Fast Iterative Image Restoration with a Spatially-Varying PSF

James G. Nagy[a] and Dianne P. O'Leary[b]

[a]Department of Mathematics,
Southern Methodist University, Dallas, TX 75275 USA

[b]Department of Computer Science and Institute for Advanced Computer Studies,
University of Maryland, College Park, MD 20742 USA

ABSTRACT

We describe how to efficiently apply a spatially-variant blurring operator using linear interpolation of measured point spread functions. Numerical experiments illustrate that substantially better resolution can be obtained at very little additional cost compared to piecewise constant interpolation.

Keywords: Conjugate gradient, image restoration, interpolation, preconditioner, space variant blur

1. Introduction

Obtaining medical and astronomical images is often quite expensive, so it is important to recover images degraded by imperfections in the imaging system or by environmental effects. Image restoration is the process of reconstructing the true image from the degraded one. Mathematically, the image formation process is modeled as

$$g(s) = \int_{\Omega} k(s,t) f(t) dt + \eta(s). \tag{1}$$

Here the spatial coordinates are $s \in \mathcal{R}^2$ and $t \in \mathcal{R}^2$, and Ω is a closed region containing the domain of the image. The function $g : \mathcal{R}^2 \to \mathcal{R}$ is the measured image, usually known only for certain discrete values of s. This function results from blurring the unknown true image f with the kernel k and then adding noise η. The number of measurements is finite—usually an $n \times n$ array of pixel values—so the model is discretized into a matrix equation

$$\mathbf{g} = \mathcal{K}\mathbf{f} + \mathbf{n}. \tag{2}$$

This type of integral equation (1) is known to be ill-posed, and the discretized matrix $\mathcal{K} \in \mathcal{R}^{n^2 \times n^2}$ is severely ill-conditioned. Specifically, the singular values of $\mathcal{K}$ decrease gradually and cluster at zero. Because of the presence of noise, solving $\mathcal{K}\mathbf{f} = \mathbf{g}$, does not yield an accurate restoration. *Regularization techniques* must be applied to stabilize the numerical methods. These techniques include Tikhonov regularization[1]; truncated value decomposition[2]; truncated iterative methods such as Landweber[3] and conjugate gradient (CG) iterations[4]; and mixed approaches.[5]

Because of the large dimensions involved in image restoration, iterative methods are usually the methods of choice. Typically, the most time consuming computations are matrix-vector products involving $\mathcal{K}$ and possibly $\mathcal{K}^T$.

Most often, the kernel k is assumed to be *spatially invariant*: $k(s,t) = k(s-t)$.[6,7] In this case, the integral equation is a convolution, and $\mathcal{K}$ is then a block Toeplitz matrix with Toeplitz blocks. Thus matrix-vector multiplication can be accomplished efficiently using fast Fourier transforms (FFT). The matrix $\mathcal{K}$ can be constructed from an idealized model of the true kernel or determined experimentally by imaging a single point source to determine the *point spread function* (PSF). There are situations, though, in which it is important to take account of the spatial variation of the point spread function.

Further author information -
J.G.N.: Email: jnagy@mail.smu.edu; URL: www.smu.edu/~jnagy; research supported by an NSF Postdoctoral Research Fellowship in the Mathematical Sciences
D.P.O.: Email: oleary@cs.umd.edu; URL: www.cs.umd.edu/~oleary; research supported by National Science Foundation Grant CCR 95-03126.

Spatially variant blurs occur in a variety of applications. The errors in the shaping mirrors of the original Hubble Space Telescope resulted in large spatial variation in the PSF, and despite the much improved imaging quality after the HST was fixed in 1994, there is still a need to exploit older images.[8] Two or more objects moving with different velocities relative to a recording device produce spatially-variant motion blurs.[9] Spatially variant blurs also occur when the object and image coordinates are tilted relative to each other, as well as in X-ray projection imaging,[10] positron emission tomography,[11,12] lens distortions,[13] wave aberrations,[13] and spatially varying Gaussian type blurs.[14] Moreover, it is unlikely that any blur is truly spatially invariant in any realistic application, especially over large image planes.[13] We now review some techniques that have been used to model spatially variant blurs.

If the blur is spatially invariant, then the kernel $k(s,t) = k(s-t)$ in the integral equation (1) gives rise to a matrix $\mathcal{K}$ in (2) that is block Toeplitz with Toeplitz blocks. Thus, for an $n \times n$ image, at most $O(n^2)$ storage and (using fast Fourier transforms) only $O(n^2 \log n)$ arithmetic operations are needed* to form matrix-vector products with $\mathcal{K}$.

If the blur is spatially variant, then $k(s,t)$, and hence $\mathcal{K}$, may not have any special structure. Thus, matrix vector multiplications with the $n^2 \times n^2$ matrix $\mathcal{K}$ will be extremely expensive, even for moderately sized images. One exception is if k is separable; that is, $k(s,t) = k_1(s)k_2(t)$. In this case, $\mathcal{K}$ is a Kronecker product of two matrices having dimensions $n \times n$. Thus, standard numerical linear algebra techniques, such as the singular value decomposition, become computationally feasible. Separable, spatially variant PSFs are considered by Angel and Jain.[14] Since this situation is not computationally difficult, our further discussions will assume the blur is not separable.

Among the earliest methods for restoring images degraded by (non-separable) spatially variant blurs is a geometrical coordinate transformation technique.[15–18] Essentially, this technique uses coordinate "distortions" or known symmetries to transform the spatially variant PSF into one that is spatially invariant. After applying a spatially invariant restoration method, another coordinate distortion is applied to obtain the result. Although this technique is useful for certain blurs such as rotational motion, it is not very practical for complicated blurs since the operator k, as well as the coordinate transformation functions, need to be known explicitly. Moreover, in general it is not possible to transform every spatially variant blur into one that is spatially invariant. Recently, McNown and Hunt[15] have developed a general technique (for one dimensional problems) to approximate the given spatially variant blur by one in which the coordinate transformations are known.

Another approach, which can treat a more general class of blurs, is based on the assumption that the blur is approximately spatially invariant in small regions of the image domain. These *sectioning methods*[19,20,9,21] partition the image, restoring each local region using its corresponding spatially invariant PSF, and the results are then sewn together to obtain the restored image. To reduce blocking artifacts at the region boundaries, larger, overlapping regions are used, and then the restored sections are extracted from their centers. Trussell and Hunt[9] proposed using the Landweber iteration for the local deblurring, and suggested a complicated stopping criteria based on a combination of local and global convergence constraints. Fish, Grochmalicki and Pike[20] use a truncated singular value decomposition (TSVD) to obtain the local restorations.

A third, more recent, approach is related to the sectioning methods in that the image is partitioned into subregions on which the blur is assumed to be spatially invariant. However, rather than deblurring the individual subregions locally and then sewing the individual results together, this method sews (interpolates) the individual PSFs, and restores the image globally. In algebraic terms, the blurring matrix $\mathcal{K}$ can be written as

$$\mathcal{K} = \sum_{i=1}^{p} \sum_{i=1}^{p} \mathcal{D}_{ij} \mathcal{K}_{ij}, \tag{3}$$

where $\mathcal{K}_{ij}$ is a block Toeplitz matrix with Toeplitz blocks representing the spatially invariant PSF in region (i,j), and $\mathcal{D}_{ij}$ is a nonnegative diagonal matrix satisfying $\sum\sum \mathcal{D}_{ij} = \mathcal{I}$. For example, if piecewise constant interpolation is used, then the lth diagonal entry of $\mathcal{D}_{ij}$ is 1 if the lth point is in region (i,j), and 0 otherwise.

Faisal et al.[22] use this formulation of the spatially variant PSF, apply the Richardson-Lucy algorithm with piecewise constant interpolation of the PSFs, and discuss a parallel implementation. Boden et al.[23] also describe

*For simplicity, operations counts assume that the argument of the log function is a power of two, perhaps by padding the original data. In practice, a general-radix FFT routine should be used to avoid padding whenever n has many factors.

a parallel implementation of the Richardson-Lucy algorithm, and consider piecewise constant as well as piecewise linear interpolation. Nagy and O'Leary[24] use a conjugate gradient algorithm with piecewise constant and linear interpolation, and also suggest a preconditioning scheme (for both interpolation methods) that can substantially improve rate of convergence.

Boden et al.[23] suggest that linear interpolation of the PSFs be rejected: for their data, they obtain roughly a factor of 340 decrease in throughput performance per iteration compared to piecewise constant interpolation, without much improvement in resolution. Nagy and O'Leary[24] also noted little improvement in resolution on their test problems. In this work, though, we carefully consider the implementation of linearly interpolated PSFs, and show that the cost is only modestly larger than the cost for constant interpolation. Moreover, we demonstrate that for some blurs, linear interpolation can produce significantly better restorations than piecewise constant interpolation.

2. Efficient Application of a Spatially-Variant PSF

In this section we provide a detailed description of the implementation and the computational cost of computing matrix-vector multiplications using the approximation to the spatially variant blur given in equation (3). We begin with the spatially invariant case and, in particular, implementations based on the *overlap-add* and *overlap-save* approaches. This discussion then leads to an obvious extension to applying a constant interpolated spatially variant blur, and only a slight further modification is needed for the linearly interpolated case.

2.1. Spatially Invariant Blurs

For spatially invariant blurs, the $n^2 \times n^2$ blurring matrix $\mathcal{K}$ is block Toeplitz with Toeplitz blocks, and it is well known that FFTs can be used to efficiently form the matrix vector products

$$\mathbf{y} = \mathcal{K}\mathbf{x} \quad \text{and} \quad \mathbf{y} = \mathcal{K}^T\mathbf{x}.$$

The nonzero entries of $\mathcal{K}$ are obtained from an image of the PSF, and $\mathbf{x}$ and $\mathbf{y}$ are obtained by stacking rows (or columns) of $n \times n$ image arrays X and Y.

The matrix-vector interpretation for applying the blurring operator is useful. However, for implementation, it is more convenient to consider obtaining Y by applying a two-dimensional convolution to X. Specifically, let P be an $(r+1) \times (r+1)$ array containing an image of the PSF, with $r \leq n$ and r even. Then the array Y (*i.e.,* the vector $\mathbf{y} = \mathcal{K}\mathbf{x}$) can be obtained by convolution of P with X, producing an $(n+r) \times (n+r)$ array, which we denote as $Y^{(n+r)}$. Y is then obtained by stripping off the r superfluous rows and columns of $Y^{(n+r)}$; that is,

$$Y^{(n+r)} = \begin{bmatrix} \times & \times & \times \\ \times & Y & \times \\ \times & \times & \times \end{bmatrix} = P * X.$$

Remarks on notation:

- $*$ is used to denote 2-D linear convolution, and $\circ$ will denote element-wise multiplication.
- $\times$ is used to denote superfluous rows and columns.
- Superscripts on arrays are used to emphasize that rows and columns need to be removed to obtain the base array (In this case, r rows and columns are removed from $Y^{(n+r)}$ to obtain the $n \times n$ base array Y).
- `fft2`$(\cdot)$ denotes a 2-D FFT of an array.
- `sfft2`$(\cdot)$ denotes a 2-D FFT of a shifted array. Specifically, `sfft2`$(\cdot)$ = `fft2(shift`$(\cdot)$`)`, where `shift` swaps the $(1,1)$ and $(2,2)$ blocks, and the $(1,2)$ and $(2,1)$ blocks of the array, putting the center of the point spread function in the upper left corner.

The standard approach to computing $Y^{(n+r)}$ is to use zero padding to first embed X and P into arrays of dimension $(n+r)\times(n+r)$, which, using our notation, are denoted as $X^{(n+r)}$ and $P^{(n+r)}$, respectively. Then $Y^{(n+r)}$ is computed using 2-D FFTs as

$$Y^{(n+r)} = \texttt{ifft2}(\texttt{sfft2}(P^{(n+r)}) \circ \texttt{fft2}(X^{(n+r)}))\ .$$

To assess computational cost here, and in the rest of this section, we assume that $cm^2 \log m$ arithmetic operations are needed to compute a 2-D FFT on an $m \times m$ array, where c is a constant of moderate size. Thus, to compute Y as described above, we require $3c(n+r)^2\log(n+r) + (n+r)^2$ arithmetic operations. We note, though, that $\texttt{sfft2}(P^{(n+r)})$ is fixed and thus needs to be computed only once, no matter how many matrix-vector products are formed. Therefore, the cost of applying a spatially invariant blurring operator using this standard approach is

$$(n+r)^2(2c\log(n+r)+1)\,.$$

Although X and P are real arrays, we need three complex arrays of dimension $(n+r)\times(n+r)$ to perform this operation. If the extent of the PSF is small compared to the dimension of the images (*i.e.*, $r \ll n$), the storage requirements can be substantially reduced using the *overlap-add* method or the *overlap-save* method.

2.1.1. Overlap-Add

Suppose P is an $(r+1)\times(r+1)$ array containing an image of the PSF, X is an $n \times n$ image, and $r \ll n$. Partition X as

$$X = \begin{bmatrix} \mathrm{X}_{11} & \mathrm{X}_{12} & \cdots & \mathrm{X}_{1p} \\ \mathrm{X}_{21} & \mathrm{X}_{22} & \cdots & \mathrm{X}_{2p} \\ \vdots & \vdots & & \vdots \\ \mathrm{X}_{p1} & \mathrm{X}_{p2} & \cdots & \mathrm{X}_{pp} \end{bmatrix},$$

where we assume each X_{ij} has dimension $s \times s$. Throughout this section we assume that s is even, and hence $s/2$ is an integer. Define M_{ij} to be the $n \times n$ *masking* array whose elements are 1 for components in the region corresponding to X_{ij}, and 0 otherwise. Then

$$X = \sum_{i=1}^{p}\sum_{j=1}^{p} M_{ij}\circ X,$$

and because convolution is a linear operation, it follows that

$$Y = P * X = \sum_{i=1}^{p}\sum_{j=1}^{p} P * (M_{ij}\circ X). \tag{4}$$

Observe that the nonzero entries in the convolution $P*(M_{ij}\circ X)$ can be computed from $P*\mathrm{X}_{ij}$. Since the dimensions of P and X_{ij} are much smaller than those of X, much less storage is required: P and X_{ij} can be embedded into arrays $\mathrm{P}^{(r+s)}$ and $\mathrm{X}_{ij}^{(r+s)}$ having dimensions $(r+s)\times(r+s)$, so the storage is $3(r+s)^2$ rather than $3(r+n)^2$.

We can efficiently accumulate the sum in equation (4) from the convolutions $\mathrm{Y}_{ij}^{(r+s)} = P*\mathrm{X}_{ij}$ of size $(r+s)\times(r+s)$. Those pixels outside the image domain are superfluous and can be discarded. Those that correspond to points in adjacent subregions within the image domain must be accumulated in the sum; hence the name "overlap-add". We summarize this discussion in the following algorithm.

Algorithm: Overlap-Add Convolution

for $i = 1, 2, \ldots, p$
  for $j = 1, 2, \ldots p$
    Extract region X_{ij} from X.
    Obtain $\mathrm{P}^{(r+s)}$ and $\mathrm{X}_{ij}^{(r+s)}$ by padding P and X_{ij} with zeros.
    Compute $\mathrm{Y}_{ij}^{(r+s)} = \texttt{ifft2}(\texttt{sfft2}(\mathrm{P}^{(r+s)}) \circ \texttt{fft2}(\mathrm{X}_{ij}^{(r+s)}))$.
    Add the relevant points into the sum Y .
  end
end

Note that $\texttt{sfft2}(\mathrm{P}^{(r+s)})$ needs to be computed only once, and not each time through the loop. To assess the cost, let $r+s=\ell s$. Then the computational cost of the overlap-add algorithm is

$$p^2\{2c(r+s)^2\log(r+s)+2(r+s)^2\}=2\ell^2n^2(c\log(r+s)+1)\,.$$

Although the total computational cost is slightly more than the standard approach described above, the overlap-add method has the significant advantage that storage can be kept at a reasonable level, even for large images. Moreover, this approach has obvious parallelism.

In matrix-vector notation, equation (4) can be written as

$$\mathbf{y}=\sum_{i=1}^{p}\sum_{j=1}^{p}\mathcal{K}\mathcal{D}_{ij}\mathbf{x},$$

where $\mathcal{D}_{ij}$ is a diagonal matrix whose diagonal entries correspond to the masking array M_{ij}: one if the corresponding element of $\mathbf{x}$ is in the (i,j) region, and zero otherwise. The matrix-vector form will be useful in establishing a relation to the spatially variant implementation.

2.1.2. Overlap-Save

In the overlap save method, instead of partitioning X, we partion the unknown result Y as

$$Y=\begin{bmatrix}\mathrm{Y}_{11} & \mathrm{Y}_{12} & \cdots & \mathrm{Y}_{1p}\\ \mathrm{Y}_{21} & \mathrm{Y}_{22} & \cdots & \mathrm{Y}_{2p}\\ \vdots & \vdots & & \vdots\\ \mathrm{Y}_{p1} & \mathrm{Y}_{p2} & \cdots & \mathrm{Y}_{pp}\end{bmatrix}.$$

For each i,j, we extract from X the smallest region that, when convolved with P, results in Y_{ij} exactly. In this case, we can write

$$Y=\sum_{i=1}^{p}\sum_{j=1}^{p}M_{ij}\circ Y=\sum_{i=1}^{p}\sum_{j=1}^{p}M_{ij}\circ(P*X). \tag{5}$$

Note that the nonzero portion of $M_{ij}\circ(P*X)$ is simply Y_{ij}.

To obtain Y_{ij}, we take an extended region around X_{ij}, which we denote as $\mathrm{X}_{ij}^{(r+s)}$, having dimension $(r+s)\times(r+s)$. That is, we take $r/2$ rows and columns surrounding X_{ij}, padding with zeros for regions on the boundary. The point source image P is embedded in an array of dimension $(r+s)\times(r+s)$, and 2-D FFTs are used for the convolution, from which Y_{ij} is extracted and saved into Y. We summarize this scheme in the following algorithm.

Algorithm: Overlap-Save Convolution

for $i=1,2,\ldots,p$
  for $j=1,2,\ldots p$
    Extract extended region $\mathrm{X}_{ij}^{(r+s)}$ from X.
    Obtain $\mathrm{P}^{(r+s)}$ by padding with zeros.
    Compute $\mathrm{Y}_{ij}^{(r+s)}=\texttt{ifft2}(\texttt{sfft2}(\mathrm{P}^{(r+s)})\circ\texttt{fft2}(\mathrm{X}_{ij}^{(r+s)}))$.
    Extract Y_{ij} from $\mathrm{Y}_{ij}^{(r+s)}$ and save into Y.
  end
end

As with the overlap-add algorithm, $\texttt{sfft2}(\mathrm{P}^{(r+s)})$ needs to be computed only once, not each time through the loop. The computational cost and storage are essentially the same as the overlap-add method.

In matrix-vector terms, equation (5) can be written as

$$\mathbf{y}=\sum_{i=1}^{p}\sum_{j=1}^{p}\mathcal{D}_{ij}\mathcal{K}\mathbf{x}.$$

2.2. Spatially Variant Blurs

Recall from (3) that we write our spatially-varying blurring matrix as

$$\mathcal{K} = \sum_{i=1}^{p}\sum_{j=1}^{p} \mathcal{D}_{ij}\mathcal{K}_{ij},$$

where $\mathcal{K}_{ij}$ is the block Toeplitz matrix associated with the PSF in region i, j, and $\mathcal{D}_{ij}$ is diagonal with $\sum\sum \mathcal{D}_{ij} = \mathcal{I}$. In this subsection we consider how to efficiently perform matrix vector products

$$\mathbf{y} = \mathcal{K}\mathbf{x} \quad \text{and} \quad \mathbf{y} = \mathcal{K}^T\mathbf{x}\,.$$

2.2.1. Piecewise Constant Interpolation

In the case of piecewise constant interpolation, the diagonal matrices $\mathcal{D}_{ij}$ are precisely those used in the above discussion. Therefore, to form the matrix-vector product:

$$\mathbf{y} = \mathcal{K}\mathbf{x} = \sum_{i=1}^{p}\sum_{j=1}^{p} \mathcal{D}_{ij}\mathcal{K}_{ij}\mathbf{x},$$

we can simply use a modification of the overlap-save method. The only difference is that we have separate PSFs, $\mathtt{P}_{ij}$, for each subregion. Using convolution notation, this matrix-vector product becomes

$$Y = \sum_{i=1}^{p}\sum_{j=1}^{p} M_{ij} \circ (\mathtt{P}_{ij} * X),$$

where the masking matrices M_{ij} are precisely those used above. Since each of the PSFs has narrow support, the dimension of $\mathtt{P}_{ij}$ will be small compared to n. Therefore, the procedure is the same as in the overlap-save method: we take the smallest extended region surrounding $\mathtt{X}_{ij}$ that allows $\mathtt{Y}_{ij}$ to be computed exactly, extract $\mathtt{Y}_{ij}$ from $\mathtt{Y}_{ij}^{(r+s)}$ and save into Y. We summarize this scheme in the following algorithm.

Algorithm: Matrix-vector multiplication $\mathbf{y} = \mathcal{K}\mathbf{x}$: Constant interpolation.

```
for i = 1, 2, ..., p
    for j = 1, 2, ...p
        Extract extended region X_ij^(r+s) from X.
        Obtain P_ij^(r+s) by padding with zeros.
        Compute Y_ij^(r+s) = ifft2(sfft2(P_ij^(r+s)) ∘ fft2(X_ij^(r+s))).
        Extract Y_ij from Y_ij^(r+s) and save into Y.
    end
end
```

Matrix-vector multiplication with $\mathcal{K}^T$ is similar:

$$\mathbf{y} = \mathcal{K}^T\mathbf{x} = \sum_{i=1}^{p}\sum_{j=1}^{p} \mathcal{K}_{ij}^T\mathcal{D}_{ij}\mathbf{x},$$

which clearly can be done through an appropriate modification of the overlap-add method. We summarize this in the following algorithm. Note that transposing the matrix $\mathcal{K}$ translates into conjugation of the spectrum of $\mathtt{P}_{ij}^{(r+s)}$.

Algorithm: Matrix-vector multiplication $\mathbf{y} = \mathcal{K}^T\mathbf{x}$: Constant interpolation.

```
for i = 1, 2, ..., p
    for j = 1, 2, ... p
        Extract region X_ij from X.
        Obtain P_ij^(r+s) and X_ij^(r+s) by padding with zeros.
        Compute Y_ij^(r+s) = ifft2(conj(sfft2(P_ij^(r+s))) ∘ fft2(X_ij^(r+s))).
        Accumulate sum.
    end
end
```

The computational cost for these algorithms is essentially the same as the overlap-save and overlap-add methods:

$$p^2\{2c(r+s)^2\log(r+s) + 2(r+s)^2\} = 2\ell^2 n^2 (c\log(r+s)+1)\,.$$

where the right hand side is obtained by assuming $r + s = \ell s$.

Precomputing $\texttt{sfft2}(\mathrm{P}_{ij}^{(r+s)})$ is more expensive than in the spatially invariant case, since there is a different PSF for each region. However, this only increases the setup cost, not the number of operations required to apply the operator. Storage requirements, though, can increase significantly, but the total amount of storage is on the same order as that needed in the standard approach to the spatially invariant blurring operator (cf. the beginning of Section 2.1).

2.2.2. Linear Interpolation

Suppose the spatially variant blurring operator is approximated by linear interpolation of PSFs in adjacent regions. In this case, the masking arrays M_{ij} specifying the main diagonal of $\mathcal{D}_{ij}$, are constructed as follows:

- The elements of M_{ij} are zero, except in a subregion of dimension $2s \times 2s$.
- The subregion where the elements are nonzero corresponds to the region $\mathrm{X}_{ij}^{(2s)}$; that is, the $s \times s$ region X_{ij} plus $s/2$ additional rows and columns surrounding X_{ij}. This is illustrated in Figure 1a, where $\bullet$ is used to indicate positions of PSFs, with P_{ij} in the center, and the box represents an $s \times s$ region corresponding to X_{ij}.
- The nonzero elements of M_{ij} are 1 in the center, and decrease linearly to 0 on the boundaries of the $2s \times 2s$ subregion. A mesh plot of a sample masking function is shown in Figure 1b.
- Regions on the boundary of the image domain have masking arrays appropriately modified so that the sum of the masking arrays is an array with every entry equal to 1.

The matrix-vector multiplication $\mathbf{y} = \mathcal{K}\mathbf{x}$ can be written in convolution notation as

$$Y = \sum_{i=1}^{p}\sum_{j=1}^{p} M_{ij} \circ (\mathrm{P}_{ij} * X). \tag{6}$$

Consider the computation of $M_{ij} \circ (\mathrm{P}_{ij} * X)$. Note that M_{ij} is nonzero only on a $2s \times 2s$ subarray corresponding to $\mathrm{X}_{ij}^{(2s)}$. We denote this part of M_{ij} as M_{ij}. By convolving P_{ij} with $\mathrm{X}_{ij}^{(2s+r)}$, we can obtain the $2s \times 2s$ subregion of $\mathrm{P}_{ij} * X$, which can then be masked with M_{ij}, and the appropriate region in the sum (6) accumulated. The following algorithm summarizes this discussion.

Algorithm: Matrix-vector multiplication $\mathbf{y} = \mathcal{K}\mathbf{x}$: Linear interpolation.

```
for i = 1, 2, ..., p
    for j = 1, 2, ... p
        Extract extended region X_ij^(2s+r) from X.
        Obtain P_ij^(2s+r) by padding with zeros.
        Compute Y_ij^(2s+r) = ifft2(sfft2(P_ij^(2s+r)) ∘ fft2(X_ij^(2s+r))).
        Extract Y_ij^(2s) from Y_ij^(2s+r).
        Mask the result M_ij ∘ Y_ij^(2s), and accumulate the sum.
    end
end
```

Note that M_{ij} and $\texttt{sfft2}(\mathrm{P}_{ij}^{(2s+r)})$ need only be computed once. The total cost for forming a matrix-vector multiplication using the above algorithm is

$$p^2(2c(2s+r)^2\log(2s+r) + (2s+r)^2) + 2n^2 \approx 2(\ell+1)^2 n^2 c \log(\ell+1)s + (\ell+1)^2 n^2 + 2n^2\,.$$

The transpose multiplication can be efficiently implemented by making a similar modification to the overlap-add method. We mention that, as with piecewise constant interpolation, transposing the matrix $\mathcal{K}$ translates into conjugating the spectrum of $\mathrm{P}_{ij}^{(2s+r)}$.

Algorithm: Matrix-vector multiplication $\mathbf{y} = \mathcal{K}^T\mathbf{x}$: Linear interpolation.

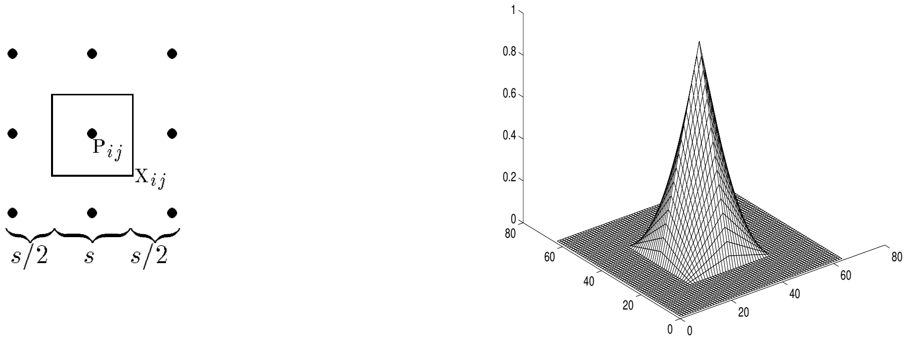
```
for i = 1, 2, ..., p
    for j = 1, 2, ... p
        Extract extended region X_ij^(2s) from X.
        Mask and overwrite: X_ij^(2s) = M_ij ∘ X_ij^(2s).
        Obtain P_ij^(2s+r) and X_ij^(2s+r) by padding with zeros.
        Compute Y_ij^(2s+r) = ifft2(conj(sfft2(P_ij^(2s+r))) ∘ fft2(X_ij^(2s+r))).
        Accumulate the sum.
    end
end
```

The computational cost is essentially the same as the algorithm for computing $\mathbf{y} = \mathcal{K}\mathbf{x}$.

3. Numerical Experiments

In this section we present some numerical results to demonstrate that using multiple PSFs can substantially improve restorations of images degraded by spatially varying blurs. Moreover, it is observed that linear interpolation can be much better than piecewise constant interpolation.

One of the difficulties in attempting to test and compare restoration methods for spatially varying blurs is setting up fair test problems. One particular set of data that has been used for Hubble Space Telescope applications is a simulated star cluster image.[23,22,24] (This data was obtained via anonymous ftp from `ftp.stsci.edu` in the directory `/software/stsdas/testdata/restore/sims/star_cluster`.) It has been shown in[23,22,24] that piecewise constant interpolation of multiple PSFs can provide better restorations than using a single PSF. We have found that for this particular data set, linear interpolation produces better solutions than constant interpolation, but the difference is not that great.



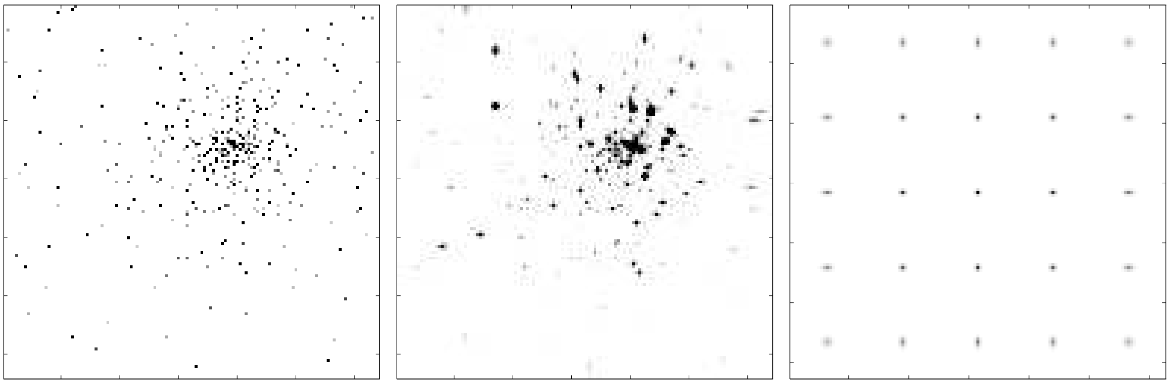
a. Sample region. b. Mesh plot of sample mask.

Figure 1. a. Region on which the mask M_{ij} is nonzero for linearly interpolated PSFs. Each $\bullet$ indicates the position of a PSF, with P_{ij} in the center, and the box represents the $s \times s$ region corresponding to X_{ij}. b. Mesh plot of sample masking array M_{ij} with $n = 64$ and $s = 16$. The masking array defines the matrix $\mathcal{D}_{ij}$.

To construct a test example, we take a 128×128 portion of the simulated star cluster image (obtained from the abovementioned ftp site) shown in Figure 2a. This "true" image is then degraded by a separable spatially variant Gaussian blur,[14] $\mathcal{K} = A \otimes A$, with entries generated as

$$a_{ij} = \exp(-\gamma_i (i-j)^2), \quad \gamma_i = 2 - \frac{19|64-i|}{640}, \quad i,j = 1,2,\ldots,128,$$

and scaled so that $A\mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ is the vector whose entries are all ones. 1% white noise was added to the blurred image, resulting in a signal-to-noise ratio of $10\log_{10}(||\mathcal{K}\mathbf{f}||_2/||\mathbf{n}||_2) = 20dB$. The degraded image is shown in Figure 2b. Special techniques could be used to exploit the separable structure of $\mathcal{K}$, but we concentrate here on comparing constant, piecewise constant, and piecewise linear approximations to the kernel. Point spread functions, with $r+1 = 25$, were created by blurring point sources in various regions of the image, as shown in Figure 2c. As can be seen a large amount of spatial variation occurs in the blurring.



a. Original image. b. Blurred image. c. Point spread functions.

Figure 2. True and blurred images.

A conjugate gradient algorithm (cf. Björck[25]) is used as an iterative image restoration method. To illustrate the difference in using various PSFs and interpolation schemes to approximate $\mathcal{K}$, we computed relative errors:

$$\frac{||\mathbf{f} - \mathbf{f}^{(j)}||}{||\mathbf{f}||},$$

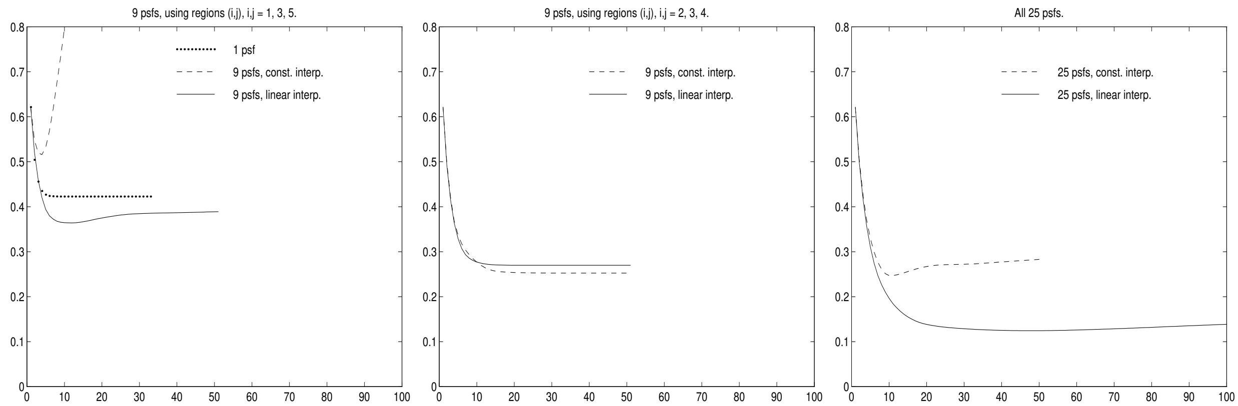
where $\mathbf{f}$ is the true solution, and $\mathbf{f}^{(j)}$ is the solution at the jth iteration of the conjugate gradient method. Figure 3 is a plot of these errors. In Figure 3a, we plot the errors using a single PSF taken from the center of the image domain (dotted curve) with the errors using nine PSFs. These nine PSFs were taken from regions (1,1), (1,3), (1,5), (3,1), (3,3), (3,5), (5,1), (5,3), and (5,5). (Regions are numbered as entries in a matrix would be; for example, (1,1) is the upper-left region.) As can be seen, piecewise constant interpolation (dashed curve) performed very poorly. This is probably due to the fact that the PSFs furthest from the center of the image are relatively smooth, and hence the corresponding $\mathcal{K}_i$ are more ill-conditioned than those near the center of the image. As a result, $\mathcal{K}$ is dominated by the ill-conditioned matrices corresponding to the PSFs in the outer regions.

However, we do see that linear interpolation significantly improves the accuracy of the solutions. In particular, Figure 3a (solid line) shows that the errors are much reduced compared to constant interpolation of these PSFs. Moreover, linear interpolation of these 9 PSFs provides better solutions than using only one PSF.

Because the PSFs in the outer regions of the image domain are more ill-conditioned, we performed the same experiment using nine PSFs in the regions closest to the center; that is, in regions (2,2), (2,3), (2,4), (3,2), (3,3), (3,4), (4,2), (4,3), (4,4). The relative errors at each iteration for this case are shown in Figure 3b, where the dashed line corresponds to using constant interpolation, and the solid line to linear interpolation. Although constant interpolation produces smaller relative errors, the difference is not significant. However, by comparing Figures 3a and b, we can see that this choice of 9 PSFs provides significantly lower errors than the 9 taken from the outer regions of the image.

In Figure 3c we plot the errors using all 25 PSFs. For constant interpolation (dashed curve), including the ill-conditioned PSFs in the outer regions is not helpful. However, linear interpolation of the 25 PSFs yielded significantly smaller errors.

Figures 4 — 6 show the best solutions computed using each of these approximate kernel functions.

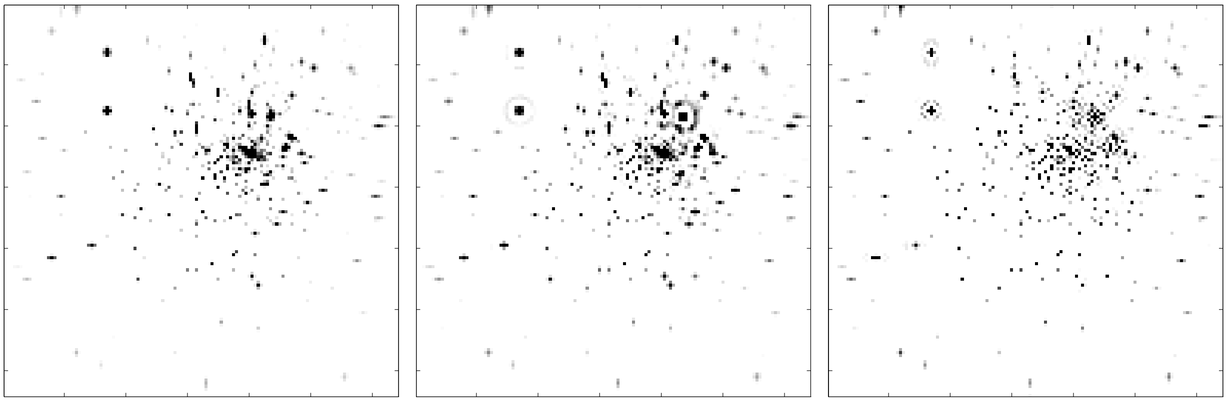


a. 9 PSFs, outer regions. b. 9 PSFs, inner regions. c. 25 PSFs.

Figure 3. Plots of relative errors at each iteration.

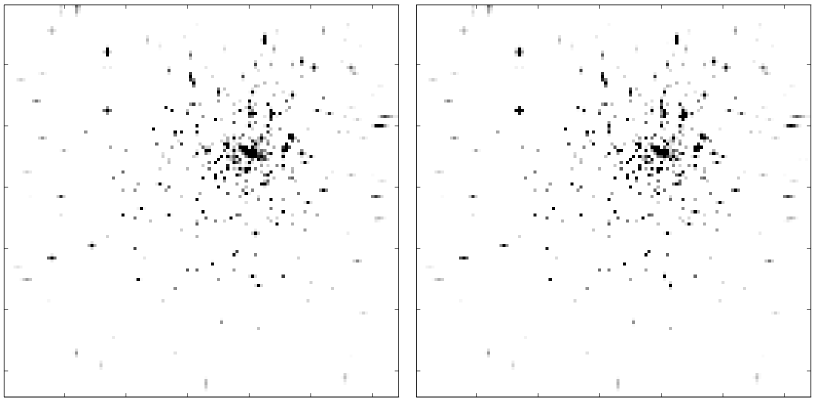
4. Concluding Remarks

We have shown that application of a spatially-variant blurring operator is closely related to the overlap-add and overlap-save convolution schemes for spatially invariant blurs, and exploiting this relationship results in efficient FFT-based algorithms. A detailed analysis showed that the additional cost for using linear interpolation of the point spread functions is only modestly larger than for constant interpolation. Moreover, numerical experiments illustrated that substantially better resolution of iterative image restoration methods can be obtained using linear interpolation, especially if the blur changes dramatically from one region of the image to another.



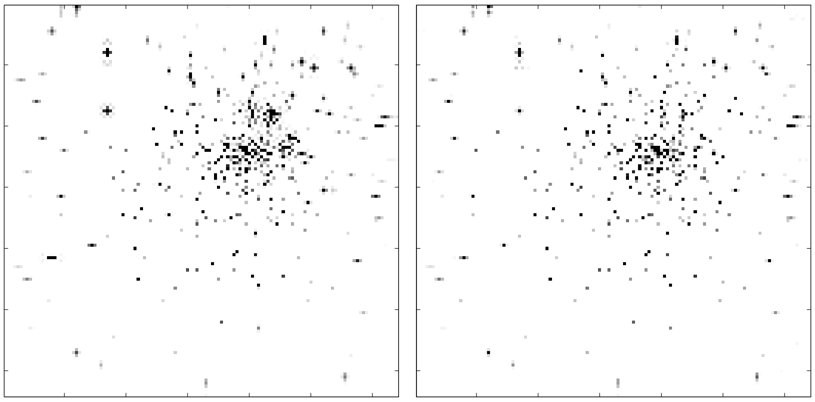
a. 1 PSF b. 9 (outer) PSFs, cons. interp. c. 9 (outer) PSFs, lin. interp.

Figure 4. Computed restoration using 1 PSF, and the 9 PSFs from the outer regions of the image domain.



a. 9 (inner) PSFs, cons. interp. b. 9 (inner) PSFs, lin. interp.

Figure 5. Computed restoration using the 9 PSFs from the inner regions of the image domain.



a. 25 PSFs, cons. interp. b. 25 PSFs, lin. interp.

Figure 6. Computed restoration using all 25 PSFs.

REFERENCES

1. C. W. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Integral Equations of the First Kind*, Pitman, Boston, 1984.
2. P. C. Hansen, "The truncated SVD as a method for regularization," *BIT* **27**, pp. 534–553, 1987.
3. M. Hanke and P. C. Hansen, "Regularization methods for large-scale problems," *Surv. Math. Ind.* **3**, pp. 253–315, 1993.
4. M. Hanke, *Conjugate Gradient Type Methods for Ill-Posed Problems*, Pitman Research Notes in Mathematics, Longman Scientific & Technical, Harlow, Essex, 1995.
5. D. P. O'Leary and J. A. Simmons, "A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems," *SIAM J. Sci. Stat. Comp.* **2**, pp. 474–489, 1981.
6. H. Andrews and B. Hunt, *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ, 1977.
7. R. L. Lagendijk and J. Biemond, *Iterative Identification and Restoration of Images*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.
8. H. M. Adorf, "Hubble Space Telescope image restoration in its fourth year," *Inverse Problems* **11**, pp. 639–653, 1995.
9. H. J. Trussell and S. Fogel, "Identification and restoration of spatially variant motion blurs in sequential images," *IEEE Trans. Image Proc.* **1**, pp. 123–126, 1992.
10. F. C. Wagner, A. Macovski, and D. G. Nishimura, "A characterization fo the scatter point-spread-functions in terms of air gaps," *IEEE Trans. Med. Imaging* **7**, pp. 337–344, 1988.
11. B. Karuta and R. Lecomte, "Effect of detector weighting functions on the point spread function of high-resolution pet tomographs: A simulation study," *IEEE Trans. Med. Imaging* **11**, pp. 379–385, 1992.
12. S. Rathee, A. J. Koles, and T. R. Overton, "Image restoration in computed tomography: Estimation of the spatially variant point spread function," *IEEE Trans. Med. Imaging* **11**, pp. 539–545, 1992.
13. A. W. Lohmann and D. P. Paris, "Space-variant image formation," *J. Opt. Soc. Am.* **55**, pp. 1007–1013, 1965.
14. E. S. Angel and A. K. Jain, "Restoration of images degraded by spatially varying pointspread funcitons by a conjugate gradient method," *Applied Optics* **17**, pp. 2186–2190, 1978.
15. S. R. McNown and B. R. Hunt, "Approximate shift-invariance by warping shift-variant systems," in *The Restoration of HST Images and Spectra II*, R. J. Hanisch and R. L. White, eds., pp. 181–187, 1994.
16. G. M. Robbins and T. S. Huang, "Inverse filtering for linear shift-variant imaging systems," *Proc. of the IEEE* **60**, pp. 862–872, 1972.
17. A. A. Sawchuk, "Space-variant image motion degradation and restoration," *Proc. of the IEEE* **60**, pp. 854–861, 1972.
18. A. A. Sawchuk, "Space-variant image restoration by coordinate transformations," *J. Opt. Soc. Am.* **64**, pp. 138–144, 1974.
19. H.-M. Adorf, "Towards HST restoration with space-variant PSF, cosmic rays and other missing data," in *The Restoration of HST Images and Spectra II*, R. J. Hanisch and R. L. White, eds., pp. 72–78, 1994.
20. D. A. Fish, J. Grochmalicki, and E. R. Pike, "Scanning singular-value-decomposition method for restoration of images with space-variant blur," *J. Opt. Soc. Am. A* **13**, pp. 1–6, 1996.
21. H. J. Trussell and B. R. Hunt, "Image restoration of space-variant blurs by sectional methods," *IEEE Trans. Acoust. Speech, Signal Processing* **26**, pp. 608–609, 1978.
22. M. Faisal, A. D. Lanterman, D. L. Snyder, and R. L. White, "Implementation of a modified Richardson-Lucy method for image restoration on a massively parallel computer to compensate for space-variant point spread function of a charge-coupled device camera," *J. Opt. Soc. Am. A* **12**, pp. 2593–2603, 1995.
23. A. F. Boden, D. C. Redding, R. J. Hanisch, and J. Mo, "Massively parallel spatially-variant maximum liklihood restoration of Hubble Space Telescope imagery," *J. Opt. Soc. Am. A* **13**, pp. 1537–1545, 1996.
24. J. G. Nagy and D. P. O'Leary, "Restoring images degraded by spatially-variant blur," *SIAM J. Sci. Comput.* **to appear**, 1997.
25. Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.