Pattern-invariant Unrolling for Robust Demosaicking

Matthieu Muller, Daniele Picone, Mauro Dalla Mura, Magnus O Ulfarsson

Pattern-invariant Unrolling for Robust Demosaicking

Matthieu Muller
GIPSA-Lab
Univ. Grenoble Alpes, CNRS, Grenoble INP
38000 Grenoble, France
matthieu.muller@gipsa-lab.fr

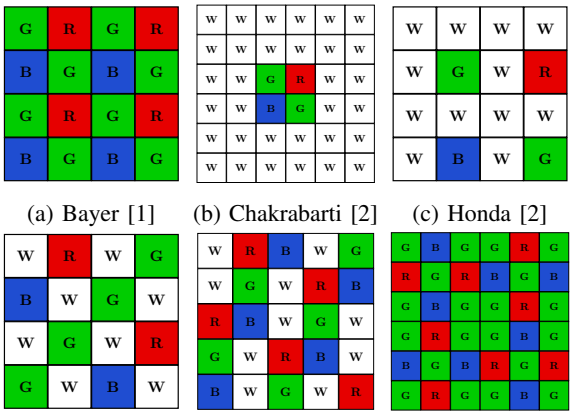
Daniele Picone
GIPSA-Lab
Univ. Grenoble Alpes, CNRS, Grenoble INP
38000 Grenoble, France
daniele.picone@gipsa-lab.fr

Mauro Dalla Mura
GIPSA-Lab
Univ. Grenoble Alpes, CNRS, Grenoble INP & IUF
38000 Grenoble, France
mauro.dalla-mura@gipsa-lab.fr

Magnus O. Ulfarsson
Faculty of Electrical and Computer Engineering
University of Iceland
101 Reykjavik, Iceland
mou@hi.is

***Abstract*—To acquire color images, most commercial cameras rely on color filter arrays (CFAs), which are a pattern of color filters overlaid over the sensor's focal plane. Demosaicking describes the processing techniques to reconstruct a full color image for all pixels on the focal plane array. Most demosaicking methods are tailored for a specific CFA, and tend to work poorly for others. In this work we present an algorithm for demosaicking a wide variety of CFAs. The proposed method allows to blend the knowledge of the CFA with information coming from data, employing a novel transformation and pattern-invariant loss function. The method is based on the unrolling of an algorithm based on a neural network learned on available examples. Preliminary experiments over RGB and RGBW CFAs show that the method performs well over a range of CFAs and is competitive for CFAs for which competing methods were tailored to work well on.**

***Index Terms*—Demosaicking, unrolling, image processing, deep learning, color filter arrays.**



(a) Bayer [1] (b) Chakrabarti [2] (c) Honda [2]

(d) Sony [2] (e) Wang [2] (f) Xtrans [3]

Fig. 1: Examples of CFAs.

I. Introduction

In many applications such as small satellites or smartphone cameras the size of the sensor plays a crucial role. A common way to keep the sensors small is to use color filter arrays (CFAs). A CFA defines a pattern of color filters which is overlaid over the sensor's pixels. Because each pixel only embeds the information for one color demosaicking is required to restore the missing colors.

The choice of the CFA itself leads to drastic changes in the raw acquisition, increasing dependency of the demosaicking algorithm to the CFA. The Bayer pattern shown on Fig. 1a is the most widespread pattern that uses RGB filters. Additionally, a newer family of CFAs uses RGBW filters (e.g., Fig. 1c) which improves signal-to-noise ratio (SNR), especially in low light condition.

This great variety of configurations makes demosaicking difficult as most methods are tailored to a specific CFA. Many categories of demosaicking techniques exist. For example classic interpolation and its variants [4], [5]. Those methods should be tailored to each CFA and thus lack invariance to the choice of the CFA to work on.

Model-based methods rely on a forward model representing the CFA operation [6], [7] and prior knowledge of the images to be reconstructed. Those methods are invariant to the CFA, but depend greatly on the quality of the forward model and the choice of the prior, which can be a limiting factor.

Deep learning methods have recently emerged [8], [9]. They deliver great performances thanks to an end-to-end training, with couples of reference images and related raw acquisitions. Yet this same training comes with a specialization on the CFA

This work was partly supported by grant ANR FuMultiSPOC (ANR-20-ASTR-0006) and the Icelandic Research Fund 207233-52. All (or most of) the computations presented in this paper were performed using the GRICAD infrastructure (https://gricad.univ-grenoble-alpes.fr), which is supported by Grenoble research communities.

the network is trained on and could need vast quantities of data. In practice those methods are not robust to new CFAs and are too computationally expensive to train on classic computers.

Finally, a growing research domain aims at fusing model-based methods with data-driven ones. Deep Image Prior (DIP) [10] uses a neural network architecture as a prior. This method can be considered as unsupervised and works like an iterative algorithm, which provides a good invariance to the choice of the CFA. Such methods do not need a prior but a forward model. However, they are very computationally expensive, and its performances depends on the choice of the architecture.

Algorithm unrolling [11]–[13] is another possibility which has proven successful in various image processing tasks. It tries to learn prior knowledge from a dataset while still using a forward-model. Such methods should retain high robustness to CFA changes while benefiting from the training of the prior. However, experiences show that unrolling algorithms are prone to overfit the dataset used for training and the learned prior becomes dependent on the type of forward model used losing the advantage of invariance provided by conventional model-based approaches.

Therefore, the following work proposes a demosaicking method based on unrolling which is robust to the choice of the CFA. This paper presents the following novel contributions:

- An unrolled algorithm for demosaicking for varied CFAs. It uses an inexpensive way to compute matrix inversion in the demosaicking problem and a loss function enhancing robustness to the CFA.
- A loss function which limits the network from learning CFA-specific characteristics, focusing on the demosaicking problem itself regardless of the considered CFAs.

II. Methodology

A. Problem Formulation

Let us denote a vectorized RGB image by $\boldsymbol{x} \in \mathbb{R}^{3n\times 1}$. The image formation usually assumes a linear model and additive Gaussian noise [6], [7], [14]. It takes the RGB image and produces the raw acquisition to represent the camera's image formation model, which is thus written as:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \eta, \tag{1}$$

where the degradation operator $\boldsymbol{A} \in \mathbb{R}^{n\times 3n}$ encodes a CFA. It will sample the raw acquisition $\boldsymbol{y} \in \mathbb{R}^{n\times 1}$ from the RGB image $\boldsymbol{x}$, and $\eta \in \mathbb{R}^{n\times 1}$ is a vector noise. The objective is to reconstruct $\boldsymbol{x}$ using $\boldsymbol{y}$ and the knowledge of $\boldsymbol{A}$.

The problem is approached by minimizing a convex cost function which is the sum of two terms. First is the data fidelity term $F = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ ensuring consistency with the acquisition, which derives from the log-likelihood estimator when η is a zero-mean Gaussian noise. The second term is a regularizer $R : \mathbb{R}^{3n\times 1} \to \mathbb{R}$ encoding constraints on the reconstruction:

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\arg\min}\ F(\boldsymbol{x}) + R(\boldsymbol{x}). \tag{2}$$

The arbitrary choice of R is a limitation of model-based approaches as it is hard to find the optimal regularizer for a given forward model and a given acquisition. Nonetheless, some regularizers are widely used like Total Variation (TV) [15] which yields good results in a wide range of image processing tasks.

B. Unrolled Algorithm

a) ADMM solver: The cost function (2) is minimized by an iterative method such as the Alternating Direction Method of Multipliers (ADMM) [7], [16]. Which consists in iterating the following three steps [17], [18]:

$$\begin{cases} \boldsymbol{x}^{k+1} &= \mathbf{prox}_{\rho,F}(\boldsymbol{z}^k - \boldsymbol{u}^k) \\ \boldsymbol{z}^{k+1} &= \mathbf{prox}_{\tau,R}(\boldsymbol{x}^{k+1} + \boldsymbol{u}^k) \\ \boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1} \end{cases} \tag{3}$$

where $k \in \mathbb{N}$ is the iteration index, $\mathbf{prox}_{\rho,F}(\boldsymbol{v}) = \underset{\boldsymbol{w}}{\arg\min}\ F(\boldsymbol{w}) + \frac{\rho}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2$ and $\mathbf{prox}_{\tau,R}$ is defined in the same way. $\rho, \tau \in \mathbb{R}_*^+$ are the proximal steps of the algorithm and one has to tune them empirically in addition to the complex choice of R which leads to suboptimal performances.

b) Unrolling ADMM: Algorithm unrolling offers a way to benefit from deep learning networks, by replacing the term $\mathbf{prox}_{\tau,R}$ and the hyperparameters with learnable layers [11], [19]. It seeks to fuse the expressive power of deep learning methods with the interpretability and robustness of model-based methods.

Algorithm unrolling takes an iterative algorithm, such as ADMM, and unfolds a certain number K iterations of the algorithm. At iteration $k \in [0, K-1]$ the call to $\mathbf{prox}_{\tau,R}$ is replaced by a neural network N_{θ_k}, θ_k being the trainable parameters. Algorithm 1 presents the pseudocode of the method, which is similar to ADMM. However, if the overall algorithm keeps the iterative framework, each iteration can be different.

Algorithm 1 Unrolled ADMM algorithm f_θ over K iterations.

Input: $\boldsymbol{y} \in \mathbb{R}^{n\times 1}, \boldsymbol{A} \in \mathbb{R}^{n\times 3n}, K \in \mathbb{N}$
Output: $\boldsymbol{x}_{K-1} \in \mathbb{R}^{3n\times 1}$

$\boldsymbol{x}_0 \leftarrow \boldsymbol{A}^T\boldsymbol{y}$
$\boldsymbol{z}_0 \leftarrow \boldsymbol{A}^T\boldsymbol{y}$
$\boldsymbol{u}_0 \leftarrow 0$
for $k = 0$ to $K-1$ **do**
    $\boldsymbol{x}_{k+1} \leftarrow \mathcal{D}_{\boldsymbol{A}k}(\boldsymbol{z}_k, \boldsymbol{u}_k)$
    $\boldsymbol{z}_{k+1} \leftarrow \mathcal{R}_k(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k)$
    $\boldsymbol{u}_{k+1} \leftarrow \mathcal{M}_k(\boldsymbol{x}_{k+1}, \boldsymbol{z}_{k+1}, \boldsymbol{u}_k)$
end for

As shown in Algorithm 1, each iteration of the unrolled algorithm mirrors those of ADMM in (3) and is composed of three distinct blocks of layers. At iteration k the blocks are: the data block $\mathcal{D}_{\boldsymbol{A}k}$ performing the $\boldsymbol{x}_{k+1}$ update, the regularizer block $\mathcal{R}_k$ for $\boldsymbol{z}_{k+1}$ and the multiplier block $\mathcal{M}_k$ for $\boldsymbol{u}_{k+1}$. All those blocks contain learnable parameters which are independent of one iteration to another. However, the network

tends to lose its invariance property and overfits on the CFAs seen during the training. In the following, $f_\theta(\boldsymbol{A}, \boldsymbol{y})_k$ will denote the result of the k-th iteration of Algorithm 1 with learnable parameters θ as a function of a CFA $\boldsymbol{A}$ and a raw acquisition $\boldsymbol{y}$.

C. Proposed Method

a) Data block $\mathcal{D}_{\boldsymbol{A}k}$: This block remains largely unchanged compared to its ADMM counterpart. It consists in a closed-form formula involving a learnable scalar which serves as the proximal iteration ρ_k:

$$\mathcal{D}_{\boldsymbol{A}k}(\boldsymbol{z}_k, \boldsymbol{u}_k) = (\rho_k \boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A})^{-1}(\rho_k(\boldsymbol{z}_k - \boldsymbol{u}_k) + \boldsymbol{A}^T\boldsymbol{y}), \quad (4)$$

where $\boldsymbol{I}$ is the identity matrix, $\rho_k \in \mathbb{R}_*^+$ is learnable and $\boldsymbol{A}$ is the degradation operator presented in (1). The computation of $(\rho_k \boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A})^{-1}$ is unfeasible on a computer as the matrix is too large. We propose to compute using the Sherman-Morrison-Woodbury formula found in [17]:

$$(\rho_k \boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A})^{-1} = \frac{1}{\rho_k}(\boldsymbol{I} - \boldsymbol{A}^T(\boldsymbol{I} + \frac{1}{\rho_k}\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{A}). \quad (5)$$

Eq. (5) is particularly useful in the demosaicking problem as the matrix $\boldsymbol{A}\boldsymbol{A}^T$ is a diagonal matrix. Even if the matrix $(\boldsymbol{I} + \frac{1}{\rho_k}\boldsymbol{A}\boldsymbol{A}^T)$ is still large its inversion is straight forward as it is only the element-wise inversion of its diagonal elements. Combining (4) and (5) gives the final form of the data block.

b) Regularizer block $\mathcal{R}_k$: We choose to implement the regularizer block as a scaled version of the U-Net network [20] as using relatively small convolutional neural network (CNN) at each iteration helps keeping the size of the network globally constrained. The architecture of the network is composed of the same kind of layers as U-Net, using double convolutions, max pooling and upsampling layers. Compared to the U-Net this network is much more lightweight, keeping the overall algorithm accessible for personal computers as the use of the degradation model will guide the reconstruction, compensating for the small size of the network. The block is thus represented by the following equation:

$$\mathcal{R}_k(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) = N_{\theta_k}(\boldsymbol{x}_{k+1} + \boldsymbol{u}_k),$$

where N_{θ_k} is the network presented previously. It is important to note that even if the architecture is the same for all iterations the weights θ_k are different. Forcing the weights to be equal across the iterations would greatly reduce the expressive power of the network.

c) Multiplier block $\mathcal{M}_k$: The multiplier block is the simplest one as it applies a straight forward closed-form formula with a learnable scalar:

$$\mathcal{M}_k(\boldsymbol{x}_{k+1}, \boldsymbol{z}_{k+1}, \boldsymbol{u}_k) = \boldsymbol{u}_k + \eta_k(\boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1}),$$

with $\eta_k \in \mathbb{R}$ a learnable weight which allows the algorithm to have a greater flexibility regarding the data. This block keeps track of the residual between $\boldsymbol{x}_{k+1}$ and $\boldsymbol{z}_{k+1}$ and should help the future iterations to refine the other variables.

d) Number of parameters: The last iteration of f_θ has only the $\mathcal{D}_{\boldsymbol{A}}$ block as the other blocks are only useful to the next iteration. Consequentially the total number of parameters is given by:

$$|\theta| = (K-1)(2 + |\theta_k|) + 1, \quad (6)$$

where K the number of iterations and $|\theta_k|$ the number of parameters in the regularizer block at iteration k.

e) On vanishing gradients: Even if the network has few parameters it is relatively deep as it stacks CNNs one on top of the others, which quickly leads to vanishing gradients. We propose to replace the ReLU activation functions in the regularizer blocks $\mathcal{R}_k$ by LeakyReLU functions, which limits the appearance of vanishing gradient.

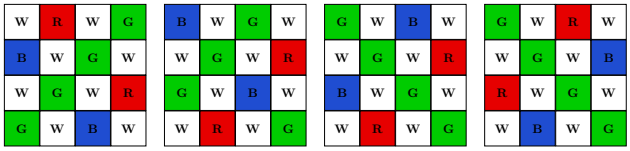
Another important element resides in the loss function. Instead of performing the back propagation on the output of the final iteration $f_\theta(\boldsymbol{A}, \boldsymbol{y})_{K-1}$, we propose to use all outputs from the intermediate iterations $f_\theta(\boldsymbol{A}, \boldsymbol{y})_k$, $k \in [0, K-1]$. Computing the loss between the iterations allows the gradient flow to go directly to the previous iterations instead of going through all the previous iterations, with less risk of vanishing gradients along the way.

f) CFA invariant loss function: Unrolled algorithms are prone to overfit on the CFA they were trained on, limiting their robustness when applied on images acquired with other CFAs. Indeed, each new CFA to demosaick is a new problem in itself, which explains the specialization of the demosaicking methods found in the literature. To alleviate this problem we propose a novel loss function that will provide the algorithm a global view of the demosaicking task. CFAs (see Fig. 1) can be seen as different spatial arrangements of patterns. If we consider a set of geometrical transformation $\mathcal{T}$ such as a subset of the isometries of $\mathbb{R}^2$: $\mathcal{T} \subseteq \{g : \mathbb{R}^2 \to \mathbb{R}^2 \mid |g(a) - g(b)| = |a - b|, \ \forall a, b \in \mathbb{R}^2\}$, we can generate new CFAs increasing their diversity. For example, Fig. 2 shows such geometric transformations applied to a CFA. Each initial CFA seen during training generates multiple new ones which are geometric transformations of the initial one. This ensures the network will not overfit on specific CFAs and increases the capability to handle images acquired with other CFAs. In other terms this loss function favors robustness to new CFAs but also an invariant reconstruction to geometric transformations as follows:

$$f_\theta(T(\boldsymbol{A}), \boldsymbol{y}_{T(\boldsymbol{A})}) = f_\theta(\boldsymbol{A}, \boldsymbol{y}_{\boldsymbol{A}}) = \boldsymbol{x}, \quad \forall T \in \mathcal{T},$$

where $\boldsymbol{A}$ is a CFA seen during training, $\boldsymbol{y}_{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{x}$ the raw acquisition from $\boldsymbol{A}$ and $\boldsymbol{y}_{T(\boldsymbol{A})} = T(\boldsymbol{A})\boldsymbol{x}$ the raw acquisition from $T(\boldsymbol{A})$. The concept is related to the equivariance concept proposed in [21] for promoting some structural features in the results (e.g., translation equivariance). In this work the concept is adapted to favor the image reconstruction operator to be CFA-agnostic.

Given Φ the training dataset of clean images and Ψ the set of CFAs to be trained on, avoiding vanishing gradients and the invariant reconstruction property are enforced by the following loss function:



(a) Sony (b) Translation (c) Reflection (d) Rotation

Fig. 2: Sony CFA and some possible transformations.

$$\mathcal{L}_\theta(\Xi,\Psi,\mathcal{T}) = \frac{1}{K|\Xi||\Psi||\mathcal{T}|}\sum_{k=0}^{K-1}\sum_{i=1}^{|\Xi|}\sum_{j=1}^{|\Psi|}\sum_{l=1}^{|\mathcal{T}|}\|\boldsymbol{x}^i - \hat{\boldsymbol{x}}^i_{k,j,l}\|^2,$$

where $\boldsymbol{x}^i \in \Xi$, $\boldsymbol{A} \in \Psi$ and $\hat{\boldsymbol{x}}^i_{k,j,l} = f_\theta(T_l(\boldsymbol{A}_j), T_l(\boldsymbol{A}_j)\boldsymbol{x}^i)_k$ the output of the algorithm at iteration k.

III. Experiments

The proposed method [1] is initialized with $K = 4$ iterations and 32 channels for the first convolution of the networks. Each iteration contains about 1 million of parameters to train, thus following (6) the overall method contains 3.2 million of parameters. This is relatively small compared to other deep learning methods in image processing. The training is done on a set Φ of 64×64 overlapping patches from the BSD500 [22] for 200 epochs. The initial learning rate is set to 1×10^{-2}.

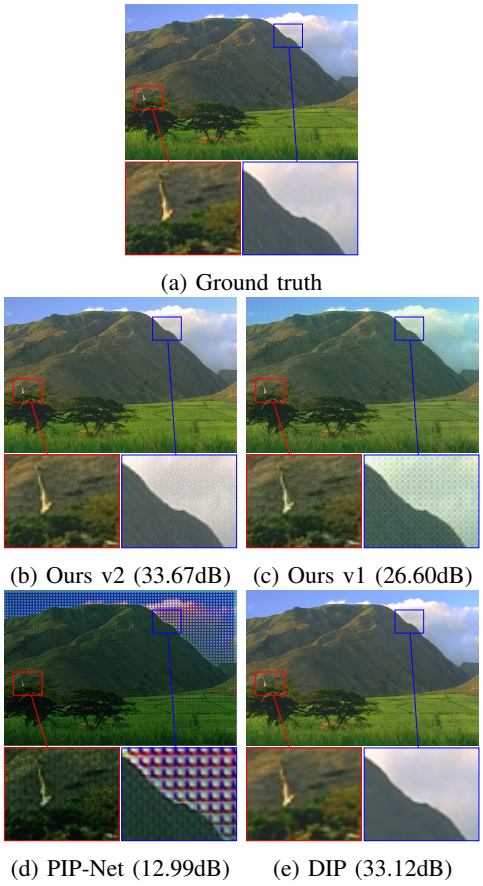
The method is trained on a set Ψ of 13 different CFAs (both RGB and RGBW). Some of the masks that were used for the training are seen in Fig. 1a to Fig. 1e, and we label them as "seen" in the tables and figures. Some other patterns, such as the Xtrans in Fig. 1f, were never utilized in the training and are used to test the ability of the proposed network to generalize to unknown patterns. Those are labeled as "unseen" in tables and figures.

Additionally, the set $\mathcal{T}$ of geometrical transformations is composed of all the possible cyclic translations, their 4 different rotations with an angle of $\pi/2$ and vertical and horizontal reflections. To increase the number of variations $\mathcal{T}$ also includes all the possible compositions of those transformations. An example of such transformations is shown in Fig. 2.

It is worth noting that the unseen patterns cannot be generated by the geometric transformation of the seen patterns, making them effectively never seen during the training phase.

The method, denoted "Ours v2", will be compared to six other methods: bilinear interpolation; pan-sharpening, adapted from [4] to work with multiple CFAs; PIP-Net [23] a supervised demosaicking method trained for Bayer demosaicking, which uses 4.5 million of parameters (a 50% increase compared to our method); total variation [7] which is fully model-based; Deep Image Prior (DIP) [10] a state-of-the-art technique for image reconstruction. Finally, the method will be compared to plain ADMM unrolling, denoted "Ours v1", using the same architecture and training set as proposed above without the invariant procedure. This highlights the fact that unrolling alone is not able to generalize to other CFAs and will overfit on the seen ones. Tables I and II present the PSNR (dB) and the SSIM of all those methods on the testing subset of the BDS500 dataset, on seen or unseen CFAs.

[1]The code, written in Pytorch, and the weights are publicly available at https://github.com/mattmull42/unrolled_demosaicking



(a) Ground truth

(b) Ours v2 (33.67dB) (c) Ours v1 (26.60dB)

(d) PIP-Net (12.99dB) (e) DIP (33.12dB)

Fig. 3: Ouputs and PSNR for various methods on the unseen Xtrans CFA (best viewed with zoom).

Table I shows that our method yields good results and is on par with unrolling without invariance on CFAs that were seen during training. The high quality of the reconstruction is expected as the networks were explicitly trained for this task. PIP-Net achieves comparable results but only for Bayer, which is a clear limitation of the method, the network should be retrained for each CFA whereas our method is lighter and works on thirteen CFAs with one training. Finally, bilinear interpolation, pan-sharpening technique, TV-ADMM and DIP work on all the tested CFAs but are not competitive enough.

Fig. 3 presents the outputs of the method without the invariance property. Indeed, the checkerboard pattern present in Fig. 3c shows that the unrolling method alone is unable to perform with this CFA. On the contrary once invariance is enabled the reconstruction is much more successful as in Fig. 3b.

On CFAs that were not seen during training, unrolling without invariance fails even though it was trained on a wide variety of CFAs. However, with invariance the method is much more robust to new CFAs, even if the results are not as good as for seen ones. The geometric transformations in the training effectively helped the network to focus on generic features of

Method	Bayer (seen)		Chakrabarti (seen)		Xtrans (unseen)	
	PSNR (dB) ↑	SSIM ↑	PSNR (dB) ↑	SSIM ↑	PSNR (dB) ↑	SSIM ↑
Ours v2	41.54 ± 2.77	$\underline{0.992 \pm 0.003}$	$\mathbf{39.50 \pm 2.85}$	$\mathbf{0.990 \pm 0.004}$	$\mathbf{35.43 \pm 1.81}$	$\mathbf{0.959 \pm 0.021}$
Ours v1	$\underline{42.30 \pm 2.89}$	$\mathbf{0.993 \pm 0.002}$	$\underline{38.90 \pm 2.49}$	0.988 ± 0.005	$\underline{29.33 \pm 1.65}$	0.839 ± 0.082
DIP [10]	31.37 ± 2.96	0.925 ± 0.039	33.20 ± 2.41	0.963 ± 0.011	30.21 ± 2.57	$\underline{0.904 \pm 0.064}$
TV-ADMM [7]	29.65 ± 3.54	0.896 ± 0.052	34.22 ± 3.04	$\underline{0.975 \pm 0.008}$	29.08 ± 3.49	0.883 ± 0.057
PIP-Net [23]	$\mathbf{43.14 \pm 1.80}$	$\mathbf{0.993 \pm 0.002}$	9.06 ± 1.91	0.143 ± 0.029	13.56 ± 1.17	0.461 ± 0.080
Pan-sharpening [4]	25.14 ± 2.75	0.859 ± 0.056	28.79 ± 2.70	0.934 ± 0.017	24.41 ± 2.45	0.812 ± 0.060
Bilinear	27.79 ± 3.21	0.869 ± 0.063	21.62 ± 2.70	0.583 ± 0.132	26.03 ± 2.92	0.791 ± 0.079

TABLE I: Mean and standard deviation of the PSNR and SSIM on two seen CFAs and one unseen. **Best/**second best.

the CFAs and avoided a specialization on specific ones.

Method	Ψ (13 trained CFAs)		3 non-trained CFAs	
	PSNR (dB) ↑	SSIM ↑	PSNR (dB) ↑	SSIM ↑
Ours v2	$\underline{40.6 \pm 2.9}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{34.5 \pm 2.3}$	$\mathbf{0.94 \pm 0.04}$
Ours v1	$\mathbf{40.8 \pm 2.8}$	$\mathbf{0.99 \pm 0.00}$	28.7 ± 2.2	0.84 ± 0.09
TV-ADMM [7]	31.7 ± 3.4	$\underline{0.94 \pm 0.03}$	$\underline{29.9 \pm 3.6}$	$\underline{0.90 \pm 0.05}$
PIP-Net [23]	12.9 ± 1.7	0.28 ± 0.05	15.1 ± 1.3	0.49 ± 0.08
Pan-sharpening [4]	26.9 ± 2.9	0.88 ± 0.05	24.8 ± 2.6	0.83 ± 0.07
Bilinear	24.8 ± 3.0	0.73 ± 0.10	26.9 ± 3.1	0.83 ± 0.07

TABLE II: Mean and standard deviation of the PSNR and SSIM on seen and unseen CFAs. **Best/**second best.

Additionally, Table II summaries the performances of the methods on Ψ and on the 3 unseen CFAs. The invariant loss does not improve the performances on Ψ but clearly gives robustness to the unrolled algorithm. The DIP method is absent from the table as it was not fast enough (above 10 hours) over the 16 studied CFAs. On the contrary all the other methods ran for less than an hour for the same task.

IV. Conclusion

This paper presents a generic method for demosaicking a wide variety of CFAs, being either traditional RGB or RGBW. To achieve state-of-the-art results the proposed method uses ADMM unrolling coupled with a novel loss function that allows the trained network to avoid certain CFA-specific features and to focus independent representations. This combination helps to blend the knowledge on the model with knowledge acquired from data, which solves demosaicking for multiple CFAs, including new ones. The algorithm, while being constrained on size, delivers performances on par with specialized demosaicking methods while being much more generic. Future work will explore ways to reduce the remaining gap between seen CFAs and unseen ones.

References

[1] B. Bayer, "Color imaging array," *United States Patent, no. 3971065*, 1976.

[2] J. Li, B. C., and H. Huang, "Universal demosaicking for interpolation-friendly RGBW color filter arrays," *IEEE Transactions on Image Processing*, vol. 32, pp. 3592–3605, 2023.

[3] M. Rafinazari and E. Dubois, "Demosaicking algorithm for the fujifilm x-trans color filter array," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 660–663.

[4] B. Gunturk, J. Glotzbach, Y. Altunbasak, R. Schafer, and R. Mersereau, "Demosaicking: color filter array interpolation," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, Jan. 2005.

[5] H. Malvar, L. wei He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, ser. ICASSP-04. IEEE, 2004.

[6] H. Tan, X. Zeng, S. Lai, Y. Liu, and M. Zhang, "Joint demosaicing and denoising of noisy bayer images with admm," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017.

[7] M. Muller, D. Picone, M. Dalla Mura, and M. O. Ulfarsson, "Model-based demosaicking for acquisitions by a rgbw color filter array," in *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2023.

[8] D. S. Tan, W.-Y. Chen, and K.-L. Hua, "DeepDemosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2408–2419, May 2018.

[9] K. Feng, Y. Zhao, J. C.-W. Chan, S. Kong, X. Zhang, and B. Wang, "Mosaic convolution-attention network for demosaicing multispectral filter array images," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 864–878, 2021.

[10] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1867–1888, Mar. 2020.

[11] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, Mar. 2021.

[12] K. Zhang, L. Van Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2020.

[13] J. M. Ramirez, J. I. Martínez-Torre, and H. Arguello, "LADMM-net: An unrolled deep network for spectral image fusion from compressive data," *Signal Processing*, vol. 189, p. 108239, Dec. 2021.

[14] D. Picone, M. Dalla Mura, and L. Condat, "Joint demosaicing and fusion of multiresolution coded acquisitions: A unified image formation and reconstruction method," *IEEE Transactions on Computational Imaging*, vol. 9, pp. 335–349, 2023.

[15] C. Laroche, A. Almansa, and M. Tassano, "Deep model-based super-resolution with non-uniform blur," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2023.

[16] L. Condat, D. Kitahara, A. Contreras, and A. Hirabayashi, "Proximal splitting algorithms for convex optimization: A tour of recent advances, with new twists," *SIAM Review*, vol. 65, no. 2, pp. 375–435, 2023.

[17] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[18] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[19] D. You, J. Xie, and J. Zhang, "Ista-net++: Flexible deep unfolding network for compressive sensing," in *IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, Jul. 2021.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[21] D. Chen, J. Tachella, and M. E. Davies, "Robust equivariant imaging: a fully unsupervised framework for learning to image from noisy and partial measurements," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv, 2021, pp. 5647–5656.

[22] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[23] S. M. A. Sharif, R. Ali Naqvi, and M. Biswas, "Beyond joint demosaicking and denoising: An image processing pipeline for a pixel-bin image sensor," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Jun. 2021.