THE PSEUDOPOLAR FFT AND ITS APPLICATIONS

AMIR AVERBUCH*, RONALD COIFMAN†, DAVID DONOHO‡, MOSHE ISRAELI§, AND JOHAN WALDÉN¶

Abstract. We present a two- and three-dimensional Fast Fourier Transform, where the result in the 2-D dimensional case is contained in coordinates that are "close to polar," while in the 3-D case they are "close to spherical." We call these transforms the *pseudopolar FFT* and *pseudospherical FFT*, repsectively. The move from Cartesian coordinates to polar coordinates is algebraically accurate and no interpolation is needed. We give examples of applications to computer tomography.

Key words. Fast Fourier Transform, Radon transform, computer tomography, Fractional Fourier Transform

AMS subject classifications. 44A12, 65R10, 65T20, 92C55

1. Introduction.

There are several applications where we would like to know the Fourier transform of a function, in polar coordinates. One important application is if we want to do computer tomography. The Fourier slice theorem establishes a relationship between the Fourier transform in polar coordinates and the Radon transform of the function. Another example is when we analyze wave front sets, where we are interested in "cutting out cones" of the Fourier transform of a function.

Ideally, we would like to have a version of the Fast Fourier Transform (FFT) that produces a result in polar coordinates. As points in polar coordinates are unequally spaced (compared with our original Cartesian data), one approach would be to use an unequally spaced FFT (see [2, 4, 5, 9]) to get the result in polar coordinates. With such an approach we make a trade-off between accuracy and complexity. If we want an algebraic method, we lose the "fastness" and the complexity will be $\mathcal{O}(n^4)$ (compared with $\mathcal{O}(n^2 \log n)$ for a two-dimensional FFT), for an $n \times n$ image.

In this paper we present an algebraic method that uses the fractional FFT (the FRFT) and is $\mathcal{O}(n^2 \log n)$. We accomplish this by redefining the grid points as in Figure 1.1. We call the new grid a *pseudopolar grid* and the transform the *pseudopolar FFT*.

We note that the pseudopolar grid is close to the polar grid in the following senses:

- As in polar coordinates, we get transformed values along rays passing through the origin. Furthermore, Δr (the distance between two points in the radial direction) is constant along each ray, permitting us to perform an FFT (or FRFT) in this direction which, by the Fourier slice theorem, gives us the Radon transform.
- Unlike in polar coordinates, in pseudopolar coordinates, Δr is different for different rays. However, it varies slowly, and we have $\Delta r_{\max}/\Delta r_{\min} = \sqrt{2}$.
- Similarly, the change in the angular direction, $\Delta\Theta$, varies slowly, and we have $\Delta\Theta_{\max}/\Delta\Theta_{\min} \leq 2$.

*School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel (amir@math.tau.ac.il).

†Yale University, Department of Mathematics, 10 Hillhouse Avenue, P.O. Box 208283, New Haven, Connecticut 06520 (coifman@math.yale.edu).

‡Stanford University, Department of Statistics, Stanford, CA 94305 (donoho@stat.stanford.edu).

§Faculty of Computer Science, Technion, Haifa 32000, Israel (israeli@cs.technion.ac.il).

¶Yale University, Department of Mathematics, 10 Hillhouse Avenue, P.O. Box 208283, New Haven, Connecticut 06520 (jwalden@math.yale.edu).

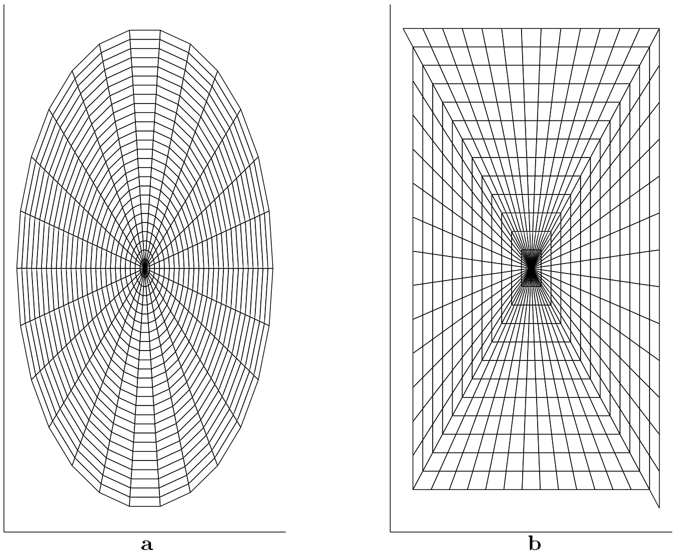


FIG. 1.1. *Polar grid a) and Pseudopolar grid b).*

- As when working in polar coordinates, it is simple to "cut out" cones in the transformed space.

Thus, the pseudopolar FFT has many properties in common with a polar transform.

The rest of the paper is organized as follows. In Section 2 we introduce some preliminary notation. In Section 3 we define the transform, derive an error estimate, and show some numerical examples. We also define a three-dimensional transform. Finally, in Section 4 we construct an algorithm to calculate the inverse transform. In particular, we show numerical experiments for the inverse Radon transform, which can be used to perform computer tomography.

2. Preliminaries. We define the continuous Fourier transform of a function, $f \in L^1(R^N)$ by

$$\hat{f}(\xi) = \int f(x) e^{-i\langle \xi, x\rangle}\, dx.$$

We will work mainly with discrete functions (vectors) of length n (where $n = 2^p$, p integer). We denote the kth element of a vector, u by u_k or by $u(k)$. It will be natural for us to let indices of vectors be centered around zero, i.e., $k \in \{-n/2, -n/2+1, \ldots, n/2-1\}$.

We now define the Discrete Fourier Transform (the DFT):

$$(2.1) \qquad (Fu)(k) \stackrel{\text{def}}{=} \sum_{j=-n/2}^{n/2-1} u_j e^{-i2\pi jk/n}, \qquad k = -n/2, -n/2+1, \ldots, n/2-1.$$

We can compute the DFT with $5n \log n$ arithmetic operations by using the Fast Fourier Transform (FFT) [3]. Here, and in the rest of the paper, $\log n$ means $\log_2 n$.

The Fractional FFT (the FRFT) is defined by

$$(F_\alpha u)(k) \stackrel{\text{def}}{=} \sum_{j=-n/2}^{n/2-1} u_j e^{-i\alpha 2\pi jk/n}, \qquad k=-n/2,-n/2+1,\dots,n/2-1,\ \alpha\in[-1,1], \tag{2.2}$$

(note that, again we have arguments between $-n/2$ and $n/2-1$ instead of the more common 1 and n). The FRFT can be computed with $20n\log n$ operations (disregarding lower order terms) [1]. If we need to stress the size of the transform, we write F^n and F^n_α, respectively.

We denote the identity operator by I and define the zero-padding operator, $E: C^n \to C^{2n}$ (or, if we want to stress the size, I^n, E^n) by

$$(Eu)(k) \stackrel{\text{def}}{=} \begin{cases} 0, & -n \le k < -n/2, \\ u_k, & -n/2 \le k < n/2-1, \\ 0, & n/2 \le k < n. \end{cases} \tag{2.3}$$

More generally, we can zero-pad m times, giving

$$(E_m u)(k) \stackrel{\text{def}}{=} \begin{cases} 0, & -mn/2 \le k < -n/2, \\ u_k, & -n/2 \le k < n/2-1, \\ 0, & n/2 \le k < mn/2. \end{cases} \tag{2.4}$$

We define the diagonal operator $diag(\lambda_{-n/2}, \lambda_{-n/2+1}, \dots, \lambda_{n/2-1})$ by

$$(diag(\lambda_{-n/2}, \lambda_{-n/2+1}, \dots, \lambda_{n/2-1})u)(k) \stackrel{\text{def}}{=} \lambda_k u_k. \tag{2.5}$$

In particular, we will use the ramp-functions,

$$D \stackrel{\text{def}}{=} diag(n/2, n/2-1, \dots, 2, 1, 1/4, 1, 2, \dots, n/2-1), \tag{2.6}$$

$$\sqrt{D} \stackrel{\text{def}}{=} diag(\sqrt{n/2}, \sqrt{n/2-1}, \dots, \sqrt{2}, \sqrt{1}, \sqrt{1/4}, \sqrt{1}, \sqrt{2}, \dots, \sqrt{n/2}), \tag{2.7}$$

and

$$\begin{aligned} D_\alpha \stackrel{\text{def}}{=} diag(&\sqrt{1+(n/n)^2}, \sqrt{1+((n-2)/n)^2}, \dots, \sqrt{1+(2/n)^2}, \\ &\sqrt{1+(0/n)^2}, \sqrt{1+(2/n)^2}, \dots, \sqrt{1+((n-2)/n)^2}), \end{aligned} \tag{2.8}$$

(also denoted D^n, $\sqrt{D^n}$, and D^n_α). Note the 1/4 factor for the zeroth coordinate of D. It is needed as we will otherwise "miss" the zero-frequency component when we define the inverse pseudopolar FFT.

We will be concerned with two-dimensional discrete functions $w \in C^{n\times n}$, and we have the tensor product of two one-dimensional functions $(u\otimes v)(j,k) = u(j)v(k)$, $-n/2 \le j,k, \le n/2-1$. Here, j is the argument along the x-axis, and k along the y-axis. We also have the restriction operators that "pick out lines" from two-dimensional discrete functions: $(R^x_k u)(j) = u(j,k)$ and $(R^y_k u)(j) = u(k,j)$ for all u and v.

As we will work with two- and three-dimensional discrete functions, it is more straightforward to define linear algebraic operators directly, *not* using a matrix interpretation, which is artificial in higher dimensions. For linear operators, we use the standard l^2 inner product to define the adjoint operation, *. We define the direct sum of two operators, $G: L_1 \to L_2$ and $H: L_1 \to L_3$ as $(G\oplus H)u = (Gu)\oplus(Hu)$, and for $G: L_1 \to L_3$ and $H: L_2 \to L_3$ as $(G\oplus H)(u\oplus v) = Gu + Hv$. These definitions are similar to vertical and horizontal concatenation of matrices in the one-dimensional case. We also define the tensor product of two linear operators, $G\otimes H$ as the linear operator, such that $(G\otimes H)(u\otimes v) = (Gu)\otimes(Hv)$ for all u and v.

3. The Pseudopolar Transforms.

3.1. The Pseudopolar FFT. We shall now define the pseudopolar FFT in a way such that it can be computed with a sequence of fast operations. The idea is to view the pseudopolar grid as a union of two subgrids as in Figure 3.1. We start

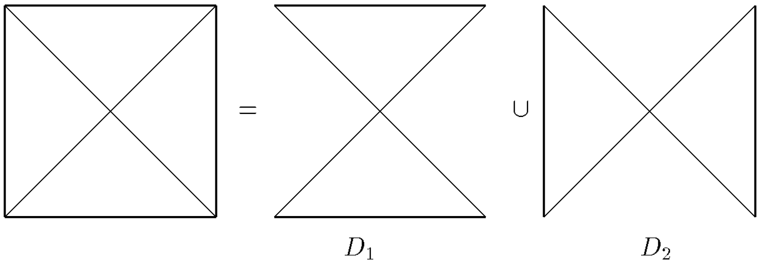


FIG. 3.1. *Pseudopolar grid viewed as union of two grids.*

with the left of these subgrids, D_1. In the untransformed domain, we have a two-dimensional discrete function, $u \in C^{n\times n}$. We zero-pad the function and perform an FFT in the y-direction, forming $(I^n \otimes (F^{2n}E^n))u$. In the x-direction, we use the FRFT to perform a transform evaluating with an α that decreases as we get closer to the center. We call this operation Z, and it is defined by

$$(Zu)(j,k) \stackrel{\text{def}}{=} (F^n_{-k/n}(R^x_k u))(j),$$

where we have chosen the letter Z as it looks like the domain D_1 we are working on. The total operation is defined by

$$\mathcal{Z}(j,k) \stackrel{\text{def}}{=} \frac{4}{n^2}\Big(Z(I^n \otimes (F^{2n}E^n))\Big).$$

Similarly, for the second subgrid we do the zero-padding and FFT in the x-direction, followed by an FRFT part in the y-direction. We call this operation N,

$$(Nu)(j,k) \stackrel{\text{def}}{=} (F^n_{j/n}(R^y_j u))(k),$$

and we define

$$\mathcal{N} \stackrel{\text{def}}{=} \frac{4}{n^2}\Big(N((F^{2n}E^n)\otimes I^n)\Big).$$

If we put all the steps together, we get:

$$\mathcal{F} \stackrel{\text{def}}{=} \mathcal{Z} \oplus \mathcal{N} = \frac{4}{n^2}\Big(Z(I\otimes(FE))\Big) \oplus \Big(N((FE)\otimes I)\Big). \tag{3.1}$$

We call this the *Pseudopolar FFT*, and the following remarks are valid:

- The transform takes an $n \times n$ grid function to a $2n \times 2n$. If $u(k_1,k_2) = f(2k_1/n, 2k_2/n)$ is a discretization of a function, f, then $(\mathcal{F}u)(j_1,j_2)$ approximates $\hat{f}$ in the following way:

$$(\mathcal{Z}u)(j_1,j_2) \approx \hat{f}(-\pi j_1 j_2/n, \pi j_2/2),$$

$$(\mathcal{N}u)(j_1,j_2) \approx \hat{f}(\pi j_1/2, \pi j_1 j_2/n).$$

  We shall show this shortly.

- The transform is made up as a composition of fast operations, and can be computed with $100n^2 \log n$ arithmetic operations (disregarding lower-order terms). If u is real, the transform can be computed with $50n^2 \log n$ operations.
- Except for Z and N, all operations are one-dimensional (i.e., tensor products of identity and one-dimensional operator).
- The transform performs the following operation

$$(\mathcal{Z}u)(j_1, j_2) = \frac{4}{n^2} \sum_{k_1=-n/2}^{n/2-1} \sum_{k_2=-n/2}^{n/2-1} e^{-\frac{i\pi}{n}(j_2 k_2 - 2 j_2 j_1 k_1/n)} u(k_1, k_2),$$
$$-n/2 \le j_1 \le n/2 - 1, \quad -n \le j_2 \le n-1,$$

and

$$(\mathcal{N}u)(j_1, j_2) = \frac{4}{n^2} \sum_{k_1=-n/2}^{n/2-1} \sum_{k_2=-n/2}^{n/2-1} e^{-\frac{i\pi}{n}(j_1 k_1 + 2 j_1 j_2 k_2/n)} u(k_1, k_2),$$
$$-n \le j_1 \le n-1, \quad -n/2 \le j_2 \le n/2 - 1.$$

Remark. In our implementation of the algorithm, the result is stored in a $2n \times 2n$ matrix, A, where $A_{j,k}$ is the kth element along the jth angle.

We now show an approximation theorem.

THEOREM 3.1. *If u is a discretization of a function, f (i.e., $u(j_1, j_2) = f(2j_1/n, 2j_2/n)$, $-n/2 \le j_1, j_2 \le n/2 - 1$) and $f \in C_0^{2m+2}((-1,1) \times (-1,1))$ (i.e., f has its support inside $(-1,1) \times (-1,1)$ and $\frac{\partial^q \partial^r}{\partial x^q \partial y^r} f$ is a continuous function for $r + q \le 2m+2$), then*

$$|(\mathcal{Z}u)(j_1, j_2) - \hat{f}(\pi j_1 j_2/n, \pi j_2/2)| \le C \left(\frac{(1 + |j_1 j_2/n|)(1 + |j_2/n|)}{n^2} \right)^{m+1},$$

$$|(\mathcal{N}u)(j_1, j_2) - \hat{f}(\pi j_1/2, \pi j_1 j_2/n)| \le C \left(\frac{(1 + |j_1|)(1 + |j_1 j_2/n|)}{n^2} \right)^{m+1},$$

where C depends on f but not on j_1, j_2 or n. Thus, the pseudopolar FFT gives an approximation of the Fourier transform (in pseudopolar coordinates), that pointwise is of order $2m+2$.

Proof. Similarly to a standard way of proving that the FFT gives a high-order approximation of the Fourier transform for functions with compact support, we use the Euler–MacLaurin summation formula.

We prove the theorem for the $\mathcal{N}$ part. The proof for the $\mathcal{Z}$ part is completely similar. We start by defining

$$g(x, y, j_1, j_2) \stackrel{\text{def}}{=} f(x, y) e^{-i\pi(j_1 x/2 + j_1 j_2 y/n)}, \tag{3.2}$$

and we note that $g(\cdot, \cdot, j_1, j_2)$ is a periodic function on $(-1,1) \times (-1,1)$ (as f's support is contained within $(-1,1) \times (-1,1)$). We define $\Delta x = \Delta y = 2/n$. Now, (3.2) together with using the Euler–MacLaurin summation formula in both the x and y direction gives us

$$
\begin{aligned}
(\mathcal{F}u)(j_1,j_2) &= \sum_{k_2=-n/2}^{n/2-1} \Big(\Delta y \sum_{k_1=-n/2}^{n/2-1} \Delta x\, g(k_1\Delta x, k_2\Delta y, j_1, j_2) \Big) \\
&= \int_{-1}^{1}\int_{-1}^{1} g(x,y,j_1,j_2)\, dx\, dy + (\Delta x)^{m+1}(\Delta y)^{m+1} \\
&\quad \times \int_{-1}^{1} \tilde{P}_{j+1}(nx/2) \frac{\partial^{m+1}}{\partial x^{m+1}} \Big(\int_{-1}^{1} \tilde{P}_{j+1}(ny/2) \frac{\partial^{m+1} g}{\partial y^{m+1}}(x,y,j_1,j_2)\, dy \Big)\, dx. \qquad (3.3)
\end{aligned}
$$

Here, $\tilde{P}_r(z)$ is defined as $P_r(z - \lfloor z \rfloor - 1/2)$ where P_r is the rth-order Bernoulli polynomial.

Now, the first term on the right-hand side of (3.3) evaluates to the Fourier transform we are looking for. As g is smooth, and we are integrating over a compact interval, we can change the order of integration and differentiation for the second term to get

$$
\begin{aligned}
e \stackrel{\text{def}}{=} (\mathcal{F}u)(j_1,j_2) - \hat{f}(\pi j_1/2, \pi j_1 j_2/n) \\
&= (\Delta x \Delta y)^{m+1} \int_{-1}^{1}\int_{-1}^{1} \tilde{P}_{j+1}(ny/2)\tilde{P}_{j+1}(nx/2) \\
&\qquad \times \frac{\partial^{2m+2} g}{\partial y^{m+1}\partial x^{m+1}}(x,y,j_1,j_2)\, dy\, dx.
\end{aligned}
$$

We use Hölder's inequality for the right-hand side and get

$$
\begin{aligned}
|e| &\le (\Delta x \Delta y)^{m+1} \Big\| \tilde{P}_{j+1}(nx/2)\tilde{P}_{j+1}(nx/2) \Big\|_1 \Big\| \frac{\partial^{2m+2} g}{\partial y^{m+1}\partial x^{m+1}}(\cdot,\cdot,j_1,j_2) \Big\|_\infty \\
&\le Const (\Delta x \Delta y)^{m+1} \Big\| \frac{\partial^{2m+2} g}{\partial y^{m+1}\partial x^{m+1}}(\cdot,\cdot,j_1,j_2) \Big\|_\infty. \qquad (3.4)
\end{aligned}
$$

Finally, we use Leibniz' formula to expand $\frac{\partial^{2m+2} g}{\partial x^{m+1}\partial y^{m+1}}$ and get

$$
\begin{aligned}
&\Big\| \frac{\partial^{2m+2} g}{\partial x^{m+1}\partial y^{m+1}}(\cdot,\cdot,j_1,j_2) \Big\|_\infty \\
&\quad \le \sum_{q=0}^{m+1}\sum_{r=0}^{m+1} C_{q,r} \Big\| \frac{\partial^{r+q}}{\partial x^r \partial y^q} e^{-i\pi(j_1 x/2 + j_1 j_2 y/n)} \Big\|_\infty \Big\| \frac{\partial^{2m+2-r-q}}{\partial x^{m+1-r}\partial y^{m+1-q}} f(x,y) \Big\|_\infty \\
&\quad \le Const (1+|j_1|)^{m+1}(1+|j_1 j_2/n|)^{m+1} \max_{0\le q,r\le m+1} \Big\| \frac{\partial^{r+q}}{\partial x^r \partial y^q} f(x,y) \Big\|_\infty. \qquad (3.5)
\end{aligned}
$$

Now, by replacing Δx and Δy with $2/n$, and inserting (3.5) into (3.4) we arrive at

$$
|e| \le \left(\frac{(1+|j_1|)(1+|j_1 j_2/n|)}{n^2} \right)^{m+1},
$$

and we have proved the theorem.

□

Remark. If we want a higher resolution in Fourier space, we zero-pad u before transforming. A straightforward generalization of the previous theorem gives

$$|(\mathcal{Z}(E^n_{s_1} \otimes E^n_{s_2})u)(j_1, j_2) - \hat{f}(\pi j_1 j_2/(n s_1)), \pi j_2/(2 s_2))| \leq C \left(\frac{(1 + |j_1|)(1 + |j_1 j_2/n|)}{n^2 s_1 s_2} \right)^{m+1},$$

$$|(\mathcal{N}(E^n_{s_1} \otimes E^n_{s_2})u)(j_1, j_2) - \hat{f}(\pi j_1/(2 s_1)), \pi j_1 j_2/(n s_2)| \leq C \left(\frac{(1 + |j_1 j_2/n|)(1 + |j_2|)}{n^2 s_1 s_2} \right)^{m+1}.$$

We compute the pseudopolar FFT of a discretized Gaussian, $f(x, y) = e^{-200((x-0.1)^2+(y-0.05)^2)}$ for $n = 32 - 256$. The program is written in C, compiled with SUN's compiler with the **-fast** flag, and run on an UltraSPARC 1. All computations in this and following examples are carried out in double precision. The results are shown in Table 3.1. We see that the error (the difference between the pseudopolar FFT and the Fourier transform of f) is of machine precision, for the size 128×128, and that the method clearly is of complexity $\mathcal{O}(n^2 \log n)$.

n	$\|e\|_\infty$	CPU (s)
32	6.67×10^{-4}	0.01
64	5.12×10^{-8}	0.05
128	1.37×10^{-16}	0.24
256	2.25×10^{-16}	1.16

TABLE 3.1

Error and CPU-time for approximation of Fourier transform by pseudopolar FFT for Gaussian.

3.2. The Radon Transform. The Radon transform of a function is defined as

$$p(r, \theta) \stackrel{\text{def}}{=} \int_{L(r,\theta)} f(x, y)\, ds = \int\int f(x, y) \delta(x \cos\theta + y \sin\theta - r)\, dx\, dy,$$
$$-\infty < r < \infty, \qquad 0 \leq \theta < \pi,$$

which is interpreted as the line integral along the ray at distance r and angle θ. By the Fourier slice theorem, it is know that

$$\hat{f}(\xi_1, \xi_2) = \hat{p}(\omega, \theta), \qquad \xi_1 = \omega \cos\theta,\ \xi_2 = \omega \sin\theta, \tag{3.6}$$

where $\hat{p}$ is the (one-dimensional) Fourier transform of p in the radial direction, and $\hat{f}$ is the (two-dimensional) Fourier transform of f. Thus, if we have the Fourier transform in some radial (e.g. pseudopolar) coordinates, we can find the Radon transform by computing a one-dimensional inverse Fourier transform along each ray.

We could perform an FFT along each ray, but as the spacing Δr is different along different rays, the result would not be equidistant for different rays. Instead, we use an FRFT with an appropriate scaling factor along each ray to correct this. Thus, we define

$$(\Lambda_{\mathcal{Z}} u)(j, k) \stackrel{\text{def}}{=} \left(F^{2n}_\alpha (R^y_k u)\right)(j),$$
$$\alpha = \sqrt{\frac{1 + (2k/n)^2}{2}}, \quad -n \leq j \leq n - 1, \ -n/2 \leq k \leq n/2 - 1,$$

and

$$(\Lambda_{\mathcal{N}} u)(j,k) \stackrel{\text{def}}{=} \left(F^{2n}_{-\alpha}(R^x_j u)\right)(k), \tag{3.7}$$

$$\alpha = \sqrt{\frac{1+(2j/n)^2}{2}}, \quad -n/2 \le j \le n/2-1, \ -n \le k \le n-1.$$

The total pseudopolar Radon transform is defined as,

$$\mathcal{R} \stackrel{\text{def}}{=} \left(\mathcal{R}_{\mathcal{Z}} \oplus \mathcal{R}_{\mathcal{N}}\right), \qquad \text{where,} \tag{3.8}$$

$$\mathcal{R}_{\mathcal{Z}} = \frac{1}{4}(D^n_\alpha \otimes I^{2n})\Lambda_{\mathcal{Z}}\mathcal{Z}, \qquad \text{and} \qquad \mathcal{R}_{\mathcal{N}} = \frac{1}{4}(I^{2n} \otimes D^n_\alpha)\Lambda_{\mathcal{N}}\mathcal{N}.$$

Here, D^n_α (defined in (2.8)) appears as we have different $\Delta\omega$ along different rays, when we approximate the inverse Fourier transform in (3.6). With this definition, and the discretization $u(k_1,k_2) = f(2k_1/n, 2k_2/n)$, $-n/2 \le k_1, k_2 \le n/2-1$, we will have the approximation

$$(\mathcal{R}_{\mathcal{Z}} u)(j_1, j_2) \approx p(j_1 \Delta r, \theta_{j_2}),$$

with

$$\Delta r = \frac{\sqrt{2}}{n}, \qquad \text{and} \qquad \theta_{j_2} = \ -\pi/2 + \tan^{-1}(2j_2/n),$$

$$-n \le j_1 \le n-1, \qquad -n/2 \le j_2 \le n/2-1,$$

and similarly for $\mathcal{R}_{\mathcal{N}}$, but with

$$\theta_{j_2} = \ \pi + \tan^{-1}(2j_2/n).$$

Assuming that f is a real function, we can compute the pseudopolar Radon transform with $130n^2 \log n$ operations (disregarding lower-order terms and using the symmetries that arise when f is real). We compute the pseudopolar Radon transform of the Shepp–Logan phantom (which can be seen as a model of a head and, e.g., is defined in [8]). The Shepp–Logan phantom, shown in Figure 3.2 is defined as a sum of characteristic functions on ellipses, and the Radon transform can be calculated analytically. In Table 3.2 we show the error in maximum norm and the CPU-time for sizes $32 - 256$. We see that we have first-order convergence in 2-norm and 1-

n	$\|\|e\|\|_1$	$\|\|e\|\|_2$	$\|\|e\|\|_\infty$	CPU (s)
32	3.6×10^{-2}	5.9×10^{-2}	0.41	0.03
64	1.5×10^{-2}	2.9×10^{-2}	0.25	0.13
128	7.5×10^{-3}	1.5×10^{-2}	0.21	0.55
256	3.3×10^{-3}	7.0×10^{-3}	0.15	2.76

TABLE 3.2

Error and CPU-time for approximation of Radon transform by pseudopolar transform for Shepp–Logan phantom.

norm, whereas it is slower in ∞-norm. This comes from the Shepp–Logan phantom having low regularity (it is not even continuous). In Figure 3.3 we show the exact and the approximated transform along one line, $\mathcal{R}_{\mathcal{N}} f(\cdot, 0)$. We see that we have a good approximation except where p has a sharp derivative. This explains the poor performance in ∞-norm.

FIG. 3.2. *Shepp–Logan phantom. The function varies between* 0 *(black) and* 2 *(white). The part inside the "skull" varies between* 1 *and* 1.04*.*

3.3. The Three-Dimensional Transform. We can use the same idea to define a transform in three dimensions. We have three directions of doing the transform, each consisting of two pyramids, as shown in Figure 3.4. In Table 3.3 we show the error when performing this transform for (a discretization of) the function $f(x, y, z) = e^{-30((x-0.1)^2+(y-0.05)^2+z^2)}$. We call this transform the *pseudospherical FFT*.

n	$\|\|e\|\|_\infty$
8	0.47
16	0.34
32	2.1×10^{-3}
64	2.5×10^{-12}

TABLE 3.3
Error for approximation of Fourier transform by pseudospherical FFT for Gaussian.

4. Inverse Transforms.

4.1. A First Approximation. We wish to compute the inverse of the pseudopolar FFT. A good starting point is Figure 3.1. From the continuous theory, we know that

$$f(x, y) = \frac{1}{2\pi}\int_{D_1} \hat{f} e^{i\langle x,y\rangle\langle\omega_1,\omega_2\rangle} + \frac{1}{2\pi}\int_{D_2} \hat{f} e^{i\langle x,y\rangle\langle\omega_1,\omega_2\rangle}.$$

Now, the discrete version of these integrals will be to perform a FRFT in one direction, with a scaling factor corresponding to $\Delta\omega$. In the other direction, we perform an

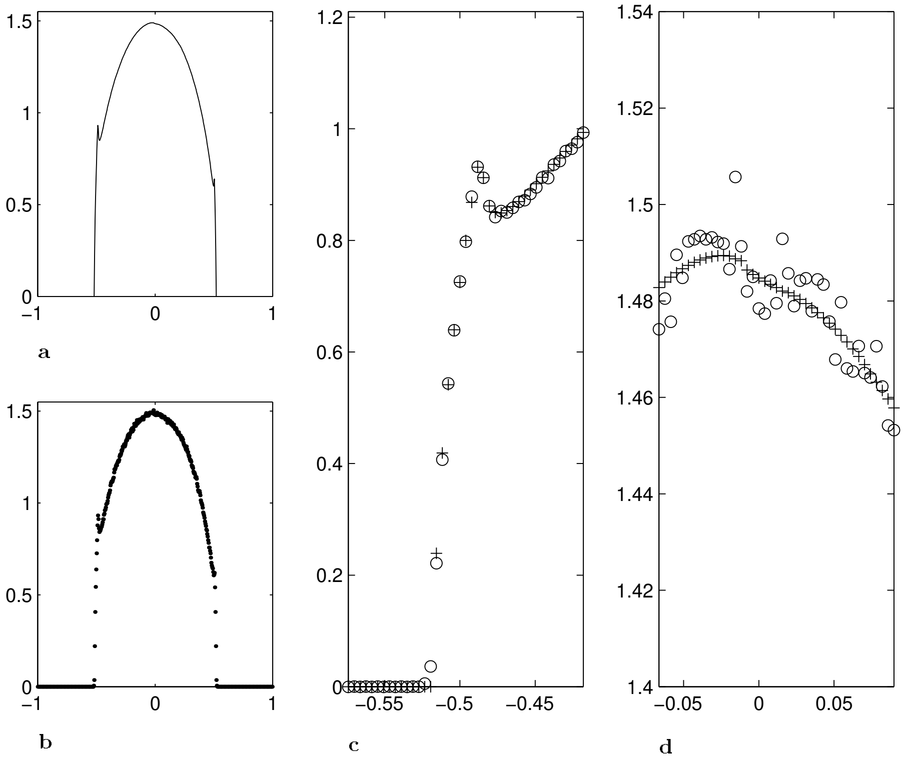


FIG. 3.3. *a) Radon transform of Shepp–Logan phantom $\mathcal{R}f(\cdot,0)$, b) Pseudopolar transform, c) blow-up around left edge of skull (exact is '+', and approximation is 'o'), d) blow-up in smooth region.*

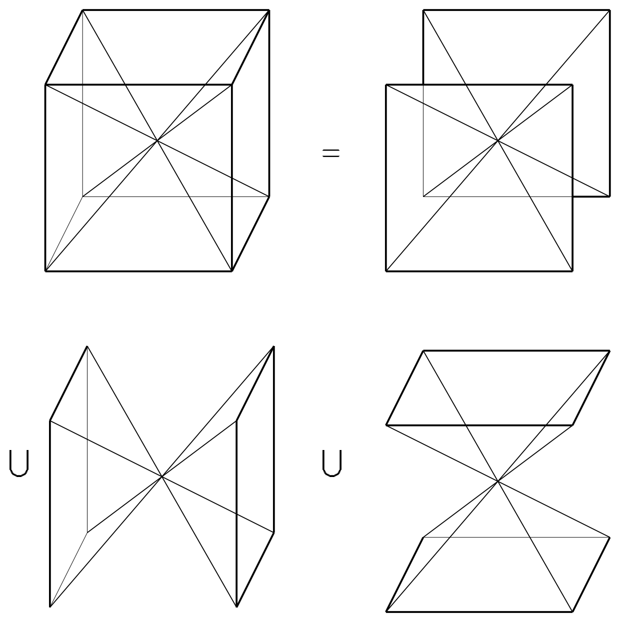


FIG. 3.4. *Transform defined in three dimensions.*

inverse FFT. The total approximation is defined as:

$$\tilde{\mathcal{F}} = \frac{1}{8n}\Big(((E^*F^*)\otimes I)Z^*(I\otimes D)\Big) \oplus \Big((I\otimes (E^*F^*))N^*(D\otimes I)\Big), \tag{4.1}$$

(with D defined in (2.6)) and for the Radon transform:

$$\tilde{\mathcal{R}} = \frac{\sqrt{2}}{8n^2}\Big(\Big(((E^*F^*)\otimes I)Z^*(I\otimes D)\Big)\Lambda_{\mathcal{Z}}^*\Big) \oplus \Big(\Big((I\otimes (E^*F^*))N^*(D\otimes I)\Big)\Lambda_{\mathcal{N}}^*\Big). \tag{4.2}$$

We note that the complexity for these transforms basically are the same as for the forward transforms.

We test the approximative inverse pseudopolar FFT. The result is shown in Table 4.1. The results are similar for the pseudopolar Radon transform. We see that the error is 3-5 % and nondecreasing in all norms. Thus, the approximative inverses are not very exact.

n	$\|\|e\|\|_1$	$\|\|e\|\|_2$	$\|\|e\|\|_\infty$
32	2.9×10^{-2}	3.0×10^{-2}	3.7×10^{-2}
64	2.9×10^{-2}	3.0×10^{-2}	3.9×10^{-2}
128	3.0×10^{-2}	3.1×10^{-2}	4.3×10^{-2}

TABLE 4.1
Error and CPU-time for inverse approximation $\tilde{\mathcal{F}}$ for different problem sizes.

We can understand that this will be the case by investigating the eigenvalues of the FRFT. The continuous transform, $F^{-1}((Ff)\times\chi_D)$ has eigenvalues that are either one (for frequencies inside the domain of integration) or zero (for frequencies outside). In Figure 4.1 we have ordered the eigenvalues of $F_{1/2}^*F_{1/2}/128$, for the 64-point FRFT. Ideally, we would like to have 32 eigenvalues equal to 1 and 32 equal to 0. However, there are some eigenvalues in a transition region, and the result is that our "cutting" of the total square into D_1 and D_2 makes the discrete approximation rather crude.

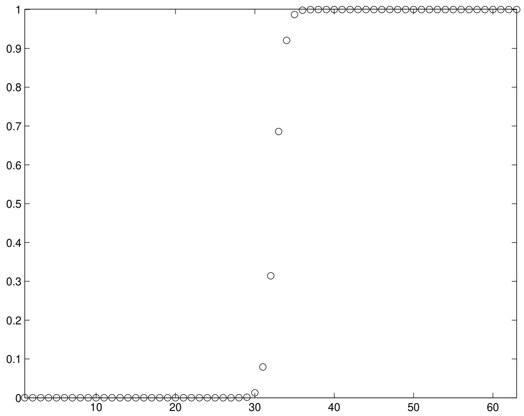


FIG. 4.1. *Eigenvalues for $F_{1/2}^*F_{1/2}/128$ and the 64-point FRFT.*

We therefore seek a way of improving the results.

4.2. An Iterative Approach. We may think of $\tilde{\mathcal{F}}$ as a preconditioner, and use an iterative algorithm to find an inverse. From the previous section, we expect the condition number, $\kappa(\tilde{\mathcal{F}}\mathcal{F})$ to be small, and we also have the property that the preconditioner works as a symmetrizer:

THEOREM 4.1. *$\tilde{\mathcal{F}}\mathcal{F}$ is Hermitian.*

Proof. Define

$$Q \stackrel{\text{def}}{=} \frac{1}{\sqrt{2n^3}}\Big((I \otimes \sqrt{D})Z((FE)\otimes I)\Big) \oplus \Big((\sqrt{D}\otimes I)N(I \otimes (FE))\Big).$$

Then, we directly get $Q^*Q = \tilde{\mathcal{F}}\mathcal{F}$ and as $(Q^*Q)^* = Q^*Q$ the theorem is proved. □

Remark. Another way of proving the theorem is to look at the elements of $\tilde{\mathcal{F}}\mathcal{F}$. We have:

$$(\tilde{\mathcal{F}}\mathcal{F}u)(s_1,s_2) = \frac{1}{2n^2}\Big(\sum_{k_1=-n/2}^{n/2-1}\sum_{k_2=-n/2}^{n/2-1} A(s_1,s_2,k_1,k_2)u(k_1,k_2)\Big), \tag{4.3}$$

with

$$\begin{aligned} A(s_1,s_2,k_1,k_2) = \sum_{j_1=-n/2}^{n/2-1}\sum_{j_2=-n}^{n-1} \Big(& d(j_2)e^{\frac{i\pi}{n}(j_2(s_2-k_2)+2j_2j_1(s_1-k_1)/n)} \\ &+ d(j_2)e^{\frac{i\pi}{n}(j_2(s_1-k_1)-2j_1j_2(s_2-k_2)/n)}\Big), \\ d(j) &= |j|, \qquad j \neq 0, \\ d(j) &= 1/4, \qquad j = 0. \end{aligned} \tag{4.4}$$

Remark. From (4.4) we see that $\tilde{\mathcal{F}}\mathcal{F}$ is a Toeplitz operator. We can use this to compute $\tilde{\mathcal{F}}\mathcal{F}$ in a more efficient way by embedding it in a circulant operator. Using this trick, $\tilde{\mathcal{F}}\mathcal{F}u$ can be evaluated with $80n^2 \log n$ operations, (instead of $200n^2 \log n$).

Remark. We can interpret the construction of Q as if we are computing a scaled pseudoinverse. The scaling is performed by introducing the ramp-filter, D. It comes natural, as points close to the origin, where the density of points is high, will have less influence on the result.

We use the Hermitian property to calculate the inverse with the Conjugate Gradient method. We do this for the pseudopolar FFT, the pseudopolar Radon and the pseudospherical FFT. In Figure 4.2 we have plotted the error in ∞-norm as a function of the number of iterations. We see that the preconditioning is nearly perfect. In particular, the error decreases to machine precision in few iterations (about 10 for the two-dimensional transforms and 20 for the three-dimensional) regardless of problem size. If we are content with an error of 10^{-5}, then 3-4 iterations is enough.

Finally, we use the inverse pseudopolar Radon transform to do computer tomography. We calculate the inverse transform of the *analytic* Radon transform of the Shepp–Logan phantom. The results are shown in Figure 4.3. We see that the reconstructed image (apart from some Gibb's related phenomena close to the skull) does a good job, and that the 256×256 example resolves all the important features of the image. In Figure 4.4 we plot the reconstructed image across the line $f(\cdot,0)$. compared with the exact function. We see that the different ellipses have been "found."

The method described here is closely connected to the so called *Linogram method* which was derived from a different point of view in [6, 7].

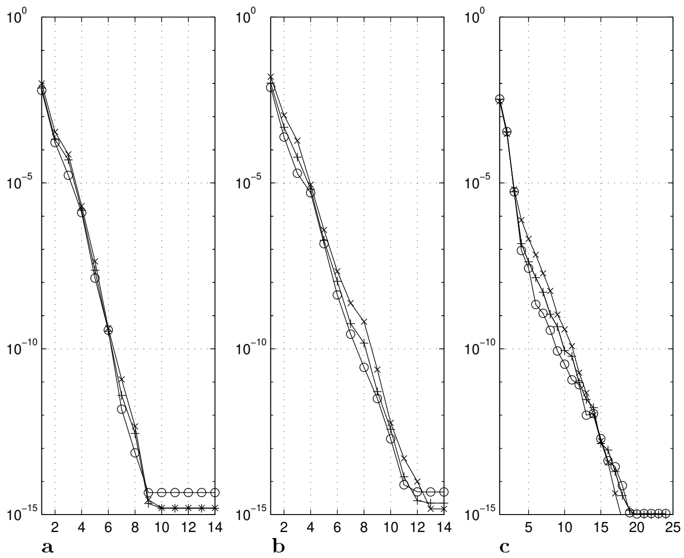


FIG. 4.2. *Error of inverse in ∞-norm as a function of iterations for: a) pseudopolar FFT ('X'=32×32, '+'=64×64, 'o'=128×128), b) pseudopolar Radon ('X'=32×32, '+'=64×64, 'o'=128×128), and c) pseudospherical FFT ('X'=$16 \times 16 \times 16$, '+'=$32 \times 32 \times 32$, 'o'=$64 \times 64 \times 64$).*

5. Conclusions. In many applications, computing a transform in pseudopolar coordinates is as good as doing it in polar coordinates, e.g. when doing computer tomography or analyzing wave front sets. The pseudopolar transforms presented in this paper offer fast and algebraically correct methods for computing these transforms. Other application areas will be investigated in forthcoming papers.

REFERENCES

[1] D.H. Bailey and P. Swarztrauber. The fractional Fourier transform and applications. *SIAM Review*, 33(3):389–404, 1991.

[2] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2(4):363–381, 1995.

[3] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier serier. *Math. Comput.*, 19:297–301, 1965.

[4] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, 1993.

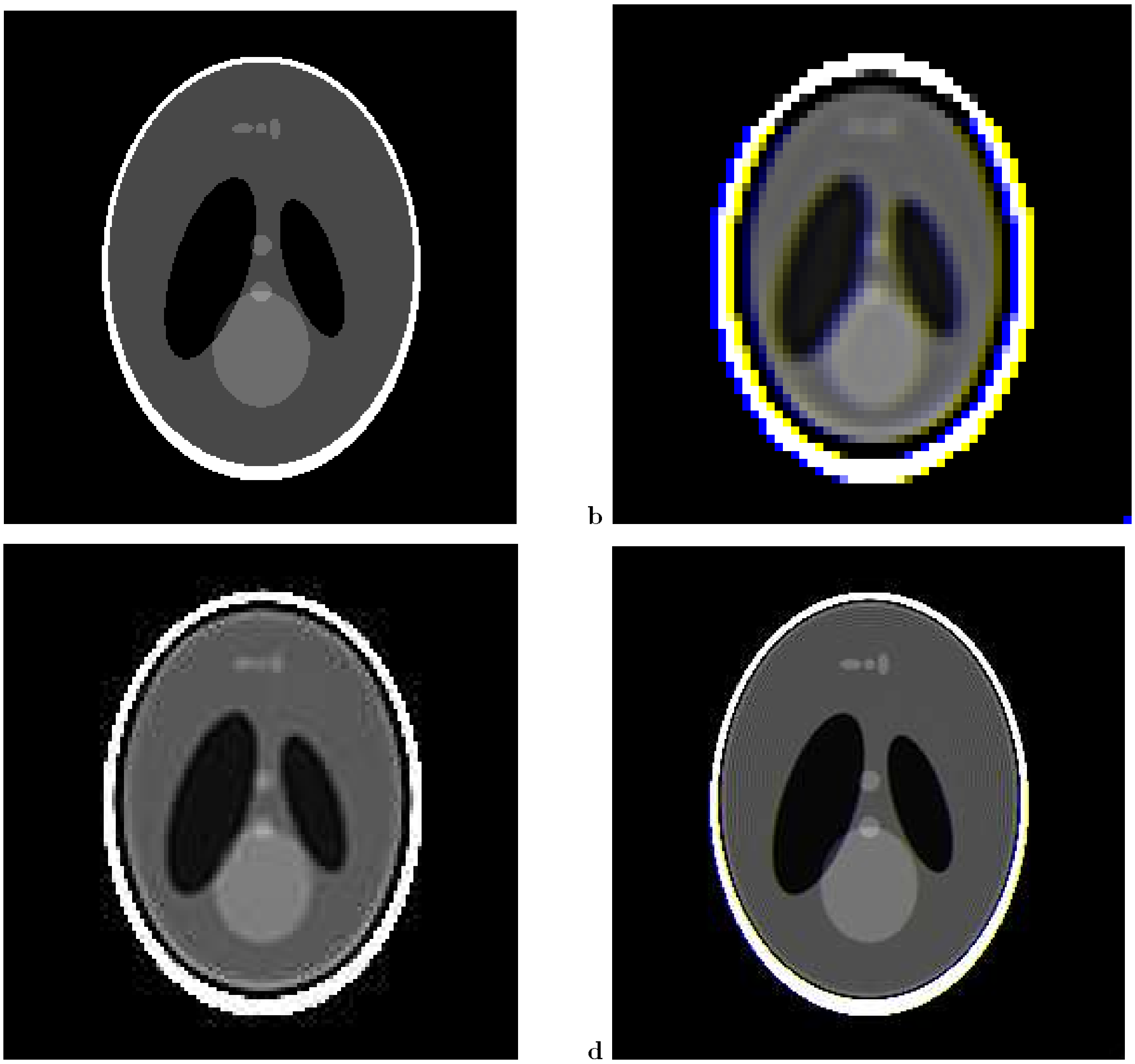
[5] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data, II. *Appl. Comp. Harmon. Anal.*, 2:85–100, 1995.

[6] P. Edholm and G. T. Herman. Linograms in image reconstruction from projections. *IEEE Trans. Med. Imag.*, MI-6(4):301–307, 1987.

[7] P. Edholm, G. T. Herman, and D. A. Roberts. Image reconstruction from linograms: Implementation and evaluation. *IEEE Trans. Med. Imag.*, 7(3):239–246, 1988.

[8] D. Gottlieb and B. Gustafsson. On the direct Fourier method for computer tomography. Technical Report 207, Dept. of Scientific Computing, Uppsala University, 1998.

[9] J. Waldén. Analysis of the direct Fourier method for computer tomography. Technical Report YALE/DCS/RR-1163, Yale University, Department of Computer Science, 1998.



a b c d

FIG. 4.3. *a) The Shepp–Logan phantom, b) Inverse with* 64×64 *points, c) Inverse with* 128×128 *points, d) Inverse with* 256×256 *points.*

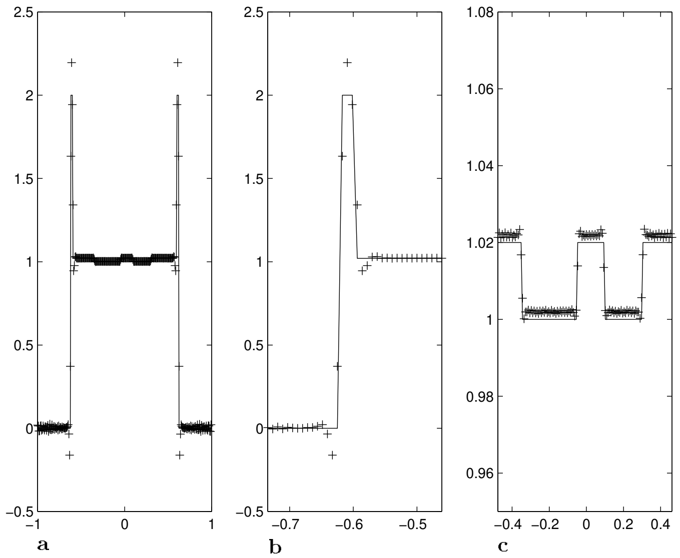


FIG. 4.4. *Reconstructed ('+') and exact ('-') Shepp–Logan phantom along the line* $f(\cdot, 0)$.