Fast Two-step Blind Optical Aberration Correction - Supplementary material

We provide additional results and companion analyses to those of the main paper. Section A provides more details on the blind deblurring algorithm, Section B focuses on the proposed CNN and its impact on the restoration pipeline, Section C discusses general implementation details of the method, and Section D shows additional qualitative results for raw images we have taken as well as the JPEG images of [9].

A Deblurring implementation details

A.1 Normalizing function

The normalizing function of Delbracio *et al.* [3] ensures that the images all have ideal latent edges between 0 and 1. We have observed that this is a critical component to make the blur estimation algorithm work. We use the same function as Delbracio *et al.*, defined by

$$n(v_G) = \min\left(\max\left(\frac{v_G - v_G[q]}{v_G[1-q] - v_G[q]}, 0\right), 1\right), \tag{1}$$

where $v_G[q]$ is the q-th quantile of the pixel values in v_G. The quantile value q is set to 0.001 in all the experiments of this presentation.

A.2 Interpolating the angles

We cannot compute the directional image derivative $\nabla_\varphi n(v_G)$ in all the possible angular directions; It would be too slow. We follow [3] and actually compute the derivatives for φ in $\{0, 30, 60, 90, 120, 150, 180\}^\circ$. The then compute the corresponding gradient magnitudes infinite norms

$$\|\nabla_\varphi n(v_G)\|_\infty = \max_x |\nabla_\varphi n(v_G)(x)|,$$

and linearly interpolate these values at every 6° angle, *i.e.,* we predict the infinite norm values for φ in $\{0, 6, 12, \ldots, 174, 180\}^\circ$, before computing the argmax with respect to φ. We have found that in practice this strategy was fast and accurate enough to approximate the real lens blurs.

A.3 Bounding the standard deviation predictions

We predict the parameters of the Gaussian approximation of the blur kernel σ_c (resp. ρ_c) with

$$\sigma_c = \sqrt{\frac{C^2}{\|\nabla_\varphi n(v_c)\|_\infty^2} - \sigma_b^2}, \tag{2}$$

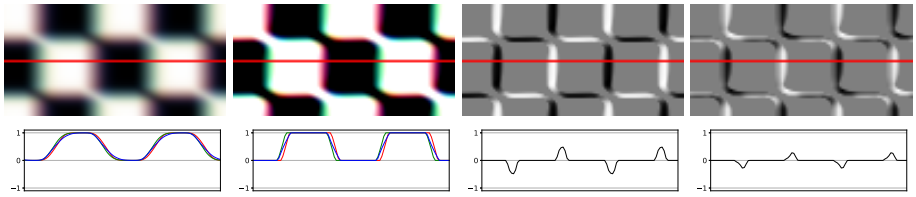


Fig. 1: From left to right: A synthetic aberrated image v, a sharpened version z with noticeable lateral chromatic aberration, the differences $z_R - z_G$ and $z_B - z_G$ showing the support of the colored artifacts. The profile of z shows that the miscorrelation of the three color channels causes the remaining artifacts, detected in the profiles of the images $z_R - z_G$ and $z_B - z_G$.

as in [3]. We however have remarked that when the magnitude of the gradients was to small, *e.g.,* in textured areas like a tree seen from afar, this equation was predicting a very large blur, even with the normalization function. As discussed in Section C, increasing the patch size may help. To limit this problem Delbracio *et al.* proposed a clipping strategy. However, in this work we use the following conservative strategy

$$\sigma_c^\star = \begin{cases} 0.2 \text{ if } \sigma_c > 4 \text{ or } V(n(v_c)) < \tau, \\ \sigma_c \text{ otherwise,} \end{cases} \tag{3}$$

where V is the variance operator and the threshold τ is set to 0.09. This strategy leads to a filter similar to a Dirac impulse, preventing deblurring artifacts in case of ill-blur prediction or "flat" patch, *e.g.,* a patch with only the sky. The same technique is also applied to ρ_c.

B CNN details

B.1 Motivation for the loss design

As we said in the main paper, we follow Chang *et al.* [2] and leverage the property that the green/red and green/blue image residuals are good features to detect chromatic aberrations in a photograph. Figure 1 shows an example for a checker grid image. Bumps on the profiles of the residuals indicate the presence of colored edges, most likely aberrations. When training a CNN, we minimize these quantities so the bumps are as small as possible.

B.2 Architecture

We detail the architecture of ϕ for predicting the colored residual in Table 1. We call C a convolutional layer, R a ReLU activation, B a batch normalization module and "Add d" a block that adds to the current feature map the output

Tag	1	2	3	4	5	6	7	8	9	10	11
Layer	CBR	CBR	CBR	CBR	CBR	Add 3	CBR	Add 2	CBR	Add 1	C
Dim	16×2	32×16	64×32	64×64	64×64	-	32×64	-	16×32	-	1×16

Table 1: Detail of the architecture of ϕ in the main paper for edge correction.

of layer d. All the convolutions have 3×3 filters and the dimensions are given with the format "output/input" channels.

Note that the input channel width is set to 2 since it combines the green and either the red or the blue channel, and returns a residual for the red or the blue channel.

B.3 Training data generation

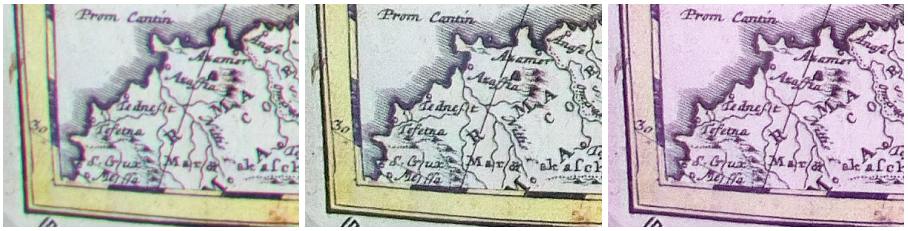
We detail in this section the training data for learning the optimal parameter ν of the CNN. The generation may be divided into four main stages, resulting in a deblurred but with colored-edges image and its sharp counterpart:

1. Unprocessing a JPEG image with the pipeline of Brooks *et al.* [1]. We invert tone-mapping with their proposed inverse S-curve function and gamma compression with the exponent 2.2. This yields a synthetic RGB image with linear values with respect to the electron counts;
2. Blurring and adding noise to the raw image. The simulated blurs are Gaussian filters with standard deviation values ρ_c and σ_c ($c = R, G, B$) in [0.2, 4] and sub-pixel horizontal and vertical translations in [-4,4]. The blurry and noisy patch is finally mosaicked with the RGGB Bayer pattern;
3. Denoising and demosaicking the synthetic raw image to mimic the two first stages of an image editing software. Because of speed for generating the training data, we use the bilateral filter [11] for denoising and the Hamilton-Adams algorithm [5] for demosaicking;
4. Deblurring the raw patch with the blind deblurring technique detailed in Section 4.1: From the denoised and demosaicked patch, we first predict the orientation θ and the color-dependent standard deviation values σ_c and ρ_c, and second we remove the blur with the approximate inverse filter defined by the polynomial p.

In this work we used the bilateral filter of for denoising and the Hamilton-Adams interpolator for fast demosaicking. We used these algorithms since they are fast but training may be indeed enhanced with CNN-based algorithms, *e.g.,* the blind denoiser of Wang *et al.* [12] and the demosaicking module of Gharbi *et al.* [4].

B.4 Choice of the loss

We have shown in the main paper that the proposed loss in Eq. (8), built over red/green and blue/green residuals is pivotal to achieve colored edge correction.



(a) Aberrated. (b) Loss in Eq. (8). (c) Gradient variant.

Fig. 2: Comparison of the color biases introduces by training loss, whether we evaluate the difference of the pixel values, *i.e.*, $\|(\widehat{u}_c - z_G) - (u_c - u_G)\|_1$, or the gradient values, *i.e.*, $\|\nabla(\widehat{u}_c - z_G) - \nabla(u_c - u_G)\|_1$, as advised by the prior of Heide *et al.* [6]. The loss on pixel value residuals retains the same exposure and color palette as in the original aberrated image, whereas the one on the gradients introduces a pinkish bias. Note that both versions actually compensate the colored edges.

Fig. 3: The ten 6000×4000 images (24 megapixels) we use for the quantitative analysis of the edge correction algorithm. Each image features salient edges, prone to lateral chromatic aberrations. The reader is invited to zoom in on a computer screen.

However the prior of Heide *et al.* [6] in Eq. (13) compares the gradients of the color channel. Training the CNN ϕ with a loss minimizing the gradients of the residuals instead, and reminiscent of Eq. (13), is sub-optimal since there is no reference to the pixel, and leads to a wrong average color in the image. Thus, the residual-based training loss prevents these issues, leverages the property of the lateral chromatic aberrations detailed in [2], and leads to solutions minimizing the prior of Heide *et al.* at the same time. Figure 2 shows an example of this phenomenon. We have noted that the combination of the loss on the pixel values in Eq. (8) and on the gradients of the color residuals was leading to marginal gains compared to that only minimizing the pixel values of the colored residuals, validating Eq. (8).

(a) Original (b) Size 100 (c) Size 200 (d) Size 400 (e) Size 800

Fig. 4: Qualitative comparison of the impact of the patch size on the peformance of the blind deblurring module on a real aberrated image. From left to right: the aberrated image and restored versions where the patch size is respectively 100, 200, 400 and 800. For patch sizes of 400 or 800, the image is actually deblurred. Under, the image is either blurry or contains artifacts. Note the presence of colored edges, *e.g.,* next ot the electrical cable, since we show images solely deblurred, prior to any evaluation with the CNN ϕ. The reader is invited to zoom in on a computer screen.

C Whole pipeline details

C.1 Test images

We show in Figure 3 the ten images we use for evaluating the edge correction algorithms in the experiment section of the main paper. The images where taken with the Sony α6000 camera, the Sony FE 35mm $f/1.8$ lens at maximal aperture and the Sigma 18-50mm $f/2.8$ DC DN lens at maximal aperture and shortest focal length.

C.2 Patch size

Setting the size of the patches is critical for the success of the blur estimation technique of Delbracio *et al.* [3] in the context of spatially-varying blurs. Indeed, this method is based on the presence of salient edges and may fail if there are too few edges on the patch support, *e.g.,* for too small patches. We show in Fig. 4 the comparison of an image crop for images deblurred with different patch sizes, ranging from 100 to 800 pixels. We restore non-overlapping patches to visualize what the deblurring exactly restores on each patch. In this figure one can see that for the patch size set to 100 the image looks almost like the original one. For the patch size set to 200, noticeable deconvolution artifacts can be seen next to the leaves. For the patch size set to 400 and 800, the restored results are plausible.

The result for the patch size of 200 may be explained by the fact that, to work well the blur estimation method needs edges with important contrast. However, in textured regions with only moderate gradients, the affine rule may predict

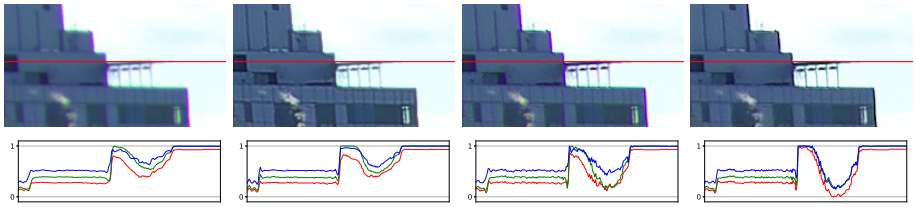


Fig. 5: Comparison of purple fringe removal from a real raw image. From left to right: The blurry image, the version restored with DxO PhotoLab 5 (non-blind), and the images only deblurred and deblurred+corrected by our model (blind).

a larger standard deviation value than the real one, resulting in a too large deconvolution filter and thus artifacts in the final image. A Weakness of this affine rule is thus such regions, and a simple way to prevent these artifacts is selecting larger patches to favor the presence of more contrasted edges. In this presentation, we set the patch size to 400, which is valid for most images we have tested our approach on.

C.3 Saturation

The combination of optical chromatic aberrations and saturation is called by photographers "purple fringes", an artifact challenging to remove. Yet, our technique successfully compensates these fringes as shown in Fig. 5. However, despite good performance on real images our approach cannot remove all the purple fringes, and leaves a thin dark line next to saturated areas. As previously noted, the performance of our method is closely related to that of the blur estimation stage, which makes the assumption that there is at least one strong edge in the patch, and thus may fail in textured regions.

D Additional images

D.1 Raw images

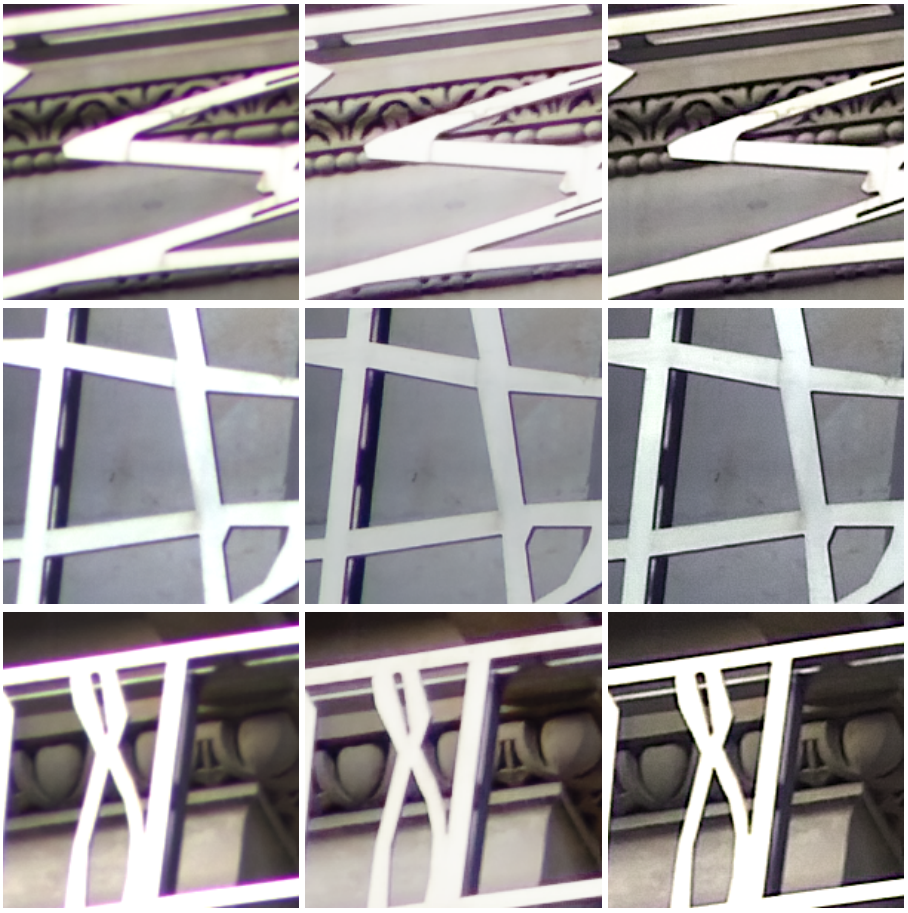
To qualitatively validate our approach, we have taken several photographs with a Sony α6000 camera, and combined with the Sony FE 35mm $f/1,8$ and the Sigma 18-50mm $f/2.8$ DC DN lenses. We compare our approach to the commercial software DxO PhotoLab 5, whose catalog contains the profile of the Sony lens, but not that of the Sigma one recently released in October 2021. PhotoLab thus runs in a non-blind setting for the 35mm lens, and should achieve the best result over our technique, whereas it runs in a blind setting for the Sigma lens. The images dubbed "culture", "map" and "tree" are shown in Figure 6, and magnified crops are shown in Figures 7, 8 and 9. The tree example in Figure 9 illustrates in particular the robustness of our method to "purple fringes", *i.e.*, the combination of optical aberrations and saturation.

(a) Culture (Sony lens). (b) Map (Sigma lens). (c) Tree (Sony lens).

Fig. 6: The additional images for qualitative evaluation. The images are denoised and demosaicked with DxO PhotoLab 5.

D.2 JPEG images

We also compare the restoration of JPEG images, with the methods of [8], [9] and [10] when the images are available. Our method achieves overall the best results. The images dubbed "facade" and "bridge" are shown in Figure 10, and magnified crops are shown in Figures 11 and 12.



(a) Blurry. (b) DxO. (c) Ours.

Fig. 7: Crops for the "map" image taken with the Sony FE 35mm $f/1.8$ lens. From left to right: the original blurry image, the optical aberration correction of DxO PhotoLab (non-blind setting), and ours.

(a) Blurry. (b) DxO. (c) Ours.

Fig. 8: Crops for the "map" image taken with the Sigma 18-50mm $f/2.8$ DC DN lens. From left to right: the original blurry image, the optical aberration correction of DxO PhotoLab (blind setting), and ours.

(a) Blurry. (b) DxO. (c) Ours.

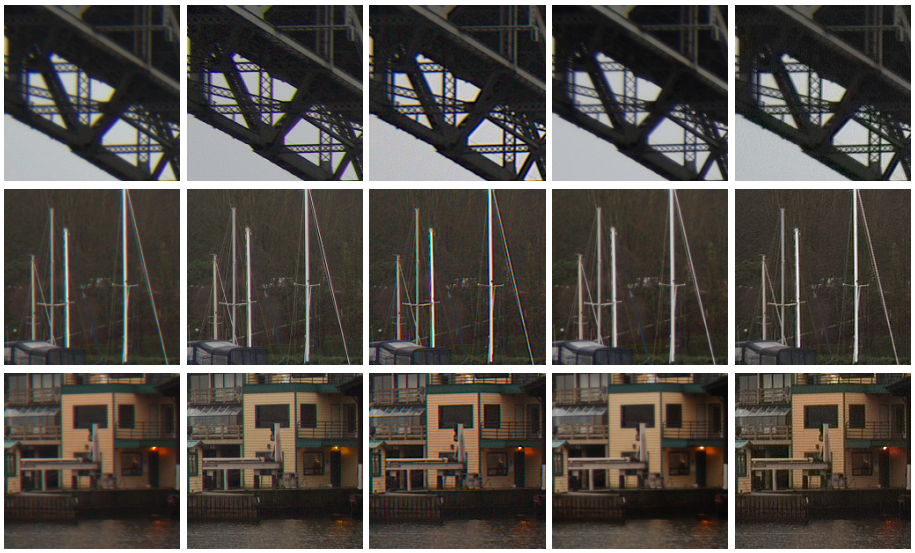
Fig. 9: Crops for the "tree" image taken with the Sony FE 35mm $f/1.8$ lens. From left to right: the original blurry image, the optical aberration correction of DxO PhotoLab (non-blind setting), and ours.

(a) Facade (b) Bridge

Fig. 10: The additional JPEG images from [9] for qualitative evaluation.

(a) Blurry. (b) [8]. (c) [9]. (d) [10]. (e) Ours.

Fig. 11: Crops for the "Facade" image from [9]. From left to right: the original blurry image, the non-blind result of Schuler *et al.* [8], the blind result of Schuler *et al.* [9], the blind result of Sun *et al.* [10], and ours.



(a) Blurry. (b) [7]. (c) [9]. (d) DxO ([9]). (e) Ours.

Fig. 12: Crops for the "Bridge" image from [9]. From left to right: the original blurry image, the non-blind result of Kee *et al.* [8], the blind result of Schuler *et al.* [9], the blind result of DxO (runned by Schuler *et al.* [9]), and ours.