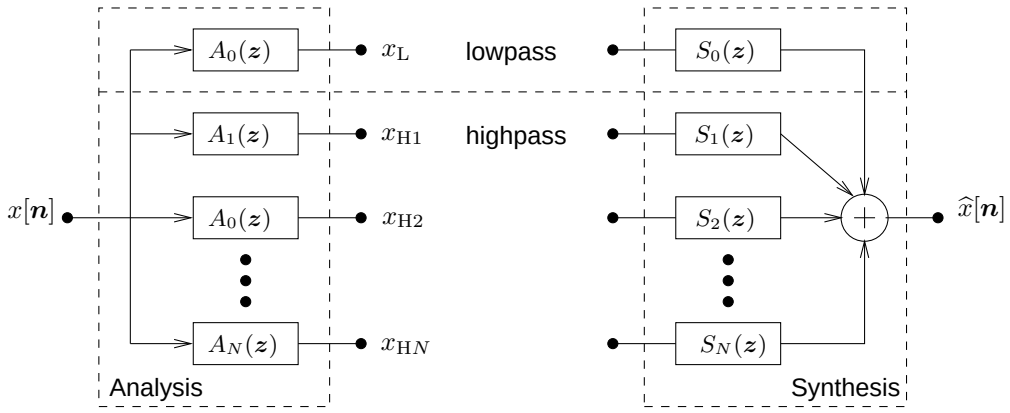


Figure 2. A generic 2-D perfect reconstruction filter bank with one lowpass channel [$A_0(\boldsymbol{z})$ and $S_0(\boldsymbol{z})$] and N highpass channels [$A_i(\boldsymbol{z})$ and $S_i(\boldsymbol{z})$ for $i = 1...N$].

The first constraint introduced by the AP algorithm exploits the correlation between the highpass (*i.e.* "detail") subbands of the green and red channels. Consider a generic 2-D filter bank shown in Figure 2, which decomposes an input image $x[\boldsymbol{n}]$ into one lowpass subband $x_{\mathrm{L}}[\boldsymbol{n}]$ and N different highpass subbands $x_{\mathrm{H}i}[\boldsymbol{n}]$ for $i = 1...N$. The sets of analysis filters $\{A_i(\boldsymbol{z})\}$ and synthesis filters $\{S_i(\boldsymbol{z})\}$ satisfy the perfect reconstruction condition

$$\sum_{i=0}^{N} A_i(\boldsymbol{z}) S_i(\boldsymbol{z}) \equiv 1. \tag{1}$$

One concrete example of such a filter bank is the 2-D undecimated wavelet transform used in Ref 4. With one level of wavelet decomposition, we obtain one lowpass channel and three highpass channels. Meanwhile, the channel filters are all separable products of 1-D filters.

Based on the observation[4] that the highpass subbands of the green and red channels of a natural image tend to be very similar, the AP algorithm proposes to search for full-resolution red images within the following "detail" constraint set:

$$\mathcal{C}_d \stackrel{\text{def}}{=} \{x[\boldsymbol{n}] : |(a_i * x)[\boldsymbol{n}] - (a_i * \widehat{g})[\boldsymbol{n}]| \le T, \text{ for } 1 \le i \le N\}, \tag{2}$$

where $a_i[\boldsymbol{n}]$ is the ith highpass analysis filter in Figure 2, $\widehat{g}[\boldsymbol{n}]$ is the previously estimated full-resolution green channel, and T is a threshold indicating how "similar" the two signals should be. In practice, it is often sufficient to set T to zero, which works well when the red and green channels are strongly correlated.

To enforce the constraint defined in (2) (with $T = 0$), the AP algorithm employs a detail update operator $P_d : \ell^2(\mathbb{Z}^2) \longmapsto \ell^2(\mathbb{Z}^2)$, defined in the transform domain as

$$y[\boldsymbol{n}] = (P_d\, x)[\boldsymbol{n}] \;\Rightarrow\; Y(\boldsymbol{z}) = S_0(\boldsymbol{z})\,(A_0(\boldsymbol{z})X(\boldsymbol{z})) + \sum_{i=1}^{N} S_i(\boldsymbol{z}) \left(A_i(\boldsymbol{z})\widehat{G}(\boldsymbol{z})\right). \tag{3}$$

The above formulation can be intuitively understood by referring to Figure 2. For any input image $X(\boldsymbol{z})$, the operator P_d takes the lowpass subband of X [*i.e.* $A_0(\boldsymbol{z})X(\boldsymbol{z})$] and the highpass subbands of the previously estimated green channel $\widehat{G}$ [*i.e.* $A_i(\boldsymbol{z})\widehat{G}(\boldsymbol{z})$], and combine them through the synthesis filter bank to get the output $Y(\boldsymbol{z})$.

The formula (3) for the detail update operator can be simplified as follows. Let

$$H(\boldsymbol{z}) \stackrel{\text{def}}{=} S_0(\boldsymbol{z})A_0(\boldsymbol{z}). \tag{4}$$

It then follows from the perfect reconstruction property (1) of the filter bank that

$$1 - H(\boldsymbol{z}) = \sum_{i=1}^{N} A_i(\boldsymbol{z}) S_i(\boldsymbol{z}). \tag{5}$$

Substituting (4) and (5) into (3), we get

$$y[\boldsymbol{n}] = (P_d\, x)[\boldsymbol{n}] \;\Rightarrow\; Y(\boldsymbol{z}) = H(\boldsymbol{z})X(\boldsymbol{z}) + (1 - H(\boldsymbol{z}))\widehat{G}(\boldsymbol{z}). \tag{6}$$

Compared with the original definition (3) for P_d, the proposed formula in (6) only requires the filtering of $X(\boldsymbol{z})$ and $\widehat{G}(\boldsymbol{z})$ in the lowpass channel of the filter bank.

The second constraint that the AP algorithm employs is based on the available sensor measurements $s[\boldsymbol{n}]$. More specifically, the interpolated red channel should belong to the following "observation" constraint set

$$\mathcal{C}_o \stackrel{\text{def}}{=} \{x[\boldsymbol{n}] : x[\boldsymbol{n}] = s[\boldsymbol{n}],\ \text{for } \boldsymbol{n} \in \Lambda_r\}, \tag{7}$$

where $\Lambda_r \stackrel{\text{def}}{=} \{\boldsymbol{n} = (2k_1, 2k_2 + 1) : \text{for all } k_1, k_2 \in \mathbb{Z}\}$ represents the locations of the red pixels in the Bayer CFA shown in Figure 1.

To enforce the above constraint, we can define the observation update operator $P_o : \ell^2(\mathbb{Z}^2) \longmapsto \ell^2(\mathbb{Z}^2)$ as follows

$$y[\boldsymbol{n}] = (P_o\, x)[\boldsymbol{n}] = \begin{cases} s[\boldsymbol{n}] & \text{if } \boldsymbol{n} \in \Lambda_r, \\ x[\boldsymbol{n}] & \text{otherwise.} \end{cases} \tag{8}$$

In words, the operator P_o replaces the values of $x[\boldsymbol{n}]$ at the red pixel locations with the available sensor measurements, but leaves the rest of the pixels intact.

After defining the two update operators P_o and P_d, we can now summarize the main iterations of the AP algorithm as follows:

Algorithm 1 Interpolate the Missing Red Pixels by Alternating Projections[4]

Input: The sensor measurement $s[\boldsymbol{n}]$ from the Bayer CFA, and the estimated green channel $\widehat{g}[\boldsymbol{n}]$.
Output: An estimated full-resolution red channel $\widehat{r}[\boldsymbol{n}]$ of size M-by-N.

Use bilinear interpolation to obtain an initial estimate $r^{(0)}[\boldsymbol{n}]$ of the red channel.
Initialize the iteration number: $k \leftarrow 0$
repeat
  Enforce the detail constraint by having $r^{(k+0.5)}[\boldsymbol{n}] = (P_d\, r^{(k)})[\boldsymbol{n}]$.
  Enforce the observation constraint by having $r^{(k+1)}[\boldsymbol{n}] = (P_o\, r^{(k+0.5)})[\boldsymbol{n}]$.
  $k \leftarrow k + 1$
until The mean squared error MSE $\stackrel{\text{def}}{=} \|r^{(k)} - r^{(k-1)}\|^2/(MN)$ is smaller than a given threshold δ.
return $\widehat{r}[\boldsymbol{n}] = r^{(k)}[\boldsymbol{n}]$

2.2 Problem Statement

Starting from an initial estimate $r^{(0)}[\boldsymbol{n}]$, the AP algorithm described above generates a sequence of updated estimates $\{r^{(1)}[\boldsymbol{n}], r^{(2)}[\boldsymbol{n}], r^{(3)}[\boldsymbol{n}], \ldots\}$, where

$$r^{(k+1)}[\boldsymbol{n}] = (P_o\, P_d\, r^{(k)})[\boldsymbol{n}], \quad \text{for } k = 0, 1, 2, \ldots$$

Numerical experiments indicate that the above iterative procedure converges within a small tolerance (MSE $< \delta = 0.2$) after about 5 to 8 iterations. In Ref. 4, the authors attribute this desirable convergent property to the classical *alternating projection theorem*. We first recall the following facts about projections onto closed convex sets (POCS).

DEFINITION 2.1. *Let $\mathcal{C}$ be a closed convex set in a Hilbert space $\mathcal{H}$. For any $x \in \mathcal{H}$, there exists a* unique *element $y_x \in \mathcal{C}$ such that*

$$\|x - y_x\| \leq \|x - z\|, \quad \textit{for all } z \in \mathcal{C}.$$

We call the mapping $P_{\mathcal{C}} : \mathcal{H} \longmapsto \mathcal{H}, P_{\mathcal{C}}\, x \stackrel{def}{=} y_x$ the projection *operator onto $\mathcal{C}$.*

THEOREM 2.2 (VON NEUMANN[9]). *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two closed convex sets in a Hilbert space $\mathcal{H}$, and $P_{\mathcal{C}_1}$ and $P_{\mathcal{C}_2}$ the corresponding projection operators. Suppose that $\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$. For any $x^{(0)} \in \mathcal{H}$, the sequence $\left\{x^{(k+1)} : x^{(k+1)} = P_{\mathcal{C}_1} P_{\mathcal{C}_2} x^{(k)}\right\}_{k=0,1,2,\ldots}$ converges to the projection of $\boldsymbol{x}^{(0)}$ onto $\mathcal{C}_1 \cap \mathcal{C}_2$,* i.e.,

$$\lim_{k\to\infty} \|x^{(k)} - P_{\mathcal{C}_1 \cap \mathcal{C}_2} x^{(0)}\| = 0.$$

To invoke the above theorem in the context of the AP algorithm, one can easily verify that the two constraint sets $\mathcal{C}_o$ and $\mathcal{C}_d$ defined in Section 2.1 are closed and convex. Meanwhile, the observation operator P_o in (8) is indeed the projection (*i.e.* best approximation) onto $\mathcal{C}_o$. However, this is in general not the case for the detail update operator P_d.

PROPOSITION 1. *P_d is a projection operator* if and only if

$$H(e^{j\boldsymbol{\omega}}) = \mathbb{1}_{\mathcal{D}}(\boldsymbol{\omega}), \tag{9}$$

where $H(e^{j\boldsymbol{\omega}})$ is the filter defined in (4)*, and $\mathbb{1}_{\mathcal{D}}(\boldsymbol{\omega})$ is the indicator function defined on the frequency domain support $\mathcal{D}$ of $H(e^{j\boldsymbol{\omega}})$.*

Proof. To show the necessity of (9), we suppose that P_d is a projection. Recall that a well-known property of projection operators is *idempotence*, *i.e.*, $P_d P_d = P_d$. From the definition of P_d in (6), we must have, for any input image $X(e^{j\boldsymbol{\omega}})$,

$$H\left(HX + (1-H)\widehat{G}\right) + (1-H)\widehat{G} = HX + (1-H)\widehat{G}. \tag{10}$$

Note that we omit the argument $e^{j\boldsymbol{\omega}}$ for simplicity of notation. It follows from (10) that

$$H\,(1-H)(\widehat{G} - X) = 0.$$

Since the above equality holds for all possible input X, we must have

$$H(e^{j\boldsymbol{\omega}})(1 - H(e^{j\boldsymbol{\omega}})) = 0,$$

which implies that $H(e^{j\boldsymbol{\omega}})$ has binary-valued ideal frequency responses, *i.e.*, $H(e^{j\boldsymbol{\omega}}) = 1$ for $\boldsymbol{\omega}$ in the passband $\mathcal{D}$ and $H(e^{j\boldsymbol{\omega}}) = 0$ otherwise. The sufficiency of (9) is shown in Ref. 6. □

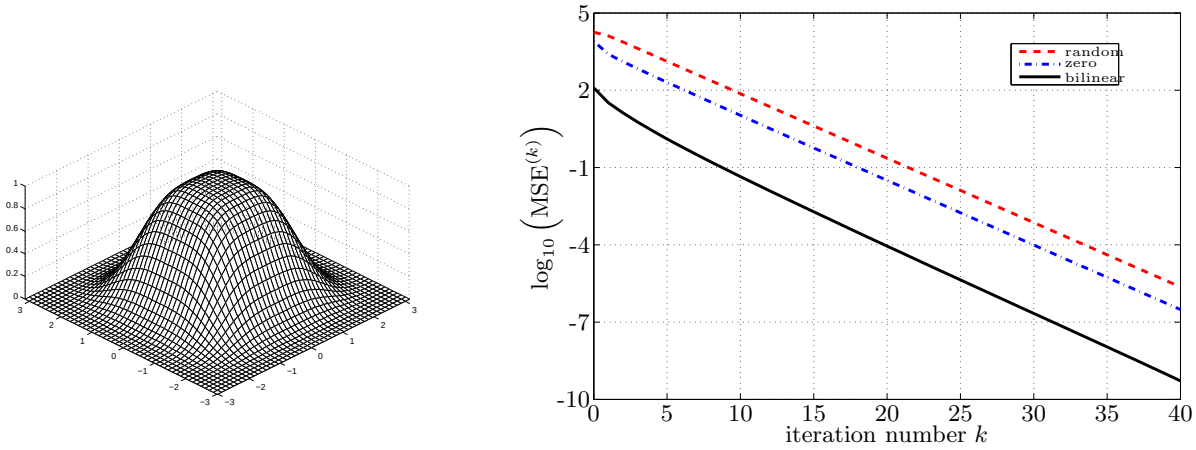
REMARK 1. *Proposition 1 states that, for P_d to be a projection operator, the lowpass filter $H(e^{j\boldsymbol{\omega}})$ must be ideal in the frequency domain. Consequently, $h[\boldsymbol{n}]$ has to be a sinc-like filter in the spatial domain, and cannot have a finite impulse response (FIR).*

EXAMPLE 1. *In Ref. 4, the authors use the following separable filters in the wavelet filter bank*

$$a_0[\boldsymbol{n}] = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 2 & 1\end{bmatrix}^T, \textit{ and } \; s_0[\boldsymbol{n}] = \frac{1}{64}\begin{bmatrix}-1 & 2 & 6 & 2 & -1\end{bmatrix} \otimes \begin{bmatrix}-1 & 2 & 6 & 2 & -1\end{bmatrix}^T. \tag{11}$$

We show in Figure 3(a) the magnitude frequency response $|H(e^{j\boldsymbol{\omega}})| = |S_0(e^{j\boldsymbol{\omega}})A_0(e^{j\boldsymbol{\omega}})|$. Evidently, since both $a_0[\boldsymbol{n}]$ and $s_0[\boldsymbol{n}]$ are FIR filters, $|H(e^{j\boldsymbol{\omega}})|$ is nonideal and does not satisfy (9)*, and thus P_d is not a projection operator. Nevertheless, we know from numerical experiments that the AP algorithm based on this filter still converges after a small number of iterations.*

EXAMPLE 2. *Another interesting fact about the AP algorithm is that the final result of the iteration process appears to be independent of the starting point. To demonstrate this phenomenon, we apply the AP algorithm to the standard test image "light house" from the Kodak set, with three different initial estimates for $r^{(0)}[\boldsymbol{n}]$:*



(a) (b)

Figure 3. (a) The magnitude frequency response of the lowpass filter $H(e^{j\boldsymbol{\omega}})$ used in Ref. 4. It has a smooth transition band, and hence does not satisfy (9). (b) The convergence of the AP algorithm. The different choices of the initial estimates $r^{(0)}[\boldsymbol{n}]$ (solid line: bilinear interpolation, dash-dot line: all zero, dashed line: random) only affect the speed of convergence, but not the final convergence value.

bilinear interpolation as in Ref. 4, all zeros, and i.i.d. random numbers uniformly distributed from 0 *to* 255. *For each initial estimate, we carry out the iteration process as described in Algorithm 1 and calculate*

$$\log_{10}\left(MSE^{(k)}\right) \stackrel{def}{=} \log_{10}\left(\|r^{(k)}[\boldsymbol{n}] - \widehat{r}_{BL}[\boldsymbol{n}]\|^2/(MN)\right),$$

where $r^{(k)}[\boldsymbol{n}]$ *is the estimated red channel at the* k*th iteration,* $\widehat{r}_{BL}[\boldsymbol{n}]$ *is the convergence value obtained by choosing bilinear interpolation as the starting point. The results are shown in Figure 3(b). We see that the different choices of the initial estimates only affect the speed of convergence, but all three processes eventually converge to the same result* $\widehat{r}_{BL}[\boldsymbol{n}]$.

The two examples presented above prompt us to ask the following questions:

1. We know that when $H(e^{j\boldsymbol{\omega}})$ does not have an ideal frequency response (which is always the case in practice), the detail update operator P_d is not a projection. What is then a rigorous explanation (other than the POCS) for the convergence of the AP algorithm?
2. For convergence, what conditions do we need to impose on $H(e^{j\boldsymbol{\omega}})$?
3. The convergence value of the iteration process appears to be unique, and does not depend on the starting points. Can we then directly reach the convergence result of the AP algorithm without going through iterations?

The remaining parts of this paper center around the rigorous analysis of the above questions. In particular, we answer the first two questions in Section 3 based on the contraction mapping theorem. The third question is addressed in Section 4, where we propose an efficient noniterative implementation of the AP algorithm.

3. THE CONVERGENCE OF THE AP ALGORITHM

In this section, we provide a rigorous analysis of the convergent property of the AP algorithm. Our theoretical analysis will be heavily based on the *polyphase representation*[10,11] of signals. To facilitate readers who are not familiar with this concept, we start our discussion by having a brief overview of this useful signal representation.

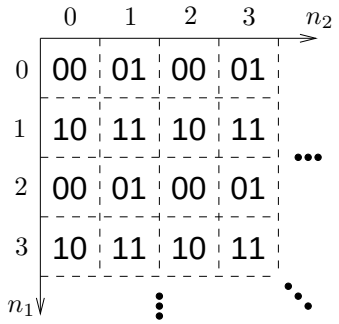


Figure 4. The sampling locations of the four polyphase components of a 2-D signal.

3.1 The Polyphase Representation

The polyphase representation (transform) is a powerful tool in multirate signal processing. It also plays a critical role in the derivations presented in later parts of this paper. For a 2-D image $x[\boldsymbol{n}]$, we can decompose it into four *polyphase components* $x_{00}[\boldsymbol{n}], x_{01}[\boldsymbol{n}], x_{10}[\boldsymbol{n}]$, and $x_{11}[\boldsymbol{n}]$, defined as

$$x_{i,j}[n_1, n_2] \stackrel{\text{def}}{=} x[2n_1 + i, 2n_2 + j], \quad \text{for } i, j \in \{0, 1\}. \tag{12}$$

As illustrated in Figure 4, the polyphase components in (12) are simply downsampled versions of the original signal $x[\boldsymbol{n}]$. The sampling locations of all four polyphase components form a partition. Note that the mapping between the signal $x[\boldsymbol{n}]$ and its polyphase components is one-to-one. To go back to the original signal from its polyphase components, we can easily verify that

$$X(\boldsymbol{z}) = X_{00}(\boldsymbol{z}^2) + z_2^{-1} X_{01}(\boldsymbol{z}^2) + z_1^{-1} X_{10}(\boldsymbol{z}^2) + z_1^{-1} z_2^{-1} X_{11}(\boldsymbol{z}^2). \tag{13}$$

The polyphase representation becomes especially handy when we try to describe the periodic sampling structures of color filter arrays. For example, the sampling patterns of the Bayer CFA (see Figure 1) can be succinctly described in the polyphase domain as follows:

$$s_{00}[\boldsymbol{n}] = g_{00}[\boldsymbol{n}], \; s_{01}[\boldsymbol{n}] = r_{01}[\boldsymbol{n}], \; s_{10}[\boldsymbol{n}] = b_{10}[\boldsymbol{n}], \text{ and } s_{11}[\boldsymbol{n}] = g_{11}[\boldsymbol{n}],$$

where $s[\boldsymbol{n}]$ represents the sensor measurements.

To describe the filtering operation in the polyphase domain, consider two signals $x[\boldsymbol{n}], y[\boldsymbol{n}]$ such that

$$y[\boldsymbol{n}] = (h * x)[\boldsymbol{n}], \tag{14}$$

for some filter $h[\boldsymbol{n}]$. Denote by

$$\boldsymbol{x}_p[\boldsymbol{n}] \stackrel{\text{def}}{=} [x_{00}[\boldsymbol{n}], x_{01}[\boldsymbol{n}], x_{10}[\boldsymbol{n}], x_{11}[\boldsymbol{n}]]^T \quad \text{and} \quad \boldsymbol{X}_p(\boldsymbol{z}) \stackrel{\text{def}}{=} [X_{00}(\boldsymbol{z}), X_{01}(\boldsymbol{z}), X_{10}(\boldsymbol{z}), X_{11}(\boldsymbol{z})]^T \tag{15}$$

the vector of all four polyphase components of $x[\boldsymbol{n}]$, and the corresponding vector in the transform domain, respectively. Similarly, we can define $\boldsymbol{y}_p[\boldsymbol{n}]$ and $\boldsymbol{Y}_p(\boldsymbol{z})$ for $y[\boldsymbol{n}]$ as well.

LEMMA 3.1. *The filtering operation* (14) *can be described in the polyphase domain as*

$$\boldsymbol{Y}_p(\boldsymbol{z}) = \boldsymbol{H}(\boldsymbol{z})\, \boldsymbol{X}_p(\boldsymbol{z}), \tag{16}$$

where

$$\boldsymbol{H}(\boldsymbol{z}) \stackrel{def}{=} \begin{bmatrix} H_{00}(\boldsymbol{z}) & H_{01}(\boldsymbol{z}) z_2^{-1} & H_{10}(\boldsymbol{z}) z_1^{-1} & H_{11}(\boldsymbol{z}) z_1^{-1} z_2^{-1} \\ H_{01}(\boldsymbol{z}) & H_{00}(\boldsymbol{z}) & H_{11}(\boldsymbol{z}) z_1^{-1} & H_{10}(\boldsymbol{z}) z_1^{-1} \\ H_{10}(\boldsymbol{z}) & H_{11}(\boldsymbol{z}) z_2^{-1} & H_{00}(\boldsymbol{z}) & H_{01}(\boldsymbol{z}) z_2^{-1} \\ H_{11}(\boldsymbol{z}) & H_{10}(\boldsymbol{z}) & H_{01}(\boldsymbol{z}) & H_{00}(\boldsymbol{z}) \end{bmatrix}, \tag{17}$$

and $\{H_{00}, H_{01}, H_{10}, H_{11}\}$ *are the polyphase components of the filter* H*.*

At this point, it may seem that going to the polyphase domain only makes the filtering operation more complicated. As we shall see below however, the primary advantage in using the polyphase notation is that it can convert the *shift-variant* operator $P_o P_d$ used in the AP algorithm to a multiple-input-multiple-output system of *shift-invariant* (filtering) operations. The added complexity of dealing with matrix-vector multiplications [such as in (16) and (17)] are more than compensated for by the convenience of working with shift-invariant operators.

3.2 Simplifying the AP Algorithm in the Polyphase Domain

Equipped with the tool of polyphase representation, we can now express the iterative procedure of the AP algorithm

$$r^{(k+1)}[\boldsymbol{n}] = (P_o\, r^{(k+0.5)})[\boldsymbol{n}] = (P_o\, P_d\, r^{(k)})[\boldsymbol{n}], \quad \text{for } k = 0, 1, 2, \ldots$$

in a simplified form. To that end, we first introduce the following change of variables

$$c^{(k)}[\boldsymbol{n}] \stackrel{\text{def}}{=} r^{(k)}[\boldsymbol{n}] - \widehat{g}[\boldsymbol{n}] \quad \text{and} \quad c^{(k+0.5)}[\boldsymbol{n}] \stackrel{\text{def}}{=} r^{(k+0.5)}[\boldsymbol{n}] - \widehat{g}[\boldsymbol{n}], \tag{18}$$

where $\widehat{g}[\boldsymbol{n}]$ is the estimated green channel. We refer to $\left\{c^{(k)}[\boldsymbol{n}], c^{(k+0.5)}[\boldsymbol{n}]\right\}$ the "chrominance" signals, as they are the differences between the red and green channels. Note that studying $c^{(k)}[\boldsymbol{n}]$ is equivalent to studying $r^{(k)}[\boldsymbol{n}]$, because $\widehat{g}[\boldsymbol{n}]$ is estimated before—and does not change during—the iteration process.

As shown in the following lemma, the main advantage in working with the chrominance signals in (18) is the simplification of the detail update operator P_d from the form of (6) to a *single filtering* operation.

LEMMA 3.2. *After the detail update step at the* k*th iteration, we have*

$$C^{(k+0.5)}(\boldsymbol{z}) = H(\boldsymbol{z})\, C^{(k)}(\boldsymbol{z}), \tag{19}$$

where $H(\boldsymbol{z})$ *is the filter defined in* (4)*.*

Next, we focus on the observation update operator P_o, which can also be conveniently described in the polyphase domain.

LEMMA 3.3. *After the observation update step at the* k*th iteration, we have*

$$c_{ij}^{(k+1)}[\boldsymbol{n}] = \begin{cases} s_{01}[\boldsymbol{n}] - \widehat{g}_{01}[\boldsymbol{n}] & \text{if } i = 0, j = 1 \\ c_{ij}^{(k+0.5)}[\boldsymbol{n}] & \text{otherwise.} \end{cases} \tag{20}$$

We see from (20) that only three of the polyphase components $(c_{00}^{(k)}[\boldsymbol{n}], c_{10}^{(k)}[\boldsymbol{n}], c_{11}^{(k)}[\boldsymbol{n}])$ are ever modified during the iteration process (due to the filtering operation in P_d), whereas the fourth one, $c_{01}^{(k)}[\boldsymbol{n}]$, always remains the same. This observation implies that we only need to study the evolution and convergence of the following vector

$$\boldsymbol{c}_v^{(k)}[\boldsymbol{n}] \stackrel{\text{def}}{=} \left[c_{00}^{(k)}[\boldsymbol{n}], c_{10}^{(k)}[\boldsymbol{n}], c_{11}^{(k)}[\boldsymbol{n}]\right]^T \tag{21}$$

in the iteration process. As concatenations of the three varying polyphase components, $\left\{\boldsymbol{c}_v^{(k)}[\boldsymbol{n}]\right\}$ defined above are elements of the following Cartesian product space

$$\ell_3^2 \stackrel{\text{def}}{=} \ell^2(\mathbb{Z}^2) \times \ell^2(\mathbb{Z}^2) \times \ell^2(\mathbb{Z}^2).$$

THEOREM 3.4. *At every iteration of the AP algorithm, the new estimate* $\boldsymbol{c}_v^{(k+1)}[\boldsymbol{n}]$ *is obtained from the previous one* $\boldsymbol{c}_v^{(k)}[\boldsymbol{n}]$ *through a* fixed *mapping* $T : \ell_3^2 \longmapsto \ell_3^2$*, defined in the transform domain as*

$$\boldsymbol{C}_v^{(k+1)} = \left(T\boldsymbol{C}_v^{(k)}\right)(\boldsymbol{z}) \stackrel{def}{=} \boldsymbol{T}(\boldsymbol{z})\, \boldsymbol{C}_v^{(k)}(\boldsymbol{z}) + \boldsymbol{B}(\boldsymbol{z}), \tag{22}$$

where $\boldsymbol{T}(\boldsymbol{z})$ *is a matrix obtained from* $\boldsymbol{H}(\boldsymbol{z})$ *in* (17) *after removing its 2nd row and 2nd column, and*

$$\boldsymbol{B}(\boldsymbol{z}) \stackrel{def}{=} [H_{01}(\boldsymbol{z})z_2^{-1}, H_{11}(\boldsymbol{z})z_2^{-1}, H_{10}(\boldsymbol{z})]^T(S_{01}(\boldsymbol{z}) - \widehat{G}_{01}(\boldsymbol{z})). \tag{23}$$

Proof. Let $\boldsymbol{S} = [\boldsymbol{e}_1, \boldsymbol{e}_3, \boldsymbol{e}_4]$ be a constant 4×3 matrix, where $\boldsymbol{e}_i$ is the ith standard basis vector of $\mathbb{R}^4$. According to Lemma 3.3, we have

$$\boldsymbol{C}_v^{(k+1)}(\boldsymbol{z}) = \boldsymbol{C}_v^{(k+0.5)}(\boldsymbol{z}) = \boldsymbol{S}^T\boldsymbol{C}_p^{(k+0.5)}(\boldsymbol{z}) = \boldsymbol{S}^T\,\boldsymbol{H}(\boldsymbol{z})\,\boldsymbol{C}_p^{(k)}(\boldsymbol{z}), \tag{24}$$

where the second equality can be verified from the definition of $\boldsymbol{c}_v[\boldsymbol{n}]$ in (21), and the last equality is due to Lemma 3.2. Again, from the definition of $\boldsymbol{c}_v[\boldsymbol{n}]$ in (21), we can write

$$\boldsymbol{C}_p^{(k)}(\boldsymbol{z}) = \boldsymbol{S}\,\boldsymbol{C}_v^{(k)}(\boldsymbol{z}) + [0,1,0,0]^T C_{01}^{(k)}(\boldsymbol{z}).$$

Substituting the above equality into (24), we reach

$$\boldsymbol{C}_v^{(k+1)}(\boldsymbol{z}) = \boldsymbol{S}^T\,\boldsymbol{H}(\boldsymbol{z})\,\boldsymbol{S}\,\boldsymbol{C}_v^{(k)}(\boldsymbol{z}) + \boldsymbol{S}^T\,\boldsymbol{H}(\boldsymbol{z})\,[0,1,0,0]^T C_{01}^{(k)}(\boldsymbol{z})$$

Using the fact that $C_{01}^{(k)}(\boldsymbol{z}) \equiv S_{01}(\boldsymbol{z}) - \widehat{G}_{01}(\boldsymbol{z})$ for all k, we get (22) after some straightforward calculations. □

3.3 Convergence by Contraction Mapping

We are now ready to present one of the key results of this work: the convergence of the AP algorithm by contraction mapping. First, we recall the following facts that are relevant to our discussions.

Definition 3.5. *A mapping T from a Hilbert space $\mathcal{H}$ to itself is call* Lipschitz continuous*, if there is some real number $\alpha > 0$ such that*

$$\|T(x_1) - T(x_2)\| \le \alpha\|x_1 - x_2\|, \quad \textit{for all } x_1, x_2 \in \mathcal{H}.$$

The smallest such value of α, denoted by α_T, is called the Lipschitz constant *of T. Furthermore, if $\alpha_T < 1$, then T is a* contraction mapping*.*

Theorem 3.6 (Contraction Mapping Theorem[5]). *Let $T : \mathcal{H} \longmapsto \mathcal{H}$ be a contraction mapping with Lipschitz constant $0 < \alpha_T < 1$.*

1. *The mapping T admits one and only one fixed point $\widehat{x}$,* i.e.*, $T(\widehat{x}) = \widehat{x}$.*
2. *For arbitrary $x^{(0)} \in \mathcal{H}$, the iterated sequence $x^{(k+1)} = T(x^{(k)})$ always converges to $\widehat{x}$.*
3. *The speed of convergence is bounded by the following inequality*

$$\|x^{(k)} - \widehat{x}\| \le \frac{(\alpha_T)^k}{1-\alpha_T}\,\|x^{(0)} - \widehat{x}\|.$$

Now we just need to check that the mapping T as defined in (22) is indeed a contraction. If that is the case, then the convergence of the AP algorithm will be automatically guaranteed by the contraction mapping theorem stated above. The following proposition presents a simple way to obtaining the Lipschitz constant of T.

Proposition 2. *The Lipschitz constant of the mapping T defined in* (22) *can be calculated as*

$$\alpha_T = \max_{\boldsymbol{\omega}} \sigma_{\max}\left(\boldsymbol{T}(e^{j\boldsymbol{\omega}})\right), \tag{25}$$

where $\sigma_{\max}(\cdot)$ denotes the largest singular value of a matrix.

In practice, the Lipschitz constant in (25) can be estimated as follows. Suppose that all the polyphase filters in (17) are FIR, and of sizes up to $L \times L$. For any of these filters (*e.g.* $h_{00}[\boldsymbol{n}]$), we can zero-pad the filter to size $N \times N$ (with $N \ge L$) and then apply a 2-D discrete Fourier transform (via FFT). Doing so yields the Fourier

transform $H_{00}(e^{j\boldsymbol{\omega}})$ on a discrete grid $\{\boldsymbol{\omega} = (k_1, k_2)/N, \text{ for } 0 \le k_1, k_2 < N\}$. Thus, we can obtain the following estimate

$$\alpha_T(N) = \max_{\boldsymbol{\omega}=\frac{2\pi}{N}(k_1,k_2)} \sigma_{\max}\left(\boldsymbol{T}(e^{j\boldsymbol{\omega}})\right),$$

which approaches the true quantity in (25) as N goes to infinity.

EXAMPLE 3. *Consider the filters proposed in Ref. 4, whose coefficients are given in Example 1. Choosing* $N = 2048$ *leads to the estimate*

$$\alpha_T \approx \alpha_T(N) = 0.75 < 1.$$

By applying Theorem 3.6, we know that the AP algorithm using this filter will always converge. Meanwhile, the convergence value is unique, and does not depend on the starting point, which corresponds to our previous observation described in Example 2.

So far, we have used the polyphase representation and the contraction mapping theorem to address the first two questions listed at the end of Section 2.2, namely, a rigorous explanation for the convergent property, and the corresponding condition for the filter $h[\boldsymbol{n}]$ used in the iteration process. In the following section, we address the third question by proposing an efficient algorithm that directly obtain the convergence result of the AP algorithm, without going through iterations.

4. AN EFFICIENT NONITERATIVE IMPLEMENTATION OF THE AP ALGORITHM

4.1 General Schemes

We know from Theorem 3.6 that, if the AP algorithm is convergent, then it must converge to the *unique* fixed point of the mapping T. Let $\widehat{\boldsymbol{C}}_v(\boldsymbol{z})$ denote the transform domain representation of the fixed point. Applying the definition of the mapping T in Theorem 3.4, we obtain

$$\widehat{\boldsymbol{C}}_v(\boldsymbol{z}) = \left(T\,\widehat{\boldsymbol{C}}_v\right)(\boldsymbol{z}) = \boldsymbol{T}(\boldsymbol{z})\,\widehat{\boldsymbol{C}}_v(\boldsymbol{z}) + \boldsymbol{B}(\boldsymbol{z}).$$

The above equality implies that

$$\begin{aligned}
\widehat{\boldsymbol{C}}_v(\boldsymbol{z}) &= (1 - \boldsymbol{T}(\boldsymbol{z}))^{-1}\,\boldsymbol{B}(\boldsymbol{z}) \\
&= (1 - \boldsymbol{T}(\boldsymbol{z}))^{-1}\,[H_{01}(\boldsymbol{z})z_2^{-1}, H_{11}(\boldsymbol{z})z_2^{-1}, H_{10}(\boldsymbol{z})]^T(S_{01}(\boldsymbol{z}) - \widehat{G}_{01}(\boldsymbol{z})), \qquad (26) \\
&\stackrel{\text{def}}{=} [F_{00}(\boldsymbol{z}), F_{10}(\boldsymbol{z}), F_{11}(\boldsymbol{z})]^T(S_{01}(\boldsymbol{z}) - \widehat{G}_{01}(\boldsymbol{z})) \qquad (27)
\end{aligned}$$

where (26) is due to (23). Recall that $\widehat{\boldsymbol{C}}_v = [\widehat{C}_{00}(\boldsymbol{z}), \widehat{C}_{10}(\boldsymbol{z}), \widehat{C}_{11}(\boldsymbol{z})]^T$ is a concatenation of the three polyphase components of the chrominance signal $\widehat{c}[\boldsymbol{n}]$. Consequently, (27) means that

$$\widehat{C}_{ij}(\boldsymbol{z}) = F_{ij}(\boldsymbol{z})\left(S_{01}(\boldsymbol{z}) - \widehat{G}_{01}(\boldsymbol{z})\right), \quad \text{where } i, j \in \{(0,0), (1,0), (1,1)\}.$$

In words, the final convergence result of the AP algorithm can be directly obtained by three filtering operations in the polyphase domain. Meanwhile, as shown in Lemma 3.3, the remaining polyphase component, $\widehat{C}_{10}(\boldsymbol{z})$, is equal to $S_{01}(\boldsymbol{z}) - \widehat{G}_{01}(\boldsymbol{z})$. We summarize the block diagram of the proposed noniterative implementation of the AP algorithm in Figure 5.

4.2 Numerical Experiments

To demonstrate the performance of the proposed scheme shown in Figure 5, we apply both the proposed non-iterative algorithm and the original AP algorithm to the 24 standard Kodak test images. For the proposed algorithm, the three polyphase filters $F_{00}(\boldsymbol{z})$, $F_{10}(\boldsymbol{z})$, and $F_{11}(\boldsymbol{z})$ defined in (27) are generally not FIR. However, these filters can be well-approximated by their finitely-truncated versions. In our experiments, we truncated the filters to the size of 7×7. For the AP algorithm, we test two different options for the maximum number of

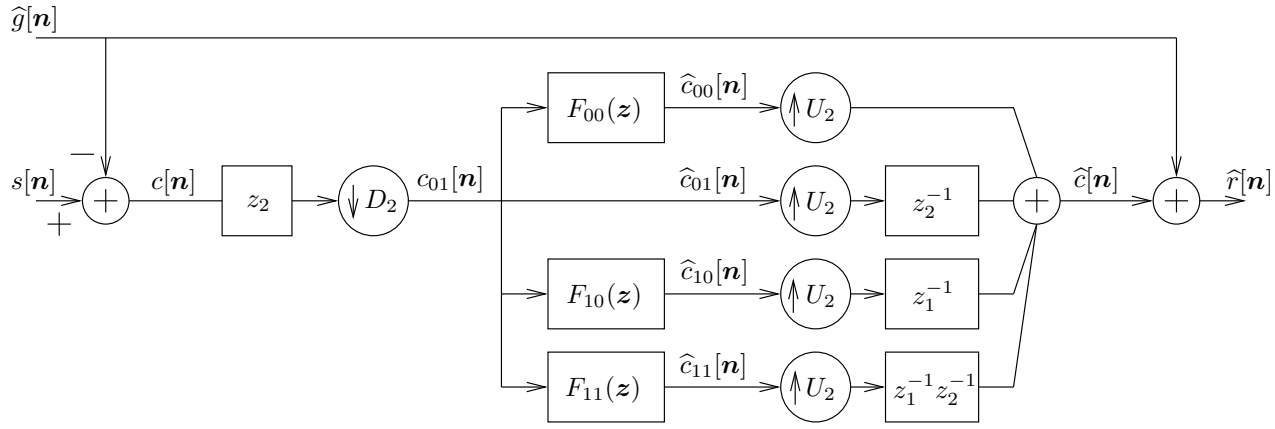


Figure 5. The block diagram of the proposed noniterative implementation of the AP algorithm. The estimated full-resolution red channel $\widehat{r}[\boldsymbol{n}]$ can be obtained by applying three filtering operations in the polyphase domain. In the figure, D_2 represents downsampling by 2 along both dimensions, and U_2 represents upsampling by 2 along both dimensions.

iterations: 15 and 5. The former will ensure us to obtain the convergence value of the procedure, while the latter is more commonly used in practice due to its reduced computational load.

We implement both algorithms in Matlab*, and the experiments are carried out on a computer with a 2.2 GHz CPU. Table 1 summarizes the main results, averaged over the 24 test images. We measure the accuracy of the algorithms in terms of the peak signal-to-noise ratio (PSNR) of the demosaicked images. Note that the PSNR values for the green channel are the same for different algorithms, since both AP and the proposed scheme follow the same steps in estimating the missing green pixels. For the red and blue pixels, the PSNR values obtained by the proposed scheme are very close to those reached by AP after 15 iterations. This indicates that our noniterative algorithm can accurately compute the final convergence values of the AP procedure, even though the polyphase filters used in our scheme are truncated approximations. It is also interesting to notice that applying the AP algorithm with only 5 iterations can actually generate better results in terms of PSNR, though the differences are quite small.

The main advantage of the proposed algorithm is its computational efficiency. As shown in the table, we divide the total running time into two parts: the initialization stage (obtaining the green channel) and the refinement stage (obtaining the red and blue channels). For the former, the proposed algorithm implements the simplified formula (6) for the detail update operator P_d, which contributes to reducing the running time by a factor of 3 (from 0.388 s to 0.125 s). More substantial improvements can be found in the refinement stage: the original AP algorithm approaches the reconstruction problem via iteration, whereas the proposed scheme achieves the same goal by direct filtering operations in the polyphase domain. Correspondingly, the running time can be dramatically reduced. For example, compared with the AP algorithm (with 5 iterations), the proposed noniterative scheme reduces the time spent on the refinement stage by a factor of 13, and reduces the total running time by a factor of 9. As we see from the case of AP with 15 iterations, the speedup will become even more significant when the number of iterations in the AP algorithm increases.

5. CONCLUSIONS

We presented in this paper a detailed treatment of a classical color image demosaicking algorithm based on alternating projections. On the theoretical side, our analysis provides a rigorous foundation for the convergence of the AP algorithm, based on the contraction mapping theorem. On the practical side, we propose an efficient noniterative implementation of the AP algorithm in the polyphase domain. Numerical experiments confirm that the proposed algorithm can achieve the same results obtained by the iterative AP procedure at convergence, but is significantly faster than the latter.

*For the AP algorithm, we use the Matlab code provided by the authors of Ref. 4.

Table 1. Performance comparison of the original AP algorithm and the proposed noniterative implementation. We exclude 10 pixels along each side of the border in calculating the PSNR.

	PSNR (dB)			CPU Time (second)		
	Red	Green	Blue	Initialization	Refinement Stage	Total
AP (5 iterations)	38.47	41.81	38.60	0.388	2.138	2.526
AP (15 iterations)	38.44	41.81	38.48	0.388	6.267	6.656
Proposed Scheme	38.43	41.81	38.47	0.125	0.161	0.286

REFERENCES

[1] Ramanath, R., Snyder, W. E., Bilbro, G. L., and Sander III, W. A., "Demosaicking methods for Bayer color arrays," *J. Electron. Imag.* **11**, 306–315 (Jul. 2002).

[2] Gunturk, B. K., Glotzbach, J., Altunbasak, Y., Schafer, R. W., and Mersereau, R. M., "Demosaicking: Color filter array interpolation," *IEEE Signal Process. Mag.* **22**, 44–54 (Jan. 2005).

[3] Li, X., Gunturk, B., and Zhang, L., "Image demosaicing: A systematic survey," in [*Visual Communications and Image Processing 2008*], Pearlman, W. A., Woods, J. W., and Lu, L., eds., *Proc. SPIE* **6822** (Jan. 2008).

[4] Gunturk, B. K., Altunbasak, Y., and Mersereau, R. M., "Color plane interpolation using alternating projections," *IEEE Trans. Image Process.* **11**, 997–1013 (Sep. 2002).

[5] Luenberger, D. G., [*Optimization by Vector Space Methods*], John Wiley & Sons (1969).

[6] Lu, Y. M., Karzand, M., and Vetterli, M., "Iterative demosaicking accelerated: Convergence, optimality, and noniterative fast implementations." to be submitted to *IEEE Trans. Image Process.* (2009).

[7] Bayer, B. E., "Color imaging array." U.S. Patent 3971065 (Jul. 1976).

[8] Hamilton, J. F. and Adams, J. E., "Adaptive color plane interpolation in single sensor color electronic camera." U.S. Patent 5 629 734 (May 1997).

[9] Deutsch, F., [*Best approximation in inner product spaces*], Springer-Verlag, New York, NY (2001).

[10] Vaidyanathan, P. P., [*Multirate Systems and Filter Banks*], Prentice-Hall, Englewood Cliffs, NJ (1993).

[11] Vetterli, M. and Kovačević, J., [*Wavelets and Subband Coding*], Prentice Hall, Englewood Cliffs, NJ (1995).