Fast and accurate computation of two-dimensional non-separable quadratic-phase integrals

Aykut Koç,[1,*] Haldun M. Ozaktas,[2] and Lambertus Hesselink[1]

[1]*Department of Electrical Engineering, Stanford University, Stanford, California 94305, USA*
[2]*Department of Electrical Engineering, Bilkent University, Bilkent, Ankara TR-06800, Turkey*
**Corresponding author: aykutkoc@stanford.edu*

Received February 17, 2010; accepted March 22, 2010;
posted April 6, 2010 (Doc. ID 124345); published May 12, 2010

We report a fast and accurate algorithm for numerical computation of two-dimensional non-separable linear canonical transforms (2D-NS-LCTs). Also known as quadratic-phase integrals, this class of integral transforms represents a broad class of optical systems including Fresnel propagation in free space, propagation in graded-index media, passage through thin lenses, and arbitrary concatenations of any number of these, including anamorphic/astigmatic/non-orthogonal cases. The general two-dimensional non-separable case poses several challenges which do not exist in the one-dimensional case and the separable two-dimensional case. The algorithm takes $\sim\tilde{N}\log\tilde{N}$ time, where $\tilde{N}$ is the two-dimensional space-bandwidth product of the signal. Our method properly tracks and controls the space-bandwidth products in two dimensions, in order to achieve information theoretically sufficient, but not wastefully redundant, sampling required for the reconstruction of the underlying continuous functions at any stage of the algorithm. Additionally, we provide an alternative definition of general 2D-NS-LCTs that shows its kernel explicitly in terms of its ten parameters, and relate these parameters bidirectionally to conventional *ABCD* matrix parameters. © 2010 Optical Society of America

OCIS codes: 070.2580, 350.6980, 070.2590.

1. INTRODUCTION

The class of two-dimensional non-separable linear canonical transforms (2D-NS-LCTs) is the class of linear integral transforms [1–3] that includes among its several special cases non-separable two-dimensional (2D) fractional Fourier transforms (FRTs) [4], two-dimensional chirp multiplication (2D-CM) and 2D chirp convolution operations, the two-dimensional Fourier transform (2D-FT), and generalized astigmatic scaling (magnification) operations, as well as their separable special cases. These transform integrals can represent a broad class of optical systems including Fresnel propagation in free space, propagation in graded-index media, passage through thin lenses, and arbitrary concatenations of any number of these. The class of non-separable transforms is significantly more general than two-dimensional separable linear canonical transforms (2D-S-LCTs) since it can represent a wide variety of anamorphic/astigmatic/non-orthogonal systems as well. The systems these integrals represent are also known as *ABCD* systems, which are also known as lossless first-order optical systems [5–12]. The classification of first-order optical systems and their representation through linear canonical transforms are studied in [13,9,14–16] for one-dimensional (1D) and 2D cases, respectively.

Linear canonical transforms (LCTs), which are commonly referred to as quadratic-phase integrals or quadratic-phase systems in optics [3], have also been referred to by different names such as generalized Huygens integrals [17], generalized Fresnel transforms [18,19], special affine Fourier transforms (FTs) [20,21], extended FRTs [22], and Moshinsky–Quesne transforms [8], among other names.

2D *separable* LCTs or symmetrical transforms that do not include the general non-separable case are addressed in [6,8,23–27]. The most special case possible is the isotropic two-dimensional linear canonical transforms (2D-LCTs) in which the system is fully symmetric, orthogonal, and the parameters for both of the dimensions are identical. This case can be represented by only three parameters as in a one-dimensional linear canonical transform (1D-LCT) [16]. When the system is still orthogonal but the parameters for the orthogonal dimensions differ, the system becomes a 2D-S-LCT, which is represented by six parameters [16]. This case is also termed as axially symmetric [15]. The separable 2D transforms do not pose much difficulty because the separable transform is essentially two independent 1D transforms along the two dimensions and the dimensions can be treated independently. However, the *non-separable* transform (2D-NS-LCT) is the most general case of this class of integrals where the two dimensions are coupled to each other by four additional cross-parameters, increasing the total number of parameters to ten. This general case is non-separable, non-axially symmetric, non-orthogonal, and anamorphic/astigmatic [11,15–17]. 2D-NS-LCTs are able to represent not only systems involving anamorphic/astigmatic components and reference surfaces, but also other interesting systems such as optical mode converters and resonators since they can represent the coupling between the dimensions [16,28,29]. Another prominent feature of 2D-NS-LCTs is their ability to represent systems

with rotations between any arbitrary planes in phase-space, like rotations and gyrations [15,16]. These systems are collected under the general name of gyrators and are useful in 2D image processing, signal processing, mode transformation, etc. [15,30–33]. As a result, the efficient and accurate digital computation of 2D-NS-LCTs is of importance in many areas of optics, optical signal processing, and general digital image processing.

Given an algorithm for efficiently computing 1D-LCTs [34–36], the efficient computation of separable 2D transforms is straightforward because the kernel can be separated and the 2D transform can be reduced to two successive 1D-LCTs. Much work has been done on 1D and 2D separable LCTs in terms of sampling issues and fast algorithms for their digital computation [37–40]. On the other hand, in the non-separable case, the two dimensions are coupled. Handling this case requires special attention and to the best of our knowledge has not been addressed before. The current established LCT computation algorithms [34–36] are not able to compute 2D-NS-LCTs.

An alternative representation of LCTs is presented and studied in [41]. This decomposition is based on the well-known Iwasawa decomposition [42]. In [41], the authors further decomposed the first matrix of the Iwasawa decomposition into a 2D separable FRT that is sandwiched between two coordinate rotators. We had earlier employed a 1D Iwasawa decomposition to develop a fast and efficient algorithm for 1D-LCTs [34,35]. In the present paper, we use the 2D version of this Iwasawa-type decomposition to derive our efficient algorithm. As in the 1D case, the distinguishing feature of our approach is the way our algorithm carefully addresses sampling and space-bandwidth product issues from an information-theoretical perspective. Special care is taken to ensure that the output samples represent the continuous transform in the Nyquist–Shannon sense during every stage of the algorithm so that the continuous transform can be fully recovered from the samples.

To our knowledge, there is no algorithm in the literature that efficiently calculates 2D-NS-LCTs. Despite the highly oscillatory nature of the integral kernel, we carefully manage the sampling rate so as to ensure that the number of samples used is sufficient, but not much larger than the space-bandwidth product of the input signal so that the algorithms are as efficient as possible. The straightforward method of sampling the input field and the kernel, and then calculating the output field, is not suitable for several reasons. First of all, due to the highly oscillatory nature of the integral kernel, a naive application of the Nyquist sampling theorem to determine the sampling rate would result in an excessively large number of samples and inefficient computation. On the other hand, ignoring the oscillations of the kernel and determining the sampling rate according to the input field alone may cause an under-representation of the output field in the Nyquist–Shannon sense. This unacceptable situation arises due to the fact that the particular 2D-LCT that we are calculating may increase the space-bandwidth product in one or both of the dimensions. If we do not increase the number of samples that we are working with so as to compensate for this increase, there will be information loss and we will not be able to recover the true transformed output from our computed samples. Thirdly, such a straightforward sampled integral computation takes $\tilde{N}^2$ time, where $\tilde{N}=MN$ for a 2D signal sampled on a $M\times N$ grid. In contrast, the complexity of our algorithm is $\sim\tilde{N}\log\tilde{N}$. This efficiency is even more crucial in the 2D case than in the 1D case since the number of points is much larger. By choosing the number of samples $\tilde{N}$ equal to the 2D space-bandwidth product of the signals, we ensure that the efficiency is near the best that is theoretically possible. More generally, through each stage of the algorithm, we carefully manage the sampling rate to maintain the information theoretically sufficient, but not wastefully redundant, sampling required for the reconstruction of the underlying continuous functions at any stage of the algorithm.

In Section 2, the definition of 2D-NS-LCTs is given. An explicit-kernel definition with the least possible number of independent variables is provided and the forward and backward relations between the parameters of this definition and the parameters of conventional **ABCD**-matrices are derived. Section 3 provides the preliminary mathematical background and the tools that we use in the algorithm. In Section 4, our algorithm is presented. Section 5 addresses the issue of the sampling rate and space-bandwidth product control in order to ensure the necessary sampling rates sufficient for the proper reconstruction in the Nyquist–Shannon sense at each step of the algorithm. Next, numerical results are reported in Section 6. We conclude in Section 7.

2. DEFINITION OF 2D-NS-LCTs

The 2D-NS-LCT with parameter matrix $\mathbf{M}$, of an input function $f(\mathbf{u})$, can be denoted and defined as [41,43]

$$g(\mathbf{u}) = f_{\mathbf{M}}(\mathbf{u}) = (\mathcal{C}_{\mathbf{M}}f)(\mathbf{u}) = \frac{1}{\sqrt{\det i\mathbf{B}}}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\exp[i\pi(\mathbf{u}'^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{A}\mathbf{u}' - 2\mathbf{u}'^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{u} + \mathbf{u}^{\mathrm{T}}\mathbf{D}\mathbf{B}^{-1}\mathbf{u})]f(\mathbf{u}')d\mathbf{u}', \tag{1}$$

where $\mathbf{u}=[u_x u_y]^{\mathrm{T}}$, $\mathbf{u}'=[u_x' u_y']^{\mathrm{T}}$, with T denoting the transpose operation. $\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}$ are 2×2 submatrices defining the transformation matrix $\mathbf{M}$ of the system that represents the 2D-LCT, with $\mathbf{B}$ being non-singular. The matrix $\mathbf{M}$, which is given as

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}, \tag{2}$$

is real and symplectic so that the following hold ($\mathbf{I}$ stands for the 2×2 identity matrix) [14,41]:

$$\mathbf{A}\mathbf{B}^{\mathrm{T}} = \mathbf{B}\mathbf{A}^{\mathrm{T}}, \quad \mathbf{C}\mathbf{D}^{\mathrm{T}} = \mathbf{D}\mathbf{C}^{\mathrm{T}}, \quad \mathbf{A}\mathbf{D}^{\mathrm{T}} - \mathbf{B}\mathbf{C}^{\mathrm{T}} = \mathbf{I},$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{C} = \mathbf{C}^{\mathrm{T}}\mathbf{A}, \quad \mathbf{B}^{\mathrm{T}}\mathbf{D} = \mathbf{D}^{\mathrm{T}}\mathbf{B}, \quad \mathbf{A}^{\mathrm{T}}\mathbf{D} - \mathbf{C}^{\mathrm{T}}\mathbf{B} = \mathbf{I}. \tag{3}$$

From a group-theoretical point of view, 2D-NS-LCTs form the ten-parameter symplectic group $Sp(4,\mathbb{R})$. ($\mathbf{M}$ has 16 parameters with six constraints leaving ten independent parameters.) More on group-theoretical properties of LCTs can be found in [8,26].

We will write the integral relationship between the input function $f(u_x,u_y)$ and the output function $g(u_x,u_y)$ more explicitly as

$$g(u_x,u_y) = e^{-i\pi/2}\sqrt{\beta_x\beta_y - \eta_x\eta_y} \times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K(u_x,u_y,u_x',u_y')f(u_x',u_y')du_x'du_y',$$

$$K(u_x,u_y,u_x',u_y') = \exp[i\pi\alpha_x u_x^2 - 2\beta_x u_x u_x' + 2\eta_x u_x u_y' + \eta_\alpha u_x u_y + \gamma_x u_x'^2 + \alpha_y u_y^2 - 2\beta_y u_y u_y' + 2\eta_y u_x' u_y + \eta_\gamma u_x' u_y' + \gamma_y u_y'^2], \quad (4)$$

where α_x, β_x, γ_x, α_y, β_y, γ_y, η_α, η_x, η_y, η_γ are the ten independent parameters defining the 2D-NS-LCT (we will refer to them as the "scalar parameters"). These parameters also uniquely define the LCT. We will use this set of parameters for two reasons. First, although the definition using matrices gives us a compact and streamlined representation, the kernel and coefficients are not seen easily and explicitly in this case. When one needs to restrict the parameters to obtain the kernel of any desired particular subclass of 2D-LCTs, it is not easy to derive the elements of the *ABCD* matrices directly, whereas this is straightforward with Eq. (4). Secondly, and more importantly, when the *ABCD* submatrices are used directly, we need to manipulate 16 parameters (four 2×2 matrices with four elements each), despite the fact that only ten of them are independent. However, with the explicit definition we use the least number of required parameters, namely, ten, and match the corresponding ten-parameter symplectic group with exactly these ten parameters.

It is easy to convert from one set of parameters to the other. The ten scalar parameters are given in terms of the elements of **A**, **B**, **D** as follows (only three of the submatrices are independent):

$$\alpha_x = \frac{D_{11}B_{22} - D_{12}B_{21}}{\det \mathbf{B}}, \quad (5)$$

$$\beta_x = \frac{B_{22}}{\det \mathbf{B}}, \quad (6)$$

$$\eta_x = \frac{B_{21}}{\det \mathbf{B}}, \quad (7)$$

$$\eta_\alpha = \frac{D_{12}B_{11} + D_{21}B_{22} - D_{11}B_{12} - D_{22}B_{21}}{\det \mathbf{B}}, \quad (8)$$

$$\gamma_x = \frac{B_{22}A_{11} - B_{12}A_{21}}{\det \mathbf{B}}, \quad (9)$$

$$\alpha_y = \frac{D_{22}B_{11} - D_{21}B_{12}}{\det \mathbf{B}}, \quad (10)$$

$$\beta_y = \frac{B_{11}}{\det \mathbf{B}}, \quad (11)$$

$$\eta_y = \frac{B_{12}}{\det \mathbf{B}}, \quad (12)$$

$$\eta_\gamma = \frac{A_{21}B_{11} + A_{12}B_{22} - A_{11}B_{21} - B_{12}A_{22}}{\det \mathbf{B}}, \quad (13)$$

$$\gamma_y = \frac{B_{11}A_{22} - A_{12}B_{21}}{\det \mathbf{B}}. \quad (14)$$

If we wish to obtain the submatrices **A**, **B**, **D** in terms of the scalar parameters, we can use the following reverse formulas:

$$\mathbf{A} = \frac{1}{2(\beta_x\beta_y - \eta_x\eta_y)}\begin{bmatrix} \eta_y\eta_\gamma + 2\beta_y\gamma_x & \eta_\gamma\beta_y + 2\eta_y\gamma_y \\ \eta_\gamma\beta_x + 2\eta_x\gamma_x & \eta_x\eta_\gamma + 2\beta_x\gamma_y \end{bmatrix}, \quad (15)$$

$$\mathbf{B} = \frac{1}{\beta_x\beta_y - \eta_x\eta_y}\begin{bmatrix} \beta_y & \eta_y \\ \eta_x & \beta_x \end{bmatrix}, \quad (16)$$

$$\mathbf{D} = \frac{1}{2(\beta_x\beta_y - \eta_x\eta_y)}\begin{bmatrix} \eta_x\eta_\alpha + 2\beta_y\alpha_x & \eta_\alpha\beta_x + 2\eta_y\alpha_x \\ \eta_\alpha\beta_y + 2\eta_x\alpha_y & \eta_y\eta_\alpha + 2\beta_x\alpha_y \end{bmatrix}. \quad (17)$$

As noted earlier, the submatrix **C** is not independent and can be expressed in terms of **A**, **B**, **D** as follows:

$$C_{11} = (A_{11}D_{11}B_{22} + A_{12}D_{12}B_{22} - B_{22} - B_{12}A_{21}D_{11} - B_{12}A_{22}D_{12})/\det \mathbf{B},$$

$$C_{21} = (A_{12}D_{22} + A_{11}D_{21} - B_{12}C_{22})/B_{11},$$

$$C_{12} = (A_{21}D_{11} + A_{22}D_{12} - B_{21}C_{11})/B_{22},$$

$$C_{22} = (A_{22}D_{22}B_{11} + A_{21}D_{21}B_{11} - B_{11} - B_{21}A_{12}D_{22} - B_{21}A_{11}D_{21})/\det \mathbf{B}. \quad (18)$$

Equations (18), along with the corresponding entries in Eqs. (15)–(17), define **C** in terms of the scalar parameters. (Because the final expressions for **C** are cumbersome we do not write them here explicitly.) Note that when we set the "cross" parameters η_α, η_x, η_y, η_γ to zero, the generalized 2D non-separable transformation matrix **M** will reduce to the transformation matrix of the 2D separable case studied in [23]. Also note that **A**, **B**, **C**, and **D** as given in Eqs. (15)–(18) satisfy the required properties given in Eq. (3).

3. PRELIMINARIES

A. Wigner Distributions

We will review the relationship between LCTs and the Wigner distribution (WD), which will aid us in understanding the effects of the elementary blocks used in our decompositions. The WD $W_f(u_x,u_y,\mu_x,\mu_y)$ of a 2D signal $f(u_x,u_y)$ can be defined as follows [3]:

$$W_f(u_x,u_y,\mu_x,\mu_y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u_x+u_x'/2,u_y+u_y'/2)f^*(u_x - u_x'/2,u_y-u_y'/2)e^{-2\pi i(\mu_x u_x'+\mu_y u_y')}du_x'du_y'. \tag{19}$$

Roughly speaking, $W(u_x,u_y,\mu_x,\mu_y)$ is a function that gives the distribution of the signal energy over both space variables and their corresponding frequency variables. We call this four-dimensional (4D) WD the "4D Wigner distribution," whereas the usual 2D WD used for 1D functions will be referred to as the "2D Wigner distribution." Let f denote a function and $f_{\mathbf{M}}$ be its 2D-LCT with parameter matrix $\mathbf{M}$. Then, the relation between the WD of $f_{\mathbf{M}}$ and the WD of f can be expressed as [3]

$$W_{f_{\mathbf{M}}}(\mathbf{Ms})=W_f(\mathbf{s}), \tag{20}$$

where the vector $\mathbf{s}=[u_x\ \ u_y\ \ \mu_x\ \ \mu_y]^{\mathrm{T}}$ is used for the sake of notational simplicity. An example of the use of the WD in sampling issues from another perspective can be found in [44].

B. Fractional Fourier Transformation

The FRT plays an important role in our algorithm. Therefore, here we briefly give its definition. The ath order 1D FRT [26,45–51] of a function $f(u)$, denoted $f_a(u)$, can be defined as

$$\mathcal{F}^a f(u)=f_a(u)=\int_{-\infty}^{\infty}K_a(u,u')f(u')du',$$

$$K_a(u,u')=A_\theta\exp[i\pi(\cot\theta\,u^2-2\csc\theta\,uu'+\cot\theta\,u'^2)],$$

$$A_\theta=\sqrt{1-i\cot\theta},\quad \theta=\frac{a\pi}{2} \tag{21}$$

when $a\neq 2j$, $K_a(u,u')=\delta(u-u')$ when $a=4j$, and $K_a(u,u')=\delta(u+u')$ when $a=4j\pm 2$, where j is an integer. The square root is defined such that the argument of the result lies in the interval $(-\pi/2,\pi/2]$. For $0<|a|<2$ ($0<|\theta|<\pi$), A_θ can be rewritten without ambiguity as

$$A_\theta=\frac{e^{-i[\pi\,\mathrm{sgn}(\theta)/4-\theta/2]}}{\sqrt{|\sin\theta|}}, \tag{22}$$

where sgn( ) is the sign function. When a is outside the interval $0\leq|a|\leq 2$, we simply need to replace a with its modulo 4 equivalence lying in this interval and use this value in Eq. (22).

C. A 3-Sphere for Space-Bandwidth Control for 2D Functions and Dimensional Normalization

When we study 1D input functions and 1D-LCTs, the corresponding WD is 2D. One dimension represents the space extent and the other represents the spatial-frequency extent of the signal. However, for 2D signals, there exist two space extents and two corresponding spatial-frequency extents resulting in a 4D Wigner distribution. In [34,35] we used 2D Wigner distributions for tracking and control of the space-bandwidth products of signals through the stages of our algorithms. 2D Wigner distributions are easy to visualize and therefore easy to understand. However, for 2D signals we cannot graphically show the WD because it is a 4D function. Therefore we will develop and use a more abstract and rigorous approach to space-bandwidth tracking and control in order to achieve the information-theoretical minimum sampling rate for the lossless reconstruction of the continuous output function from the output samples.

First, we need to recall the geometrical object known as a "3-sphere." In general, for a natural number n, an "n-sphere" is the generalization of the ordinary "2-sphere" in common three-dimensional Euclidian space to any dimension. Explicitly, an n-sphere, denoted as S^n and centered at the origin, is the analog of a sphere in $(n+1)$-dimensional Euclidian space and is defined as

$$S^n=\{x\in\mathbb{R}^{n+1}:\|x\|=r\}, \tag{23}$$

where the positive real number r is the radius of the n-sphere and $\mathbb{R}^{n+1}$ is an $(n+1)$-dimensional vector space over $\mathbb{R}$. More on n-spheres can be found in [52,53]. Then, we can provide the generic definition of the 3-sphere, S^3, centered at the origin explicitly as

$$S^3=\{(x_1,x_2,x_3,x_4)\in\mathbb{R}^4:x_1^2+x_2^2+x_3^2+x_4^2=r^2\}. \tag{24}$$

Now, let us turn our attention to our 2D input functions. It is well known that a non-zero function and its FT cannot both be confined to finite regions. However, in practice, we always work with samples of finite extent functions by assuming that the energy of the signal falling outside of some region is negligible. In general, the signal will exhibit some distribution of energy in the 2D space-frequency hypervolume (which is four dimensional). We will assume that a finite hyperellipsoidal boundary in $\mathbb{R}^4$ is chosen so as to confine most of the energy of the signal. This hyperellipsoidal boundary will imply finite extents in the two space dimensions and the two spatial-frequency dimensions. The intervals of the confinement thus defined will be denoted by $[-\Delta S_x/2,\Delta S_x/2]$ and $[-\Delta S_y/2,\Delta S_y/2]$ in the space dimensions, and $[-\Delta B_x/2,\Delta B_x/2]$ and $[-\Delta B_y/2,\Delta B_y/2]$ in the spatial-frequency dimensions. The space and spatial-frequency representations of the signal will be approximately confined within these intervals. Given these, it also follows that both space-domain extents are confined within the worst-case interval $[-\Delta S_{\max}/2,\Delta S_{\max}/2]$, where $\Delta S_{\max}=\max\{\Delta S_x,\Delta S_y\}$, and both frequency-domain extents are confined within the interval $[-\Delta B_{\max}/2,\Delta B_{\max}/2]$, where $\Delta B_{\max}=\max\{\Delta B_x,\Delta B_y\}$. Under these conditions, the WD of the function is confined within the boundary $\mathbf{O}$ in $\mathbb{R}^4$ (note that this is not defining a 3-sphere yet),

$$\mathbf{O}=\left\{(s_x,s_y,b_x,b_y)\in\mathbb{R}^4:\frac{s_x^2}{(\Delta S_{\max}/2)^2}+\frac{b_x^2}{(\Delta B_{\max}/2)^2}+\frac{s_y^2}{(\Delta S_{\max}/2)^2}+\frac{b_y^2}{(\Delta B_{\max}/2)^2}=1\right\}, \tag{25}$$

where s_x and s_y are temporary space variables and b_x and b_y are temporary spatial-frequency variables of the WD of the signal.

Let us now introduce the scaling parameter P and scaled dimensionless coordinates

$$u_x = s_x/P,$$

$$u_y = s_y/P,$$

$$\mu_x = b_x P,$$

$$\mu_y = b_y P, \tag{26}$$

such that the two space-domain and two frequency-domain representations are confined to intervals of length $\Delta S_{\max}/P$ and $\Delta B_{\max} P$, respectively. Let $P = \sqrt{\Delta S_{\max}/\Delta B_{\max}}$ so that the lengths of all the four intervals become equal to the dimensionless quantity $\sqrt{\Delta S_{\max}\Delta B_{\max}}$, which we denote by Δu. Expressed in dimensionless coordinates, the boundary **O** defined in Eq. (25) reduces to the desired 3-sphere, denoted by $\mathbf{O_{sp}}$,

$$\mathbf{O_{sp}} = \left\{ (u_x, u_y, \mu_x, \mu_y) \in \mathbb{R}^4 : u_x^2 + u_y^2 + \mu_x^2 + \mu_y^2 = \left(\frac{\sqrt{\Delta S_{\max}\Delta B_{\max}}}{2} \right)^2 = \left(\frac{\Delta u}{2} \right)^2 \right\}. \tag{27}$$

To summarize, after the dimensional normalization procedure given above has been performed, the 4D Wigner distribution of our 2D input function can be assumed to be confined within a 3-sphere $\mathbf{O_{sp}}$ of diameter Δu.

4. ALGORITHM

As noted before, one of the most important features of our method is to control the sampling rate of the function with the goal of having enough samples to be able to reconstruct the continuous function without information loss, and at the same time without needlessly increasing the number of samples to maintain the efficiency. In this section, we present our algorithm, discuss the stages in the decomposition, and derive the parameters of each stage from the parameters of the 2D-NS-LCT that is being computed. The effects of each stage of the decomposition on the WD of our function (thus on the space-bandwidth products) and associated sampling rate issues will be addressed in Section 5.

The Iwasawa decomposition is the core of our algorithm. After the dimensional normalization explained in Subsection 3.C, any transformation matrix **M** can be written in the following Iwasawa form [42,41]:

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{G} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ -\mathbf{Y} & \mathbf{X} \end{bmatrix}, \tag{28}$$

where

$$\mathbf{G} = -(\mathbf{C}\mathbf{A}^{\mathrm{T}} + \mathbf{D}\mathbf{B}^{\mathrm{T}})(\mathbf{A}\mathbf{A}^{\mathrm{T}} + \mathbf{B}\mathbf{B}^{\mathrm{T}})^{-1}, \tag{29}$$

$$\mathbf{S} = (\mathbf{A}\mathbf{A}^{\mathrm{T}} + \mathbf{B}\mathbf{B}^{\mathrm{T}})^{1/2}, \tag{30}$$

$$\mathbf{X} = (\mathbf{A}\mathbf{A}^{\mathrm{T}} + \mathbf{B}\mathbf{B}^{\mathrm{T}})^{-1/2}\mathbf{A}, \tag{31}$$

$$\mathbf{Y} = (\mathbf{A}\mathbf{A}^{\mathrm{T}} + \mathbf{B}\mathbf{B}^{\mathrm{T}})^{-1/2}\mathbf{B}. \tag{32}$$

Given the 4×4 matrix **M**, we can determine the 2×2 matrices **G**, **S**, **X**, **Y** by using Eqs. (29)–(32). If we are able to develop a fast algorithm to compute the three stages in $\sim\tilde{N}\log\tilde{N}$ time, the overall transform can also be calculated in $\sim\tilde{N}\log\tilde{N}$ time. In this decomposition, the first operation is an orthosymplectic system, followed by a scaling (magnification) system, and finally followed by a 2D-CM. (Note that each of the stages of the algorithm is a special case of 2D-NS-LCTs.)

We begin with the first and the most sophisticated stage of the decomposition, the orthosymplectic system. This stage of the decomposition can be further decomposed into a two-dimensional *separable* fractional Fourier transform (2D-S-FRT) that is sandwiched between two coordinate rotators [41],

$$\begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ -\mathbf{Y} & \mathbf{X} \end{bmatrix} = \mathbf{R}_{\mathbf{r_2}} \mathbf{F}_{\mathbf{a_x},\mathbf{a_y}} \mathbf{R}_{\mathbf{r_1}}, \tag{33}$$

where the 4×4 matrices $\mathbf{R}_{\mathbf{r_1}}$, $\mathbf{F}_{\mathbf{a_x},\mathbf{a_y}}$, $\mathbf{R}_{\mathbf{r_1}}$ are defined as

$$\mathbf{R}_{\mathbf{r_1}} = \begin{bmatrix} \cos(r_1) & \sin(r_1) & 0 & 0 \\ -\sin(r_1) & \cos(r_1) & 0 & 0 \\ 0 & 0 & \cos(r_1) & \sin(r_1) \\ 0 & 0 & -\sin(r_1) & \cos(r_1) \end{bmatrix}, \tag{34}$$

$$\mathbf{R}_{\mathbf{r_2}} = \begin{bmatrix} \cos(r_2) & \sin(r_2) & 0 & 0 \\ -\sin(r_2) & \cos(r_2) & 0 & 0 \\ 0 & 0 & \cos(r_2) & \sin(r_2) \\ 0 & 0 & -\sin(r_2) & \cos(r_2) \end{bmatrix}, \tag{35}$$

$$\mathbf{F}_{\mathbf{a_x},\mathbf{a_y}} = \begin{bmatrix} \cos(a_x\pi/2) & 0 & \sin(a_x\pi/2) & 0 \\ 0 & \cos(a_y\pi/2) & 0 & \sin(a_y\pi/2) \\ -\sin(a_x\pi/2) & 0 & \cos(a_x\pi/2) & 0 \\ 0 & -\sin(a_y\pi/2) & 0 & \cos(a_y\pi/2) \end{bmatrix}, \tag{36}$$

where $\mathbf{R}_{\mathbf{r_1}}$ and $\mathbf{R}_{\mathbf{r_2}}$ are rotation matrices that impose rotations of angles r_1 and r_2, respectively, through the spatial variables (u_x, u_y) and through their frequency variables (μ_x, μ_y). Unlike these traditional rotators which rotate within space and spatial-frequency separately, the FRT rotates within the space-frequency planes of each dimension. $\mathbf{F}_{\mathbf{a_x},\mathbf{a_y}}$ stands for a 2D-S-FRT that makes separable rotations of angle $a_x\pi/2$ over the variables (u_x, μ_x) and of angle $a_y\pi/2$ over the variables (u_y, μ_y). Since this 2D FRT operation is separable, it corresponds to two 1D FRT operations performed over each of the dimensions. Explicitly, this means first performing 1D-FRTs with the fractional order a_x for each of the rows (or columns) and then performing 1D-FRTs with the fractional order a_y for each of the columns (or rows) of the sampling grid. It is this observation that enables us to implement this stage of the decomposition efficiently in $O(\tilde{N}\log\tilde{N})$ time. There are fast and established algorithms to compute 1D-FRTs

[34,35,54,55] so that this stage can be calculated in $O(\tilde{N}\log\tilde{N})$ time easily.

The interpretation of the coordinate rotators requires care. When we are working with sampled functions, we know the value and coordinates (the location where the particular sample is taken) of all the samples we have. A coordinate rotation can be interpreted in this situation as a rotation of the locations of the samples, resulting in a new sampling grid, rather than a change in the sample values. If we assume that we start with a regular rectangular grid, after the coordinate rotation, the grid would no longer coincide with the original grid unless the rotation is an integer multiple of $\pi/2$. Unfortunately, in order to perform FRT operations along the horizontal and vertical directions, we need the samples to be on a regular rectangular grid in order to employ available fast algorithms. Therefore, we must carry out an interpolation operation to determine the values of the function on a regular rectangular grid. There are several techniques and algorithms to perform this interpolation efficiently. We have chosen to use in our numerical simulations fast and standard implementations of nearest neighbor, bilinear, and cubic interpolations [56,57], but any other efficient method may also be used. This interpolation step and its performance can be a major source of error in our algorithm, as we will further discuss later.

We now turn our attention to determining the coordinate rotation angles r_1 and r_2, and the FRT fractional orders a_x and a_y. When we plug Eqs. (34)–(36) in Eq. (33), carry out the matrix multiplications, and equate the entries of both sides of Eq. (33), we get the following equations in the four unknowns r_1, r_2, a_x, and a_y:

$$\begin{aligned}
X_{11} &= \cos r_1 \cos r_2 \cos(a_x\pi/2) - \sin r_1 \sin r_2 \cos(a_y\pi/2),\\
X_{12} &= \sin r_1 \cos r_2 \cos(a_x\pi/2) + \cos r_1 \sin r_2 \cos(a_y\pi/2),\\
X_{21} &= -\cos r_1 \sin r_2 \cos(a_x\pi/2) - \sin r_1 \cos r_2 \cos(a_y\pi/2),\\
X_{22} &= -\sin r_1 \sin r_2 \cos(a_x\pi/2) + \cos r_1 \cos r_2 \cos(a_y\pi/2),\\
Y_{11} &= \cos r_1 \cos r_2 \sin(a_x\pi/2) - \sin r_1 \sin r_2 \sin(a_y\pi/2),\\
Y_{12} &= \sin r_1 \cos r_2 \sin(a_x\pi/2) + \cos r_1 \sin r_2 \sin(a_y\pi/2),\\
Y_{21} &= -\cos r_1 \sin r_2 \sin(a_x\pi/2) - \sin r_1 \cos r_2 \sin(a_y\pi/2),\\
Y_{22} &= -\sin r_1 \sin r_2 \sin(a_x\pi/2) + \cos r_1 \cos r_2 \sin(a_y\pi/2).
\end{aligned} \tag{37}$$

These equations are sufficient to solve for and unambiguously determine the rotation and fractional Fourier angles of the decomposition in a straightforward manner, provided one pays proper attention to sign considerations when inverting the trigonometric functions.

To summarize, the first stage of our algorithm involves determining the angles from the above equations, performing the first coordinate rotation, following this by two 1D-FRTs over each of the dimensions, and then finishing with the second coordinate rotation. All these steps can be calculated in $O(\tilde{N}\log\tilde{N})$ time.

The second stage is the scaling operation and it seems to be the simplest of the three stages. It is not, however, as trivial as in the 1D case [35]. In one dimension, it corresponds to only a reinterpretation of the spacing between the samples. The sampling interval scales with the scaling parameter. Intuitively, it squeezes in or stretches out the total number of samples as the word scaling implies. This means that there is no change in the total number of samples and thus no need to oversample the input samples. The analog of the 1D scalar scaling parameter in the 2D case is the matrix $\mathbf{S}$. When $\mathbf{S}$ is diagonal, which means that there is no coupling between the two dimensions of the function for scaling purposes, the scaling is separable. Due to this separability, this situation does not impose an increase in the space-bandwidth products and thus does not require oversampling, just as in the 1D case. But when the off-diagonal elements of $\mathbf{S}$ are non-zero, the scaling operation is no longer so trivial. Although the total number of degrees of freedom of the signal remains the same, the space-bandwidth products may increase and an oversampling to match this increase may be necessary. Readers wishing to better understand how the space-bandwidth product may increase despite the fact that the number of degrees of freedom remains the same are referred to [35], where these issues are studied graphically for the 1D case. An analogous, although not visually demonstrable, situation exists for 2D signals. The sampling rate control mechanism for such 2D scaling operations will be developed in detail in Section 5. At this point, we note that in those cases where the number of samples needs to be increased, the oversampling should be performed first, prior to the scaling. Afterward, the scaling is achieved by the mere reinterpretation of the locations of the samples without changing the samples themselves (other than a constant multiplicative factor). Computationally, such a scaling operation amounts to modifying the information that tells us which coordinates the samples belong to. Since it requires only the reinterpretation of the coordinates of the samples plus a possible oversampling, it does not impose much computational load. Equation (30) gives us the scaling parameters. The matrix $\mathbf{S}$ can be easily used to determine the output samples by using the input-output relation of the scaling operation,

$$f_{sc}(\mathbf{u}) = \frac{1}{\sqrt{\det \mathbf{S}}} f(\mathbf{S}^{-1}\mathbf{u}), \tag{38}$$

where f is the function to be scaled, f_{sc} is the scaled function, and $\mathbf{u}=[u_x \;\; u_y]^{\mathrm{T}}$.

The last stage of our main Iwasawa decomposition is the 2D-CM operation whose parameters are given by the matrix $\mathbf{G}$ as defined in Eq. (29). The input-output relation of this 2D-CM is given as

$$f_{ch}(\mathbf{u}) = e^{-i\pi(G_{11}x^2+(G_{12}+G_{21})xy+G_{22}y^2)} f(\mathbf{u}), \tag{39}$$

where f_{ch} stands for the chirp-multiplied function. The 2D-CM operation is the stage that is mainly burdened with any shears inherent in the 2D-NS-LCT to be computed. Such shears may considerably increase the space-bandwidth products of the function. Thus, before the 2D-CM operation, the space-bandwidth products of the

function should be calculated carefully and any necessary oversampling should be performed. This CM operation may turn out to be non-separable or separable for particular 2D-NS-LCTs but, regardless, it requires only one multiplication for each sample, resulting in $O(\tilde{N})$ time computation. As a result, we see that we can perform all of the stages of the decomposition in $O(\tilde{N}\log\tilde{N})$ time or faster, which makes the computational complexity (cost) of our overall algorithm $O(\tilde{N}\log\tilde{N})$.

5. SPACE-BANDWIDTH AND SAMPLING RATE CONTROL

In this section, we develop a method to track the space-bandwidth products of our functions as we perform the consecutive operations in our decomposition and focus on how to control the number of samples efficiently. We need to calculate the necessary sampling intervals and sampling rates for both dimensions that are necessary to represent the continuous signal without information loss for each of the stages. Oversampling should be undertaken prior to any stage that increases either of the space-bandwidth products.

As given in Subsection 3.C, the WD of the input signal is assumed to be confined within a 3-sphere with radius $\Delta u/2$, which also means that the signal is assumed to be almost space- and band-limited in both dimensions. The 4D Wigner representation gives us two space extents, two spatial-frequency extents and two space-bandwidth products, one for each dimension of the function. Let us denote the space-bandwidth product along the u_x direction by N_x and that along the u_y direction by N_y. These extents define the minimum required number of samples along the corresponding direction, with the total number of samples being $N_x\times N_y$. Since the WD is confined within a 3-sphere of diameter Δu, all the extents of the function (space and spatial-frequency) are equal to Δu at the beginning. Thus, the function should be sampled on a $N_x\times N_y$ grid, where the u_x-coordinate of the function spans the interval $u_x=(-\Delta u/2,\Delta u/2)$ and the u_y-coordinate spans the interval $u_y=(-\Delta u/2,\Delta u/2)$. The distance between two adjacent samples is equal to Δu^{-1} along both dimensions. As a result, the space-bandwidth products are initially $N_x=N_y=\Delta u^2$.

We will track the effects of each stage in our algorithm to the WD boundary to which the original function is confined, and calculate the extents and the two space-bandwidth products and eventually the required structure of the sampling grid before each stage is performed. We will address each of the three stages (the first with three steps) given in Section 4 in sequence.

To start, we first write down the 3-sphere boundary in hyperspherical coordinates. The 3-sphere $\mathbf{O_{sp}}$ given in Eq. (27) can be transformed to the equivalent hyperspherical coordinates with the following coordinate transformation:

$$\mathbf{O_{sp}}=\begin{bmatrix}u_x\\u_y\\\mu_x\\\mu_y\end{bmatrix}=\frac{\Delta u}{2}\begin{bmatrix}\cos\phi_1\\\sin\phi_1\cos\phi_2\\\sin\phi_1\sin\phi_2\cos\phi_3\\\sin\phi_1\sin\phi_2\sin\phi_3\end{bmatrix}, \tag{40}$$

where the angular hyperspherical coordinates ϕ_1 and ϕ_2 range over $[0,\pi]$, and the angular hyperspherical coordinate ϕ_3 ranges over $[0,2\pi]$. (Note that this coordinate system transformation is not unique.) The sum of the squares of the elements of the vector on the right-hand side of Eq. (40) again equals $(\Delta u/2)^2$ as expected. Equation (20) allows us to calculate the new boundary $\mathbf{s}_{\text{out}}$ of the WD after any operation from the boundary $\mathbf{s}_{\text{in}}$ before the operation. Just as the old boundary confined most of the energy of the signal represented by the WD, so does the new boundary. This is because the mapping in Eq. (20) merely maps values of the WD to new space-frequency points, and values which were confined within the old boundary remain confined within the new boundary.

A. First Coordinate Rotator

At the very beginning of the algorithm, we start with the boundary vector $\mathbf{s}_{\text{in}}=\mathbf{O_{sp}}$. In other words, the input boundary vector $\mathbf{s}_{\text{in}}$ before the first coordinate rotator is given by Eq. (40). Then $\mathbf{s}_{\text{out}}$ is found by multiplying $\mathbf{s}_{\text{in}}$ with the transformation matrix of the coordinate rotator as

$$\begin{aligned}\mathbf{s}_{\text{out}}=\mathbf{R}_{\mathbf{r}_1}\mathbf{s}_{\text{in}}&=\frac{\Delta u}{2}\begin{bmatrix}\cos(r_1)&\sin(r_1)&0&0\\-\sin(r_1)&\cos(r_1)&0&0\\0&0&\cos(r_1)&\sin(r_1)\\0&0&-\sin(r_1)&\cos(r_1)\end{bmatrix}\begin{bmatrix}\cos\phi_1\\\sin\phi_1\cos\phi_2\\\sin\phi_1\sin\phi_2\cos\phi_3\\\sin\phi_1\sin\phi_2\sin\phi_3\end{bmatrix}\\&=\frac{\Delta u}{2}\begin{bmatrix}\cos(r_1)\cos\phi_1+\sin(r_1)\sin\phi_1\cos\phi_2\\-\sin(r_1)\cos\phi_1+\cos(r_1)\sin\phi_1\cos\phi_2\\\cos(r_1)\sin\phi_1\sin\phi_2\cos\phi_3+\sin(r_1)\sin\phi_1\sin\phi_2\sin\phi_3\\-\sin(r_1)\sin\phi_1\sin\phi_2\cos\phi_3+\cos(r_1)\sin\phi_1\sin\phi_2\sin\phi_3\end{bmatrix},\end{aligned} \tag{41}$$

with ϕ_1 and ϕ_2 ranging over $[0,\pi]$, ϕ_3 ranging over $[0,2\pi]$, and $\mathbf{s}_{\text{out}}$ represents the boundary of the output WD. We can show that this boundary remains a 3-sphere of radius $\Delta u/2$ by writing

$$
\begin{aligned}
&u_x^2 + u_y^2 + \mu_x^2 + \mu_y^2 \\
&\quad = \left(\frac{\Delta u}{2}\right)^2 [(\cos(r_1)\cos\ \phi_1 + \sin(r_1)\sin\ \phi_1\ \cos\ \phi_2)^2 \\
&\quad + (-\sin(r_1)\cos\ \phi_1 + \cos(r_1)\sin\ \phi_1\ \cos\ \phi_2)^2 \\
&\quad + (\cos(r_1)\sin\ \phi_1\ \sin\ \phi_2\ \cos\ \phi_3 \\
&\quad + \sin(r_1)\sin\ \phi_1\ \sin\ \phi_2\ \sin\ \phi_3)^2 \\
&\quad + (-\sin(r_1)\sin\ \phi_1\ \sin\ \phi_2\ \cos\ \phi_3 \\
&\quad + \cos(r_1)\sin\ \phi_1\ \sin\ \phi_2\ \sin\ \phi_3)^2] = \left(\frac{\Delta u}{2}\right)^2, \qquad (42)
\end{aligned}
$$

as can be verified after some algebra with trigonometric functions. This result means that the coordinate rotation operation does not change the 3-sphere nature of the confining boundary of the WD, and since rotating an n-sphere (just like an ordinary sphere) does not change its extent along any direction, it does not have any effect on the space-bandwidth products. Therefore, no matter what the angles are, the coordinate rotation operation does not require a change in the number of samples and the sampling grid.

B. 2D-S-FRT

Since the previous rotation operation left the WD confined within the original 3-sphere, and since we are interested only in the worst-case boundary, $\mathbf{s}_{\text{in}}$ ($\mathbf{s}_{\text{out}}$ of the preceding operation) can still be expressed as in Eq. (40). Then, the new $\mathbf{s}_{\text{out}}$ is found as

$$
\begin{aligned}
\mathbf{s}_{\text{out}} = \mathbf{F}_{\mathbf{a_x},\mathbf{a_y}}\mathbf{s}_{\text{in}} &= \frac{\Delta u}{2}\begin{bmatrix} \cos(a_x\pi/2) & 0 & \sin(a_x\pi/2) & 0 \\ 0 & \cos(a_y\pi/2) & 0 & \sin(a_y\pi/2) \\ -\sin(a_x\pi/2) & 0 & \cos(a_x\pi/2) & 0 \\ 0 & -\sin(a_y\pi/2) & 0 & \cos(a_y\pi/2) \end{bmatrix}\begin{bmatrix} \cos\ \phi_1 \\ \sin\ \phi_1\cos\ \phi_2 \\ \sin\ \phi_1\sin\ \phi_2\cos\ \phi_3 \\ \sin\ \phi_1\sin\ \phi_2\sin\ \phi_3 \end{bmatrix} \\
&= \frac{\Delta u}{2}\begin{bmatrix} \cos(a_x\pi/2)\cos\ \phi_1 + \sin(a_x\pi/2)\sin\ \phi_1\sin\ \phi_2\cos\ \phi_3 \\ \cos(a_y\pi/2)\sin\ \phi_1\cos\ \phi_2 + \sin(a_y\pi/2)\sin\ \phi_1\sin\ \phi_2\sin\ \phi_3 \\ -\sin(a_x\pi/2)\cos\ \phi_1 + \cos(a_x\pi/2)\sin\ \phi_1\sin\ \phi_2\cos\ \phi_3 \\ -\sin(a_y\pi/2)\sin\ \phi_1\cos\ \phi_2 + \cos(a_y\pi/2)\sin\ \phi_1\sin\ \phi_2\sin\ \phi_3 \end{bmatrix}. \qquad (43)
\end{aligned}
$$

As in the coordinate rotator step, $\mathbf{s}_{\text{out}}$ again defines the boundary of the output WD. Once again it defines a 3-sphere since $u_x^2+u_y^2+\mu_x^2+\mu_y^2=(\Delta u/2)^2$. This too can be easily shown by using simple algebra and trigonometric function properties. This is an expected result since the FRT corresponds to rotation in the joint space-frequency; if the original confinement region is an n-sphere, it remains an n-sphere after a 2D-S-FRT. Therefore, we again need not change the number of samples and sampling grid before the FRT step.

C. Second Coordinate Rotator

Similar considerations as with the first coordinate rotation apply so that an increase in the number of samples or a change in the sampling grid is not needed.

D. 2D Scaling Operation

Up to the scaling stage, we do not have to worry at all about the sampling rate. The three steps which constitute the first stage have the effect of rotating the original 3-sphere, and the extent of the 4D Wigner distribution remains unchanged in all directions. We are able to track the confinement boundary through the steps precisely since we are able to write down the entire boundary parametrically by using hyperspherical coordinates and since after each step, the transformed points still form a 3-sphere. In fact, the WD of the signal is confined within the same 3-sphere as at the beginning. However, the scaling operation does not preserve the 3-sphere and thus it is very difficult to track all the points on the boundary since they may not constitute an easily trackable geometrical object by analytical and parametric means. Therefore, instead of tracking the infinite number of boundary points of our 3-sphere, we will use a tesseract (a 4-cube), which is basically the counterpart of an ordinary cube in $\mathbb{R}^4$, just as the 3-sphere is the counterpart of the ordinary sphere [52]. The unit tesseract is defined as

$$
\{(x_1,x_2,x_3,x_4) \in \mathbb{R}^4 : -1 \le x_i \le 1\}. \qquad (44)
$$

It has 16 vertices and we will use these 16 points to track the WD after the scaling operation. We take the smallest tesseract that contains the 3-sphere within itself and use its 16 vertices to find the 16 vertices of the output. These 16 vertices define the maximum extents of the distribution, and by employing them we can safely define the worst-case boundary confining the WD after the operation. Then the two space-bandwidth products can be calculated by finding, separately for each of the four coordinates, the maximum distances between the corresponding coordinates of the 16 vertices. Readers wishing to find a simpler example of such a streamlined procedure in a 1D setting can refer to [55].

Let us represent, in $\mathbb{R}^4$, the coordinates of the 16 vertices of the tesseract of edge length Δu (which is the smallest one confining the 3-sphere with diameter Δu) with columns of the matrix $\mathbf{V}$,

$$\mathbf{V}=\frac{\Delta u}{2}\begin{bmatrix}1&1&1&1&1&1&1&1&-1&-1&-1&-1&-1&-1&-1&-1\\1&1&1&1&-1&-1&-1&-1&1&1&1&1&-1&-1&-1&-1\\1&1&-1&-1&1&1&-1&-1&1&1&-1&-1&1&1&-1&-1\\1&-1&1&-1&1&-1&1&-1&1&-1&1&-1&1&-1&1&-1\end{bmatrix}. \quad (45)$$

After the scaling operation is performed, these coordinates of the 16 input vertices are mapped to 16 new vertices, which we will hold in the columns of $\bar{\mathbf{V}}$ as follows:

$$\bar{\mathbf{V}}=\begin{bmatrix}\mathbf{v_1}&\mathbf{v_2}&\mathbf{v_3}&\cdots&\mathbf{v_{15}}&\mathbf{v_{16}}\end{bmatrix}=\begin{bmatrix}\mathbf{S}&0\\0&\mathbf{S}^{-1}\end{bmatrix}\mathbf{V}, \quad (46)$$

where $\mathbf{v_i}$ ($\mathbf{i}=1,2,\ldots,16$) are vectors in $\mathbb{R}^4$ that hold the coordinates of the scaled vertices. Then, we need to find the coordinate-wise distances for every possible combination of pairs of vertices, for each of the four coordinates separately. There are 120 possible combinations of pairs out of 16 vectors. We calculate the distances between their coordinates and denote this with $\mathbf{d_{i,j}}$ as

$$\mathbf{d_{i,j}}=\begin{bmatrix}|\mathbf{v_i}(1)-\mathbf{v_j}(1)|\\|\mathbf{v_i}(2)-\mathbf{v_j}(2)|\\|\mathbf{v_i}(3)-\mathbf{v_j}(3)|\\|\mathbf{v_i}(4)-\mathbf{v_j}(4)|\end{bmatrix}, \quad (47)$$

and then construct the 4×120 distance matrix $\mathbf{D}$,

$$\mathbf{D}=\begin{bmatrix}\mathbf{d_{1,2}}&\mathbf{d_{1,3}}&\cdots&\mathbf{d_{1,16}}&\mathbf{d_{2,3}}&\mathbf{d_{2,4}}&\cdots&\mathbf{d_{2,16}}&\cdots&\mathbf{d_{15,16}}\end{bmatrix} \quad (48)$$

where $i=1,2,\ldots,15$ and $j=i+1,\ldots,16$. By using $\mathbf{D}$, we can define the sampling grid and sampling rates that are necessary to represent the function without any information loss. The extent of the function along u_x and u_y should be $\max(D_{1,1},D_{1,2},D_{1,3},\ldots,D_{1,120})$ and $\max(D_{2,1}, D_{2,2},D_{2,3},\ldots,D_{2,120})$, respectively. On the intervals along u_x and u_y given above, the samples should be taken with intersample spacings of $(\max(D_{3,1},D_{3,2},D_{3,3},\ldots, D_{3,120}))^{-1}$ and $(\max(D_{4,1},D_{4,2},D_{4,3},\ldots,D_{4,120}))^{-1}$, respectively. The corresponding space-bandwidth products are then equal to

$$N_{Sx}=\max(D_{1,1},D_{1,2},D_{1,3},\ldots,D_{1,120})\times\max(D_{3,1},D_{3,2},D_{3,3},\ldots,D_{3,120}), \quad (49)$$

$$N_{Sy}=\max(D_{2,1},D_{2,2},D_{2,3},\ldots,D_{2,120})\times\max(D_{4,1},D_{4,2},D_{4,3},\ldots,D_{4,120}), \quad (50)$$

and the total necessary number of samples after the scaling is given by $\tilde{N}=N_{Sx}N_{Sy}$. Remember that the number of samples should be increased to this number $\tilde{N}=N_{Sx}N_{Sy}$ before the scaling operation is performed (the minimum appropriate integer number of samples greater than the calculated values may be used for simplicity). The determined number of samples should be uniformly spread so as to snugly fit the original extents (thus they will be spaced closer than the original samples). After the scaling operation is performed by using the matrix $\mathbf{S}$, these samples are transformed to new and extended positions as predicted by the above calculations. Finally, although not a necessity, a similar simple interpolation may be employed (as done after the coordinate rotations) to carry the samples from these transformed locations to the regular grid within the predicted extents. This may facilitate the implementation of the next operation.

E. 2D-CM

The 2D-CM operation is the stage that is mainly burdened with any shears which may be inherent in the 2D-NS-LCT to be computed. Such shears may considerably increase the space-bandwidth products of the function. These increases are unavoidable if these elongating distortions in the space-frequency are part of the 2D-NS-LCT which we wish to compute. This will in turn require an increase in the number of samples if we wish to be able to reconstruct the continuous output function without any information loss. Therefore, as in the previous subsection, we must increase the number of samples prior to the chirp multiplication operation. The vertices obtained as a result of the scaling operation are taken as the starting vertices for the 2D-CM operation. We begin with the coordinates of these vertices, denoted by $\bar{\mathbf{V}}$, determine what happens to them as a result of the 2D-CM operation, and calculate the new difference matrix $\mathbf{D}$ by using the following equation along with Eqs. (47)–(50):

$$\bar{\bar{\mathbf{V}}}=\begin{bmatrix}\mathbf{v_1}&\mathbf{v_2}&\mathbf{v_3}&\cdots&\mathbf{v_{15}}&\mathbf{v_{16}}\end{bmatrix}=\begin{bmatrix}\mathbf{I}&0\\-\mathbf{G}&\mathbf{I}\end{bmatrix}\bar{\mathbf{V}}. \quad (51)$$

Finally, the sampling extents, rates, and locations can be determined similarly as in the scaling stage. After the number of samples has been increased, the 2D-CM stage can be safely performed to complete the entire transformation.

F. Summary of the Algorithm

Having explained all the stages in detail, we summarize the entire algorithm stage by stage. The algorithm can be compactly stated in operator notation as follows:

$$\mathcal{C}_{\mathbf{M}}=\mathcal{Q}_{\mathbf{G}}K_G\mathcal{M}_{\mathbf{S}}K_S\mathcal{R}_{r_2}\mathcal{F}_{a_x,a_y}J\mathcal{R}_{r_1}, \quad (52)$$

where the operators $\mathcal{Q}_{\mathbf{G}}$, $\mathcal{M}_{\mathbf{S}}$, $\mathcal{R}_{r_2}$, $\mathcal{F}_{a_x,a_y}$, and $\mathcal{R}_{r_1}$, respectively, represent the 2D-CM with parameter matrix $\mathbf{G}$, the 2D scaling with parameter matrix $\mathbf{S}$, the coordinate rotation with angle r_2, the 2D-S-FRT with orders a_x and a_y, and the coordinate rotation with angle r_1. J stands for a simple interpolation without oversampling that is performed to obtain the function on a regular rectangular grid from the rotated samples. K_S and K_G stand for the interpolation operations before the scaling and chirp multiplication operations, respectively. Beyond the task of J,

these also increase the number of samples as explained in Subsections 5.D and 5.E.

The algorithm can be summarized as follows:

1. Normalize the input field (function) as explained in Subsection 3.C and obtain the input samples.
2. Given the transform matrix **M**, obtain the chirp multiplication (**G**) and scaling (**S**) matrices, the coordinate rotation angles (r_1 and r_2), and FRT orders (a_x and a_y) by using Eqs. (28)–(37).
3. Perform the first coordinate rotation and obtain the samples on a regular grid by simple interpolation.
4. Use the fast algorithm for 1D-FRTs to implement the 2D-S-FRT by successively applying 1D-FRTs along the two dimensions.
5. Perform the second coordinate rotation and obtain the samples on a regular grid by simple interpolation.
6. Use the method given in Subsection 5.D to obtain the necessary number of samples before the 2D scaling operation, perform the oversampling, and then apply the scaling operation. Optionally, go back to a regular rectangular grid by simple interpolation after the scaling has changed the locations of the samples to a non-rectangular grid.
7. Use the method given in Subsection 5.E to obtain the necessary number of samples before the 2D chirp multiplication operation, perform the oversampling, and then apply the chirp multiplication operation.

6. NUMERICAL RESULTS

Here we report numerical results for some example functions and transforms in order to demonstrate the performance and accuracy of our algorithm. We also discuss sources of error in our algorithm and the effect of interpolation methods on the error. As example input functions, we consider the 2D Gaussian field $\exp(-\pi(x^2+y^2))$ and denote it with $F1$, the 2D chirped-Gaussian field $\exp(-\pi(x^2+y^2))\exp(-i\pi(x^2+y^2))$ and denote it with $F2$, and a 2D non-symmetric chirped Gaussian field $\exp(-\pi(3x^2+y^2))\exp(-i\pi(x^2+2y^2))$ and denote it with $F3$. All these first three input fields are sampled on a 64×64 grid. Additionally, we also consider a more challenging function exhibiting discontinuities and larger frequency extents depicted in Fig. 1. This S-shaped function is denoted with $F4$ and is sampled on a 256×256 grid. We consider two different arbitrarily chosen 2D-NS-LCTs: the first one

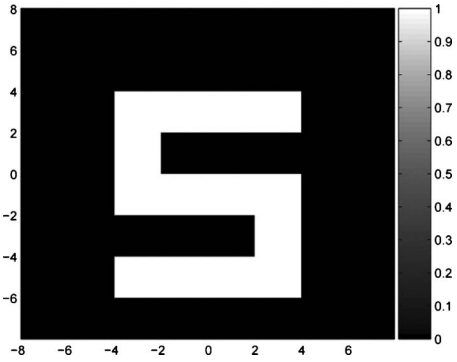
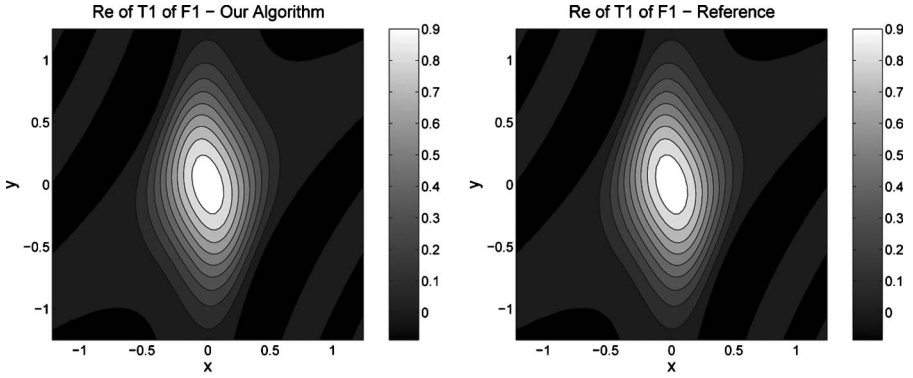
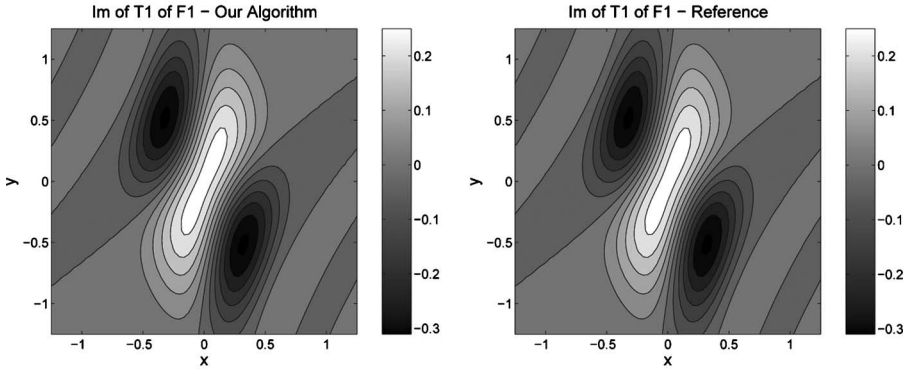


Fig. 1. Example function $F4$.

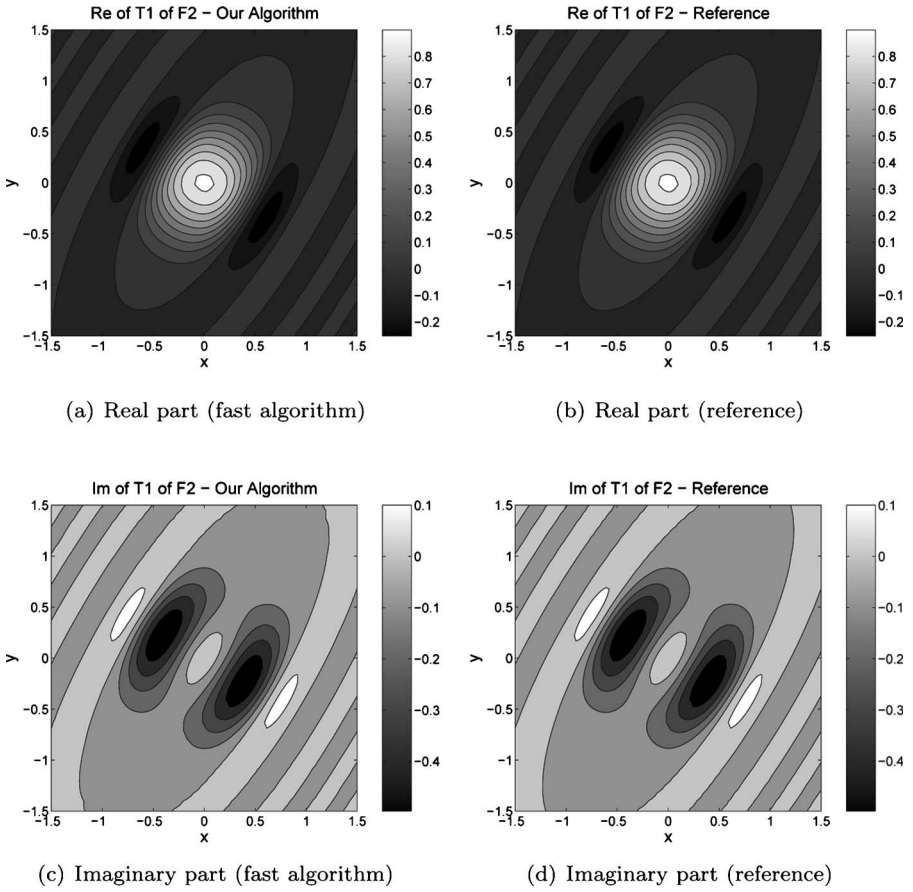


(a) Real part (fast algorithm) (b) Real part (reference)



(c) Imaginary part (fast algorithm)) (d) Imaginary part (reference)

Fig. 2. $T1$ of $F1$ (our algorithm and reference).



(a) Real part (fast algorithm) (b) Real part (reference)

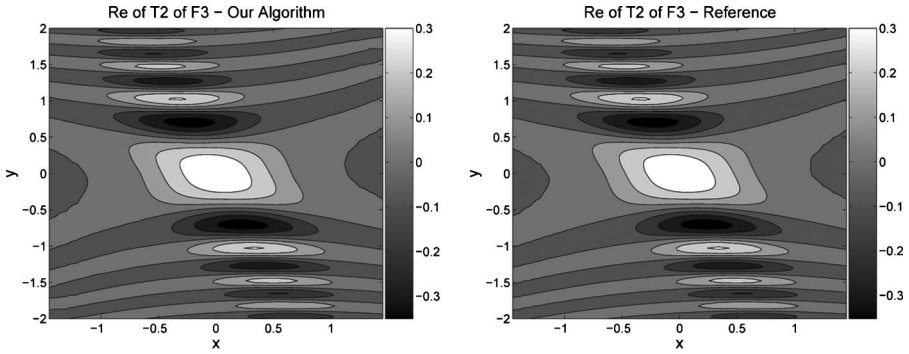
(c) Imaginary part (fast algorithm) (d) Imaginary part (reference)

Fig. 3. $T1$ of $F2$ (our algorithm and reference).

($T1$) has a parameter set $(\alpha_x, \beta_x, \gamma_x, \alpha_y, \beta_y, \gamma_y, \eta_x, \eta_y, \eta_\alpha, \eta_\gamma) = (-3, -2, -1, 2, 3, 4, 0.1, 0.2, 1, -0.1)$ and the second one ($T2$) has a parameter set (1, 2, 3, −2, −1, −0.8, 0.6, −0.5, 0.3, –0.4). As a result of the space-bandwidth and sampling rate control procedure presented in Section 5 and for the given number of initial samples, the output fields are obtained by the algorithm on 141×166 and 740×211 sampling grids for $T1$ and $T2$, respectively, for the input functions $F1$, $F2$, and $F3$. For the input function $F4$, the output grids are 563×663 and 2958×842 for $T1$ and $T2$, respectively. $T1$ is of such a nature that it requires a relatively small amount of oversampling, whereas $T2$ is of such a nature that it requires a relatively large amount of oversampling. These oversamplings are necessary to be able to recover the continuous output from the output samples produced by the algorithm.

The 2D-NS-LCTs ($T1$ and $T2$) of the functions $F1$, $F2$, $F3$, $F4$ have been calculated both by the presented fast algorithm and by an extremely finely tuned and inefficient brute force numerical approach based on the 2D Simpson's method [58] which we use as an accurate reference. The results for $T1$ of ($F1$, $F2$, $F4$) and $T2$ of ($F3$, $F4$) along with the corresponding brute force reference results are plotted in Figs. 2–6. The error percentages for all functions ($F1$, $F2$, $F3$, $F4$) are tabulated in Table 1, for both transforms $T1$ and $T2$. There are no visible differences for $F1$, $F2$, $F3$ and very small visible difference for $F4$. We define the error as the energy of the difference of the two results normalized by the energy of the reference, expressed as a percentage. The tabulated error percentages show that the presented fast algorithm is very accurate. Another important observation from Table 1 is that the error does not depend so much on the transform parameter set as is does on the transformed function; the error percentages for $T1$ and $T2$ are close to each other. In general, our algorithm maintains approximately the same performance over different transforms. A similar conclusion was reached for the 1D case [34,35]. To the best of our knowledge the presented algorithm is the first fast and accurate algorithm that is capable of computing the very general class of 2D-NS-LCTs and the first generalization of the 1D fast algorithms for LCTs to two dimensions. Moreover, it also deals with the space-bandwidth and sampling rate issues very carefully so that the output samples—indeed the samples at any stage—are sufficient to accurately reconstruct the underlying continuous function, but are not wastefully redundant either. Therefore our algorithm is able to effectively obtain a continuous transform from a continuous input function.

In Table 1, we also show the errors that arise when the discrete Fourier transform (DFT) is used to approximately compute the ordinary 2D-FT of the same functions. [The DFT would most likely be implemented with the fast Fourier transform (FFT) algorithm but how the DFT is implemented does not affect the error comparison.] The same reference method that we use in calculating the error percentages for our algorithm is used to nu-



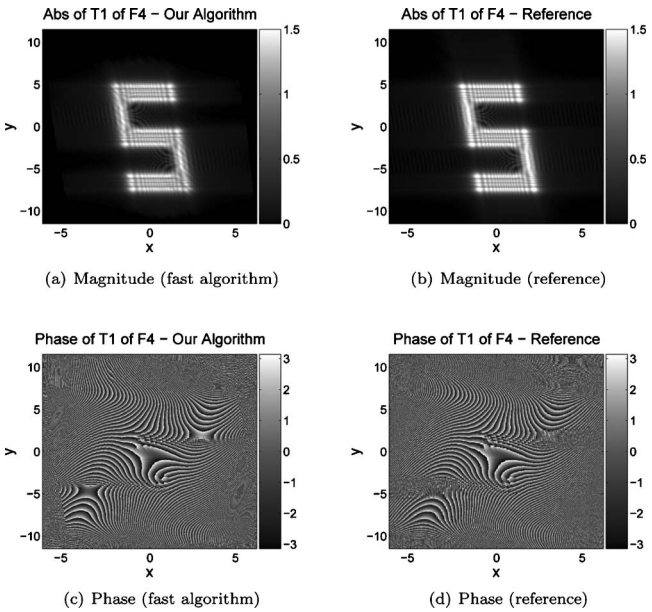
(a) Real part (fast algorithm) (b) Real part (reference)

(c) Imaginary part (fast algorithm) (d) Imaginary part (reference)

Fig. 4. $T2$ of $F3$ (our algorithm and reference).

merically calculate "exact" continuous FTs of the example functions. The DFT serves as an ultimate benchmark for comparing our results. Theoretically, our algorithm cannot reduce the error below that value which results from computing a FT with the DFT because they share the same inevitable source of error that arises from the fundamental fact that a signal and its transform cannot both be of finite extent. In the 1D version of our algorithm, as well as the separable 2D case, it is possible to achieve errors which approach that for the DFT, and which are thus the best which one may ever hope to obtain [34,35]. Unfortunately, the necessity of interpolation in the 2D case does not allow this, but still it is possible to achieve very low errors that would be acceptable in most applications.

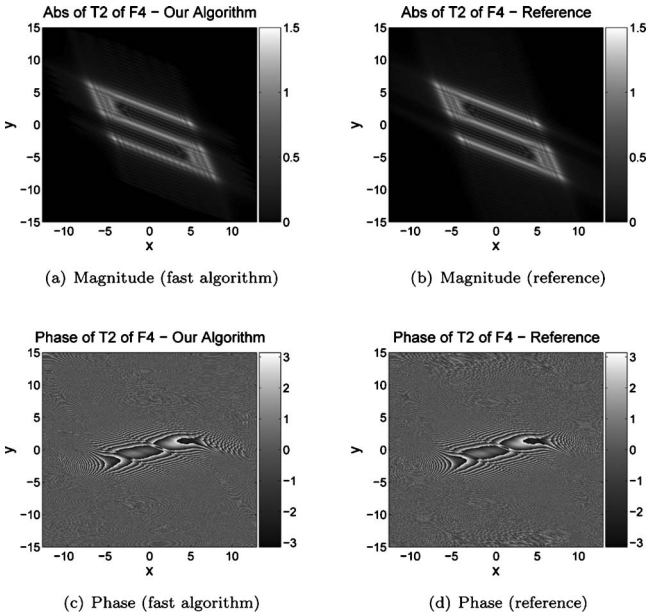
The key observations that can be made from Table 1 are as follows. The resulting errors depend strongly on



(a) Magnitude (fast algorithm) (b) Magnitude (reference)

(c) Phase (fast algorithm) (d) Phase (reference)

Fig. 5. $T1$ of $F4$ (our algorithm and reference).



(a) Magnitude (fast algorithm) (b) Magnitude (reference)

(c) Phase (fast algorithm) (d) Phase (reference)

Fig. 6. $T2$ of $F4$ (our algorithm and reference).

Table 1. Percentage Errors for Different Functions *F* and Transforms *T*

	*T*1	*T*2	DFT
*F*1	2.25×10^{-3}	3.82×10^{-4}	2.12×10^{-23}
*F*2	1.12×10^{-2}	1.09×10^{-3}	2.02×10^{-21}
*F*3	7.17×10^{-2}	3.21×10^{-3}	2.58×10^{-8}
*F*4	2.07	1.92	1.05

Table 2. Percentage Errors for Different Interpolation Methods and Functions *F* for *T*1

	*F*1	*F*2	*F*3	*F*4
Nearest	3.4×10^{-3}	1.75×10^{-1}	6.18×10^{-1}	11.3
Bilinear	1.02×10^{-2}	5.04×10^{-2}	2.66×10^{-1}	4.33
Cubic	2.25×10^{-3}	1.12×10^{-2}	7.17×10^{-2}	2.07

Table 3. Percentage Errors for Different Interpolation Methods and Functions *F* for *T*2

	*F*1	*F*2	*F*3	*F*4
Nearest	1.72×10^{-1}	3.28×10^{-1}	4.59×10^{-1}	11.24
Bilinear	1.78×10^{-2}	5.58×10^{-2}	8.4×10^{-2}	6.24
Cubic	3.82×10^{-4}	1.09×10^{-3}	3.21×10^{-3}	1.92

the function and the assumed space and spatial-frequency extents. Indeed, this is the main determinant of the error for a given interpolation method. Different functions have differing degrees of decay rates of their tails and, for given assumed extents, different amounts of energy left out of the extents. Since a function cannot be made to contain 100% of its energy in both the space and spatial-frequency domains, a compromise between error and computational complexity is necessary. If we choose the extents within which we assume the function and its FT to be mostly contained in a conservative manner, the extents will be relatively large and the number of samples will be relatively large. If we economize on the extents and the number of samples, a relatively large fraction of the energy will be left outside and the resulting error will be large. Among our examples, *F*4 is an example where the space-bandwidth product has been chosen less conservatively than the other examples, and therefore the error is relatively large around 2%. The error can be reduced by increasing the number of samples taken.

From a fundamental perspective, our algorithm is supposed to compute 2D-NS-LCTs with a performance similar to the DFT in computing the FT. As noted, this is achieved for separable transforms which reduce to 1D transforms. However, in the general non-separable case, although our algorithm is quite accurate, for the first three functions, its accuracy is quite below that for the DFT. This degradation is due to the complex and challenging nature of non-separable LCTs which forces us to employ interpolation operations from irregular grids to regular grids. This is an important source of error. For *F*4, our algorithm has a comparable accuracy with the DFT. This is due to the fact that, in this case, the error is a result of the significant amount of signal energy that lies outside the assumed space and spatial-frequency extents. This source of error, which affects both our algorithm and the DFT in the same way, dominates the error arising from the interpolation (which affects only the non-separable LCT computation) so that the results are similar. On the other hand, for the other functions, the interpolation error (which does not affect the DFT) results in higher errors for the LCT computations as compared to the DFT.

To be more confident in the above claims, we also studied the effects of the method of interpolation on our algorithm and studied how they change the accuracy of the algorithm. We employed in our algorithm *nearest neighbor*, *bilinear*, and *cubic* interpolation methods because they are among the most standard, mainstream, and efficient methods [56,57]. Different versions of the algorithm have been implemented by using each of the above methods. The error percentages resulting from the use of different interpolation methods are tabulated in Tables 2 and 3 for *T*1 and *T*2, respectively. As can be seen from the tabulated data, the error values are affected considerably by the interpolation method chosen. The best results are obtained when we use the cubic interpolation method, which is the most advanced among the three. Since there are essentially two sources of error, the one that is fundamental equally affecting LCTs and the DFT, we are not surprised to observe that as the quality of the interpolation is increased, the accuracy of the algorithm improves and approaches the DFT benchmark.

The results of our fast algorithm were obtained within a couple of seconds by using Matlab code running on a standard personal computer. The calculation of the brute force reference results took several days.

7. CONCLUSIONS

We presented an algorithm for the fast digital computation of the most general family of two-dimensional non-separable linear canonical transforms (2D-NS-LCTs). This family of transform integrals represents a quite general class of two-dimensional (2D) quadratic-phase systems in optics. Our approach is based on concepts from the signal analysis and processing rather than the conventional numerical analysis. With careful consideration of sampling issues, the number of samples $M\times N$ of the sampling grid can be chosen very close to the space-bandwidth product of the functions. A naive approach based on the examination of the frequency content of the integral kernels would, on the other hand, result in an unnecessarily high number of samples being taken due to the highly oscillatory nature of the kernels, which would not only be representationally inefficient but also increase the computation time and storage requirements. The transform output may have a higher space-bandwidth product than the input due to the nature of the transform family. Through careful space-bandwidth tracking and control, we can assure that the output samples obtained are accurate approximations to the true ones and that

they are sufficient (but not unnecessarily redundant) in the Nyquist–Shannon sense, allowing the full reconstruction of the underlying continuous function.

The algorithm takes the samples of the input function and maps them to the samples of the continuous 2D-NS-LCT of this function in the same sense that the FFT implementation of the DFT computes the samples of the continuous FT of a function. The presented algorithm can be used for the fast and efficient realization of filtering in linear canonical transform (LCT) domains [59].

The only inevitable source of deviation from exactness in our algorithm arises from the fundamental fact that a function and its Fourier transform (FT) cannot both be of finite extent. This limitation affects not only the separable and one-dimensional (1D) versions of the algorithm reported earlier, but also the computation of FTs using the DFT. Thus this is a source of error we cannot hope to overcome.

A second source of error which was not of substantial impact in the 1D case or the separable 2D case but which is significant in the non-separable 2D case arises from the necessity to carry out interpolations to revert samples on rotated grids to the original rectangular grid. This error depends on how accurately the interpolation operation is handled. We have used well-established and standard methods for interpolation that are readily available, since advancing methods of interpolation is beyond the scope of this paper. While we believe that the levels of accuracy attained with these interpolation methods will be sufficient for most applications, in those cases where they are not, more efficient and customized interpolation methods for non-rectangular grids can be utilized to further improve the accuracy. We have also developed the link between the compact matrix-based 16-parameter definition of 2D-NS-LCTs and the ten-parameter explicit-kernel definition.

ACKNOWLEDGMENTS

A. Koç and L. Hesselink acknowledge support from the Research Laboratories of the General Electric (GE) Corporation in New York. H. M. Ozaktas acknowledges partial support of the Turkish Academy of Sciences.

Note in proof: We have recently been made aware of a new work [60] that reviews algorithms for real one-dimensional and real symmetrical (separable) two-dimensional LCTs.

REFERENCES

1. M. J. Bastiaans, "The Wigner distribution function applied to optical signals and systems," Opt. Commun. **25**, 26–30 (1978).
2. M. J. Bastiaans, "Applications of the Wigner distribution function in optics," in *The Wigner Distribution: Theory and Applications in Signal Processing* (Elsevier, 1997), pp. 375–426.
3. M. J. Bastiaans, "Wigner distribution function and its application to first-order optics," J. Opt. Soc. Am. **69**, 1710–1716 (1979).
4. A. Sahin, M. A. Kutay, and H. M. Ozaktas, "Nonseparable two-dimensional fractional Fourier transform," Appl. Opt. **37**, 5444–5453 (1998).
5. R. K. Luneburg, *Mathematical Theory of Optics* (University of California Press, 1966).
6. M. Nazarathy and J. Shamir, "First-order optics—a canonical operator representation: lossless systems," J. Opt. Soc. Am. **72**, 356–364 (1982).
7. J. W. Goodman, *Introduction to Fourier Optics*, 3rd ed. (Roberts & Company, 2005).
8. K. B. Wolf, "Construction and properties of canonical transforms," in *Integral Transforms in Science and Engineering* (Plenum, 1979), Chap. 9.
9. M. J. Bastiaans and T. Alieva, "Classification of lossless first-order optical systems and the linear canonical transformation," J. Opt. Soc. Am. A **24**, 1053–1062 (2007).
10. S. A. Collins, Jr., "Lens system diffraction integral written in terms of matrix optics," J. Opt. Soc. Am. **60**, 1168–1177 (1970).
11. M. Bastiaans and T. Alieva, "Synthesis of an arbitrary *ABCD* system with fixed lens positions," Opt. Lett. **31**, 2414–2416 (2006).
12. U. Sümbül and H. M. Ozaktas, "Fractional free space, fractional lenses, and fractional imaging systems," J. Opt. Soc. Am. A **20**, 2033–2040 (2003).
13. S. C. Pei and J. J. Ding, "Eigenfunction of linear canonical transform," IEEE Trans. Signal Process. **50**, 11–26 (2002).
14. J. Rodrigo, T. Alieva, and M. Luisa Calvo, "Optical system design for orthosymplectic transformations in phase space," J. Opt. Soc. Am. A **23**, 2494–2500 (2006).
15. R. Simon and K. B. Wolf, "Structure of the set of paraxial optical systems," J. Opt. Soc. Am. A **17**, 342–355 (2000).
16. T. Alieva and M. J. Bastiaans, "Properties of the canonical integral transformation," J. Opt. Soc. Am. A **24**, 3658–3665 (2007).
17. A. E. Siegman, *Lasers* (University Science Books, 1986).
18. D. F. V. James and G. S. Agarwal, "The generalized Fresnel transform and its applications to optics," Opt. Commun. **126**, 207–212 (1996).
19. C. Palma and V. Bagini, "Extension of the Fresnel transform to *ABCD* systems," J. Opt. Soc. Am. A **14**, 1774–1779 (1997).
20. S. Abe and J. T. Sheridan, "Generalization of the fractional Fourier transformation to an arbitrary linear lossless transformation: an operator approach," J. Phys. A **27**, 4179–4187 (1994); Corrigenda in pp. 7937–7938.
21. S. Abe and J. T. Sheridan, "Optical operations on wavefunctions as the Abelian subgroups of the special affine Fourier transformation," Opt. Lett. **19**, 1801–1803 (1994).
22. J. Hua, L. Liu, and G. Li, "Extended fractional Fourier transforms," J. Opt. Soc. Am. A **14**, 3316–3322 (1997).
23. A. Sahin, H. M. Ozaktas, and D. Mendlovic, "Optical implementations of two-dimensional fractional Fourier transforms and linear canonical transforms with arbitrary parameters," Appl. Opt. **37**, 2130–2141 (1998).
24. A. Sahin, H. M. Ozaktas, and D. Mendlovic, "Optical implementation of the two-dimensional fractional Fourier transform with different orders in the two dimensions," Opt. Commun. **120**, 134–138 (1995).
25. M. F. Erden, H. M. Ozaktas, A. Sahin, and D. Mendlovic, "Design of dynamically adjustable anamorphic fractional Fourier transformer," Opt. Commun. **136**, 52–60 (1997).
26. H. M. Ozaktas, Z. Zalevsky, and M. A. Kutay, *The Fractional Fourier Transform with Applications in Optics and Signal Processing* (Wiley, 2001).
27. M. Moshinsky and C. Quesne, "Linear canonical transformations and their unitary representations," J. Math. Phys. **12**, 1772–1780 (1971).
28. E. G. Abramochkin and V. G. Volostnikov, "Generalized Gaussian beams," J. Opt. A, Pure Appl. Opt. **6**, S157–S161 (2004).
29. L. Allen, M. W. Beijersbergen, R. J. C. Spreeuw, and J. P. Woerdman, "Orbital angular momentum of light and the transformation of Laguerre Gaussian laser modes," Phys. Rev. A **45**, 8185–8189 (1992).
30. J. Rodrigo, T. Alieva, and M. Calvo, "Experimental implementation of the gyrator transform," J. Opt. Soc. Am. A **24**, 3135–3139 (2007).
31. J. Rodrigo, T. Alieva, and M. Calvo, "Gyrator transform: properties and applications," Opt. Express **15**, 2190–2203 (2007).
32. J. A. Rodrigo, T. Alieva, and M. L. Calvo, "Applications of

gyrator transform for image processing," Opt. Commun. **278**, 279–284 (2007).
33. K. Wolf and T. Alieva, "Rotation and gyration of finite two-dimensional modes," J. Opt. Soc. Am. A **25**, 365–370 (2008).
34. H. M. Ozaktas, A. Koç, I. Sari, and M. A. Kutay, "Efficient computation of quadratic-phase integrals in optics," Opt. Lett. **31**, 35–37 (2006).
35. A. Koç, H. M. Ozaktas, C. Candan, and M. A. Kutay, "Digital computation of linear canonical transforms," IEEE Trans. Signal Process. **56**, 2383–2394 (2008).
36. B. M. Hennelly and J. T. Sheridan, "Fast numerical algorithm for the linear canonical transform," J. Opt. Soc. Am. A **22**, 928–937 (2005).
37. J. J. Healy and J. T. Sheridan, "Cases where the linear canonical transform of a signal has compact support or is band-limited," Opt. Lett. **33**, 228–230 (2008).
38. J. J. Healy, B. M. Hennelly, and J. T. Sheridan, "Additional sampling criterion for the linear canonical transform," Opt. Lett. **33**, 2599–2601 (2008).
39. J. J. Healy and J. T. Sheridan, "Sampling and discretization of the linear canonical transform," Signal Process. **89**, 641–648 (2009).
40. F. Oktem and H. M. Ozaktas, "Exact relation between continuous and discrete linear canonical transforms," IEEE Signal Process. Lett. **16**, 727–730 (2009).
41. T. Alieva and M. J. Bastiaans, "Alternative representation of the linear canonical integral transform," Opt. Lett. **30**, 3302–3304 (2005).
42. K. B. Wolf, *Geometric Optics on Phase Space* (Springer, 2004).
43. G. B. Folland, *Harmonic Analysis in Phase Space* (Princeton U. Press, 1989).
44. A. Stern and B. Javidi, "Sampling in the light of Wigner distribution," J. Opt. Soc. Am. A **21**, 360–366 (2004).
45. D. Mendlovic and H. M. Ozaktas, "Fractional Fourier transforms and their optical implementation: I," J. Opt. Soc. Am. A **10**, 1875–1881 (1993).
46. H. M. Ozaktas and D. Mendlovic, "Fractional Fourier transforms and their optical implementation: II," J. Opt. Soc. Am. A **10**, 2522–2531 (1993).
47. H. M. Ozaktas and D. Mendlovic, "Fourier transforms of fractional order and their optical interpretation," Opt. Commun. **101**, 163–169 (1993).
48. H. M. Ozaktas, B. Barshan, D. Mendlovic, and L. Onural, "Convolution, filtering, and multiplexing in fractional Fourier domains and their relation to chirp and wavelet transforms," J. Opt. Soc. Am. A **11**, 547–559 (1994).
49. L. B. Almeida, "The fractional Fourier transform and time-frequency representations," IEEE Trans. Signal Process. **42**, 3084–3091 (1994).
50. H. M. Ozaktas and D. Mendlovic, "Fractional Fourier optics," J. Opt. Soc. Am. A **12**, 743–751 (1995).
51. H. M. Ozaktas and M. F. Erden, "Relationships among ray optical, Gaussian beam, and fractional Fourier transform descriptions of first-order optical systems," Opt. Commun. **143**, 75–86 (1997).
52. H. S. M. Coxeter, *Regular Polytopes* (Dover, 1973).
53. D. W. Henderson, *Experiencing Geometry: In Euclidean, Spherical, and Hyperbolic Spaces*, 2nd ed. (Pearson, 2001).
54. H. M. Ozaktas, O. Arıkan, M. A. Kutay, and G. Bozdağı, "Digital computation of the fractional Fourier transform," IEEE Trans. Signal Process. **44**, 2141–2150 (1996).
55. B. M. Hennelly and J. T. Sheridan, "Generalizing, optimizing, and inventing numerical algorithms for the fractional Fourier, Fresnel, and linear canonical transforms," J. Opt. Soc. Am. A **22**, 917–927 (2005).
56. B. Jahne, *Digital Image Processing*, 5th ed. (Springer, 2002).
57. R. G. Keys, "Cubic convolution interpolation for digital image processing," IEEE Trans. Acoust., Speech, Signal Process. **29**, 1153–1160 (1981).
58. J. C. Brégains, I. C. Coleman, F. Ares, and E. Moreno, "Calculating directivities with the two-dimensional Simpson's rule," IEEE Antennas Propag. Mag. **46**, 106–112 (2004).
59. B. Barshan, M. A. Kutay, and H. M. Ozaktas, "Optimal filtering with linear canonical transformations," Opt. Commun. **135**, 32–36 (1997).
60. J. Healy and J. T. Sheridan, "Fast linear canonical transforms," J. Opt. Soc. Am. A **27**, 21–30 (2010).