An exact and fast computation of Discrete Fourier Transform for Polar and Spherical grid

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

***Abstract*—Numerous applied problems of 2D and 3D imaging formulated in continuous domain places great emphasis on obtaining and manipulating the Fourier transform in Polar and Spherical coordinates respectively. But the translation of continuum ideas in practice with the discrete data sampled on a Cartesian grid is problematic. There exists no exact and fast solution to the problem of obtaining Discrete Fourier Transform (DFT) for Polar and Spherical grid. In this paper we develop exact algorithms for 2D and 3D which has the special feature that it involves only 1D equispaced FFT's and no interpolation or approximation at any stage. The result is we get the 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 transform. For inverse process we do the analysis and describe the special matrix structure that results due to the exact forward formulation. We show the accuracy of the result for inverse problem, for which we propose a hybrid grid having drastic improvement in the condition number, compared to the other girds in the literature, and solve it with the iterative procedure of preconditioned conjugate gradient using a simple pre-conditioner.**

***Index Terms*—Polar coordinates, Cartesian coordinates, Fast Fourier transform, Spherical coordinates, 2D, 3D imaging**

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), denoted $\hat{F}(u, v)$, is given by the integral [1]-[3],

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

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*,

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

with a slightly abused notation.

Numerous applications of this can be found especially in imaging. The significance of this trivial change of variables is that several fundamental procedures for manipulation of such continuum functions f (for instance rotation) can best be described most elegantly in terms of Polar variables. The Polar FT can be a powerful tool for organizing the understanding of operators and functions on the 2D continuum [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 May 18, 2016.

Although the continuum concepts are simple and obvious, in practice the data exists only in sampled digital form, such is the case in estimating image rotation, image registration, tomography or Radon transform, synthetic aperture radar imaging [6] and more. The prevailing belief is there exists no such algorithm for fast and exact computation of discrete polar FT, see [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 must 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]. The approach used here 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 other 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]-[26]. In this approach since the frequencies is by definition unequally spaced, the problem of evaluating Polar FT can be considered as special case of USFFT problem.

A detailed discussion of different approaches taken in the literature is present in [13], [4], [15], [16] and references therein. The method followed in these 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 and exact. This grid has been explored by many since the 1970s [4]. For long it has been argued that this is the best way to obtain polar-like sampling which is specially adapted for the digital data and hence it has proliferated to numerous applications (see [15]) as a substitute for the true polar form i.e. the true discrete analog of its continuous form. 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, yet 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 PP grid as an alternative to 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 again an oversampling factor. However for applications like image registration [27] which requires accurate computation of the true Polar FT from the given digital image, the method [4] and the related USFFT [24] [5] interpolation methods are very slow for high accuracy and it can be hard to choose the parameter of the oversampling rate, see [28], for an arbitrary image. For the computation of Polar grid [4], PP grid acts as a stepping stone. In order to obtain high accuracy, the PP grid must then be over sampled, both radially and angularly, the minimal oversampling required is 2 as discussed in [4]. The result is still interpolated and inaccurate, to improve the interpolation errors a multi-layered FFT (MLFFT) was introduced in [28]. With the use of MLFFT method the registration results are improved compared to the PP grid but again it requires the selection of cuts (of layers) which must be chosen carefully for it to work and the inaccuracies still exists since it also uses interpolation scheme in the end. Some other variants of the PP grid such as *Generalized* PP grid has been introduced which are said to be closer approximation to the true polar grid whose fast and accurate estimation has long been a major challenge for many image processing applications including phase-correlation based motion estimation [29]. The advantage of using a true polar grid instead of PP grid, Generalized PP, MLFFT grid or log polar grid is that we can do a straightforward computation of the phase correlation (due to uniformly spaced angles) operation in the polar Fourier domain for the registration problem, as it has been successfully demonstrated in our paper [30] for both 2D and 3D which is a direct application of the proposed algorithms in this paper.

There exists no direct and exact computational procedure for obtaining the true *Polar* FT (2D) or *Spherical Polar* FT (3D), that is fast (using FFT's) and yet devoid of any oversampling or interpolations (in [4], [5]) and doesn't require any design parameters (in [28]). Our approach in this paper borrows the ideas from the both literatures but it is uniquely in contrast to all the approaches discussed above, i.e. it doesn't 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 Fourier coefficients at grid points outside the true desired gird points of Polar and Spherical grids respectively. In the references [4],[5], [28] and [29], 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 paper is organized as follows: Section II discusses the proposed method in 2D setting, the mathematical solution is described and the implementation details are discussed, Section III discusses the inverse 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, Section IV discusses the formulation in 3D its solution, finally followed by the conclusions in section V.

II. The Proposed Method- 2D

A. Preliminaries: 1D Fractional Fourier Transform

Given a discrete 1D signal $f(n)$ over the support $-N/2 \leq n \leq N/2$, where N is even, and given an arbitrary scalar value α, the definition of 1-D fractional FT (FrFT) (also known as Chirp-Z transform) [31] is given by,

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

There is a fast algorithm for this that computes the above 1D FrFT in the order of $(N+1)\log_2(N+1)$ [31], [4]. Consider the factor in the exponent, $2kn = k^2 + n^2 - (k-n)^2$, we have,

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

Let us define a sequence,

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

then the above equation reduces to,

$$F^{\alpha}(k) = E(k) \sum_{n=-N/2}^{N/2} f(n)E(n)E(k-n), \quad -N/2 \leq k \leq N/2 \quad \text{(II.3)}$$

To evaluate the transform (II.1) using FFT, we use convolution in (II.3), we multiple $f(n)$ by $E(n)$ and compute its FFT, then we take FFT of $E(n)$ and multiply it to compute the convolution. Then we take 1 more IFFT to the product of 2 FFTs and post multiply it with $E(k)$ which gives us the desired sequence. As noted in [31] and [4], the FFT should be computed with zero padding to length $2(N+1)$ in order to avoid cyclic effects. A $(N+1)$ point 1-D FFT can be evaluated with operational cost $5(N+1)\log_2(N+1)$. Since three 1D-FFT's are needed, and since the zero padded sequence length is $2(N+1)$, we have a total operation cost of evaluating (II.1) as $30(N+1)\log_2(N+1)$ floating point operations (flops).

B. The Algorithm

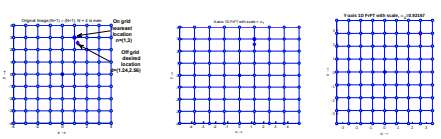

Fig. 1. Image with uniform grid, (a)Image with x-axis scaled grid (b)Image with xy-axis scaled grid

We begin with the idea of using 1D FrFT's to compute frequency domain information in 2D images for an arbitrary

point. Consider the 2D discrete data such as image $f(r,c)$ of size $(N+1)\times(N+1)$, where N is even, r and c are row and column indexes respectively. To compute the Fourier coefficient at any arbitrary off-grid location falling inside or outside grid, we can make use of separately scaled 1-D FrFT in X and Y direction, see for example Fig 1, with a total computational complexity of $30(N+1)^2\log_2(N+1)+(N+1)$, where the first term in the sum represents 1D FrFT in X or Y direction (order doesn't matter) for all rows or columns respectively, and the second term represents a direct computation 1D FrFT for a single point.

As shown in the Fig. 2, we define the Polar grid of frequencies $\{u_x(m,n), v_y(m,n)\}$ in the circle inscribed in the fundamental region, $\{u,v\}\in[N+1,N+1)^2$, and given digital Cartesian data $f(r,c)$ we define the Polar FT to be collection of samples $F(m,n)$, where it is defined by the trigonometric polynomial,

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

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

The polar grid is then 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 even. Each radial line oriented at a specific angle θ_m is composed of $N+1$ points. The polar grid can be considered as a structure $P(m,R_m)$, where R_m is the radial line data composed of uniformly spaced $N+1$ points oriented at angle θ_m. Such a definition is useful in context of Radon transform [13].

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

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

Similarly, the frequency domain scaling of a 2D image along Y axis can be computed by using 1D FrFT along each column and the expression is given by,

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

Fig. 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 *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 level 1 scaling

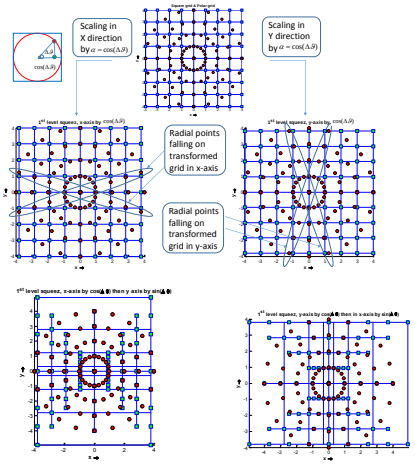

Fig. 2. Overlapping points of the polar grid with the rectangular scaled grid, each grid has exactly 4 radial lines that fall on it. X-axis scaling has horizontally oriented and Y-axis scaling has vertically oriented radial lines (both shown with ellipses).

which is composed of two solutions obtained using (II.5) and (II.6). Now to compute the 2D Fourier domain radial data on X-axis scaled grid, we take another 1D FrFT but with a variable scales for each column. For the column c, we scale it by a factor $|c|\beta$ where $\beta=\sin(\Delta\theta)$, then,

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

Similarly to compute 2D DFT data of radial lines on Y axis scaled grid, for each row r, the scale factor $|r|\beta$ is same, we have,

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

Observe that we are interested only in 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 needs to be computed. There are 4 symmetrical radial lines, 2 at an angle $\Delta\theta$ and $180^o-\Delta\theta$ from X axis scaled grid or the *basically horizontal lines* (see [13]), and 2 at angles $90^o\pm\Delta\theta$ from 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 overlaps exactly with the scaled butterfly grids in (II.7) and (II.8) respectively obtained using just 1D FrFT operations with properly selected scales α and β, see Fig 2 where the overlap is marked. This gives us the 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 the complete level 1 solution we denote as 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. M is even, the different radial

lines oriented in X-axis and Y-axis is given by angles, $\theta_l = l \times \Delta\theta$, l is an integer, and we have the corresponding scale factors $\alpha_l = \cos\theta_l$ and $\beta_l = \sin\theta_l$. Consider for level 1 using (II.7) we pick points that radially crosses the dc value by 1 point as shown in the figure 2, to get the desired radial line, similarly for level 2 we pick points that radially crosses the dc value by 2 points as shown in figure 3 to get the level 2 radial lines. The solution for all levels can then be written as:

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

and

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

The set of scales can be written as $S = [\alpha_l, \beta_l; 1 \le l \le L]$, where $2L$ is the cardinality of the set S. So the complete DFT information on the polar grid is then given as,

$$F = \bigcup_{l=1}^{L} F_l$$

In our case we have $M = 10$ therefore we require just $L = 8/4 = 2$ levels. Fig. 2 and 3 show the two levels F_1 and F_2 with the procedural trigonometric scaling applied in X and Y axis. This is the complete solution for the polar grid defined in top of figure 2.

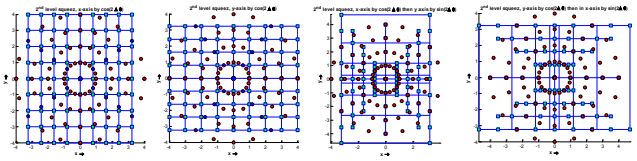

Fig. 3. (a)Image with x-axis scaled grid (b)Image with y-axis scaled grid. This completes all the lines of the polar grid with $M = 10, N = 8$ elements.

1) Exploiting Symmetry: If the scales α and β can be reduced to rational numbers, then the computations would simply reduce to conventional FFT which is based on exploiting periodicity of the data sequence to get the speed in computations. But it is meaningless to talk about periodicity when dealing with irrational numbers in the set of scales S. So we can exploit only the symmetry by first choosing M as even, the use of same scaling within each level gives us some added advantages. By using uniform scaling along X or Y axis we can obtain two symmetrical radial lines due to its conjugate symmetry in uniformly scaled grid, though due to the use of irrational scales there is hardly any connection across scales, however for the same scaling, α along X and Y axis, for the calculations in (II.5) and (II.6), we have the relation,

$$F_x^{\alpha}(r,c) = F_y^{\alpha}(r,c) \quad \text{(II.11)}$$

when $r = c$, or the diagonal elements. Element wise there are some unique common factors common to grids within a single level that need not be computed twice, with efficient use of caching we can improve performance,

$$f(r,c)e^{-i\frac{2\pi r\alpha n_1}{N+1}} = f(r,c)e^{-i\frac{2\pi c\alpha n_2}{N+1}} \quad \text{(II.12)}$$

when, $rn_1 = cn_2$.

2) Computational Complexity: The number of levels required are approximately $L = (M-2)/4$. When M is singly-even, i.e. M is divisible by 2, but not by 4 then the number of levels is exactly $L = (M-2)/4$, when M is doubly-even, i.e. M is divisible by 4, we have the special radial lines along the angles 45^o and 135^o hence we need to compute it only once using X-axis or Y-axis scaled grid, then the number of levels is $L' = \lfloor (M-2)/4 \rfloor + 1$. At every level, the most expensive operation is the first uniform scaling by α_l along X axis 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 for X axis scaling using real data,

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

Similarly for Y axis we have,

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

There are roughly $M/4$ levels we need to find, so the total computational complexity of our algorithm is,

$$M/4 \times (N+1)^2 \log_2(N+1) \quad \text{(II.13)}$$

This is due to the perfect symmetry of the equispaced points both angularly and radially. In the worst case when all the M radial lines are non-equispaced angularly then the computational complexity will increase to its maximum, $M(N+1)^2 \log_2(N+1)$. So 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. But when even the points on the radial lines are not equi-spaced we cannot use the 1D FFT based proposed solution and we must resort to approximation techniques of USFFT [5].

This gives us the true discrete Polar Fourier domain solution, (II.4) that is algebraically exact. It only involves the use of 1D FFT's and hence it is fully vectorizable, memory efficient and fast. It does not involve any design parameter choice for user, it involves no preprocessing, oversampling or padding of any kind. It should be emphasized that the computation of different levels is also completely independent, and it is easy to compute each level in parallel way on a GPU, multi-CPU, multi-GPU and grid computing (CPU/GPU) systems.

C. Implementation Details

Consider (II.1), the index is in the unaliased form, but the FFT routines in most software systems is zero based or aliased form (see. [31]), hence 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

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

Similarly, $k' = k + \frac{N}{2}$, then $k = k' - \frac{N}{2}$,

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

Let, $C(k') = e^{i\frac{\pi\alpha N k'}{N+1}}, B(n') = e^{i\frac{\pi\alpha N(n'-\frac{N}{2})}{N+1}}$,
then rearranging and simplifying we get,

$$\begin{aligned} F^\alpha(k' - \frac{N}{2}) &= C(k') \sum_{n'=0}^{N} \{f(n' - \tfrac{N}{2}) B(n')\} e^{-i\frac{2\pi\alpha n' k'}{N+1}}, \\ &\quad 0 \le k' \le N \end{aligned} \tag{II.14}$$

Thus for unalaised form, using the premultiplication factor B, and post multiplication factor, C, (II.14) reduces to zero based solution which can then be computed via convolution using FFT routines as described in (II.3).

The following is the pseudo code for the 2D Polar FFT computations which is mostly formula based calculations as discussed above.

Input: Image f of size $(N+1) \times (N+1)$ and M the number 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 levels l, $1 \le l \le L$, where $L \approxeq M/4$.

Step 2: Use the scaling α_l to compute row-wise 1D FrFT for the image f according to (II.7) .

Step 3: Use the scaling α_l to compute column-wise 1D FrFT for the image f according to (II.8) .

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 fast Polar Fourier Transform

Convolution is an important mathematical tool in both fields of signal and image processing [32]. It is employed in filtering, denoising, edge detection, correlation, compression, deconvolution, simulation, and in many other applications [33]. Although the concept of convolution is not new, the efficient computation of convolution is still an open topic [34].

While the convolution in time domain performs an inner product in each sample, in the Fourier domain, it can be computed as a simple point-wise multiplication. Due to this convolution property and the fast FT the convolution can be performed in time $O(N \log N)$. This approach is well known as *fast convolution* [34],[35], which we use in the computation of convolution in (II.3) to evaluate the 1D FrFT in (II.1) which in turn is used to compute the desired fast polar FT (II.4).

This use of fast FT for the numerical calculation of convolutions has a deep and fundamental problem of aliasing error associated with it [36]. This error manifests itself only for larger arrays while for the computations of smaller arrays the accuracy approaches machine precision. Figure 4 shows the precision loss for the computation of 1D FrFT, for randomly generated signal $f(n), -N/2 \le n \le -N/2$ and α where the maximum absolute error is given by,

$$E_{max} = max_n |F^\alpha_{direct}(n) - F^\alpha_{fast}(n)| \tag{II.15}$$

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

This fast convolution step is the only source of error present in the entire algorithm for Polar grid computations, all the other steps are exact upto machine precision since there are no interpolations, approximations or any form of oversampling. Also note that 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 i.e. 1D FrFT based on fast convolution, is the same for the computation of both grids. Also note that the zero padding used for computation of convolution (II.3) is not necessary, there are ways to compute this without padding with additional variables, see [38] for details.

The following are some of the techniques discussed in the literature, [34]-[37], for the precision improvement:

- Padding by zeros: This is the simplest and easiest tool to implement but with an obvious drawback, in order to eliminate the aliasing error a multiple of N grid points will be needed and this will slow down the calculation [36].
- Increasing the speed of tending to zero: A natural way to reduce the aliasing error in the FFT algorithm is to apply a transformation which decreases the function in use at the right end of the grid and which can be easily inverted right after the convolution procedure. The most simple of such transformations is the exponential one, $f_\tau(n) = f(n)e^{-n/\tau}$ where the parameter τ needs to be adjusted by the user, this transformation is called the exponential window following the terminology used mainly in signal analysis theory. The exponential windows are widely used, e.g. in visual signal processing, see [37].
- Splitting: If a direct computation of fast convolution of larger signals or images is not realizable using common computers one can reduce the whole problem to several subtasks which could benefit in speed, accuracy and memory requirements. In practice, this leads to splitting

the signal and kernel into smaller pieces, see [34] and [35] for details of this approach.

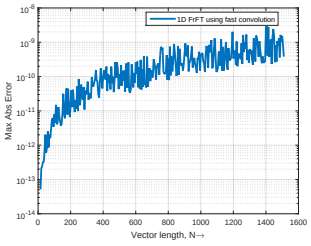

Fig. 4. Precision loss due to fast convolution, E_{max} vs. N. As can be seen from the figure that although there is a steady loss of precision for larger array sizes, for moderately sized array the error is still of order 10^{-10}. This is the same precision loss with the polar grid computations using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

E. Comparison

The following numerical example is computed with the NFFT C-subroutine library [39]. The parameters to choose are ρ, which is the cutoff parameter for Keiser-Bessel window functions, and the oversampling factor internally used is a constant $\sigma = 2$.

The NFFT evaluates the corresponding trigonometric polynomial (II.4) at $N \times M$ arbitrary nodes (in our case the polar grid) in

$$O((\sigma N)^2 \log(\sigma N)^2 + \rho^2 NM) \tag{II.16}$$

arithmetical operations.

Figure 5 shows the simulation for oversampled grid as well as normally sampled grid. Here we choose, $N = 128$, $M = 3N$ for an angularly oversampled grid and $M = N$ for normally sampled one. The relative accuracy of the NFFT is given by,

$$E_{max}^{rel} = \frac{max_{m,n}|F_{m,n} - F_{m,n}^{NFFT}|}{max_{m,n}|F_{m,n}|} \tag{II.17}$$

Thus for the simulation presented, the computational complexity of our algorithm for $M = 128$ ($M/4 = 32$) is, $32 \times 128^2 \log_2 128 = 224 \times 128^2$ and the accuracy is achieved is very high, while the complexity of NFFT is $O(18^2 \times 128^2 + 4 \times 128^2 \log_2 256) \approx O(324 \times 128^2)$, since $\rho = 18$ and it still is not that accurate i.e. E_{max}^{rel} is of order 10^{-3} as shown in figure 5. Also note that only in the oversampled case i.e. $M = 3N$ does the accuracy improves (as shown in the blue line), this is due to the fact that dense angular sampling reduces the interpolation errors for NFFT. But it is not advisable to arbitarily increase the cutoff parameter ρ as the error increases after certain threshold see 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_{max}^{rel} is of order 10^{-14} and the absolute error E_{max} is of order 10^{-10}. Finally the computations based on oversampling and interpolation of NFFT are likely to be more dependent on the signal content, while the proposed solution is not. NFFT is a good tool for a truly non-uniform

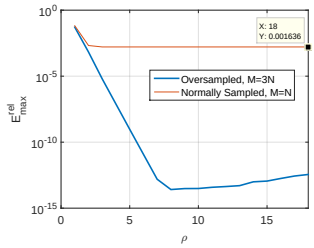

Fig. 5. In this simulation we study the accuracy and computational complexity of NFFT when compared with the proposed algorithm for the Polar grid.

sampling, for the polar sampling grid which has a lot of structure and symmetry to it, the proposed method proves to be more advantageous.

For the speed of computations comparison, the results are shown in the Figure 6. Although we get fast solution for 2D, in the later section for 3D computations, we show that by writing the CUDA level device kernels and using GPU-accelerated library the optimized code runs faster than the implementation directly done in MATLAB. Finally the proposed solution used to approximate other kinds of grid used in the literature, such as spiral grid [40], log-polar grid [28] 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.

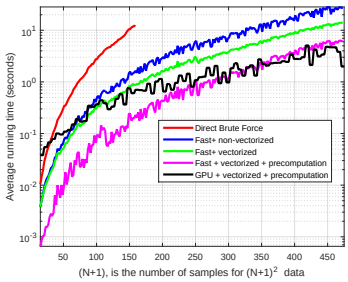

Fig. 6. A comparison for the execution times vs the input array size given by N where the images are of size $(N+1)^2$. As it is clear from the graph, the direct computation of Discrete Polar FT is of order $O(N^4)$ quickly grows in time. While the proposed fast technique with far reduced complexity can handle images of sufficiently large sizes. We use MATLAB gpuArray structure to speed up the operations on a GPU. We also get a speed up of operations with precomputations of pre and post multiplication vectors (see graphs with magenta and black lines) as discussed in (II.14). As is evident that the proposed solution is orders of magnitude faster, while maintaining the same level of accuracy as brute force computations up to machine precision. Faster solution for larger N can be achieved with parallelization as the proposed technique can be fully distributed and scales naturally with the computational resources.

III. Digital Inverse Problematics

Although 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 similar algorithm. In practice a fast way is needed for the reconstruction given the polar Fourier data especially for large size images. In the following sections we do the analysis and show how the iterative inversion scheme can be designed using linear algebra

techniques, we also discuss the ill-conditioning inherent in the polar sampling process and give solutions that drastically improves the conditioning of the designed linear systems.

A. Analysis

Since the proposed method described above is a linear operator, it corresponds implicitly to some matrix say A_p which is the discrete polar transform operator. This matrix representation is for analysis only, 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 Polar FT values F in (II.4) row-wise i.e the radial lines stacked sequentially, then the linear transformation can be represented as a matrix-vector product form,

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

which is the generic linear system of form $Ax = b$, here, 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 W_α where α is the scale factor can be defined as:

$$W_\alpha = \begin{bmatrix} \omega^{N^2/4} & \omega^{(N^2-2N)/4} & \dots & \omega^{N} & \omega^{N/2} & 1 & \omega^{-N/2} & \omega^{-N} & \dots & \omega^{-(N^2-2N)/4} & \omega^{-N^2/4} \\ \omega^{(N^2-2N)/4} & \omega^{(N/2-1)^2} & \dots & \omega^{N-2} & \omega^{(N/2-1)} & 1 & \omega^{-(N/2-1)} & \omega^{-(N-2)} & \dots & \omega^{-(N/2-1)^2} & \omega^{-(N^2-2N)/4} \\ \dots & \dots & \dots & \dots & \dots & 1 & \dots & \dots & \dots & \dots & \dots \\ \omega^{N/2} & \omega^{(N/2-1)} & \dots & \omega^{4} & \omega^{2} & 1 & \omega^{-2} & \omega^{-4} & \dots & \omega^{-(N/2-1)} & \omega^{-N/2} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \omega^{-N/2} & \omega^{-(N/2-1)} & \dots & \omega^{-4} & \omega^{-2} & 1 & \omega^{2} & \omega^{4} & \dots & \omega^{(N/2-1)} & \omega^{N/2} \\ \dots & \dots & \dots & \dots & \dots & 1 & \dots & \dots & \dots & \dots & \dots \\ \omega^{-(N^2-2N)/4} & \omega^{-(N/2-1)^2} & \dots & \omega^{-(N-2)} & \omega^{-(N/2-1)} & 1 & \omega^{(N/2-1)} & \omega^{(N-2)} & \dots & \omega^{(N/2-1)^2} & \omega^{(N^2-2N)/4} \\ \omega^{-N^2/4} & \omega^{-(N^2-2N)/4} & \dots & \omega^{-N} & \omega^{-N/2} & 1 & \omega^{N/2} & \omega^{N} & \dots & \omega^{(N^2-2N)/4} & \omega^{N^2/4} \end{bmatrix}$$

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) describing the uniform scaling along X axis can be written in matrix-vector product form as,

$$\mathbf{r}^\alpha_\mathbf{n} = W_\alpha \mathbf{r_n}, 1 \leq n \leq N+1$$

Stacking these new row vectors together we get the $(N+1) \times (N+1)$ matrix, F^α_x 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,

$$S_\beta = \begin{bmatrix} \xi^{N^2/8} & \xi^{(N^2(N/2-1))/4} & \dots & \xi^{N^2/2} & \xi^{N^2/4} & 1 & \xi^{-N^2/4} & \xi^{-N^2/2} & \dots & \xi^{(-(N^2(N/2-1))/4} & \xi^{-N^2/8} \\ \xi^{(N(N/2-1)^2)/2} & \xi^{(N/2-1)^3} & \dots & \xi^{2(N/2-1)^2} & \xi^{(N/2-1)^2} & 1 & \xi^{-(N/2-1)^2} & \xi^{-2(N/2-1)^2} & \dots & \xi^{-(N/2-1)^3} & \xi^{(-(N/2-1)^2)/2} \\ \dots & \dots & \dots & \dots & \dots & 1 & \dots & \dots & \dots & \dots & \dots \\ \xi^{N/2} & \xi^{N/2-1} & \dots & \xi^{2} & \xi^{1} & 1 & \xi^{-1} & \xi^{-2} & \dots & \xi^{-(N/2-1)} & \xi^{-N/2} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \xi^{-N/2} & \xi^{-(N/2-1)} & \dots & \xi^{-2} & \xi^{-1} & 1 & \xi^{1} & \xi^{2} & \dots & \xi^{N/2-1} & \xi^{N/2} \\ \dots & \dots & \dots & \dots & \dots & 1 & \dots & \dots & \dots & \dots & \dots \\ \xi^{(-(N(N/2-1)^2)/2} & \xi^{-(N/2-1)^3} & \dots & \xi^{-2(N/2-1)^2} & \xi^{-(N/2-1)^2} & 1 & \xi^{(N/2-1)^2} & \xi^{2(N/2-1)^2} & \dots & \xi^{(N/2-1)^3} & \xi^{(N(N/2-1)^2)/2} \\ \xi^{-N^3/8} & \xi^{(-(N^2(N/2-1)))/4} & \dots & \xi^{-N^2/2} & \xi^{-N^2/4} & 1 & \xi^{N^2/4} & \xi^{N^2/2} & \dots & \xi^{(N^2(N/2-1))/4} & \xi^{N^3/8} \end{bmatrix}$$

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,

$$S_\beta \circ F^\alpha_x$$

where $\circ$ is the Hadamard element wise product. The complementary line solution can be obtained from matrix, $S^H_\beta \circ F^\alpha_x$ where S^H_β represent the conjugate transpose or Hermitian transpose of S_β. Let $J = (1, 1, \cdots, 1)$, where J is a 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 [41]. Finally the line oriented at an angle $\Delta\theta$ and the complementary line oriented at an angle $180^o - \Delta\theta$ are given by,

$$R_1 = J^T(S_\beta \circ F^\alpha_x)$$
$$R_{M-1} = J^T(S^H_\beta \circ F^\alpha_x)$$

Similarly all the other radial lines can be computed in the above forms. Thus the matrix A_p has elements of form, $S_{\beta_l} \circ W_{\alpha_l}$ and $S^H_{\beta_l} \circ W_{\alpha_l}$ stacked either row-wise or column-wise, for all the trigonometric 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. Separately we can express A_p as a product of two matrices,

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

where V is a large and sparse matrix that is entirely composed of values obtained using terms of β_l and U is also a large and sparse matrix that is obtained from terms of α_l. The *adjoint* operator of A_p is given by,

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

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

$$A^H_p A_p x = A^H_p y$$

Let $A^+_p = (A^H_p A_p)^{-1} A^H_p$ which is the generalized inverse or the well known Moore Penrose pseudo inverse, the solution 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 Polar FT transform and the associated reconstruction error heavily relies on the distribution of the polar sampled nodes y. The problem inherent in the use of polar coordinate system is due to the geometry of the polar grid, it leaves out the corners of the square grid. Since the corners are not represented, this leads to some loss of information. From the algebraic point of view, the matrix A_p in (III.1) is ill-conditioned and the condition number quickly grows as N the number of samples in spatial grid increases. From the reconstruction perspective, visible artifacts are left and it will require numerous iterations to achieve desired low residual error and the solution obtained will be highly sensitive to numerical errors in the measurement process.

One way to counter this problem of missing corners is simply to extend the sampling set as suggested in [5] for corner compensation. There a modified polar grid is proposed,

which has the polar sampling set with more concentric circles added. Those nodes that are not located in the square grid are excluded. If the number of polar grid nodes is $M \times (N+1)$ with origin included multiple times, then 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 polar FT operator A_p can be seen in figure 8. The extra measurements improves the conditioning of the system but it is still not drastic improvement, through a different arrangement we can further stabilize the inverse transform. Rather than use modified grid discussed above, we propose the hybrid grid, where with the pure polar grid we include the corners from the Cartesian grid as shown in fig 7 (v) and (vi).

Figure 7 shows the different sampling grids used and Fig 8 shows the effect on the condition number on 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 can be seen in the 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 [42] and [5] discuss about the density of 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 using the proposed hybrid grid it can be easily shown (similar to [43]) that the reconstruction problem is well conditioned.

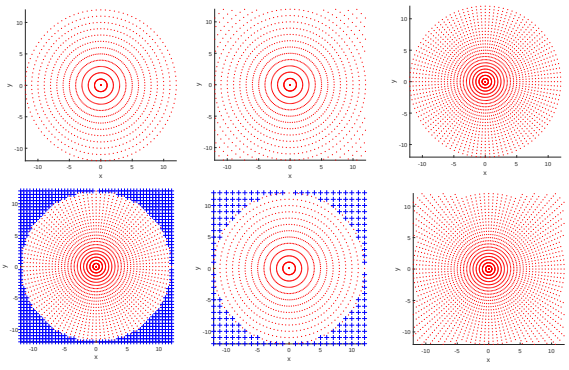

Fig. 7. Clockwise from top left, (i)Polar grid (pure), (ii) Polar grid with radial corners included, (iii) Polar grid 2x radial sampling, (iv) Polar grid 2x radial sampling with radial corners included, (v) Polar grid with Cartesian corners (hybrid), and (vi) Polar grid 2x radial sampling with Cartesian 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, it is useless as a computational approach for reasonable sizes of input arrays, a pseudo direct inversion through the matrix A_p^+ is computationally prohibitive, and definitely beyond the desired low complexity and high speed of the algorithm.

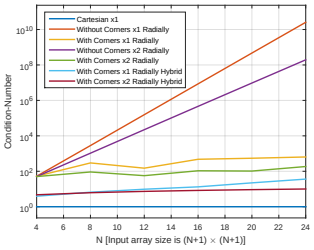

Fig. 8. Improving the ill-conditioning of the linear system in (III.1) by inclusion of additional measurements. Without the inclusion of corners the condition number of A_p even for really small sizes of grids (25×25) is very high (order of 10^{10}). Inclusion of corners (see [5]) along radial direction 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.

For iterative image reconstruction algorithms, one also needs the *adjoint* A_p^H, of the discrete Polar FT operation A_p in (III.2), i.e. one must perform matrix-vector multiplications of the form $A_p^H y = \widetilde{x}$. Since A_p itself is too large to store in the imaging problems of interest, and since the direct matrix-vector multiplication would be computationally inefficient, we must 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). From the above discussion, (II.7),(II.8), (II.9) and (II.10), it is fairly easy to show that the adjoint is computable with the same complexity as the forward transform.

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

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

Due to the variation in the local sampling density of the points in the polar grid and its modification, weights are introduced to compensate for it effects. We can use the formula for weights either in [4] or [5] which are mostly similar in concept. The preconditioner or weights W is used in the following way, rather than minimizing $||A_p x - y||_2^2$, we use

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

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

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

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 near origin points in order to equalize the condition number.

For corner compensation as discussed in the previous section we have to 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 only 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 corners only. Then using the augmented system $[A_p : A_{cc}]$ with augmented output $[y : y_c]$, the new symmetric positive definite linear system becomes,

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

where,

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

A way around this problem is either we try to suppress or remove the fixed pattern noise by a suitably designed filtering in the post processing stage (similar to filtered back projection) or to assume that the image is concentrated only in the center. If the given image is padded by a factor $\sqrt{2}$ in both axes, and applying 2D-FFT followed by 2D-IFFT, the resultant image is twice as big in pixel count and its frequency domain support is concentrated in the center. These operations only add $(N+1)^2 \log_2(N+1)$ operations to the process. Figure 9 shows the reconstruction with padding. From the reconstructed image using the corner compensation and extra padding by factor of $\sqrt{2}$, as shown we get near perfect reconstruction. In all the figures we have the initialization of image as all zeros for first iteration. For corner values we assume that there is no corner Fourier information present and image is concentrated at its center i.e. the values for vector y_c are all set to zero, this is similar to the approach followed in [4].

D. Faster Inversion by Structured Matrix Computations

Although we do get a fast iterative reconstruction based on linear algebra analysis and the use of conjugate gradient technique, we have not fully exploited the underlying special structure of the matrix A.

Toeplitz matrices are dense, but very structured matrices that arise in a variety of applications, for example in deblurring images [44], radon transform and image reconstruction, the discretization process of partial differential equations, etc. The matrix A is a real symmetric two-level toeplitz. 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, also having Toeplitz structure on the block level. Block-circulant-circulant-block (BCCB) matrices are the 2D analogues of circulant matrices. They are circulant within each block and on the block level. BCCB matrices are diagonalized efficiently by the 2D FFT. With the use of the concept of circulant embedding we can transform a real symmetric BTTB matrix into a larger real symmetric BCCB matrix.

The BTTB system is of the form,

$$Ax = b$$

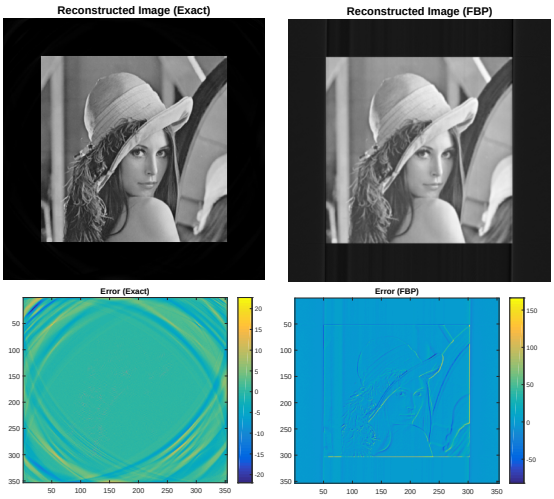

Fig. 9. (Top row) Corner compensated and padded reconstruction, where $M = 2(N+1)$, and residual error is $= 8.7 \times 10^{-5}$ after just 12 iterations. Here we get a near perfect reconstruction at the centered image. Observe that the sharp features have been destroyed by the use of Filtered back propagation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

The BTTB system can be extended to a BCCB system, especially simple is the case of real symmetric BTTB matrices. If the input image stacked vector x is of size, $(N+1)^2 \times 1$, then 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 discrete FT (DFT) matrix and the diagonal matrix Σ represents the eigenvalues of $\tilde{A}$ [44]. The specialty of this computation lies in that neither F nor F^H needs 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 $\tilde{x}$, multiplied element wise with the 2D FFT of the first column of $\tilde{A}$ arranged in 2D array, and one final 2D IFFT of the product to get the multiplication $\tilde{A}\tilde{x}$ result.

For the computation of system (III.8) using 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

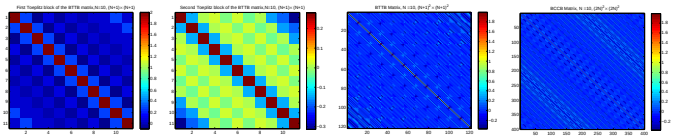

Fig. 10. BTTB to BCCB conversion, performing extension of real symmetric BTTB matrix using concept of embedding.

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. For each subsequent iteration of conjugate gradient algorithm the operation Ax for any arbitrary vector x now is of complexity $O((N+1)^2 \log(N+1))$, which is independent of the number of angles M but only depends on the image size $(N+1)$. Also unlike the single level toeplitz systems discussed in [15], [16] and [17] for PP grid which gives a direct inversion algorithm, higher level (≥ 2) toeplitz systems do not have a direct closed form inverse solution, 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 T-SVD, Tikhonov, etc (see [44]). methods can be used in presence of noise for further stability.

Thus the fast forward and inverse transform discussed above is based on algebraically exact formulation and has full geometric fidelity, and compared to both the USFFT and the pseudo-polar grid approach it can provides the sparsest representation devoid of any unnecessary oversampling. The USFFT and the pseudo-polar grid both require oversampling factors (≥ 2) both angularly and radially, and the available implementations [4],[5],[13] and [14] for discrete data in terms of pseudo-polar grid or modified grid are only vaguely related and not precise analogs to the continuum transforms.

IV. The Proposed Method- 3D

There is no exact Polar sampling method for 3D well defined, thus we must settle for Polar-like grids [4]. Extension to 3D will require the definition of Polar-like coordinates. 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 $\theta \in [0, 180]$ and *azimuth* angle $\phi \in [0, 360]$. 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, see [45]. 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 polar elevation angle or *zenith* angles from the $x-y$ 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 depth indexes respectively, we define the *Spherical Polar* or *Spherical* FT to be the collection of samples $F(k, m, n)$, defined by the trigonometric polynomial similar to 2D form in (II.4) given by,

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

where, $-N/2 \leq n \leq -N/2$ represents the points on a radial line, $\theta_k = k\Delta\theta, 0 \leq k \leq K-1$ represents the *zenith* angles, $\Delta\theta = 180^o/K$ is the angular spacing in *zenith* angles, $\phi_m = m\Delta\phi, 0 \leq m \leq M-1$ represents 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 Spherical Polar transform as,

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

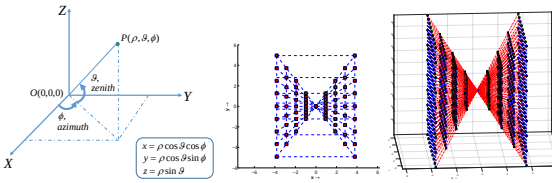

Fig. 11. (From left to right) Spherical coordinate conventions, Butterfly grid 2D and 3D. Conventions followed here are shown. We measure the *azimuth* angle with respect to the x-axis and *zenith* angle with respect to the x-y plane. In 2D the butterfly grid is a set of concentric rectangles and in 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 2D case shown in figure 2, since this grid has special selection of points that exactly match the positions of the desired radial points of the polar grid. Similarly we have a 3D version of the butterfly grid, which is a set of concentric rectangular points in 3D 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 axis is determined by the geometrical position of the radial line. For 3D butterfly grid also 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 of 2D as discussed in section II B, let p represent the level number for angle θ, q represent the level number for angle ϕ, then $1 \leq p \leq P$, where P is the number of levels in angles θ, and $1 \leq q \leq Q$, where Q is the number of levels in angles ϕ, see figure 12 for all the transformation of Cartesian to

Spherical coordinates for all angles. Then similar to the 2D polar divisions in terms of *basically horizontal* (X-axis) and *basically vertical* (Y-axis) lines, we can first select two polar slices oriented along X-axis with fixed angles $q\Delta\phi, 180-q\Delta\phi$. Further from the selected two polar slices of a spherical grid, we can select two planes (just like lines for 2D case) oriented along X-axis with angles $p\Delta\theta, 180-p\Delta\theta$. Further from these two planes we can 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 X-axis we will 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 belonging 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 the figure 13 for the schematic of the spherical divisions, note the different scale factors chosen for different block arrangements, also the order of application of scale factors is important, since the first scale factor for each block is the uniform scaling factor for 1D FrFT and it is the most expensive one to calculate.

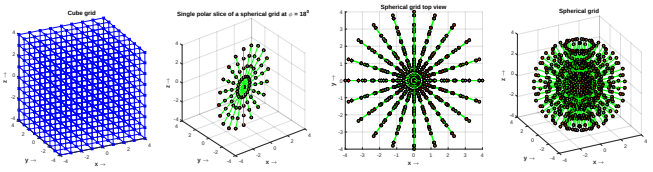

Fig. 12. 3D transformations required from spatial domain rectangular coordinates of image size $(N+1)^3$ to Fourier domain spherical coordinates grid of size $K \times M \times (N+1)$ where the origin is included multiple times.

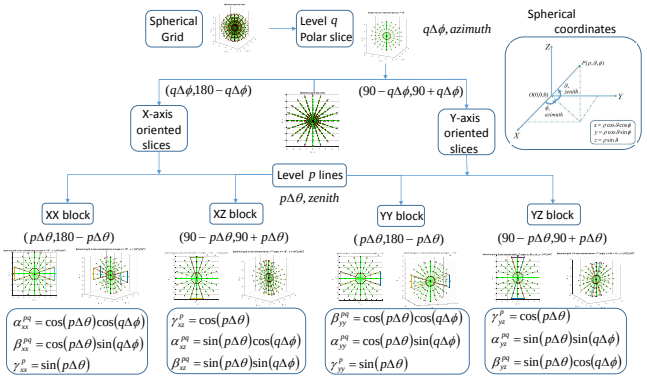

Fig. 13. Spherical divisions required for fast and exact computation of Spherical FT.

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

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

Next we can scale the transformed image $F^{\alpha_{xx}^{pq}}$ in Y-axis, we can compute 1D FrFT and the equation is given by,

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

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

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

See figure 14 for 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 target is to compute four lines in these two planes. So after this operation we get a pair of complementary 2D planes of size $(N+1)^2$. Now using the scaling as defined in (IV.5), for each plane we compute two complementary lines which are the desired four radial lines forming the complete XX-block similar to one shown in figure 14(a).

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

$$F^{\gamma_{xz}^{p}}(r,c,d) = \sum_{n=-N/2}^{N/2} f(r,c,n)e^{-j\frac{2\pi d\gamma_{xz}^{p}n}{N+1}}$$
$$\gamma_{xz}^{p} = \cos(p\Delta\theta),$$
$$r,c,d = -\frac{N}{2},\cdots,\frac{N}{2} \tag{IV.6}$$

Next we scale the transformed image $F^{\gamma_{xz}^{p}}$ in X-axis using variable scale,

$$F^{\gamma_{xz}^{p},\alpha_{xz}^{pq}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\gamma_{xz}^{p}}(n,c,d)e^{-j\frac{2\pi r|r|\alpha_{xz}^{pq}n}{p(N+1)}}$$
$$\alpha_{xz}^{pq} = \sin(p\Delta\theta)\cos(q\Delta\phi),$$
$$r,c,d = -\frac{N}{2},\cdots,\frac{N}{2} \tag{IV.7}$$

Finally scale the transformed image $F^{\gamma_{xz}^{p},\alpha_{xz}^{pq}}$ in Y-axis using variable scale ,

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

See figure 14 for 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 (IV.6)-(IV.8) 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,

$$F^{\beta_{yy}^{pq}}(r,c,d) = \sum_{n=-N/2}^{N/2} f(r,n,d)e^{-j\frac{2\pi c\beta_{yy}^{pq}n}{N+1}}$$
$$\beta_{xx}^{pq} = \cos(p\Delta\theta)\cos(q\Delta\phi),$$
$$r,c,d = -\tfrac{N}{2},\cdots,\tfrac{N}{2} \quad \text{(IV.9)}$$

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

$$F^{\beta_{yy}^{pq},\alpha_{yy}^{pq}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\alpha_{yy}^{pq}}(n,c,d)e^{-j\frac{2\pi r|r|\alpha_{yy}^{pq}n}{q(N+1)}}$$
$$\alpha_{yy}^{pq} = \cos(p\Delta\theta)\sin(q\Delta\phi),$$
$$r,c,d = -\tfrac{N}{2},\cdots,\tfrac{N}{2} \quad \text{(IV.10)}$$

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

$$F^{\beta_{yy}^{pq},\alpha_{yy}^{pq},\gamma_{yy}^{p}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\beta_{yy}^{pq},\alpha_{yy}^{pq}}(r,c,n)e^{-j\frac{2\pi d|d|\gamma_{yy}^{p}n}{p(N+1)}}$$
$$\gamma_{yy}^{p} = \sin(p\Delta\theta),$$
$$r,c,d = -\tfrac{N}{2},\cdots,\tfrac{N}{2} \quad \text{(IV.11)}$$

See figure 14 for 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 (IV.9)-(IV.11) to compute for the 4 radial lines for each pair p, q with this orientation.

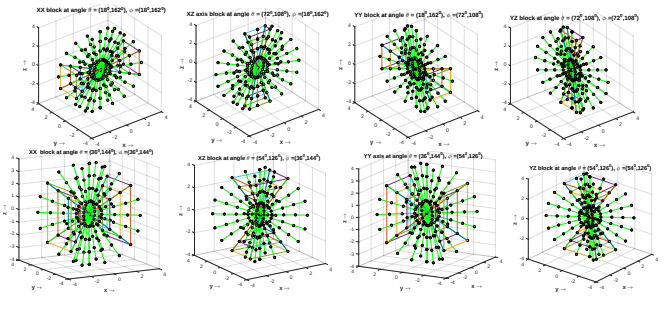

Fig. 14. 3D blocks at level $p = 1$, $q = 1$ (top row) and $p = 2$, $q = 2$ (bottom row) needed for fast computation of Spherical FT (from left to right) (a) XX-block (b)XZ-block (c)YY-block and (d)YZ-block

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

$$F^{\gamma_{yz}^{p}}(r,c,d) = \sum_{n=-N/2}^{N/2} f(r,c,n)e^{-j\frac{2\pi d\gamma_{yz}^{p}n}{N+1}}$$
$$\gamma_{yz}^{p} = \cos(p\Delta\theta),$$
$$r,c,d = -\tfrac{N}{2},\cdots,\tfrac{N}{2} \quad \text{(IV.12)}$$

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

$$F^{\gamma_{yz}^{p},\alpha_{yz}^{pq}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\gamma_{yz}^{p}}(n,c,d)e^{-j\frac{2\pi r|r|\alpha_{yz}^{pq}n}{q(N+1)}}$$
$$\alpha_{yz}^{pq} = \sin(p\Delta\theta)\sin(q\Delta\phi),$$
$$r,c,d = -\tfrac{N}{2},\cdots,\tfrac{N}{2} \quad \text{(IV.13)}$$

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

$$F^{\gamma_{yz}^{p},\alpha_{yz}^{pq},\beta_{yz}^{pq}}(r,c,d) = \sum_{n=-N/2}^{N/2} F^{\gamma_{yz}^{p},\alpha_{yz}^{pq}}(r,n,d)e^{-j\frac{2\pi c|c|\beta_{yz}^{pq}n}{p(N+1)}}$$
$$\beta_{yz}^{pq} = \sin(p\Delta\theta)\cos(q\Delta\phi),$$
$$r,c,d = -\tfrac{N}{2},\cdots,\tfrac{N}{2} \quad \text{(IV.14)}$$

See figure 14 for the YZ blocks with $p = 1, q = 1$ and $p = 2, q = 2$ respectively. Finally we can use equations (IV.12)-(IV.14) to compute for the 4 radial lines for each pair p, q with their respective orientation.

The set of scales are,

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

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

$$F_{xx} = \bigcup_{p=1}^{P}\bigcup_{q=1}^{Q} F^{\alpha_{xx}^{pq},\beta_{xx}^{pq},\gamma_{xx}^{p}} \quad \text{(IV.15)}$$

Similarly we can write the equations for other three blocks given by,

$$F_{xz} = \bigcup_{p=1}^{P}\bigcup_{q=1}^{Q} F^{\gamma_{xz}^{p},\alpha_{xz}^{pq},\beta_{xz}^{pq}} \quad \text{(IV.16)}$$

$$F_{yy} = \bigcup_{p=1}^{P}\bigcup_{q=1}^{Q} F^{\beta_{yy}^{pq},\alpha_{yy}^{pq},\gamma_{yy}^{p}} \quad \text{(IV.17)}$$

$$F_{yz} = \bigcup_{p=1}^{P}\bigcup_{q=1}^{Q} F^{\gamma_{yz}^{p},\alpha_{yz}^{pq},\beta_{yz}^{pq}} \quad \text{(IV.18)}$$

At $\phi = 0^o$ we have a single polar 2D slice (no block arrangement for this one) which we denote as F_x, and similar 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).

The complete 3D *Spherical Polar* FT information on the spherical grid defined in (IV.1), using equations (IV.3)-(IV.14) together with (IV.15)-(IV.16) is given by,

$$F = F_{xx}\bigcup F_{xz}\bigcup F_{yy}\bigcup F_{yz}\bigcup F_x\bigcup F_y \quad \text{(IV.19)}$$

For finding the computational complexity of the above scheme, we follow similar approach as in (II.13), the number of levels P required are approximately $P = (K-2)/4$, and the number of levels Q is approximately $Q = (M-2)/4$. Now at every level the most expensive operation is the first

uniform scaling with arbitrary scale along X-axis, Y-axis or Z-axis, while the other two computations are involving 2 planes and 4 radial lines which are far lesser in complexity. Then using real data, we can compute the 1D frequency domain uniform scaling with 1D FrFT along any axis with any scale with the complexity,

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

Note that out of the 4 blocks, the first uniform scaling for both blocks XZ and YZ are the same, see (IV.6) and (IV.12). So we can use the same uniformly scaled grid with 1D FrFT along Z axis for both the computations of 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,

$$3 \times \frac{M}{4} \times \frac{K}{4} \times \frac{(N+1)^3}{2} \log_2(N+1) \qquad \text{(IV.20)}$$

Figure. 15 shows the execution times for computations of discrete Spherical Polar FT, and Fig 16 shows a simple 3D reconstruction example using the proposed solution. This successfully demonstrates that the application of Forward and Inverse Spherical Polar FT. 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.

Once again it should be emphasized that the computation of different levels is also completely independent, and it is easy to compute each level in parallel. Applications of 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 [46]), for our computations of 3D spherical FT also we have done a fully vectorized decomposition using 1D data only. So the proposed solution can used in the areas of scientific computing where the computation of discrete, fast and exact solution for the 3D *Spherical Polar* FT is required. The discrete definition of our algorithm can be used for Radon transform and X-ray transform both of which are established through a sound Fourier slice theorem. This transform will be useful for many applications like for instance, the rapid and exact Fourier volume rendering used in [47] and it can be directly used in image registration in 2D/3D [30].

V. 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 arises in 2D and 3D and insightful algebraic manipulation of the transform matrix entries. More research is needed to study these problems and trigonometric transforms in the framework of modern algebra representation theory [49], which extends the relationship between signal processing and (abstract) algebra.

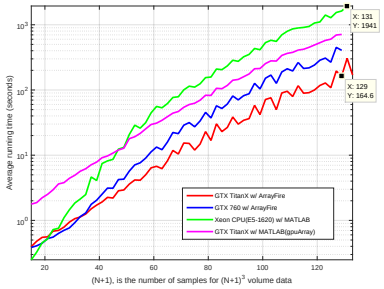

Fig. 15. A comparison for the execution times vs the input array size given by N. The brute force solution for $N = 128$, $(N+1)^3$ volume takes about 4 hours, which is of complexity $O((N+1)^6)$, to compute on a single GPU and much more higher on a CPU, but it just takes $1941/60 = 32.35$ minutes, using the proposed fast vectorized implementation on a CPU with reduced complexity. The best computational time is achieved with GPU accelerated library using ArrayFire CUDA [48] which is just, $164/60 = 2.7333$ minutes on a single Titan X GPU. As is evident that the proposed solution is orders of magnitude faster, while maintaining the same level of accuracy as brute force computations up to machine precision. Faster solution for larger N can be achieved with parallelization as the proposed technique can be fully distributed and scales naturally with the computational resources.

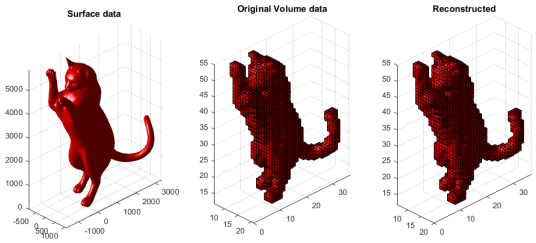

Fig. 16. Simple 3D reconstruction from a surface data (with given faces and vertices which is easier to visualize) transformed into volumetric data, $N = 54$, i.e. voxel data size is $55 \times 55 \times 55$, with $M = N + 14$, and $K = N + 10$. Here we obtain perfect reconstruction of surface data by 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 main contributions of the paper can be summarized as follows:

1. We have designed a highly specialized method to find the discrete FT on the full polar grid and the spherical grid, given the image in 2D and 3D spatial domain Cartesian coordinates respectively.

2. The forward algorithms proposed are exact and fast without any oversampling radially or angularly, and it doesn't require interpolation in spatial or frequency domain or any approximation stage.

3. 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.

4. The proposed solution of polar and spherical FT solves the problem of inversion by iteratively operating on forward

and adjoint transforms, which are based on algebraically exact computations, and doing so rapidly and using preconditioned conjugate gradient solution to get accurate inversion.

5. We improve the conditioning of the inversion operation by using the proposed hybrid grid - Polar/Spherical Polar samples with Cartesian corners.

6. We exploit the special matrix structure of block Toeplitz form and provide a solution to do a faster inversion where the inversion matrix can be pre-computed.

Further research is required to study for improvements and provide efficient parallel computing implementation, and do a quantitative analysis taking in consideration different problem sizes, scalability, network constraints etc. Similar to the proposed fast and exact solution in 2-D and 3-D, a unified program is needed for higher dimensions.

ACKNOWLEDGMENT

We thank the anonymous reviewers whose comments and suggestions helped improved the presentation of this paper. The first author would like to thank Dr. Amir Averbuch. et.al (PPFFT) and Dr. Daniel Potts et.al (NUFFT) for making their codes available to the public, which was helpful for our own research, and in the spirit of reproducible research we will also make our code freely available to the public for academic purposes.

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] Averbuch, Amir, Ronald R. Coifman, David L. Donoho, Michael Elad, and Moshe Israeli,"Fast and accurate polar Fourier transform," *Applied and computational harmonic analysis*, 21, no. 2 (2006): 145-167.

[5] Fenn, Markus, Stefan Kunis, and Daniel Potts, "On the computation of the polar FFT," *Applied and Computational Harmonic Analysis*, 22, no. 2 (2007): 257-263.

[6] A new Polar Fourier-transform for computer-aided tomography and spotlight synthetic aperture radar. 1988

[7] Briggs, William L. The DFT: an owners' manual for the discrete Fourier transform. SIAM, 1995.

[8] Natterer, Frank, The mathematics of computerized tomography. Vol. 32. SIAM, 1986.

[9] Delaney, Alexander H., and Yoram Bresler. "A fast and accurate Fourier algorithm for iterative parallel-beam tomography." *Image Processing, IEEE Transactions*, on 5, no. 5 (1996): 740-753.

[10] Gottleib, D., Bertil Gustafsson, and Patrik Forssn. "On the direct Fourier method for computer tomography," *Medical Imaging, IEEE Transactions* , on 19, no. 3 (2000): 223-232.

[11] Basu, Samit, and Yoram Bresler, "$O(N^2 \log_2 N)$ filtered backprojection reconstruction algorithm for tomography," *Image Processing, IEEE Transactions*, on 9, no. 10 (2000): 1760-1773.

[12] Matej, Samuel, JeffreyA Fessler, and Ivan G. Kazantsev. "Iterative tomographic image reconstruction using Fourier-based forward and back-projectors," *Medical Imaging, IEEE Transactions*, on 23, no. 4 (2004): 401-412.

[13] A Averbuch, Coifman, R. R., 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] Averbuch, Amir, and Yoel Shkolnisky, "3D Fourier based discrete Radon transform," *Applied and Computational Harmonic Analysis*, 15, no. 1 (2003): 33-69.

[15] Averbuch, Amir, Ronald R. Coifman, David L. Donoho, Moshe Israeli, and Yoel Shkolnisky, "A framework for discrete integral transformations I-The pseudopolar Fourier transform," SIAM Journal on Scientific Computing 30, no. 2 (2008): 764-784.

[16] Averbuch, Amir, Ronald R. Coifman, David L. Donoho, Moshe Israeli, Yoel Shkolnisky, and Ilya Sedelnikov, "A framework for discrete integral transformations II-the 2D discrete Radon transform," SIAM Journal on Scientific Computing 30, no. 2 (2008): 785-803.

[17] Averbuch, Amir, Gil Shabat, and Yoel Shkolnisky,"Direct Inversion of the 3D Pseudo-polar Fourier Transform," arXiv preprint arXiv:1507.06174 (2015).

[18] Sayed Masoud Hashemi, Soosan Beheshti, Patrick R. Gill, Narinder S. Paul, and Richard S. C. Cobbold, Accelerated Compressed Sensing Based CT Image Reconstruction, Computational and Mathematical Methods in Medicine, vol. 2015, Article ID 161797, 16 pages, 2015. doi:10.1155/2015/161797

[19] Miao, Jianwei, Friedrich Frster, and Ofer Levi. "Equally sloped tomography with oversampling reconstruction." Physical Review B 72, no. 5 (2005): 052103.

[20] Beylkin, Gregory, "On the fast Fourier transform of functions with singularities," *Applied and Computational Harmonic Analysis* 2, no. 4 (1995): 363-381.

[21] Dutt, Alok, and Vladimir Rokhlin,"Fast Fourier transforms for nonequispaced data," SIAM Journal on Scientific computing 14, no. 6 (1993): 1368-1393.

[22] Dutt, Alok, and Vladimir Rokhlin, "Fast Fourier transforms for nonequispaced data, II," *Applied and Computational Harmonic Analysis* ,2, no. 1 (1995): 85-100.

[23] Potts, Daniel, Gabriele Steidl, and Manfred Tasche,"Fast Fourier transforms for nonequispaced data: A tutorial," In Modern sampling theory, pp. 247-270. Birkhuser Boston, 2001.

[24] Fessler, Jeffrey, and Bradley P. Sutton, "Nonuniform fast Fourier transforms using min-max interpolation," *Signal Processing, IEEE Transactions* on 51, no. 2 (2003): 560-574.

[25] Greengard, Leslie, and June-Yub Lee, "Accelerating the nonuniform fast Fourier transform," SIAM review 46, no. 3 (2004): 443-454.

[26] Jacob, Mathews, "Optimized least-square nonuniform fast Fourier transform." *Signal Processing, IEEE Transactions*, on 57, no. 6 (2009): 2165-2177.

[27] Reddy, B. Srinivasa, and Biswanath N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *IEEE transactions on image processing* 5, no. 8 (1996): 1266-1271.

[28] Pan, Wei, Kaihuai Qin, and Yao Chen, "An adaptable-multilayer fractional Fourier transform approach for image registration." *Pattern Analysis and Machine Intelligence*, IEEE Transactions on 31, no. 3 (2009): 400-414.

[29] Chou, Nigel, Joseph Izatt, and Sina Farsiu, "Generalized pseudo-polar Fourier grids and applications in registering ophthalmic optical coherence tomography images," In Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on, pp. 807-811. IEEE, 2009.

[30] S. Alam Abbas, H. Foroosh, "Image Registration with exact 2-D Polar and 3-D Spherical Polar discrete Fourier Transform," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Submitted.

[31] Bailey, David H., and Paul N. Swarztrauber,"The fractional Fourier transform and applications," SIAM review 33, no. 3 (1991): 389-404.

[32] J. Jan. Digital Signal Filtering, Analysis and Restoration (Telecommunications Series). INSPEC, Inc., 2000.

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

[34] Pavel, Karas, and Svoboda David. Algorithms for efficient computation of convolution. INTECH Open Access Publisher, 2013.

[35] Karas, Pavel, and David Svoboda, "Convolution of large 3D images on GPU and its decomposition," EURASIP journal on advances in signal processing 2011, no. 1 (2011): 1-12.

[36] Schaller, Peter, and Grigory Temnov, "Efficient and precise computation of convolutions: applying fft to heavy tailed distributions," Computational Methods in Applied Mathematics Comput. Methods Appl. Math. 8, no. 2 (2008): 187-200.

[37] A. Jerri, Integral and discrete transforms with applications and error analysis, CRC, 1992.

[38] Bowman, John C., and Malcolm Roberts, "Efficient dealiased convolutions without padding," SIAM Journal on Scientific Computing 33, no. 1 (2011): 386-406.

[39] Kunis, S., and D. Potts. "NFFT, Softwarepackage, C subroutine library." (2002).Retrieved from: https://github.com/NFFT/nfft.

[40] Lustig, Michael, Jacob Tsaig, Jin Hyung Lee, and David Donoho, "Fast spiral Fourier transform for iterative MR image reconstruction," In Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on, pp. 784-787. IEEE, 2004.

[41] Rencher, Alvin C., and G. Bruce Schaalje. Linear models in statistics. John Wiley Sons, 2008.

[42] Bass, Richard F., and Karlheinz Grchenig, “Random sampling of multivariate trigonometric polynomials.” SIAM journal on mathematical analysis 36, no. 3 (2005): 773-795.
[43] Knopp, Tobias, Stefan Kunis, and Daniel Potts. “A note on the iterative MRI reconstruction from nonuniform k-space data.” International journal of biomedical imaging (2007).
[44] Hansen, Per Christian, James G. Nagy, and Dianne P. O’leary. Deblurring images: matrices, spectra, and filtering. Vol. 3. SIAM, 2006.
[45] Dray, Tevian, and Corinne A. Manogue, “Conventions for spherical coordinates.” (2002).
[46] Ayala, Orlando, and Lian-Ping Wang,“Parallel implementation and scalability analysis of 3D fast Fourier transform using 2D domain decomposition.” Parallel Computing 39, no. 1 (2013): 58-77.
[47] Lichtenbelt, Barthold. Fourier volume rendering. Hewlett-Packard Laboratories, Technical Publications Department, 1995.
[48] Yalamanchili, P., Arshad, U., Mohammed, Z., Garigipati, P., Entschev, P., Kloppenborg, B., Malcolm, James and Melonakos, J. (2015). ArrayFire - A high performance software library for parallel computing with an easy-to-use API. Atlanta: AccelerEyes. Retrieved from https://github.com/arrayfire/arrayfire .
[49] Puschel, Markus, and Jose M. F. Moura, “Algebraic signal processing theory: Foundation and 1-D time.” *Signal Processing, IEEE Transactions* on 56.8 (2008): 3572-3585.