BLOCK ITERATIVE RESTORATION OF ASTRONOMICAL IMAGES WITH THE MASSIVELY PARALLEL PROCESSOR

Sara R. Heap
Laboratory for Astronomy and Solar Physics
Goddard Space Flight Center

Don J. Lindler
Advanced Computer Concepts
Potomac Maryland

ABSTRACT

We describe a method for algebraic image restoration capable of treating astronomical images. For a typical 500 x 500 image, direct algebraic restoration would require the solution of a 250,000 x 250,000 linear system. We use the block iterative approach to reduce the problem to solving 4900 121x121 linear sytems. We have implemented the algorithm on the Goddard Massively Parallel Processor, which can solve a 121 x 121 system in approximately 0.06 seconds. Here, we show examples of our results for various astronomical images.

Keywords: image restoration, constrained least-squares, block iteration

1. INTRODUCTION

The discrete model of linear image degradation is specified by the equation:

$$b = Hx + n \qquad (1)$$

where b and x are the pixel values of the degraded and original undegraded images stacked into column vectors, H is a matrix constructed from the impulse response (or point spread function) of the degradation, and n is an unknown additive noise vector. The object of restoration is to determine x, given b and possibly information on the properties of n. If the point spread function used to construct H is not known for the given optical-detector configuration, it must be estimated from the blurred image b. The point spread function is most easily estimated from point sources (i.e. stars) on the blurred image.

Since H may be ill-conditioned or singular, and only the statistical properties of the noise are known, there are many solutions, $\hat{x}$, for x which satisfy equation (1). In order to obtain a reasonable solution for x, it is necessary to utilize properties of x. The success of the restoration will therefore depend on the ability to model and apply to the restoration, known or assumed properties of the desired solution. Properties of x may consist of its "smoothness" or the restriction that all values in x be positive.

Some advantages of algebraic image restoration are:

1) The point spread function may be spatially variant;
2) If a constrained least squares method is used, the applied constraints may be varied from pixel to pixel to make maximum use of the known image properties;
3) Missing or bad pixel values in the blurred images can be easily handled without directly attempting to repair their values;
4) Noise properties can vary from pixel to pixel.

The main disadvantage of algebraic image restoration is the size of the linear system. For a 500 x 500 pixel image, H is a 250,000 x 250,000 matrix. Even with the most powerful computers available (including the MPP), a direct solution of the system would be impossible. In the next section, we describe a technique -- the block iterative method, of solving large linear systems.

2. THE BLOCK ITERATIVE RESTORATION ALGORITHM

2.1 Block Jacobi Iteration

In most astronomical images, the point spread function has a much smaller spatial extent than the image, so it is appropriate to work on the image locally. We therefore divide the image into blocks and restore each block separately, using values from the previous iteration as estimates of the unblurred image values outside the block. In most instances the blurred image is a good choice for the starting or zeroth iteration. This type of iteration is called block Jacobi or group Jacobi (Young 1971) iteration and can be formulated in matrix notation as follows.

Consider the blurred image, b, divided into m blocks of equal size B_i, i = 1,m.

$$B = \begin{vmatrix} B_1 & B_2 & \cdots & \\ & B_{i-1} & B_i & B_{i+1} \\ & & \cdots & \\ & & B_{m-1} & B_m \end{vmatrix}$$

Stack the elements of each block and place them into a vector:

$$B = \begin{vmatrix} B_1 \\ B_2 \\ \cdots \\ B_m \end{vmatrix} .$$

Ignoring the noise for now, we write the system as:

$$HX = B \ ,$$

where H is partitioned into blocks:

$$H = \begin{vmatrix} H_{11} & H_{12} & \cdots & H_{1m} \\ H_{21} & H_{22} & \cdots & H_{2m} \\ & \cdots & & \\ H_{m1} & H_{m2} & \cdots & H_{mm} \end{vmatrix} ,$$

and X contains the restored values, blocked in the same manner as B. If the image were divided into blocks of n pixels each, then the blocks H_{ij} would have size n x n. The block Jacobi method can now be written as:

$$H_{ii} X_i^{r+1} = B_i - \sum_{\substack{j=1 \\ j \neq i}}^{m} H_{ij} X_j^r \ , \qquad (2)$$

i=1,...,m, and where X_i^r is the stacked values for iteration r of block i. If we define the vector on the right hand side of equation (2) as $BMOD_i$ (i.e. the blurred image less contributions from outside the block as estimated from the previous iteration of the undegraded image), the linear system for block i can now be written as:

$$H_{ii} X_i^{r+1} = BMOD_i \ . \qquad (3)$$

Using the block Jacobi method, we can reduce the problem to solving m smaller systems of size n x n of the form:

$$H x = b \qquad (4)$$

where H is H_{ii} for block i; x is X_i^{r+1} for block i, iteration r+1; and b is $BMOD_i$ for block i.

The solution for block i now requires the solution of an n x n linear system. For example, to restore a 100 x 100 pixel image divided into m=100 blocks, each of size, nxn=10x10, the largest system to be solved would have H_{ii} of size 100 x 100. Since solutions of linear systems require on the order of n^3 operations, the block approach compares favorably to the direct solution of the 10,000 x 10,000 system. For a spatially invariant point spread function, the problem is further reduced since H_{ii} will be identical for all i=1,m.

If a constrained least squares approach is used to solve the linear system, the solution will converge to acceptable results even with a block size as small as the full-width-at-half- maximum (FWHM) of the point spread function. Overlapping the blocks (accepting only the central portion for the next iteration) gives faster convergence or may produce convergence when no overlap results in divergence.

2.2 Image Constraints

The block Jacobi method reduces the restoration to solution of many smaller linear systems, but it does not address the ill-conditioned nature of H or the presence of noise in the blurred image. An ill-conditioned matrix means small changes in b, caused by noise, yield large changes in the solution $x = H^{-1} b$. In this section, we show how constrained solutions can handle these problems.

In most images, the data vary smoothly except at isolated points or edges. For example, an image of a star field will vary smoothly, except at locations of individual stars. We can make use of this image property, smoothness, by applying a constrained least squares fit. Specifically, we minimize a linear operator $\|Qx\|$ (i.e. The sum of the squares in Qx), where Q is a matrix designed to control smoothness or other characteristics of the image (Twomey 1963, Philips 1962). For example, we can control smoothness in the one dimensional case by minimizing the second difference in the solution subject to some other constraint. If the statistical properties of the noise are known, we could minimize the second difference such that the norm of $\|Hx-b\| = \|n\|$; that is to say, the difference of the blurred image and the solution reconvolved with the point spread function should have the same properties as the noise. In this case (minimize the second difference), Q would have the form:

$$Q = \begin{vmatrix} 0 & 0 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & 0 & & \\ & & \cdots & & & & \\ & & & & -1 & 2 & -1 \\ & & & & & 0 & 0 \end{vmatrix}$$

We use the method of Lagrangian multipliers, sometimes called the method of undetermined multipliers, to make the constrained least-squares fit. The solution of x is then given by (Andrews 1977):

$$x = (H^T H + \gamma Q^T Q)^{-1} H^T b$$

where γ is the reciprocal Lagrangian multiplier. The value of γ can be iteratively selected to control the amount of constraint in the solution. The solution using Lagrangian multipliers place no restrictions on the form of Q. This flexibility allows the development of a variety of constraints depending on the known properties of the image.

Figure 1 shows the application of this constrained least squares filter for a test case with different values of γ_2. The subscript 2 is used to

indicate that the constraint is the minimum second difference. Note in figure 1.C-1.E that as γ_2 increases, noise in the solution decreases, but "ringing" at the edges increases. The ringing results from an inappropriate constraint at edges: the second difference should have a large value at an edge and should *not* be minimized. We therefore minimize the second difference at every location *except* the edges by setting the rows of Q corresponding to the edge locations to zeros. Figure 1.F shows a restoration of the same test image when the second difference constraint is not applied at the edges. A significant improvement results.

A direct extension of the method to two dimensional images is to minimize the Laplacian at each point. The Laplacian operator has a value at each pixel equal to four times the pixel value minus the values of the four immediate neighboring pixels. We use the subscript, *1*, to indicate the presence of the Laplacian constraint. As before, we set rows of the matrix Q to zero when the Laplacian constraint is not appropriate (i.e. edges or point sources).

The constraint need not be binary: we can vary the amount of constraint between no constraint to full constraint for any pixel, simply by multiplying the appropriate row in Q by a constant factor running from 0 to 1.

Another useful constraint is to minimize the difference of x from a trial solution (i.e. minimize $||p-x||$). The solution using Lagrangian multipliers is given by (Twomey 1963):

$$x = (H^T H + \gamma_t I)^{-1} (H^T b + \gamma_t p)$$

where p is the trial solution, I is the identity matrix, and γ_t is the reciprocal Lagrangian multiplier. The subscript, t, will be used to identify

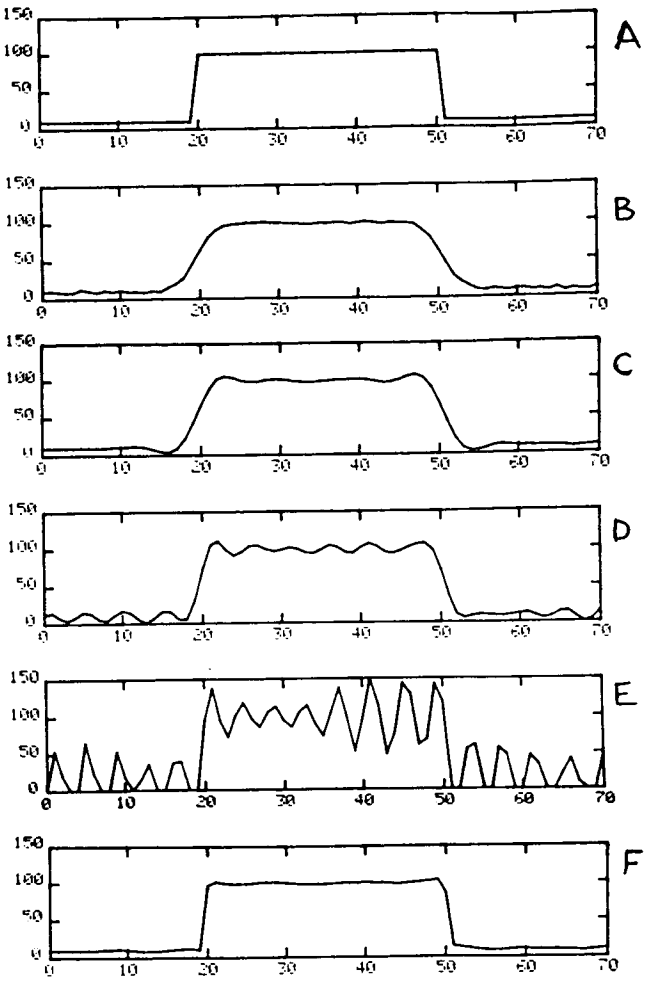

Figure 1. Effect of Langrangian multipliers. (A) original image; (B) image blurred with a Gaussian PSF (σ=2.0 pixels) and noise added (σ=1 DN); (C) restoration with γ_2=0.1; (D) restoration with γ_2=0.001; (E) restoration with γ_2=0.0001; (F) restoration with γ_2=0.1 with constraint removed at the two edges.

the constraint as minimization of the solution from a trial solution. Some possible choices for the trial solution, p, are a constant value (i.e. all zeros) or the blurred image itself. In either case, the ill-conditioned nature of H can be avoided and reasonable solutions obtained.

Multiple image constraints can be applied simultaneously:

$$x = (H^T H + \gamma_a Q_a^T Q_a + \gamma_b Q_b^T Q_b + \ldots + \gamma_t I)^{-1} (H^T b + \gamma_t p)$$

where a different value of γ can be selected for each constraint.

Selection of the reciprocal Lagragian multipliers is done by trial and error with the evaluation of γ by visual inspection of the results for various values of γ or by examination of the difference of blurred image and the solution reconvoled with the point spread function. This difference should have the same properties as the noise.

2.3 Missing or Bad Data Values

A problem results when trying to restore images with missing or bad data values (i.e. cosmic ray hits or bad CCD columns). If they are not taken into account in the restoration, the bad values will propagate to a larger portion of the output solution. To some extent, every point in the solution depends on all other values in the blurred image.

One method of handling bad pixels is to attempt to repair them before restoration by interpolating from neighboring values. This approach is successful only if the repair is accurate. An alternative method is to make no attempt at prior repair but handle them in the restoration process. In this approach, the restored image will have more data values than the blurred image, and the linear system is underdetermined and, therefore, singular (i.e. no direct inverse exists). To ignore bad data values, we set their corresponding rows in matrix H to zeros. This method of implementation (as opposed to removing row H creating a non-square underdeterminimed system) allows us to keep the matrix H square and decrease the complexity of implementation. Keeping H square in no way alleviates the problem of singularity. However, using the constrained least squares solution, the problem of singularity can be alleviated and reasonable solutions obtained.

3. IMPLEMENTATION OF THE ALGORITHM

The procedure for block iterative restoration described in section 2 is actually carried out over three computers. We use our laboratory VAX 750 with Gould DeAnza image display for interactive analysis of the blurred and restored images. We then use a local area network to copy the blurred image and point-spread function from the laboratory computer over to the MPP VAX 780 host computer. We use the latter machine to prepare input to the MPP, invoke the MPP, and reconstruct the output from the MPP into restored images. Preparation mainly involves dividing the various images (blurred image, constraint image, and trial solution) into blocks of 11x11 pixels, stacking them into vectors, and formatting them for access by the FORTRAN driver. Reconstruction of MPP output is the reverse procedure. The MPP itself is saved for the computer-intensive tasks of matrix inversion and matrix multiplication.

The primary software system that we use for interactive image analysis is Interactive Data Language (IDL, Research Systems Inc., Denver CO). To generate a single PSF from the intensity distributions of stars on the blurred image, we use DAOPHOT by Peter Stetson at Dominion Astrophysical Observatory.

We have also installed IDL on the MPP VAX-host as the user's high-level language to guide the restoration. The following IDL statements constitute the complete set of commands to restore an image stored as the variable, BLUR, with a point-spread function, PSF, and a block-size of 11 and step-size of 7 (i.e. the blocks overlap, and only the central 7x7 portion of a block is retained). C is an image controlling the constraint for each pixel, varying from 0.0 (no constraint) to 1.0 (full constraint) for each pixel. The two reciprocal Lagrangian multipliers, γ_I and γ_t, are both set to 0.001. TRIAL is the trial solution, and X is the first estimate of the restored image. The routine, dnext, invokes the next iteration. The last statement reads in the output from the MPP stored in the file, 'OUT.TMP', into the variable, X.

```
IDL> setblur,BLUR,PSF,11,7
IDL> setgamma, .001, .001
IDL> setcon, C
IDL> settrial, TRIAL

IDL> dnext, X
----   MPP   ----
IDL> bresult, X
```

Transfer of data and computations are carried out by the FORTRAN driver on the VAX 780 and Parallel Pascal and assembly language on the MPP. The appendix gives the PP code for matrix inversion and matrix multiplication implemented on the MPP.

Typically one iteration takes a few minutes of CPU-time when the VAX/MPP is not bogged down with other users. This time does not include wait-time for the MPP, overhead in transferring the data, etc. Since it is possible to examine an image *while* the MPP task invoked by DNEXT is running, the turn-around time is short enough that interactive work is a reasonable proposition. For restoration, it is essential: the eye can spot minute but systematic imperfections in the restored image as well as catch glaring errors.

The matrix multiplication algorithm given above takes 0.03 seconds to multiply two 128x128 matrices, and 0.06 seconds to invert the same size matrix. A typical image (512x512 pixels) requires the solution of about 5000 linear systems of size 121x121. Consequently, one iteration (block size of 11x11 pixels, step size of 7 pixels in both line and sample directions, spatially varying constraints) takes about 11 CPU-minutes. On a VAX 11/750, the identical procedure would take over 3 cpu-days!

4. APPLICATION OF THE ALGORITHM

We now describe the practical application of the algorithm, by considering four types of astronomical images requiring restoration.

Case 1: Clusters of point sources. Examples are double stars or crowded star fields, such as globular clusters. For the brighter sources, specialized observational techniques, such as speckle interferometry, are obviously superior approaches to higher resolution. For fainter sources in crowded fields, such as Cepheids in other galaxies, these techniques may not be feasible, and deconvolution of the image data at hand may be necessary.

Case 2: Point-source juxtaposed to or superposed on an extended source, where the point source is much *brighter* than the underlying extended source. The extreme example of this case is the quasar/host-galaxy, in which the nucleus of the galaxy (the quasar) has a surface brightness ~10 times that of the underlying galaxy. Even the far wings of the quasar's point-spread function swamp the light from the galaxy, making it difficult to ascertain what kind of galaxy plays host to a quasar. Deconvolution of a

quasar image holds the promise of separating, in effect, the quasar from the galaxy so that the galaxy can be examined directly. If the galaxy has structure, such as spiral arms, deconvolution will enhance the structural detail.

Case 3: Point-source juxtaposed to or superposed on an extended source, where the point-source is much *fainter* than the underlying extended source. This situation occurs in "deep sky" images, where the peak core brightness of a galaxy may be only 1% of the underlying sky. Astronomical seeing conditions or some other blurring mechanism may push the "nose" of a faint galaxy below the detection threshold. Conversely, deconvolution of the image may extend the detection limit.

Case 4: An extended source with structure. There are numerous examples of this case, such as planetary atmospheres or galaxies with dust lanes or jets. Usually, the structure is too arbitrary or complex to deduce the detailed physical structure via the route of convolving the point-spread function with some analytical model. Direct deconvolution is necessary.

In the following section, we show examples of each of these cases, paying particular attention to selection of the constraint factors, which as we noted earlier, are often a matter of judgement.

5. EXAMPLES

5.1 *Detection and photometry of objects in a crowded field* (image courtesy, T. Stecher, GSFC).

The UV rocket image of galaxy M101 (figure 2, upper left) suffers from a motion blur due to instabilities in the rocket's pointing system. Most sources in the image are OB/HII associations, which can be considered point sources at this resolution. It is of astronomical interest to make photometric measurements of these HII regions.

Figure 2 (upper right) shows the restoration using a constant constraint, γ_I=0.01 and γ_t=0.01, for all pixels in the image. The blurred image served as the zeroth iteration and as the trial solution. Some increased resolution is evident. We then used a simple thresholding technique to locate the HII regions in the first iteration (upper right image) and made a second iteration with the constraints removed at the locations of the detected HII regions. Also in the second iteration, we increased γ_t by a factor of 10 (for reasons explained in the next paragraph). As shown in figure 2 (lower left) the HII regions are better resolved, so that fluxes of individual HII regions can now be estimated.

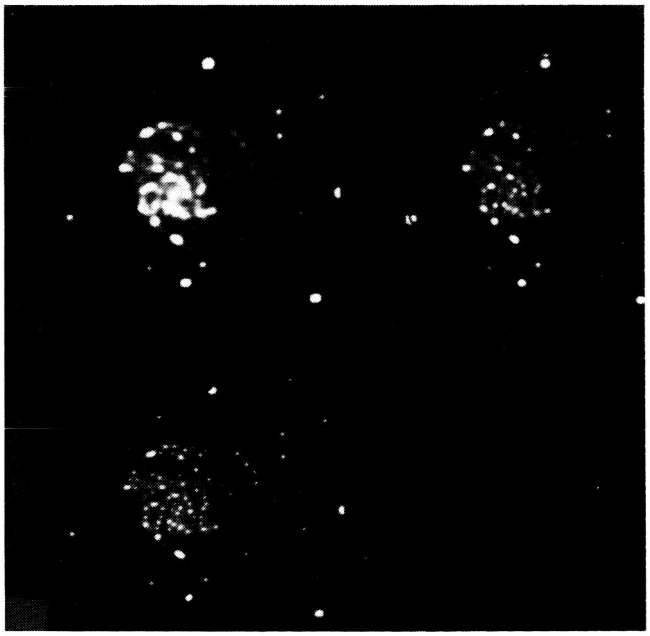

Figure 2. Ultraviolet Image of M101.
Upper Left: Original, Blurred Picture;
Upper Right: First Iteration (with constant constraints);
Lower Left: Second Iteration (with variable constraints).

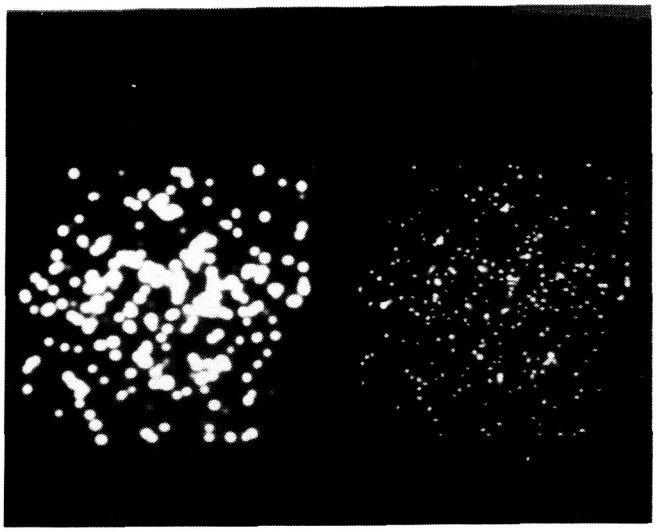

Figure 3. Computer-Generated Star Field.
Left: Original, Blurred Image;
Right: Restored Image.

We used a computer-generated star field (figure 3) to investigate the accuracy of detection and photometry that can be achieved in a restored image. The left image shows a blurred star field with a point spread function equal to the sum of two Gaussians, one with σ= 2.0 pixels, the other with σ= 4.0 pixels. The latter has a peak flux of 10% of the first. Three hundred stars were generated at random positions and with random magnitudes. Magnitudes ranged from 443 peak counts for the dimmest star to 90,750 peak counts for the brightest star. In an attempt to make the simulation realistic, we constructed each star individually using an analytical form of the point spread function, so that star centers fall at arbitrary positions between pixel centers. We simulated counting-statistic noise by adding Gaussian- distributed random numbers scaled by the square root of the counts in each pixel. In the first iteration, we used constant constraints for all pixels, with γ_1 is set to 0.001. We experimented with different values of γ_t and found that setting γ_t at 0.01 gives the best results. In the second iteration (figure 3, right), we removed the constraints at the locations of stars detected on the first iteration.

A comparison between star detection in the blurred image and the restored image shows no significant improvement for the stars with a neighbor less than 3.5 pixels away. The most significant improvement in detection is for the 54 stars having neighbors between 3.5 to 5.0 pixels away (i.e. less then the full width-half maximum of the point spread function, which is approximately 5 pixels). Only 21 out of 54 stars are detected in the blurred image, while 48 stars are detected in the restored image. The average photometric error for these stars is 8%. For stars with separations greater than 5.0 pixels, the average error is 6%.

5.2 *Deconvolution of the quasar, 2130+099 (II Zw 136)* (courtesy, T. Heckman, U. Md.)

This quasar is a relatively bright (m_V=14.8), low-redshift (z=0.06) quasar. Figure 4 (top) compares the cross- sectional profile of the quasar with that of a nearby star on the same image. The difference between the two profiles is the contribution of the host galaxy. The aim of the restoration is to distinguish the host-galaxy from the quasar.

We found that it is essential to center the point-spread function *exactly* on the quasar (i.e. to a hundredth of a pixel); otherwise the misalignment causes ringing in the restored image. We used the intensity distribution of a nearby star to represent the point-spread function at the location of the quasar. To the extent that there is noise in this distribution or the PSF is spatially variant, the assumed PSF will be in error and generate spurious data in the restored image.

As γ_J increases, the noise in the solution decreases but ringing starts to set in. We need to impose the constraint for the host galaxy to avoid excess noise and to release the quasar from the Laplacian constraint to avoid the ringing. Thus, we could set the constraint image to 1.0 everywhere except at the location of the quasar and its immediate neighbors.

The host galaxy is not much brighter than the background sky, so we let the trial solution be given by the sky brightness (254 counts per pixel) everywhere except for at the quasar, where we set it to 3.5×10^5 (computed as the quotient of the maximum count-level in the quasar (21990) divided by the maximum count-level of the PSF (0.07).

The restored image should have properties somewhere between the original (blurred) image and the trial solution. Thus, we set the first estimate, X, of the restored images by calculating:

$$X = BLUR - T \otimes PSF + T ,$$

where the middle term at right is the convolution of the trial image and the point-spread function.

The cross-sectional profile of the restored image is shown in figure 4 (bottom). The quasar is now seen for what it is: a galaxy with an exceedingly bright nucleus. The morphological structure of the galaxy is consistent with its interpretation as a spiral galaxy.

The contrast level between the quasar and the host galaxy is more than 1000. Thus, "ringing" at the 1% level in the restored image would completely wipe out the galaxy image. This is why deconvolution of the quasar image is such a difficult problem. Deconvolution in one dimension has been carried out before (Bendinelli et al. 1984), but a full 2D deconvolution such as this requires the MPP.

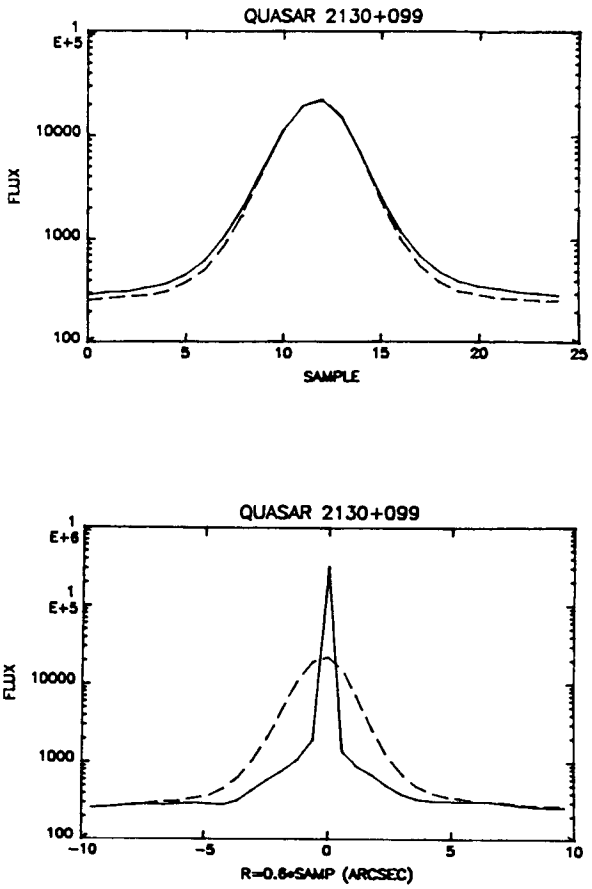

Figure 4. Cross-sectional profiles of the quasar, 2130+099. Top: profile of quasar (solid line) and nearby star (dashed line). Bottom: profile of restored image (solid line) and original, blurred image (dashed line).

5.3 *Restoration of Voyager images of Jupiter and Ganymede.* (courtesy, E. Danielson, CalTech)

The left-hand side of figure 5 shows images of Jupiter and Ganymede taken by Voyager. A point spread function was constructed using an image of a star taken with Voyager. The images on the right show the restoration after two iterations with a constant constraint with γ_I and γ_t set to 0.03 (selected by visual examination of results with various values for the reciprocal Lagragian multipliers). The improved resolution in the images on the right will be important for analysis of weather patterns on Jupiter and study of planetary detail with images from the Hubble Space Telescope.

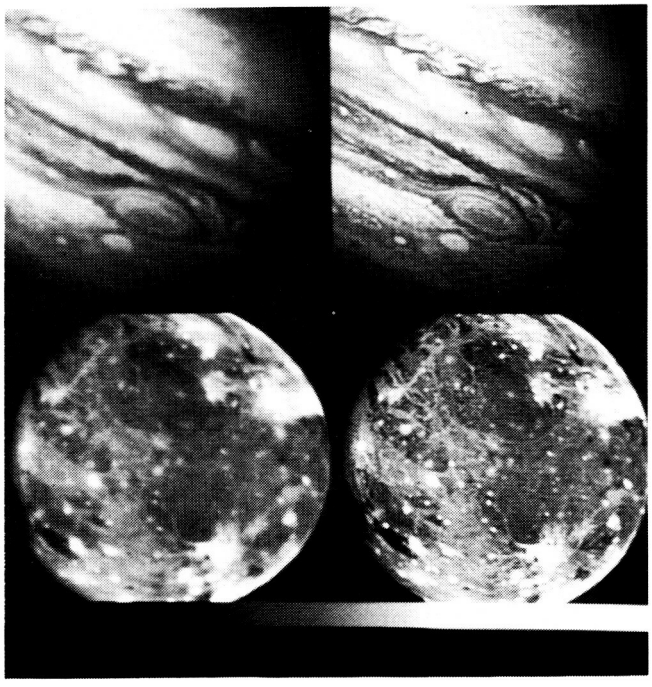

Figure 5. Voyager Images of Jupiter and Ganymede.
Left: Original, Blurred Images;
Right: Restored Images.

REFERENCES

Andrews, H. C., Hunt, B. R., *Digital Image Restoration*, (Prentice Hall: New Jersey), pp. 148-149 (1977)

Bendinelli, O., Lorenzutta, S., Parmeggiani, G., and Zavatti, F., "Deconvolution of photographic images of quasars from point spread functions", *Astron. Astrophys.* 128, 337-342 (1984).

Philips, D. L., "A technique for the numerical solution of certain integral equations of the first kind", *J. ACM*, 9, 84-97 (1962).

Smith, E., Heckman, T., Bothun, G., Romanishin, W., Balick, B., "On the nature of QSO host galaxies", *Astrophys. J.* 306, 64-89 (1986).

Twomey, S. "On the numerical solution of the Fredholm integral equations of the first kind", *J. ACM* 10, 97-101 (1963).

Young, D. M., "Iterative solution of large linear systems", (Academic Press: New Yourk), pp. 434-437 (1971).

APPENDIX

A.1 Matrix Inversion

The matrix inversion implemented on the MPP in Parallel Pascal uses Gaussian elimination with no pivoting. The following Parallel Pascal code will invert the matrix A, size M + 1, where X, COLK, and ROWK are 128 x 128 floating point arrays. ROWFLAG and COLFLAG are 128 x 128 boolean arrays. ROW-INDEX and COL-INDEX are 128 x 128 integer arrays which contain the row index and column index of each element respectively (indices run from 0 to 127). Initially, X contains the matrix A to be inverted and upon completion X will contain B, the inverse of A. Columns in X will be referred to by columns in matrix A or B, whichever is appropriate since both matrices are stored in X.

```
1. for K:= 0 to M do
2.  begin
3.   ROWFLAG:= ROW-INDEX = K;
4.   COLFLAG:= COL-INDEX = K;
5.   rowbroad(X,COLK,128,COLFLAG);
6.   where COLFLAG do X:= 0;
7.   where COLFLAG and ROWFLAG do X:= 1;
8.   where ROWFLAG do X:= X/COLK;
9.   colbroad(X,ROWK,128,ROWFLAG);
10.   where not ROWFLAG do X:=
            X-COLK*ROWK;
11.  end;
```

Line 1 begins a loop on each column K in the matrix X. Line 3 sets the boolean variable ROWFLAG true for elements in row K and false everywhere else. Line 4 sets COLFLAG true for elements in column K. ROWFLAG and COLFLAG will be used as masks for subsequent operations. Line 5 takes the Kth column in matrix A and propagates each element along its corresponding row. We no longer need to retain the Kth column of matrix A. This column can now be used to store the Kth column in matrix B, which initially is set to 1 in row K and zero elsewhere. This is done by lines 6 and 7 which set column K to all zeros (line 6) and then set b_{kk} to 1 (line 7). Line 8 divides row K by the value in position a_{kk}. (Remember that line 5 had propagated the value of a_{kk} along the entire row K). Line 9 will take the Kth row and propagate each element along its corresponding column. Line 10 is the real work horse and subtracts a_{ik} * row k from every row except k in A. Since B is also stored in X, the same operations are automatically performed on B. This step would have produced zeros in column k of the original matrix A in every location except a_{kk} which would have the value of one. Upon completion of the loop for all columns, the matrix X will contain the inverse.

A.2 Matrix Multiplication

The matrix product of two arbitrary size square matrices A and B can be written as:

$$C = \sum_{i=1}^{n} (A_i \otimes B_i)$$

where n is the number of rows and columns in the matrices, and $\otimes$ indicates element by element multiplication (not matrix multiplication). The following Parallel Pascal code gives the implementation of the algorithm on the Massively Parallel Processor. A and B are the input matrices and C will contain the result. M is the number of rows and columns minus one. COLFLAG and ROWFLAG are boolean variables and AI and BI are matrices used to store the propagated columns and rows of A and B. ROW-INDEX and COL-INDEX are defined in Section A.1.

```
1.  C:= 0;
2.  for I:= 0,M do
3.   begin
4.    ROWFLAG:= ROW-INDEX = I;
5.    COLFLAG:= COL-INDEX = I;
6.    rowbroad(A,AI,128,COLFLAG);
7.    colbroad(B,BI,128,ROWFLAG);
8.    C:= C + AI * BI
9.   end;
```