THE IMPULSE RESPONSES OF BLOCK SHIFT-INVARIANT SYSTEMS AND THEIR USE FOR DEMOSAICING ALGORITHMS

Yacov Hel-Or

School of Computer Science
The interdisciplinary Center, Herzliya, Israel

ABSTRACT

Shift-invariant linear algorithms can be described completely by the algorithm's response to an impulse input. The so called *impulse response* can be used as filter kernels when the algorithm is equivalently implemented using convolution. This, however, is not true when the system is block-shift invariant, i.e. when invariance is only at repetitive locations. This paper describes a generalization of the impulse response for block shift-invariant systems. The proposed technique, takes any computer program which implements a linear block-shift-invariant algorithm, and produces its equivalent filter kernels. These kernels can then be applied efficiently by using convolution.

This scheme can assist in finding the filter kernels of linear operations applied directly to mosaic images acquired by digital cameras. For example, using this approach any algorithmic description for the demosaicing problem, which is linear and block shift-invariant, can be translated into actual filter kernels that can be applied efficiently.

1. INTRODUCTION

A color image is typically represented by three bands each of which is a matrix of values representing the responses of an array of photo-sensors to the viewed scene. The three bands are often referred to as red (R), green (G), and blue (B) according to their spectral sensitivity. Thus, three numbers are given at each matrix location composing a *pixel* value. Most CCD cameras, however, provide only a single value for each pixel due to their inability to position three photo-sensors at the same location. In these cases, the captured image is composed of a *mosaic* of color values which is the partially sampled color image. The Bayer color filter array (CFA), given in Figure 1-left, is an example of a CCD array with a typical photo-sensor arrangement.

The *demosicing* problem deals with the reconstruction of a color image I from a partial sampling $D = S(I)$ of its pixel values. Here, S represents the sampling operator applied to the image I. This reconstruction problem is, of course, an under-determined system since the solution space includes infinite many images satisfying D. Several approaches were introduced to solve the demosaicing problem (see e.g. [1, 2, 3, 4, 5]). These methods assume some prior knowledge about the probability of the input images $P(I)$. The solution is then chosen such that it maximizes the *a posteriori* probability $P(I|D)$.

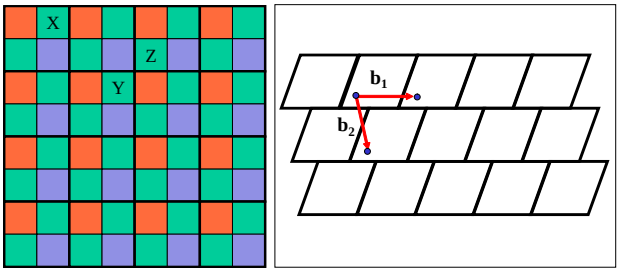

Fig. 1. Left: Typical arrangement of photo-sensors in CCD arrays. Right: A signal domain is tiled by a collection of similarly shaped blocks. Two positions with similar impulse response are far apart by integer shifts of b_1 and b_2.

Regardless of the probability model chosen to describe the input space, some solutions are implemented using a finite set of linear operations (e.g.[5, 4]). These operations are implemented using an iterative or closed form scheme. An Example of such an algorithms might be:

1. Interpolate each missing data using a bilinear interpolation of same color neighboring values.
2. Transform RGB values to chrominance/luminance values, using e.g. the RGB to YIQ linear transformation.
3. Spatially smooth the chrominance bands (bands I and Q) using Gaussian filtering.
4. Transform back from YIQ to RGB representation using a linear transformation.
5. Reset sampled values to their original values.
6. Iterate again from step 2 several times.
7. Sharpen the result using linear unsharp masking.

This work has been conducted at HP Labs- Israel.

The algorithm described above indeed gives very good results after few iterations (typically 3-5 iterations). This algorithm is well understood, and it is efficiently described in an algorithmic manner [5]. Moreover, since each step of this algorithm is a linear operation, the entire demosaicing process is linear. However, the algorithm cannot be implemented using a single filtering with a filter kernel since it is not shift-invariant in the classical manner. For example, the algorithm response to an impulse input at location X in Figure 1-left, is not identical to its response to an impulse input at location Z. Nevertheless, the above algorithm is shift-invariant if the shifted steps are of even numbers. Hence, the algorithm has identical responses to impulse inputs at locations X and Y. We call such an algorithm a *block-shift invariant* (BSI).

This paper shows that BSI algorithms can be implemented efficiently using a fixed number filter kernels. Moreover, a technique is suggested here to extract automatically such kernels given a BSI algorithm. The suggested approach is general, and can be applicable for any such algorithm. The proposed procedure does not require any elaborated knowledge or description of the algorithm. The algorithm can be regarded as a black box where the input and the associated outputs are available. There are only two requirements that must be satisfied:

1. The algorithm process must be linear, i.e., if $A\{s\}$ represents the results of applying the algorithm A to the input signal $\mathbf{s}$ then:

   $$A\{\mathbf{s}_1+\mathbf{s}_2\} = A\{\mathbf{s}_1\}+A\{\mathbf{s}_2\} \text{ and } A\{\lambda\mathbf{s}\} = \lambda A\{\mathbf{s}\}$$

2. The algorithm is block-shift-invariant. For 1D signals, a block-shift-invariant system with a block shift b satisfies:

   $$A\{\mathbf{s}(x-bn)\}(y) = A\{s(x)\}(y-bn)$$

   where n is an integer number. Similarly, a block-shift-invariant system for signals in k-D satisfies:

   $$A\{\mathbf{s}(\mathbf{x}-B\mathbf{n})\}(\mathbf{y}) = A\{s(\mathbf{x})\}(\mathbf{y}-B\mathbf{n}) \qquad (1)$$

   where $\mathbf{x}$ and $\mathbf{y}$ are k-D vectors, B is a $k \times k$ matrix whose columns define basic block shifts (Figure 1-right), and $\mathbf{n}$ is a k-D vector of integers.

The Bayer pattern described above has a repetitive pattern of 2×2 blocks (Figure 1-left). Thus, the demosaicing algorithm described above is shift-invariant, only if the shifts are integer numbers of blocks, i.e. Equation 1 is satisfied where B is a diagonal 2×2 matrix with 2 in its diagonal.

1.1. Determining Block-Shift-Invariant Kernels

In principle, a shift-invariant linear process can be implemented using an appropriate linear filter, i.e. a convolution with a filter kernel. In the case of a block-shift invariant linear system, as in the case of the proposed demosaicing process, a linear filtering system can be used as well, however the filtering is based on a set of filter kernels rather than a single one, each of which is associated with a particular location in the block. Thus, in the Bayer pattern depicted in Figure 1, 12 kernels are defined; for 3 different colors and 4 different locations. In this section a technique is described for automatically determining the appropriate set of kernels. For clarity, the classical shift-invariant case is first described, and then extended to describe the block-shift-invariant case.

To simplify notations, throughout the paper, index variables will be written as scalar variables, although they can represent k-D vectors, if the signal domain is k-dimensional.

1.2. Shift-Invariant Case

The appropriate kernel for a given shift-invariant linear algorithm can be extracted by applying the algorithm to an impulse input as in Figure 2a . This is known as the *impulse response* of the algorithm. Due to shift-invariance of the algorithm, the response for any shifted impulse is the impulse response shifted by the the same amount (Figure 2b). To compute the kernel values, a single output pixel position is considered (Figure 2c). At that particular pixel location, the collection of impulse responses is determined for all possible impulse inputs (Figure 2d).

Formally speaking, let $\delta(x-p)$ be an impulse input, positioned at location p, and the algorithm response for such an input be $R_p(y)$, i.e.

$$A\left\{\delta(x-p)\right\}(y) = R_p(y)$$

Due to the shift-invariance of the algorithm we have:

$$R_p(p+y) = R_q(q+y) \doteq \hat{R}(y) \quad \forall\, p,q$$

Similarly, the algorithm response to any shifted delta function $\delta(x-k)$ is:

$$A\{\delta(x-k)\}(y) = R_k((y-k)+k) = \hat{R}(y-k)$$

Using this property, the algorithm output $s_{out}(y)$ for a given signal input $s_{in}(x)$ can be calculated:

$$\begin{aligned} s_{out}(y) &= A\left\{s_{in}(x)\right\}(y) = \qquad (2)\\ &= A\left\{\sum_k s_{in}(y-k)\delta(x-(y-k))\right\}(y) = \\ &= \sum_k s_{in}(y-k)A\left\{\delta(x-(y-k))\right\}(y) = \\ &= \sum_k s_{in}(y-k)\hat{R}(k) \end{aligned}$$

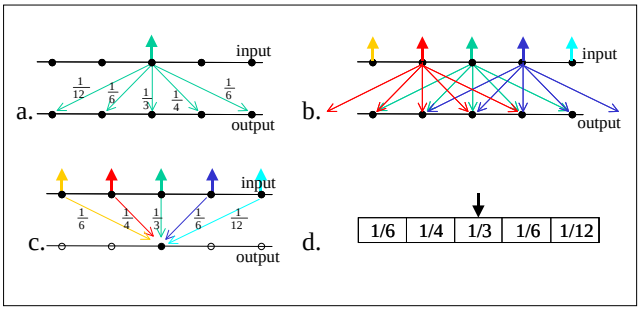

Fig. 2. Finding the appropriate filer kernel for a shift-invariant linear system (see text).

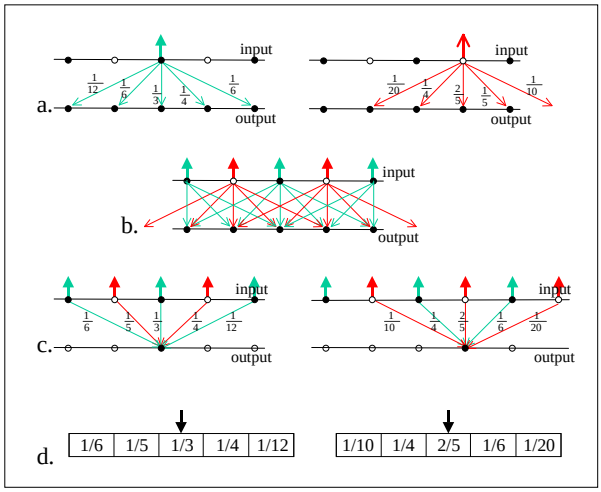

Fig. 3. Finding the appropriate filter kernels for a block-shift-invariant linear system (see text).

where the third equality is due to the linearity of the algorithm. This result implies that the impulse response of a shift-invariant linear algorithm completely characterizes the algorithm; i.e. the output of the algorithm for any input signal can be determined. This can be viewed as a kernel convolution where the convolution kernel is $\hat{R}(y)$. The question explored in the following section, is whether this property is valid also for block-shift-invariant algorithm, and if so, what are the kernels that should be used in such systems.

1.3. Block-Shift-Invariant Case

In a block-shift-invariant linear system, the entire signal domain is tiled by a collection of similar block shapes (e.g. see Figure 1-right). A canonical block consists of N indexed grid positions $p_1 \cdots p_N$. Define the function $g(p)$ that maps a general grid position p to its position index, i.e. $g(p) \in \{1, \cdots, N\}$. In Figure 1-right, for example, $g(p) = g(p+m\mathbf{b}_1+n\mathbf{b}_2)$ for any integers m and n. Applying the algorithm to an impulse input at position p, results in the output response:

$$A\{\delta(x-p)\}(y) = R_p(y)$$

However, due to the block-shift invariance of the algorithm:

$$R_p(y+p) = R_q(y+q) \doteq \hat{R}_{g(p)}(y) \quad \text{iff } g(p) = g(q)$$

Therefore, we have a set of N different impulse responses: $\hat{R}_i(y)$, $i = 1 \cdots N$, where

$$\hat{R}_i(y) \doteq R_{p_i}(y + p_i)$$

Given this set of N impulse responses, the algorithm output to *any* shifted impulse can be constructed:

$$A\{\delta(x-k)\}(y) = \hat{R}_{g(k)}(y-k)$$

Similar to the classical shift-invariant system, the algorithm output $s_{out}(y)$ for a given signal input $s_{in}(x)$ can be calculated from the set of $\hat{R}_i(y)$:

$$\begin{aligned} s_{out}(y) &= A\{s_{in}(x)\}(y) = \\ &= A\left\{\sum_k s_{in}(y-k)\delta(x-(y-k))\right\}(y) = \\ &= \sum_k s_{in}(y-k)A\{\delta(x-(y-k))\}(y) = \\ &= \sum_k s_{in}(y-k)\hat{R}_{g(y-k)}(k) \end{aligned}$$

The last equation states that, similar to the classical shift-invariant systems, the set of impulse-responses of the block-shift-invariant algorithm $\hat{R}_i(k)$ entirely characterizes the algorithm, i.e. the output of the algorithm for any given input signal can be calculated. However, due to block-shift-invariance, the algorithm output is calculated similarly only for those positions y having the same position index $g(y)$. Therefore, there are N different convolution kernels $H_i(k)$, $i = 1..N$, where:

$$H_{g(y)}(k) \doteq \hat{R}_{g(y-k)}(k)$$

To calculate the algorithm output at position y, the kernel $H_{g(y)}(k)$ is applied in the following manner:

$$s_{out}(y) = \sum_k s_{in}(y-k)H_{g(y)}(k)$$

As explained above, the number of kernels required is equal to the size of a block. Figure 3 shows a 1D example where the block size is two. In this case, 2 filters are required to implement the linear system. The filters can be found by applying the algorithm to two impulse inputs as in Figure 3a. To compute the filter kernel values at a single output pixel position, the collection of system outputs at that pixel location is determined for all possible impulse inputs (Figures 3b and 3c).

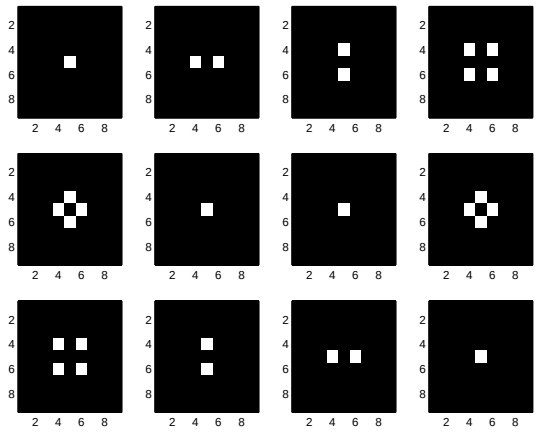

Fig. 4. Kernels calculated for a bilinear interpolation algorithm. Each row displays 4 kernels for block positions (left to right) $(1,1),(1,2),(2,1)$, and $(2,2)$. Rows (top to bottom) are for the Red, Green, and Blue images. For better visualization, gray values were normalized for each kernel.

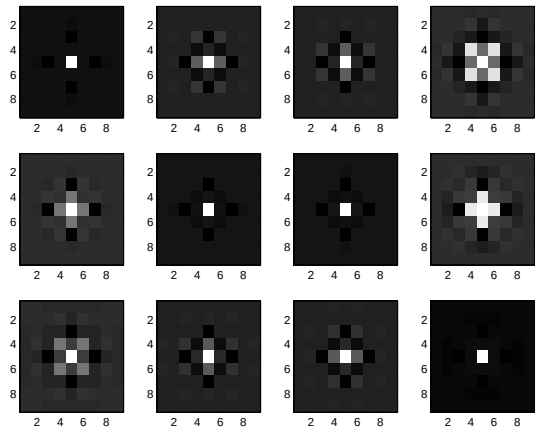

Fig. 5. The kernels that were calculated for the demosaicing algorithm proposed in Section 1. The kernels are ordered as in Figure 4

1.4. Block Shift-invariant Kernels for Demosaicing

As described above, the suggested Demosaicing algorithm in Section 1 is a BSI linear system applied on a 2D Bayer Pattern which is 2×2 repetitive. The input is a single 2D array of sensor inputs. The output of the process consists of three 2D arrays representing the Red, Green and Blue contents of a color image. Thus, in practice, the Demosaicing process can be viewed as 3 independent BSI linear systems (receiving the same input and producing the R, G or B arrays). Following the discussion above, the Demosaicing process can be implemented using 12 filters (3 filters for the R, G and B outputs for each of the 2×2 block positions.

2. DEMONSTRATED RESULTS

In order to demonstrate the proposed technique, two demosaicing algorithms were considered. The first algorithm, is the trivial bilinear interpolation for the Bayer pattern. The 12 kernels that were generated by the proposed technique, are shown in Figure 4. In the figure each row has 4 kernels, for block positions $(1,1),(1,2),(2,1)$, and $(2,2)$, from left to right. The rows show the kernels for the Red, Green, and Blue images, from top to bottom. Note, that one of the kernels in each column is the delta kernel. This corresponds to the associated color sample at this block position. The second algorithm considered, is that given in Section 1. The kernels that were generated for this algorithm are shown in Figure 5, in the same order. These kernels were applied to mosaiced images. Due to inferior visibility quality of printed reproductions demosaicing examples applying the above kernels are available online in:
http://www.faculty.idc.ac.il/toky/Pub/bsiRes/results.htm.

3. CONCLUSIONS

A technique for generating the appropriate kernels for block-shift-invariant algorithms was proposed. These kernels can be calculated without specific knowledge about the algorithm, i.e the algorithm can be regarded as a black-box. The advantage of the proposed technique is that the kernel derivation is automatic and does not require complicated mathematical derivations. A hardware device that can apply convolution kernels to an image, can be adjusted efficiently to perform any proposed BSI linear algorithm.

Acknowledgements

The author would like to thank Renato Keshet for his assistance in writing this paper and for helpful discussions.

4. REFERENCES

[1] R. Kimmel, "Demosaicing: Image reconstruction from color CCD samples," in *EVVC (1)*, 1998, pp. 610–622.

[2] D. R. Cok, "Reconstruction of CCD images using template matching," in *Proc. IS&T's*, 1994.

[3] Y. Hel-Or and D. Keren, "Demosaicing of color images using steerable wavelets," Tech. Rep. HPL-97-104, HP Labs, August 1997.

[4] D. Taubman, "Generalized wiener reconstruction of images from colour sensor data using a scale invariant prior," in *ICIP*, 2000.

[5] Y. Hel-Or, "The canonical correlations of color images and their use for demosaicing," Tech. Rep. HPL-2003-164R1, HP Labs, Feb 2004.