International Journal of Engineering & Technology, 14 (1) (2025) 14-23

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
https://doi.org/10.14419/50xc1e97
Research paper

2D-filter design based on Volterra equation and machine learning to tackle the aliasing effect in image demosicing

Nodjigoto Blague [1]*, Boukar Ousman [1], Libouga Li Gwet David [2], Kamgang Jean Claude [3], Ndogotar Nelio [4]

[1] *University of Ngaoundere, ENSAI, Department of Electrical, Energy and Automatic Engineering, Cameroon*
[2] *University of Ngaoundere, EGCIM, Department of Robotic and Industrial Computing, Cameroon*
[3] *University of Ngaoundere, ENSAI, Department of Mathematics and Computer Sciences, Cameroon*
[4] *Department of Mathematics and Informatics, Sarh University, Chad*
**Corresponding author E-mail: ndoningalain@gmail.com*

Received: January 23, 2025, Accepted: February 9, 2025, Published: March 20, 2025

Abstract

For nonlinear problems, the discrete Volterra equation is a helpful model. An aliasing effect between components from a CFA image is known as the main problem in Demosaicing on frequency-domain analysis, which denotes a nonlinear problem. The discrete Volterra equa-tion can be used as a model to solve Demosaicing. A suitable Volterra kernel is required. We show that there is an analogy between the weights and bias of the perceptron and the coefficients of the Volterra kernel. This analogy allows us to discover the coefficients of 2D-nonlinear filters by numerically solving the discrete Volterra equation. On the images from the Kodak database, this Demosaicing procedure is applied in accordance with the Volterra equation's order and the Volterra kernel's size. Effectiveness is judged using several measures. The study's findings show that the aliasing effect is lessened depending on the order of the Volterra equation and the Volterra kernel size chosen. Our mean signal-to-noise ratio performance, using a 3x3 kernel size and the second-order Volterra equation, is 37.7 dB.

Keywords: *2-Dimensional Discrete Volterra Equation; Perceptron; Aliasing; Demosaicing.*

1. Introduction

In most single-sensor cameras, the colour images produced come from a CFA image from the sensor and a demosaicing algorithm. In frequency-domain analysis, demosaicing is the extraction of luminance and chrominance components from a CFA image. These components are mostly aliased. Linear Filters have been proposed in the frequency domain by [1] and [2]. after them, most demosaicing authors continued to work in the spatial-domain, where demosaicing is defined as reconstructing a full three-colour representation of colour images (Red, Green and Blue) by estimating the missing pixel components in each colour plane. We noticed that there is a nonlinear structure in the discrete Volterra equation of order higher than or equal to two. Since the aliasing effect is nonlinear, we think the discrete Volterra equation can handle it. In this case, it would be necessary to find the appropriate Volterra filter kernels to estimate the luminance and chrominance components of CFA images database. We found an analogy between the parameters of the perceptron and the coefficients of the Volterra filter, which enables us to obtain these coefficients and simultaneously suggest a numerical approach for resolving the two-dimensional discrete Volterra equation. Therefore, the question is whether or not the aliasing effect can be solved using these Volterra filter coefficients. To address this query, section 2 describe the suggested approach. Experimental results are presented in section 3, and a conclusion in section 4.

2. Proposed method

We proposed a numerical resolution of the discrete Volterra equation by using the perceptron to carry out Demosaicing. In this section, we first introduce the perceptron and its training concept to help better understand the approach used. Next, we show the similarities we found between the Volterra kernel and the perceptron's parameters. Lastly, we present the meaning of Demosaicing in the frequency domain and describe our suggested algorithm.

2.1. Perceptron and it training

Copyright © Nodjigoto Blague et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The perceptron is a supervised learning technique in machine learning that was developed in 1943 by Warren McCulloch and Walter Pitts [3]. Its representation is shown in figure 1. A perceptron is a function that determines the output for a given input.

- Inputs: $X(x_1, \cdots, x_n)$
- Perceptron parameters: weights W $(w_1, \cdots, w_n)$ and b the bias
- Activation function: a(·)
- Output: y.

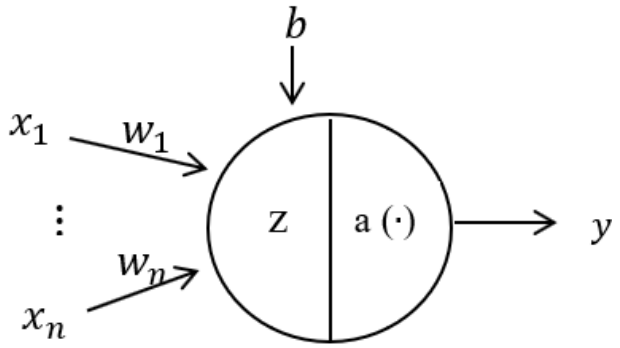


Fig. 1: The Perceptron Representation.

Training the perceptron is to find a good set of weights and bias by an optimization algorithm like, gradient descent method [4]. Inputs $(x_1, \cdots, x_n)$ and output y are known. W $(w_1, \cdots, w_n)$ and b are initially fixed randomly. We choose an activation function and a cost function, such as the sigmoid and likelihood functions, respectively:

$a(z) = (1 + e^{-z})^{-1}$; $l = -y \cdot \log a - (1 - y)\log(1 - a)$

A learning rate $\alpha \in [0,1]$ is also used in the gradient descent approach. For one inputs $(x_1, \cdots, x_n)$ there is one output y. For m inputs we have the input data matrix

$X = \begin{bmatrix} x_1^1 & \dots & x_n^1 \\ \vdots & \vdots & \vdots \\ x_1^m & \dots & x_n^m \end{bmatrix} \in R^{m \times n}$ and the corresponding output vector $Y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \in R^{m \times 1}$

Let $Z = \begin{bmatrix} z_1 \\ \vdots \\ z_m \end{bmatrix} \in R^{m \times 1}$, the bias $b \in R$,

Figure 2 depicted the flowchart of gradient descent method which is construct by equations (1) to (4). The purpose is to find after each epoch, a good set of W $(w_1, \cdots, w_n)$ and b based on the minimal lost function.

$$\begin{cases} Z = W \cdot X + b \\ A = (1 + e^{-Z})^{-1} \end{cases} \quad (1)$$

$$L = \frac{-1}{m} \sum_{i=1}^{m} Y \cdot \log A + (1 - Y) \log(1 - A) \quad (2)$$

$$\begin{cases} \frac{\partial L}{\partial W} = \frac{1}{m} \cdot X^t (A - Y) \\ \frac{\partial L}{\partial b} = \frac{1}{m} \cdot \sum_{i=1}^{m} (A - Y) \end{cases} \quad (3)$$

$$\begin{cases} W = W - \alpha \cdot \frac{\partial L}{\partial W} \\ b = b - \alpha \cdot \frac{\partial L}{\partial b} \end{cases} \quad (4)$$

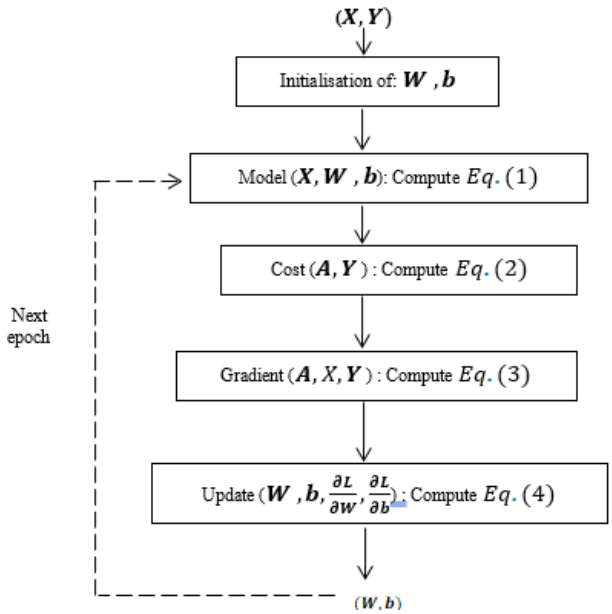


Fig. 2: Flowchart of Gradient Descent Method.

2.2. Estimation of Volterra filter coefficients with perceptron training

Discrete-time Volterra models are often used in the study of nonlinear physical and physiological systems using stimulus-response data [5]. Equation (5) shows one-dimensional Discrete Volterra equation with order's K and kernel size's N_1^K. For an input X and an output Y there is K-order kernel h_K.

$$Y(n) = h_0 + \sum_{p=1}^{K}\left\{\sum_{i_1=a}^{N_1-1}\cdots\sum_{i_p=a}^{N_1-1} h_p(i_1,\cdots,i_p)\cdot\prod_{\lambda=1}^{p} X(n-i_\lambda)\right\} \quad (5)$$

For example, equation (6) and equation (7) show the first order, and the second order of one-dimensional Discrete Volterra equation respectively.

$$Y(n) = h_0 + \sum_{i_1=0}^{N_1-1} h_1(i_1)\cdot X(n-i_1) \quad (6)$$

$$Y(n) = h_0 + \sum_{i_1=0}^{N_1-1} h_1(i_1)\cdot X(n-i_1) + \sum_{i_1=0}^{N_1-1}\sum_{i_2=0}^{N_1-1} h_2(i_1,i_2)\cdot X(n-i_1)\cdot X(n-i_2) \quad (7)$$

We are interested on image processing, where the signal is two-dimensional. equation (8) shows the two-dimensional Discrete Volterra equation with order's K and kernel size's N_1^{2K}.

$$Y(n,m) = h_0 + \sum_{p=1}^{K}\left\{\sum_{i_1=0}^{N_1-1}\sum_{j_1=0}^{N_1-1}\cdots\sum_{i_p=0}^{N_1-1}\sum_{j_p=0}^{N_1-1} h_p(i_1,j_1\cdots,i_p,j_p)\cdot\prod_{\lambda=1}^{p} X(n-i_\lambda,m-j_\lambda)\right\} \quad (8)$$

For example, equation (9) and equation (10) show the first order, and the second order of two-dimensional Discrete Volterra equation respectively.

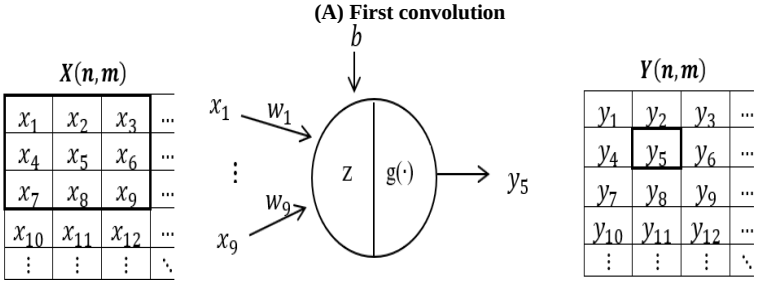
$$Y(n,\ m) = h_0 + \bar{h}_1[X(n,\ m)] \quad (9)$$

$$Y(n,\ m) = h_0 + \bar{h}_1[X(n,\ m)] + \bar{h}_2[X(n,m)] \quad (10)$$

With

$$\begin{cases} \bar{h}_1[X(n,\ m)] = \sum_{i_1=0}^{N_1-1}\sum_{j_1=0}^{N_1-1} h_1(i_1,\ j_1)\,X(n-i_1,\ m-j_1) \\ \bar{h}_2[X(n,\ m)] = \sum_{i_1=0}^{N_1-1}\sum_{j_1=0}^{N_1-1}\sum_{i_2=0}^{N_1-1}\sum_{j_2=0}^{N_1-1} h_2(i_1,\ j_1,i_2,\ j_2)\,X(n-\ i_1,\ m-\ j_1)X(n-\ i_2,\ m-\ j_2) \end{cases}$$

(A) First convolution



(B) second convolution along the column

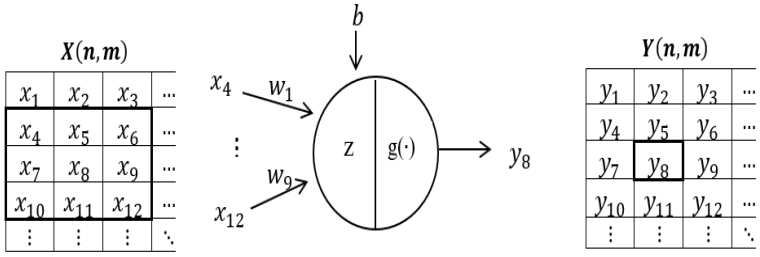


Fig. 3: Perceptron to Find the First Order Kernel.

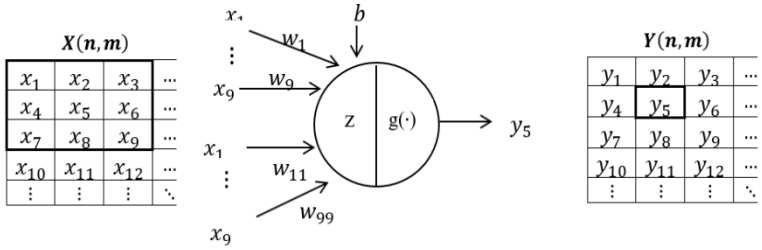


Fig. 4: Perceptron to Find the Second Order Kernel.

The purpose here in the two-dimensional Discrete Volterra equation, is to find appropriate kernels h_0, …, h_K for a database where each element has X-value and it corresponding Y-value. Kernels are filters which are commonly used to refer to a system that is designed to extract information about the prescribed quantity of interest X from noisy data. It's a recursive algorithm which test all the database, thus this filter is an adaptive filter.

More, it's 2-Dim FIR (finite impulse response) filters. The linear phase of the transfer function makes them useful for many applications. In the same logic with the work of [5] in one-dimensional, we've observed a parallel between the kernels h_0, …, h_K and the perceptron parameters (weights W and b the bias).

Let's look at figure 3, where g(.) is the activation function. For an input matrix X (n, m) and an output matrix Y (n, m) known, the perceptron can win optimal parameters (weights W and b the bias). The perceptron takes a N_1 x N_1=3x3 neighbourhood in the input matrix X (n, m) and a corresponding output in Y (n, m). Throughout the entire matrix, the process is repeated in both vertical and horizontal directions. The first step is shown in figure 3(a), which is followed by figure 3(b) for the next vertical position, and so on. The equation that corresponds to figure 3(a) is shown in (11). There is a similar process in figure 3(b). Let's rewrite equation (9) as equation (12) with N_1 x N_1=3x3 When we select the activation function g(.) as a linear function, equations (11) and (12) are analogous.

$$y_5 = g\left(b + \sum_{i=1}^{9} w_i \cdot x_i\right) \tag{11}$$

$$Y(n,\ m) = h_0 + \sum_{i_1=0}^{2} \sum_{j_1=0}^{2} h_1(i_1,\ j_1)\, X(n - i_1,\ m - j_1) \tag{12}$$

Lets $g(x) = x$. According to figure 3(a), we have $(n, m) = (2,\ 2)$, so we can observe the following analogy,

- Output: $y_5 = Y(2, 2)$
- The bias: $b = h_0$
- Weights: $\{w_1, \cdots, w_9\} = h_1(i_1,\ j_1)$ with $i_1\ ,\ j_1\ \in \{0, 1, 2\}$
- Inputs: $\{x_1, \cdots, x_9\} = X(2 - i_1,\ 2 - j_1)$ with $i_1\ ,\ j_1\ \in \{0, 1, 2\}$

From figure 3(b), we have the same analogy with $(n, m) = (3,\ 2)$. equations (11) and equations (12) shows us, how to find zero order (h_0), and the first order (h_1) kernels of Volterra equation through the perceptron parameters (weights W and b the bias) trained. If we need the second order (h_2) kernel of equations (10), all we must do is add additional inputs, as illustrated in figure 4 where,

- Output: $y_5 = Y(2, 2)$
- The bias: $b = h_0$
- Weights: $\{w_\lambda\} = h_1(i_1,\ j_1)$ with $i_1\ ,\ j_1\ \in \{0, 1, 2\}$ and $\lambda\ \in \{1, 2, 3, 4, 5, 6\ ,7, 8, 9\}$ $\{w_{kp}\} = h_2(i_1,\ j_1, i_2,\ j_2)$ with $i_1\ ,\ j_1, i_2,\ j_2\ \in \{0, 1, 2\}$ and $k\ ,\ p\ \in \{1, 2, 3, 4, 5, 6\ ,7, 8, 9\}$
- Inputs: $\{x_1, \cdots, x_9\} = X(2 - i_1,\ 2 - j_1)$ with $i_1\ ,\ j_1\ \in \{0, 1, 2\}\ \{x_1, \cdots, x_9\} = X(2 -\ i_2,\ 2 -\ j_2)$ with $i_2\ ,\ j_2\ \in \{0, 1, 2\}$

Additional inputs are required if we require a higher kernel order. From kernels order upper than two, the perceptron have a nonlinear action. Furthermore, since the activation function is still a linear function, using a multilayer perceptron (a link of many perceptrons) is not required because the model is polynomial. Notice that, it's necessary to use a linear function as activation function to have a similarity with the two-dimensional Discrete Volterra equation.

2.3. Demosaicing on frequency-domain analysis

Demosaicing approach in frequency domain is suggested by [1]. This method's guiding idea is to show CFA image as a combination of a luminance component at low spatial frequencies and two chrominance components modulated at high spatial frequencies, then to estimate the image demosaiced by selecting the frequencies appropriately. The formalism is given by [2].

Suppose that for each component k of a colour image, $k \in \{R, G, B\}$, there is a corresponding underlying signal f^k. Demosaicing then consists of searching in each pixel an estimate $\hat{f}^k$. Let us also assume that there exists a signal f^{CFA} sub underlying the CFA image I^{CFA}, called here CFA signal and matching at every pixel. The value of the CFA signal at each pixel of coordinates (x, y) can be stated as the total of the signals' values f^k spatially sampled as presented in equations (13).

$$f^{CFA}(x, y) = \sum_{k=R,G,B} f^k(x, y) \cdot m^k(x, y) \tag{13}$$

Where $m^k(x,y)$ is the sampling function of the component k, shown in equations (14) for a corresponding Bayer CFA shown in figure 5.

$$\begin{cases} m^R(x,y) = \frac{1}{4}\,[1-(-1)^x][1+(-1)^y] \\ m^G(x,y) = \frac{1}{2}\,[1+(-1)^{x+y}] \\ m^B(x,y) = \frac{1}{4}\,[1+(-1)^x][1-(-1)^y] \end{cases} \quad (14)$$

$f^G(0,0)$	$f^R(1,0)$	$f^G(2,0)$	⋯
$f^B(0,1)$	$f^G(1,1)$	$f^B(2,1)$	⋯
$f^G(0,2)$	$f^R(1,2)$	$f^G(2,2)$	⋯
⋯	⋯	⋯	⋯

Fig. 5: Bayer CFA Structure.

$$\text{Let's } \begin{bmatrix} f^L(x,y) \\ f^{C_1}(x,y) \\ f^{C_2}(x,y) \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ -1 & 2 & -1 \\ -1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f^R(x,y) \\ f^G(x,y) \\ f^B(x,y) \end{bmatrix} \quad (15)$$

Signal f^{CFA} becomes,

$$f^{CFA}(x,y) = f^L(x,y) + f^{C_1}(x,y)(-1)^{x+y} + f^{C_2}(x,y)[(-1)^x - (-1)^y]$$

$$= f^L(x,y) + f^{C_1}(x,y)e^{j2\pi(x+y)/2} + f^{C_2}(x,y)\left[e^{j2\pi(x)/2} - e^{j2\pi(y)/2}\right] \quad (16)$$

The CFA signal can therefore be interpreted as the sum of a luminance component f^L in base band, a chrominance component f^{C_1} modulated at the spatial frequency (horizontal and vertical) (0.5, 0.5), and another modulated chrominance component f^{C_2} at the two spatial frequencies (0.5, 0) and (0, 0.5). This interpretation can be verified simply on an achromatic image, for which $f^R = f^G = f^B$, the two components of chrominance are then zero. If we can estimate the functions f^L , f^{C_1} and f^{C_2} in each pixel from the CFA signal, the estimated levels $\hat{f}^R$, $\hat{f}^G$ and $\hat{f}^B$ of the colour components R, G and B, are then found simply by:

$$\begin{bmatrix} \hat{f}^R(x,y) \\ \hat{f}^G(x,y) \\ \hat{f}^B(x,y) \end{bmatrix} = \begin{bmatrix} 1 & -1 & -2 \\ 1 & 1 & 0 \\ 1 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} \hat{f}^L(x,y) \\ \hat{f}^{C_1}(x,y) \\ \hat{f}^{C_2}(x,y) \end{bmatrix} \quad (17)$$

To do this, [1] apply the Fourier transform of CFA signal (equations (16), which provides:

$$F^{CFA}(u,v) = F^L(u,v) + F^{C_1}(u-0.5, v-0.5) + F^{C_2}(u-0.5, v) - F^{C_2}(u, v-0.5) \quad (17)$$

By observing the distribution of the energy of a CFA image in the frequency plane, it is concentrated in nine positions. $F^L(u,v)$ The centred value on the spatial frequencies. $F^{C_2}(u-0.5, v)$ is on the u-axis of horizontal frequencies and $F^{C_2}(u, v-0.5)$ on the v-axis of vertical frequencies. $F^{C_1}(u-0.5, v-0.5)$ is in the diagonal zones of the plan.
The design of filters to isolate these three components (f^L, f^{C_1}, f^{C_2}) on the CFA image f^{CFA}, is therefore the key to this method. Their bandwidth must be chosen carefully, given the mutual overlaps (aliasing) of the spectrum of the three functions.

2.4. Demosaicing algorithm propose

Algorithm that we propose is carried out in two steps. The first step consists of determining the appropriate Volterra kernel coefficients to extract the luminance and chrominance components from the CFA image. The second step is to construct the estimated colour image from the estimated luminance and chrominance components.

2.4.1. Estimation of volterra kernel coefficients

The usual approach for evaluating a demosaicing algorithm consists of simulating CFA images from colour images (obtained from three-sensor devices) and the chosen CFA filter mask. Then apply the demosaicing algorithm on the CFA image and compare the estimated colour image with the initial colour image [6], [7]. Thus, we choose twelve colour images from the Kodak base as shown in figure 6. We consider the Bayer filter mask [8], following the configuration of figure 5. The twelve CFA images (f^{CFA}) are simulated. The RGB components (f^R, f^G, f^B) of Kodak images are also used to construct the luminance (f^L) and chrominance components (f^{C_1}, f^{C_2}) according to equations (15) which will serve as references. The next step is to find appropriate kernels $h_0, \ldots, h_K$ according to equations (8) when,

- Input: $X(n,m) = f^{CFA}(n,m)$
- Outputs: $Y(n,m) = f^L(n,m)$ or $Y(n,m) = f^{C_1}(n,m)$ or $Y(n,m) = f^{C_2}(n,m)$

depending on whether we are looking for the Volterra kernels for luminance or for chrominance
f^{CFA} Is therefore, the input of the perceptron. The outputs are f^L or f^{C_1} or f^{C_2}. The perceptron is trained by the gradient descent method [4]. The weights and bias are then identified and associated to Volterra kernels $h_0, \ldots, h_K$. Notice that, the database used for training the perceptron depend on the size of image. Each pixel of image, is an element of the database.



Fig. 6: The Twelve Images Kodak Database Used [9].

2.4.2. Demosaicing

When we have Volterra kernels $h_0, \dots, h_K$, for the luminance (f^L) and chrominance components (f^{C_1}, f^{C_2}), we used it with the CFA image according to equations (19) for estimation of luminance ($\hat{f}^L$) and chrominance components ($\hat{f}^{C_1}, \hat{f}^{C_2}$).

$$Y(n,m) = h_0 + \sum_{p=1}^{K} \left\{ \sum_{i_1=0}^{N_1-1} \sum_{j_1=0}^{N_1-1} \cdots \sum_{i_p=0}^{N_1-1} \sum_{j_p=0}^{N_1-1} h_p(i_1, j_1 \cdots, i_p, j_p) \cdot \prod_{\lambda=1}^{p} f^{CFA}(n - i_\lambda, m - j_\lambda) \right\} \quad (19)$$

It means that,

$Y(n,m) = \hat{f}^L(n,m)$ or $Y(n,m) = \hat{f}^{C_1}(n,m)$ or $Y(n,m) = \hat{f}^{C_1}(n,m)$.

Where Volterra equation order's K and kernel size's N_1^{2K} are chosen. With the estimated luminance ($\hat{f}^L$) and chrominance components ($\hat{f}^{C_1}, \hat{f}^{C_2}$) we used equations (17) to have estimated levels $\hat{f}^R$, $\hat{f}^G$ and $\hat{f}^B$, which are the components of the colour image estimated by our demosaicing algorithm. Thus, for evaluating algorithm we must compare,

- The component f^L with the component $\hat{f}^L$
- The components (f^{C_1}, f^{C_2}) with the component ($\hat{f}^{C_1}, \hat{f}^{C_2}$)
- The colour image (f^R, f^G, f^B) with the colour image ($\hat{f}^R$, $\hat{f}^G$, $\hat{f}^B$)

All of this in order to known if the problem of aliasing between luminance and chrominance components in Bayer CFA image have been settled.

3. Experimental results

In the following sections, three sets of results are presented. The first section shows the performances of the perceptron training and second order Volterra filter coefficients (kernel coefficients) obtains from the training. The second section explores the links which exist between the choice of the parameters of the Volterra equation (equation order's K and kernel size's N_1^{2K}) and the resolution of aliasing between luminance and chrominance components. To evaluate the performance of the proposed method with state-of-the-art of Demosaicing algorithms, the last section shows, colour peak-signal to noise ratio (CPSNR) between colour image (f^R, f^G, f^B) and estimated colour image ($\hat{f}^R$, $\hat{f}^G$, $\hat{f}^B$).

3.1. Perceptron training of second-order Volterra equation

Twelve colour images from the Kodak base [9] are considered. Each colour image has been normalized (grey level range between zero and one) and a simulation of the Bayer CFA image [8]. Luminance and chrominance components have been done according to equation (15) . We set CFA image in the perceptron input. We set luminance or chrominance components in the perceptron output depending on whether we want to obtain the Volterra filter coefficients to estimate the luminance or chrominance components. For experimental results we set, $K = 2$ and $N_1 = 3$, which means that each element of the database during the perceptron training, have a vector with $3^2 + 3^2 \cdot 3^2 = 90$ components (from CFA image) in input and a vector with one component (from luminance or chrominance components) in output. On all the dataset, 20% have been taken randomly as validation set. We don't use a test set, because after the training, we test the perceptron model during the Demosaicing process by using it weights and bias.

Thus, for each of the twelve images which have 768x512 in size, we take only a few sections of 100x100 because we want to train only a neuron, which need a few databases in opposite of a neural network. It means, for one image we have $100 \times 100 = 10^4$ elements. With twelve images the database has $12 \cdot 10^4$ elements where $12 \cdot 10^4 \times 20\% = 24000$ elements are randomly taken as validation set and $12 \cdot 10^4 \times 80\% = 96000$ elements as training set.

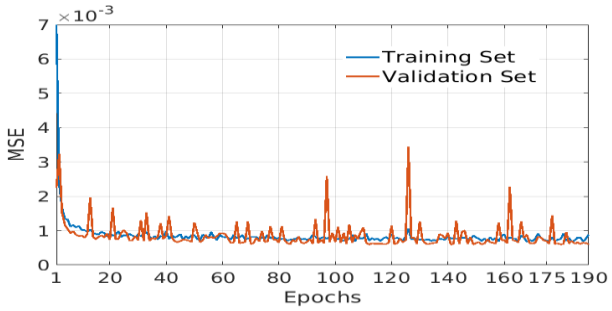


Fig. 7: Training and Validation Loss Curves for Luminance Component.

Figure 7, present loss curves obtained when the luminance component is set at the perceptron output. The loss function choose is the mean-square error (MSE). Generally, two main problems encountered during the model training, namely, underfitting and overfitting. The problem of underfitting is observed when there is a significant difference between the training loss curve and the validation loss curve. Overfitting is observed when the validation error is greater than the training error when the number of epochs increases [10]. According to figure 7, underfitting and overfitting are not relevant. The MSE converge under 0.001. The perceptron training is well done for luminance component.

The results are similar for chrominance components, we did not present it to avoid overloading the results. After the perceptron training, the 90 weights and the bias are taken. Following the analogy between the weights, the bias and Volterra filter coefficients presented in section 2.2, the bias is the zero-order coefficient (h_0), the first 9 weights are the coefficients of first-order (h_1) and the 81 last weights are the coefficients of second-order (h_2). The table 1, shows those coefficients in the case of the luminance component.

Table 1: Volterra Filter Coefficients for Luminance Component

h_0				
		$j=-1$	$j=0$	$j=1$
$h_1(i,j)$	$i=-1$	0.054	0.074	0.086
	$i=0$	0.123	0.359	0.128
	$i=1$	0.085	0.087	0.063

$h_2(i_1, j_1, i_2, j_2)$		(i_2, j_2)								
		$(-1,-1)$	$(0,-1)$	$(1,-1)$	$(-1,0)$	$(0,0)$	$(1,0)$	$(-1,1)$	$(0,1)$	$(1,1)$
(i_1, j_1)	$(-1,-1)$	-0.107	-0.011	0.126	-0.088	-0.039	-0.156	0.006	-0.041	-0.09
	$(0,-1)$	0.148	-0.039	0.213	-0.013	0.088	0.001	-0.023	0.144	0.006
	$(1,-1)$	-0.113	-0.124	-0.159	-0.175	0.017	-0.046	0.096	-0.111	0.111
	$(-1,0)$	0.289	-0.049	0.146	-0.089	-0.123	-0.105	-0.027	0.021	0.108
	$(0,0)$	-0.042	-0.183	-0.021	-0.021	0.614	-0.170	-0.055	-0.278	0.242
	$(1,0)$	0.141	0.035	0.138	0.122	-0.147	0.015	0.023	0.165	0.073
	$(-1,1)$	-0.098	0.054	-0.126	0.170	0.036	-0.014	-0.106	0.007	0.060
	$(0,1)$	0.052	-0.152	0.145	0.075	-0.063	-0.088	0.024	0.035	0.144
	$(1,1)$	0.070	-0.099	-0.089	-0.098	-0.117	0.005	-0.048	-0.099	-0.133

The perceptron model is tested, by using the several Volterra filter coefficients (h_0, h_1, h_2) and CFA image (I_{CFA}) according to equations (19) to estimate luminance component ($\hat{I}_{Lum}$). The process is similar for estimation of chrominance components ($\hat{I}_{C_1}, \hat{I}_{C_2}$).

$$\hat{I}_{Lum}(n, m) = h_0 + \bar{h}_1[I_{CFA}(n, m)] + \bar{h}_2[I_{CFA}(n, m)] \quad (20)$$

with,

$$\begin{cases} \bar{h}_1[I_{CFA}(n, m)] = \sum_{i_1=0}^{2}\sum_{j_1=0}^{2} h_1(i_1, j_1)\, I_{CFA}(n - i_1, m - j_1) \\ \bar{h}_2[I_{CFA}(n, m)] = \sum_{i_1=0}^{2}\sum_{j_1=0}^{2}\sum_{i_2=0}^{2}\sum_{j_2=0}^{2} h_2(i_1, j_1, i_2, j_2)\, I_{CFA}(n - i_1, m - j_1) I_{CFA}(n - i_2, m - j_2) \end{cases}$$

(n, m) : the pixels position in image $\hat{I}_{Lum}$ or I_{CFA}

3.2. Analysis of several order volterra equation and kernel dimensions

In this section we present the performance of the Volterra equation for several orders and kernel dimensions. Table2 shows the number of coefficient of Volterra kernel for those several orders and kernel dimensions set experimentally during implementation. $N \times N$ is, the kernel dimensions and h_i the kernel with i-order. The eight cases in table 2 have been noted according to table 3 and implemented on the twelve colour images of Kodak database. For example, «3x3-h0h1h2» means, Volterra equation from zero-order to second-order for 3×3 kernel size's, like equations (10).

As we explained in part 2.3, the CFA image have a luminance component, mainly modulated at low frequencies, and two chrominance components, mainly modulated at high frequencies. Demosaicing can be interpreted as an estimate of luminance and chrominance components which suffer from aliasing effect. Consequently, the four potential frequency domain artefacts are: blur, grid effect, false colour and watercolour. When the bandwidth of the filter applied to the CFA image to estimate the luminance is too narrow, a blurring phenomenon is present in the estimated colour image. When the bandwidth of this filter is too wide and selects therefore frequencies in the region representing the high frequencies of the chrominance, a grid effect can result. Furthermore, false colours can be caused by the mutual overlap of the estimated luminance and chrominance spectrum, when the spectrum of the filter used to estimate the chrominance occupies too wide band. Finally, if the spectrum of the filter used to estimate chrominance is too narrow, the watercolour effect is likely to appear. From our readings, we find three metrics which allow us to evaluate the four artefacts mentioned above.

Table 2: Coefficient Number of Volterra Kernel Set Experimentally

N × N	h_0	h_1	h_2	h_3	Total
3 × 3	1	$3^2 = 9$	$3^2 \times 3^2 = 81$	$3^2 \times 3^2 \times 3^2 = 729$	820
5 × 5	1	$5^2 = 25$	$5^2 \times 5^2 = 625$		651
7 × 7	1	$7^2 = 49$	$7^2 \times 7^2 = 2401$		2451
11 × 11	1	$11^2 = 121$			122

Table 3: Notation of the Eight Configurations of Volterra Equation Implemented

Cases	1	2	3	4
N × N-h_i	3x3-h0h1	3x3-h0h1h2	3x3-h0h1h2h3	5x5-h0h1

Cases	5	6	7	8
N × N-h_i	5x5-h0h1h2	7x7-h0h1	7x7-h0h1h2	11x11-h0h1

MAHMOUD et al [11] have suggested a metric for blurriness of images. The algorithm is as follows. For every pixel in both the reference image and the blurred image, compute the difference between the center pixel intensity and its 8 neighbour pixels. Store the maximum values in matrix A for the reference image, and in matrix B for the blurred image. Compute μ_A the average of matrix A and μ_B the average of matrix B. The measurement of image blurriness is calculated as percentage, according to equations (21). $\text{Blur}_{\%}$ is better for small values.

$$\text{Blur}_{\%} = \frac{100 \times |\mu_A - \mu_B|}{\mu_A} \tag{21}$$

Yanqin et al (2007) [12], proposed a measurement of false colours. The algorithm is as follows. Compute the difference between the reference image ($I^k_{x,y}$) and image with false colours ($\hat{I}^k_{x,y}$) which are two colour images ($k = R, G, B$). Take for each pixel (P), the maximum of the three colour components. Count the number of pixels for which, the previous difference is greater than a threshold (T). The measurement of false colour is calculated as percentage, according to equations (22). where $X \cdot Y$ is the size of image used. $FC_{\%}$ is better for small values.

$$FC_{\%} = \frac{100}{X \cdot Y} \text{Card}\left\{P(x,y) \mid \max_{k=R,G,B} \left|I^k_{x,y} - \hat{I}^k_{x,y}\right| > T\right\} \tag{22}$$

Grid effect and watercolour, can be jointly considered as zipper effect, or edge blurring that occurs in pattern along an edge. This effect occurs when the demosaicing averages pixels values over an edge especially in the red and blue planes, resulting in its characteristic blur. Wenmiao Lu et Yap-Peng Tan., (2003) [13] proposed a measurement of zipper effect. The algorithm is as follows: for each pixel (P) in the reference image I, identify the neighboring pixel (P_n) whose colour is closest to that of (P) in the CIE $L^*a^*b^*$ colour space according to the Euclidean norm. Compute distances $\Delta I(P)$ and $\Delta\hat{I}(P)$ between the two colour points (P, P_n) in reference image and image with zipper effect respectively. Compute the difference between those two distances. Count the number of pixels for which, the previous difference is greater than 2,3. The measurement of zipper effect is calculated as percentage, according to equations (23), where $X \cdot Y$ is the size of image used. $ZE_{\%}$ is better for small values.

$$ZE_{\%} = \frac{100}{X \cdot Y} \text{Card}\left\{P(x,y) \mid \Delta\hat{I}(P) - \Delta I(P) > 2{,}3\right\} \tag{23}$$

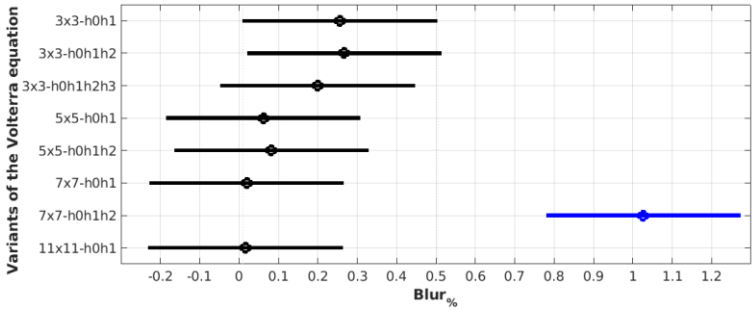


Fig. 8: Tukey's HSD Test of $\text{Blur}_{\%}$.

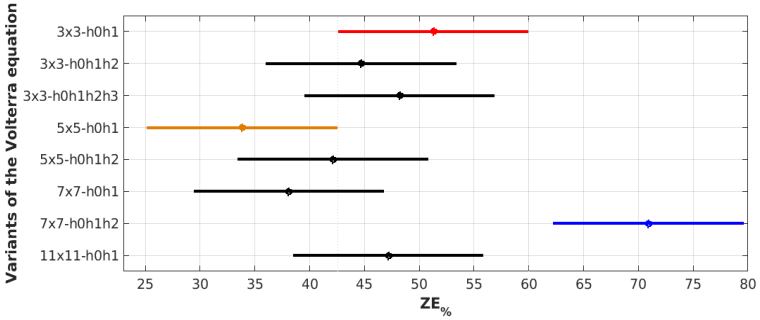


Fig. 9: Tukey's HSD Test of $ZE_{\%}$.

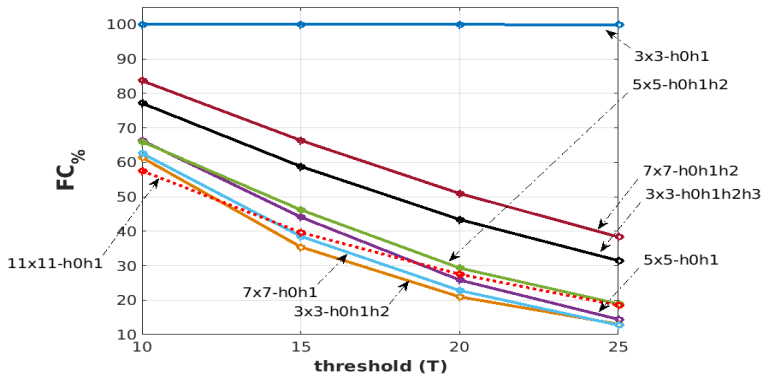


Fig. 10: Training Curves.

On the twelve colour images of Kodak database, the eight cases of Volterra equation from table 3 is implemented and evaluated through $Blur_{\%}$, $ZE_{\%}$ and $FC_{\%}$. The Tukey's HSD (honestly significant difference) test have been done to appreciate the significant differences that could exist between the eight variants of the Volterra equation according to.$Blur_{\%}$ and $ZE_{\%}$. Results are show in figure8 and figure9. For $FC_{\%}$, four thresholds values were considered, $T \in \{10; 15; 20; 25\}$. the results obtained are presented in the figure 10.
In figure8 and figure9, we have line segments that depict the interval performances of the eight variations (or Volterra equation configurations). If the segments of two variants' performances don't overlap based on the values on the x-axis, then the variants will be considered statistically different. The configurations are not statistically different when two segments overlap, however the segments with smaller x-axis values may be viewed as better. We can therefore conclude from Figure 8 that Configuration «7x7-h0h1h2» differs statistically from the other seven configurations. Furthermore, due to their low (Blur)_% values, the configurations «7x7-h0h1» and «11x11-h0h1» have the best results. Based on Figure 9, it can be inferred that Configuration «7x7-h0h1h2» differs statistically from the remaining seven configurations. Additionally, there are statistical differences between Configurations «3x3-h0h1», «5x5-h0h1», and «7x7-h0h1h2». Due to its minimal (ZE)_% value, configuration «5x5-h0h1» has the best performance, followed by «7x7-h0h1», which has the second-best performance. Therefore, the configuration «7x7-h0h1» has the best performance based on the (Blur)_% and (ZE)_% values. An observation from (Blur)_% appears in figure 8 when «...-h0h1», «...-h0h1h2h3» are taken into consideration as Volterra configurations with odd order. For an odd order's, largest is the size of the kernel, better we extract the luminance component ($\hat{f}^{L}$) in CFA image. The same observation can be made for the $ZE_{\%}$ in figure9. Which means that, grid effect and watercolour are less present for an odd order's and largest kernel size's. Among the eight configurations, «7x7-h0h1», is the best one which reduce grid effect and watercolour and so better extract the luminance component ($\hat{f}^{L}$) in CFA image. When it comes to the false colours in Figure 10, we can observe that the case «3x3-h0h1» produces a completely flawed result, while the case «3x3-h0h1h2» performs the best due to the smallest (FC)_% value for many thresholds. If we do not consider the case «3x3-h0h1», an observation emerges in figure 10, for a fixed kernel size, increasing the Volterra order decreases performance. Thus, among the eight configurations, «7x7-h0h1» is the best to extract the luminance component luminance ($\hat{f}^{L}$) from the chrominance components ($\hat{f}^{C_1}, \hat{f}^{C_2}$) in CFA image and «3x3-h0h1h2» is the best to extract the chrominance components ($\hat{f}^{C_1}, \hat{f}^{C_2}$) from the luminance component luminance ($\hat{f}^{L}$) in CFA image. In the next section, we focus on the configuration «3x3-h0h1h2», because it's better to extract two ($\hat{f}^{C_1}, \hat{f}^{C_2}$) from the three components ($\hat{f}^{L}, \hat{f}^{C_1}, \hat{f}^{C_2}$) that we need during the Demosaicing process.

3.3. Demosaicing results

In the experiments, we conduct the performance comparison of the proposed algorithm and some recent state-of-the-art demosaicing algorithms which have performed in the Kodak database images. The method of Kiku et al (2016) [6] noted M1, the method of Oh et al (2017) [7] noted M2, the method of Kim et al (2020) [14] noted M3, and the method of Jeong et al (2022) [15] noted M4. Method M1 used the residual model for demosaicing. The second M2 is a colourization-based method. The third M3 is a demosaicing algorithms based on deep learning. Method M4 used colour difference channel and Laplacian pyramid decomposition of the estimated white channel for demosaicing. Table 4 shows the results obtains, where the best performances are in bold. We used the proposed «3x3-h0h1h2» algorithm to make this comparison for its good performance. We achieved seven of the twelve performances, with a better performance on average and the lowest variance of the results.

Table 4: Comparison with State-of-the-Art Results in PSNR (Db)

N° Image	M1	M2	M3	M4	3x3_h0h1h2
1	37.67	28.23	36.92	38.65	32.43
2	35.81	33.01	32.75	37.11	38.37
3	37.43	34.16	36.42	38.18	37.79
4	32.74	31.81	32.47	34.22	36.56
5	39.23	35.80	38.06	39.79	38.62
6	36.18	33.01	33.70	36.72	37.69
7	39.87	37.02	37.20	40.43	40.26
8	37.87	34.12	34.35	38.40	38.43
9	36.74	34.70	36.27	38.61	38.77
10	35.54	33.22	33.50	36.52	35.13
11	36.18	33.01	33.70	36.72	38.77
12	32.00	30.84	30.79	32.98	39.74
Mean $\pm$ Std	36.44 $\pm$2.31	33.24$\pm$ 2.29	34.68$\pm$ 2.25	37.36 $\pm$ 2.15	37.71 $\pm$ 2.14

4. Conclusion

In this paper, a new Demosaicing algorithm base on two-dimensional Discrete Volterra equation through 2D-FIR filters obtained from perceptron training has been presented. An observation has been done in the state-of-the-art. Demosaicing is a problem of aliasing between

the components of luminance and chrominance in CFA image. This problem being nonlinear, its resolution requires nonlinear operations. We observed that the discrete Volterra equation in two dimensions of order greater than or equal to two has a nonlinear structure. Our task was to find the appropriate coefficients of 2D-FIR filters. Having observed the similarity between these coefficients of the kernel and the parameters of the perceptron, we obtained them after training the perceptron. Thus, this work allows, among other things, a resolution of two-dimensional Discrete Volterra equation through the training of the perceptron. The results showed us that we can tackle aliasing effect by choosing as best as possible the order of Volterra equation and the it kernel size's. Experimental data shows how well the algorithm performs.

References

[1] Alleysson, David, Sabine Susstrunk, and Jeanny Hérault. "Linear demosaicing inspired by the human visual system." IEEE Transactions on Image Processing 14, no. 4 (2005): 439-449. https://doi.org/10.1109/TIP.2004.841200.

[2] Dubois, Eric. "Frequency-domain methods for demosaicking of Bayer-sampled color images." IEEE Signal Processing Letters 12, no. 12 (2005): 847-850. https://doi.org/10.1109/LSP.2005.859503.

[3] McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." The bulletin of mathematical biophysics 5 (1943): 115-133. https://doi.org/10.1007/BF02478259

[4] Amari, Shun-ichi. "Backpropagation and stochastic gradient descent method." Neurocomputing 5, no. 4-5 (1993): 185-196. https://doi.org/10.1016/0925-2312(93)90006-O

[5] Marmarelis, Vasilis Z., and Xiao Zhao. "Volterra models and three-layer perceptrons." IEEE Transactions on neural networks 8, no. 6 (1997): 1421-1433. https://doi.org/10.1109/72.641465.

[6] Kiku, Daisuke, Yusuke Monno, Masayuki Tanaka, and Masatoshi Okutomi. "Beyond color difference: Residual interpolation for color image demosaicking." IEEE Transactions on Image Processing 25, no. 3 (2016): 1288-1300. https://doi.org/10.1109/TIP.2016.2518082.

[7] Oh, Paul, Sukho Lee, and Moon Gi Kang. "Colorization-based RGB-white color interpolation using color filter array with randomly sampled pattern." Sensors 17, no. 7 (2017): 1523. https://doi.org/10.3390/s17071523.

[8] B. E. Bayer, "colour imaging array," U.S. Patent 3 971 065, July 1976.

[9] Kodak Lossless True Colour Image Suite, (Online). Available from: http://r0k.us/graphics/kodak/〉 (accessed 01.07.2024).

[10] Libouga, Ideal Oscar, Laurent Bitjoka, David Libouga Li Gwet, Ousman Boukar, and Alexandre Michel Njan Nlôga. "A supervised U-Net based color image semantic segmentation for detection & classification of human intestinal parasites." e-Prime-Advances in Electrical Engineering, Electronics and Energy 2 (2022): 100069. https://doi.org/10.1016/j.prime.2022.100069.

[11] Elsayed, Mahmoud, Fawaz Sammani, Abdelsalam Hamdi, Asem Albaser, and Haitham Babalghoom. "A new method for full reference image blur measure." Int J Simul Syst Sci Technol 19, no. 4 (2018). https://doi.org/10.5013/IJSSST.a.19.01.7.

[12] Yang, Yanqin, Olivier Losson, and Luc Duvieubourg. "Quality evaluation of color demosaicing according to image resolution." In 2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, pp. 689-695. IEEE, 2007. https://doi.org/10.1109/SITIS.2007.33.

[13] Lu, Wenmiao, and Yap-Peng Tan. "Color filter array demosaicking: new method and performance measures." IEEE transactions on image processing 12, no. 10 (2003): 1194-1210. https://doi.org/10.1109/TIP.2003.816004

[14] Kim, Hansol, Sukho Lee, and Moon Gi Kang. "Demosaicing of RGBW color filter array based on rank minimization with colorization constraint." Sensors 20, no. 16 (2020): 4458. https://doi.org/10.3390/s20164458

[15] Jeong, Kyeonghoon, Jonghyun Kim, and Moon Gi Kang. "Color demosaicing of RGBW color filter array based on laplacian pyramid." Sensors 22, no. 8 (2022): 2981. https://doi.org/10.3390/s22082981.