Fast Space-Varying Convolution Using Matrix Source Coding

Jianing Wei, *Member, IEEE,* Charles A. Bouman, *Fellow, IEEE,*
and Jan P. Allebach, *Fellow, IEEE*

***Abstract*—Many imaging applications require the implementation of space-varying convolution for accurate restoration and reconstruction of images. Moreover, these space-varying convolution operators are often dense, so direct implementation of the convolution operator is typically computationally impractical. One such example is the problem of stray light reduction in digital cameras, which requires the implementation of a dense space-varying deconvolution operator. However, other inverse problems, such as iterative tomographic reconstruction, can also depend on the implementation of dense space-varying convolution. While space-invariant convolution can be efficiently implemented with the Fast Fourier Transform (FFT), this approach does not work for space-varying operators. So direct convolution is often the only option for implementing space-varying convolution.**

In this paper, we develop a general approach to the efficient implementation of space-varying convolution, and demonstrate its use in the application of stray light reduction. Our approach, which we call matrix source coding, is based on lossy source coding of the dense space-varying convolution matrix. Importantly, by coding the transformation matrix, we not only reduce the memory required to store it; we also dramatically reduce the computation required to implement matrix-vector products. Our algorithm is able to reduce computation by approximately factoring the dense space-varying convolution operator into a product of sparse transforms. Experimental results show that our method can dramatically reduce the computation required for stray light reduction while maintaining high accuracy.

***Index Terms*—Stray light, space-varying point spread function, image restoration, fast algorithm, inverse problem, digital photography.**

I. INTRODUCTION

In many important imaging applications, it is useful to model the acquired data vector, y, as

$$y = Ax + w, \tag{1}$$

where x is the unknown image to be determined, A is a linear transformation matrix, and w is some additive noise that is independent of x with inverse covariance Λ_w. This simple linear model can capture many important inverse problems including image deblurring [1], tomographic reconstruction [2], and super-resolution [3].

This research work was done when Jianing Wei was a Ph.D. student at the Department of Electrical and Computer Engineering, Purdue University. Jianing Wei is with US Research Center, Sony Electronics Inc., San Jose, CA 95112, USA. Charles A. Bouman and Jan P. Allebach are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, USA. Email: weijianing@gmail.com, {bouman, allebach}@purdue.edu.

This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number 56541-CI, and the National Science Foundation under Contract CCR-0431024.

There is a vast literature related to the effective solution of inverse problems with the form of Eq. (1). Solution to these inverse problems can be very difficult if the transformations A or $A^t\Lambda_w A$ are ill conditioned. This can and often does happen if the data is sparse or of low quality. In these cases, many algorithms have been proposed to recover x from y. Classical approaches include algebraic reconstruction techniques (ART) [4]; simultaneous iterative reconstruction technique (SIRT) [5]; simultaneous algebraic reconstruction technique (SART) [6]; Lucy-Richardson algorithm [7]; Van Cittert's method [1], [8]; approximate [9] or true [10] maximum likelihood estimation; regularized inversion [11]; and maximum a posteriori (MAP) inversion [12], [13], [14]. A common objective in all these methods is to balance the goal of matching the observed (but noisy) data with the goal of producing a physically realistic image.

While many methods exist for addressing the inverse problem of Eq. (1), all these methods typically require the computation of matrix-vector products, such as Ax, or A^ty. In fact, iterative inversion methods typically require the repeated computation of the matrix-vector product A^tAx or $A^t\Lambda_w Ax$. If the matrix A is sparse, as is the case with a differential operator [15], or if it can be decomposed as a product of sparse transformations, as is the case with the wavelet [16] or fast Fourier transform (FFT) [17], then computation of Ax is fast and tractable. For example, if Ax is space-invariant convolution, then A can be decomposed as the product of two FFTs and a diagonal transform, resulting in a total computation for the evaluation of Ax that has an order of $O(P \log P)$ where P is the size of x. However, if A represents space-varying convolution with a large point-spread function, then each direct evaluation of the matrix-vector product Ax or A^ty can require enormous computation. Even when A^tA is approximately Toeplitz, as is often the case in tomographic reconstruction [4], the matrix $A^t\Lambda_w A$ will not be Toeplitz when the noise is space varying.

The specific application considered in this paper is that of stray light reduction for digital photographs [18]. Stray light refers to the portion of the entering light flux that is misdirected to undesired locations in the image plane. It is sometimes referred to as lens flare or veiling glare [19]. It can be caused by imperfections in lens elements, or reflections between optical surfaces; but in the end, it tends to reduce the contrast of images by scattering light from bright regions of the image into dark regions. Since stray light is typically scattered across the imaging field, its associated point spread function (PSF) typically has large support and is space-varying

[18]. With this in mind, an imaging system with stray light can be modeled using Eq. (1) with

$$A = (1 - \beta)I + \beta S, \tag{2}$$

where I is the identity matrix, S is a dense matrix describing the space-varying scattering of light across the imaging array[1], and β is a scalar constant representing the fraction of scattered light.

Typically the fraction of scattered light is small, and we know that the matrix A is very well conditioned, so in contrast to many inverse problems, stray light reduction is a well-posed inverse problem, and the solution can be computed using a formula such as that proposed by Jansson and Fralinger [20]

$$\hat{x} = (1 + \beta)y - \beta Sy, \tag{3}$$

where y is the observed image, and $\hat{x}$ is the estimate of the underlying image to be recovered.

However, even in this simple form, the computation of Eq. (3) is not practical for a modern digital camera, because the matrix S is both dense and lacks a Toeplitz structure. In typical stray light models, the PSF is space-varying with heavy tails, so the FFT cannot be directly used to speed up computation [18]. For example, if the image contains P pixels, the computation is $O(P^2)$; so for a 10^6 pixel image, it takes 10^6 multiplies to compute each output pixel, resulting in a total of 10^{12} multiplies. This much computation requires hours of processing time, which makes accurate stray light reduction infeasible for implementation in low cost digital cameras.

In this paper, we propose a novel algorithm for fast computation of space-varying convolution, and we demonstrate its effectiveness for the particular application of stray light reduction. Our method, which we refer to as matrix source coding (MSC), is based on the use of lossy source coding techniques to compress the dense matrix S into a sparse form. Importantly, the effect of this compression is not only to reduce the memory required for storing S, but also to dramatically reduce the number of multiplies required to approximately compute the matrix-vector product Sy. Our approach works by first decorrelating the rows and columns of S and then quantizing the resulting compacted matrix so that most of its entries become zero.

Our MSC method requires both on-line and off-line components. While the on-line component is fast, it requires the pre-computation of a sparse matrix decomposition in an off-line procedure. We introduce a method for efficiently implementing this off-line computation for a broad class of space-varying kernels whose wavelet coefficients are localized in space.

In order to assess the value of our approach, we apply it to the problem of stray light reduction for digital cameras. Using the stray light model of [18], we demonstrate that the MSC method can achieve an accuracy of 1% with approximately 7 multiplies per pixel at image resolution 1024×1024, a dramatic reduction when compared to direct space-varying convolution. We also demonstrate a practical pre-computaion algorithm that can be performed in time proportion to the kernel size.

In Section II we introduce the MSC method. The on-line algorithm for space-varying convolution is derived and presented in Section II-A, and the off-line pre-computation method is presented in Section II-B and Section IV. Section III presents the stray light application and model, and Section V follows with experimental results. Section VI concludes this paper.

II. Matrix Source Coding Approach to Space-Varying Convolution

In this section, we present the matrix source coding (MSC) approach for fast space-varying convolution. The MSC approach has two parts, which we will refer to as the on-line and off-line computations. The on-line algorithm computes the space-varying convolution using a product of sparse matrices. Section II-A presents the on-line algorithm along with its associated theory.

While the on-line computation is very fast, it depends on the result of off-line computation, which is to pre-compute a sparse version of the original space-varying convolution matrix. This computation is not input image-dependent, and only depends on the PSF of the imaging system. However, naive computation of this sparse matrix can be enormously computationally demanding. So in Section II-B, we introduce an efficient algorithm for accurate but approximate computation of the required sparse transformation for a broad class of problems.

A. Matrix Source Coding Theory

Our goal is to speed up the computation of general matrix-vector products with the form

$$z = Sy,$$

where y is the input, z is the output, and S is a dense $P \times P$ matrix without a Toeplitz structure.[2] If S is Toeplitz, then Sy can be efficiently computed using the fast Fourier transform (FFT) [21]. However, when S implements space-varying convolution, then it is not Toeplitz; and the FFT cannot be used directly.

Our strategy for speeding up computation of Sy is to use lossy transform coding of the rows and columns of S. The resulting coded matrix then becomes sparse after quantization, and this sparsity reduces both storage and computation. However, in order to perform lossy source coding of S, we must first determine the relevant distortion metric.

In general, the mean squared error (MSE) is not a good distortion metric for coding S because it does not account for any correlation in the vector y. In order to see this, let $[S]$ be a quantized version of S produced by lossy coding and decoding. Then the distortion introduced into the output z is given by

$$\delta z = S\,y - [S]\,y = \delta S\,y,$$

[1]Due to conservation of energy, we know that the columns of S sum to less than 1.

[2]In fact, all our results hold when S is a non-square $P_1 \times P_2$ matrix. However, we consider the special case of $P = P_1 = P_2$ here for notational simplicity. The extension to the more general case is direct.

where $\delta S = S - [S]$ is the distortion introduced into S through lossy coding and δz is the resulting distortion introduced into z. Using the argument of [22], [23] and assuming that y is independent of δS, then the expected MSE in δz is given by

$$\begin{aligned} \mathrm{E}\left[\|\delta z\|^2 | \delta S\right] &= \mathrm{E}\left[\mathrm{trace}\{\delta z \delta z^t\} | \delta S\right] \\ &= \mathrm{E}\left[\mathrm{trace}\{\delta S y y^t \delta S^t\} | \delta S\right] \\ &= \mathrm{trace}\{\delta S \, \mathrm{E}\left[y y^t | \delta S\right] \delta S^t\} \\ &= \mathrm{trace}\{\delta S \, R_y \, \delta S^t\}, \end{aligned}$$

where $R_y = \mathrm{E}[yy^t] = \mathrm{E}\left[yy^t | \delta S\right]$. Notice that in the special case when y is white, then we have that $R_y = I$, and

$$\mathrm{E}\left[\|\delta z\|^2 | \delta S\right] = \|\delta S\|^2 \ . \tag{4}$$

In other words, when the input y is white, minimizing the squared error distortion of the matrix S (i.e. the Frobenius norm) is equivalent to minimizing the expected value of the squared error distortion for z. However, when the components of y are correlated, then minimum MSE coding of S can be quite far from minimizing the MSE in z.

Based on this result, our strategy for source-coding of S will be to first transform the matrix S so its input is white, and its rows and columns are decorrelated; and then to quantize the transformed matrix so it becomes sparse. More specifically, we define the transformed matrix $\tilde{S}$ as

$$\tilde{S} = W_1 S T^{-1}, \tag{5}$$

where W_1 is an orthonormal transform[3] and T is an invertible transform. Then the output z can be computed as

$$z = W_1^{-1} \tilde{S} T y \ . \tag{6}$$

Our objective is then to select the matrices T and W_1 so that T whitens (i.e. spheres) the covariance of y, and together, T and W_1 deorrelate the rows and columns of S. Since the covariance of Ty is whitened and the rows and columns of S are decorrelated, we can apply MMSE quantization of $\tilde{S}$ in order to achieve MMSE estimation of z. So if $[\tilde{S}]$ denotes the quantized version of $\tilde{S}$, then we have that

$$\hat{z} = W_1^{-1} [\tilde{S}] T y \ , \tag{7}$$

where $\hat{z}$ is the approximation of z.

Given that the rows and columns of $\tilde{S}$ are approximately decorrelated, then we should expect that the quantized matrix, $[\tilde{S}]$, will be sparse, and that therefore, multiplying with $[\tilde{S}]$ will be computationally efficient and, just as importantly, $[\tilde{S}]$ will be easy to store. However, in practice, if the transforms T and W_1 are dense, then the application of these decorrelating transforms might require as much computation as the original multiplication by S. This would, of course, defeat the purpose of the sparsification of S.

In fact, the "optimal" decorrelating transforms, T and W_1, have exactly this property. Appendix A constructs transforms T and W_1 that exactly achieve the goal of whitening (i.e. sphereing) y and decorrelating the rows and columns (i.e. diagonalizing) of S. This can be done by selecting T as the generalized eigendecomposition of the matrix pair $(R_y, R_c \triangleq S^t S)$ [21] and by selecting W_1 as the eigendecomposition for the covariance matrix $R_r \triangleq S S^t$. However, while this "optimal" choice produces perfect decorrelation and sparsity, it is not practical because the transforms T and W_1 are then, in general, dense transformations with no fast implementation.

[3]Without loss of generality, these transforms can be orthogonal, but for notional simplicity we also assume they are normal.

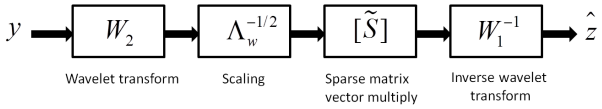


Fig. 1. Block diagram of on-line computation of matrix source coding.

So our objective is to find computationally efficient transformations for T and W_1 which approximately whiten y and decorrelate the rows and columns of S. In order to achieve this objective, we will use wavelet transforms to decorrelate both the vector y, and the rows and columns of S. The wavelet transform is a reasonable choice because it is known to be an approximation to the Karhunen-Loeve transform for stationary sources [24], and it is commonly used as a decorrelating transform for various source coding applications [25]. Using this strategy, we form the matrix T by a wavelet transform followed by gain factors designed to normalize the variance of wavelet coefficients. This combination of decorrelation and scaling whitens or spheres the vector y. Specifically, we choose

$$T = \Lambda_w^{-1/2} W_2, \tag{8}$$

where W_2 is a wavelet transform and $\Lambda_w^{-1/2}$ is a diagonal matrix of gain factors so that

$$\Lambda_w = \mathrm{diag}\left(W_2 R_y W_2^t\right) \ , \tag{9}$$

where $R_y = \mathrm{E}[yy^t]$ is the covariance of y. At the same time, notice that the transform $T^{-1} = W_2^{-1} \Lambda_w^{1/2}$ approximately decorrelates the columns of S.[4] Similarly, if we choose the transform W_1 to be a wavelet transform, then it will also decorrelate the rows of S, so that $[\tilde{S}]$ will be sparse after quantization.

To summarize, Fig. 1 illustrates a block diagram of the on-line computation required for matrix source coding of Eq. (7). First, the input data y is transformed and scaled. Then it is multiplied by a sparse matrix $[\tilde{S}]$. Then a second inverse wavelet transform is applied to compute the approximate result $\hat{z}$. The accuracy of the approximation is determined by the degree of quantization applied to $\tilde{S}$. So if little or no quantization is used, then the computation can be arbitrarily close to exact, but at the cost of less sparsity in $\tilde{S}$ and therefore more computation and storage. In this way, matrix source coding allows for a continuous tradeoff between accuracy and computation/storage.

Finally, as a practical matter, the diagonal matrix Λ_w can be easily estimated from training images. In fact, we will assume that the gain factors for each subband are constant since the wavelet coefficients in the same subband typically have the same variance [1]. So to estimate Λ_w, we take the wavelet

[4]This is true because if the S is slowly space-varying operator, then the covariance matrix $R_r = S^t S$ is approximately Toeplitz; and the wavelet transform approximates the Karhunen-Loeve transform.

transform W_2 of the training images, compute the variance of the wavelet coefficients in each subband, and average over all images to obtain an estimate of the gain factors for each subband.

B. Efficient Off-Line Computation for Matrix Source Coding

In order to implement the fast space-varying convolution method described by Eq. (7), it is first necessary to compute the source coded matrix $[\tilde{S}]$. We will refer to the computation of $[\tilde{S}]$ as the off-line portion of the computation. Once the sparse matrix $[\tilde{S}]$ is available, then the space-varying convolution may be efficiently computed using on-line computation shown in Eq. (7).

However, even though the computation of $[\tilde{S}]$ is performed off-line, exact evaluation of this sparse matrix is still too large for practical problems. For example, if the image contains 16 million pixels, then temporary storage of S requires approximately 10^6 Giga bytes of memory, exceeding the capacity of modern computers. The following section describes an efficient approach to compute $[\tilde{S}]$ which eliminates the need for any such temporary storage.

In our implementation, we use the Haar wavelet transform [16] for both W_1 and W_2. The Haar wavelet transform is an orthonormal transform, so we have that

$$\begin{aligned} \tilde{S} &= W_1 S T^{-1} \\ &= W_1 S W_2^t \Lambda_w^{1/2}. \end{aligned}$$

Therefore, direct implementation of this off-line computation consists of a wavelet transform W_2 along the rows of S, scaling of the matrix entries, and a wavelet transform W_1 along the columns. Unfortunately, the computational complexity of these two operations is $O(P^2)$, where P is the number of pixels in the image, because the operation requires that each entry of S be touched. This computation can take a tremendous amount of time for high resolution images. Therefore, we aim at reducing the computational complexity to be linearly proportional to P and eliminating the need to store the entire matrix S. In order to achieve this goal, we will use a two stage quantization procedure combined with a recursive top-down algorithm for computing the Haar wavelet transform coefficients.

The first stage of this procedure is to zero out the entries in $SW_2^t\Lambda_w^{1/2}$ whose magnitudes are less than a specified threshold. We use $Q_t(SW_2^t\Lambda_w^{1/2})$ to describe this thresholding stage, where for a scalar x

$$Q_t(x) = \begin{cases} x & \text{for } |x| > t \\ 0 & \text{otherwise} \end{cases}. \tag{10}$$

The resulting matrix $Q_t(SW_2^t\Lambda_w^{1/2})$ is already quite sparse, thus there is no need to store the entire dense matrix. Then the second stage of this procedure is to compute the wavelet transform W_1 over the columns of this already sparse matrix. Taking advantage of the sparsity resulting from the first stage, if the average number of remaining nonzero entries in each row of $Q_t(SW_2^t\Lambda_w^{1/2})$ is N ($N \ll P$), then the computation of the wavelet transform in the second stage is reduced from $O(P^2)$ to $O(NP)$. In summary, this two stage procedure is expressed mathematically as

$$[\tilde{S}] \approx [W_1 Q_t(SW_2^t\Lambda_w^{1/2})]. \tag{11}$$

However, there is still a problem as to how to efficiently compute $Q_t(SW_2^t\Lambda_w^{1/2})$. Since S is dense, direct evaluation of this expression requires $O(P^2)$ operations. However, after thresholding, most values in $Q_t(SW_2^t\Lambda_w^{1/2})$ are zero, so we can dramatically reduce computation by first identifying the regions where non-zero values are likely to occur. We refer to these non-zero values as significant wavelet coefficients. As mentioned before, we use N to denote the number of significant wavelet coefficients per row. Once we can predict the location of the significant wavelet coefficients, we can use a recursive top-down approach to compute them without having to do the full wavelet transform for each row of S. We will discuss how to predict the location of the significant wavelet coefficients in constant time in Sec. IV.

Next, we describe our algorithm for computing the significant wavelet coefficients once their location is known. Our strategy is to first compute the approximation coefficients that are necessary to compute the significant wavelet coefficients using a recursive top-down approach. Then significant detail coefficients can be computed from the approximation coefficients at finer resolutions. Our approach is able to achieve $O(N)$ computational complexity for computing the significant Haar wavelet coefficients per row, which leads to a total complexity of $O(NP)$ for computing $Q_t(SW_2^t\Lambda_w^{1/2})$.

We use $f(k,i,j)$ to represent the approximation coefficient at location (i,j) at level k, where larger values of k correspond to coarser scales, and $f(0,i,j)$ corresponds to the full resolution image. We use $g_h(k,i,j)$, $g_v(k,i,j)$, and $g_d(k,i,j)$ to represent the detail coefficients in horizontal, vertical and diagonal bands, respectively. We can then compute $f(k,i,j)$ as the average of its corresponding 4 higher resolution neighbors:

$$\begin{aligned} f(k,i,j) = \frac{1}{2}(&f(k-1,2i,2j) + f(k-1,2i+1,2j) \\ &+ f(k-1,2i,2j+1) + f(k-1,2i+1,2j+1)). \end{aligned} \tag{12}$$

Equation (12) specifies the parent/child relations of a quad-tree as shown in Figure 2. Our algorithm transverses this quad-tree in a pruned depth-first search; and at each step of the search, the corresponding wavelet approximation coefficient is computed. The search is pruned whenever the wavelet detail coefficients are determined to be "insignificant". In this case, the wavelet approximation coefficient is directly approximated by the local value of the actual image using the relationship

$$f(k,i,j) \approx 2^k f(0, 2^k i, 2^k j). \tag{13}$$

When the search is not pruned, then the wavelet approximation coefficient is computed using Eq. (12), where the finer resolution coefficients are evaluated using recursion. Of course, we also terminate the search when $k = 0$ since this is the finest resolution at which the wavelet approximation coefficient can be evaluated exactly. Figure 2 provides a graphic illustration of how the search transverses from the coarsest to finest

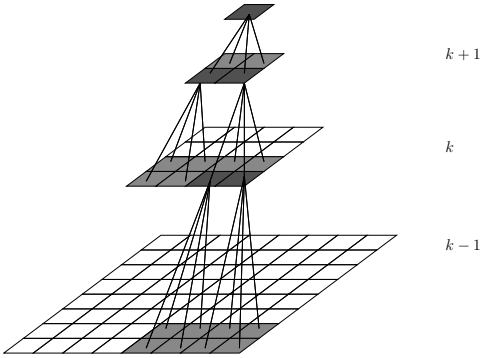


Fig. 2. An illustration of the quad-tree structure of Haar approximation coefficients. Dark gray squares are internal nodes of the pruned quad-tree. Corresponding detail coefficients are significant at these nodes, and we go on to the next level recursion. Light gray squares are leaf nodes of the pruned quad-tree. The termination condition of our recursive algorithm is met at these nodes; so the values are directly computed and returned. Empty squares are nodes pruned off from the quad-tree. Values at these nodes are not computed or touched.

resolutions, and Fig. 3 provides a more precise pseudo-code specification of the algorithm.

After obtaining the necessary approximation coefficients, we can compute the significant detail coefficients using the following equations:

$$\begin{aligned} g_h(k,i,j) &= \frac{1}{2}(f(k-1,2i,2j) - f(k-1,2i,2j+1) \\ &\quad + f(k-1,2i+1,2j) - f(k-1,2i+1,2j+1)), \\ g_v(k,i,j) &= \frac{1}{2}(f(k-1,2i,2j) + f(k-1,2i,2j+1) \\ &\quad - f(k-1,2i+1,2j) - f(k-1,2i+1,2j+1)), \\ g_d(k,i,j) &= \frac{1}{2}(f(k-1,2i,2j) - f(k-1,2i,2j+1) \\ &\quad - f(k-1,2i+1,2j) + f(k-1,2i+1,2j+1)). \end{aligned} \tag{14}$$

```
Main routine:
K ← coarsest scale;
for all (i, j)
    f(K, i, j) ← FastApproxCoef(K, i, j, f);
end
```

(a)

```
float FastApproxCoef(k, i, j, f) {
    if (k==0) {
        f(0, i, j) ← directly compute from PSF;
        return f(0, i, j);
    }
    if ( SignificantCoef(k, i, j)==1 ) {
        f(k, i, j) = FastApproxCoef(k − 1, 2i, 2j, f);
        f(k, i, j) += FastApproxCoef(k − 1, 2i, 2j + 1, f);
        f(k, i, j) += FastApproxCoef(k − 1, 2i + 1, 2j, f);
        f(k, i, j) += FastApproxCoef(k − 1, 2i + 1, 2j + 1, f);
        f(k, i, j) ← f(k, i, j)/2;
    }
    else {
        f(k, i, j) ← 2^k f(0, 2^k i, 2^k j);
    }
    return f(k, i, j);
}
```

(b)

```
int SignificantCoef(k, i, j) {
    if (any of g_h(k, i, j), g_v(k, i, j), g_d(k, i, j) is significant) {
        return 1;
    }
    else {
        return 0;
    }
}
```

(c)

Fig. 3. Pseudo code for fast computation of necessary Haar approximation coefficients. Part (a) is the main routine. Part (b) is a subroutine called by (a). Part (c) is a subroutine called by (b). The function FastApproxCoef() described in part (b) is a recursive function. The function SignificantCoef() described in part (c) determines whether any of the corresponding detail coefficients is significant. Section IV will discuss how to predict which wavelet coefficients are significant coefficients. After the main routine call, the necessary Haar approximation coefficients have been computed and stored in array f.

Notice that the complexity of our algorithm to compute $Q_t(SW_2^t\Lambda_w^{1/2})$ is $O(NP)$. To see that this is true, notice that the total complexity of the depth-first search is linearly proportional to the number of nodes in the tree, and the number of nodes in the tree is proportional to N. Moreover, the computation performed at each node has fixed computational complexity because we have assumed that the subroutine SignificantCoef of Fig. 3(c) has constant complexity. So putting this together, we know that the complexity of computing the wavelet transform for each row is $O(N)$. Thus, the total complexity of evaluating the wavelet transform for P rows must be order $O(NP)$.

The second stage of our matrix source coding is a wavelet transform W_1 along the columns of the sparse matrix $Q_t(SW_2^t\Lambda_w^{1/2})$. Storing the entire sparse matrix $Q_t(SW_2^t\Lambda_w^{1/2})$ prior to computing the wavelet transform W_1 may exceed the memory capacity of computers for high resolution images. So in practice, we use block wavelet transforms for W_1 to reduce memory usage. In this way, we can perform the transform block-by-block, and only need to keep $Q_t(SW_2^t\Lambda_w^{1/2})$ for each block prior to computing wavelet transform W_1. Once the transform W_1 for a block is complete, the values in $Q_t(SW_2^t\Lambda_w^{1/2})$ for this block may be discarded, and the process is repeated for each new block. This stage also has a total computational complexity of $O(NP)$. Thus the complete algorithm reduces the off-line computation of $[\tilde{S}]$ from $O(P^2)$ to $O(NP)$, making it feasible for implementation.

III. MODELS OF STRAY LIGHT REDUCTION

In this paper, we will use the example of stray light reduction to demonstrate the effectiveness of our matrix source coding algorithm. Stray light occurs in all optical imaging systems. It refers to the portion of the entering light flux that is misdirected to undesired locations in the image plane. It is sometimes referred to as lens flare or veiling glare [19]. Causes for this phenomena include but are not limited to: (1) Fresnel reflections from optical element surfaces; (2) scattering from surface imperfections on lens elements; (3) scattering from air bubbles in transparent glass or plastic lens elements; and (4) scattering from dust or other particles. Images contaminated by stray light suffer from reduced contrast and saturation.

We reduce the stray light by first formulating the PSF model of the imaging system and then inverting the model. This

	a (mm^{-1})	b (mm^{-1})	α	c (mm)
PSF parameter set 1	-1.65×10^{-4}	-8.35×10^{-4}	1.21	0.015
PSF parameter set 2	4×10^{-3}	0	1.15	0.02

TABLE I
TWO SETS OF PSF PARAMETERS

method is also used in previous work [18], [26], [27], [28]. The PSF model for a stray light contaminated optical imaging system is formulated as

$$y = ((1-\beta)I + \beta S)x, \tag{15}$$

where I is an identity matrix, S accounts for stray light due to scattering and undesired reflections, and β represents the weight factor of stray light. The entries of S are described by

$$S_{q,p} = s(i_q, j_q; i_p, j_p), \tag{16}$$

where (i_q, j_q) and (i_p, j_p) are the 2D indices corresponding to q and p respectively, and $s(i_q, j_q; i_p, j_p)$ is the value of the stray light PSF at (i_q, j_q) due to a point source located at (i_p, j_p). We follow [18], [26], [27], [29] and model the stray light PSF as:

$$s(i_q, j_q; i_p, j_p) = \frac{1}{\gamma} \times \frac{1}{\left(1 + \frac{1}{i_p^2+j_p^2}\left(\frac{(i_q i_p + j_q j_p - i_p^2 - j_p^2)^2}{(c+a(i_p^2+j_p^2))^2} + \frac{(-i_q j_p + j_q i_p)^2}{(c+b(i_p^2+j_p^2))^2}\right)\right)^{\alpha}},$$

where γ is a normalizing constant that makes $\int_{i_q}\int_{j_q} s(i_q, j_q; i_p, j_p) di_q dj_q = 1$. An analytic expression for γ is $\gamma = \frac{\pi}{\alpha - 1}(c + a(i_p^2 + j_p^2))(c + b(i_p^2 + j_p^2))$. Figure 4 shows example stray light PSFs due to different point source locations with two sets of parameters in Table I. We can see that the PSF described by parameter set 2 is more space-varying than the PSF described by parameter set 1. The PSF parameters vary with different imaging systems. For a particular digital camera model and lens setting, we can use the apparatus and algorithm described in [18], [26], [27], [29] to estimate the specific stray light PSF parameters.

Once the model parameters are estimated, we follow the method in [30] to compute the restored image. The restoration formula is described in Eq. (3). Note that since our stray light PSF is global and space-varying, directly computing $z = Sy$ is quite expensive. For a 6 million pixel image, it requires 6 million multiplies per output pixel, and a total of 3.6×10^{13} multiplies. So our objective is to compress the transform matrix S using our matrix source coding technique described in Sec. II to reduce the on-line computation.

IV. PREDICTING LOCATIONS OF SIGNIFICANT WAVELET COEFFICIENTS

As discussed in Sec. II-B, we need a method to predict the location of the significant wavelet coefficients for each row of the matrix $SW_2^t \Lambda_w^{1/2}$. Notice that each row of S is an image, so the corresponding row of the matrix $SW_2^t \Lambda_w^{1/2}$ will be a scaled version of the wavelet transform of that image.

We use the stray light PSF as an example. Using stray light parameter set 1 described in Sec. III, we show the magnitude of a row of S in Fig. 5(a) for image resolution 256×256. We show the magnitude of its corresponding eight-level Haar wavelet transform in Fig. 5(b). White pixels in Fig. 5(c) indicate the location of the largest (in magnitude) 1000 wavelet coefficients after being scaled by $\Lambda_w^{1/2}$. In order to predict the location of those 1000 significant wavelet coefficients, we use a circle, as depicted in Fig. 5(d), in each band to capture all significant wavelet coefficients. Then once we can predict the center location and radius of the circle in each band, we can locate the significant wavelet coefficients.

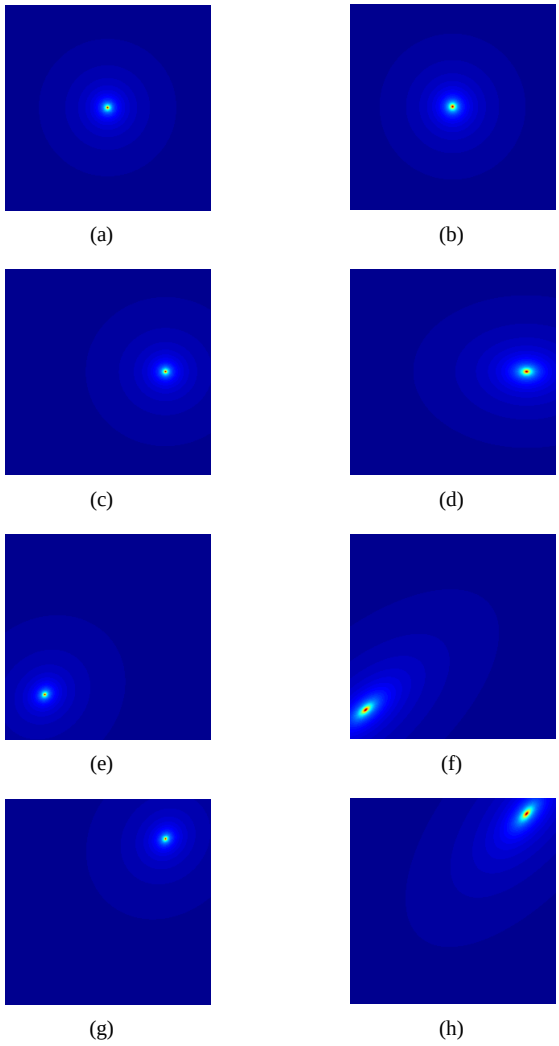


Fig. 4. Examples of stray light PSFs with different parameters and due to different point source locations in the image. Figures (a), (c), (e), and (g) show the stray light PSFs due to center, center right, bottom left, and top right point sources for PSF parameter set 1. Figures (b), (d), (f), and (h) show the stray light PSFs for PSF parameter set 2. PSFs with parameter set 2 are more space-varying than with parameter set 1.

Given a row q of S, our prediction of the circle center $(i_q^{(k)}, j_q^{(k)})$ in the kth level wavelet band is

$$\begin{aligned} i_q^{(k)} &= \frac{1}{2^k} i_q, \\ j_q^{(k)} &= \frac{1}{2^k} j_q, \end{aligned} \tag{17}$$

where $k = 1$ represents the finest resolution wavelet band.

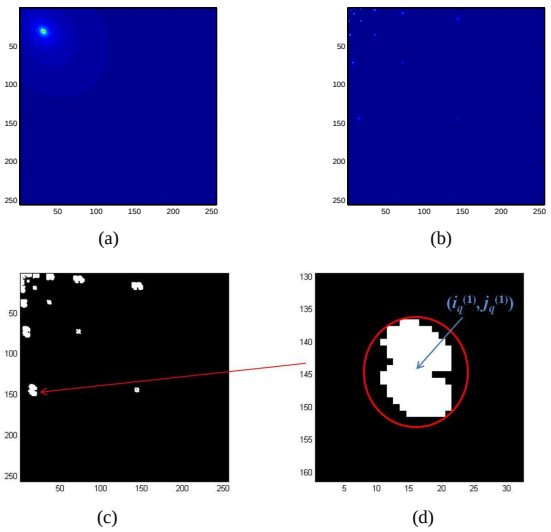


Fig. 5. An example of a row of S displayed as an image together with its wavelet transform, and the locations of significant wavelet coefficients. Figure (a) shows the amplitude map of a row image of S raised to the power of $1/3$. Figure (b) shows the absolute value of the scaled wavelet transform of the row image shown in Figure (a) raised to the power of $1/3$. Figure (c) shows the locations of non-zero pixels of Figure (b) after quantization. These locations are locations of significant wavelet coefficients for the row image shown in Figure (a). Figure (d) zooms in to one vertical band in Figure (c), with a red circle to capture all significant wavelet coefficients in that band.

In order to predict the radius of the circle in each band, we first need to decide the average number of significant wavelet coefficients we want to keep for each row of S. As mentioned before, we let this target number of significant wavelet coefficients per row be N. In the example shown in Fig. 5, $N = 1000$. We then take 49 rows of S with corresponding (i_q, j_q)'s uniformly distributed across the image. We take the eight-level Haar wavelet transform of those 49 row images, respectively. We then scale the resulting wavelet transforms using $\Lambda_w^{1/2}$, and keep only the largest N coefficients in each row. For each wavelet band, we find the maximum Euclidean distance between the location of the significant wavelet coefficients and the predicted center of the circle for each row, and then we take the maximum among all rows as the radius of the circle. After the center and radii of the circles are predicted, we consider all pixels falling inside the circles as significant wavelet coefficients.

V. EXPERIMENTAL RESULTS

Our assessment of performance is distortion versus computation. We measure distortion using the normalized root mean square error (NRMSE) in computing $z = Sy$. We measure computation using the number of real multiplies per output pixel. The online computation of our matrix source coding algorithm takes $1 + |[\tilde{S}]|/P$ multiplies per output pixel, where $|[\tilde{S}]|$ represents the total number of non-zero entries in the sparse matrix. The online computation includes a Haar wavelet transform W_2, a scaling by $\Lambda_w^{-1/2}$, a multiplication with a sparse matrix $[\tilde{S}]$, and an inverse wavelet transform W_1^{-1}. The Haar wavelet transform W_2 and inverse wavelet transform W_1^{-1} can be implemented without multiplies. Scaling takes 1 multiply per output pixel. Multiplication with sparse matrix $[\tilde{S}]$ takes $|[\tilde{S}]|/P$ multiplies per output pixel. So that is a total of $1 + |[\tilde{S}]|/P$ multiplies per output pixel.

In our experiments, in order to measure distortion, we take 10 different natural images as testing input images y and average the NRMSE in computing z. This average NRMSE is used to quantify distortion. Note that since computing the exact space-varying convolution result z over the whole image is expensive, we arbitrarily select 1000 pixels in each image at which to compute the NRMSE. In addition, as described in Sec. II-A, we also use 30 natural images as training images to estimate the scaling matrix Λ_w. We only estimate a single scalar value for each subband. So a total of $3K + 1$ scalar values are estimated for Λ_w, where K is the number of levels of the wavelet transform W_2.

In the following experiments, we first show the result if we only decorrelate the columns of S. Then we show the result if we decorrelate both the columns and the rows of S. We also compare our algorithm with the overlap-and-add method. Finally, we apply our matrix source coding algorithm to stray light reduction in natural images.

A. Results of Decorrelating Only the Columns of S

In this subsection, we only decorrelate the columns of S. So we approximate z as

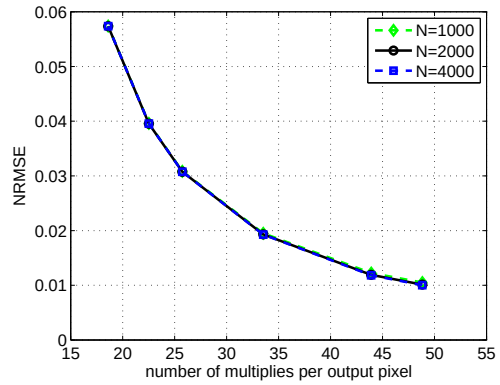
$$\begin{aligned} \hat{z} &= [Q_t(SW_2^{-1}\Lambda_w^{1/2})]\Lambda_w^{-1/2}W_2 y, \\ &\approx z. \end{aligned} \tag{18}$$

The normalized root mean square error (NRMSE) can be computed as:

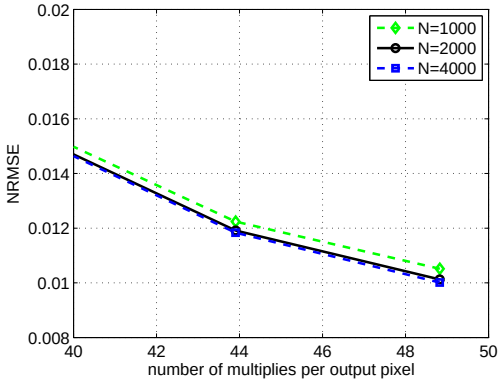
$$\text{NRMSE} = \sqrt{\frac{\|z - \hat{z}\|_2^2}{\|z\|_2^2}}. \tag{19}$$

In this experiment, the image resolution is 256×256. We use the eight-level Haar wavelet transform for W_2. We choose to use stray light PSF parameter set 1 described in Sec. III in this experiment.

When we use our fast algorithm to compute the significant Haar wavelet coefficients for the rows of S, we can choose a number N as the target number of significant coefficients per row to compute before quantization. We conduct three experiments with $N = 1000$, $N = 2000$, and $N = 4000$, respectively. We then vary quantization intervals when quantizing $Q_t(SW_2^{-1}\Lambda_w^{1/2})$ in Eq. (18), and plot the NRMSE as a function of the number of multiplies per output pixel for these three choices in Fig. 6. From the plot, we can see that the more significant coefficients we compute, the better performance we can achieve. However, more coefficients to compute also means more off-line computation. In addition, if we take a closer look at Fig. 6, we will find that $N = 4000$ produces little performance gain compared to $N = 2000$. But, the off-line computation is about twice as great. So in the following experiments, unless otherwise specified, we choose $N = 2000$ for image resolution 256×256. Figure 6 also shows that by only decorrelating the columns S, our matrix source coding algorithm produces 1.2% error with approximately 44 multiplies per output pixel.



(a) Plot of NRMSE



(b) Zoom in partial plot

Fig. 6. Plot of NRMSE as a function of the number of multiplies per output pixel using our matrix source coding algorithm, but only decorrelating the columns of S. The green dashed line, the black solid line, and the blue dashed line represent the results when the target number of significant wavelet coefficients per row $N = 1000$, $N = 2000$, and $N = 4000$, respectively. We can see that for the same number of multiplies per output pixel, increasing N reduces NRMSE, although the difference is small.

B. Results of Decorrelating Both the Columns and Rows of S

In addition to decorrelating the columns, we also decorrelate the rows with wavelet transform W_1. Then z is approximated as

$$\begin{aligned} \hat{z} &= W_1^{-1}[W_1 Q_t(SW_2^t \Lambda_w^{1/2})]\Lambda_w^{-1/2} W_2 y, \\ &\approx z. \end{aligned} \tag{20}$$

We use PSF parameter set 1, and plot in Fig. 7 the average NRMSE against number of multiplies per output pixel, and compare it with the performance of decorrelating only columns. We can see that to achieve 1% error, decorrelating only columns requires approximately 48 multiplies per output pixel, whereas decorrelating both rows and columns requires approximately 14 multiplies per output pixel. Therefore, this experiment demonstrates the significance of decorrelating both rows and columns of the matrix S.

In addition, to better understand the computational savings, we use two different image resolutions of 256×256 and 1024×1024 in our experiments. In both cases, we set the target number of significant wavelet coefficients per row to be $N = 2000$. We use two sets of PSF parameters described in Sec. III to conduct our experiments. In the 1024×1024 case, we use a ten-level Haar wavelet transform for W_2 and

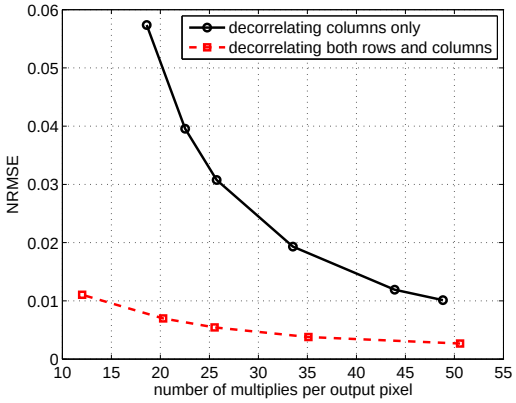
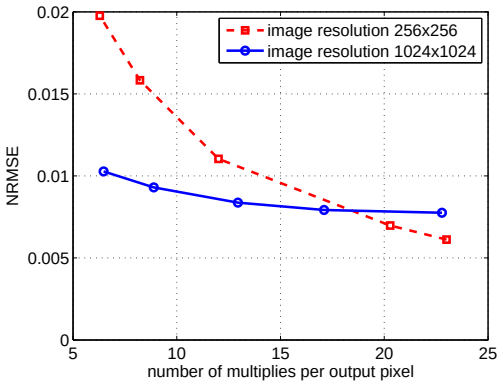


Fig. 7. Comparison of performance between decorrelating columns only and decorrelating both rows and columns. The black solid curve depicts the performance of decorrelating columns only. The red dashed curve depicts the performance of decorrelating both rows and columns.

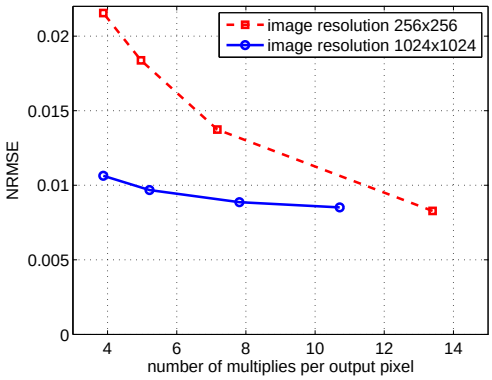
a three-level block Haar wavelet transform for W_1. We plot the NRMSE against number of multiplies per output pixel in Fig. 8. We can see that for PSF parameter set 1, at 1024×1024 resolution, with only 1% error, our algorithm reduces the number of multiplies per output pixel from $1048576 = 1024^2$ to only 7, which is a $149796:1$ reduction in computation. At 256×256 resolution, to achieve the same accuracy, we need about 14 multiplies per output pixel. So our algorithm achieves a higher reduction in computation for higher resolutions. In addition, we find that the error for the 1024×1024 case does not decrease much when the number of multiplies per pixel becomes large. That is because the error introduced in the first stage quantization $Q_t(SW_2^t \Lambda_w^{1/2})$ does not change for different quantization intervals in the second stage. If we try to reduce the error in the first stage by increasing N from 2000 to 4000, we can see the difference in performance in Fig. 9, especially for the 1024×1024 case. For the 256×256 case, the performance is almost the same for $N = 2000$ and $N = 4000$. This is because the error introduced in the first stage is almost identical for $N = 2000$ and $N = 4000$, as shown in Fig. 6. But for the 1024×1024 case, the performance is boosted by increasing the value of N.

C. Comparison with Overlap-and-add Method

A conventional way to compute space-varying convolution is to use the overlap-and-add method [31], [32]. This method first partitions the input image y into blocks, then computes the output for each block over a larger (overlapped) region, and finally adds the results together to approximate the final output image z. The reason why it can be fast is that when computing the output for each block, it assumes spatially invariant PSF for that particular block, thus fast Fourier transform (FFT) can be used to significantly reduce computational complexity. Similar to our algorithm, this method also has a trade-off between speed and accuracy. More block partitions will produce higher accuracy, but requires more computation. The size of the output region for each block depends on the size of the block and the spatial support of the PSF. If the block is of size $h \times h$, and the PSF has a spatial support of $(r+1) \times (r+1)$, then the

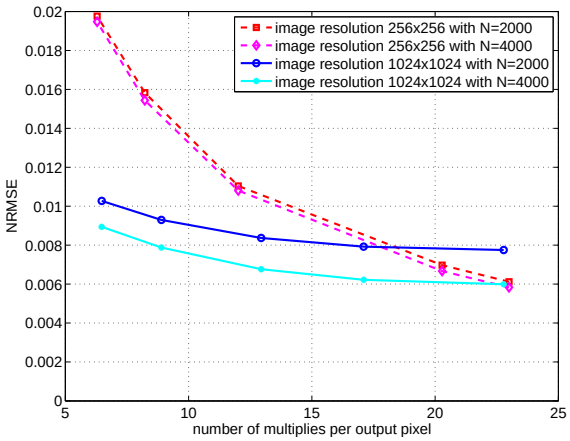


(a) PSF parameter set 1

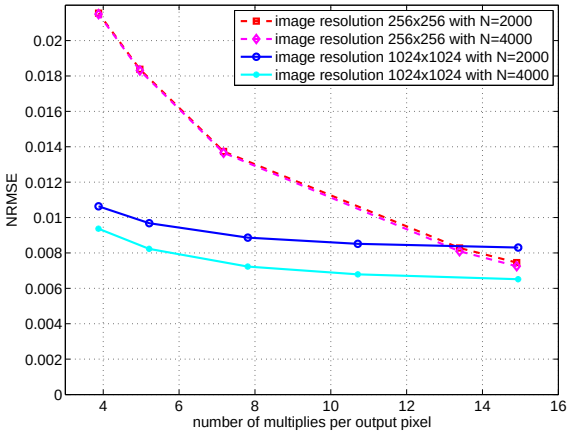


(b) PSF parameter set 2

Fig. 8. Comparison of performance between different resolutions. The red dashed line shows the performance for image resolution 256×256. The blue solid line shows the performance for image resolution 1024×1024. Figures (a) and (b) show the results for PSF parameter set 1 and set 2 respectively.



(a) PSF parameter set 1



(b) PSF parameter set 2

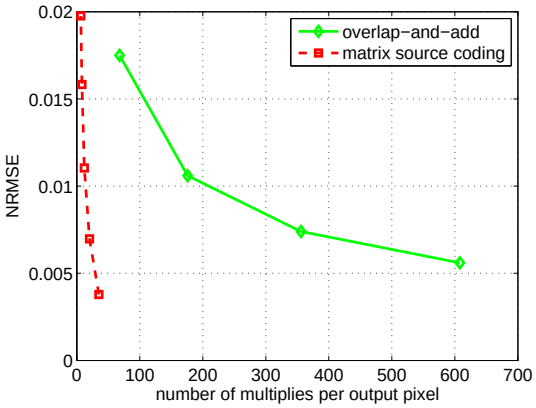
Fig. 9. Comparison of performance between different target number of significant wavelet coefficients per row. This number is denoted by N, and it is decided in the first stage quantization. Figures (a) and (b) show the results for PSF parameter set 1 and set 2, respectively.

output region for each block should be of size $(r+h)\times(r+h)$ [31]. Since the spatial support of stray light PSF is comparable to the size of the image, the output region for each block will be the entire image area.

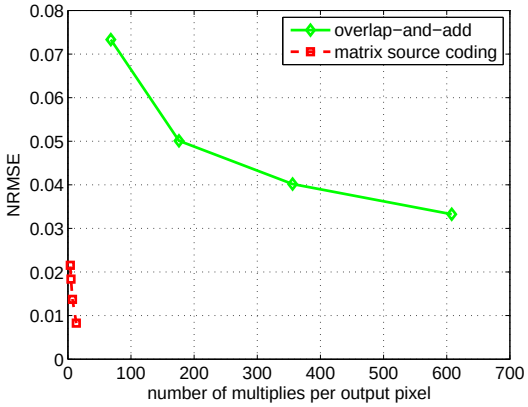
Now we compare our algorithm with the overlap-and-add method [31] in terms of the amount of computation, storage requirement, and accuracy. The amount of computation of the overlap-and-add method is $2P(M+1)\log(P)+4MP$ real multiplies, where M is the number of blocks. A discrete Fourier transform (DFT) of each of the input image blocks costs $2P\log(P)$ real multiplies using the divide and conquer approach described by Cooley and Tukey [33]. Multiplication with the DFT of the PSF requires $4P$ real multiplies for each block. An inverse DFT takes $2P\log(P)$ real multiplies. So that is a total of $2P(M+1)\log(P)+4MP$ real multiplies, and equivalently $2(M+1)\log(P)+4M$ multiplies per output pixel.

In terms of storage or memory usage, the overlap-and-add method requires storing the Fourier transform of the PSF corresponding to each of the M blocks. Otherwise, computing the PSF online will require power operations and divisions, which are very expensive. So the overlap-and-add method requires storing $2M$ real values per output pixel. For our matrix source coding method, the storage requirement is $||[\tilde{S}]||/P$ real values per output pixel.

We perform experiments using the two sets of stray light PSF parameters given in Sec. III. We test on 10 natural images with resolution 256×256. We use different numbers of blocks for the overlap-and-add method: $M=1$, $M=4$, $M=9$, and $M=16$. We then plot the average NRMSE (over 10 input images) against the number of multiplies per output pixel in Fig. 10 to get a straightforward comparison with our matrix source coding method. We can see that our matrix source coding method outperforms the overlap-and-add method by a large margin. In addition, the performance of our matrix source coding method does not change much for the two different sets of PSF parameters, whereas the performance of the overlap-and-add method deteriorates significantly when the PSF becomes more space-varying with parameter set 2. We also plot the average NRMSE against storage in Fig. 11. For PSF parameter set 1, if the number of stored real values per output pixel is less than 18, our matrix source coding method introduces more error than the overlap-and-add method with the same amount of storage. Otherwise, our method introduces less error than the overlap-and-add method with the same amount of storage. In addition, when the PSF becomes more

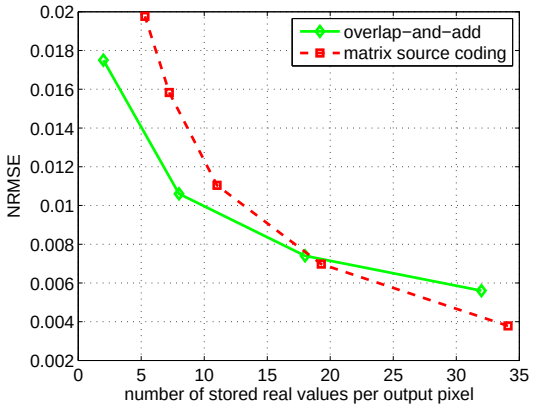


(a) PSF parameter set 1

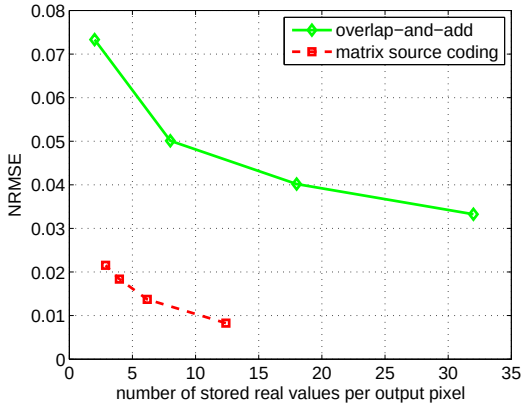


(b) PSF parameter set 2

Fig. 10. Plots of NRMSE against the number of multiplies per output pixel for both the overlap-and-add method and matrix source coding algorithm over two different PSF parameter sets. The green solid line shows the result of the overlap-and-add method. The red dashed line shows the result of matrix source coding algorithm. The PSF using parameter set 2 is more space-varying than the PSF using parameter set 1.



(a) PSF parameter set 1



(b) PSF parameter set 2

Fig. 11. Plots of NRMSE against storage for both the overlap-and-add method and matrix source coding algorithm over two different PSF parameter sets. The green solid line shows the result of the overlap-and-add method. The red dashed line shows the result of matrix source coding algorithm. The PSF using parameter set 2 is more space-varying than the PSF using parameter set 1.

space-varying in PSF parameter set 2, our matrix source coding method introduces much less error than the overlap-and-add method with the same amount of storage. For example, with 5 real values stored per output pixel, our matrix source coding method produces an error of approximately 1.5%, whereas the overlap-and-add method produces an error of approximately 6%.

In order to demonstrate the results more intuitively, we use the PSF parameter set 2 provided in Sec. III, and compute the exact convolution of the stray light PSF with a sample input image shown in Fig. 12(a) at resolution 256×256. The exact convolution result is shown in Fig. 12(b). We show the approximation results using the overlap-and-add method with one partition, four partitions, and sixteen partitions in Fig. 13 (a), (c), and (e) respectively. The absolute difference images between the approximation results and the ground truth in Fig. 12 (b) are shown in Fig. 13 (b), (d), and (f). The NRMSE is 9.89% for one partition, 6.25% for four partitions, and 3.96% for sixteen partitions. We can see that the error is much reduced by using more partitions. We then compute the approximation using our matrix source coding algorithm with 15 multiplies per output pixel and show the result in

(a) (b)

Fig. 12. A sample test image for space-varying convolution with stray light PSF. Figure (a) shows an input image. Figure (b) shows the exact result of the input image convolved with a space-varying stray light PSF with parameters set 2.

Fig. 13(g). The difference image between our approximation and the ground truth is shown in Fig. 13(h). The NRMSE is 0.74%. This error is even less than the one achieved by the overlap-and-add method with sixteen partitions, which needs 608 multiplies per output pixel.

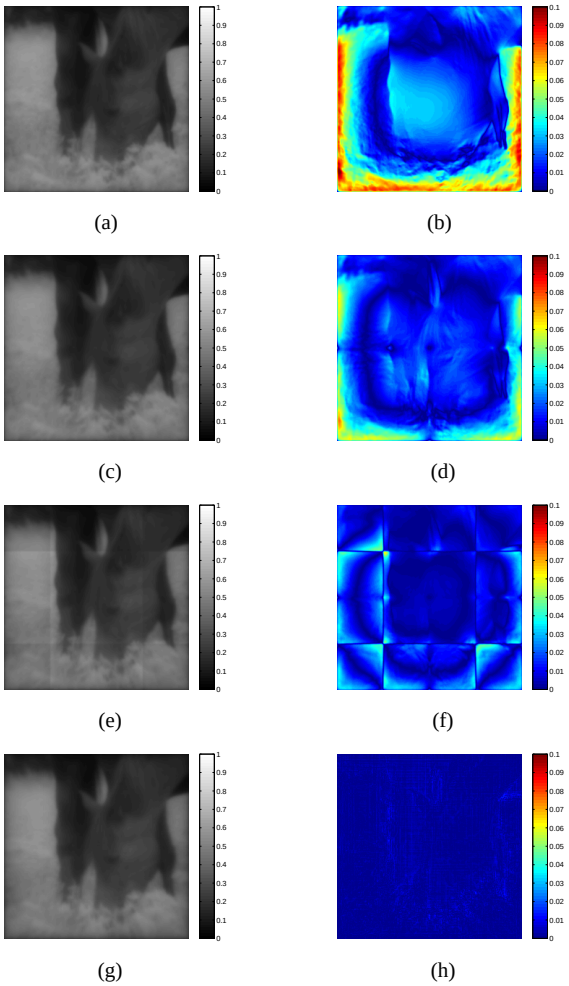


Fig. 13. Approximation to space-varying convolution with stray light PSF (parameter set 2) using the overlap-and-add method and our matrix source coding method. Figures (a), (c), and (e) show the result using the overlap-and-add method with 1, 4, and 16 partitions, respectively. Figures (b), (d), and (f) show the absolute difference images between ground truth (shown in Fig. 12(b)) and the approximation result in (a), (c), and (e), respectively. Figure (g) shows the approximation result using matrix source coding method with 15 multiplies per output pixel, and Figure (h) shows the absolute difference image between (g) and the grouth truth. The results (a), (c), (e), and (g) have NRMSE of 0.0989, 0.0625, 0.0396, and 0.0074, respectively, with corresponding number of multiplies per pixel 68, 176, 608, and 15.

D. Matrix Source Coding Results for Stray Light Reduction

We use Eq. (3) to perform stray light reduction for images taken by an Olympus SP-510UZ camera[5]. The estimated stray light PSF parameters are $a = -1.65 \times 10^{-5}$ mm^{-1}, $b = -5.35 \times 10^{-5}$ mm^{-1}, $\alpha = 1.31$, $c = 1.76 \times 10^{-3}$ mm, $\beta = 0.3937$. In the stray light reduction formula described by Eq. (3), the space-varying convolution Sy is efficiently computed using our matrix source coding approach. The image resolution of this camera is 3088×2310. In this experiment, we use a ten-level Haar wavelet transform for W_2, and a three-level block Haar wavelet transform for W_1, with block size 64×64. We randomly select 1000 pixels on each of the 10 testing images, and calculate the average NRMSE in computing Sy of Eq. (3). We vary the quantization intervals in the second stage quantization of our matrix source coding, and plot the average NRMSE against the number of multiplies per output pixel and the number of stored real values per output pixel in Fig. 14.

[5]Olympus America Inc., Center Valley, PA 18034

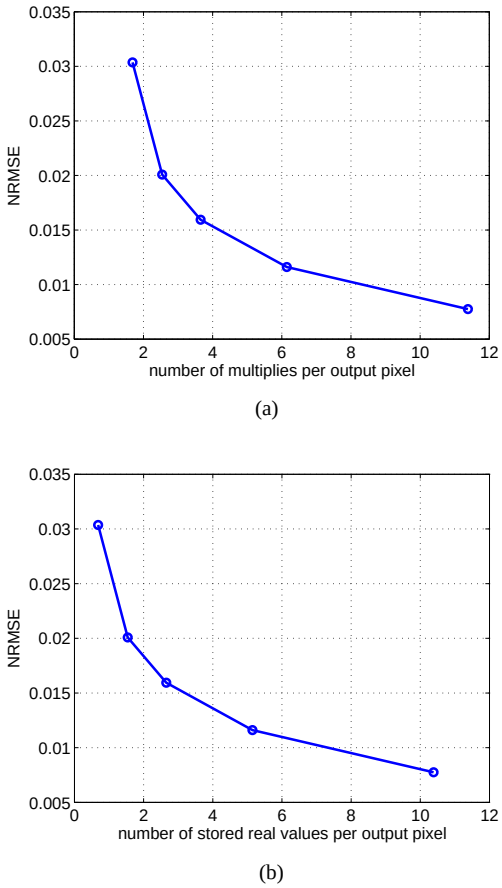
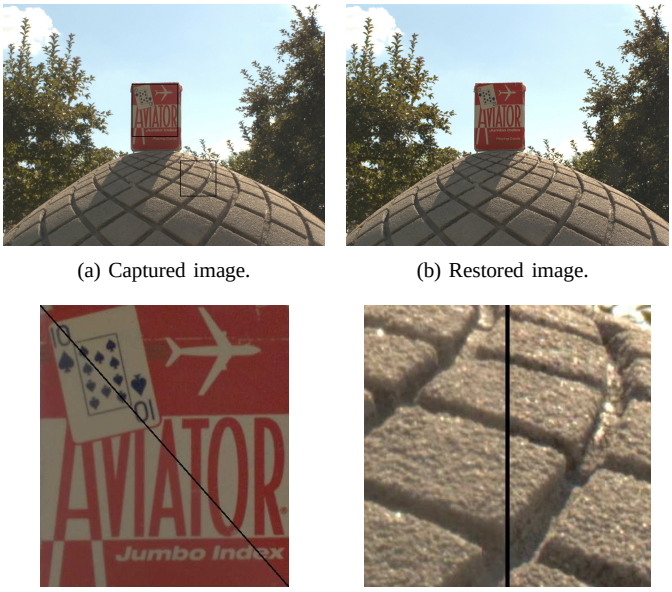


Fig. 14. Plot of NRMSE for computing Sy against the number of multiplies per output pixel and the number of stored real values per output pixel for a real camera stray light PSF for testing images with resolution 3088×2310. Figure (a) shows the plot of NRMSE against number of multiplies per output pixel. Figure (b) shows the plot of NRMSE against the number of stored real values per output pixel.

We select a quantization level such that the online computation requires 6 multiplies per output pixel, and the corresponding storage requirement is 5 real values per output pixel. We then use our matrix source coding technique to compute Sy and apply Eq. (3) to perform stray light reduction. Figure 15 shows an example of the results of the stray light reduction. Figure 15(a) is the captured image. Figure 15(b) is the restored image. Figures 15(c-d) shows a comparison between the captured and restored versions for different parts of the image. From this example, we can see that the stray light reduction algorithm increases the contrast and recovers lost details of the original scene.

VI. CONCLUSION

In this paper, we proposed a matrix source coding algorithm for efficiently computing space-varying convolution. Our approach reduced the computation from $O(P^2)$ to $O(P)$, where P is the number of pixels in the image. We presented both the on-line and the off-line parts of our algorithm, along



(a) Captured image. (b) Restored image.

(c) Captured \ restored. (d) Captured | restored.

Fig. 15. Example of stray light reduction on a 7 mega-pixel camera image: (a) shows a captured image, (b) shows the restored image, (c) and (d) show a comparison between captured and restored images for two parts of the image.

with its associated theory. In the on-line part, we developed an algorithm to approximate a dense transform with wavelet transforms and a sparse matrix-vector product.

In the off-line part, we developed a two stage approach to make the sparsification of the dense transform matrix feasible. We also developed a fast algorithm for computing significant Haar wavelet coefficients of PSF images to accelerate our off-line computation. We use a top-down recursive approach, rather than the bottom-up approach used by a conventional filter-bank implementation. The value of this algorithm is to obtain the significant wavelet coefficients with $O(N)$ complexity, where N is the number of significant wavelet coefficients. By using this algorithm and our two stage approach, our off-line matrix source encoding process has a computational complexity of $O(NP)$.

Finally, we applied our matrix source coding algorithm to stray light reduction, where space-varying convolution with a PSF of large support is necessary. Our experimental results showed a trade-off between speed and accuracy. Our algorithm is able to achieve a $149796:1$ reduction in computation with only 1% error at image resolution 1024×1024. The experimental results also demonstrated that our matrix source coding algorithm outperforms the overlap-and-add method by a large margin.

Appendix A
Ideal Decorrelating Transforms for Matrix Source Coding

In this appendix, we derive the ideal decorrelating transforms W_1 and T that simultaneously whiten y and decorrelate the rows and columns of S. To do this, we define the transformed matrix $\tilde{S}$ and vector $\tilde{y}$ as

$$\begin{aligned} \tilde{S} &= W_1 S T^{-1}, && (21)\\ \tilde{y} &= Ty, && (22)\end{aligned}$$

where W_1 is an orthonormal transform[6] and T is an invertible transform. Then the output z can be computed as

$$z = W_1^{-1}\tilde{S}\tilde{y}\ . \quad (23)$$

So the our objective is to select matrices W_1 and T that simultaneously whiten the vector y and decorrelate the rows and columns of S.

We choose transformation W_1 to be the eigen-transformation for $R_r \triangleq SS^t$, the covariance of the rows of S, so that

$$W_1^t R_r W_1 = \Sigma_r, \quad (24)$$

where Σ_r is a diagonal matrix. Under this choice, W_1 decorrelates the rows of S. In the rest of this appendix, we construct transform T, and we show that they solve the equations

$$\begin{aligned} TR_yT^t &= I\\ T^{-t}R_cT^{-1} &= \Sigma_c, && (25)\end{aligned}$$

where $R_c \triangleq S^tS$ is the covariance matrix for the columns of S, Σ_c is a diagonal matrix, and T^{-t} denotes the transposed inverse matrix of T. We notice that R_y is a symmetric matrix, so we have the following eigendecomposition

$$R_y = E_y\Lambda_y E_y^t, \quad (26)$$

where Λ_y is a diagonal matrix with the eigenvalues of R_y on its diagonal, and E is an orthonormal matrix, and

$$E_y^t R_y E_y = \Lambda_y. \quad (27)$$

If we define

$$F \triangleq E_y\Lambda_y^{-1/2}, \quad (28)$$

then we have

$$F^t R_y F = I. \quad (29)$$

We let matrix

$$\tilde{R_c} \triangleq F^{-1}R_cF^{-t}. \quad (30)$$

We perform eigendecomposition of matrix $\tilde{R_c}$,

$$\tilde{R_c} = E_{\tilde{c}}\Sigma_c E_{\tilde{c}}^t. \quad (31)$$

Then the matrix T can be constructed as

$$\begin{aligned} T &= E_{\tilde{c}}^{-1}F^t\\ &= E_{\tilde{c}}^{-1}\Lambda_y^{-1/2}E_y^t. && (32)\end{aligned}$$

Now we can verify that this construction of T follows Eq. (25).

$$\begin{aligned} TR_yT^t &= E_{\tilde{c}}^{-1}\Lambda_y^{-1/2}E_y^tR_yE_y\Lambda_y^{-1/2}E_{\tilde{c}}^{-t}\\ &= E_{\tilde{c}}^{-1}\Lambda_y^{-1/2}\Lambda_y\Lambda_y^{-1/2}E_{\tilde{c}}^{-t}\\ &= E_{\tilde{c}}^{-1}E_{\tilde{c}}^{-t}\\ &= I. && (33)\end{aligned}$$

[6]Without loss of generality, these transforms can just be orthogonal, but for notional simplicity we also assume that they are normal.

The last line of the above equation follows from the fact that matrix $E_{\tilde{c}}$ is an orthonormal matrix, whose inverse is equal to its transpose. In addition,

$$\begin{aligned} T^{-t}R_cT^{-1} &= E_{\tilde{c}}^t F^{-1} R_c F^{-t} E_{\tilde{c}} \\ &= E_{\tilde{c}}^t \tilde{R}_c E_{\tilde{c}} \\ &= \Sigma_c. \end{aligned} \tag{34}$$

So the matrix T from our construction fits Eq. (25). From our construction, we can see that both T and W_1 are orthogonal matrices. But they are not necessarily sparse or can be converted to sparse matrices. Therefore, these ideal decorrelating matrices are not useful in practice for the purpose of fast computation.

REFERENCES

[1] A. Bovik, *Handbook of image and video processing*. San Diego, CA: Academic Press, 2000.

[2] G. T. Herman and A. Kuba, *Discrete Tomography: Foundations, Algorithms, and Applications*. New York, NY: Birkhauser Boston, 1999.

[3] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, May 2003.

[4] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2001.

[5] P. Gilbert, "Iterative methods for the three-dimensional reconstruction of an object from projections," *Journal of Theoretical Biology*, vol. 36, no. 1, pp. 105–117, July 1972.

[6] A. H. Anderson and A. C. Kak, "Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm," *Ultrasonic Imaging*, vol. 6, no. 1, pp. 81–94, January 1984.

[7] W. H. Richardson, "Bayesian-based iterative method of image restoration," *Journal of the Optical Society of America*, vol. 62, no. 1, pp. 55–59, 1972.

[8] P. H. V. Cittert, "Zum einfluss der spaltbreite auf die intensitatswerteilung in spektrallinien," *Z. Physik*, vol. 69, pp. 298–308, 1931.

[9] H. M. Hudson and R. S. Larkin, "Accelerated image reconstruction using ordered subsets of projection data," *IEEE Transactions on Medical Imaging*, vol. 13, no. 4, pp. 601–609, 1994.

[10] L. A. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Transactions on Medical Imaging*, vol. 2, pp. 113–122, 1982.

[11] A. N. Tychonoff and V. Y. Arsenin, *Solution of Ill-posed Problems*. Washington D. C.: V. H. Winston & Sons, January 1977.

[12] T. Hebert and R. Leahy, "A generalized EM algorithm for 3-D Bayesian reconstruction from Poisson data using Gibbs priors," *IEEE Transactions on Medical Imaging*, vol. 8, pp. 194–202, 1989.

[13] P. J. Green, "Bayesian reconstruction from emission tomography data using a modified EM algorithm," *IEEE Transactions on Medical Imaging*, vol. 9, pp. 84–93, 1990.

[14] C. Bouman and K. Sauer, "A generalized gaussian image model for edge-preserving map estimation," *IEEE Transactions on Image Processing*, vol. 2, no. 3, pp. 296–310, July 1993.

[15] C. Lanczos, *Linear Differential Operators*. Mineola, NY: Dover Publications, 1997.

[16] S. Mallat, *A Wavelet Tour of Signal Processing*. San Diego, CA: Academic Press, 1998.

[17] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, April 1965.

[18] J. Wei, B. Bitlis, A. Bernstein, A. Silva, P. A. Jansson, and J. P. Allebach, "Stray light and shading reduction in digital photography – a new model and algorithm," in *Proceedings of the SPIE/IS&T Conference on Digital Photography IV*, vol. 6817, San Jose, CA, January 2008.

[19] W. J. Smith, *Modern Optical Engineering : the Design of Optical Systems*. New York, NY: McGraw Hill, 2000.

[20] P. A. Jansson and J. H. Fralinger, "Parallel processing network that corrects for light scattering in image scanners," *U.S. Patent 5,153,926*, 1992.

[21] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore, Maryland: The Johns Hopkins University Press, 1996.

[22] G. Cao, C. A. Bouman, and K. J. Webb, "Fast and efficient stored matrix techniques for optical tomography," in *Proceedings of the 40th Asilomar Conference on Signals, Systems, and Computers*, October 2006.

[23] ——, "Results in non-iterative map reconstruction for optical tomography," in *Proceedings of the SPIE/IS&T Conference on Computational Imaging VI*, vol. 6814, San Jose, CA, January 2008.

[24] D. Tretter and C. A. Bouman, "Optimal transforms for multispectral and multilayer image coding," *IEEE Transactions on Image Processing*, vol. 4, no. 3, pp. 296–308, 1995.

[25] D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Norwell, MA: Kluwer Academic Publishers, 2002.

[26] B. Bitlis, P. A. Jansson, and J. P. Allebach, "Parametric point spread function modeling and reduction of stray light effects in digital still cameras," in *Proceedings of the SPIE/IS&T Conference on Computational Imaging VI*, vol. 6498, San Jose, CA, January 2007.

[27] P. A. Jansson, "Method, program, and apparatus for efficiently removing stray-flux effects by selected-ordinate image processing," *U.S. Patent 6,829,393*, 2004.

[28] J. Wei, G. Cao, C. A. Bouman, and J. P. Allebach, "Fast space-varying convolution and its application in stray light reduction," in *Proceedings of the SPIE/IS&T Conference on Computational Imaging VII*, vol. 7246, San Jose, CA, February 2009.

[29] P. A. Jansson and R. P. Breault, "Correcting color-measurement error caused by stray light in image scanners," in *Proceedings of the Sixth Color Imaging Conference: Color Science, Systems, and Applications*, Scottsdale, AZ, November 1998.

[30] P. A. Jansson, *Deconvolution of Images and Spectra*. New York, NY: Academic Press, 1996.

[31] J. G. Nagy and D. P. O'Leary, "Fast iterative image restoration with a spatially varying PSF," in *Proceedings of SPIE Conference on Advanced Signal Processin g: Algorithms, Architectures, and Implementations VII*, vol. 3162, San Diego, CA, 1997, pp. 388–399.

[32] J. Bardsley, S. Jefferies, J. Nagy, and R. Plemmons, "A computational method for the restoration of images with an unknown, spatially-varying blur," *Optics Express*, vol. 15, no. 5, pp. 1767–1782, 2006.

[33] P. S. R. Diniz, E. A. B. da Silva, and S. L. Netto, *Digital Signal Processing System Analysis and Design*. New York, NY: Cambridge University Press, 2002.