Fast Space-varying Convolution and Its Application in Stray Light Reduction*

Jianing Wei[a], Guangzhi Cao[a], Charles A. Bouman[a], and Jan P. Allebach[a]

[a]School of Electrical Engineering, Purdue University, West Lafayette, IN 47907-0501, USA

ABSTRACT

Space-varying convolution often arises in the modeling or restoration of images captured by optical imaging systems. For example, in applications such as microscopy or photography the distortions introduced by lenses typically vary across the field of view, so accurate restoration also requires the use of space-varying convolution. While space-invariant convolution can be efficiently implemented with the Fast Fourier Transform (FFT), space-varying convolution requires direct implementation of the convolution operation, which can be very computationally expensive when the convolution kernel is large.

In this paper, we develop a general approach to the efficient implementation of space-varying convolution through the use of matrix source coding techniques. This method can dramatically reduce computation by approximately factoring the dense space-varying convolution operator into a product of sparse transforms. This approach leads to a tradeoff between the accuracy and speed of the operation that is closely related to the distortion-rate tradeoff that is commonly made in lossy source coding.

We apply our method to the problem of stray light reduction for digital photographs, where convolution with a spatially varying stray light point spread function is required. The experimental results show that our algorithm can achieve a dramatic reduction in computation while achieving high accuracy.

Keywords: space-varying convolution, stray light, image restoration, matrix source coding

1. INTRODUCTION

Space-varying convolution often arises in the modeling or restoration of images captured by optical imaging systems. This is due to the fact that the point spread function (PSF) and/or distortion introduced by a lens typically varies across the field of view,[1] so accurate restoration also requires the use of space-varying convolution.

In this work, we will consider the problem of stray light reduction for digital photographs.[2] In all optical imaging systems, a small portion of the entering light flux is misdirected to undesired locations in the image plane. We refer to this severely misdirected light as stray light, but it is sometimes referred to as lens flare or veiling glare.[1] 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.

The stray light reduction algorithm is originally due to Jansson and Fralinger.[3] It can be described by[2,4] the following equation:

$$\hat{x} = 2y - (1-\beta)y - \beta Sy, \tag{1}$$

where y is the observed image, $\hat{x}$ is the estimate of the underlying image to be recovered, S is a matrix representing convolution with the stray light PSF, and β is the weight of stray light. If S implements space-invariant 2D circular convolution, then Sy can be computed using a 2D FFT, which can dramatically reduce computation.[5]

*Research partially supported by E. I. du Pont De Nemours and Company.

Further author information: (Send correspondence to Jianing Wei)
Jianing Wei: E-mail: wei2@purdue.edu, Telephone: 1 765 494 3465
Guangzhi Cao: E-mail: gcao@purdue.edu, Telephone: 1 765 494 6553
Charles A. Bouman: E-mail: bouman@purdue.edu, Telephone: 1 765 494 0340
Jan P. Allebach: E-mail: allebach@purdue.edu, Telephone: 1 765 494 3535

However, the stray light PSF is space-varying, so FFT cannot be directly used to speed up computation. In addition, the support of the stray light PSF is very large, which makes S a dense matrix, and the computation of Sy extremely expensive. If the image contains N pixels, the computation is $O(N^2)$. 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 means hours of processing time, which makes it infeasible for real-world applications of the stray light reduction algorithm.

In this paper, we introduce a novel approach for efficient computation of space-varying convolution, based on the theory of matrix source coding;[6] and we demonstrate how the technique can be used to efficiently implement the stray light reduction for digital cameras. The matrix source coding technique uses the method of lossy source coding which makes the dense matrix S sparse. This is done by first decorrelating the rows and columns of S and then quantizing the resulting compacted matrix so that most of its entries become zero. By making S sparse, we not only save storage, but we also dramatically reduce the computation of the required matrix-vector product.

Our experimental results indicate that, by using this approach, we can reduce the computational complexity of space-varying convolution from $O(N^2)$ to $O(N)$. For digital images exceeding 1 Meg pixel in size, this makes deconvolution practically possible, and results in deconvolution algorithms that can be computed with 5 to 10 multiplies per output pixel.

In the rest of this paper, we start by introducing the stray light contamination model and the algorithm for stray light reduction. We then elaborate on the theory and implementation of the matrix source coding approach for space-varying convolution with the stray light point spread function. Finally, we present experimental results to demonstrate the effectiveness of our algorithm.

2. STRAY LIGHT REDUCTION STRATEGY

2.1 PSF Model Formulation

The PSF model for a stray light contaminated optical imaging system is supposed to account for the following contributions: (1) diffractive spreading owing to the finite aperture of the imaging device; (2) aberrations arising from non-ideal aspects of the device design and device fabrication material; and (3) stray light due to scattering and undesired reflections. Therefore, the forward model of stray light contamination can be expressed as

$$y = ((1-\beta)G + \beta S)x, \tag{2}$$

where G accounts for diffraction and aberration, S accounts for stray light due to scattering and undesired reflections, β represents the weight factor of stray light. The entries of G are described by

$$G_{q,p} = g(i_q, j_q; i_p, j_p),$$

where (i_p, j_p) and (i_q, j_q) are the 2D positions of the input and output pixel locations, measured relative to the center of the camera's field of view. Using this notation, we have modeled the diffraction and aberration PSF with the following function[2,4,7,8]

$$g(i_q, j_q; i_p, j_p) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i_q - i_p)^2 + (j_q - j_p)^2}{2\sigma^2}\right). \tag{3}$$

Similarly, the entries of S are described by

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

We model this stray light PSF with the following function[2,4,7,8]

$$s(i_q, j_q; i_p, j_p) = \frac{1}{z}\frac{1}{\left(1 + \frac{1}{i_p^2+j_p^2}\left(\frac{(i_q i_p + j_q j_p - i_p^2 - j_p^2)^2}{(c+a(i_p^2+j_p^2))^2} + \frac{(-i_q j_p + j_q i_p)^2}{(c+b(i_p^2+j_p^2))^2}\right)\right)^{\alpha}}, \tag{4}$$

where z 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 z is $z = \frac{\pi}{\alpha-1}(c + a(i_p^2 + j_p^2))(c + b(i_p^2 + j_p^2))$. The model parameters are then $(a, b, c, \alpha, \beta, \sigma)$. Previous work[2,4,7,8] has established an approach for estimating these parameters. We use the same apparatus and algorithm to do parameter estimation.

2.2 Stray Light Reduction

Once the model parameters are estimated, we use Van Cittert's method[9,10] to compute the restored image. Van Cittert's method is an iterative approach to restore images. The iteration formula is as follows:

$$\hat{x}^{(k+1)} = \hat{x}^{(k)} + y - A\hat{x}^{(k)}, \tag{5}$$

where $\hat{x}^{(k)}$ is the estimate of the original image at the kth iteration, $A = ((1-\beta)G + \beta S)$, and we initialize $\hat{x}^{(0)}$ with the observed image y. We can approximate the diffraction and aberration PSF with a delta function,[2] leading to $G = I$. It has been shown that the result of the first iteration is already a good estimate of the original image.[2] Therefore, our restoration equation becomes:

$$\hat{x} = 2y - (1-\beta)y - \beta Sy. \tag{6}$$

However, since our stray light PSF is global and space-varying, directly computing Sy is quite expensive. For a 6 Megapixel image, it requires 6 million multiplies per output pixel, and a total of 3.6×10^{13} multiplies. So we compress the transform matrix S using matrix source coding to reduce the on-line computation.

3. MATRIX SOURCE CODING

3.1 Matrix Source Coding Theory

The computationally expensive part in Eq. (6) is

$$\tilde{x} = Sy. \tag{7}$$

Our strategy for speeding up computation is to compress the matrix S, such that it becomes sparse. In order to find a sparse representation of the matrix S, we use the techniques of lossy source coding. Let $[S]$ represent the quantized version of S. Then $S = [S] + \delta S$, where δS is the quantization error. This error δS results in some distortion in $\tilde{x}$ given by

$$\delta\tilde{x} = \delta Sy. \tag{8}$$

A conventional distortion metric for the source coding of the matrix S is $\|\delta S\|^2$. However, this may be very different from the squared error distortion in the matrix-vector product result which is given by $\|\delta\tilde{x}\|^2$. Therefore, we would like to relate the distortion metric for S to the quantity $\|\delta\tilde{x}\|^2$.

It has been shown that if the image y and the quantization error δS are independent, then the distortion in $\tilde{x}$ is given by[6,11]

$$\mathrm{E}\left[\|\delta\tilde{x}\|^2|\delta S\right] = \|\delta S\|^2_{R_y} = \mathrm{trace}\{\delta S R_y \delta S^t\}, \tag{9}$$

where $R_y = \mathrm{E}[yy^t]$. So if the data vector y is white, then $R_y = I$, and

$$\mathrm{E}\left[\|\delta\tilde{x}\|^2|\delta S\right] = \|\delta S\|^2. \tag{10}$$

In other words, when the data is white, minimizing the squared error distortion of the matrix S is equivalent to minimizing the expected value of the squared error distortion for the restored image $\hat{x}$.

So our strategy for source coding of matrix S is to simultaneously:

- Whiten the image y, so that minimizing the distortion in the coded matrix is equivalent to minimizing the distortion in the restored image.
- Decorrelate the columns of S, so that they will code more efficiently after quantization.
- Decorrelate the rows of S, so that they will code more efficiently after quantization.

The above goals can be achieved by applying the following transformation

$$\tilde{S} = W_1 S T^{-1} \quad (11)$$

$$\tilde{y} = T y, \quad (12)$$

where T is a matrix that simultaneously whitens the components of y and decorrelates the columns of S, and W_1 is a matrix that approximately decorrelates the rows of S. In our case, W_1 is a wavelet transform, since wavelet transforms are known to be an approximation to the Karhunen-Loeve transform for stationary sources, and are commonly used as decorrelating transforms.[12] We form the matrix T by applying a wavelet transform to y followed by gain factors designed to normalize the variance of the wavelet coefficients. Specifically, $T = \Lambda_w^{-1/2} W_2$ and $T^{-1} = W_2^{-1} \Lambda_w^{1/2}$, where W_2 is also a wavelet transform, and $\Lambda_w^{-1/2}$ is a diagonal matrix of gain factors designed so that

$$\Lambda_w = \mathrm{E}[\mathrm{diag}(W_2 y y^t W_2^t)].$$

This matrix T approximately whitens and decorrelates the image y. The diagonal matrix Λ_w can be estimated from training images. We take the wavelet transform W_2 of these training images, compute the variance of wavelet coefficients in each band, and average over all images. In this way, we only have a single gain factor for each band of the wavelet transform.

Using the above matrix source coding technique, the computation of the space-varying convolution now becomes

$$\tilde{x} \approx W_1^{-1}[\tilde{S}]\tilde{y}, \quad (13)$$

where $[\tilde{S}]$ is the quantized version of $\tilde{S}$. Therefore, the on-line computation consists of two wavelet transforms and a sparse matrix vector multiply. We will show later that this technique results in huge savings in computation of space-varying convolution.

3.2 Efficient Implementation of Matrix Source Coding

In order to implement the fast space-varying convolution method of Eq. (13), 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. (13).

However, even though the computation of $[\tilde{S}]$ is performed off-line, rigorous evaluation of this sparse matrix is still too large for practical problems. For example, if the image is 16 Megapixel in size, then temporary storage of S requires approximately 1 Petabyte of memory. 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[13] for both W_2 and W_1. 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, the precomputation 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 $O(N^2)$, where N is the number of pixels in the image, because the operation requires that each entry of S be touched. In order to reduce this computation to order $O(N)$, 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 reduce the computation of the wavelet transform W_1. To do this, we first compute the wavelet transform of each row using W_2, and then we perform an initial quantization step. This procedure is expressed mathematically as

$$[\tilde{S}] \approx [W_1 [S W_2^t \Lambda_w^{1/2}]], \quad (14)$$

where the notation $[\cdots]$ denotes quantization. After the initial quantization, the resulting matrix $[SW_2^t\Lambda_w^{1/2}]$ is quite sparse. If the number of remaining nonzero entries in each row is K, then the computational complexity of the second wavelet transform W_1 is reduced to order $O(KN)$ rather than $O(N^2)$.

However, the problem still remains as to how to efficiently implement the first wavelet transform W_2, since naive application of W_2 to every row still requires $O(N^2)$ operations. The next section describes an approach for efficiently computing this first wavelet transform by only evaluating the significant coefficients. This can be done by using a top-down recursive method to only evaluate the wavelet coefficients that are likely to have absolute values substantially larger than zero.

3.2.1 Determining the Location of Significant Wavelet Coefficients

As discussed above, we need a method for computing only the significant wavelet coefficients in order to reduce the complexity of the wavelet transform W_2. To do this, we will first predict the location of the largest K entries in every row of the matrix $SW_2^t\Lambda_w^{1/2}$. Then we will present an approach for computing only those coefficients, without the need to compute all coefficients in the transform.

Since each row of S is actually an image, we will use the 2D index (i_q, j_q) to index a row S, and (i_p, j_p) to index a column of S. Using this notation, $S_{(i_q,j_q),(i_p,j_p)}$ represents the effect of the point source at position (i_p, j_p) on the output pixel location (i_q, j_q).

Figure 1(a) shows an example of the row of S displayed as an image, together with its wavelet transform in Fig. 1(b). The wavelet transform is then scaled by gain factors $\Lambda_w^{-1/2}$, and represented by the corresponding row of $SW_2^t\Lambda_w^{1/2}$. Finally, Fig. 1(c) shows the locations of the wavelet coefficients that are non-zero after quantization, and a circle in each band that is used to indicate the region containing all non-zero entries.

By identifying the regions containing non-zero wavelet coefficients, we can reduce computation. In practice, we determine these regions by randomly selecting rows of the matrix and determining the center and maximum radius necessary to capture all non-zero coefficients.

3.2.2 A Top-Down Approach for Computing Sparse Haar Wavelet Coefficients

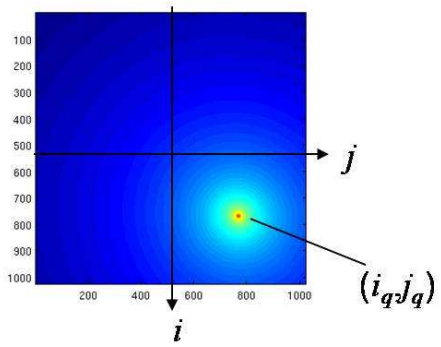
In order to efficiently compute sparse Haar wavelet coefficients, we compute only the necessary approximation coefficients at each level using a top-down tree-based approach. The tree structure is illustrated in Fig. 2. Since the approximation coefficients at different scales form a quad-tree, the value of each node can be computed from its children as

$$\begin{aligned} f_a[k][m][n] &= \frac{1}{2}(f_a[k-1][2m][2n] + f_a[k-1][2m+1][2n] \\ &\quad + f_a[k-1][2m][2n+1] + f_a[k-1][2m+1][2n+1]), \end{aligned} \tag{15}$$

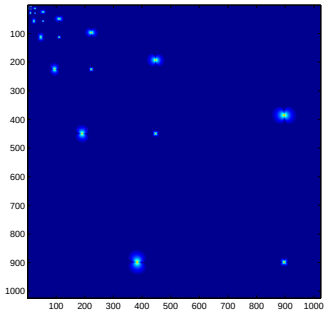
where $f_a[k][m][n]$ represents an approximation coefficient at level k and location (m, n). Here level $k = 0$ is the lowest level corresponding to the full resolution image. A recursive algorithm can be easily formulated from Eq. (15). We know that when the corresponding detail coefficients of an approximation coefficient are all zero, it means that the approximation is perfect. So in this case, for the nodes whose corresponding detail coefficients are all zeroed out after quantization, we stop and compute the value of the approximation coefficient by

$$f_a[k][m][n] = 2^k f_a[0][2^k m][2^k n]. \tag{16}$$

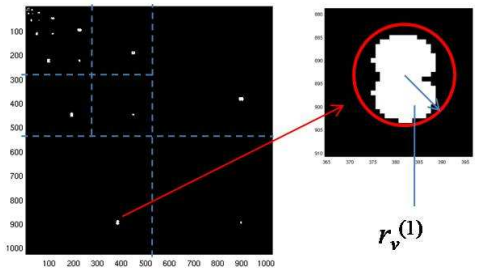
Note that $f_a[0][2^k m][2^k n]$ can be directly computed from the expression of the PSF. Another stopping point occurs when the leaves of the tree are reached, i.e. $k = 0$. Therefore, we have a recursive algorithm for computing the significant Haar wavelet approximation coefficients as shown in Fig. 3.



(a) An example of the row image.



(b) The corresponding wavelet transform.



(c) Location of non-zero wavelet coefficients after quantization.

Figure 1. An example of a row of S displayed as an image together with its wavelet transform, and the locations of non-zero entries after quantization: (a) shows the log amplitude map of the row image, (b) shows the log of the absolute value of its wavelet transform, (c) shows the locations of non-zero entries of the corresponding row of $SW_2^t\Lambda_w^{1/2}$.

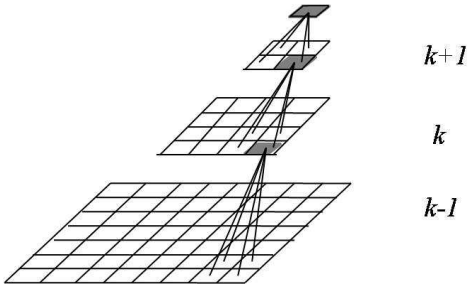


Figure 2. An illustration of the tree structure of approximation coefficients. Shaded squares indicate that the detail coefficients are not zero at those locations. So we go to the next level recursion. Empty squares indicate that the detail coefficients are zero, and we stop to compute their values.

After obtaining the significant approximation coefficients, we can compute the necessary wavelet coefficients[13] using the following equation:

$$
\begin{aligned}
f_d^h[k][m][n] &= \frac{1}{2}(f_a[k-1][2m][2n] - f_a[k-1][2m][2n+1] + f_a[k-1][2m+1][2n] - f_a[k-1][2m+1][2n+1]) \\
f_d^v[k][m][n] &= \frac{1}{2}(f_a[k-1][2m][2n] + f_a[k-1][2m][2n+1] - f_a[k-1][2m+1][2n] - f_a[k-1][2m+1][2n+1]) \\
f_d^d[k][m][n] &= \frac{1}{2}(f_a[k-1][2m][2n] - f_a[k-1][2m][2n+1] - f_a[k-1][2m+1][2n] + f_a[k-1][2m+1][2n+1]),
\end{aligned}
\tag{17}
$$

where $f_d^h[k][m][n]$, $f_d^v[k][m][n]$, and $f_d^d[k][m][n]$ denote the horizontal, vertical, and diagonal detail coefficients respectively, at level k, location (m, n). We can show that the complexity of our algorithm for computing

```
float FastApproxCoef(k, m, n) {
    float value = 0;
    if (k==0) {
        value = s(i_q, j_q; m, n);
        f_a[k][m][n] = value;
        return value;
    }
    if (ZeroDetailCoef(k,m,n)) {
        value = 2^k s(i_q, j_q; 2^k m, 2^k n);
        f_a[k][m][n] = value;
        return value;
    }
    value += FastApproxCoef(k-1, 2m, 2n);
    value += FastApproxCoef(k-1, 2m, 2n+1);
    value += FastApproxCoef(k-1, 2m+1, 2n);
    value += FastApproxCoef(k-1, 2m+1, 2n+1);
    value = value/2;
    f_a[k][m][n] = value;
    return value;
}
```

(a) Primary function.

```
int ZeroDetailCoef(k,m,n) {
    if (d((m,n),(m_c^k, n_c^k)) > max{r_h^(k), r_v^(k), r_d^(k)}) {
    /* (m_c^k, n_c^k) is the center of the circle at level k */
    /* d((m,n),(m_c^k, n_c^k)) is the distance from (m,n) to (m_c^k, n_c^k) */
        return 1;
    }
    else {
        return 0;
    }
}
```

(b) Subroutine.

Figure 3. Pseudo code for fast computation of significant Haar approximation coefficients. Part (a) is the main routine. Part (b) is a subroutine called by (a). The approximation coefficient at level k, location (m, n) is denoted $f_a[k][m][n]$.

sparse Haar wavelet coefficients of an N-pixel image is $O(K)$, where K represents the number of nonzero entries after quantization. The number of nodes that we visit in the tree of approximation coefficients is at most $4K$. Computing these $4K$ nodes requires at most $16K$ additions. So the complexity of computing the approximation tree is $O(K)$. Once the necessary approximation coefficients are computed, computing the detail coefficients only requires $O(K)$ operations. Thus the complexity of our algorithm is $O(K)$ for one row of S. Therefore, we reduce the complexity of precomputation of $[\tilde{S}]$ from $O(N^2)$ to $O(KN)$.

4. EXPERIMENTAL RESULTS

4.1 Experiments on Space-varying Convolution with Stray Light PSF

In order to test the performance of our algorithm in computing $\tilde{x} = Sy$, we use the test image shown in Fig. 4 for y, and a realistic stray light PSF for S. The parameters of this PSF 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. The parameters of this PSF were estimated for an Olympus SP-510 UZ* camera. We picked three natural images as training images to obtain the scaling matrix

*Olympus America Inc., Center Valley, PA 18034

Figure 4. Test image for space-varying convolution with stray light PSF.

Λ_w. We use normalized root mean square error (NRMSE) as the distortion metric:

$$\text{NRMSE} = \sqrt{\frac{\|\tilde{x} - W_1^{-1}[\tilde{S}]\tilde{y}\|_2^2}{\|\tilde{x}\|_2^2}} \ . \tag{18}$$

Our measure of computation is the average number of multiplies per output pixel (rate), which is given by the average number of non-zero entries in the sparse matrix $[\tilde{S}]$ divided by the total number of pixels N.

In order to better understand the computational savings, we used two different image sizes of 256×256 and 1024×1024 for our experiments. In the 256×256 case, we used an eight level Haar wavelet transform for W_2. In the 1024×1024 case, we used a ten level Haar wavelet transform for W_2. In both cases, W_1 was chosen to be a three level CDF 5-3 wavelet transform.[14] In addition, the wavelet transform W_1 was computed using a 64×64 block structure so as to minimize the size of temporary storage used in the off-line computation. Using a block structure enables us to compute the rows of $[SW_2^t \Lambda_w^{1/2}]$ corresponding to one block, take the wavelet transform along the columns, then quantize and save, and then go to the next block.

Figure 5(a) shows the distortion rate curves for a 256×256 image using two different techniques, i.e. simple quantization of S and matrix source coding of S as described by Eq. (13). Note that direct multiplication by S without quantization requires $65,536$ multiplies per output pixel.

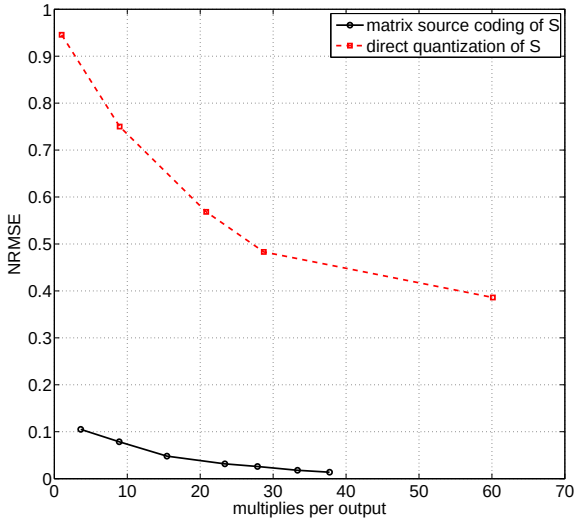
Figure 5(b) compares the distortion rate curves of our matrix source coding algorithm on two different image sizes — 256×256 and 1024×1024. Notice that the computation per output pixel actually *decreases* with the image resolution. This is interesting since the computation of direct multiplication by S *increases* linearly with the size of the image N.

At 1024×1024 resolution, with only 2% distortion, our algorithm reduces the computation from $1048576 = 1024^2$ multiplies per output pixel to 12 multiplies per output pixel; and that is a $87,381{:}1$ reduction in computation.

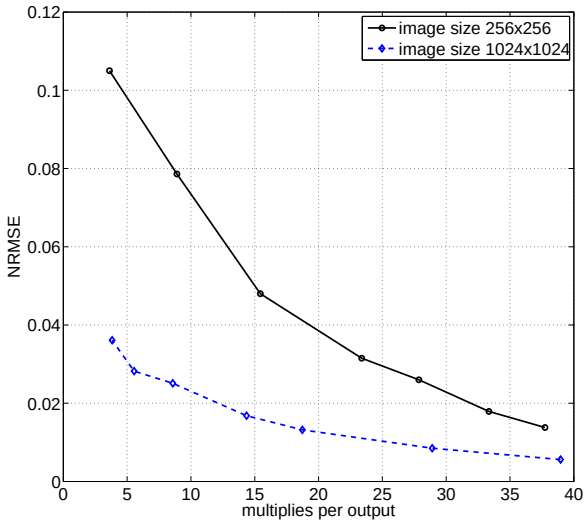
4.2 Experiment on Stray Light Reduction

We use Eq. (6) to perform stray light reduction. The space-varying convolution Sy is efficiently computed using our matrix source coding approach. We take outdoor images with an Olympus SP-510UZ camera. The stray light PSF parameters were shown in the previous subsection. The other parameter $\beta = 0.3937$. We use an eight level Haar wavelet for W_2, and a three level CDF 5-3 wavelet[14] for W_1, with block structure of block size 64×64.

Figure 4.2 shows an example of stray light reduction. Figure 4.2(a) is the captured image. Figure 4.2(b) is the restored image. Figures 4.2(c-d) shows a comparison between 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 more details of the original scene.



(a) Distortion vs. computation for two methods on image size 256×256.



(b) Distortion vs. computation for two image sizes using matrix source coding of S.

Figure 5. Experiments demonstrating fast space-varying convolution with stray light PSF: (a) relative distortion versus number of multiplies per output pixel (i.e. rate) using two different matrix source coding strategies: the black solid line shows the curve for our proposed matrix source coding algorithm as described in Eq. (13), and the red dashed line shows the curve resulting from direct quantization of the matrix S; (b) comparison of relative distortion versus computation between two different resolutions: the solid black line shows the curve for 256×256, and the dashed blue line shows the curve for 1024×1024.

(a) Captured image.

(b) Restored image.

(c) Captured | restored.

(d) Captured | restored.

Figure 6. Example of stray light reduction: (a) shows a captured image, (b) shows the restored image, (c) shows a comparison between captured and restored for a part of the image, (d) shows a comparison between captured and restored for another part of the image.

5. CONCLUSION

In this paper, we proposed a systematic approach for efficiently computing space-varying convolution using matrix source coding theory. Our approach reduced the computation from $O(N^2)$ to $O(N)$, where N is the number of pixels in the image. We also developed a fast algorithm for computing sparse entries of Haar wavelet transform to accelerate our encoding process, which involves precomputation of quantized wavelet transforms along the rows and columns of the convolution matrix. Our algorithm for computing the Haar wavelet transform uses 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 sparse entries of Haar wavelet coefficients with $O(K)$ complexity, where K is the number of sparse entries to compute. Finally, we showed experimental results of space-varying convolution which demonstrated the efficiency of our algorithm. We also showed a stray light reduction example using our fast algorithm.

REFERENCES

1. W. J. Smith, *Modern Optical Engineering : the Design of Optical Systems*, McGraw Hill, New York, NY (2000).
2. 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*, **6817**, (San Jose, CA) (2008).

3. P. A. Jansson and J. H. Fralinger, "Parallel processing network that corrects for light scattering in image scanners," *US Patent 5,153,926* (1992).
4. 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*, **6498**, (San Jose, CA) (2007).
5. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ (1975).
6. 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*, (2006).
7. P. A. Jansson, "Method, program, and apparatus for efficiently removing stray-flux effects by selected-ordinate image processing," *US Patent 6,829,393* (2004).
8. 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) (1998).
9. P. H. V. Cittert, "Zum einfluss der spaltbreite auf die intensitatswerteilung in spektrallinien," *Z. Physik* **69**, 298–308 (1931).
10. P. A. Jansson, *Deconvolution of Images and Spectra*, Academic Press, New York, NY (1996).
11. G. Cao, C. A. Bouman, and K. J. Webb, "Results in non-iterative map reconstruction for optical tomography," in *Proceedings of the SPIE/IS&T Conference on Computational Imaging VI*, **6814**, (San Jose, CA) (2008).
12. D. Tretter and C. A. Bouman, "Optimal transforms for multispectral and multilayer image coding," **4**(3), 296–308 (1995).
13. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, CA (1998).
14. M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. on Image Processing* **1**(2), 205–220 (1992).