Published in Image Processing On Line on 2023–06–28.
Submitted on 2022–10–22, accepted on 2023–05–22.
ISSN 2105–1232 © 2023 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.2023.443

Fast Chromatic Aberration Correction with 1D Filters

Thomas Eboli

Université Paris-Saclay, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
thomas.eboli@ens-paris-saclay.fr

Communicated by C. Hessel, L. Raad and G. Facciolo *Demo edited by* J. Anger and T. Eboli

Abstract

This article presents an implementation of the chromatic aberration correction technique of Chang et al. [Correction of Axial and Lateral Chromatic Aberration with False Color Filtering, IEEE Transactions on Image Processing, 2013]. This method decomposes aberration correction into a cascade of two 1D filters. The first one locally sharpens the red and blue edges such that they have similar profiles to that of the green channel serving as guiding image throughout restoration. The second one shifts the red and blue corrected edges to the location of the green ones to remove the color fringes. These two successive estimates are ultimately merged into a final prediction, free of most chromatic aberrations.

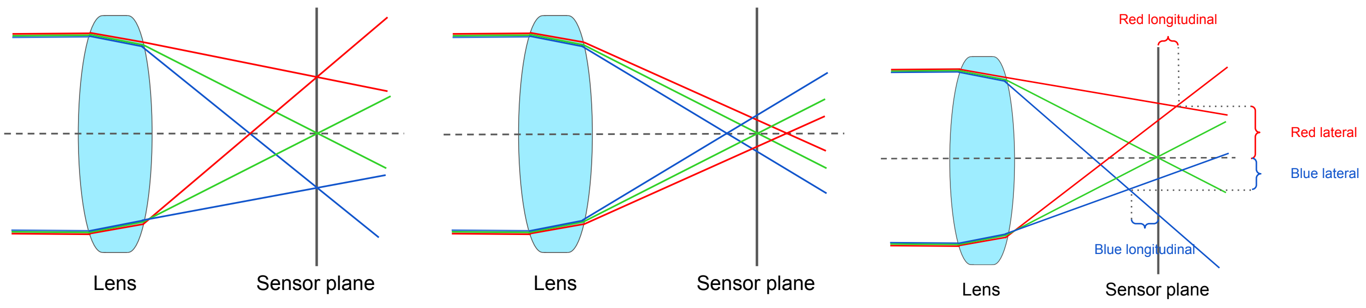
Source Code

An implementation of the algorithm, together with an online demo, are available at the associated web page[1]. The code is implemented in Cython to accelerate the implementation of the separable filters. The latest version of the code can be found at the Github repository[2].

Keywords: chromatic aberration; separable filters

1 Introduction

Optical aberrations (OA) are unavoidable artifacts caused by the refraction of the light rays inside the thick lens put in front of the camera sensor. In particular, chromatic aberrations (CA) are the resulting unnatural color artifacts in the photograph caused by the wavelength-dependent paths of the rays within the lens. These latter analog phenomena come in two sorts: lateral aberrations causing a color point to have its red, green and blue components being focused at different *locations* on the sensor plane, and the axial aberrations causing a color point to have its red, green and blue components being focused at different *depths* along the optical axis. In the captured photograph, the first causes color fringes next to the most salient edges, and is mostly located next to the corners, i.e., the farther from the optical center. The latter generates color-dependent blur everywhere on



(a) Lateral aberration. (b) Axial aberration. (c) Combined aberrations.

Figure 1: Illustration of the chromatic aberrations in photographs. From left to right: a lateral aberration shifting the colors *foci* on the sensor plane, an axial aberration shifting the colors *foci* along the optical axis, and a combination of both displacing the red and blue *foci* with 3D shifts. We assume the sensor plane corresponds to that of the focus of the green image.

the camera field-of-view. Figure 1 illustrates the two chromatic aberrations and the combination of both, which is what is usually observed in the photographs.

The absence of the OA, in particular the CA, is an important criterion to assess of the visual quality of a photograph. Their compensation can be achieved by using top-of-the-line lenses that may be bulky and/or expensive, or by postprocessing algorithms in the digital camera pipeline, a more accessible option for most photographers. An interesting line of work to OA removal that leverages the properties of the aberrations consists in decomposing the correction into sharpening/blind deblurring to compensate the loss of contrast [3, 8, 9, 5], and CA compensation by inverse warping or edge correction [1, 7, 4].

In this article, we present the algorithmic solution of [2] targeting CA correction from a single RGB image via edge-aware filtering. It consists in a series of two 1D filters applied sequentially to the rows and columns of the image. They are designed to first make the red and blue edge profiles fit that of the green, thus compensating local color-specific blurs, and, second, correct the differences of color magnitudes next to the salient edges leading to color fringes. Since the second filtering stage may over-compensate the color fringes by turning into gray color gradients that are actually part of the scene, a final arbitration module combines the outputs of the two filtering stages into a final image with correct color fringe correction. Figures 2, 3 and 4 are examples throughout the paper of what can be expected after each stage of the algorithm.

We will strive throughout this article to alleviate confusions and ambiguities in the original paper that made the reimplementation of this method tedious, especially because no official code is available online. We hope this endeavor will facilitate future possible extensions based on this work.

The rest of this paper is organized as follows. In Section 2 we present the different stages of the method, in Section 3 we present several qualitative results and discuss the strengths and limits of the method, and finally in Section 4 we conclude this article.

2 Method

The presented technique first applies two 1D $2L + 1$ filters: *transient improvement* (TI) and *false color* (FC) filtering sequentially in the horizontal and vertical directions. The filter radius L is the main parameter that should be tuned prior to running the algorithm, and may be different for the horizontal and the vertical passes. We derive each individual equation throughout the section, and summarize in Algorithm 5 the full approach.

[1]https://doi.org/10.5201/ipol.2023.443

[2]https://github.com/teboli/chromatic_aberration_filtering

2.1 Transient Improvement Filter

The first stage is a kind of local edge sharpener for the red and blue channels R_{in} and B_{in} with the corresponding green one G_{in} acting as a guide. Indeed, the widely-used Bayer pattern has twice the amount of green pixels than red or blue ones, and the three color channels are focused at different depths on the optical axis because of the wavelength-dependent refraction inside the lens. Thus, most of the camera manufacturers make sure that the green channel is focused on the sensor plane. The red and blue channels are thus focused at slightly different depths, causing additional defocus blur for these two color channels [10].

In this stage, red and blue filtering are done in two intermediate steps. Since the red and blue color corrections are the same and independent, the filter is applied to these two channels separately. In the rest of this article, we will dub X either the red R or blue B channel to write once the equations that are the same for these two colors. In this note, we will have to handle both 2D images and 1D subarrays centered at each pixel location (i, j) in the image domain (that we also refer by "1D slices" in the rest of the paper). We respectively call X the image, and $\widehat{X}$ the $2L + 1$ 1D slices within.

As for any window-based approach, a technical question is about the edges of the images. Nothing is said in the original paper so we simply adopt a "valid" approach: We restore the inner sub-image cropped by L on each side and simply copy-paste the missing edges from the original image. Other edge-handling approaches may be adopted if needs be, without changing the core of the method.

The first step in TI filtering consists in computing a prefiltered image $2L + 1$ horizontal slice, defined for a location (i, j) in the input, and for ℓ in the set $\{-L, \dots, L\}$

$$\widehat{X}^h_{pf}(\ell) = \begin{cases} \rho_0 X^h_{max}(i,j) + \rho_1 X_{in}(i, j+\ell) + \rho_2 X^h_{min}(i,j) & \text{if } X_{in}(i,j) > G_{in}(i,j), \\ \rho_0 X^h_{min}(i,j) + \rho_1 X_{in}(i, j+\ell) + \rho_2 X^h_{max}(i,j) & \text{otherwise,} \end{cases} \tag{1}$$

with predefined weights ρ_0, ρ_1 and ρ_2 summing to 1 to preserve the local mean value. In the original paper [2], these values are set to -0.25, 1.375 and -0.125 respectively. The uneven shape of this local filter is deemed important by the authors of the original paper to efficiently remove colored fringes in practice. For instance in a blue fringe region, the dot product leads to $0.25B_{max}$ to be larger than $0.125B_{min}$, thus to compensate the blue color, we have to give more importance to the removal of B_{max}. In the opposite case where the green color is more important than the blue one, removing instead $0.125B_{max}$ and $0,25B_{min}$ increase the value of the blue component, thus also turning the pixel color into a more achromatic value. The explanation in the original paper is not clear so we simply replicated their filter for the sake of reimplementation.

In this equation and in the rest of the paper, the index ℓ has negative values, whereas we should query the values in the array $\widehat{X}^h_{pf}$ with indices in $\{0, 2L\}$ instead. We use this slight abuse of notation the make all the presentation more concise and consistent with the queried pixel location $(i, j + \ell)$ in the neighborhood of (i, j).

In the equation above, $X^h_{min}(i, j)$ and $X^h_{max}(i, j)$ are local minimum and maximum values in the image 1D slice of size $2L + 1$ centered at pixel location (i, j). These local extrema are computed as follows. We first compute local minimum and maximum values on half windows of size $L + 1$ to the left and the right of the pixel location (i, j) (included in both windows), and second select a max/min pair that maximizes the contrast between the left and right parts of the 1D slice, i.e., the maximum contrast at pixel location (i,j) in the horizontal direction. Formally, for ℓ in $\{-L,\dots,L\}$

$$X_{E,max} = \max_{0\le\ell\le L} X_{in}(i,j+\ell), \tag{2a}$$

$$X_{E,min} = \min_{0\le\ell\le L} X_{in}(i,j+\ell), \tag{2b}$$

$$X_{W,max} = \max_{-L\le\ell\le 0} X_{in}(i,j+\ell), \tag{2c}$$

$$X_{W,min} = \min_{-L\le\ell\le 0} X_{in}(i,j+\ell), \tag{2d}$$

$$\begin{pmatrix} X^h_{max} \\ X^h_{min} \end{pmatrix} = \begin{cases} \begin{pmatrix} X_{E,max} \\ X_{W,min} \end{pmatrix} & \text{if } X_{E,max} - X_{W,min} \ge X_{W,max} - X_{E,min} \\ \begin{pmatrix} X_{W,max} \\ X_{E,min} \end{pmatrix} & \text{otherwise.} \end{cases} \tag{2e}$$

With these two local extrema, and after computing the prefiltered image slice with Equation (1), we proceed to the second step of TI filtering. We clip the entries of $\widehat{X}^h_{pf}$ accordingly with the limits set by X^h_{min}, X^h_{max} and the guiding green image G_{in}. The thresholds are tailored for each pixel location (i,j) and for ℓ in $\{-L,\dots,L\}$ as

$$\begin{pmatrix} \widehat{X}^h_{TI,max}(\ell) \\ \widehat{X}^h_{TI,min}(\ell) \end{pmatrix} = \begin{cases} \begin{pmatrix} X_{in}(i,j+\ell) \\ \max(X^h_{min}, G_{in}(i,j+\ell)) \end{pmatrix} & \text{if } X_{in}(i,j) > G_{in}(i,j), \\ \begin{pmatrix} \min(X^h_{max}, G_{in}(i,j+\ell)) \\ X_{in}(i,j+\ell) \end{pmatrix} & \text{otherwise.} \end{cases} \tag{3}$$

The values in the prefiltered slice X_{pf} are finally clipped to prevent over and under-shooting, and mimic the green channel's edge profile, yielding the TI horizontally filtered signal centered in (i,j) for ℓ in $\{-L,\dots,L\}$

$$\widehat{X}^h_{TI}(\ell) = \begin{cases} \widehat{X}^h_{TI,max}(\ell) & \text{if } \widehat{X}^h_{pf}(\ell) > \widehat{X}^h_{TI,max}(\ell) \\ \widehat{X}^h_{TI,min}(\ell) & \text{if } X^h_{pf}(\ell) < \widehat{X}^h_{TI,min}(\ell) \\ \widehat{X}^h_{pf}(\ell) & \text{otherwise.} \end{cases} \tag{4}$$

The central value of this $2L+1$ image slice, is stored at the location (i,j) in the 2D array $X^h_{TI}(i,j) = \widehat{X}^h_{TI}(0)$ (or X^v_{TI} and $\widehat{X}^v_{TI}$ during the vertical pass) that will be used in the final arbitration module. Recall that $\widehat{X}^h_{TI}$ is a function of the location (i,j). We drop this dependency in what follows for conciseness' sake. We now detail how the TI-filtered slices are further processed with the FC filter.

This subroutine is summarized in Algorithm 1 illustrating the pseudo-code for the `TransientImprovement1D` function.

In prevision of the arbitration stage in Section 2.3, we have to compute the images X^v_{TI} and X^h_{TI}. After each call to `TransientImprovement1D` which returns the $2L+1$ 1D slices $\widehat{X}^v_{TI}$ and $\widehat{X}^h_{TI}$ centered at the pixel location (i,j), we store in arrays X^v_{TI} and X^h_{TI} at the location (i,j) the central value of the slice which is the locally filtered version of the channel X_{in}

$$X^h_{TI}(i,j) = \widehat{X}^h_{TI}(0). \tag{5}$$

This operation is repeated to the vertical directional image X^v_{TI} from the slices $\widehat{X}^v_{TI}$. Recall that we vary ℓ in $\{-L,\dots,L\}$, thus the index of the central value in the slice is 0.

2.2 False Color Filter

The second stage further refines the TI results by getting rid of the remaining color fringes next to the most contrasted edges. In [2], the resulting color fringes are called "false colors", for which a

Algorithm 1: `TransientImprovement1D`

```
input : Color image $X_{in}$, Green image $G_{in}$, Radius $L$, Pixel location $(i,j)$, Coefficients
        $(\rho_0, \rho_1, \rho_2)$
output: TI horizontally filtered $2L+1$ 1D slice $\widehat{X}^h_{TI}$, Local horizontal minimum $X^h_{min}(i,j)$,
        Local horizontal maximum $X^h_{max}(i,j)$
$\widehat{X}^h_{pf} = \texttt{zeros}(2L+1)$                                        ▷ Initialize
$\widehat{X}^h_{TI} = \texttt{zeros}(2L+1)$                                        ▷ Initialize
$X_{E,max} = \max_{0 \leq \ell \leq L} X_{in}(i, j+\ell)$                    ▷ Max on the right (Equation (2a))
$X_{E,min} = \min_{0 \leq \ell \leq L} X_{in}(i, j+\ell)$                    ▷ Min on the right (Equation (2b))
$X_{W,max} = \max_{-L \leq \ell \leq 0} X_{in}(i, j+\ell)$                   ▷ Max on the left (Equation (2c))
$X_{W,min} = \min_{-L \leq \ell \leq 0} X_{in}(i, j+\ell)$                   ▷ Min on the left side(Equation (2d))
if $X_{E,max} - X_{W,min} > X_{W,max} - X_{E,min}$ then
    $X^h_{max}(i,j) = X_{E,max}$                                   ▷ Select best max/min pair (Equation (2e))
    $X^h_{min}(i,j) = X_{W,min}$
else
    $X^h_{max}(i,j) = X_{W,max}$
    $X^h_{min}(i,j) = X_{E,min}$
if $X_{in}(i,j) > G_{in}(i,j)$ then
    for $\ell \in \{-L, \dots, L\}$ do
        $\widehat{X}_{pf}(\ell) = \rho_0 X^h_{max}(i,j) + \rho_1 X_{in}(i,j+\ell) + \rho_2 X^h_{min}(i,j)$    ▷ Update prefiltered signal
        (Equation (1))
        $\widehat{X}^h_{TI,max}(\ell) = X_{in}(i,j+\ell)$
        $\widehat{X}^h_{TI,min}(\ell) = \max\big(X^h_{min}(i,j), G_{in}(i,j+\ell)\big)$          ▷ Compute local limits (Equation (3))
else
    for $\ell \in \{-L, \dots, L\}$ do
        $\widehat{X}_{pf}(\ell) = \rho_0 X^h_{min}(i,j) + \rho_1 X_{in}(i,j+\ell) + \rho_2 X^h_{max}(i,j)$    ▷ Update prefiltered signal
        (Equation (1))
        $\widehat{X}^h_{TI,max}(\ell) = \min\big(X^h_{max}(i,j), G_{in}(i,j+\ell)\big)$
        $\widehat{X}^h_{TI,min}(\ell) = X_{in}(i,j+\ell)$                              ▷ Compute local limits (Equation (3))
for $\ell \in \{-L, \dots, L\}$ do
    if $\widehat{X}^h_{pf}(\ell) > \widehat{X}^h_{TI,max}(\ell)$ then
        $\widehat{X}^h_{TI}(\ell) = \widehat{X}^h_{TI,max}(\ell)$                         ▷ Prediction clipping (Equation (4))
    else if $\widehat{X}^h_{pf}(\ell) < \widehat{X}^h_{TI,min}(\ell)$ then
        $\widehat{X}^h_{TI}(\ell) = \widehat{X}^h_{TI,min}(\ell)$
    else
        $\widehat{X}^h_{TI}(\ell) = \widehat{X}^h_{pf}(\ell)$
```

numerical definition is given. Chang et al. claim that in the false color regions: (i) the gradients of each RGB channel is large since we look at salient edges, and (ii) the color differences between the channels are large since the color-specific edges are not at the same locations. The FC filter discussed below leverages these two remarks to efficiently get rid of the color fringes.

In this section again, we present only the process along the horizontal direction since in the vertical direction it is exactly the same. In practice, the FC 1D filter is applied to transformed values of the RGB images, converted to a luma-chroma domain. The chroma signals are in practice cheap and effective detectors of the CA [7, 4]. Indeed, an edge in a natural image is mostly a contrasted interface between two regions of different colors and intensity. The slopes of the transitions may vary for each color but they should be at the same locations for the three channels. The color fringes are caused by the color-specific displacement of this interface between two regions, leading to the unnatural larger and brutal variations in the chroma signals that are thus good detectors of CA.

As a result, at the beginning if this stage the RGB signal produced by the TI earlier step is first converted within a certain luma-chroma domain. In [2], the authors chose the following simple conversion

$$\begin{bmatrix} G_{in}(i,j+\ell) \\ \widehat{K}^h_{b,TI}(\ell) \\ \widehat{K}^h_{r,TI}(\ell) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \widehat{R}^h_{TI}(\ell) \\ G_{in}(i,j+\ell) \\ \widehat{B}^h_{TI}(\ell) \end{bmatrix}. \tag{6}$$

It leaves untouched the green signal that can be seen as the luma signal in the decomposition since it serves as a guide of the image geometry in this work. It also separately blends the green image with either the red or the blue one, which are the corresponding chroma signals of the decomposition $\widehat{K}^h_{r,TI}(\ell)$ and $\widehat{K}^h_{b,TI}(\ell)$, with ℓ in $\{-L,\dots,L\}$. Recall that these image slices are implicit functions of (i,j) with our notations.

The horizontal filter applied to the $\widehat{K}^h_{r,TI}$ or $\widehat{K}^h_{b,TI}$ component, simply dubbed $\widehat{K}^h_{TI}$ since the following steps are the same for the two chroma images, takes the form at any location (i,j) of a local average

$$K^h_{FC}(i,j) = \frac{\sum_{\ell=-L}^{L} w(\ell) c\left(\widehat{K}^h_{TI}(\ell)\right)}{\sum_{\ell=-L}^{L} w(\ell)}, \tag{7}$$

where K^h_{FC} is the horizontally-filtered image after false color correction. In this equation, the w's are the coefficients of a $2L+1$ 1D filter specific at the location (i,j) in the image domain, and c is a clipping function defined as

$$c\left(\widehat{K}^h_{TI}(\ell)\right) = \begin{cases} \min(\widehat{K}^h_{TI}(\ell), \widehat{K}^h_{TI}(0)), & \text{if } \widehat{K}^h_{TI}(0) > 0, \\ \max(\widehat{K}^h_{TI}(\ell), \widehat{K}^h_{TI}(0)), & \text{if } \widehat{K}^h_{TI}(0) < 0, \\ \widehat{K}^h_{TI}(0) & \text{otherwise.} \end{cases} \tag{8}$$

The $2L+1$ filter's coefficients read, for ℓ in $\{-L,\dots,L\}$ and every pixel location (i,j)

$$w(\ell) = \frac{s(\ell)}{|\nabla_h G_{in}(i,j+\ell)| + |Y_{in}(i,j+\ell) - Y_{in}(i,j)| + D_X(\ell)}, \tag{9}$$

where

$$s(\ell) = \begin{cases} 1 & \text{if } [\text{sign}(\widehat{K}^h_{TI}(0)) = \text{sign}(\widehat{K}^h_{TI}(\ell))] \text{ or } |\widehat{K}^h_{TI}(\ell)| < \tau, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

and

$$D_X(\ell) = \max(|\nabla_h X_{in}(i,j+\ell)|, \alpha_X |\widehat{K}^h_{TI}(\ell)|). \tag{11}$$

As stated above for the slices $\widehat{X}^h_{TI}$, the weights w in the merging scheme (7) are dependent on the local position (i,j). In the equations above, $\nabla_h G_{in}$ is the horizontal gradient of the input green channel, $Y_{in} = 0.299R_{in} + 0.587G_{in} + 0.114B_{in}$ is the luma channel of I_{in}. In the original paper, no details are given regarding the computation of the gradients $\nabla_h G$ and $\nabla_h X_{in}$. We simply used a finite-difference scheme corresponding to the convolution with filters $[-1, 1]$ and $[-1, 1]^\top$. This simple approach leads to satisfactory enough visual results in practice.

The binary weight s discards, from the averaging in Equation (7), the pixels in the 1D slice that have chroma values deemed too different from that of the pixel of interest when their signs are different. Yet, the nearly achromatic pixels with possibly a different sign, i.e., K^h_{TI} is close to 0 up to the threshold τ, are also retained. With this scheme, if a pixel is accepted by s, its weight is then inversely proportional to the finite differences or the gradients of the guide images G_{in} and Y_{in}, and the chroma contrasts estimated either by the image $\widehat{K}^h_{TI}$ or the gradients of X_{in}, i.e., large variations of the colors and geometry. Thus, the more different the color of a neighboring pixel is, the smaller w will be, which corresponds to rejecting the pixels that are parts of the false color regions as claimed by the definition we gave earlier.

Lastly, α_X is a pre-defined scalar introduced by the authors to scale the ℓ-th entry of the TI-filtered slice $\widehat{K}^h_{TI}$. In the original paper, the authors handtuned the values for the red and blue channels to be respectively $\alpha_R = 0.5$ and $\alpha_B = 1.0$. The decision of choosing different values for each color is not explained, we simply employ this strategy to be as faithful as possible to the original work, but other values may also be valid in practice. We have not explored such direction during our reimplementation.

Filtering in the horizontal direction K^h_{TI} and vertical direction K^v_{TI} ultimately leads to two images K^h_{FC} and K^v_{FC}. The final corrected chroma image K_{FC} is built by selecting the most achromatic pixel, i.e., selecting the most possible aggressive color correction, at each location (i, j) among the vertical and horizontal images. Formally, it reads

$$K_{FC}(i, j) = \min(|K^h_{FC}(i, j)|, |K^v_{FC}(i, j)|). \tag{12}$$

Applied to the red and blue chroma components, Equation (7) to (12) yield two filtered chroma images $K_{r,FC}$ and $K_{b,FC}$. In the next section, we present how to efficiently merge the predictions of the TI and FC stages into the final estimate.

This subroutine is summarized in Algorithm 2 illustrating the pseudo-code for the `FalseColor1D` function.

2.3 Arbitration

Since the FC stage may over-compensate the correction of color fringes, leading to too-achromatic edges where in the latent signal there are actually colored elements, we must combine the results from the TI and FC stages into a single prediction with reasonable chroma signals. This stage is called "arbitration" in the original paper [2].

First, the TI stage results in two independently filtered images K^h_{TI} and K^v_{TI}, via the slice-to-image relationship in Equation (5), that are further gathered into a single frame in a manner similar to Equation (12) for all pixel location (i, j)

$$K_{TI}(i, j) = \min(|K^h_{TI}(i, j)|, |K^v_{TI}(i, j)|), \tag{13}$$

As a result, K_{TI} is an image with the most achromatic content from the vertical and horizontal independent estimates. Similar to the previous section, K_{TI} refers indistinctly to $K_{r,TI}$ or $K_{b,TI}$, i.e., the chroma images predicted by the TI stage.

For the sake of the arbitration process, we also have to build the local minima and maxima images X_{max} and X_{min} from the directional images X^v_{max}, $X^v min$, X^h_{max} and X^h_{min}. It is not detailed in the

Algorithm 2: `FalseColor1D`

```
input : Color input $X_{in}$, Green input $G_{in}$, Luma input $Y_{in}$, TI color slice $\widehat{X}^h_{TI}$, Radius $L$,
        Pixel location $(i,j)$, Color weight $\alpha_X$, Contrast threshold $\tau$
output: FC horizontally filtered entry $K^h_{FC}(i,j)$
$s = \mathtt{zeros}(2L+1)$                                          ▷ Initialize
for $\ell \in \{-L,\dots,L\}$ do
    $\widehat{K}^h_{TI}(\ell) = \widehat{X}^h_{TI}(\ell) - G_{in}(i,j+\ell)$      ▷ Color conversion (Equation (6))
for $\ell \in \{-L,\dots,L\}$ do
    if $[sign(\widehat{K}^h_{TI}(0)) = sign(\widehat{K}^h_{TI}(\ell))]$ or $|\widehat{K}^h_{TI}(\ell)| < \tau$ then
        $s(\ell) = 1$                                   ▷ Update binary rejection (Equation (10))
    $D_X(\ell) = \max\left(|\nabla_h X_{in}(i,j+\ell)|, \alpha_X|\widehat{K}^h_{TI}(\ell)|\right)$   ▷ Local maximum color contrast (Equation (11))
    $w(\ell) = \frac{s(\ell)}{(|\nabla_h G_{in}(i,j+\ell)| + Y_{in}(i,j+\ell) - Y_{in}(i,j) + D_X(\ell))}$   ▷ Build weights (Equation (9))
    if $\widehat{K}^h_{TI}(0) > 0$ then
        $c(K^h_{TI}(\ell)) = \min\left(\widehat{K}^h_{TI}(\ell), \widehat{K}^h_{TI}(0)\right)$   ▷ Input clipping (Equation (8))
    else if $\widehat{K}^h_{TI}(0) < 0$ then
        $c(K^h_{TI}(\ell)) = \max\left(\widehat{K}^h_{TI}(\ell), \widehat{K}^h_{TI}(0)\right)$
    else
        $c(K^h_{TI}(\ell)) = \widehat{K}^h_{TI}(\ell)$
$K^h_{FC}(i,j) = \sum_{\ell=-L}^{L} w(l)c(K^h_{TI}(\ell)) / \sum_{\ell=-L}^{L} w(\ell)$   ▷ Local averaging (Equation (7))
```

original publication, and we propose the simple following approach that works well in practice for all pixel location (i,j)

$$X_{min}(i,j) = \min\left(X^v_{min}(i,j), X^h_{min}(i,j)\right), \tag{14a}$$

$$X_{max}(i,j) = \max\left(X^v_{max}(i,j), X^h_{max}(i,j)\right). \tag{14b}$$

Based on these intermediate fused images, the arbitration takes the form of a pixelwise convex sum of the TI and FC results

$$K_{out}(i,j) = (1-\alpha_K(i,j))K_{TI}(i,j) + \alpha_K(i,j)K_{FC}(i,j). \tag{15}$$

The fusion scheme is indeed simple, the bulk of this part being computing the local weights α_K that are specific to each location (i,j) in the image domain. These coefficients measure the contributions of the images FC and TI based on which image has the most important contrasts to prevent grayish colors, particularly favored by the FC stage.

The arbitration process, through the design of α_K, leverages two additional properties of false color regions: (iii) the false color regions are located next to the most contrasted edges, and (iv) the false colors are mostly spotted next to the latent achromatic edges. Property (iii) differs from (i) since it states that the aberrations will be even more pronounced next to the edges that contain enough high frequencies in their spectra. Consequently, an extra care must be given to these edges preserved from blur. Property (iv) claims that the color fringes are more visible at edges where the interface between two regions is at the same location for the three channels, otherwise we cannot as easily tell if the color fringes are caused by the signal or the aberrations. Chang et al. [2] thus proposed a weight function of the contrast, and the extrema previously computed during the TI stage that further enforces these two additional properties. Recall that the two first ones resulted in K_{FC}. The weights α_K leveraging properties (iii) and (iv) are defined as

$$\alpha_K(i,j) = \min\left(\frac{\max(X_{contrast}(i,j),0)}{\min(\max(X_{max}(i,j)-X_{min}(i,j),\gamma_2),\gamma_1)},1\right), \tag{16}$$

where $X_{max}(i,j)$ and $X_{min}(i,j)$ are the pointwise respective maximum and minimum values of the pairs $\big(X^h_{max}(i,j),X^v_{max}(i,j)\big)$, and $\big(X^h_{min}(i,j),X^v_{min}(i,j)\big)$, computed during the TI stage in Equation (2e). The clipping values γ_1 and γ_2 are scalars set in the original paper to 0.5 and 0.25 respectively.

The $X_{contrast}$ image is obtained with similar local max/min computations shown below for all pixel location (i,j)

$$X'_{E,max} = \max_{0\le\ell\le L} X_{in}(i,j+\ell) - \beta_X|X_{in}(i,j+\ell)-G_{in}(i,j+\ell)|, \tag{17a}$$

$$X'_{E,min} = \min_{0\le\ell\le L} X_{in}(i,j+\ell) + \beta_X|X_{in}(i,j+\ell)-G_{in}(i,j+\ell)|, \tag{17b}$$

$$X'_{W,max} = \max_{-L\le\ell\le 0} X_{in}(i,j+\ell) - \beta_X|X_{in}(i,j+\ell)-G_{in}(i,j+\ell)|, \tag{17c}$$

$$X'_{W,min} = \min_{-L\le\ell\le 0} X_{in}(i,j+\ell) + \beta_X|X_{in}(i,j+\ell)-G_{in}(i,j+\ell)|, \tag{17d}$$

$$X^h_{contrast}(i,j) = \begin{cases} X'_{E,max}-X'_{W,min} & \text{if } X'_{E,max}-X'_{W,min} \ge X'_{W,max}-X'_{E,min} \\ X'_{W,max}-X'_{E,min} & \text{otherwise,}\end{cases} \tag{17e}$$

In the equations above, and in contrast to Equations (2a), (2b), (2c) and (2d), the local computations of the max and min values on the half windows now features regularization weights β_R and β_B considering the differences of intensities between the input green and red/blue channels. These additional terms in the half-windowed local max/min values are here to measure if a local signal is actually an achromatic edge. Similarly to Equation (2e), a pair of max/min values is chosen to detect the maximum contrast between the left and right-parts of the 1D slice in Equation (17e), resulting in the horizontal contrast image $X^h_{contrast}$. Repeated for all the locations but with slices on the vertical direction yields the companion vertical image $X^v_{contrast}$.

The two images $X^h_{contrast}$ and $X^v_{contrast}$ are finally merged by selecting the larger value of contrast between the two directions for all pixel location (i,j) as follows

$$X_{contrast}(i,j) = \begin{cases} X^h_{contrast}(i,j) & \text{if } X^h_{contrast}(i,j) > X^v_{contrast}(i,j), \\ X^v_{contrast}(i,j) & \text{otherwise.}\end{cases} \tag{18}$$

This blended image is finally fused into Equation (16) to predict the local weights $\alpha_K(i,j)$. After applying the convex sum of Equation (15), the arbitrated image is converted back to the RGB format for all pixel location (i,j)

$$\begin{bmatrix} R_{out}(i,j) \\ G_{out}(i,j) \\ B_{out}(i,j)\end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0\end{bmatrix} \begin{bmatrix} G_{in}(i,j) \\ K_{b,out}(i,j) \\ K_{r,out}(i,j)\end{bmatrix}, \tag{19}$$

which yields the final prediction.

This subroutine is summarized in Algorithm 4 illustrating the pseudo-code for the `Arbitration` function. We also present in Algorithm 3 the pseudo-code of the companion subroutine dubbed `Contrast1D` for computing the contrast array $X^h_{contrast}$.

Algorithm 3: `Contrast1D`

```
input  : Input color $X_{in}$, Green color $G_{in}$, Gradient weight $\beta_X$, Radius $L$, Pixel location $(i,j)$
output: Directional contrast entry $X^h_{contrast}(i,j)$
$X_{E,max} = \max_{0\leq\ell\leq L} X_{in}(i,j+\ell) - \beta_X|X_{in}(i,j+\ell) - G_{in}(i,j+\ell)|$   ▷ Max on the right (Equation (17a))
$X_{E,min} = \min_{0\leq\ell\leq L} X_{in}(i,j+\ell) + \beta_X|X_{in}(i,j+\ell) - G_{in}(i,j+\ell)|$   ▷ Min on the right (Equation (17b))
$X_{W,max} = \max_{-L\leq\ell\leq 0} X_{in}(i,j+\ell) - \beta_X|X_{in}(i,j+\ell) - G_{in}(i,j+\ell)|$   ▷ Max on the left (Equation (17c))
$X_{W,min} = \min_{-L\leq\ell\leq 0} X_{in}(i,j+\ell) + \beta_X|X_{in}(i,j+\ell) - G_{in}(i,j+\ell)|$   ▷ Min on the left (Equation (17d))
$X^h_{contrast}(i,j) = \max(X_{E,max} - X_{W,min}, X_{W,max} - X_{E,min})$   ▷ Select best max/min pair (Equation (17e))
```

Algorithm 4: `Arbitration`

```
input  : Input color $X_{in}$, Green color $G_{in}$, TI image $K_{TI}$, FC image $K_{FC}$, Local maxima
         $X_{max}$, Local minima $X_{min}$, Gradient weight $\beta_X$, Contrast coefficients $(\gamma_1, \gamma_2)$, Window
         radii $(L_{hor}, L_{ver})$
output: Arbitrated image $K_{out}$
$X_{contrast}$ = zeros_like($X_{in}$)   ▷ Initialize
$K_{out}$ = zeros_like($X_{in}$)   ▷ Initialize
foreach pixel position $(i,j)$ do
    /* Computing the directional contrast arrays (Algorithm 3) */
    $X^h_{contrast}(i,j)$ = Contrast1D($X_{in}, G_{in}, \beta_X, L_{hor}, (i,j)$)
    $X^v_{contrast}(i,j)$ = Contrast1D($X_{in}, G_{in}, \beta_X, L_{ver}, (i,j)$)
    /* Building pixelwise weights and arbitration */
    $X_{contrast}(i,j) = \max(X^h_{contrast}(i,j), X^v_{contrast}(i,j))$   ▷ Merge directional contrasts (Equation (18))
    $\alpha_K(i,j) = \min\left(\frac{\max(X_{contrast}(i,j),0)}{\min(\max(X_{max}(i,j)-X_{min}(i,j),\gamma_2),\gamma_1)}, 1\right)$   ▷ Compute the arbitration weight (Equation (16))
    $K_{out}(i,j) = (1-\alpha_K(i,j))K_{TI}(i,j) + \alpha_K(i,j)K_{FC}(i,j)$   ▷ Arbitration (Equation (15))
```

2.4 Full Approach

We present in Algorithm 5 the full approach, built on top of Algorithms 1, 2, 3 and 4. The full method takes as input the aberrated RGB I_{in} and auxiliary parameters for computing the 1D filters, and returns a RGB aberration-compensated variant I_{out} of the same dimension.

Algorithm 5: `Method`

```
input  : Image $I_{in}$, Horizontal size $L_{hor}$, Vertical size $L_{ver}$, Coefficients $(\rho_0, \rho_1, \rho_2)$, Contrast
         threshold $\tau$, Color weights $(\alpha_R, \alpha_B)$, Gradient weights $(\beta_R, \beta_B)$, Contrast coefficients
         $(\gamma_1, \gamma_2)$
output: Corrected image $I_{out}$
$R_{in},\ G_{in},\ B_{in} = I_{in}$                                    ▷ Extract the three channels
$Y_{in} = 0.299 R_{in} + 0.587 G_{in} + 0.114 B_{in}$        ▷ Luma (Y in YUV format) of the input for FC filtering
$K_{r,TI} =$ zeros_like$(R_{in})$                                      ▷ Initialize
$K_{r,TI} =$ zeros_like$(B_{in})$                                      ▷ Initialize
$K_{r,FC} =$ zeros_like$(R_{in})$                                      ▷ Initialize
$K_{b,FC} =$ zeros_like$(B_{in})$                                      ▷ Initialize
$R_{min} =$ zeros_like$(R_{in})$                                       ▷ Initialize
$R_{max} =$ zeros_like$(R_{in})$                                       ▷ Initialize
$B_{min} =$ zeros_like$(B_{in})$                                       ▷ Initialize
$B_{max} =$ zeros_like$(B_{in})$                                       ▷ Initialize
foreach pixel location $(i, j)$ do
    /* TI horizontal filtering (Section 2.1 and Algorithm 1) */
    $\widehat{R}^h_{TI}, R^h_{min}(i,j), R^h_{max}(i,j) =$ TransientImprovement1D $(R_{in}, G_{in}, L_{hor}, (i,j), (\rho_0, \rho_1, \rho_2))$
    $\widehat{B}^h_{TI}, B^h_{min}(i,j), B^h_{max}(i,j) =$ TransientImprovement1D $(B_{in}, G_{in}, L_{hor}, (i,j), (\rho_0, \rho_1, \rho_2))$
    /* FC horizontal filtering (Section 2.2 and Algorithm 2) */
    $K^h_{r,FC}(i,j) =$ FalseColor1D $\left(R_{in}, G_{in}, Y_{in}, \widehat{R}^h_{TI}, L_{hor}, (i,j), \alpha_R, \tau\right)$
    $K^h_{b,FC}(i,j) =$ FalseColor1D $\left(B_{in}, G_{in}, Y_{in}, \widehat{B}^h_{TI}, L_{hor}, (i,j), \alpha_B, \tau\right)$
    /* TI vertical filtering (Section 2.1 and Algorithm 1) */
    $\widehat{R}^v_{TI}, R^v_{min}(i,j), R^v_{max}(i,j) =$ TransientImprovement1D $\left(R^\top_{in}, G^\top_{in}, L_{ver}, (i,j), (\rho_0, \rho_1, \rho_2)\right)$
    $\widehat{B}^v_{TI}, B^v_{min}(i,j), B^v_{max}(i,j) =$ TransientImprovement1D $\left(B^\top_{in}, G^\top_{in}, L_{ver}, (i,j), (\rho_0, \rho_1, \rho_2)\right)$
    /* FC vertical filtering (Section 2.2 and Algorithm 2) */
    $K^v_{r,FC}(i,j) =$ FalseColor1D $\left(R^\top_{in}, G^\top_{in}, Y^\top_{in}, \widehat{R}^b_{TI}, L_{ver}, (i,j), \alpha_R, \tau\right)$
    $K^v_{b,FC}(i,j) =$ FalseColor1D $\left(B^\top_{in}, G^\top_{in}, Y^\top_{in}, \widehat{B}^v_{TI}, L_{ver}, (i,j), \alpha_B, \tau\right)$
    /* Merge horizontal and vertical TI predictions (Equation (13)) + color conversion (Equation (6)) */
    $K_{r,TI}(i,j) = \min\left(\widehat{R}^h_{TI}(0), \widehat{R}^v_{TI}(0)\right) - G_{in}(i,j)$
    $K_{b,TI}(i,j) = \min\left(\widehat{B}^h_{TI}(0), \widehat{B}^v_{TI}(0)\right) - G_{in}(i,j)$
    /* Merge horizontal and vertical local maxima and minima */
    $R_{min}(i,j) = \min\left(R^h_{min}(i,j), R^v_{min}(i,j)\right)$
    $R_{max}(i,j) = \max\left(R^h_{max}(i,j), R^v_{max}(i,j)\right)$
    $B_{min}(i,j) = \min\left(B^h_{min}(i,j), B^v_{min}(i,j)\right)$
    $B_{max}(i,j) = \max\left(B^h_{max}(i,j), B^v_{max}(i,j)\right)$
    /* Merge horizontal and vertical FC predictions (12) */
    $K_{r,FC}(i,j) = \min\left(\widehat{K}^h_{r,FC}(i,j), (\widehat{K}^v_{r,FC})^\top(i,j)\right)$
    $K_{b,FC}(i,j) = \min\left(\widehat{K}^h_{b,FC}(i,j), (\widehat{K}^v_{b,FC})^\top(i,j)\right)$
/* Arbitration stage (Section 2.3 and Algorithm 4) */
$K_{r,out} =$ Arbitration $(R_{in}, G_{in}, K_{r,TI}, K_{r,FC}, R_{max}, R_{min}, \beta_R, \gamma_1, \gamma_2, L_{hor}, L_{ver})$
$K_{b,out} =$ Arbitration $(B_{in}, G_{in}, K_{b,TI}, K_{b,FC}, B_{max}, B_{min}, \beta_B, \gamma_1, \gamma_2, L_{hor}, L_{ver})$
$I_{out} = [K_{r,out} + G_{in}, G_{in}, K_{b,out} + G_{in}]$          ▷ inverse color conversion (Equation (19))
```

3 Experiments

We run the code on the companion demo on three real-world images to illustrate the pros and cons of this technique. We also show the intermediate results of the TI and FC stages, prior to arbitration, to qualitatively monitor the impact of each step.

We run the demo with the default advanced parameters: $\tau = 0.059$, $\alpha_R = 0.5$, $\alpha_B = 1.0$, $\beta_R = 1.0$, $\beta_B = 0.25$, $\gamma_1 = 0.5$, $\gamma_2 = 0.25$, $\rho_0 = -0.25$, $\rho_1 = 1.375$ and $\rho_2 = -0.125$. Note that, in most cases, these values are relevant, and we assume the users will hardly have to tune them. The most important parameters are the window radii that have two values for the horizontal and vertical directions: L_{hor} and L_{ver}. In practice, these values are by default set to 7px and 4px, and handle well sharper color fringes caused by lateral chromatic aberrations. In case of severe aberrations, we advise to increase these values such that the correction window contains the whole support of the fringes. In all the examples, the algorithm runs in less than one second, even for several megapixel images. The algorithm is naturally parallelizable, which explains the good speed of the method.

Lateral CA correction. When there are more important lateral CA than axial CA, most of the restoration work is done by the FC stage. We show in Figures 2 and 3 two instances of almost-pure lateral CA compensation from the *Facade* and *Bridge* images from [9] and [8] respectively.

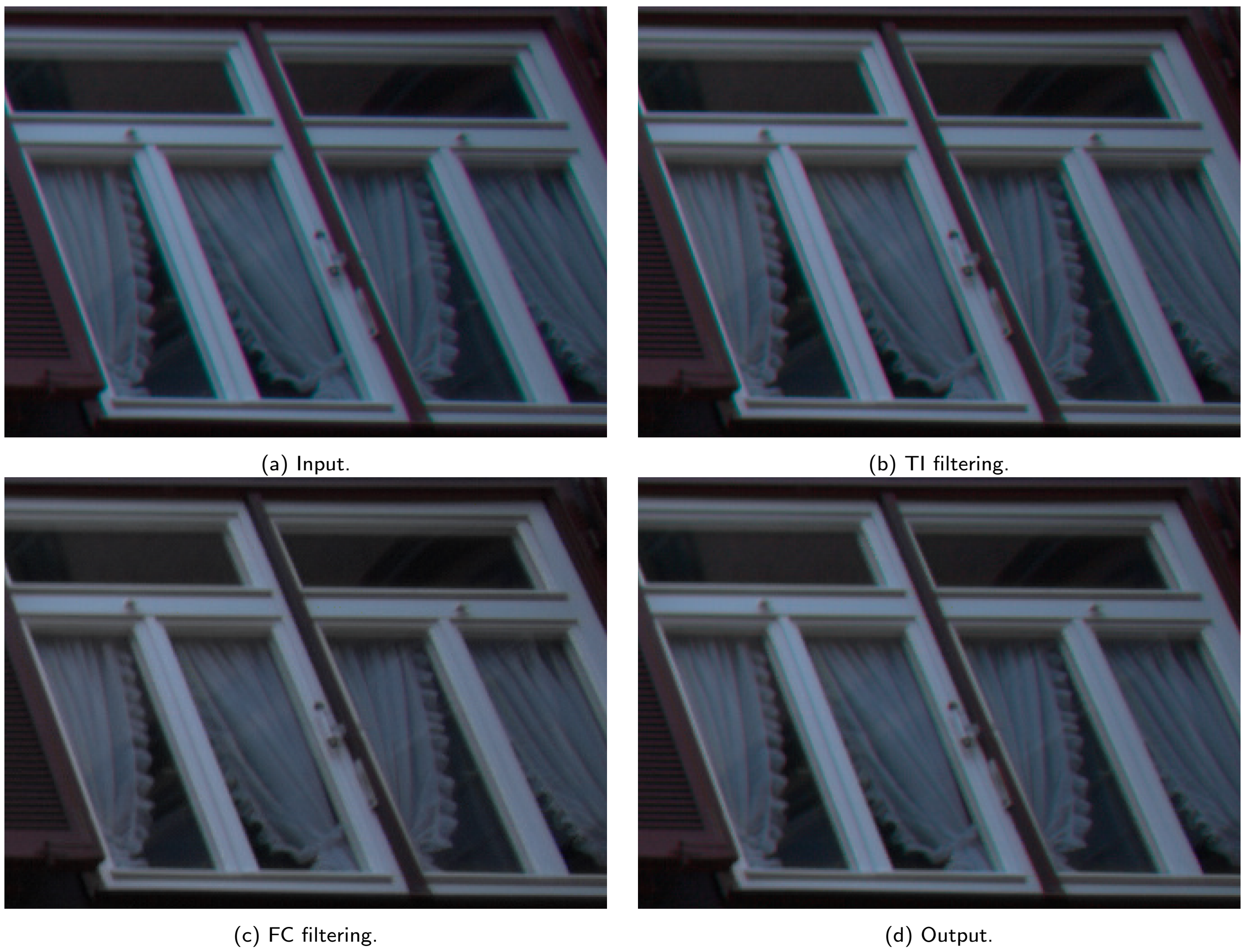
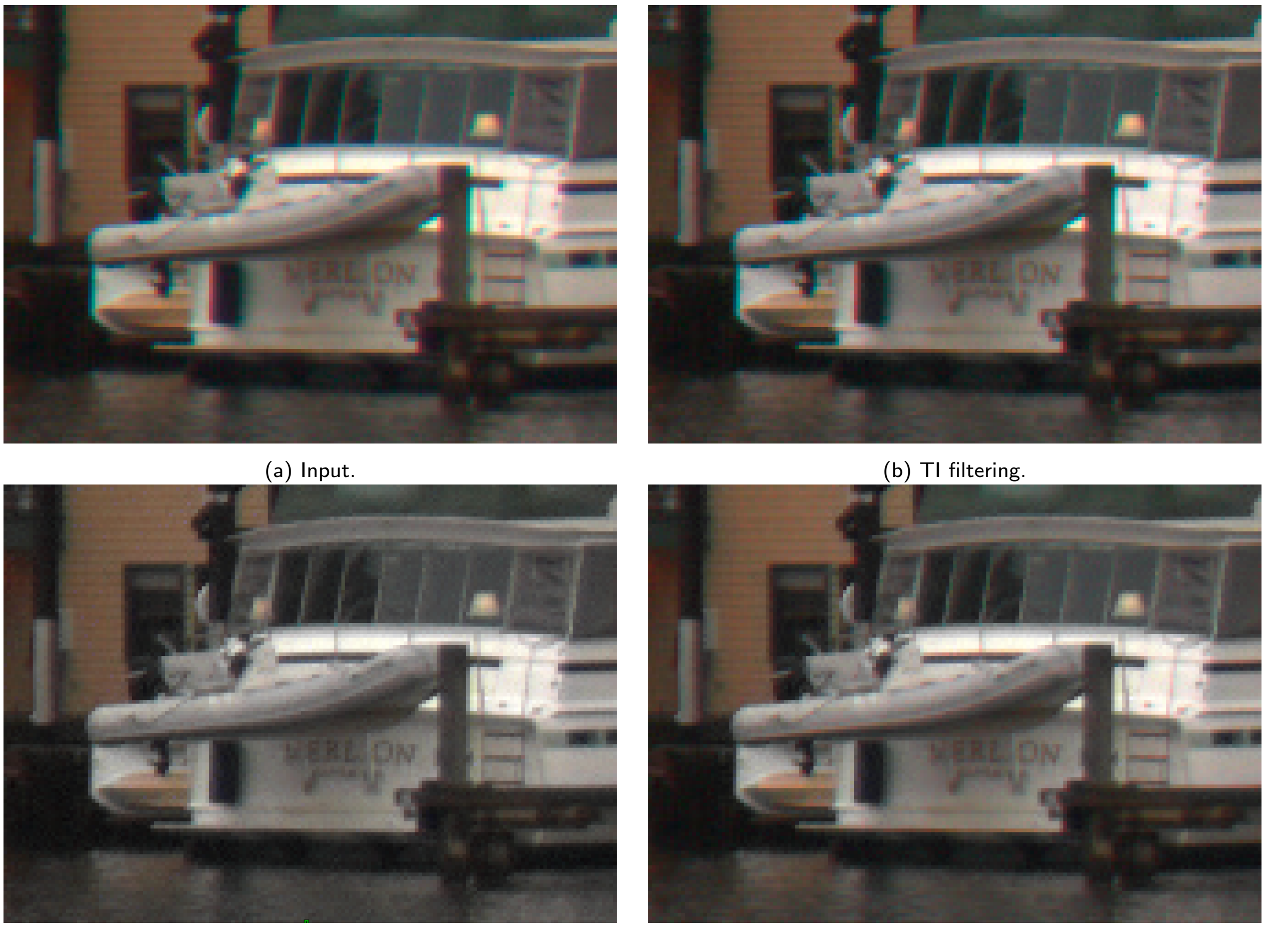

(a) Input. (b) TI filtering. (c) FC filtering. (d) Output.

Figure 2: Example of pure lateral CA removal from the *Facade* image from [9]. The TI stage slightly narrows the color fringes but the bulk of the work is performed by the FC stage that gets rid of most fringes.

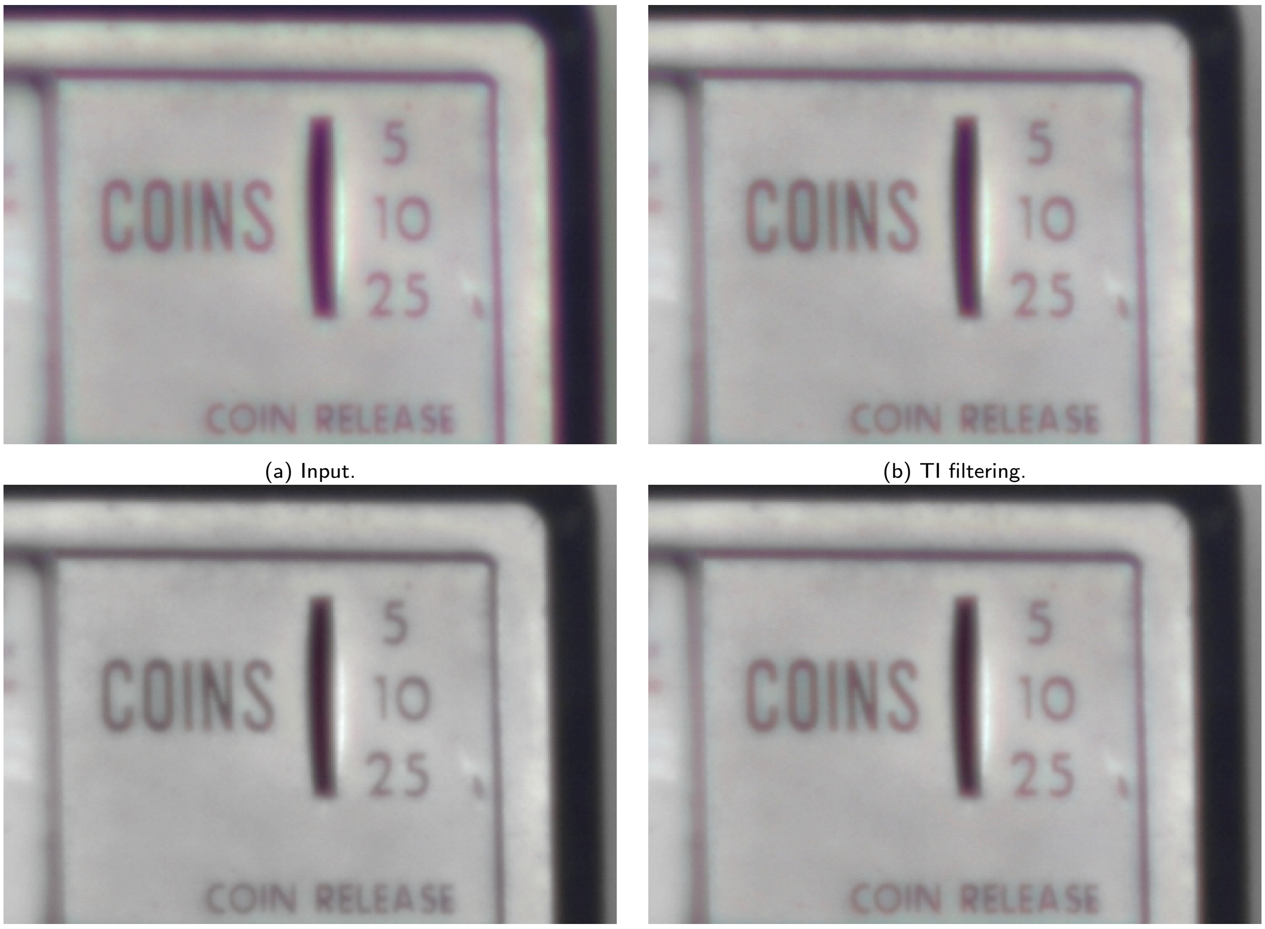
In both examples, it can be observed that the TI result is almost identical to the input image, confirming the relatively uselessness of the TI stage for purely-lateral aberration removal. However, the FC-filtered result does not contain the color fringes, at the cost of wiping out colors of the signal as well. For instance note the loss of colors on the "Merlon" sign on the boat in Figure 3. This is less pronounced in the other example that has no noticeable colors besides the aberrations next to the window frames. The final images after arbitration show no fringes, and even bring the colors of the latent clean image back in the final estimates compared to their FC counterparts. This validates the importance of the validation step in the restoration process.


(a) Input. (b) TI filtering.

(c) FC filtering. (d) Output.

Figure 3: Example of pure lateral CA removal from the *Boat* image from [8]. The TI stage slightly narrows the color fringes but most of the work is performed by the FC stage that gets rid of most fringes. Note however that "true" colors are also removed by the FC stage, highlighting the importance of the final arbitration stage that restores the red color on the "Merlon" boat sign.

Axial CA correction. In the context of more important axial over lateral CA, the TI stage becomes more important in the restoration process. When inspecting Figure 4 featuring the *Telephone* image from [6], it can be seen that the TI stage noticeably reduces the support of the axial CA next to the salient edges such as the writings, or at the silver/black color interface of the phone booth. The subsequent FC step removes the remaining artefacts that are slim color fringes, especially compared to that of the input image. In this case, both filtering steps equally contributed to the restoration process. Note that the final result retains some of the aberrations wiped out by the FC stage, and thus may appear slightly worse, but it is because the latent image in this crop is mainly monochromatic, thus favoring the FC solution. However, as noted in the previous examples, the arbitration stage is crucial to retain some of the colored elements that may be generally over-compensated by the FC stage.



(a) Input.

(b) TI filtering.

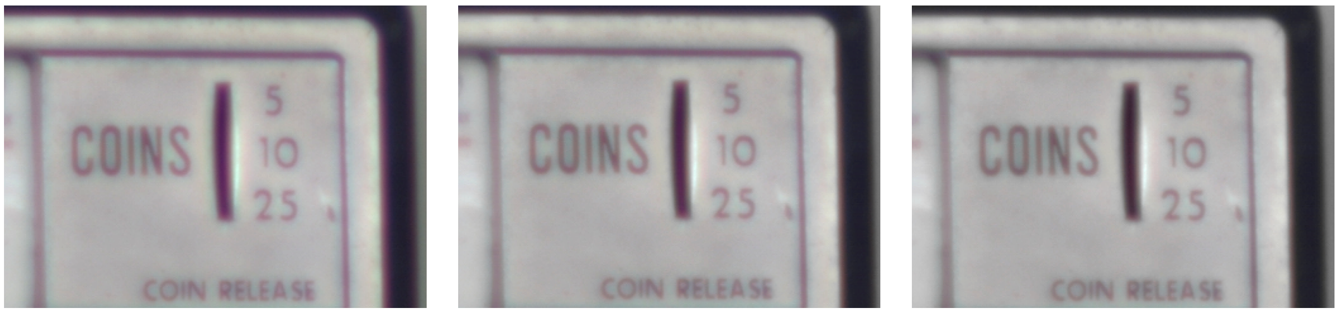
(c) FC filtering.

(d) Output.

Figure 4: Example of pure axial CA removal from the *Telephone* image from [6]. In that case, the TI stage importantly diminishes the CA in the image. The subsequent FC step gets rid of the remaining fringes. The method overall returns a more likely image compared to the input.

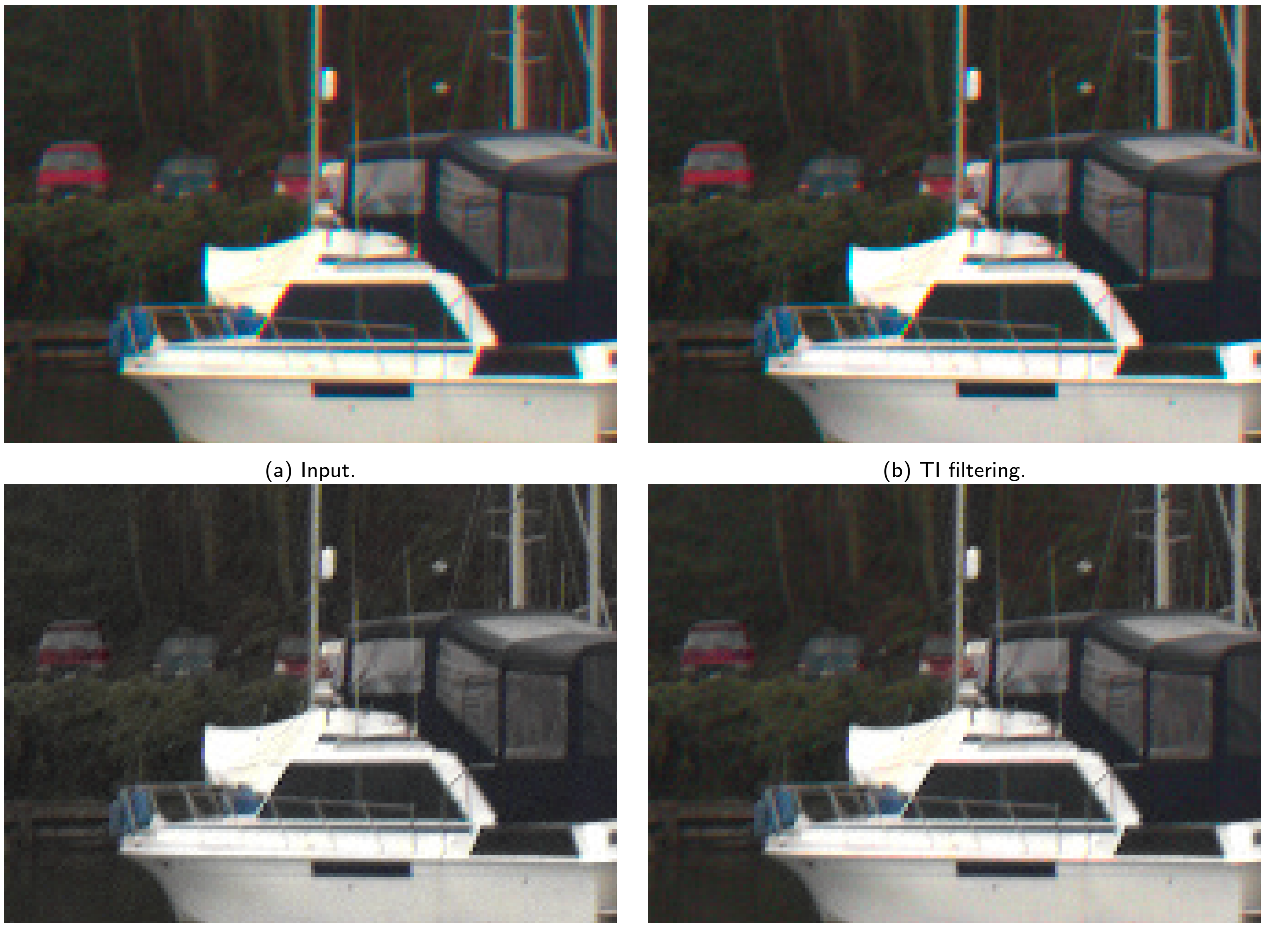
Limitations. The presented method is not perfect and has a couple of flaws that we list below. The first con concerns the choice of the L_{hor} and L_{ver} parameters, setting the size of the filters' supports. Figure 5 compares on the *Coins* image the result with the default parameters $L_{ver} = 4$px and $L_{hor} = 7$px compared with the one of the previous paragraph: $L_{ver} = 8$px and $L_{hor} = 14$px. One can see that the support of the filter for the default parameters is too short, and thus the result retains some artifacts that are not seen in the image with the filters with the larger supports. Indeed, setting larger values for L_{ver} and L_{hor} as a rule of thumb may alleviate the problems but in that case the filter's support may be too large and thus take into account other edges in the neighborhood that may alter the prediction. There is thus no universal choice of parameters for a faster and efficient algorithm.

The second con of the method is the number of hyper-parameters. Besides the simple-to-understand parameters L_{hor} and L_{ver}, there are several thresholds and weighting terms that are hard to handtune. Their tuning is however crucial to achieve satisfactory results. Figure 6 shows a consequence of a bad choice of these advanced parameters, that are here set to the default values. When not well-tuned, the parameters may lead to washed-out colors like the red cars in the figure. This was discussed in the previous examples such as that of the boat in Figure 3. Better tuning may alleviate the result but at the cost of considering 10 parameters, which is a combinatorial problem on its own. More recent works such as [5], better at disambiguating the part of chromatic aberrations in an aberrated image, may lead to better results.



(a) Input. (b) $L_{ver} = 4$px, $L_{hor} = 7$px. (c) $L_{ver} = 8$px, $L_{hor} = 14$px.

Figure 5: Example of a failure case caused by too small values of L_{ver} and L_{hor} in the *Telephone* image. In the middle image, we use the default parameters recommended in [2], whereas in the right image we doubled these values. Since we address here axial CA removal that may blur the edges on a large area, it is important to use a larger filter size like in the right image. Otherwise the final result retains colored artifacts like in the middle image.



(a) Input. (b) TI filtering. (c) FC filtering. (d) Output.

Figure 6: Example of a failure case caused by the advanced parameters in the FC stage in the *Bridge* image from [9]. Despite satisfactory correction of artifacts next to the poles on the right, the cars on the left have less pronounced red colors after correction. This is caused by the FC stage that considers these too-strong colors to be artifacts instead of part of the signal. This behavior may be lessened with more careful, but cumbersome, tuning of the advanced parameters.

4 Conclusion

In this article, we have presented a lightweight and fast algorithm for chromatic aberration compensation in a single RGB image. It consists of a series of two 1D filters that first gradually sharpen the red and blue edges, and second compensate the remaining color fringes in a luma-chroma domain with local adaptive averages. The two intermediate estimates are merged in a final arbitration stage. We have illustrated in the experimental section the performance of the technique, with the specific behaviors of the approach for pure axial or lateral chromatic aberrations in real-world images, drawing conclusions on the usefulness of each 1D filter. We finally show limitations of the technique, in particular the important number of parameters that are hard to fine-tune altogether.

Acknowledgment

This work was partly financed by the DGA Astrid Maturation project "SURECAVI" no ANR-21-ASM3-0002, Office of Naval research grant N00014-17-1-2552. We thank Thibaud Ehret for helping deploying the demo on the IPOL platform.

Image Credits

The *Telephone* image comes from [6]. The *Bridge* image comes from [8]. The *Facade* image comes from [9].

References

[1] T. E. Boult and G. Wolberg, *Correcting Chromatic Aberrations Using Image Warping*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1992, pp. 684–687. https://doi.org/10.1109/CVPR.1992.223201.

[2] J. Chang, H. Kang, and M. G. Kang, *Correction of Axial and Lateral Chromatic Aberration with False Color Filtering*, IEEE Transactions on Image Processing (TIP), 22 (2013), pp. 1186–1198. https://doi.org/10.1109/TIP.2012.2228489.

[3] T. Eboli, J. Morel, and G. Facciolo, *Breaking Down Polyblur: Fast Blind Correction of Small Anisotropic Blurs*, Image Processing On Line (IPOL), 12 (2022), pp. 435–456. https://doi.org/10.5201/ipol.2022.405.

[4] ——, *Fast Two-Step Blind Optical Aberration Correction*, in European Conference on Computer Vision (ECCV), Springer, 2022, pp. 693–708. https://doi.org/10.1007/978-3-031-20068-7_40.

[5] ——, *Collaborative Blind Image Deblurring*, ArXiv Preprint ArXiv:2305.16034, (2023). https://arxiv.org/abs/2305.16034.

[6] F. Heide, M. Rouf, M. B. Hullin, B. Labitzke, W. Heidrich, and A. Kolb, *High-Quality Computational Imaging Through Simple Lenses*, ACM Transactions on Graphics (TOG), 32 (2013), pp. 149:1–149:14. https://doi.org/10.1145/2516971.2516974.

[7] S. B. Kang, *Automatic Removal of Chromatic Aberration From a Single Image*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007. https://doi.org/10.1109/CVPR.2007.383214.

[8] E. Kee, S. Paris, S. Chen, and J. Wang, *Modeling and Removing Spatially-Varying Optical Blur*, in IEEE International Conference on Computational Photography (ICCP), 2011, pp. 1–8. https://doi.org/10.1109/ICCPHOT.2011.5753120.

[9] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf, *Blind Correction of Optical Aberrations*, in European Conference on Computer Vision (ECCV), Springer, 2012, pp. 187–200. https://doi.org/10.1007/978-3-642-33712-3_14.

[10] T. Sun, Y. Peng, and W. Heidrich, *Revisiting Cross-Channel Information Transfer for Chromatic Aberration Correction*, in IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3268–3276. https://doi.org/10.1109/ICCV.2017.352.