Sliding discrete fractional transforms

J.S. Bhat\textsuperscript{a,}\textsuperscript{*}, C. Vijaya\textsuperscript{b}

\textsuperscript{a}Department of Physics, Karnatak University, Dharwad 580003, India
\textsuperscript{b}SDM College of Engineering & Technology, Dharwad 580002, India

Received 18 June 2006; received in revised form 11 July 2007; accepted 12 July 2007
Available online 22 July 2007

Abstract

Fractional transforms are useful tools for processing of non-stationary signals. The methods of implementing sliding discrete fractional Fourier transform (SDFRFT), sliding discrete fractional cosine transform (SDFRCT) and sliding discrete fractional sine transform (SDFRST) for real time processing of signals are presented. The performances of these sliding transforms, with regard to computational complexity, variance of quantization error and signal-to-noise ratio (SNR), are presented and compared. The three sliding discrete fractional transforms are compared with sliding discrete Fourier transform (SDFT) in terms of SNR. Computational complexity in the case of sliding discrete fractional transform is less than that in the case of discrete fractional transform when a particular time–frequency bin is to be observed. In comparison with SDFT, the sliding discrete fractional transforms require less number of bits for representing coefficients. The SDFRST performs better in comparison with SDFRFT and SDFRCT.

\textcopyright{} 2007 Elsevier B.V. All rights reserved.

Keywords: Fractional transform; Sliding transform

1. Introduction

The transforms are widely employed for signal processing to obtain useful information, which is not explicitly available when the signal is in the time domain. Most of the real time signals such as speech, biomedical signals, etc., are non-stationary signals. The Fourier transform (FT), used for most of the signal processing applications, determines the frequency components present in the signal but with zero time resolution. The fractional transforms, such as fractional Fourier transform (FRFT) [1–3], fractional cosine transform (FRCT) [4–6] or fractional sine transform (FRST) [4,5] describe the energy density or signal intensity simultaneously in the time and frequency plane and have non-zero time frequency resolution in the transform domain. These transforms are used for optical signal processing, time variant filtering, swept frequency filters, pattern recognition and signal compression [1–3].

In real time applications, samples of input signal arrive in a sequential manner. The transform of the signal is obtained by processing sequentially the blocks of \(N\) samples. In this procedure, for the computation of transform of \(N\) input samples, the system has to wait till the arrival of all \(N\) new input samples. Instead, when a new sample arrives,
the transform can be computed by processing the
new block of samples consisting of the newly
arrived sample and the N-1 samples of the previous
block. This technique is referred to as the sliding
 technique. Sliding technique reduces the computa-
tional complexity and improves the speed. This
technique has been successfully employed in com-
puting the discrete Fourier transform (DFT)
of the real time signals and is called sliding discrete
Fourier transform (SDFT) [7,8]. The coefficients
obtained upon using the SDFT provide only the
frequency information. Since the fractional
transforms have non-zero time frequency resolu-
tion, they are better suited for processing the
non-stationary real time signals. FRFT is a general-
ization of ordinary FT with an order parameter
\( \alpha = \pi/2 \). The order parameter \( \alpha \) represents the
angle of rotation of the signal in time–frequency
plane. The FRFT is identical to ordinary FT
when \( \alpha = \pi/2 \) [1–3]. The discrete fractional Fourier
transform (DFRFT) with its kernel expressed in
the closed form has the lowest complexity and
negligible error in computation [9] and many of
the properties are also satisfied. It suits for most of
the real time applications due to the simpler and
closed form of discrete fractional convolution
and correlation [9]. The discrete fractional cosine
transform (DFRCT) and the discrete fractional sine
transform (DFRST) are related to DFRFT [4,5].
However, no much attention has been given to
introduce the sliding technique for fractional
transforms.

In this paper, we present the methods of
computing the three sliding discrete fractional
transforms: sliding discrete fractional Fourier tran-
sform (SDFRFT), sliding discrete fractional cosine
transform (SDFRCT) and sliding discrete fractional
sine transform (SDFRST). Their performances are
compared in terms of computational complexity,
variance of quantization error and signal-to-noise
ratio (SNR). The performances of each of the
proposed sliding discrete fractional transforms are
compared with that of the SDFT, by calculating the
SNR for one of the signals available in the sound
quality assessment material (SQAM). In Section 2, a
brief introduction is given to the closed form
definitions of DFRFT, DFRCT and DFRST. In
Section 3, the methods of implementation of sliding
discrete fractional transforms are presented. The
performance measures are presented and discussed
in Section 4 and the conclusions are given in
Section 5.

2. Closed form discrete fractional transforms

A closed form DFRFT can be obtained by
sampling the input function \( x(t) \) at \( \Delta t \) intervals
and its fractional transform \( X_\alpha(u) \) at \( \Delta u \) intervals.
The DFRFT can be written as [9]

\[
Y_{fr}(m) = \sum_{n=-N}^{N} F_{fr}(m,n)y(n),
\]

where \( y(n) \) represents the samples of input \( x(t) \) with
\(-N \leq n \leq N \) and \( Y_{fr}(m) \) the samples of fractional
transform \( X_\alpha(u) \) with \(-M \leq m \leq M \) and \( M \geq N \). With
\( M = N \), the number of transform coefficients is
same as the number of input samples. The angle \( \alpha \)
represents the angle of rotation of the signal in
time–frequency plane. The suffix \( f \) indicates FT.
The kernel \( F_{fr}(m,n) \) is defined as [9]

\[
F_{fr}(m,n) = A_{fr}e^{j(\cot \pi/2)(m^2\Delta u^2+n^2\Delta t^2)}e^{-j(2\pi nm)p},
\]

where

\[
A_{fr} = \sqrt{\frac{|\sin \alpha| - j \text{sgn}(|\sin \alpha|)\cos \alpha}{P}}
\]

with \( P = 2N+1 \) being the number of input samples.
Eq. (2) is written with the condition that

\[
\Delta t \Delta u = S2\pi \sin \alpha,
\]

with \( S \) being some integer prime to \( P \), usually
denoted as \text{sgn}(\sin \alpha). Sampling intervals, \( \Delta t \) and \( \Delta u \),
are selected such that the condition in (4) is satisfied.
In this method of discretization of the FRFT, the
error is of the order of \( 10^{-6}–10^{-14} \) [9].

The fractional cosine transform of \( x(t) \) is defined as [5]

\[
C_\alpha(u) = X_\alpha(u) + X_\alpha(-u).
\]

The method of discretizing FRCT is same as that
adapted in the case of FRFT. The closed form
DFRCT, \( Y_{cr}(m) \), is given by

\[
Y_{cr}(m) = \sum_{n=0}^{P} F_{cr}(m,n)y(n), \quad 0 \leq m \leq P,
\]

where the kernel, \( F_{cr}(m,n) \), is given by

\[
F_{cr}(m,n) = k_mk_nA_{cr}e^{j(\cot \pi/2)(m^2\Delta u^2+n^2\Delta t^2)}\cos\left(\frac{mn\pi}{P}\right),
\]

with

\[
A_{cr} = \sqrt{\frac{2|\sin \alpha| - j \text{sgn}(|\sin \alpha|)\cos \alpha}{P}},
\]
\[ k_i = \sqrt{(1/2)} \quad \text{for } i = 0 \]
\[ = 1 \quad \text{otherwise} \]
and \[ \Delta t \Delta u = \frac{S \pi \sin \alpha}{P}. \]

This definition reduces to the DCT-I given in [10] when \( \alpha = \pi/2 \).

The fractional sine transform of \( x(t) \) is defined as [5]
\[ S_x(u) = e^{i\pi} (X_x(u) - X_x(-u)). \]  
(8)

The closed form DFRST, \( Y_{sx}(m) \), is given by
\[ Y_{sx}(m) = \sum_{n=1}^{P-1} F_{sx}(m,n)y(n), \quad 1 \leq m \leq P - 1, \]  
(9)
where the kernel, \( F_{sx}(m,n) \), is given by
\[ F_{sx}(m,n) = A_{sx} \frac{e^{i(\cot \alpha)(mn^2/\Delta u^2 + \Delta t^2)}}{P} \sin \left( \frac{mn\pi}{P} \right). \]  
(10)

with
\[ A_{sx} = e^{i(\pi/2)} \sqrt{2(|\sin \alpha| - j \text{sgn}(\sin \alpha) \cos \alpha)} \]
and
\[ \Delta t \Delta u = \frac{S \pi \sin \alpha}{P}. \]

This definition reduces to DST-I given in [10] when \( \alpha = \pi/2 \). In all the three discrete fractional transforms, the inverse transform can be computed by changing the order parameter to \(-\alpha\) and sampling interval at the input to \( \Delta u \) and sampling interval at the output to \( \Delta t \).

Based on the above definitions, a generalized expression for the discrete fractional transform can be written as
\[ Y_{Gx}(m) = \sum_{n=L}^{U} F_{Gx}(m,n)y(n) \]  
(11)
with the limits of the summation index being \( L = -N \) and \( U = N \) for DFRFT, \( L = 0 \) and \( U = P \) for DFRCT and \( L = 1 \) and \( U = P - 1 \) for DFRST. The suffix \( G \) distinguishes the three transforms. \( G = f, c \) and \( s \) correspond to Fourier, cosine and sine transforms, respectively. The kernel, \( F_{Gx}(m,n) \), can be expressed as
\[ F_{Gx}(m,n) = e^{i(\cot \alpha)(mn^2/\Delta u^2 + \Delta t^2)} K_G(m,n). \]  
(12)

The term \( K_G(m, n) \) is different for different fractional transforms.
\[ K_G(m,n) = A_f e^{-j\left(2\pi mn/P\right)} \quad \text{for } G = f \]
\[ = A_c k_m k_n \cos \left( \frac{mn\pi}{P} \right) \quad \text{for } G = c \]
\[ = A_s \sin \left( \frac{mn\pi}{P} \right) \quad \text{for } G = s. \]  
(13)

3. Sliding discrete fractional transforms and their implementations

In the computation of the sliding transform, the process performed on \( P-1 \) samples of the previous block of \( P \) samples is required while considering them along with the new sample [7,8]. However, the sliding discrete fractional transforms cannot be recursive, as in the case of the SDFT, because of the chirp multiplication in the fractional transforms. Consider a sequence of input samples
\[ \ldots y(n-2), y(n-1), y(n), y(n+1), \]
\[ y(n+2), y(n+3) \ldots \]

The discrete fractional transform of \( P \) input samples is given by
\[ Y_{Gx,u-1}(m) = \sum_{k=L}^{U} F_{Gx}(m,k)y(n-1-k) \]
\[ = \sum_{k=L}^{U} e^{i(\cot \alpha)(mn^2/\Delta u^2 + k^2\Delta t^2)} \]
\[ \times K_G(m,k)y(n-1-k) \]
\[ = e^{i(\cot \alpha)(mn^2/\Delta u^2)} \left( y(n-1-L) \right) e^{i(\cot \alpha)(n^2/\Delta u^2)} \]
\[ \times K_G(m,L) + y(n-1-(L+1)) \]
\[ \times e^{i(\cot \alpha)(n^2/\Delta u^2)} K_G(m,L+1) \]
\[ \vdots \]
\[ + y(n-1-(U-1)) e^{i(\cot \alpha)(n^2/\Delta u^2)} \]
\[ \times K_G(m, U-1) + y(n-1-U) e^{i(\cot \alpha)(n^2/\Delta u^2)} \]
\[ \times K_G(m, U). \]  
(14)

Sliding the window by one sample to the right, the discrete fractional transform of another \( P \) samples is given by
\[ Y_{Gx,u}(m) = \sum_{k=L}^{U} F_{Gx}(m,k)y(n-k) \]
\[ = \sum_{k=L}^{U} e^{i(\cot \alpha)(mn^2/\Delta u^2 + k^2\Delta t^2)} K_G(m,k)y(n-k) \]
\[ = e^{i(\cot \alpha)(mn^2/\Delta u^2)} \left( y(n-L) \right) e^{i(\cot \alpha)(n^2/\Delta u^2)} K_G(m,L) \]
+ y(n - (L + 1))e^{j\cot \frac{z}{2}(L+1)\Delta t^2}K_G(m, L + 1)

\vdots

+ y(n - (U - 1))e^{j\cot \frac{z}{2}(U-1)\Delta t^2}K_G(m, U - 1)

+ y(n - U)e^{j\cot \frac{z}{2}U\Delta t^2}K_G(m, U)). \quad (15)

Comparing (14) and (15), all the samples involved in computation of $Y_Gz_{n-1}(m)$ are also in $Y_Gz_{n}(m)$ except for the last sample, $y(n-1-U)$. There is one new sample, $y(n-L)$, in $Y_Gz_{n}(m)$ as the first sample. A first-in-first-out (FIFO) buffer can be employed to hold the input data samples from $y(n-1-L)$ to $y(n-1-U)$. The discrete fractional transform of the samples in the FIFO buffer is computed. The new sample arrived is put into the FIFO buffer and so on.

The first output of sliding discrete fractional transform will be available only after processing the first $P$ input samples. Subsequent output will be available soon after the arrival of new input sample. The implementation of sliding discrete fractional transforms is shown in Fig. 1 for $P = 5$. The multiplication factors are different for different fractional transforms. Table 1 shows the multiplication factors for the three sliding discrete fractional transforms for $P = 5$. Only in the case of the SDFRCT, $A_G$ is real. As the summation index in the case of SDFRCT is 0 to $P$, there is an additional term in its computation in comparison with SDFRFT. Similarly, the summation index in the case of SDFRST is from 1 to $P-1$ and the number of terms is one less in comparison with the SDFRFT.

4. Performance evaluation of sliding discrete fractional transforms

The advantages of using the fractional transforms can easily be observed by calculating the normalized square magnitude of the transform coefficients for a chirp signal [3] such as shown in Fig. 2. The normalized square magnitude of the fractional transforms obtained using DFRFT, DFRCT and DFRST, respectively, is shown in Fig. 3(a)–(c). The normalized square magnitude shown in Fig. 3(d)–(f) is obtained by using the ordinary transforms DFT, discrete cosine transform (DCT) and discrete sine transform (DST), respectively. The optimum value of order parameter $z$ for the compact domain [1,3] is 0.58, 1.3 and 0.78, respectively, for DFRFT, DFRCT and DFRST. The spread of the signal in the transform domain, in all the three ordinary transforms, is wider than those obtained with discrete fractional transform of the samples from $y(n-L)$ to $y(n-U)$ is computed and so on.

<table>
<thead>
<tr>
<th>Multiplication factor</th>
<th>SDFRFT</th>
<th>SDFRCT</th>
<th>SDFRST</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_G$</td>
<td>$e^{j2\Delta t^2 \cot \frac{z}{2}(84m/5)}$</td>
<td>$1/\sqrt{2}$</td>
<td>$e^{j\cot \frac{z}{2}2m^2 \sin(3m\pi/5)}$</td>
</tr>
<tr>
<td>$B_G$</td>
<td>$e^{j\cot \frac{z}{2}2\Delta t^2 (62m/5)}$</td>
<td>$\sqrt{2}\cos(m2\pi/5)$</td>
<td>$e^{j2\Delta t^2 \cot \frac{z}{2}(2m\pi/5)}$</td>
</tr>
<tr>
<td>$C_G$</td>
<td>$1$</td>
<td>$\cos(3m\pi/5)$</td>
<td>$e^{j\cot \frac{z}{2}2\Delta t^2 \sin(3m\pi/5)}$</td>
</tr>
<tr>
<td>$D_G$</td>
<td>$e^{j\cot \frac{z}{2}2\Delta t^2 (62m/5)}$</td>
<td>$\cos(3m\pi/5)$</td>
<td>$e^{j2\Delta t^2 \cot \frac{z}{2}(2m\pi/5)}$</td>
</tr>
<tr>
<td>$E_G$</td>
<td>$e^{j2\Delta t^2 \cot \frac{z}{2}(84m/5)}$</td>
<td>$\cos(4m\pi/5)$</td>
<td>$e^{j2\Delta t^2 \cot \frac{z}{2}(3m\pi/5)}$</td>
</tr>
<tr>
<td>$F_G$</td>
<td>$-\frac{1}{\sqrt{2}}$</td>
<td>$\cos(5m\pi/5)$</td>
<td>$-\frac{1}{\sqrt{2}}$</td>
</tr>
<tr>
<td>$H_G$</td>
<td>$A_{z}\frac{1}{\sqrt{2}}e^{j\cot \frac{z}{2}2m^2 \Delta t^2}$</td>
<td>$A_{z}\frac{1}{\sqrt{2}}e^{j\cot \frac{z}{2}2m^2 \Delta t^2}$</td>
<td>$A_{z}\frac{1}{\sqrt{2}}e^{j\cot \frac{z}{2}(2m\pi/5)}$</td>
</tr>
</tbody>
</table>
fractional transforms. It is evident that the number of coefficients, which are significant and required for representing the signal, is less in fractional transform domain. This feature of fractional transforms will enhance the performance of their sliding versions.

It is well known that each complex multiplication requires three real multiplications [11]. In the case of SDFRFT, for a real input sequence, multiplication of each term in the buffer with two complex numbers requires five real multiplications. The sum is multiplied by a chirp signal and the scaling factor,
$A_{fx}$, requiring another six real multiplications. These operations lead to a total of $5P+6$ real multiplications. In the case of SDFRCT and SDFRST, total number of real multiplications is $3P+10$ and $3P+3$, respectively. The number of computations may further be reduced to $2(P+1)$, $2(P+1)+1$, and $2P$, in the case of SDFRFT, SDFRCT and SDFRST, respectively, by pre-computation of multiplication factors, $A_G$ to $H_G$. In all the three sliding discrete fractional transforms, the time required for computing the transform is less due to the minimization of number of multiplications per coefficient. When a new fractional transform output is desired for every input sample, the sliding discrete fractional transform is computationally less intensive than the discrete fractional transform. The computational complexity per coefficient in SDFRFT is of the order of $5P$ and in SDFRCT and SDFRST it is of the order of $3P$. The computational complexity in computing all the $P$ coefficients in the case of discrete fractional transform is of the order of $3(2P+(P/2)\log_2 P)$ [4,9], which is more than $5P$.

The computation of all the $P$ sliding transform coefficients can be performed by parallel implementation of $P$ number of units shown in Fig. 1, with one unit for each of the coefficient. Such an arrangement requires $P$ times the number of real multipliers required per coefficient. Recently, because of the development of large-scale integrated circuits, emphasis on minimizing the number of multipliers has reduced. Many parallel devices are affordable rather than a few high-speed devices[12]. In the present method, the sliding discrete fractional transform coefficients are computed nearly at the rate of arrival of input samples.

In the case of sliding discrete fractional transform, after an initial waiting period for the arrival of $P$ input samples, the output is available at the rate of $T = \max(T_I, T_s)$. Here, $T_I$ is the time gap between the arrival of input samples and $T_s$ is the time taken for the computation of the sliding discrete fractional transform. In the case of discrete fractional transform, $P$ point discrete fractional transform is obtained after a time interval $T_D = T_P + PT_I$, where $T_P$ is the time taken for the computation of $P$ point discrete fractional transform. The term $PT_I$ represents the waiting time for the arrival of $P$ input samples. As the number of multiplications is less in the sliding discrete fractional transform, $T_s \ll T_P$ and hence $T \ll T_D$. The waiting time for the arrival of $P$ input samples can be overlapped with the computation of $P$ point discrete fractional transform, requiring $\max(T_P, PT_I)$ time to compute $P$ point discrete fractional transform. We find that the sliding discrete fractional transforms perform better compared to discrete fractional transforms as $T \ll \max(T_P, PT_I)$ in all the three cases.

The performance of any transform can be evaluated by considering number of bits required for representing a real value. If $b$ bits are required for a real value, in fixed-point arithmetic, every real multiplication results in a maximum of $2b$ bits. The product if rounded from $2b$ bits to $b$ bits leads to the quantization error. This quantization error has a variance given by [13,14]

$$\sigma_c^2 = \frac{2^{-2b}}{12}. \quad (16)$$

Since $5P + 6$ real multiplications are required in computing SDFRFT, the variance of quantization error is

$$\sigma_f^2 = \frac{2^{-2b}}{12}(5P + 6). \quad (17)$$

Similarly, in the computation of SDFRCT and SDFRST the variance of quantization error can be shown to be

$$\sigma_c^2 = \frac{2^{-2b}}{12}(3P + 10)$$
and
$$\sigma_s^2 = \frac{2^{-2b}}{12}(3P + 3), \quad (18)$$

Letting $2N = 2^n$ and $P \approx 2N$ for large $N$, in (17) and (18), we get

$$\sigma_f^2 = \frac{5}{3}2^{-2b+n-2}$$
and
$$\sigma_s^2 = \sigma_c^2 = 2^{-2b+n-2}. \quad (19)$$

The SNR is defined as [11]

$$\text{SNR}_G = 10\log_{10} \frac{\sigma_{G_Y}^2}{\sigma_{G_c}^2}, \quad G = f, c, s, \quad (20)$$

with $\sigma_{G_Y}^2$ being the variance of the coefficients of the sliding discrete fractional transforms. Table 2 lists the performance measures of the sliding discrete fractional transforms for a single coefficient. The number of multipliers required in the case of SDFRST is less in comparison with that in SDFRFT and SDFRCT.

The samples of audio signal $X_1$, from SQAM [15] shown in Fig. 4, are considered for verifying above results. The sliding discrete fractional transform coefficients can be computed with and without
restriction on the number of bits in each multiplication operation. The coefficients obtained are treated as practical values when computed with restriction on number of bits. The coefficients computed without restriction on the number of bits are treated as ideal values. The practical value of SNR is computed as the ratio of the sum of the square of practical values of coefficients to the sum of square of difference between the practical and ideal values of coefficients. The ideal value of SNR is shown as a function of number of bits in Fig. 5 for all the three sliding discrete fractional transforms with $P = 5$. The SNR in all the three sliding discrete fractional transforms is nearly the same. The practical values of SNR obtained for all the three sliding discrete fractional transforms and the SDFT are shown as a function of number of bits in Fig. 6. The SNR is 15, 10 and 30 dB for SDFRFT, SDFRCT and SDFRST, respectively, with 8 bits for representing the coefficients. By increasing the number of bits by twofold, SNR is 65, 58 and 78 dB, respectively, for SDFRFT, SDFRCT and SDFRST. With 8 bits for representing the coefficients, the ideal value of SNR is 45 dB and with 16 bits it is 93 dB. We find that SDFRST has better SNR in comparison with SDFRFT or SDFRCT. SDFT has higher SNR values in comparison with the sliding discrete fractional transforms. In the case of SDFT, with 8 bits for representing the coefficients, SNR is 36 dB and with 16 bits it is 85 dB. The sliding discrete fractional transforms require 10–13 bits for representing the coefficients with SNR in the range of 30–40 dB, whereas SDFT requires 7–9 bits. The difference in the number of bits may be attributed to the time–frequency information available in the coefficients of fractional transforms. However, in the case of SDFT, as more coefficients are required to represent the signal, more bits are required to
represent its coefficients. This is in contrast with the number of coefficients and number of bits required for representing the coefficients obtained in a fractional transform. Therefore, the fractional transforms may be preferred over the ordinary transforms.

5. Conclusions

The methods of implementing sliding discrete fractional transforms are presented. Their performances with regard to the computational complexity, variance of quantization error and SNR are compared. Because of reduced number of multiplications, the sliding discrete fractional transforms are faster and simpler to implement. They are useful for real-time practical applications when a particular time-frequency bin is to be analyzed. The computational complexity and the variance of quantization error are less in the case of the SDFRST. In comparison with the SDFT, the sliding discrete fractional transforms require less number of coefficients to represent the signal and so also the total number of bits.

Acknowledgment

The authors are thankful to S.V. Narasimhan, Digital Signal Processing and System Group, Aerospace Electronics and System Division, National Aerospace Laboratory, Bangalore, India, for valuable suggestions and discussions.

References

[15] MPEG web site at ⟨http://www.tnt.uni-hannover.de/project/mpeg/audio⟩.