An Exact and Fast Computation of Discrete Fourier Transform for Polar and Spherical Grid

Syed Alam Abbas*, *Student Member, IEEE*, Qiyu Sun‡, *Member, IEEE*, and Hassan Foroosh†, *Senior Member, IEEE*

***Abstract*—Numerous applied problems of 2D and 3D imaging are formulated in continuous domain. They place great emphasis on obtaining and manipulating the Fourier transform in polar and spherical coordinates. However, the translation of continuum ideas with the discrete sampled data on a Cartesian grid is problematic. There exists no exact and fast solution to the problem of obtaining discrete Fourier transform for polar and spherical grids in the literature. In this paper, we develop exact algorithms to the above problem for 2D and 3D, which involve only 1D equispaced FFT with no interpolation or approximation at any stage. The result of the proposed approach leads to a fast solution with very high accuracy. We describe the computational procedure to obtain the solution in both 2D and 3D, which includes fast forward and inverse transforms. We find the nested multi-level matrix structure of the inverse process, and we propose a hybrid grid and use a preconditioned conjugate gradient method that exhibits a drastic improvement in the condition number.**

***Index Terms*—Nonuniform Fourier Transform, Fractional Fourier Transform, Discrete Fourier Transform, 2D imaging, 3D imaging, Polar Coordinates, Spherical Coordinates, Hybrid Grid, Conjugate Gradient Method, Block Toeplitz Matrix.**

I. Introduction

Given a function $f(x,y)$ of a continuous argument $(x,y) \in \mathbb{R}^2$, its two-dimensional (2D) Fourier transform (FT) [1]-[3] is given by

$$\hat{F}(u,v) = \int_{\mathbb{R}^2} f(x,y) e^{-2\pi i(xu+yv)} dxdy, \quad u,v \in \mathbb{R}. \quad \text{(I.1)}$$

Transforming the frequency coordinates from Cartesian to polar coordinates, i.e. $(u,v) = (\rho\cos\theta, \rho\sin\theta)$, we get the continuous polar Fourier transform

$$\hat{F}(\rho,\theta) = \hat{F}(\rho\cos\theta, \rho\sin\theta) \quad \text{(I.2)}$$

with a slightly abused notation.

Numerous applications of polar FT can be found especially in imaging. The significance of this trivial change of variables is that several fundamental procedures for manipulation of such continuous functions f (for instance rotation) can be described most elegantly in terms of polar variables. The polar FT is a powerful tool for organizing the understanding of operators and functions on the 2D continuous domain [4], [5]. Much the same can be said of higher dimensions.

The authors are with the Department of Electrical Engineering and Computer Science *† and the Department of Mathematics ‡, University of Central Florida, Orlando, FL, 32816, USA. e-mails: (*syedalamabbas@knights.ucf.edu, *syedalamabbas@gmail.com, †foroosh@eecs.ucf.edu, ‡qiyu.sun@ucf.edu).

Manuscript received November 1, 2016.

The continuous domain concepts are simple and obvious, but most data in practice exists only in sampled digital form [6]. This is the case in estimating image rotation, image registration, tomography or Radon transform, synthetic aperture radar imaging and more. The prevailing belief is that there exists no such algorithm for fast and exact computation of discrete polar FT ([7], [4], [5], [15] and [16]). Several difficulties as discussed in [4] exists for constructing fast polar FT for digital data: (1) appropriate polar grid should be defined in the frequency domain; (2) a fast way is needed to evaluate or approximate the FT on these polar grid points; and (3) a fast and stable transform inversion design is required.

For modern applications in scientific computing, two existing bodies of literature contain the solution to the above formulation of polar FT using discrete data. (1) One corresponds to methods that compute the inverse Radon transform with applications in wide range of disciplines such as radar imaging, geophysical imaging, nondestructive testing, and medical imaging [8]–[19]. This approach is called the direct Fourier method. It exploits the projection-slice theorem to convert projection data to 2D Fourier data and then reconstruct by Fourier inversion. (2) The second approach treats the general problem of evaluating the FT on non-equally spaced frequency points. This problem is known as USFFT (Unequally Spaced FFT) or the NUFFT/NFFT (non-Uniform FFT) [20]–[28]. Since the frequencies are by definition unequally spaced, the problem of evaluating the polar FT can be considered as a special case of the USFFT problem in this approach.

A detailed discussion of different approaches to polar Fourier transform adopted in the literature is presented in [13], [4], [15], [16] and references therein. The method in those papers is based on a special grid, called the pseudo polar (PP) grid, where equi-spaced angular lines are replaced by equi-sloped ones and the computation is fast. This grid has been explored extensively since the 1970s [4]. For long, it has been argued that this is the best way to obtain polar-like sampling specially adapted for the digital data, and hence it has proliferated to numerous applications as a substitute for the true polar form (i.e. the true discrete analog of its continuous form), see [15] and references therein. In comparison with true or ideal polar discrete grid, in the PP grid neither the angles are equi-spaced nor the points on the radial lines are equi-spaced with respect to one another. However, due to its data friendly nature it is considered as a well established alternative. The PP FFT is also used for compressed sensing (CS) based accelerated reconstruction [18] using a modified type of tomography called equisloped tomography [19] which was specifically designed for the PP grid as an alternative

to the true polar grid. The existing state-of-the-art solution in [4] adapts the ideas from the USFFT literature, uses an oversampled PP grid, and gives an approximate solution of the true Polar FT depending on the oversampling factor.

An alternative approach to the PP grid is the well-known USFFT or NUFFT based solution [5], which requires a cut-off parameter (for window functions) and an oversampling factor. However for applications like image registration which requires an accurate computation of the true polar FT from the given digital image ([29], [30]), the method ([4]) and the related USFFT interpolation methods ([24], [5]) are slow for high accuracy, and the choice of the parameter of the oversampling rate could be difficult for an arbitrary image [31]. For the computation of polar grid [4], PP grid acts as a stepping stone. Therefore, in order to obtain high accuracy, the PP grid must be oversampled, both radially and angularly, the minimum oversampling required is 2 ([4]), see [32] for more recent development. The result is still interpolated and inaccurate. In order to reduce the interpolation errors a multi-layered FFT (MLFFT) was introduced in [31]. With the use of the MLFFT method, the registration results are improved compared to the PP grid. However, again it requires the selection of cuts (of layers), which must be chosen carefully for it to work, but the inaccuracies still exists since it also uses an interpolation scheme. Some other variants of the PP grid such as the *Generalized* PP grid has been introduced, which are supposed to provide closer approximations to the true polar grid, but fast and accurate estimation has long been a major challenge for many image processing applications including the phase-correlation based motion estimation [30], [33]. One of the advantages of using a true polar grid instead of a PP grid, Generalized PP, MLFFT grid or log-polar grid is that we can do a straightforward computation of the phase-correlation operation in the polar Fourier domain for the registration problem, see [35] for a direct application in 3D.

To our knowledge, there exists no direct and exact computational procedure for obtaining the true *Polar* FT (2D) or *Spherical Polar* FT (3D), that is fast, devoid of any oversampling or interpolations ([4], [5]), and no design on parameters ([5], [31]). Our approach in this paper is uniquely in contrast to all the approaches discussed above, i.e. it does not require any accuracy parameter (like oversampling for PP FFT) and a cutoff parameter (like USFFT) or a design parameter (like cuts for MLFFT). Oversampling in our case for 2D and 3D refers to computing the Fourier coefficients at grid points outside the true desired gird points of polar and spherical grids, respectively. In [4], [5], [31], [33] and [34], an oversampled grid is first computed in the Fourier domain and the desired grid points (the actual points making up the polar and spherical grids) are then interpolated in it using some sort of high accuracy approximation scheme.

The remainder of this paper is organized as follows. Section II discusses the proposed method in 2D setting. The corresponding mathematical solution is described and the implementation details are given. Section III contains the digital inverse problematics and it provides a solution based on exact forward and adjoint formulation. We show the result of inversion using a simple diagonal preconditioner and a solution to improve conditioning of the inversion matrix, and exploit the underlying special matrix structure which is a nested 1-level block Toeplitz for 2D obtained from the transform operators (forward and adjoint) of polar FT. Section IV creates the formulation and its solution in 3D, which is far more complex and has a richer geometric structure to exploit. The special underlying matrix stucture is a nested 2-level deep block Toeplitz matrix and hence a very fast inversion solution can be designed. Due to the size and nature of complexity of the algorithm in 3D, a good software implementation which exploits the intricate 3D solution becomes very important. We show that the execution speed of the algorithm is greatly improved compared to the brute force (from hours to minutes) by the parallel implementation using GPU programming (CUDA) on a commodity hardware. Finally we follow by the contributions and conclusions in Section V.

In the spirit of reproducible research, we will make our code available to the public for academic purposes: https://github.com/syedalamabbas

II. The Proposed Method- 2D

In the first part of this section, we recall the definition of 1D fractional Fourier transform (FrFT) and discuss the three-step procedure to evaluate it. In the following core subsection, we use 1D FrFT to compute frequency domain information of 2D images at an arbitrary point, and then we show how it can be adapted to the true discrete polar Fourier domain solution (II.11) based on the polar grid definition (II.4) that is fast and algebraically exact. As it only involves the use of 1D FFT, it is fully vectorizable, memory efficient, and fast. It does not involve any parameter design for the user, preprocessing, or padding of any kind, but it has several levels (II.11). The computations of different levels are completely independent, and hence it is easy to parallelize the process on a GPU, multi-CPU and grid computing (CPU/GPU) systems.

A. Preliminaries: 1D Fractional Fourier Transform

Given an arbitrary scalar value α, the definition of the 1D fractional Fourier Transform (FrFT), also known as the Chirp-Z transform [36], is given by

$$F^{\alpha}(k) = \sum_{n=-N/2}^{N/2} f(n) e^{-i\frac{2\pi k \alpha n}{N+1}}, -N/2 \leq k \leq N/2, \quad \text{(II.1)}$$

where $f(n)$ is a discrete 1D signal supported on $-N/2 \leq n \leq N/2$ and N is an even integer. There is a fast algorithm to compute the above 1D FrFT in the order of $(N+1)\log_2(N+1)$, see [36] and [4] for instance. Substituting $2kn = k^2 + n^2 - (k-n)^2$ in the exponent of (II.1), we have

$$\begin{aligned} F^{\alpha}(k) &= \sum_{n=-N/2}^{N/2} f(n) e^{-\frac{i\pi\alpha}{N+1}(k^2+n^2-(k-n)^2)} \\ &= e^{-\frac{i\pi\alpha}{N+1}k^2} \sum_{n=-N/2}^{N/2} f(n) e^{-\frac{i\pi\alpha}{N+1}n^2} e^{\frac{i\pi\alpha}{N+1}(k-n)^2} \\ &= E(k) \sum_{n=-N/2}^{N/2} f(n) E(n) E(k-n) \quad \text{(II.2)} \end{aligned}$$

for all $-N/2 \le k \le N/2$, where

$$E(n) = e^{-\frac{i\pi\alpha}{N+1}n^2}, \ -N/2 \le n \le N/2. \tag{II.3}$$

To evaluate the FrFT (II.1), we use the following three-step procedure based on FFT and convolution in (II.2):

(i) We first multiply $f(n)$ by $E(n)$ and compute its FFT.
(ii) We then take the FFT of $E(n)$ and multiply it to compute the convolution.
(iii) We finally apply the IFFT to the product of the two FFTs and multiply it with $E(k)$, which gives us the desired FrFT.

As noted in [36] and [4], the FFT should be computed with zero padding to length $2(N+1)$ in order to avoid cyclic effects. A 1D FFT with $(N+1)$ points can be evaluated with operational cost $5(N+1)\log_2(N+1)$. In our procedure, since three 1D-FFT's are needed and the zero padded sequence length is $2(N+1)$, the total operation cost of evaluating (II.1) is $30(N+1)\log_2(N+1)$ floating point operations (flops).

B. The Algorithm

We begin with the idea of using 1D FrFT to compute frequency domain information in 2D images for an arbitrary point. Consider a 2D discrete data such as image $f(r,c)$ of size $(N+1)\times(N+1)$, where N is even, and r, c are the row and column indices, respectively. To compute the Fourier coefficient at any arbitrary location falling inside or outside the grid, we make use of separately scaled 1D FrFT in X and Y directions, see Figure 1. The total computational complexity is

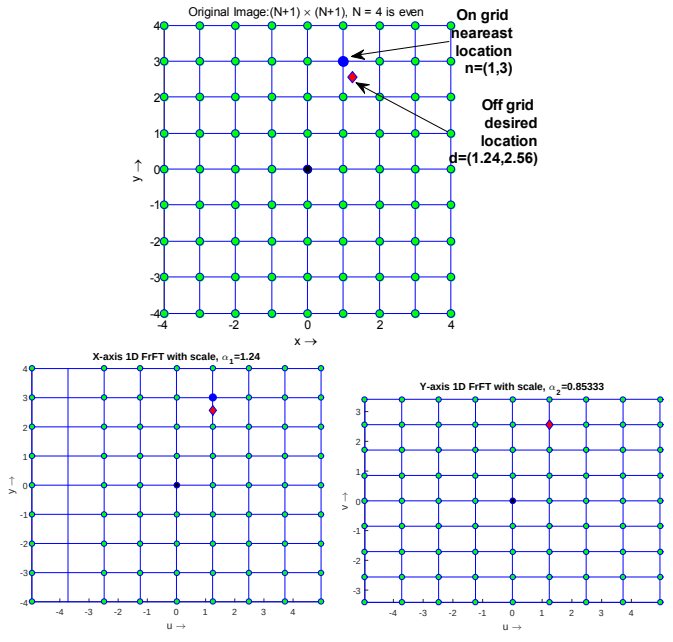

Fig. 1. Listed on the top, bottom left and bottom right are a uniform grid, an X axis scaled grid and an XY axis scaled grid, respectively. In the simulation, the desired point $(1.24, 2.56)$ for a 9×9 grid determines the exact scale factors to be used for the 1D FrFT along row or column. First is X axis scaling $\alpha_1 = 1.24$ (expansion) and then along Y axis, $\alpha_2 = .85333$ (shrinkage). These sequential scalings result in the desired point falling on coordinates $(1,3)$ of the transformed frequency grid.

$30(N+1)^2\log_2(N+1)+(N+1)$, where the first term in the sum represents 1D FrFT in X or Y direction (order does not matter) for all rows or columns, respectively, and the second term represents a direct computation of 1D FrFT for a single point. Inspired by this observation we design the solution for computation of polar grid points using 1D FrFT.

As shown in Figure 2, we define the polar grid of frequencies $\{u_x(m,n), v_y(m,n)\}$ in the circle inscribed in the fundamental region, $\{u,v\} \in [N+1, N+1)^2$. Given a discrete

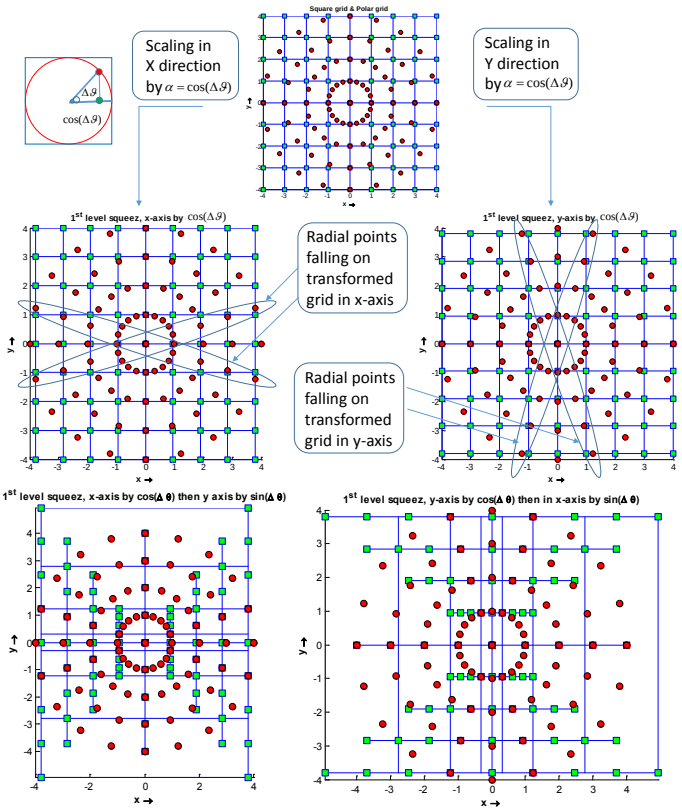

Fig. 2. Overlapping points of the polar grid with the rectangular scaled grid, each grid has exactly 4 radial lines that fall on it. X axis scaling has horizontally oriented and Y axis scaling has vertically oriented radial lines (both shown with ellipses). The four radial lines of level 1 solution are at angles $\Delta\theta$, $180^o - \Delta\theta$, $90^o - \Delta\theta$, and $90^o + \Delta\theta$, where $\Delta\theta = 18^o$.

Cartesian data $f(r,c)$, we define the polar Fourier transform (PFT) by

$$F(m,n) = \sum_{c=-N/2}^{N/2}\sum_{r=-N/2}^{N/2} f(r,c)e^{-i\frac{2\pi n}{N+1}(r\cos\theta_m + c\sin\theta_m)}, \tag{II.4}$$

where $-N/2 \le n \le -N/2$ represent the points on a radial line, $\theta_m = m\Delta\theta, 0 \le m \le M-1$, are the angles, $\Delta\theta = 180^o/M$ is the angular spacing, and M is the total number of radial lines.

The polar grid is of size $M\times(N+1)$, which is the total number of nodes with the origin included multiple times, where we choose for simplicity M as an even integer. Each radial line oriented at a specific angle θ_m is composed of $N+1$ points. The polar grid can be thought as a structure $P(m, R_m)$, where R_m is the radial line data composed of

uniformly spaced $N+1$ points oriented at an angle θ_m. Such a definition is useful in the context of Radon transform [13].

To use 1D FrFT for the computation of Fourier coefficients on radial lines of the polar grid, we require few definitions. The 1D frequency domain scaling of a 2D image along the X axis can be computed using a 1D FrFT along each row using

$$F_x^{\alpha}(r,c) = \sum_{n=-N/2}^{N/2} f(r,n) e^{-i\frac{2\pi c\alpha n}{N+1}}, \tag{II.5}$$

where $-N/2 \le r, c \le N/2$. Similarly, the frequency domain scaling of a 2D image along the Y axis can be computed by using a 1D FrFT along each column using

$$F_y^{\alpha}(r,c) = \sum_{n=-N/2}^{N/2} f(n,c) e^{-i\frac{2\pi r\alpha n}{N+1}}, \tag{II.6}$$

where $-N/2 \le r, c \le N/2$. Figure 2 shows the two scaled images, that are uniformly scaled in one axis but variably scaled in its complementary axis using 1D FrFT, which we call the *Butterfly grid*. For this special grid, the different scale factors are determined entirely by the geometry of the problem. The angular spacing $\Delta\theta$ determines the exact scale factors that are needed.

Consider the first scaling with $\alpha = \cos(\Delta\theta)$. We call this the level one scaling, which is composed of two solutions obtained using (II.5) and (II.6). To compute the 2D Fourier domain radial data on the X axis scaled grid, we take another 1D FrFT but with a variable scale for each column. For the column c, we scale it by a factor $|c|\beta$ where $\beta = \sin(\Delta\theta)$,

$$F_x^{\alpha,\beta}(r,c) = \sum_{n=-N/2}^{N/2} F_x^{\alpha}(n,c) e^{-i\frac{2\pi r|c|\beta n}{N+1}}, \tag{II.7}$$

where $-N/2 \le r, c \le N/2$. Similarly, to compute the 2D FT data of the radial lines on the Y axis scaled grid, for each row r, we use the same scale factor $|r|\beta$,

$$F_y^{\alpha,\beta}(r,c) = \sum_{n=-N/2}^{N/2} F_y^{\alpha}(r,n) e^{-i\frac{2\pi c|r|\beta n}{N+1}}, \tag{II.8}$$

where $-N/2 \le r, c \le N/2$.

Observe that we are interested in evaluating only 1 point for each column using (II.7), and 1 point in each row using (II.8). Thus on the 2D butterfly grid, not all points are computed, only the desired points which exactly overlap with the radial points on the polar grid lines need to be evaluated. There are 4 symmetrical radial lines, 2 at an angle $\Delta\theta$ and $180^o - \Delta\theta$ from the X axis scaled grid or basically the horizontal lines [13], and the other 2 at angles $90^o \pm \Delta\theta$ from the Y axis scaled grid or the basically vertical lines, not including the orthogonal lines at 0^o and 90^o. These four lines in pairs of 2 overlap exactly with the scaled butterfly grids in (II.7) and (II.8), respectively. The overlaps are obtained using the 1D FrFT operations with properly selected scales α and β, see Figure 2 with overlaps marked explicitly. This gives us the complete level one solution of the Fourier data on 4 radial lines: $F(1,n)$, $F(M-1,n)$, $F(M/2-1,n)$ and $F(M/2+1,n)$, which is denoted by F_1. The radial lines at the angles 0^o and 90^o can be computed using data from any levels.

For a perfect polar grid (i.e. when M is even), the different radial lines oriented in X-axis and Y-axis are given by angles $\theta_l = l \times \Delta\theta, l \ge 0$, and we have the corresponding scale factors $\alpha_l = \cos\theta_l$ and $\beta_l = \sin\theta_l$. Consider for level one solution using (II.7), we pick points that radially cross the dc value by 1 point, as shown in Figure 2, to get the desired radial line. Similarly, for level two solution, we select points that radially cross the dc value by 2 points to obtain the level 2 radial lines, as displayed in Figure 3. Inductively, the solutions of all levels can be written as

$$F_x^{\alpha_l,\beta_l}(r,c) = \sum_{n=-N/2}^{N/2} F_x^{\alpha_l}(n,c) e^{-i\frac{2\pi r|c|\beta_l n}{l(N+1)}} \tag{II.9}$$

and

$$F_y^{\alpha_l,\beta_l}(r,c) = \sum_{n=-N/2}^{N/2} F_y^{\alpha_l}(r,n) e^{-i\frac{2\pi c|r|\beta_l n}{l(N+1)}}, \tag{II.10}$$

where $-N/2 \le r, c \le N/2$ and $1 \le l \le L$. The set of scales $S = [\alpha_l, \beta_l; 1 \le l \le L]$ has the cardinality of $2L$, and the complete DFT information on the polar grid is given by

$$F = \bigcup_{l=1}^{L} F_l, \tag{II.11}$$

where $F_l, 1 \le l \le L$, are the solutions for different levels l. Shown in Figures 2 and 3 are the complete solution for the polar grid on the top of Figure 2, where $M = 10$ and $L = 8/4 = 2$.

1) Exploiting Symmetry: For the case that the scales α_l and β_l are rational numbers, the computation of the FrFT could reduce to conventional FFT, and the computation speed could be significantly increased by exploiting the periodicity of the data sequence. However, it is meaningless to talk about periodicity when dealing with the remaining case that one of scales α_l and β_l is irrational. So the use of same scaling within each level gives us some added advantage for that remaining case. By applying uniform scaling along X or Y axes, we obtain two symmetrical radial lines due to the conjugate symmetry in uniformly scaled grid. Due to irrational scales, there is hardly any connection across scales. However, for the same scaling α_l along X and Y axes, the calculations in (II.5) and (II.6) yield

$$F_x^{\alpha_l}(r,c) = F_y^{\alpha_l}(r,c) \tag{II.12}$$

for diagonal elements with $r = c$. For elements satisfying $rn_1 = cn_2$ for some integers n_1 and n_2, it is not necessary to compute twice for two grids on the same level, as we can efficiently use caching,

$$f(r,c) e^{-i\frac{2\pi r\alpha n_1}{N+1}} = f(r,c) e^{-i\frac{2\pi c\alpha n_2}{N+1}}. \tag{II.13}$$

2) Computational Complexity: The number of levels required for the polar grid is approximately $L = M/4$. In particular, when M is singly-even (i.e. M is divisible by 2 but not by 4), the number of levels is exactly $(M-2)/4$. For the remaining case that M is doubly-even (i.e. M is

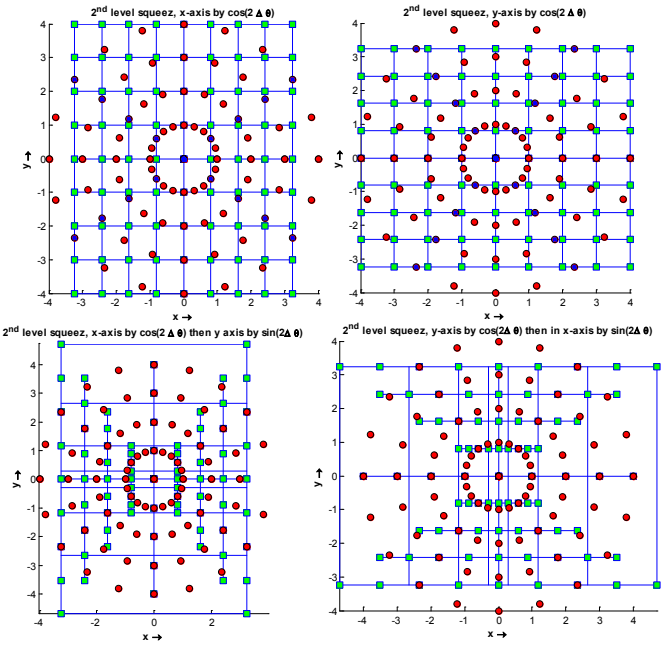

Fig. 3. Top row (left) (a) Image with X axis scaled grid, (right) (b) Image with Y axis scaled grid. Bottom row (left) (c) Image with XY axis scaled grid, and (right) (d) Image with YX axis scaled grid, with appropriate scales respectively. The four radial lines of level 2 solution are at angles $\Delta\theta$, $180^o - \Delta\theta$, $90^o - \Delta\theta$, and $90^o + \Delta\theta$, where $\Delta\theta = 2 \times 18^o$. This completes all the lines of the polar grid with $M = 10$ and $N = 8$ in (II.11).

divisible by 4), we have the special radial lines along the angles 45^o and 135^o, and hence we need to compute it only once using X or Y-axis scaled grid. Therefore the number of levels is $M/4$. At every level, the most expensive operation is the first uniform scaling by α_l along X or Y axis using (II.5) and (II.6), respectively. The second variable scaling to obtain the radial lines is simple since we are targeting far fewer points. Then, at each level, we have the complexity $(N+1)^2/2 \times \log_2(N+1)$ for X axis scaling using real data, and similarly $(N+1)^2/2 \times \log_2(N+1)$ for Y axis scaling. As we require about $M/4$ levels, the total computational complexity of our algorithm is

$$\frac{M}{4} \times (N+1)^2 \log_2(N+1). \tag{II.14}$$

The above estimate of polar grid holds due to the perfect symmetry of the equispaced points both angularly and radially. In the worst scenario when all the M radial lines are non-equispaced angularly, the computational complexity will increase to its maximum $M(N+1)^2 \log_2(N+1)$. As long as the radial lines have equispaced points (even when it is angularly non-equispaced), we can use the proposed approach to solve for this type of polar grid. When even the points on the radial lines are not equispaced, unfortunately we cannot use the proposed solution based on the 1D FFT, and we should resort to approximation techniques of nonuniform FFT ([5]).

C. Implementation Details

Considering (II.1), the index is in the unaliased form, but the FFT routines in most software systems are zero based or aliased form ([36]). We have to do a few modifications when we implement the computation of (II.1) using FFT for convolution. Let $n' = n + N/2$, then

$$\begin{aligned} F^\alpha(k) &= \sum_{n'=0}^{N} f\Big(n' - \frac{N}{2}\Big) e^{-i\frac{2\pi k\alpha(n'-\frac{N}{2})}{N+1}} \\ &= e^{i\frac{\pi\alpha N k}{N+1}} \sum_{n'=0}^{N} f\Big(n' - \frac{N}{2}\Big) e^{-i\frac{2\pi k\alpha n'}{N+1}}, \end{aligned}$$

where $-N/2 \le k \le N/2$. Similarly for $0 \le k' = k + N/2 \le N$,

$$\begin{aligned} F^\alpha\Big(k' - \frac{N}{2}\Big) &= e^{i\frac{\pi\alpha N(k'-\frac{N}{2})}{N+1}} \\ &\quad \times \sum_{n'=0}^{N} f\Big(n' - \frac{N}{2}\Big) e^{-i\frac{2\pi n'\alpha(k'-\frac{N}{2})}{N+1}}. \end{aligned}$$

Rearranging and simplifying the above expression gives

$$F^\alpha\Big(k' - \frac{N}{2}\Big) = C(k') \sum_{n'=0}^{N} \Big\{ f\Big(n' - \frac{N}{2}\Big) B(n') \Big\} e^{-i\frac{2\pi\alpha n' k'}{N+1}}, \tag{II.15}$$

where $C(k') = e^{i\frac{\pi\alpha N k'}{N+1}}$ and $B(k') = e^{i\frac{\pi\alpha N(k'-\frac{N}{2})}{N+1}}, 0 \le k' \le N$. Thus for the unalaised form, using the pre-multiplication factor B, and post-multiplication factor C, the expression in (II.15) reduces to zero based solution which can be computed via convolution using FFT routines as described in (II.2).

The following is the pseudo code for our proposed 2D fast PFT computation:

Input: `Image` f `of size` $(N+1)\times(N+1)$ `and the number` M `of angles, where` M, N `are even.`

Output: `Polar grid` F `of size` $M \times (N+1)$ `computed according to the definition` (II.4).

Step 1: `For the level` l`, where` $1 \le l \le L$ `and` $L \approxeq M/4$ `is the total number of levels, do the following.`

Step 2: `Apply the scaling` α_l `to evaluate 1D FrFT` (II.7) `for the image` f `row-wise.`

Step 3: `Apply the scaling` α_l `to evaluate 1D FrFT` (II.8) `for the image` f `column-wise.`

Step 4: `Compute two horizontal lines using the scaling` β_l `according to` (II.9).

Step 5: `Compute two vertical lines using the scaling` β_l `according to` (II.10).

Step 6: `Gather four radial lines for level` l `in the output array.`

Step 7: `If` $l ==$ `L, terminate. Else` $l = l + 1$, `go to` **`Step 2`**.

D. Accuracy of the Proposed Fast PFT

The use of FFT for numerical calculation of convolutions has a fundamental challenge of aliasing error [41]. This inherent error manifests itself for larger arrays, while for computations of smaller arrays the accuracy approaches the machine precision. Figure 4 shows the precision loss for the computation of 1D FrFT, for a randomly generated signal

$f(n), -N/2 \leq n \leq -N/2$ and scaling factor α, where the maximum absolute error is given by

$$E_{\max} = \max_n |F^{\alpha}_{\text{direct}}(n) - F^{\alpha}_{\text{fast}}(n)| \tag{II.16}$$

is plotted against the array size N. As indicated in the figure the error steadily increases for larger array size, but the loss of precision is still graceful even for a sufficiently large array size. The error is in the order of 10^{-9} for $N = 1400$.

The fast convolution step ([39], [40]) in the proposed fast PFT algorithm is the only source of error, all the other steps are exact up to machine precision, since there are no interpolations, approximations or any form of oversampling. This precision loss for larger size arrays is also present in the computation of PP grid (or just about any algorithm where fast convolution is used), since the basic building block, 1D FrFT based on fast convolution, is the same for the computation of both grids.

The zero padding used for computation of the convolution (II.2) is not necessary, as there are ways to compute this without padding with additional variables, see [43] for details. The reader may refer to [39]–[42] for various methods for improving precision.

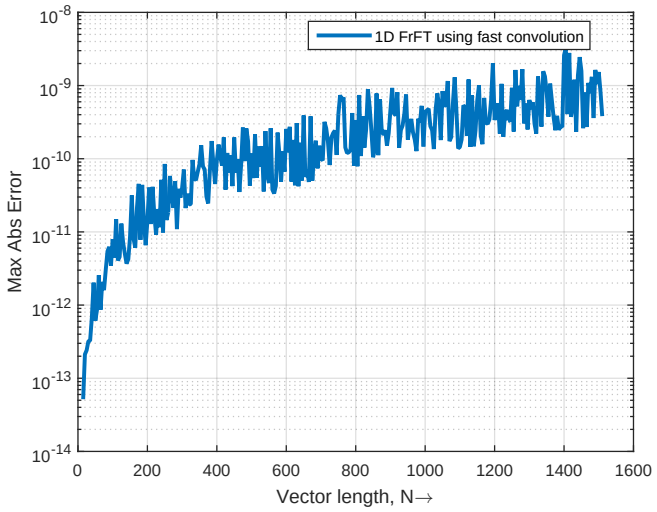

Fig. 4. Precision loss due to fast convolution, $E_{\max}$ vs. N. As can be seen from the figure that although there is a steady loss of precision for larger array sizes, for moderately sized array the error is still of order 10^{-10}. This is the same precision loss with the polar grid computations using (II.4) since the fast convolution used for 1D FrFT is the building block of it.

E. Comparison

The following numerical simulation is computed with the NFFT C-subroutine library [44]. The parameters to choose are the cutoff parameter ρ for Keiser-Bessel window functions, and the oversampling factor $\sigma = 2$ used internally. The NFFT evaluates (II.4) at $N \times M$ arbitrary nodes (in our case the polar grid) in

$$O((\sigma N)^2 \log(\sigma N)^2 + \rho^2 NM) \tag{II.17}$$

arithmetical operations.

Figure 5 shows the simulation results for an oversampled grid as well as a normally sampled grid. Here, we let $N = 128$, $M = 3N$ for an angularly oversampled grid and $M = N$ for normally sampled one. For the simulation, the computational

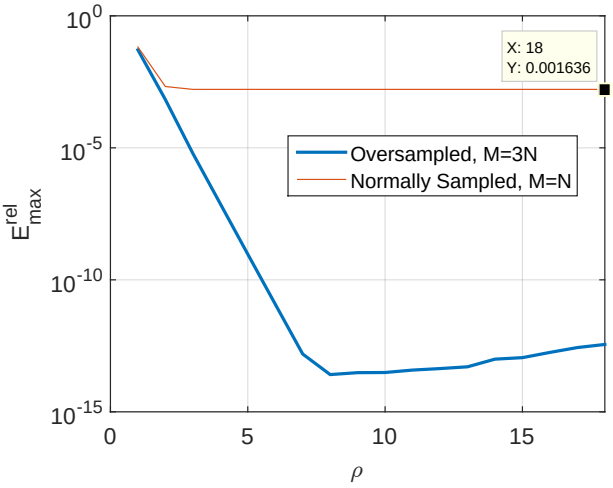

Fig. 5. In this simulation, the accuracy and computational complexity of NFFT are compared with the proposed algorithm for the polar grid.

complexity of our algorithm for $M = 128$ ($M/4 = 32$) is $O(32 \times 128^2 \log_2 128) = O(224 \times 128^2)$, and the accuracy is achieved is very high. On the other hand, the complexity of NFFT is $O(18^2 \times 128^2 + 4 \times 128^2 \log_2 256) = O(356 \times 128^2)$, since $\rho = 18$. The relative accuracy $E^{\text{rel}}_{\max}$ achieved is of the order 10^{-3}, where the relative accuracy of the NFFT is given by

$$E^{\text{rel}}_{\max} = \frac{\max_{m,n} |F_{m,n} - F^{\text{NFFT}}_{m,n}|}{\max_{m,n} |F_{m,n}|}. \tag{II.18}$$

The accuracy could improve in the oversampled case, as shown in the blue line in Figure 5 with $M = 3N$. The improvement is due to the fact that dense angular sampling reduces the interpolation errors for NFFT. But it is not advisable to arbitrarily increase the cutoff parameter ρ as the error increases after certain threshold, for instance $\rho > 8$ and $M = 3N$ in Figure 5.

The proposed method does not depend on the angular oversampling and still acheives very high-accuracy approximation i.e. the relative error $E^{\text{rel}}_{\max}$ is of order 10^{-14} and the absolute error $E_{\max}$ is of order 10^{-10}.

The execution time of the proposed method is shown in the Figure 6. The direct computation of Discrete PFT is of order $O(N^4)$, while the proposed fast technique with far reduced complexity can handle images of sufficiently large sizes. As the execution time of the proposed solution is orders of magnitude faster, while maintaining the similar level of accuracy as brute force computations. The solution for larger N can be achieved with parallelization as the proposed technique can be fully distributed and scales naturally with the computational resources.

The proposed solution can be adapted to approximate other kinds of grid used in the literature, such as spiral grid [45], log-polar grid [31] using oversampling and approximation.

It is easily extensible for limited angle tomography grid or even a completely arbitrary grid with all angles non-uniformly spaced.

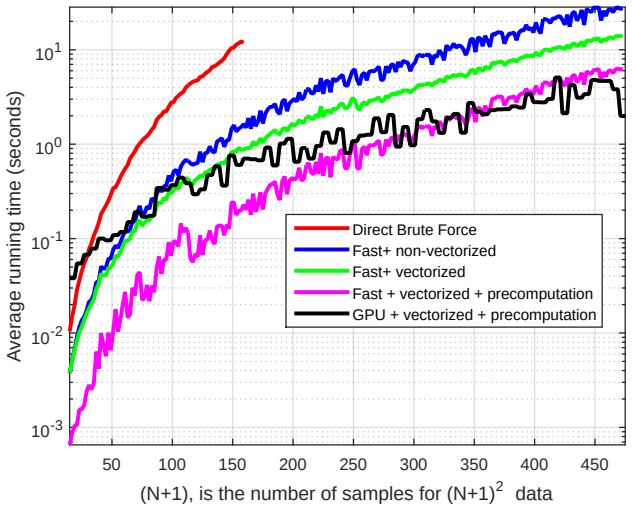

Fig. 6. Plotted are the execution times vs the input array sizes N where the images are of size $(N+1)^2$. The execution time using direct brute force computations is not plotted for $N > 150$ as it grows quickly and can take hours to complete.

III. Digital Inverse Problematics

The accuracy of the exact forward transform computations is very high, as discussed in Section II, but for inversion there is no direct, exact and fast algorithm. In practice a fast way is needed for the reconstruction given the polar Fourier data especially for large size images. In the following four subsections, we analyze and design an iterative inversion scheme using linear algebra techniques. We also discuss the ill-conditioning inherent in the polar sampling process, and give solutions that drastically improve the conditioning of the designed linear systems.

A. Analysis

The proposed method of discrete PFT can be written as a linear operator A_p, which we call the discrete PFT operator. We use this matrix representation only for analysis, not for implementation. Let x be a vector formed from stacking the image f in (II.4) column-wise, and y be the vector obtained from stacking the PFT values F in (II.4) row-wise, i.e. the radial lines stacked sequentially. Then the linear transformation can be represented as

$$A_p x = y, \tag{III.1}$$

where A_p is a complex valued matrix of size $M(N+1) \times (N+1)^2$, x is a $(N+1)^2 \times 1$ real vector representing the signal/image, and y is a $M(N+1) \times 1$ complex valued vector representing the transformed signal.

An $(N+1)$ point FrFT in (II.1) can be expressed as an $(N+1) \times (N+1)$ matrix-vector multiplication similar to an FFT matrix representation. The transformation matrix Ω_α with scale factor α can be defined as

$$\Omega_\alpha = \left(\omega^{(N/2+1-i)(N/2+1-j)}\right)_{1\le i,j\le N+1},$$

where $\omega = e^{-i\frac{2\pi\alpha}{N+1}}$. Let $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \cdots, \mathbf{r_{N+1}}$ be row vectors of the image f, then the equation (II.5) that describes the uniform scaling along X axis can be written in matrix-vector product form as

$$\mathbf{r}_\mathbf{n}^\alpha = \Omega_\alpha \mathbf{r_n}, \ 1 \le n \le N+1.$$

Stacking these new row vectors together, we get the $(N+1) \times (N+1)$ matrix F_x^α as shown in (II.5). The next scaling is variable scaling as discussed before along Y axis which requires a different transformation matrix S_β of the form,

$$S_\beta = \left(\xi^{(N/2+1-j)(N/2+1-i)}\right)_{1\le i,j\le N+1},$$

where $\xi = e^{-i\frac{2\pi\beta}{N+1}}$. Since we are only interested in just two lines which are also symmetrical in the equation (II.7), then the solution to a line (oriented at the first angle) in full matrix form becomes

$$S_\beta \circ F_x^\alpha$$

where $\circ$ is the Hadamard elementwise product. The complementary line solution can be obtained from the matrix $S_\beta^H \circ F_x^\alpha$ where S_β^H represents the Hermitian transpose of S_β.

Let $J = (1, 1, \cdots, 1)$ be the column vector of $1s$ of size $(N+1) \times 1$. Then for a square matrix A of size $(N+1) \times (N+1)$, $J^T A$ is the column-wise sums of A, and AJ is the row-wise sums of A, see [46]. Finally the line oriented at an angle $\Delta\theta$ and the complementary line oriented at an angle $180^o - \Delta\theta$ are given by

$$R_1 = J^T(S_\beta \circ F_x^\alpha) \quad \text{and} \quad R_{M-1} = J^T(S_\beta^H \circ F_x^\alpha).$$

Similarly, all the other radial lines can be computed in the above forms. Thus the matrix A_p has elements of forms, $S_{\beta_l} \circ W_{\alpha_l}$ and $S_{\beta_l}^H \circ W_{\alpha_l}$, stacked either row-wise or column-wise for all the scales $S = [\alpha_l, \beta_l]$ in (II.9) and (II.10). Only the values of the exponents will change depending on the scales but the structure of the elements obtained would be similar. We can express A_p as a product of two matrices

$$A_p = VU, \tag{III.2}$$

where V, U are large and sparse matrices composed of values obtained using terms of β_l and α_l respectively. The *adjoint* operator of A_p is given by

$$A_p^H = U^H V^H. \tag{III.3}$$

Consider the matrix-vector formulation of the inversion process from (III.1) (see [13], [24] for similar formulation), multiplying both sides by adjoint gives

$$A_p^H A_p x = A_p^H y.$$

Let $A_p^+ = (A_p^H A_p)^{-1} A_p^H$ be the generalized inverse or the well known Moore Penrose pseudo inverse, a solution to (III.1) is given by

$$x = A_p^+ y. \tag{III.4}$$

B. Ill-Conditioning

In contrast to the forward transform and its adjoint operation which have an exact formulation and where the considerations of the density of sampling (distribution of nodes) was of least concern, the inverse PFT and the associated reconstruction error heavily rely on the distribution of the polar sampled nodes y. The problem inherent is due to the geometry of the polar grid. There is some loss of information since the corners are not represented. From the algebraic point of view, the matrix A_p in (III.1) is ill-conditioned and the condition number grows quickly as the number N of samples in the spatial grid increases. From the reconstruction perspective, visible artifacts are present, the solution obtained is sensitive to numerical errors in the measurement process, and numerous iterations are required to achieve the desired low residual error.

One way to combat this problem of missing corners is simply to extend the sampling set for corner compensation, as suggested in [5], where a modified polar grid is proposed, the polar sampling set with more concentric circles is added and nodes not located in the square grid are excluded. If the number of polar grid nodes is $M \times (N+1)$, with the origin included multiple times, the modified polar grid has slightly increased total number of nodes $\approx \frac{4}{\pi}\log(1+\sqrt{2})M(N+1)$, see Figure 7 (ii) and (iv). The effect of the inclusion of corners on the conditioning of the linear PFT operator A_p can be seen in Figure 8. The extra measurements improve the conditioning of the system, but we can further stabilize the inverse transform through a different arrangement. We propose a hybrid grid, where with the pure polar grid we include the corners from the Cartesian grid as shown in Figure 7 (v) and (vi).

Figure 7 presents several sampling grids and Figure 8 shows the effect on the condition number of the operator A_p with different arrangements. The spatial grid is of size $(N+1)\times(N+1)$, the number of angles for all grids remains constant (i.e. $M = 2(N+1)$) only the sampling radial density and the set geometry varies. As we can see in Figure 8, the best conditioning is achieved when we use the proposed hybrid grid combined with oversampling radially when the number of angles is fixed. Thus using the extra measurements in the corners the condition number will be well controlled. References [47] and [5] discuss the density of the sampling set and its influence on the condition number by using a more precise mesh-norm, which gives the maximum distance allowed between neighboring nodes. For sufficiently dense sampling sets, the reconstruction problem is well conditioned using the proposed hybrid grid, c.f. [5].

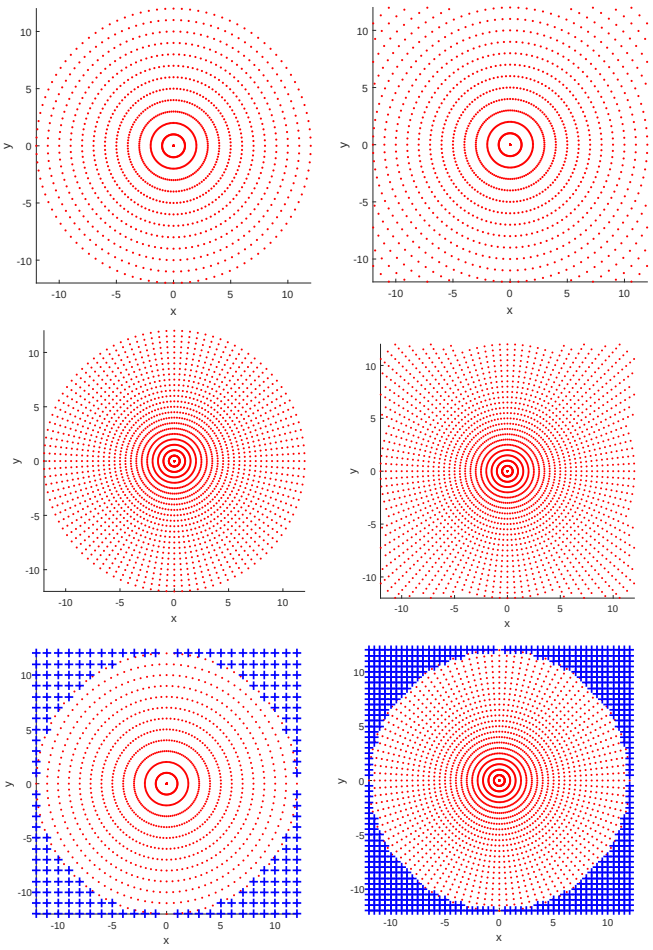

Fig. 7. Top row, left (i) Polar grid (pure), and right (ii) Polar grid with radial corners included. Middle row, left (iii) Polar grid 2x radial sampling, and right (iv) Polar grid 2x radial sampling with radial corners included. Bottom row, left (v) Polar grid with Cartesian corners (hybrid), and right (vi) Polar grid 2x radial sampling with Cartesian corners. Here, $N = 24$, i.e. the spatial domain Cartesian grid is 25×25 and only the total number of angles $M = 52$ remains constant for all the grids. See Figure 8 for the condition number comparison of these grids.

C. Fast Iterative Inversion

The matrix based solution (III.4) is for analysis only. For reasonable sizes of input arrays, a pseudo direct inversion through the matrix A_p^+ would be computationally prohibitive. It is beyond the desired low complexity and high speed of the algorithm. In this subsection we develop a fast iterative reconstruction based on linear algebra analysis and the use of conjugate gradient technique.

For iterative image reconstruction algorithms, one needs the *adjoint* A_p^H of the discrete PFT operator A_p in (III.2) to perform matrix-vector multiplications of the form $A_p^H y = \widetilde{x}$. Since the matrix A_p is too large to store in the imaging problems of interest, and since the direct matrix-vector multiplication would be computationally inefficient, we evaluate

$$A_p^H y = U^H V^H y$$

by reversing (and not inverting) the operations performed for the forward transform in (III.2) as shown in (III.3). Similar to the above discussion the reversing of the equations (II.7), (II.8), (II.9) and (II.10) can be done with the same complexity as the forward transform.

Instead of using (III.1) and solving it directly via (III.4), we approach it as an optimization problem

$$\min_x ||A_p x - y||_2^2.$$

Due to variations in the local sampling density of points in the polar grid and its modification, weights are introduced to compensate for its effects. We can use the formula for weights

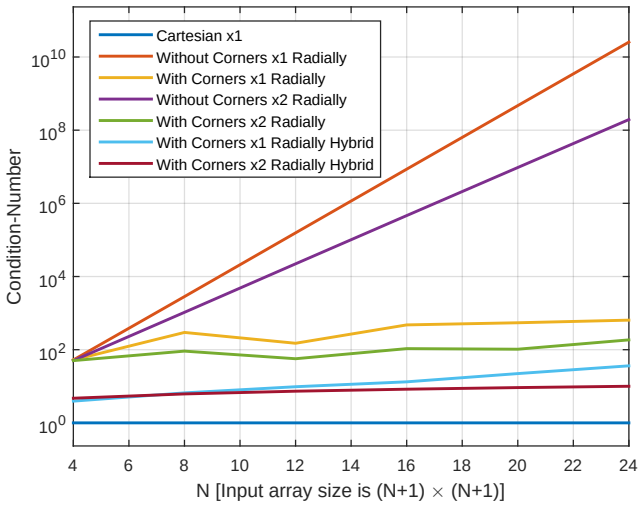

Fig. 8. Improving the ill-conditioning of the linear system in (III.1) by including the additional measurements. Without the inclusion of corners the condition number of A_p even for really small sizes of grids (25×25) is very high (order of 10^{10}). Inclusion of corners (see [5]) along radial direction improves the conditioning and so does the oversampling by 2x along radial lines. But the proposed hybrid solution (i.e. polar grid with Cartesian corners) greatly reduces the condition number of the system and combined with oversampling we get the best result which is closest to the ideal condition number of 1 for a perfect Cartesian grid.

in [4], [5]. The preconditioner or weights W is used in the following way, rather than minimizing $||A_p x - y||_2^2$, we use

$$\min_x ||W \circ A_p x - W \circ y||_2^2$$

instead and we solve it iteratively using the relation of least squares (LS) solution

$$x_{k+1} = x_k - \mu A_p^H W^2 \circ (A_p x - y) \tag{III.5}$$

where μ is the step size and k is the iteration number. The expression $A_p^H(A_p x - y)$ is the function's gradient and the multiplication by a positive definite matrix W guarantees descent in the LS error. The specific choice of W as discussed in [4] is such that it down-weights the points near the origin in order to equalize the condition number.

For corner compensation as discussed in the previous section, we consider two systems. Let A_{cc} be the operator that takes into account only the Cartesian corner points, i.e. the Fourier domain coefficients falling outside the range of the polar grid. Then, the two systems with operators A_p (representing the discrete polar form) and A_{cc} (representing the Cartesian corners) can be written as

$$A_p x = y \tag{III.6}$$
$$A_{cc} x = y_c, \tag{III.7}$$

where y_c is the set of Fourier domain information at the corners. Then using the augmented system $[A_p : A_{cc}]$ with the augmented output $[y : y_c]$, the new symmetric positive definite linear system becomes

$$Ax = b \tag{III.8}$$

where

$$A = A_p^H W^2 \circ A_p + A_{cc}^H A_{cc}$$
$$b = A_p^H W^2 \circ y + A_{cc}^H y_c.$$

Possible ways around the corner compensation problem is to suppress/remove the fixed pattern noise by a suitably designed filtering process (similar to the filtered back projection) or simply assume that only the center of the image matters. If the given image is padded by a factor of $\sqrt{2}$ along both axes, a 2D FFT is applied and 2D IFFT is followed, then the resultant image is twice as big in pixel count and its frequency domain support is concentrated in the center. These steps lead to additional $(N+1)^2 \log_2(N+1)$ operations to the process, see Figure 9 for the reconstruction with padding. From the reconstructed image using the corner compensation and extra padding by factor of $\sqrt{2}$, we get almost perfect reconstruction. Similar to the approach followed in [4], we assume in the simulation that there is no corner Fourier information present and the image is concentrated at its center, i.e. the values for vector y_c are all set to zero.

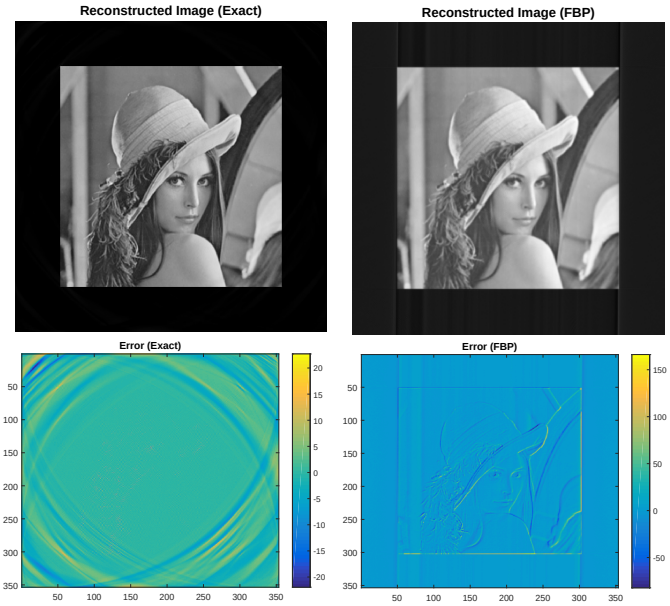

Fig. 9. (Top row) Corner compensated and padded reconstruction, where $M = 2(N+1)$, and residual error is $= 8.7 \times 10^{-5}$ after just 12 iterations. We initialize the image with zeros for the first iteration, and we get a near perfect reconstruction at the image center. Observe that the sharp features have been destroyed by the use of filtered back projection method as it shows up in the error comparison. However, using the proposed method of exact discrete PFT we are able to retrieve those sharp features with the same sampling data. (Bottom row) Error comparison using the exact solution and the FBP approach by MATLAB implementation using functions radon and iradon.

D. Faster Inversion by Structured Matrix Computations

In this subsection, we exploit the underlying special structure of the matrix A in (III.8) to get much faster solution than the one discussed in section III-C.

The matrix A is a real symmetric two-level Toeplitz matrix. Block-Toeplitz-Toeplitz-block (BTTB) matrices or two-level Toeplitz matrices are the 2D analogues of Toeplitz matrices.

A BTTB matrix is a block matrix with Toeplitz blocks, and it has Toeplitz structure on the block level. Block-circulant-circulant-block (BCCB) matrices are the 2D analogues of circulant matrices, which are diagonalized efficiently by the 2D FFT. With the concept of circulant embedding we can transform a real symmetric BTTB matrix into a larger real symmetric BCCB matrix. In higher dimensions such matrices have a nested multilevel block Toeplitz structure.

The BTTB system in (III.8) can be extended to a BCCB system. If the input image stacked vector x is of size $(N+1)^2 \times 1$, the BTTB matrix A is of size $(N+1)^2 \times (N+1)^2$. With the use of circulant embedding we have the extended BCCB matrix $\tilde{A}$ of size $(2N)^2 \times (2N)^2$,

$$\tilde{A}\tilde{x} = \tilde{b}.$$

Figure 10 shows the detailed structure of matrix A and its BCCB extension. A BCCB matrix is a normal matrix, thus a matrix $\tilde{A}$ has a unitary spectral decomposition. It is well known that any BCCB matrix has the particular spectral decomposition

$$\tilde{A} = F^H \Sigma F$$

where F is the two dimensional unitary DFT matrix and the diagonal matrix Σ represents the eigenvalues of $\tilde{A}$ [49]. The specialty of this computation lies in that neither F nor F^H need to be explicitly computed or stored. Once we know the spectral decomposition, we can efficiently perform matrix multiplication with a BCCB matrix as

$$\tilde{b} = \tilde{A}\tilde{x} = F^H \Sigma F \tilde{x},$$

i.e. the product can be computed by first taking 2D FFT of $\tilde{x}$, multiplying elementwise with the 2D FFT of the first column of $\tilde{A}$ arranged in a 2D array, and finally performing 2D IFFT of the product to get the multiplication $\tilde{A}\tilde{x}$.

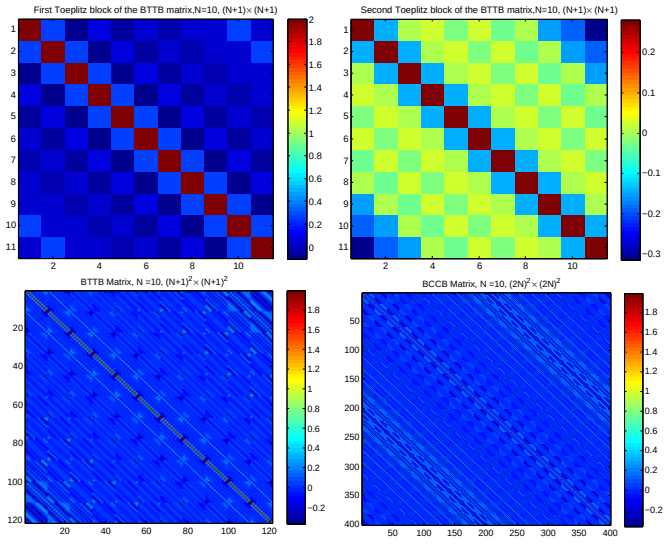

Fig. 10. BTTB to BCCB conversion, performing extension of real symmetric BTTB matrix using concept of embedding.

For the computation of the system in (III.8) using the conjugate gradient algorithm, in each iteration we require an operation with A_p^H and an application of operator A_p, both operations require complexity of order $O(M/4 \times (N+1)^2 \log(N+1))$. The other operators of Cartesian corners are simply centered 2D FFT and 2D IFFT of order $O((N+1)^2 \log(N+1))$. Now, exploiting the BTTB structure of matrix A we get faster matrix vector multiplication Ax, such that only once the first column of A needs to be computed. The impulse response of the system $A = A_p^H W^2 \circ A_p + A_{cc}^H A_{cc}$ gives the first column of the BTTB matrix. Thus for each subsequent iteration of the conjugate gradient algorithm, the operation Ax for any arbitrary vector x is of complexity $O((N+1)^2 \log(N+1))$. Moreover the complexity is independent of the number of angles M but it only depends on the image size $(N+1)$. Also, unlike the single level Toeplitz systems discussed in [15], [16] and [17] for the PP grid, which gives a direct inversion algorithm, higher level (≥ 2) Toeplitz systems do not have a direct inverse solution of closed form, hence only preconditioned iterative techniques can be used. We have discussed the inverse solution without considerations to the noise, however sophisticated regularization techniques such as the T-SVD, or the Tikhonov method ([49]) can be used in the presence of noise for stable inversion.

The fast forward and the inverse transforms discussed above are based on an algebraically exact formulation with full geometric fidelity. Compared to the NFFT and the pseudo-polar grid approach, our solution provides the sparsest representation devoid of any unnecessary oversampling. The NFFT and the pseudo-polar grid both require oversampling factors larger than or equal to 2 both angularly and radially, and the available implementations ([4], [5], [13] and [14]) for discrete data are only vaguely related and not precise analogs to the continuous transforms.

IV. The Proposed Method- 3D

In this section, a similar strategy as discussed in section II is adopted for 3D setting, which has significantly more complexity and structure. Extension to 3D will require the definition of polar-like coordinates, and no exact polar sampling method for 3D is well defined [4]. Thus we must settle for polar-like grids.

For DFT using spherical coordinates, we follow a simple definition of spherical transformation and show that it has a different structure compared to 2D, such that more radial lines can be computed with a single appropriately scaled grid in 3D.

Consider the transformation of 3D rectangular coordinates to 3D *spherical* or *spherical polar* coordinates, where $(x, y, z) \rightarrow (\rho, \theta, \phi)$, $x = \rho \cos\theta \cos\phi$, $y = \rho \cos\theta \sin\phi$ and $z = \rho \sin\theta$, the radius $\rho \in [0, \infty]$, polar angle or *zenith* or elevation $\theta \in [0, 180]^o$ and *azimuth* angle $\phi \in [0, 360]^o$. We follow the convention used in physics and also in the spherical harmonics notations (computer graphics) rather than the ones used in pure mathematics, where the roles of θ and ϕ are reversed [50]. We measure the azimuth angle with respect to the x-axis, also as it is standard in the 3D software systems (to keep the equations and the implementation consistent) we measure the zenith angles from the xy-plane rather than with respect to z axis, see Figure 11.

Given the Cartesian 3D matrix data $f(r,c,d)$, where r,c,d represent rows, columns and depths indices respectively, we define the *Spherical Polar Fourier Transform* (SPFT) to be the collection of samples $F(k,m,n)$, similar to 2D form in (II.4),

$$F(k,m,n) = \sum_{d=-N/2}^{N/2} \sum_{c=-N/2}^{N/2} \sum_{r=-N/2}^{N/2} f(r,c,d) \times e^{-i\frac{2\pi n}{N+1}(r\cos\theta_k\cos\phi_m + c\cos\theta_k\sin\phi_m + d\sin\theta_k)}, \tag{IV.1}$$

where $-N/2 \le n \le -N/2$ represents the points on a radial line, $\theta_k = k\Delta\theta, 0 \le k \le K-1$ is the zenith angles, $\Delta\theta = 180^o/K$ is the angular spacing in zenith angles, $\phi_m = m\Delta\phi, 0 \le m \le M-1$ means the azimuth angles, $\Delta\phi = 180^o/M$ is the angular spacing in azimuth angles, and $K \times M$ is the total number of radial lines. Thus the total number of DFT points is $K \times M \times (N+1)$, where the origin is included multiple times. Similarly, we can define the adjoint operator of the SPFT as follows

$$\hat{f}(r,c,d) = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} \sum_{n=-N/2}^{N/2} F(k,m,n) \times e^{+i\frac{2\pi n}{N+1}(r\cos\theta_k\cos\phi_m + c\cos\theta_k\sin\phi_m + d\sin\theta_k)}. \tag{IV.2}$$

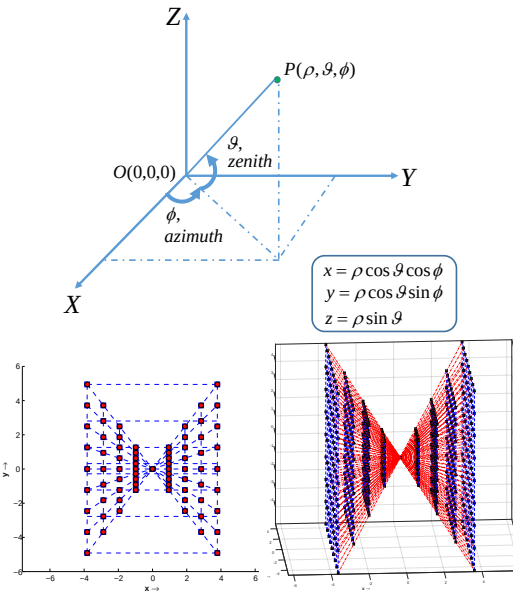

Fig. 11. Spherical coordinate conventions (top), butterfly grid 2D (bottom left) and 3D (bottom right). Conventions followed here are shown. We measure the *azimuth* angle with respect to the x-axis and *zenith* angle with respect to the xy-plane. In 2D, the butterfly grid is a set of concentric rectangles and in 3D it is a set of concentric cuboids, both obtained by applying 1D FrFT on select axes. Only one axis is uniformly scaled with 1D FrFT, the other two axes are non-uniformly scaled.

The butterfly grid was crucial for the computation of radial lines in the 2D case as shown in Figure 2, because this grid has special selection of points that exactly match the positions of the desired radial points of the polar grid. We propose a 3D butterfly grid, a set of concentric points arranged in the 3D square frustum, see Figure 11. For the 3D butterfly grid only one axis is uniformly scaled in the frequency domain using 1D FrFT, the other two axes are non-uniformly scaled, where the scale factors for all three axes are determined by the geometrical position of the radial line. For the 3D butterfly grid, we only compute the Fourier domain information on the desired points where it overlaps with the radial lines of the spherical grid and not the entire $(N+1)^3$ Cartesian grid.

Similar to the partitioning in terms of levels in 2D, let p represent the level number for angle θ, q represent the level number for angle ϕ, then $1 \le p \le P$ and $1 \le q \le Q$, where P is the number of levels in angles θ and Q is the number of levels in angles ϕ, see Figure 12 for all the transformations of Cartesian to spherical coordinates for all angles. Inspired by the 2D polar divisions in terms of basically horizontal (X-axis) and basically vertical (Y-axis) lines,

- we first select two polar slices oriented along the X-axis with fixed angles $q\Delta\phi$ and $180 - q\Delta\phi$;
- from these two polar slices, we then select two planes oriented along the X-axis with angles $p\Delta\theta$ and $180 - p\Delta\theta$; and
- finally from these two planes we choose four lines that overlap exactly with the four unique radial lines of the spherical grid.

Since we have orientation for both angles θ and ϕ along the X axis, we call these four lines as forming a XX block. With a proper choice of scale factors we can compute a butterfly grid with this orientation that would overlap exactly with these four radial lines. Similarly, using a combination of complementary angles we can get XZ, YY and YZ blocks and compute a butterfly grid for each block containing four unique lines that belong to the spherical grid. This type of divisions of radial lines is crucial for determining the proper scale choices and the desired fast computations with 1D FrFT for the 3D spherical grid, see Figure 13 for the schematic of the spherical divisions. Note that the different scale factors chosen for different block arrangements are important, and so is the order of application of scale factors, because the first scale factor for each block is the uniform scaling factor for 1D FrFT and it is the most expensive one to calculate.

For the XX block, the 1D frequency domain scaling of a 3D image f along the X axis oriented polar slices can be computed using a 1D FrFT along the X axis (we slightly change our notation and indexing here compared to 2D equations) such that

$$\begin{aligned} F^{\alpha_{xx}^{pq}}(r,c,d) &= \sum_{n=-N/2}^{N/2} f(n,c,d) e^{-j\frac{2\pi r \alpha_{xx}^{pq} n}{N+1}} \\ \alpha_{xx}^{pq} &= \cos(p\Delta\theta)\cos(q\Delta\phi), \end{aligned} \tag{IV.3}$$

where $-N/2 \le r,c,d \le N/2$.

Next, we scale the transformed image $F^{\alpha_{xx}^{pq}}$ in Y axis and compute 1D FrFT. Now the equation becomes

$$\begin{aligned} F^{\alpha_{xx}^{pq},\beta_{xx}^{pq}}(r,c,d) &= \sum_{n=-N/2}^{N/2} F^{\alpha_{xx}^{pq}}(r,n,d) e^{-j\frac{2\pi c|c|\beta_{xx}^{pq} n}{q(N+1)}} \\ \beta_{xx}^{pq} &= \cos(p\Delta\theta)\sin(q\Delta\phi), \end{aligned} \tag{IV.4}$$

where $-N/2 \le r,c,d \le N/2$.

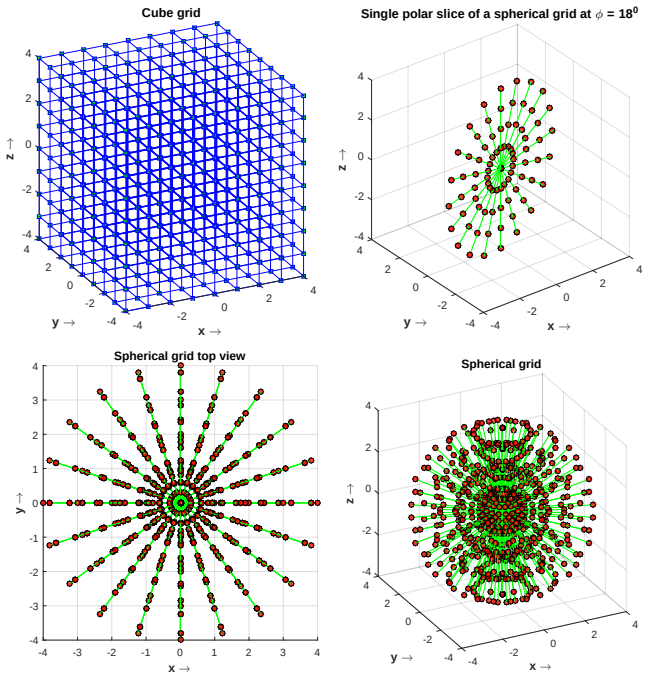

Fig. 12. 3D transformations required from spatial domain rectangular coordinates of image size $(N+1)^3$ to Fourier domain spherical coordinates grid of size $K \times M \times (N+1)$ where the origin is included multiple times.

Finally, we scale the transformed image $F^{\alpha_{xx}^{pq},\beta_{xx}^{pq}}$ in Z axis,

$$F^{\alpha_{xx}^{pq},\beta_{xx}^{pq},\gamma_{xx}^{p}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\alpha_{xx}^{pq},\beta_{xx}^{pq}}(r,c,n) e^{-j\frac{2\pi d|d|\gamma_{xx}^{p} n}{p(N+1)}}$$
$$\gamma_{xx}^{p} = \sin(p\Delta\theta), \tag{IV.5}$$

where $-N/2 \le r,c,d \le N/2$.

Figure 14 demonstrates the XX blocks with $p=1, q=1$ and $p=2, q=2$, respectively. We are only concerned with computing 2 complementary planes from (IV.4), since the goal is to eventually compute four lines in these two planes. After this operation we get a pair of complementary 2D planes of size $(N+1)^2$. Now, using the scaling in (IV.5), we compute two complementary lines for each plane. Those are the desired four radial lines forming the complete XX block, which is similar to the one shown in Figure 14(a).

We then repeat the above process for the XZ, YY, and YZ blocks to obtain a complete set of scales for the computation of the spherical polar grid (see Figures 13, 14 and the Appendix for detailed formulas), given by

$$S = \{\alpha_{xx}^{pq}, \beta_{xx}^{pq}, \gamma_{xx}^{p}, \gamma_{xz}^{p}, \alpha_{xz}^{pq}, \beta_{xz}^{pq}, \beta_{yy}^{pq}, \alpha_{yy}^{pq}, \gamma_{yy}^{p}, \gamma_{yz}^{p}, \alpha_{yz}^{pq}, \beta_{yz}^{pq},\ 1 \le p \le P, 1 \le q \le Q\},$$

where the cardinality of the set S is $12PQ$. Let the collection of all the radial lines oriented by the configuration of XX block be represented by F_{xx}. By choosing the appropriate scales from the set S, we have

$$F_{xx} = \bigcup_{p=1}^{P} \bigcup_{q=1}^{Q} F^{\alpha_{xx}^{pq},\beta_{xx}^{pq},\gamma_{xx}^{p}}. \tag{IV.6}$$

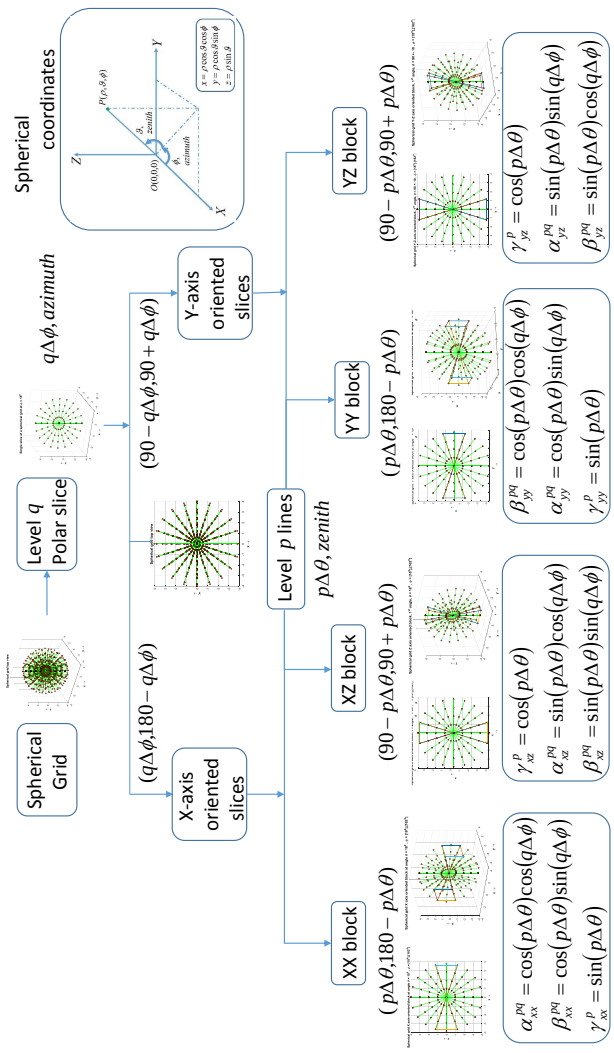

Fig. 13. Spherical divisions required for fast and exact computation of Spherical FT.

Similarly, we can write the equations for the other three blocks as follows:

$$F_{xz} = \bigcup_{p=1}^{P} \bigcup_{q=1}^{Q} F^{\gamma_{xz}^{p},\alpha_{xz}^{pq},\beta_{xz}^{pq}},$$
$$F_{yy} = \bigcup_{p=1}^{P} \bigcup_{q=1}^{Q} F^{\beta_{yy}^{pq},\alpha_{yy}^{pq},\gamma_{yy}^{p}},$$
$$F_{yz} = \bigcup_{p=1}^{P} \bigcup_{q=1}^{Q} F^{\gamma_{yz}^{p},\alpha_{yz}^{pq},\beta_{yz}^{pq}}. \tag{IV.7}$$

At $\phi = 0^o$, we have a single polar 2D slice (no block arrangement for this one), which we denote as F_x, and similarly, at $\phi = 90^o$ we have another slice denoted by F_y, both of these slices can be computed using the 2D polar FT in (II.4). Then the complete 3D SPFT information on the spherical grid defined in (IV.1), using equations (IV.3)– (IV.7), is given by

$$F = F_{xx} \bigcup F_{xz} \bigcup F_{yy} \bigcup F_{yz} \bigcup F_x \bigcup F_y. \tag{IV.8}$$

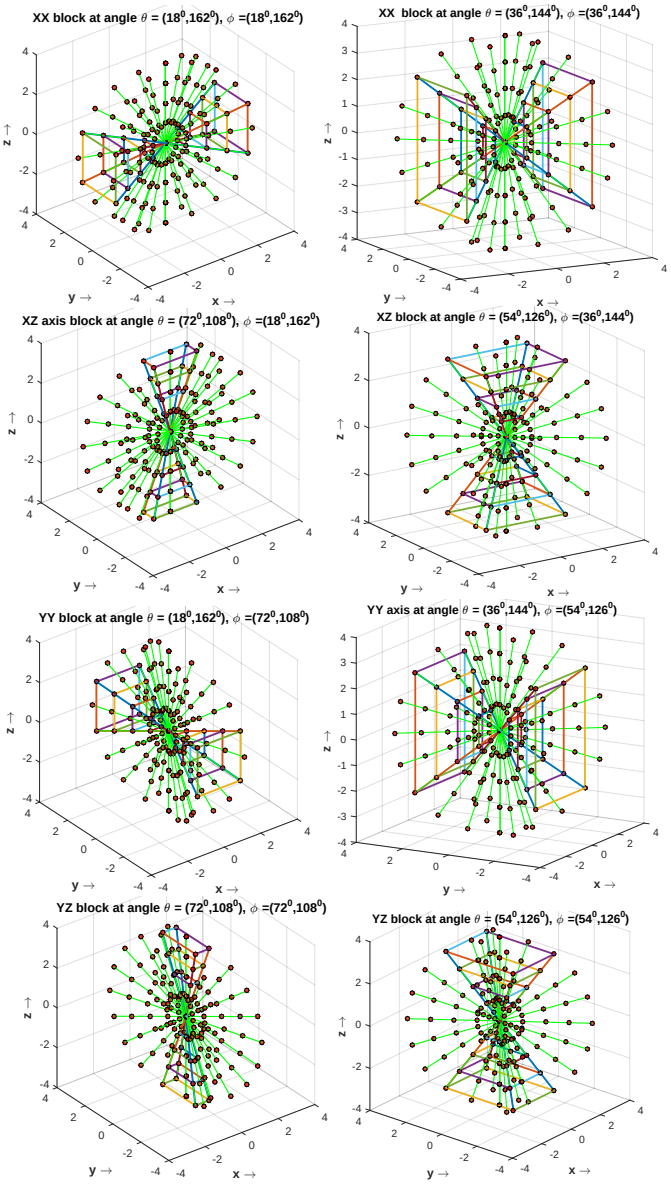

Fig. 14. 3D blocks at level $p = 1$, $q = 1$ (column left) and $p = 2$, $q = 2$ (column right) needed for fast computation of Spherical FT (from top to bottom) (a) XX-block (b) XZ-block (c) YY-block and (d) YZ-block.

To find the computational complexity of the above scheme, we follow a similar approach as in (II.14). The number of levels P required are approximately $P = (K-2)/4$, and the number of levels Q is approximately $Q = (M-2)/4$. At every level the most expensive operation is the first uniform scaling with arbitrary scale along the X, Y and Z axes, while the other two computations involve 2 planes and 4 radial lines, which are with far less complexity. Therefore, we can compute the 1D frequency domain uniform scaling with 1D FrFT along any axis of any scale, and the corresponding complexity is

$$(N+1)^3/2 \times \log_2(N+1).$$

Note that out of the 4 blocks, the first uniform scaling for both blocks XZ and YZ are the same. Therefore, we can use the same uniformly scaled grid with 1D FrFT along the Z axis for computations of the XZ and YZ blocks (the other scales still differ). Since there are now only 3 blocks corresponding to the three axis X, Y and Z with unique uniform scaling for each level, we have the reduced complexity with this sort of spherical divisions given by

$$3 \times \frac{M}{4} \times \frac{K}{4} \times \frac{(N+1)^3}{2} \log_2(N+1). \tag{IV.9}$$

Figure 15 shows the execution times for computations of discrete SPFT, and Figure 16 shows a simple 3D reconstruction example using the proposed solution. This successfully demonstrates that the application of Forward and Inverse SPFT. The inversion process is again accelerated by exploiting the nested 2-level deep block Toeplitz structure of the product of the transform operators similar to the one discussed in Section III. Since this is a simplified case where the signal (volume) is binary and well-padded with zeros, the reconstruction error is zero, however in the real world problems, partial data, distortions, additive noise, sampling errors and other sources of noise will cause reconstruction errors.

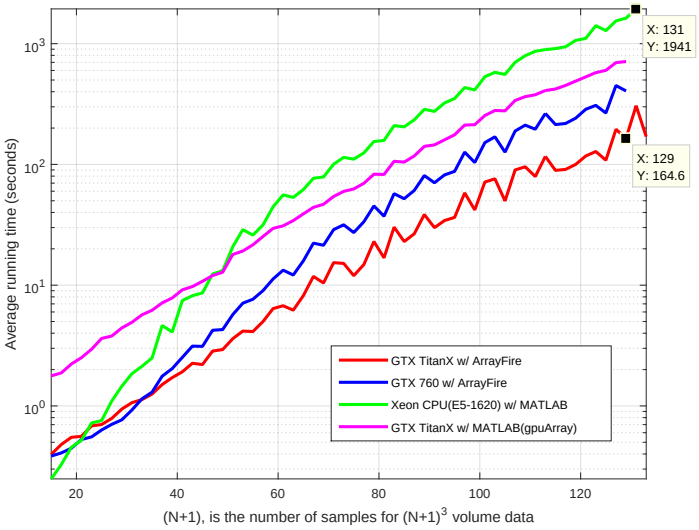

Fig. 15. A comparison for the execution times vs the input array size given by N. The brute force solution for $N = 128, (N+1)^3$ volume is of complexity $O((N+1)^6)$, and it takes about 4 hours on a single GeForce Titan X GPU, albeitto much longer on a CPU, but it just takes 32.35 minutes, using the proposed fast vectorized implementation on a CPU with reduced complexity. The best computational time is achieved with GPU accelerated library (CUDA) using ArrayFire [53] which is just, 2.7333 minutes on a single Titan X GPU. The proposed solution is orders of magnitude faster, while maintaining the similar level of accuracy as the brute force method. Faster solution for larger N can be achieved with additional parallelization as the proposed technique can be fully distributed and scales naturally with the computational resources.

Once again it should be emphasized that the computation of different levels is independent, and it can be implemented in parallel. Applications of the 3D FT is of particular importance for many numerical simulations (e.g. molecular dynamics) used in high performance computing codes. The parallelization of the 3D FFT is usually done using a 1D or 2D decomposition of the input data (see [51]), for our computations of the 3D SPFT also we have done a fully vectorized decomposition using 1D data only. So the proposed solution can be used in

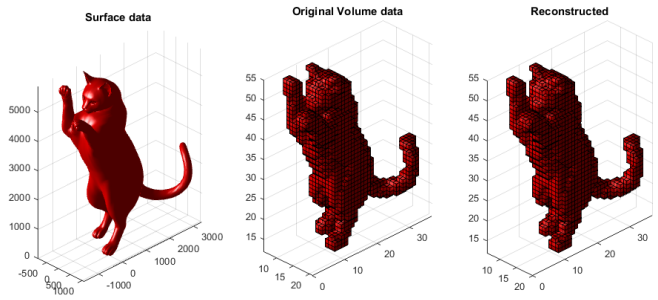

Fig. 16. Simple 3D reconstruction from a surface data (with given faces and vertices which is easier to visualize) transformed into volumetric data, $N = 54$, i.e. voxel data size is $55 \times 55 \times 55$, with $M = N + 14$, and $K = N + 10$. We obtain perfect reconstruction of surface data by simply thresholding the values for the reconstructed volume. This is simply due to the fact that the signal/volume is in binary form and well-padded with zeros i.e. the signal is concentrated in the center of the volume hence the corner artifacts are eliminated.

the areas of scientific computing where the computation of discrete, fast and exact solution for the 3D SPFT is required. The discrete definition of our algorithm can be used for Radon transform and X-ray transform both of which are established through the well-known Fourier slice theorem. The 2D PFT and 3D SPFT transforms will be useful for many applications, for instance, the rapid and exact Fourier volume rendering used in [52] and image registration in 2D/3D [35].

V. Contributions and Conclusions

Many algorithms in digital signal processing are based on the use of linear discrete signal transforms. Mathematically, such a transform is a matrix-vector multiplication $y = Ax$ where x is the sampled signal, and A is the transform over some base field, which in this paper represents 2D and 3D transforms reduced in terms of 1D. Each of these algorithms has been derived through exploiting the geometric structures that arise in 2D and 3D and insightful algebraic manipulation of the transform matrix entries.

The main contributions of the paper are as follows:

- We have designed a highly specialized method to find the discrete FT on the full polar grid and the spherical grid for images in the 2D and 3D spatial domain Cartesian coordinates.
- The forward 2D and 3D algorithms are exact and fast without any oversampling radially or angularly, and they do not require interpolation in the spatial or the frequency domain, or any approximation stage.
- The fully parallel algorithms for both the polar and spherical grids can take full advantage of commodity GPGPU processing hardware. Most importantly each of the components used in our solution has a parallelizable structure and scales naturally with processing and memory resources.
- The proposed solutions of PFT and SPFT solve the problem of inversion by iteratively operating on forward and adjoint transforms. They are based on algebraically exact computations, and they are computed rapidly by using preconditioned conjugate gradient solution to get accurate inversion.
- We drastically improve the conditioning of the inversion operation by using the proposed hybrid grid - Polar/Spherical Polar samples with Cartesian corners.
- We exploit the special matrix structure of block Toeplitz form, and we provide a solution to do a faster inversion.

Further research is needed to do a quantitative analysis taking in consideration different problem sizes, scalability, network constraints etc. More research is also needed to study these problems and transforms in the framework of modern algebra representation theory [54], which extends the relationship between signal processing and (abstract) algebra.

Appendix
Complete XZ, YY and YZ block formulas for SPFT

For XZ oriented blocks, first we uniformly scale 3-D image f along Z axis

$$\begin{aligned} F^{\gamma^p_{xz}}(r,c,d) &= \sum_{n=-N/2}^{N/2} f(r,c,n) e^{-j\frac{2\pi d \gamma^p_{xz} n}{N+1}} \\ \gamma^p_{xz} &= \cos(p\Delta\theta). \end{aligned} \tag{A.1}$$

Next we scale the transformed image $F^{\gamma^p_{xz}}$ in X axis using variable scale

$$\begin{aligned} F^{\gamma^p_{xz},\alpha^{pq}_{xz}}(r,c,d) &= \sum_{n=-N/2}^{N/2} F^{\gamma^p_{xz}}(n,c,d) e^{-j\frac{2\pi r|r|\alpha^{pq}_{xz} n}{p(N+1)}} \\ \alpha^{pq}_{xz} &= \sin(p\Delta\theta)\cos(q\Delta\phi). \end{aligned} \tag{A.2}$$

Finally we scale the transformed image $F^{\gamma^p_{xz},\alpha^{pq}_{xz}}$ in Y axis using variable scale

$$\begin{aligned} F^{\gamma^p_{xz},\alpha^{pq}_{xz},\beta^{pq}_{xz}}(r,c,d) &= \sum_{n=-N/2}^{N/2} F^{\gamma^p_{xz},\alpha^{pq}_{xz}}(r,n,d) e^{-j\frac{2\pi c|c|\beta^{pq}_{xz} n}{q(N+1)}} \\ \beta^{pq}_{xz} &= \sin(p\Delta\theta)\sin(q\Delta\phi), \end{aligned} \tag{A.3}$$

where $-N/2 \le r,c,d \le N/2$. Figure 14(b) demonstrates the XZ blocks with $p = 1, q = 1$ and $p = 2, q = 2$ respectively. Similar to the XX block we can use the above equations (A.1)–(A.3) to compute for the 4 radial lines for each pair p, q with this orientation.

Similarly, for YY block we perform 1D frequency domain uniform scaling of the 3D image f along Y axis oriented polar slices such that

$$\begin{aligned} F^{\beta^{pq}_{yy}}(r,c,d) &= \sum_{n=-N/2}^{N/2} f(r,n,d) e^{-j\frac{2\pi c \beta^{pq}_{yy} n}{N+1}} \\ \beta^{pq}_{xx} &= \cos(p\Delta\theta)\cos(q\Delta\phi). \end{aligned} \tag{A.4}$$

Next we can scale the transformed image $F^{\beta^{pq}_{yy}}$, in X axis, we can compute 1D FrFT along each column and the equation is given by

$$\begin{aligned} F^{\beta^{pq}_{yy},\alpha^{pq}_{yy}}(r,c,d) &= \sum_{n=-N/2}^{N/2} F^{\alpha^{pq}_{yy}}(n,c,d) e^{-j\frac{2\pi r|r|\alpha^{pq}_{yy} n}{q(N+1)}} \\ \alpha^{pq}_{yy} &= \cos(p\Delta\theta)\sin(q\Delta\phi). \end{aligned} \tag{A.5}$$

Finally we scale the transformed image $F^{\beta_{yy}^{pq},\alpha_{yy}^{pq}}$ in Z axis

$$F^{\beta_{yy}^{pq},\alpha_{yy}^{pq},\gamma_{yy}^{p}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\beta_{yy}^{pq},\alpha_{yy}^{pq}}(r,c,n) e^{-j\frac{2\pi d|d|\gamma_{yy}^{p} n}{p(N+1)}}$$
$$\gamma_{yy}^{p} = \sin(p\Delta\theta), \tag{A.6}$$

where $-N/2 \leq r,c,d \leq N/2$. Figure 14(c) demonstrates the YY blocks with $p=1,q=1$ and $p=2,q=2$ respectively. Similar to the XX block we can use the above equations (A.4)–(A.6) to compute for the 4 radial lines for each pair p,q with this orientation.

For YZ block first we uniformly scale 3-D image f along Z axis

$$F^{\gamma_{yz}^{p}}(r,c,d) = \sum_{n=-N/2}^{N/2} f(r,c,n) e^{-j\frac{2\pi d\gamma_{yz}^{p} n}{N+1}}$$
$$\gamma_{yz}^{p} = \cos(p\Delta\theta). \tag{A.7}$$

Next we scale the transformed data $F^{\gamma_{yz}^{p}}$ in X axis

$$F^{\gamma_{yz}^{p},\alpha_{yz}^{pq}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\gamma_{yz}^{p}}(n,c,d) e^{-j\frac{2\pi r|r|\alpha_{yz}^{pq} n}{q(N+1)}}$$
$$\alpha_{yz}^{pq} = \sin(p\Delta\theta)\sin(q\Delta\phi). \tag{A.8}$$

Finally we scale the transformed image $F^{\gamma_{yz}^{p},\alpha_{yz}^{pq}}$ in Y axis

$$F^{\gamma_{yz}^{p},\alpha_{yz}^{pq},\beta_{yz}^{pq}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\gamma_{yz}^{p},\alpha_{yz}^{pq}}(r,n,d) e^{-j\frac{2\pi c|c|\beta_{yz}^{pq} n}{p(N+1)}}$$
$$\beta_{yz}^{pq} = \sin(p\Delta\theta)\cos(q\Delta\phi), \tag{A.9}$$

where $-N/2 \leq r,c,d \leq N/2$. Figure 14(d) demonstrates the YZ blocks with $p=1,q=1$ and $p=2,q=2$ respectively. Finally we can use equations (A.7)–(A.9) to compute for the 4 radial lines for each pair p,q with their respective orientation.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions, which improved the presentation of this paper. We also thank Drs. Amir Averbuch, Daniel Potts and their groups for making codes of NFFT and PPFFT available to the public, which was helpful for our research.

REFERENCES

[1] R. M. Gray and J. W. Goodman, *Fourier Transforms*, Kluwer, 1995.
[2] R. N. Bracewell, *The Fourier Transform and its Applications*, McGraw Hill, 1986.
[3] B. Osgood, *The Fourier Transform and its Applications*, Lecture Notes, Stanford University.
[4] A. Averbuch, R. R. Coifman, D. L. Donoho, M. Elad, and M. Israeli,"Fast and accurate polar Fourier transform," *Applied and Computational Harmonic Analysis*, vol. 21, no. 2, pp. 145-167, 2006.
[5] M. Fenn, S. Kunis, and D. Potts, "On the computation of the polar FFT," *Applied and Computational Harmonic Analysis*, vol. 22, no. 2, pp. 257-263, 2007.
[6] W. Lawton, "A new polar Fourier transform for computer-aided tomography and spotlight synthetic aperture radar," *IEEE Trans. Acoustics Speech and Signal Processing*, vol. 36,pp. 931-933, 1988.
[7] W. L. Briggs, *The DFT: an Owners' Manual for the Discrete Fourier Transform*, SIAM, 1995.
[8] F. Natterer, *The Mathematics of Computerized Tomography*, vol. 32, SIAM, 1986.
[9] A. H. Delaney and Y. Bresler, "A fast and accurate Fourier algorithm for iterative parallel-beam tomography," *IEEE Trans. Image Processing*, vol. 5, no. 5, pp. 740-753, 1996.
[10] D. Gottleib, B. Gustafsson, and P. Forssén, "On the direct Fourier method for computer tomography," *IEEE Trans. Medical Imaging*, vol. 19, no. 3, pp. 223-232, 2000.
[11] S. Basu, and Y. Bresler, "$O(N^2 \log_2 N)$ filtered backprojection reconstruction algorithm for tomography," *IEEE Trans. Image Processing*, vol. 9, no. 10, pp. 1760-1773, 2000.
[12] S. Matej, J. Fessler, and I. G. Kazantsev, "Iterative tomographic image reconstruction using Fourier-based forward and back-projectors," *IEEE Trans. Medical Imaging*, vol. 23, no. 4, pp. 401-412, 2004.
[13] A. Averbuch, R. R. Coifman, D. L. Donoho, M. Israeli, and J. Walden, *Fast Slant Stack: A notion of Radon transform for data in a Cartesian grid which is rapidly computible, algebraically exact, geometrically faithful and invertible*, Department of Statistics, Stanford University, 2001.
[14] A. Averbuch and Y. Shkolnisky, "3D Fourier based discrete Radon transform," *Applied and Computational Harmonic Analysis*, vol. 15, no. 1, pp. 33-69, 2003.
[15] A. Averbuch, R. R. Coifman, D. L. Donoho, M. Israeli, and Y. Shkolnisky, "A framework for discrete integral transformations I-The pseudopolar Fourier transform," *SIAM Journal on Scientific Computing*, vol. 30, no. 2, pp. 764-784, 2008.
[16] A. Averbuch, R. R. Coifman, D. L. Donoho, M. Israeli, Y. Shkolnisky, and I. Sedelnikov, "A framework for discrete integral transformations II-the 2D discrete Radon transform," *SIAM Journal on Scientific Computing*, vol. 30, no. 2, pp. 785-803, 2008.
[17] A. Averbuch, G. Shabat, and Y. Shkolnisky, "Direct inversion of the three-dimensional pseudo-polar Fourier transform," *SIAM Journal on Scientific Computing*, vol. 38, no. 2, pp. 1100-1120, 2016.
[18] S. M. Hashemi, S. Beheshti, P. R. Gill, N. S. Paul, and R. S. C. Cobbold, "Accelerated Compressed Sensing Based CT Image Reconstruction," *Computational and Mathematical Methods in Medicine*, vol. 2015, article id-161797, pp. 16, doi:10.1155/2015/161797, 2015.
[19] J. Miao, F. Frster, and O. Levi, "Equally sloped tomography with oversampling reconstruction," *Physical Review B*, vol. 72, no. 5, 052103, 2005.
[20] G. Beylkin, "On the fast Fourier transform of functions with singularities," *Applied and Computational Harmonic Analysis*, vol. 2, no. 4, pp. 363-381, 1995.
[21] A. Dutt, and V. Rokhlin, "Fast Fourier transforms for nonequispaced data," *SIAM Journal on Scientific computing*, vol. 14, no. 6, pp. 1368-1393, 1993.
[22] A. Dutt, and V. Rokhlin, "Fast Fourier transforms for nonequispaced data II," *Applied and Computational Harmonic Analysis*, vol. 2, no. 1, pp. 85-100, 1995.
[23] D. Potts, G. Steidl, and M. Tasche, "Fast Fourier transforms for nonequispaced data: A tutorial," *Modern Sampling Theory*, Birkhäuser Boston, pp. 247-270, 2001.
[24] J. Fessler, and B. P. Sutton, "Nonuniform fast Fourier transforms using min-max interpolation," *IEEE Trans. Signal Processing*, vol. 51, no. 2, pp. 560-574, 2003.
[25] L. Greengard, and J. Y. Lee, "Accelerating the nonuniform fast Fourier transform," *SIAM Review*, vol. 46, no. 3, pp. 443-454, 2004.
[26] M. Jacob, "Optimized least-square nonuniform fast Fourier transform," *IEEE Trans. Signal Processing*, vol. 57, no. 6, pp. 2165-2177, 2009.
[27] Q. Sun, "Non-uniform average sampling and reconstruction of signals with finite rate of innovation," *SIAM Journal of Mathematical Analysis*, vol. 38, no. 5, pp. 1389–1422, 2006.
[28] S. A. Abbas, Q. Sun, and H. Foroosh, "Frequency estimation of sinusoids from nonuniform samples", *Signal Processing*, 2016.
[29] B. S. Reddy, and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *IEEE Trans. Image Processing*, vol. 5, no. 8, pp. 1266-1271, 1996.
[30] H. Foroosh, J. B. Zerubia, and M. Berthod, "Extension of phase correlation to subpixel registration," *IEEE Trans. Image Processing*, vol. 11, no. 3, pp. 188-200, 2002.
[31] W. Pan, K. Qin, and Y. Chen, "An adaptable-multilayer fractional Fourier transform approach for image registration." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 400-414, 2009.
[32] F. Nestler, "Automated parameter tuning based on RMS errors for nonequispaced FFTs." *Advances in Computational Mathematics*, DOI 10.1007/s10444-015-9446-8, 2016.

[33] N. Chou, J. Izatt, and S. Farsiu, "Generalized pseudo-polar Fourier grids and applications in registering ophthalmic optical coherence tomography images," *IEEE Signals, Systems and Computers Record of the Forty-Third Asilomar Conference*, pp. 807-811, 2009.

[34] D. Potts and G. Steidl, "A new linogram algorithm for computerized tomography," *IMA J. Numer. Anal.*, vol. 21, pp. 769-782, 2001.

[35] S. A. Abbas, and H. Foroosh, "Robust 3D Registration using Spherical Polar Fourier Transform and Spherical Harmonics," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Submitted.

[36] D. H. Bailey, and P. N. Swarztrauber,"The fractional Fourier transform and applications," *SIAM review*, vol. 33, no. 3, pp. 389-404, 1991.

[37] J. Jan, *Digital Signal Filtering, Analysis and Restoration*, INSPEC Telecommunications Series, 2000.

[38] S. W. Smith, *Digital Signal Processing*, Newnes, 2003.

[39] K. Pavel, and D. Svoboda, *Algorithms for Efficient Computation of Convolution*, INTECH Open Access Publisher, 2013.

[40] K. Pavel, and D. Svoboda, "Convolution of large 3D images on GPU and its decomposition," *EURASIP Journal on Advances in Signal Processing*, no. 1, pp. 1-12, 2011.

[41] P. Schaller, and G. Temnov, "Efficient and precise computation of convolutions: applying fft to heavy tailed distributions," *Computational Methods in Applied Mathematics* , vol. 8, no. 2, pp. 187-200, 2008.

[42] A. Jerri, *Integral and Discrete Transforms with Applications and Error Analysis*, CRC, 1992.

[43] J. C. Bowman, and M. Roberts, "Efficient dealiased convolutions without padding," *SIAM Journal on Scientific Computing*, vol. 33, no. 1, pp. 386-406, 2011.

[44] S. Kunis, and D. Potts, "NFFT, Software package, C subroutine library," 2002. Retrieved from: *https://github.com/NFFT/nfft*.

[45] M. Lustig, J. Tsaig, J. H. Lee, and D. L. Donoho, "Fast spiral Fourier transform for iterative MR image reconstruction," In *IEEE International Symposium Biomedical Imaging: Nano to Macro*, pp. 784-787, 2004.

[46] A. C. Rencher, and G. B. Schaalje, *Linear Models in Statistics*, John Wiley Sons, 2008.

[47] R. F. Bass, and K. Grchenig, "Random sampling of multivariate trigonometric polynomials," *SIAM Journal on Mathematical Analysis*, vol. 36, no. 3, pp. 773-795, 2005.

[48] T. Knopp, S. Kunis, and D. Potts, "A note on the iterative MRI reconstruction from nonuniform k-space data," *International Journal of Biomedical Imaging*, vol. 2007, no. 24727, 2017

[49] P. C. Hansen, J. G. Nagy, and D. P. O'leary, *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, vol. 3, 2006.

[50] T. Dray, and C. A. Manogue, *Conventions for Spherical Coordinates*, 2002.

[51] O. Ayala, and L. P. Wang, "Parallel implementation and scalability analysis of 3D fast Fourier transform using 2D domain decomposition," *Parallel Computing*, vol. 39, no. 1, pp. 58-77, 2013.

[52] B. Lichtenbelt, *Fourier volume rendering*, Hewlett-Packard Laboratories, Technical Publications Department, 1995.

[53] P. Yalamanchili, U. Arshad, Z. Mohammed, P. Garigipati, P. Entschev, B. Kloppenborg, J. Malcolm, and J. Melonakos, *ArrayFire - A high performance software library for parallel computing with an easy-to-use API*, Atlanta: AccelerEyes, 2016. Retrieved from *https://github.com/arrayfire/arrayfire*.

[54] M. Puschel, and J. M. F. Moura, "Algebraic signal processing theory: Foundation and 1-D time," *IEEE Trans. Signal Processing*, vol. 56, no. 8, pp. 3572-3585, 2008.

Syed Alam Abbas (S'12) received his B.E degree in Electronics from University of Mumbai, Mumbai, India, in 2007. He worked as a Software Engineer in GE India for three years till 2010. Since 2011 he has been with the Graduate Department of Electrical Engineering and Computer Science at University of Central Florida, Orlando, USA, where he is currently pursuing Ph.D. degree in Electrical Engineering. His research interests are developing new algorithms for computer vision and signal/image processing applications, also for computer graphics, discrete differential geometry, signal processing on graphs, directional wavelet systems, control theory-cooperation and intelligence in distributed multi-agent systems.

Qiyu Sun (M'16) received his Ph.D. degree in mathematics from Hangzhou University, China in 1990. He is currently a professor of Mathematics at the University of Central Florida, Orlando. His prior position was with the Zhejiang University (China), the National University of Singapore (Singapore), Vanderbilt University, Nashville, TN, and the University of Houston. His research interest includes sampling theory, phase retrieval, compressive sensing, wavelets, frame theory, and Fourier analysis. He has published more than 100 papers and written a book An Introduction to Multiband Wavelets with Ning Bi and Daren Huang

Hassan Foroosh (M'02 - SM'03) is a Professor of Computer Science at the University of Central Florida (UCF). He has authored and co-authored over 120 peer-reviewed journal and conference papers, and has been in the organizing and the technical committees of numerous international conferences. Dr. Foroosh is a senior member of the IEEE, and was an Associate Editor of the IEEE Transactions on Image Processing in 2003-2008 and 2011-2015. In 2004, he was a recipient of the Pierro Zamperoni award from the International Association for Pattern Recognition (IAPR). He also received the Best Scientific Paper Award in the International Conference on Pattern Recognition of IAPR in 2008. His research has been sponsored by NSF, NASA, DIA, Navy, ONR, and industry.