Discrete Radon transform has an exact, fast inverse and generalizes to operations other than sums along lines

William H. Press*

Los Alamos National Laboratory, Los Alamos, NM 87545

* To whom correspondence should be addressed; E-mail: wpress@lanl.gov.

Contributed by William H. Press, October __, 2006.

Götz, Druckmüller, and independently Brady have defined a discrete Radon transform (DRT) that sums an image's pixel values along a set of aptly chosen discrete lines, complete in slope and intercept. The transform is fast, $O(N^2 \log N)$ for an $N \times N$ image; it uses only addition, not multiplication or interpolation; and it admits a fast, exact algorithm for the adjoint operation, namely backprojection. This paper shows that the transform additionally has a fast, exact (though iterative) inverse. The inverse reproduces to machine accuracy the pixel-by-pixel values of the original image from its DRT, without artifacts or a finite point-spread function. Fourier or FFT methods are not used. The inverse can also be calculated from sampled sinograms and is well-conditioned in the presence of noise. Also introduced are generalizations of the DRT that combine pixel values along lines by operations other than addition. For example, there is a fast transform that calculates median values along all discrete lines and is able to detect linear features at low signal-to-noise in the presence of pointlike clutter features of arbitrarily large amplitude.

Radon transform | computerized tomography | backprojection | inverse methods

The Radon transform (RT) of a two dimensional function $f(x, y)$ with compact support that includes the origin is familiar as the set of projections along angles θ, $0 \leq \theta < \pi$,

$$\begin{aligned} \mathcal{R}f \equiv p(\rho, \theta) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy \\ &= \int_{-\infty}^{\infty} f(\rho\cos\theta - \ell\sin\theta, \rho\sin\theta + \ell\cos\theta)d\ell \end{aligned} \qquad \textbf{[1]}$$

where the δ-function converts the two-dimensional integral to a line integral $d\ell$ along the line $x\cos\theta + y\sin\theta = \rho$. The transformed function $p(\rho, \theta)$ is often referred to as the *sinogram* of $f(x, y)$, because a point δ-function in f transforms to a sinusoidal line δ-function in p.

Although occasionally useful and certainly interesting in its own right, the Radon transform pales in practical importance by comparison with its inverse, $\mathcal{R}^{-1}p$, which recovers $f(x, y)$ from its projections. The inverse Radon transform and its approximations enable computer tomography (CT) and related medical and other imaging technologies. That the inverse exists (for suitably well-behaved functions f) follows immediately from the Fourier Slice Theorem, which says that the 1-D Fourier transform of a projection at angle θ has values identical to a radial slice through the origin of the 2-D Fourier transform of the original image. (The proof is straightforward, e.g. (1)). In the continuous case, then, the 2-D Fourier transform of f is recovered in polar coordinates from the slices, and an inverse 2-D Fourier transform recovers f.

Practical applications inevitably deal with images $f(x, y)$ and sinograms $p(\rho, \theta)$ that are represented discretely, usually as 2-D arrays of values. There is a large, if scattered, literature concerning approximations of the continuous Radon transform, and its inverse, in such cases. Standard treatments include refs. (1, 2, 3). Some algorithms have the character of being not only approximations of the continuous case, but also interesting discrete transforms in their own right, for example refs. (4, 5, 6, 7, 8, 9). Collectively these are known as discrete Radon transforms (DRTs). However, no single algorithm has successfully laid claim to being "the" DRT.

A key issue in DRT algorithmics is whether an algorithm, and/or its inverse, is *fast*, in the sense of achievable for an $N \times N$ image in $O[N^2(\log N)^q]$ operations, for some small integer q (ideally 1). Fast DRT algorithms are almost always based on a discretization of the Fourier slice properties of the continuous case, since the fast Fourier transform (FFT) approximates the 1-D continuous transform in $O(N \log N)$, operations and the 2-D transform in $O(N^2 \log N)$. Or, as in ref. (4), fast algorithms may derive from using the FFT for the fast solution of special sets of linear equations such as block-

circulant forms. There are also "slow" DRTs and inverses, with $O(N^3)$ and larger operations count. These are of little practical interest.

Another key issue concerning DRTs is their accuracy. This issue can be framed in various ways. A somewhat argumentative framing is: Given *your choice* for a discrete representation of $f(x, y)$ as $O(N^2)$ values and *your choice* of a DRT algorithm approximating a continuous Radon transform, is there a fast inverse DRT that *exactly* reproduces your discrete representation of f from the values of its DRT? This formulation does not even begin to address the issue of how accurately f is captured by its discrete representation in the first place, or how accurately the DRT approximates a continuous Radon transform. Nevertheless, there seems to be no published DRT that can answer unequivocally "yes" to the question. Those that come closest make special assumptions about $f(x, y)$, for example periodicity along one axis, or severely bandwidth limited. Fast inverse algorithms that are sometimes termed "accurate", are generally so in the sense that a round-trip (sampled image to DRT to reconstructed image) has a point-spread function that is highly concentrated in a few pixels, and with tolerably small tails outside of those few.

The situation is not materially different even if we allow iterative inverse algorithms, where "fast" now can mean $O[N^2(\log N)^q \log \epsilon]$, for small integer q, where ϵ is the desired r.m.s. accuracy, which can be made arbitrarily small. (This is called linear convergence by numerical analysts, and exponential convergence colloquially.)

The principal result of this paper is to show that a particular DRT, proposed independently by Götz and Druckmüller (6) and by Brady (7) (and also related to ref. (10)) has an exact inverse achievable by an (iteratively) fast algorithm. By exact, we mean capable of recovering with arbitrary precision the pixel-by-pixel values of the input image, with no spreading or other artifacts. We give a specific algorithm exhibiting $q = 3$, and suggest that $q = 2$, is likely achievable. Our inverse does not use Fourier methods or FFTs. Other interesting properties of the Götz-Druckmüller-Brady DRT, which may lead to new applications, are also described.

Having a pixel-exact, fast transform pair, DRT and its inverse, does not necessarily advance the practical art of CT. However, it does allow for a clearer distinction between the accuracy of the DRT inversion itself and the various approximations that may be made in mapping physical data to or from the DRT.

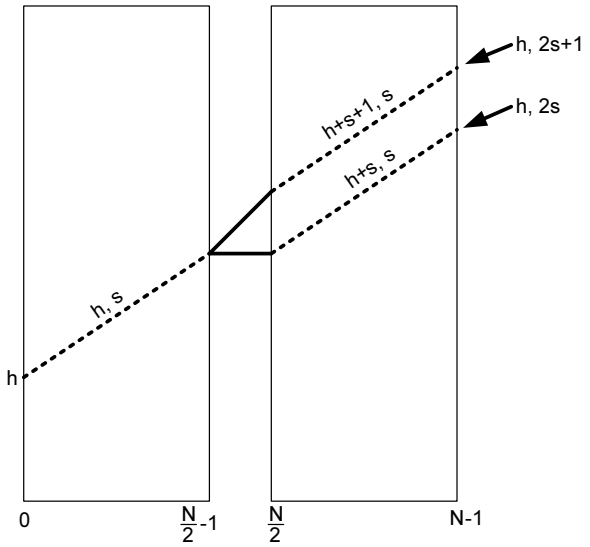

Figure 1: D-lines of width N are defined recursively in terms of d-lines of width $N/2$. The integers h and s parameterize the intercept and rise of each d-line.

The Götz-Druckmüller-Brady DRT

Discrete Approximations of Lines

For definiteness, consider an image represented as an $N \times N$ array of intensity values f_{ji}, with $0 \leq i, j < N$, and N an integer power of 2. Following (6) we define a set of *digital lines*, here called *d-lines*, that transsect the image, passing exactly through one array point in each column of the array and parameterized by integers h and s. The d-line $D_N(h, s)$ connects the array point $(0, h)$ (meaning, by convention, $i = 0$ and $j = h$) and the array point $(N-1, h+s)$. We refer to h as the *intercept* and s as the *rise* of the d-line. We consider for now only the case $0 \leq s \leq N-1$, corresponding to slopes from 0° to 45°, inclusive. D-lines are defined recursively in terms of d-lines on images half as wide,

$$\begin{aligned} D_N(h, 2s) &= D_{N/2}(h, s) \cup D_{N/2}(h+s, s) \\ D_N(h, 2s+1) &= D_{N/2}(h, s) \cup D_{N/2}(h+s+1, s) \end{aligned} \qquad \textbf{[2]}$$

where $\cup$ indicates joining left and right halves. Figure 1 illustrates the recursion, and Figure 2 shows some d-lines for the case $N = 4$.

The d-line $D_N(h, s)$ approximates a continuous line with intercept h and slope $s/(N-1)$. As suggested in ref. (7), its maximum vertical deviation from the continuous line is $\leq \frac{1}{6} \log_2 N$.

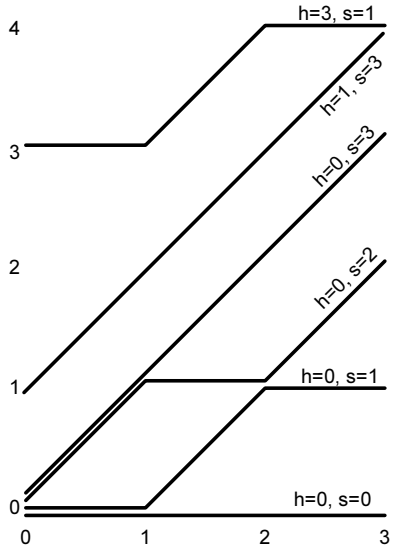

Figure 2: Examples of d-lines for the case $N = 4$. The small vertical offsets are for clarity only; all d-lines pass exactly through integer lattice points, one in each column.

DRT in One Quadrant

We define the DRT as simply the sum of image intensities over array points on a d-line,

$$R^a(h,s) = \sum_{(i,j)\in D_N(h,s)} f_{ji} \qquad \textbf{[3]}$$

with the convention that $f_{ji} = 0$ if i or j is outside of the range $[0, N-1]$. The superscript a is used to indicate the "quadrant" 0 to 45°. (We define b, c, and d quadrant transforms below.) One sees immediately that the recursive definition of the d-lines induces a recursive calculation of the DRT components, simply by associating each partial d-line with its corresponding partial sum of function values. This is illustrated in Figure 3, which shows how two half-images of partial sums are converted to one full image in a single upward sweep. As shown, one row of scratch space is used. Actually, if it mattered, the sweep could be done completely in place by the use of a bit-reversal technique similar to that of the FFT (6). The transform $R^a(h,s)$ of an image is calculated by performing $\log_2 N$ sweeps, first combining adjacent pairs of columns, then pairs of pairs, and so forth. An exact representation of the algorithm is in the Supplemental Text.

Notice that h, the intercept, takes on some negative values so as to include all d-lines that intersect the partial images, or the final full image. When sweeping to produce a partial image of width n, the most negative value of h

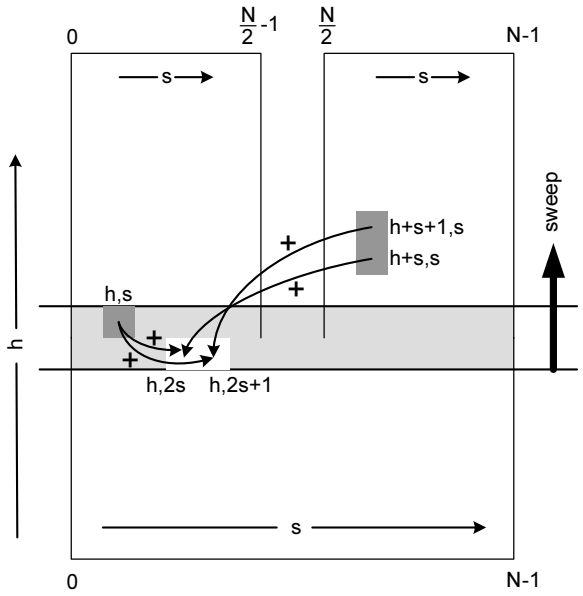

Figure 3: Partial sums on two half-images (top) are replaced by partial sums on a full image (bottom) in a single upward sweep. The DRT is computed by performing this process first for pairs of columns in an image, then pairs of pairs, and so forth in $\log_2 N$ sweeps.

is $-n+1$, since a 45° d-line with that intercept will intersect just the single pixel in the lower right of the partial image. Specifically, sums are performed for i and j on each partial image satisfying all of

$$\begin{aligned} 0 \le i &\le n-1 \\ -n+1 \le j &\le N-1 \\ 0 \le i &+ j \end{aligned} \qquad [4]$$

for a total of $nN+\frac{1}{2}n(n-1)$ in each partial image. Summing the partial images in each sweep, and the $\log_2 N$ sweeps gives a total of $(N^2-\frac{1}{2}N)\log_2 N+N(N-1) = O(N^2 \log_2 N)$ addition operations. Note that there are no multiplications, or other floating operations, a point to which we return below.

Remaining Quadrants and Global Topology

Again following refs. (6, 7), we define the DRT for other angle ranges by appropriately flipping the original image, then repeating exactly the same algorithm

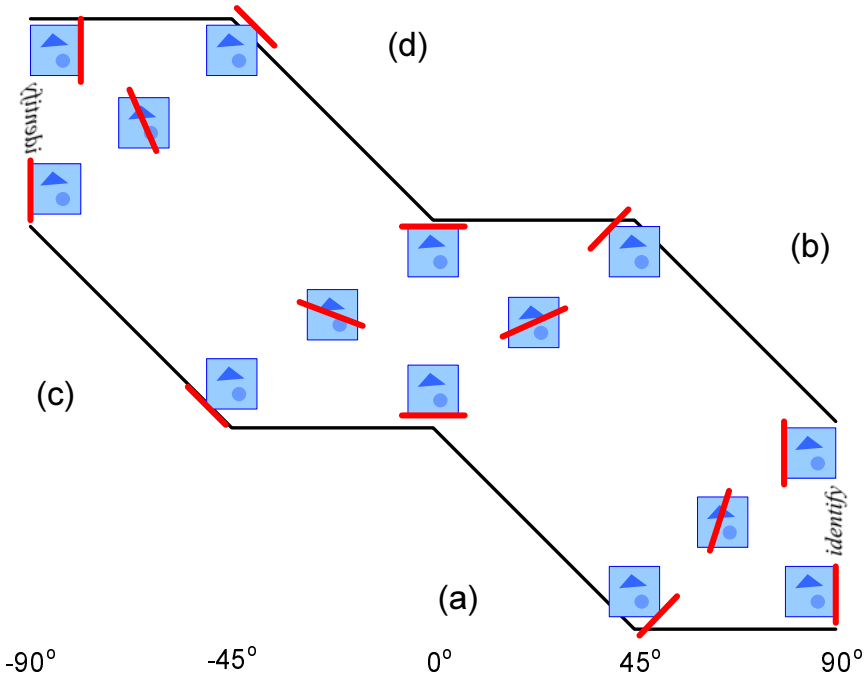

Figure 4: DRTs calcuated in four quadrants by equation [**5**] stitch together as shown to give the global DRT. In each quadrant, the parameter h varies along the vertical direction, s along the horizontal. Red lines indicate schematically the sums through the original image (blue) corresponding to locations in the DRT. Left and right edges are identified with a twist: a Möbius strip.

as for quadrant a:

$$\begin{aligned} R^b\{f_{ji}\} &= R^a\{f_{ij}\} && (45^\circ \text{ to } 90^\circ) \\ R^c\{f_{ji}\} &= R^a\{f_{i,N-1-j}\} && (-90^\circ \text{ to } -45^\circ) \\ R^d\{f_{ji}\} &= R^a\{f_{N-1-j,i}\} && (-45^\circ \text{ to } 0^\circ) \end{aligned} \qquad [\mathbf{5}]$$

Figure 4 shows how the four quadrants stitch together to form the global DRT. In each quadrant, the intercept h varies in the vertical direction, while the rise s varies horizontally. In quadrants a and c, both parameters increase from the lower-left; in b and d, from the upper right. Each quadrant contains $N^2 + \frac{1}{2}N(N-1)$ values, a triangle below (or above) a square. The whole DRT thus contains $6N^2 - 2N$ values. Each point in the DRT corresponds to a line through the original image, shown schematically as red lines through a fixed blue image. The top and bottom boundaries of the DRT correspond to bounding cases where the line only just intersects the image. The left and right edges are identified with a twist, as indicated in the Figure.

Supplemental Figure S1 shows two sample images, while Supplemental Figures S2 and S3 show their global (i.e., four quadrant) DRTs, plotted graphically. The images are $256 \times 256 \times 8$ bits and are respectively a standard photographic test image and a section of Zubal's standard head phantom (14).

Relation between DRT and Sinogram

While the DRT is defined from a discrete image, as above, one may also identify it with values sampled from the continuous sinogram of a continuous image. Details are given in the Supplemental Text. Supplemental Figure S4 shows a $256 \times 256 \times 8$ bit sinogram of the standard Shepp-Logan head phantom, captured from the web as a compressed GIF image, and its nearly perfect reconstruction by the methods of this paper.

Inverse DRT

Backprojection is Fast and Exact

For the continuous Radon transform, backprojection is the adjoint operator to the transform,

$$\begin{aligned} \mathcal{R}^* p \equiv b(x,y) &= \int_{-\infty}^{\infty} \int_0^{\pi} p(\rho,\theta)\delta(x\cos\theta + y\sin\theta - \rho)d\rho d\theta \\ &= \int_0^{\pi} p(x\cos\theta + y\sin\theta, \theta)d\theta \end{aligned} \qquad \textbf{[6]}$$

One sees that it is the integral over all directions of all the projections that pass through the point (x, y). While the adjoint operator is not the same as the inverse, we will see that its discrete analog is a useful building block.

Just as it induced a fast DRT, the recursive definition of d-lines (Eq. **[2]** and Figure 1) induces a fast backprojection algorithm (7, 11). The process essentially reverses the sweep shown in Figure 3: Every element of left- and right-half partial images is assigned the sum of two full-image elements, in correspondence with the d-line recurrence. If $B_n(h, s)$ denotes a partial back-transformation on an image of width n, then we have,

$$\begin{aligned} B^L_{n/2}(h,s) &= B_n(h,2s) + B_n(h,2s+1) \\ B^R_{n/2}(h+s,s) &= B_n(h,2s) + B_n(h-1,2s+1) \end{aligned} \qquad \textbf{[7]}$$

where superscript L and R refer to the left and right half-width images. The backprojection for each quadrant is obtained in $\log_2 N$ sweeps terminating with

$n = 1$. Each sweep can be done in place, or with a small amount of scratch memory. The full backprojection is an appropriate sum over quadrants,

$$B_{ji} = \frac{1}{4(N-1)} \left(B^a_{ji} + B^b_{ij} + B^c_{i,N-1-j} + B^d_{N-1-j,i} \right) \qquad [8]$$

where the normalization constant shown will be convenient later. As should be clear, this calculation is exact. That is, the result is exactly the sum of the discrete projections along d-lines that pass through an image pixel. The Supplemental Text has an exact statement of the algorithm represented by Eqs. [**7**] and [**8**].

"Natural" Approximate Fourier Inversion

There is a "natural" approximate inverse of the DRT as we have defined it, obtainable by Fourier methods. By natural, we mean that no interpolation is required in getting into and out of Fourier space, nor necessarily any explicit application of a ramp or other (e.g., Shepp-Logan) filter.

Because the main point of this paper is to give an exact inverse that does not use Fourier methods, we relegate discussion of the approximate Fourier inverse to the Supplemental Text. Supplemental Figure S5 shows the approximate Fourier inverses, without any additional filtering, of the sample images shown in Supplemental Figure S1. With additional filtering, the visual quality of these images could be substantially improved. However, this would lose information and merely be moving us towards existing, approximate, techniques (1, 2, 3).

Also discussed in the Supplemental Text is an exact, but ill-conditioned, formal inverse. Supplemental Figure S6 shows how the growth of instabilities render such an inverse useless. Needed is not just an inverse, but a well-conditioned inverse.

Key Ideas for a Fast, Exact Inverse

We get a fast, exact inverse by combining three ideas, each one well-known in other contexts. The first idea is to use the fast (forward) DRT for the iterative improvement of an approximate inverse. Suppose, in general, we want to solve the linear system $\mathbf{Ax} = \mathbf{b}$, and that $\mathbf{B}_0$ is an approximate inverse to $\mathbf{A}$ in that the matrix

$$\mathbf{R} \equiv \mathbf{1} - \mathbf{B}_0\mathbf{A} \qquad [9]$$

is small (in a sense we discuss below). Then one easily shows (ref. (12), §2.5) that

$$\mathbf{B}_n \equiv (1 + \mathbf{R} + \mathbf{R}^2 + \cdots + \mathbf{R}^n)\mathbf{B}_0 \qquad [10]$$

becomes $\mathbf{A}^{-1}$ as $n \to \infty$, if the series converges. It follows from this that the recurrence

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{B}_0(\mathbf{b} - \mathbf{A}\mathbf{x}_n) \qquad [\mathbf{11}]$$

converges to the solution $\mathbf{x}$ from all starting values. Note that Eq. [**11**] requires only the application of the (exact) forward operator $\mathbf{A}$ and the approximate inverse operator $\mathbf{B}_0$. At each step we calculate a correction by approximately inverting the residual of the previous step. Equation [**10**] already shows the sense in which $\mathbf{R}$ must be small: All of its eigenvalues must have magnitudes ≤ 1, or else the series will diverge. This is by no means easy to achieve for complicated operators in large-dimensional spaces. For example, the natural, approximate Fourier DRT inverse mentioned above does not satisfy this condition and cannot be iterated.

The second idea is to high-pass filter the (exact, fast) backprojection by a local convolution, with the goal of reproducing only the highest frequency information in the image to tolerable accuracy. By highest frequency, we mean frequencies between 1/2 and 1 times the Nyquist frequency. We will see below what tolerable accuracy means.

For example, the centered 9-point filter with coefficients

$$\mathcal{H} = \begin{pmatrix} -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{16} \\ -\frac{1}{8} & \frac{3}{4} & -\frac{1}{8} \\ -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{16} \end{pmatrix} \qquad [\mathbf{12}]$$

(and the obvious reflections at edges and corners) has the property that it has unity response both to a binary checkerboard (Nyquist frequency in both directions) and binary stripes (Nyquist frequency in one direction); and zero response to a constant. The intent is to use backprojection, filtered in this way, not on the actual image, but only on a residual (cf. Eq. [**11**]) that has missing or inaccurate high-frequency information. The filtering requires $O(N^2)$ operations, as compared to $O(N^2 \log_2 N)$ for the backprojection.

The third idea is the full multigrid method (FMG) (13, 12) and the related concepts of restriction and prolongation operators. A restriction operator is, here, a linear map from a DRT of size N, denoted $R_N^{a\ldots d}(h,s)$ to one of size $N/2$, with the idea that the smaller DRT should approximate the larger one. A prolongation operator is a map on images (not DRTs) from size $N/2$ to size N. There are many such operators. With foresight, we use these specific ones:

$$\begin{aligned} R_{N/2}^{a\ldots d}(h,s) &= \tfrac{1}{4}\left[R_N^{a\ldots d}(2h,2s) + R_N^{a\ldots d}(2h+1,2s)\right] \qquad \text{(restriction } \mathcal{S}\text{)} \\ f_{ji}^N &= f_{j+1,i}^N = f_{j,i+1}^N = f_{j+1,i+1}^N = f_{\lfloor j/2\rfloor,\lfloor i/2\rfloor,}^{N/2} \qquad \text{(prolongation } \mathcal{P}\text{)} \end{aligned} \qquad [\mathbf{13}]$$

where $a\ldots d$ denotes any of the values a, b, c, or d. Notice that $\mathcal{S}$ averages adjacent values of h, but samples only even values of s. This choice works,

while other seemingly equally valid choices don't work, as we shall discuss. Also to be noted is that the output of $\mathcal{S}$ is not necessarily the exact DRT of any image; it will prove useful nevertheless.

Details of the Fast, Exact Inverse

Define an algorithm $\mathcal{B}$ that constructs an approximate inverse f_{ji}^{N} of a DRT $R_N^{a\ldots d}$ as follows:

1. Apply $\mathcal{S}$ to $R_N^{a\ldots d}$, giving $R_{N/2}^{a\ldots d}$.
2. Recursively call $\mathcal{B}(R_{N/2}^{a\ldots d})$ (this algorithm), giving $f^{N/2}$.
3. Apply $\mathcal{P}$ to $f^{N/2}$, giving $f^{N\prime}$, a low-frequency approximation to f^N.
4. Calculate the forward DRT of $f^{N\prime}$ and subtract it from the original $R_N^{a\ldots d}$, giving a residual DRT.
5. Backproject and locally high-pass filter the residual DRT, giving an image correction.
6. Add the image correction to $f^{N\prime}$, producing f^N.

Supplemental Figure S7 shows the result of applying the approximate inverse operator $\mathcal{B}$ to the DRTs shown in Supplemental Figures S2 and S3.

Note that, recursion and all, $\mathcal{B}(R_N^{a\ldots d})$ is a strictly linear operator on $R_N^{a\ldots d}$. In fact, it can be used as $\mathbf{B}_0$ in Eq. [**11**], but only if the series Eq. [**10**] converges. To prove convergence, we must show that the residual matrix $\mathbf{R}$ has eigenvalues that are < 1 in magnitude. Because of $\mathcal{B}$'s complicated, recursive algorithmic description, it seems hopeless to do this analytically, so we must resort to numerical experiment.

Convergence Tests

We now show that the specific choices given for the high-pass filter (Eq. [**12**]) and prolongation and restriction operators (Eq. [**13**]) do produce a convergent series. There is art in those specific choices. As remarked above, other choices that might seem equally valid *a priori* yield divergent results. On the other hand, the specific choices shown are known to be not optimal (see Discussion) and can be improved.

Statistically, an image whose pixels are i.i.d. normal deviates will have a significant projection into all eigenmodes. We take the DRT of such images, then iterate Eq. [**11**] and verify convergence (in L_2 norm) to the original image. Initial convergence is faster than a single exponential, because it is dominated

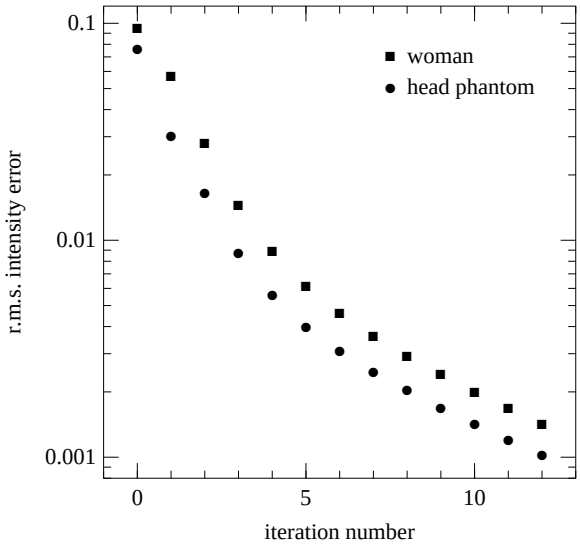

Figure 5: Error versus iteration number for the two sample images in Supplemental Figure S1. Iteration 0 is the error of the approximate inverse $\mathcal{B}$; subsequent iterations are improved by reapplying $\mathcal{B}$ to the residual DFTs. Errors are normalized to a grayscale range of $[0, 1]$.

by large numbers of modes with a wide range of small eigenvalues. As iteration number n increases, the residual error is found to approach a single exponential, characteristic of the largest eigenvalue of $\mathbf{R}$. Multiple trials over a range of image sizes from $N = 16$ to $N = 1024$ are accurately summarized, asymptotically, by the empirical relation,

$$\Delta \ln \text{ error (per iteration)} \approx -\frac{13.8}{(\log_2 N)^2} \qquad [\mathbf{14}]$$

The workload to converge to accuracy ϵ thus scales as $O[N^2(\log N)^3 \log \epsilon]$.

In practical work, asymptotic convergence rates may not matter that much. For all sizes up to $N = 2048$, realistic images are found to converge to useful accuracies (e.g., indistinguishable by eye from the exact answer) in a modest number of iterations. Figure 5 shows how the r.m.s. errors decrease with iteration number for the two sample images shown in Supplemental Figure S1. The leftmost values (plotted as iteration 0) are the errors, relative to a grayscale range $[0, 1]$, after the initial application of the approximate inverse operator $\mathcal{B}$. Iterations $1, 2, \ldots$ show the result of iterative improvement. The asymptotic approach to a single dominant exponential is clearly seen.

Supplemental Figure S8 shows the inverses obtained from the DRTs in Supplemental Figures S2 and S3 after only four iterations. After a few more iterations than this, the inverses become perceptually indistinguishable from

the original images in Supplemental Figure S1. Futher iterations converge to arbitrarily small r.m.s. errors.

The DRT Need Not Use Ordinary Addition

We already noted that the DRT defined above uses only addition, and no other arithmetic operations, in combining data values along a d-line. A consequence is that any associative and commutative operation can serve instead of addition, yielding a number of interesting generalizations of the DRT. The DRT algorithm "presents" data elements (and their partial "sums") to us in an arbitrary order; we can combine these using any rule with the semantics of addition that we want.

One example is a discrete *median* Radon transform (DMRT), where the transform output is the median of all image values along a d-line. Here the combination rule is to update an approximate representation of the cumulative distribution function (see, e.g., (15)) and finally output the median quantile. The combination rule can be viewed as an associative and commutative operator on the data values. In fact, we can use *exactly* the same code as for the standard DRT, simply overloading the addition operator with machinery that combines distribution functions.

Since the median along a line is a robust estimator of that line's central value, but is insensitive to large fluctuations in the tails, the DMRT is good at finding low signal-to-noise straight lines in the presence of much larger pointlike clutter. Such lines appear as clusters of unusually large values in the transform space. Detection can be accomplished by identifying transform values above some threshold, or above some threshold of statistical significance (adjusting for the variable number of image pixels intersected by a d-line). Alternatively, one can invert the thresholded DMRT *as if it were an ordinary DRT*. The goal is not to reconstruct an accurate image, but rather to reconstruct in the image space a mask that shows the detected lines.

Figure 6 shows an example. A synthetic image consists of Gaussian noise pixels, small positive and negative squares of arbitrarily large amplitude, and an almost invisible line that is 4 pixels wide and has values 0.5 standard deviations larger than the background. A portion of the DMRT is shown, after it has been thresholded at the 99.9% quantile level. The DRT inverse of this DMRT strongly highlights the line. The small squares appear only as faint background linear features when there are chance alignments among them.

Since the ordinary DRT itself increases the significance of line features (which project to a point) over point features (which smear to a sinusoid), the power of the DMRT may not be intuitive. For the image in Figure 6, however, ordinary projection is not nearly good enough. Supplemental Figure S9 shows

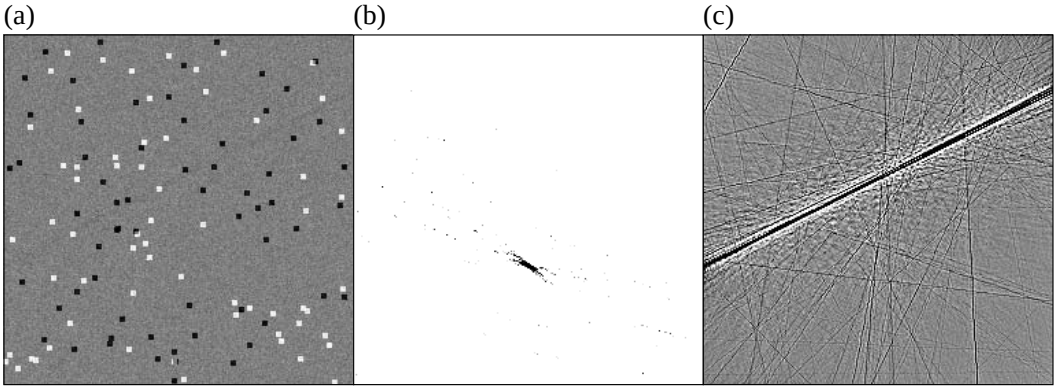

Figure 6: (a) Image with an almost invisible line, 4 pixels wide and $0.5\,\sigma$ above the noise background. (b) Portion of its discrete *median* Radon transform (DMRT), whose values are medians, not sums, along lines. Only the largest 0.1% of transform values are plotted. The line produces a significant feature. (c) Inverse of (b), computed as if it were a DRT, not a DMRT. The location of line feature is now apparent.

the same region of transform space as Figure 6 (b), but for the ordinary DRT. One sees that the weak feature of interest is completely lost in a confusion of much stronger, overlapping sinusoids.

Supplemental Figure S10 shows the result of taking the DRT inverse of the DMRT of a conventional test image of a face, without thresholding. Surprisingly, a good deal of detail from the original image is reconstructed from the median values along lines, alone.

Not only the median, but also any other property of the distribution function along lines can be computed in like manner. For example, one could compute several moments of the distribution and thus recognize linear features that have means identical to the background, but different variances or skews.

Discussion

Inverse or Pseudoinverse?

The alert reader will already have caught our consistent, but slightly inaccurate, use of the term "inverse". The DRT, as we have defined it, is a linear map from a space of dimension N^2 into one of dimension $6N^2 - 2N$. Since our inverse is also a linear map, it must map (at most) an N^2 dimensional linear subspace of the DRT back into the image space. There must therefore be a

another subspace of (at least) dimension $5N^2 - 2N$ that is mapped to zero by the inverse. The fact that numerical experiments on random matrices yield exponential convergence to known original images provides strong evidence that the respective subspace dimensions are as indicated, and moreover that the maps are well-conditioned. We can shed some additional light on the condition number by a different experiment: We put random i.i.d. normal deviates $N(0,1)$ into all $6N^2 - 2N$ components of the DRT, then take the inverse and ask what is the resulting r.m.s. value of an image pixel? Summarizing results for a range of N, an empirical relation is

$$\text{r.m.s. image pixel} \approx 0.50\, N^{-0.44} \tag{15}$$

which is never large, and decreases as N increases. Finally, we can ask what fraction of the initial variance of the DRT is removed if we subtract the DRT of the inverse image. The result is very close to 5/6, as we expect from counting dimensions, and again suggesting well-conditioned maps.

These checks are relevant to our use of a restriction operator like $\mathcal{S}$ in [**13**]. The smaller DRT obtained by applying $\mathcal{S}$ to a DRT does not exactly correspond to any image. However, we can view it as being the sum of a the DRT of a nearby image, plus noise. If the approximate inverse were ill-conditioned, we might not be able to get away with this, and iterative improvement might fail, behavior that is not seen.

Conjectured Improvements

The appearance of two powers of the logarithm in Eq. [**14**] is not surprising. It is indicative of errors diffusing in logarithmic scale by a gradient-driven relaxation process. In other such situations (e.g., (12), §19.5) the use of techniques such as over-relaxation can change the diffusion rate by one power of the problem size. We therefore conjecture that variants of the methods described here can achieve $O[N^2(\log N)^2 \log \epsilon]$ workload.

If one examines the residual images at a late stage, one sees that residual errors are dominated by frequencies $\sim 2/3$ of the Nyquist frequency, just where the 9-point filter might be expected to be starting to fail. Even for 9-point filters, numerical search easily turns up filters different from Eq. [**12**] that have slightly better convergence rates than Eq. [**14**] (though none seem to be universal for all N). It is therefore a reasonable conjecture that there should exist 25-point filters (that is, 5×5) that considerably improve the constant in Eq. [**14**]. (We would skip 4×4 filters since these can't be centered.)

References

1. Kak, A. & Slaney, M. (1988) *Principles of Computerized Tomographic Imaging*. (IEEE Press, New York).

2. Natterer, F. & O'Malley, R. (2001) *The Mathematics of Computerized Tomography*. (Cambridge University Press, Cambridge, UK).

3. Natterer, F. & Wübbeling, F. (2001) *Mathematical Methods in Image Reconstruction*. (SIAM, Philadelphia).

4. Beylkin, G. (1987) "Discrete Radon Transform," *IEEE Trans. ASSP*, **35**, 162–172.

5. Kelley, B.T. & Madisetti, V.K. (1993) "The Fast Discrete Radon Transform – I: Theory," *IEEE Trans. Image Proc.*, **2**, 382–400.

6. Götz, W.A. & Druckmüller, H.J. (1996) "A Fast Digital Radon Transform – An Efficient Means for Evaluating the Hough Transform," *Pattern Recognition*, **29**, 711-718.

7. Brady, M.L (1998) "A Fast Discrete Approximation Algorithm for the Radon Transform," *SIAM J. Comput.*, **27**, 107—119.

8. Boag, A., Bresler, Y., & Michielssen, E. (2000) "A Multilevel Domain Decomposition Algorithm for Fast $O(N^2 \log N)$ Reprojection of Tomographic Images," *IEEE Trans. Image Proc.*, **9**, 1573–1582.

9. Brandt, A., Mann, J., Brodski, M., & Galun, M. (1999) "A Fast and Accurate Multilevel Inversion of the Radon Transform," *SIAM J. Appl. Math.*, **60**, 437–462.

10. Donoho, D.L. & Huo, X. (2000) "Beamlet Pyramids: A New Form of Multiresolution Analysis, Suited for Extracting Lines, Curves, and Objects from Very Noisy Image Data," *Proc. SPIE*, **4119**, 434–444.

11. Nilsson, S. (1997) *Application of Fast Backprojection Techniques for Some Inverse Problems in Integral Geometry*, Ph.D. thesis, Department of Mathematics, Linköping University, No. 499.

12. Press, W.H., Teukolsky, S.A., Vetterling, W.T., & Flannery, B.P. (2002) *Numerical Recipes in C++* (Cambridge University Press, New York).

13. Hackbusch, W. (1985) *Multi-Grid Methods and Applications*, (Springer, New York).

14. Zubal I.G., Harrell C.R., Smith E.O., Rattner Z., Gindi G., & Hoffer P.B. (1994) "Computerized three-dimensional segmented human anatomy", *Med. Phys.*, **21**, 299-302.

15. Chambers, J.M., James, D.A., Lambert,D. & Vander Wiel, S. (2006) "Monitoring Networked Applications with Incremental Quantiles," *Statistical Sciences* (in press).

Discrete Radon transform has an exact, fast inverse and generalizes to operations other than sums along lines

William H. Press

Los Alamos National Laboratory, Los Alamos, NM 87545

Supporting Information

Supplemental Text

Relation between DRT and Sinogram

If the sinogram $p(\rho, \theta)$ is scaled to be nonzero in the range $-\frac{1}{2} \le \rho \le \frac{1}{2}$, and defined for $0 \le \theta \le \pi$, then the sampled positions are

$$\theta_s = \operatorname{atan}\frac{s}{N-1}, \qquad \rho_{hs} = \cos\theta_s \left(\frac{h}{N} - \frac{1}{2} + \frac{1}{2}\frac{s}{N-1} \right) \qquad \textbf{[1]}$$

in terms of which we have

$$\begin{aligned} R^a(h,s) &= \cos\theta_s \, p(\rho_{hs}, \theta_s) \\ R^b(h,s) &= \cos\theta_s \, p(\rho_{hs}, \pi - \theta_s) \\ R^c(h,s) &= \cos\theta_s \, p(1-\rho_{hs}, \tfrac{\pi}{2} - \theta_s) \\ R^d(h,s) &= \cos\theta_s \, p(1-\rho_{hs}, \tfrac{\pi}{2} + \theta_s) \end{aligned} \qquad \textbf{[2]}$$

with the convention that $p(\rho, \theta) = 0$ for arguments outside of the mentioned ranges. The factors $\cos\theta_s$ scale transform values to account for the fact that the DRT always sums one value in each of N columns, independent of angle, while the sinogram, main text Eq. 1, is a true line integral of length $\propto \sec\theta_s$.

Supplemental Figure S4 shows a $256 \times 256 \times 8$ bit sinogram of the standard Shepp-Logan head phantom, captured from the web as a compressed GIF image, and its reconstruction using equation **[2]** and the exact DRT inverse method that we described in this paper. All of the smallest resolution elements are fully reconstructed, showing that a significant amount of data processing (here interpolation, projection, compression, decompression, sampling) can be tolerated by the methods described here.

Natural Approximate Inverse by Fourier Methods

Defining $a \equiv s/(N-1) = \tan\theta$, we first note that the number of nonzero values $R(h,s)$ in a column of constant s varies as $N(1+a)$ (cf. Figure 4). We therefore zero-pad each column, symmetrically at the two ends, into a vector of length $2N$, and take each column's fast Fourier transform (FFT). Since the data are real, negative frequencies are the complex conjugates of positive ones, so we consider just the N nonnegative frequency components (excluding the component at the Nyquist frequency). It would be natural to think of these components as mapping directly onto the N values along the d-line in two-dimensional frequency space with the same value of s, the rise (see Supplemental Figure S11). But does this work? We need to check that the scaling in frequency space comes out correctly.

The distance represented by the $N(1+a)$ components in a column is not $N(1+a)$ but rather $N(\sin\theta+\cos\theta) = N(1+a)(1+a^2)^{-1/2}$, as shown in Supplemental Figure S12. The sampling interval is therefore $\Delta = (1+a^2)^{-1/2}$. Component $j \geq 0$ therefore represents a frequency $f_j = j/(2N\Delta) = (j/2N)(1+a^2)^{1/2}$. However, the slant distance along the d-line also varies as $\sec\theta = (1+a^2)^{1/2}$. So the scaling works out perfectly, without rescaling or interpolation, as shown in Supplemental Figure S5.

Since multiple d-lines can hit the same frequency cell, especially near the origin, we assign to each cell the *mean* of all components that hit it, a discrete approximation to the Fourier slice theorem. The d-line construction guarantees that every cell gets at least one hit. The two-dimensional inverse FFT of Figure 5 gives, approximately, the original image f_{ji}.

Exact, But Ill-Conditioned, Formal Inverse

It is not hard to exactly reverse the sweep procedure shown in Figure 3, leading to a formal algebraic inverse from the data in a single quadrant. From the Fourier slice theorem we might already expect this single-quadrant formal inverse to be extremely ill-conditioned, since it lacks information on 3/4 of the Fourier plane. In fact, it works at all only because, in the discrete case, there is inevitable (if exponentially small) leakage from one Fourier quadrant to the others.

For $N = 256$, we estimate the condition number of the formal inverse as $\sim 10^{18}$. Supplemental Figure S6 shows the result of trying to invert the DRT of a sample image in floating double precision, after adding noise of magnitude 10^{-14} to the DRT. The result shows exponentially growing stripes whose Fourier transform lies, almost exactly, in another quadrant. (Interestingly, the pattern is a Walsh function.)

This excursion highlights the fact that we desire not merely an inverse, but a well-conditioned inverse.

Principal Algorithms Stated More Precisely

Although the algorithms discussed in the main text are described there, there are some details that are left implicit, for example the handling of boundaries and out-of-range array indices. We here give precise statements. Although derived from executable code, the following fragments, absent many supporting class library definitions, are to be considered only as pseudocode.

The algorithms assume this basic structure for the DRT data and methods:

```
template<class T>
struct Radon {
    Int nn,n2;
    Matrix<T> a,b,c,d;

    Radon(Matrix<Doub> &data);  // forward DRT
    void xformpiece(Matrix<T> &a);  // forward transform, one quadrant
    Radon(Radon &rad, Bool dummy);  // restriction operator
    Radon& operator-=(const Radon &rad); // subtract two Radons
    void backproject(Matrix<T> &ans); // backprojection
    void backpiece(Matrix<T> &ans, Matrix<T> &a); // backprojection, one quadrant
};
```

Here `nn` is N in the main text, `n2` is $2N$, and the matrices `a,b,c,d` that contain the transform have row indices, corresponding to values of h, varying from 0 to `n2`-1 rather than the main text's $-N$ to $N-1$.

The forward DRT is accomplished by the following functions, which can be invoked for any object type `T` that has an addition operator. Different object types, with different overloaded addition operators, automatically implement generalizations of the DRT like the DMRT discussed in the main text.

```
template<class T>
Radon<T>::Radon(Matrix<Doub> &data) : dat(data), nn(data.ncols()), n2(2*nn),
a(n2,nn), b(n2,nn), c(n2,nn), d(n2,nn), slope(nn)  {
    Int i,j;
    if (nn<2 || nn&(nn-1)) throw("nn must be power of 2 in Radon");
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) a[j+nn][i] = data[j][i];
    xformpiece(a);
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) b[j+nn][i] = data[i][j];
    xformpiece(b);
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) c[j+nn][i] = data[nn-1-i][j];
    xformpiece(c);
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) d[j+nn][i] = data[nn-1-j][i];
    xformpiece(d);
}

template<class T>
void Radon<T>::xformpiece(Matrix<T> &a) {
    Int mm,n,ng,i,j,k,ii,nr=a.nrows(),nn=a.ncols(),noff=nr-nn;
    Vector<T> b(nn);
    for (j=0;j<noff;j++) for (i=0;i<nn;i++) a[j][i] = 0.;
```

```
    mm = 1;
    ng = nn>>1;
    while (ng) {
        for (j=0;j<nr;j++) { // loop over rows
            ii = k = 0;  // ii will be 2*n*mm+i
            for (n=0;n<ng;n++) {
                for (i=0;i<mm;i++) {
                    b[k+1] = a[j][ii];
                    b[k] = b[k+1];
                    if ((j+i)+1 < nr) b[k+1] = b[k+1] + a[(j+i)+1][ii+mm];
                    if (j+i < nr)  b[k] = b[k] + a[j+i][ii+mm];
                    ii++;
                    k += 2;
                }
                ii += mm;
            }
            for (k=0;k<nn;k++) a[j][k] = b[k];
        }
        mm += mm;
        ng >>= 1;
    }
}
```

Backprojection is accomplished by these functions:

```
template<class T>
void Radon<T>::backproject(Matrix<T> &ans) {
    Int i,j;
    Matrix<T> tmp(nn,nn);
    if (ans.ncols() != nn) ans.resize(nn,nn);
    Doub val = 0.25/(nn-1.);
    backpiece(ans,a);
    backpiece(tmp,b);
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) ans[j][i] += tmp[i][j];
    backpiece(tmp,c);
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) ans[j][i] += tmp[i][nn-1-j];
    backpiece(tmp,d);
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) ans[j][i] += tmp[nn-1-j][i];
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) ans[j][i] *= val;
}

template<class T>
void Radon<T>::backpiece(Matrix<T> &ans, Matrix<T> &a) {
    Int mm,ng,n,i,ii,j,k;
    Matrix<T> b(n2,nn);
    Vector<T> bb(nn);
    mm = nn>>1;
    ng = 1;
    b = a; // copy a only to avoid overwriting it
    while (mm) {
        for (j=n2-1;j>nn-(mm+mm);j--) { // loop over rows
            for (i=0;i<nn;i++) bb[i] = b[j][i];
            ii = 0;  // ii will be 2*n*mm+i
            for (n=0;n<ng;n++) {
                k = ii; // k will be 2*n*mm
                for (i=0;i<mm;i++) {
                    b[j][ii] = bb[k+i+i] + bb[k+i+i+1];
                    if (j+i < n2) b[j+i][mm+ii] = bb[k+i+i] + b[j-1][k+i+i+1];
                    ii++;
                }
                ii += mm;
```

```
            }
        }
        ng += ng;
        mm >>= 1;
    }
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) ans[j][i] = b[j+nn][i];
}
```

The approximate inverse algorithm $\mathcal{B}$ is:

```
void inverse_recursive(Matrix<Doub> &img, Radon<Doub> &rad) {
    Int i,j,ii,jj,k,nn=rad.nn, n2=rad.n2, nh=nn/2;
    if (rad.nn==1) {
        img[0][0] = rad.a[1][0];
    } else {
        { //scope
            Radon<Doub> radh(rad,true);
            Matrix<Doub> imgh(nh,nh);
            inverse_recursive(imgh,radh); // recursive call!
            for (jj=0,j=0;j<nh;j++,jj+=2) {
                for (ii=0,i=0;i<nh;i++,ii+=2) {
                    img[jj][ii] = img[jj+1][ii] = img[jj][ii+1] = img[jj+1][ii+1] = imgh[j][i];
                }
            }
        } // endscope
        mprove(img,rad,2);
    }
}
```

The routine for iterative improvement is called both by the approximate inverse (setting third argument = 2), and, subsequently, on its own (setting third argument = 1) to iterate to the exact inverse.

```
void mprove(Matrix<Doub> &img, Radon<Doub> &rad, Int method) {
    Int i,j,nn=img.nrows(),n2=2*nn;
    Matrix<Doub> img1(nn,nn);
    Radon<Doub> rad1(img);
    rad1 -= rad;
    if (method==1) {
        inverse_recursive(img1,rad1);
    } else if (method==2) {
        rad1.backproject(img1);
        hipass(img1);
    } else throw ("no such method in mprove");
    for (j=0;j<nn;j++) for (i=0;i<nn;i++) img[j][i] -= img1[j][i];
}
```

The method `hipass`, not listed here, merely implements the filter in the main text's Eq. 12.

References

1. Zubal I.G., Harrell C.R., Smith E.O., Rattner Z., Gindi G., & Hoffer P.B. (1994) "Computerized three-dimensional segmented human anatomy", *Med. Phys.*, **21**, 299-302.

Supplemental Figures

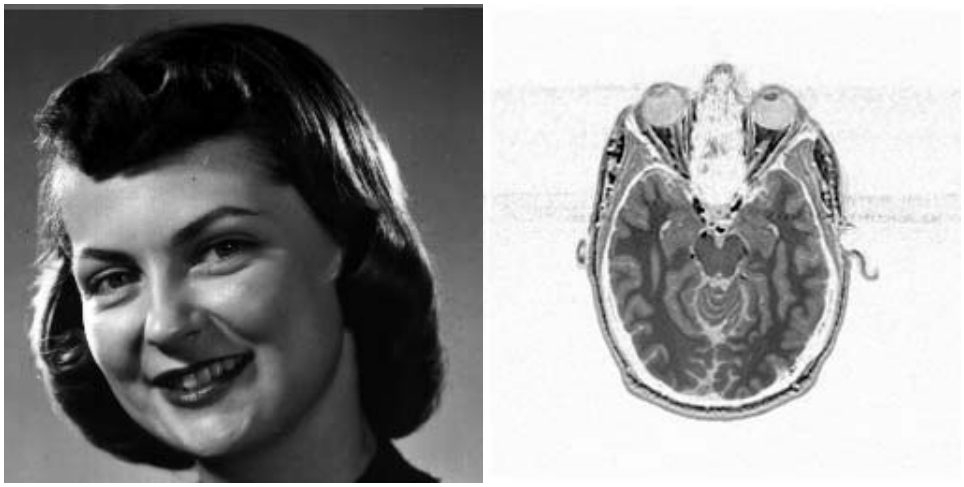

Figure 1: Test images used below. Both images are 256×256 pixels with 8 bit values. The right image is a standard head phantom MRI due to Zubal et al. (1) The horizontal bands of noise are a part of this image.

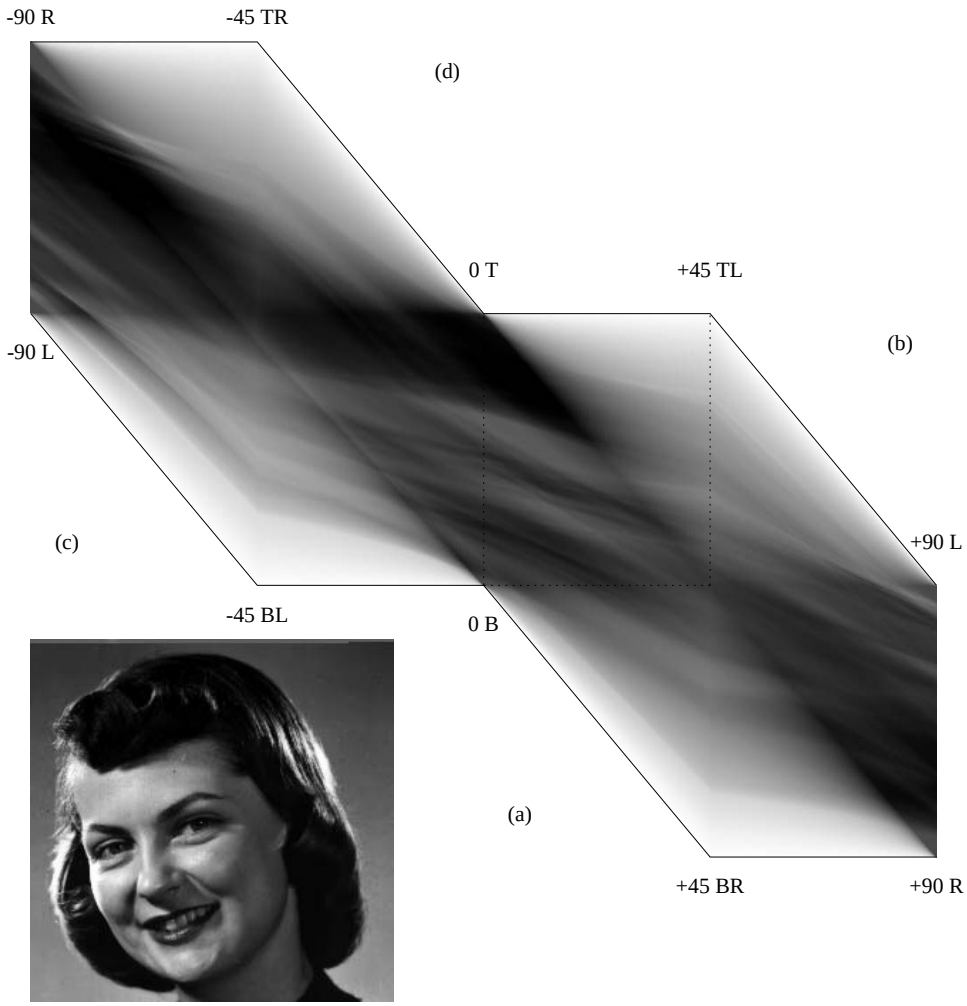

Figure 2: Radon transform of the test image shown at the lower left. The quadrants a, b, c, and d are labeled, with the labels positioned near their respective origins (smallest h and s). The notations T, B, R, and L refer to the top, bottom, right and left edges of the test image (cf. Figure 4 in main text).

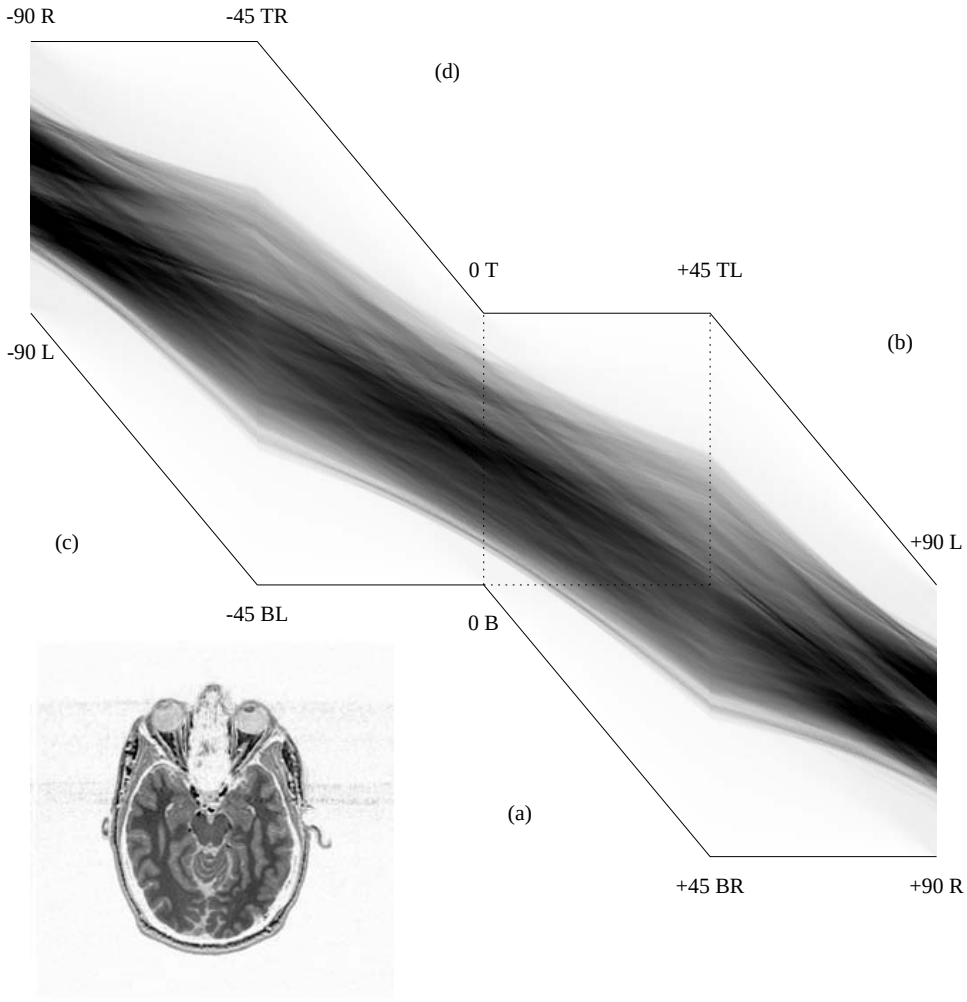

Figure 3: Same as Figure S2, but for the other test image.

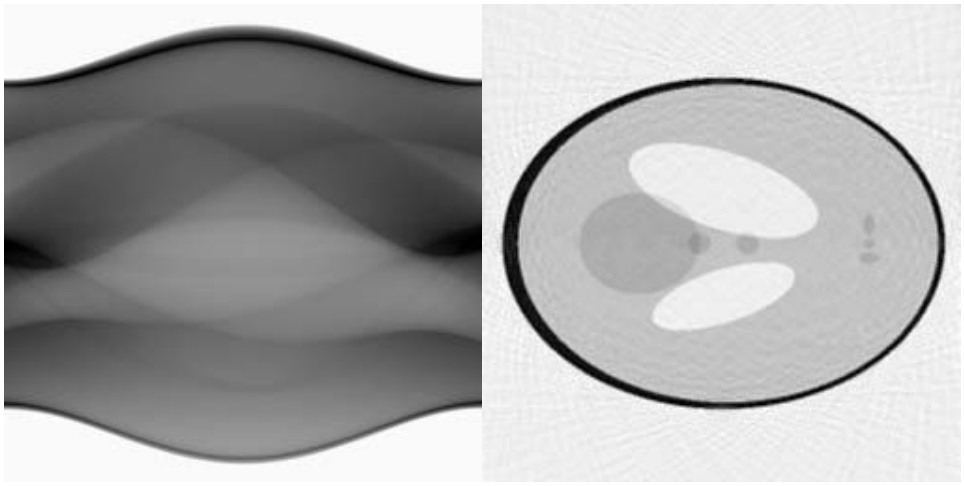

Figure 4: Sinogram of the standard Shepp-Logan phantom, captured as a 256×256 compressed GIF image; and image reconstructed by the methods of this paper. While the DRT inversion is exact, the sinogram is not exactly a DRT, so that artifacts are introduced. However, these are seen to be quite modest.

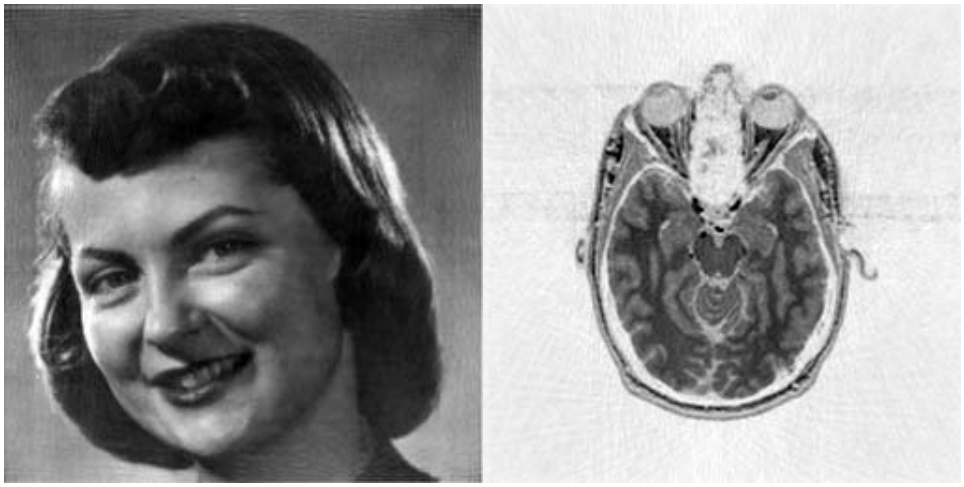

Figure 5: Approximate inverse of the DRTs in Supplemental Figures 2 and 3 by the "natural" Fourier method, with no filtering. The perceived image quality could be significantly improved by judicious filtering, but with a resulting loss of high-frequency information. This paper's exact inverse method does not have this deficiency, and does not use Fourier transforms.

Figure 6: Result of formally inverting an image from the data in a single quadrant (here a), after perturbing it with noise at a level 10^{-14}. The formal one-quadrant inverse is extremely ill-conditioned, and not useful in any practical way. The stripes are a Walsh function whose Fourier transform lies in a different quadrant.

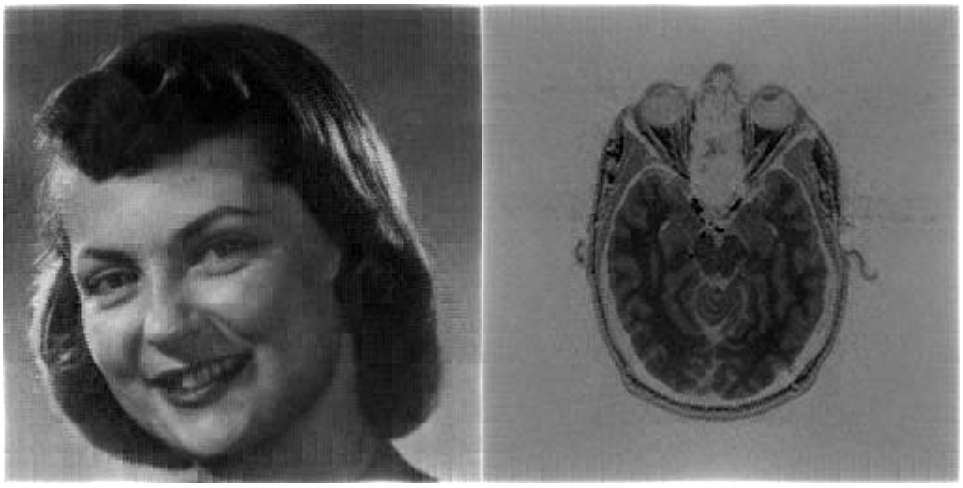

Figure 7: Approximate inverse of the DRTs in Supplemental Figures 2 and 3 by the recursive algorithm $\mathcal{B}$. While this inverse is perceptually inferior to the approximate Fourier inverse (Supplemental Figure S5), it has the important property that all of the eigenvalues of its residual matrix have magnitude < 1; this allows it to be iterated to an exact inverse.

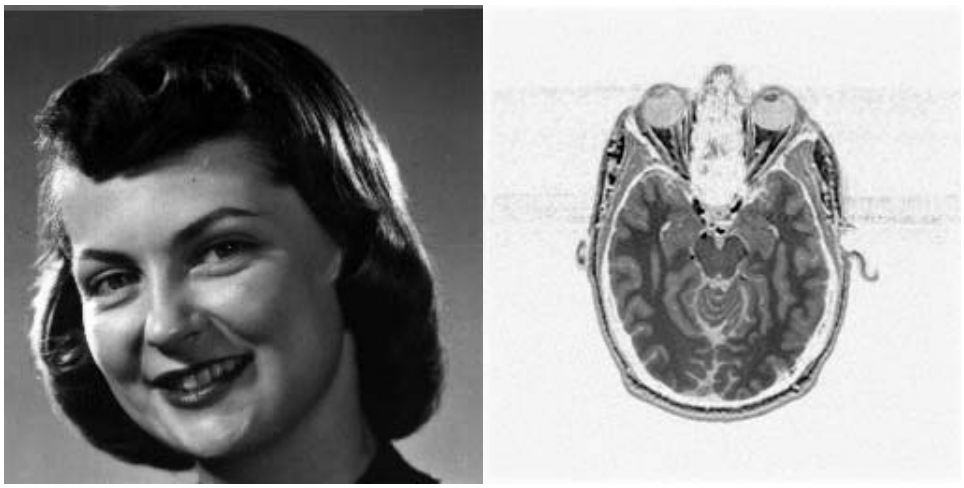

Figure 8: Images shown in Supplemental Figure S7 after four steps of iterative improvement. The r.m.s. errors at this stage are about 1%, still barely perceptable. Futher iterations reduce the errors to arbitrarily small.

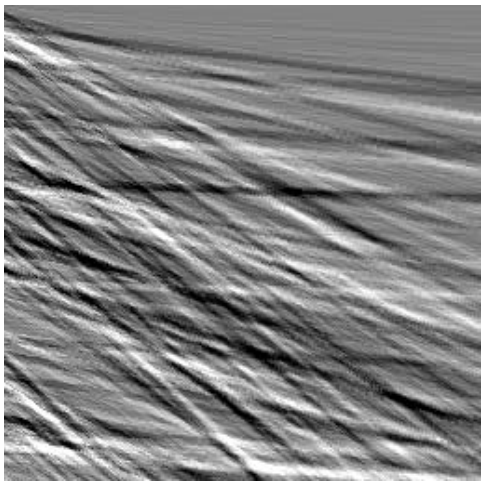

Figure 9: Ordinary DRT of the image shown in Figure 6(a). The region of transform space shown here is the same as in Figure 6(b). For the DRT, the weak line feature is lost in the confusion of sinusoids due to the pointlike clutter, in contrast to Figure 6(b) where the DMRT is seen to be insensitive to the clutter.

Figure 10: Result of taking the DMRT median transform of the test image S1 and then inverting it as if it were a DRT, not a DMRT. All of the detail shown is thus present in the median values of the original image along lines.

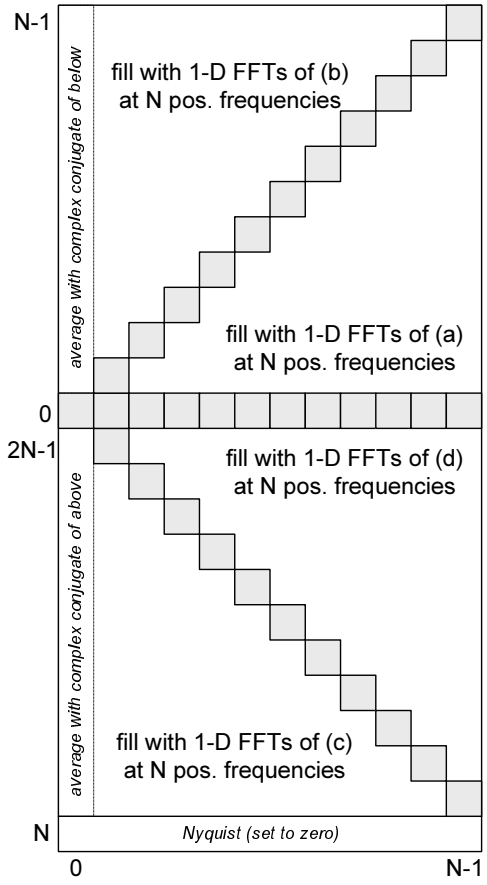

Figure 11: Positive-frequency half of two-dimensional Fourier space. Each quadrant in the DRT contributes values in a 45° wedge, as shown. Grey regions get contributions from two neighboring DRT quadrants. The transform of columns of the DRT are directly averaged into the d-lines of Fourier space, without interpolation.

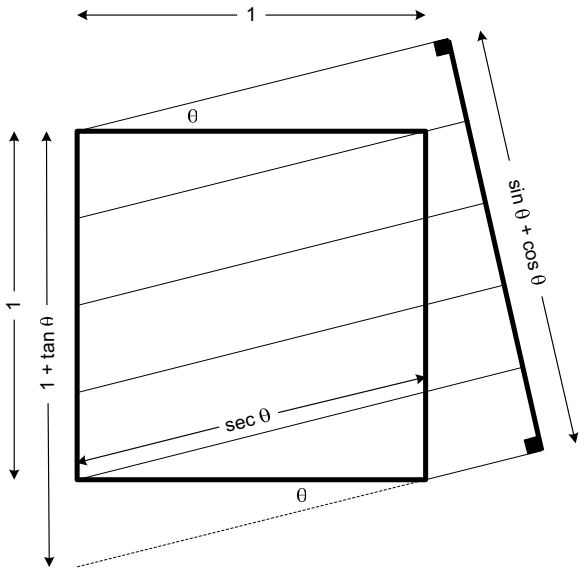

Figure 12: Trigonometric relationships between a square image and its projection at an angle θ.