elements. In these circumstances, the amplitudes of the elements of both sequences are converted to a unit magnitude before the coherence estimate is evaluated. In this section, a condition is given for the distribution of the coherence estimate to be independent of second-channel statistics in this situation. Interestingly, the independence does not rely on the notion of spherical symmetry.

Assume that the samples in sequence $a$ are statistically independent. Also, suppose $a$ and $b$ have the property that each sample has unit magnitude, i.e., $x_n^2 + y_n^2 = u_n^2 + v_n^2 = 1$ for $n = 1, \ldots, N$. Then they can be written as $a = (e^{i\phi_1}, \ldots, e^{i\phi_N})$ and $b = (e^{i\phi_1}, \ldots, e^{i\phi_N})$. The MSC estimate becomes $\gamma(a, b) = |\sum_{n=1}^{N} \exp(i(\theta_n - \phi_n))|^2/N^2$.

If each sample $e^{i\phi_n}$ of $a$ is uniformly distributed on the unit circle, then so is each term $\exp(i(\theta_n - \phi_n))$ in this expansion of $\gamma(a, b)$—regardless of the distribution of the samples of $b$. This is easily proven by convolution of an arbitrary probability density function $p(\phi_n)$ with a uniform density function, modulo $2\pi$. Furthermore, the terms $\exp(i(\theta_n - \phi_n))$ will be statistically independent of one another. Thus, in this case, the distribution of the MSC estimate is $1/N^2$ times the distribution of the length squared of a vector formed by adding $N$ unit vectors in a plane which have uniformly distributed direction. This problem has a long history as the problem of a random walk in the plane. The problem of determining this distribution was posed by Pearson in 1905 (see [7]) and its history is detailed in [8].

Note that if the elements of the $a$ sequence are allowed to have nonconstant amplitudes (i.e., $A_n = |x_n + iy_n|$) that are independently distributed, then invariance with respect to second-channel statistics is still maintained. The distribution of the coherence estimate is, in this case, related to a random walk with a random step size.

It is also important to observe that although each term of the $a$ sequence $A_n e^{i\phi_n}$ is spherically symmetric, the $a$ sequence itself is not spherically symmetric.

VII. CONCLUSIONS

We feel the important aspect of this correspondence is the demonstration of the utility of a geometric approach to this probability problem, not just for the results obtained but also for the insight into why these results come about. Although we have focused on the problem of coherence estimation with no signal present, we anticipate that the geometric approach will prove useful in the case when signal is present.

REFERENCES


FFT Pruning Applied to Time Domain Interpolation and Peak Localization

SVERRE HOLM

Abstract—The efficiency of the fast Fourier transform (FFT) algorithm may be increased by removing operations on input values which are zero, and on output values which are not required. This is applied to interpolation of complex and real valued time domain functions. For real functions, analytic signal concepts may be used to get the Hilbert transform as a byproduct, and applied to the cross correlation function this gives an efficient and accurate method for peak localization.

I. INTRODUCTION

Pruning of the fast Fourier transform (FFT) algorithm is the elimination of operations on zeros. When the number of input points is less than the number of transform points, the number of butterflies may be reduced. This is referred to as input pruning and was first described by Markel [1]. The efficiency and regularity of the algorithm was improved by Skinner [2] by pruning the decimation-in-time instead of the decimation-in-frequency algorithm. This algorithm requires $(N/2) \cdot \log_2 N$ complex multiplications where $N$ is the transform length and $N$ is the number of nonzero input values, as compared to the FFT's $(N/2) \cdot \log_2 N$ complex multiplications.

Sreenivas and Rao [3] extended the algorithm by combining input and output pruning. This applies to the situations where both the number of input points and the number of output points are less than the transform length. By viewing the decimation-in-frequency algorithm as the transpose of the decimation-in-time algorithm, Markel's pruning may be applied to the output of Skinner's algorithm, or vice versa. Furthermore, they generalized the algorithm by allowing output pruning anywhere in the output range [5]. Nagai [7] gives an alternative algorithm for generalized output pruning. It has a more regular structure than in [5], while keeping the savings in the computation. In [4] and [6], pruning algorithms are expressed using matrix formulation.

The most common application of input pruning is interpolation in the frequency domain. By appending zeros to a sequence prior to Fourier transformation, a high resolution spectrum is obtained. This is used, for instance, in autoregressive spectral analysis. The input and output pruned algorithm gives additional savings by computing the high resolution spectrum in only a preselected narrow frequency band. In [5], the amount of computation is compared to the direct DFT, unpruned FFT, and chirp Z transform methods. In this correspondence, the application is efficient time domain interpolation. It will be shown that with some modifications, the available pruning algorithms may be applied. It will also be shown that a byproduct for real valued time samples is the Hilbert transform. It has applications in estimation of time delay found by localization of the peak of the cross correlation function.

II. TIME DOMAIN INTERPOLATION

Consider a time sequence $x(n)$ with $N$ samples ($n = 0, \ldots, N - 1$), and a new interpolating sequence $x(n)$ with $rN$ samples, where $r$ is a positive integer. This new sequence should have the property that $x(n) = x(n)$ for $n = 0, \ldots, N - 1$. This requirement distinguishes interpolation by FFT pruning from inter-
polation with a low-pass filter [8]. The requirement is equivalent to increasing the sampling frequency by a factor, and corresponds to adding zeros between the old and new Nyquist frequency. Let \( X'(k) \) be the \( N \) point discrete Fourier transform of \( x(n) \), and \( X(k) \) the \( rN \) point discrete Fourier transform of \( x(n) \). Then the new \( rN \) frequency samples \( X(k) \) should be found from the \( N \) frequency samples \( X'(k) \) via

\[
X(k) = \begin{cases} 
X'(k), & k = 0, \cdots, N/2 \\
0, & \text{other values of } k \\
X'(k + N - rN), & k = N/2 + 1, \cdots, rN - N/2 \\
X'(k + N), & k = rN - N/2 + 1, \cdots, rN - 1.
\end{cases}
\] (1)

The pruned FFT of [2] cannot be applied directly to inverse transform \( X(k) \) since the zeros are in the middle of the sequence instead of at the end. However, Nagai’s decimation-in-time algorithm [7] may be transposed to handle these input data. The transposed algorithm is a decimation-in-frequency algorithm with frequency shift at the input, when operated as an inverse Fourier transform. A Fortran implementation of this algorithm may be obtained from the author. Note that Sreenivas and Rao’s generalized output pruning algorithm [5] is not so simple to transpose into an algorithm that will handle the input data of (1), since the nonzero input samples are in two separate bands. The transpose of Nagai’s algorithm, however, will handle this because of the cyclic property of the frequency shift operation.

In many cases, the time domain signal to be interpolated is real. In this case, the information in \( X(k) \) is fully contained in the Fourier transform of the analytic signal

\[ S(k) = \begin{cases} 
X'(0), & k = 0 \\
2X'(k), & k = 1, \cdots, N/2 - 1 \\
X'(N/2), & k = N/2 \\
0, & k = N/2 + 1, \cdots, rN - 1.
\end{cases} \] (2)

\( S(k) \) can be directly inverse transformed by the pruned FFT, and the result will be a complex sequence \( s(n) \), with the desired sequence \( x(n) \) as the real part, and its Hilbert transform as the imaginary part. The magnitude of \( x(n) \) is the envelope of the analytic signal.

This was also used by Markel [1] for cepstral smoothing. In this application, one originally has \( rN \) values of the cepstrum, but reduces it to \( N \) by applying a cepstral window that zeros out the samples in the middle of the sequence. Via (2) and the pruned forward FFT, this gives a smoothed log spectral estimate. A difference from time domain interpolation is that reduction instead of increase of resolution is the aim. Another difference is that the imaginary part of the result (the Hilbert transform) is of no use.

In time domain interpolation, (2) and the pruned FFT have the following advantages. Better accuracy than the often-used parabolic interpolation [9] which gives a biased estimate of the peak’s location; and simpler and more efficient implementation than the straightforward implementation in [10, sect. V]. Here the interpolating samples are found by using the time shift property of the Fourier transform. The cross spectrum is multiplied by a linear phase shift, and inverse transformed by the short, \( N \) point, discrete Fourier transform. This is done a total of \( r \) times. The results are interleaved to get the interpolated cross correlation.

A third advantage of the present method is the availability of the Hilbert transform of the cross correlation. Since it passes through zero at the lag where the correlation has a peak, it is also a sensitive indicator of the peak’s location. Cabot [11] demonstrates this property by giving examples of typical correlation functions and their Hilbert transforms. Bendat [12] also shows that the accuracy in determining the exact zero crossing of the Hilbert transform is often greater than the accuracy in determining the location of the peak of the correlation. A further advantage for bandpass signals is that the envelope is free from the influence of the oscillations caused by the center frequency of the frequency band. Peak localization in the envelope of the cross correlation function is therefore a good method for determining the delay of narrow-band signals.

### III. TIME DOMAIN INTERPOLATION IN A LIMITED REGION

For peak finding it should not really be necessary to interpolate the entire cross correlation function, but only the region around the peak. With some prior knowledge on the peak’s location, both input and output pruning could be applied. This prior knowledge could be obtained from a tracking algorithm which is often a part of a time delay estimation system, or it could come from a preliminary, low resolution cross correlation estimate. By using either the algorithm of Sreenivas and Rao [5], or Nagai’s algorithm [7] with input pruning [2] and equation (2), a high resolution, limited region, correlation estimate and its Hilbert transform may be obtained.

### IV. CONCLUSION

The pruned FFT has applications in frequency and time domain interpolation and smoothing. In this correspondence, the frequency domain interpolation methods have been extended to time domain interpolation. A byproduct for real signals is the Hilbert transform which aids in accurate peak localization.

### ACKNOWLEDGMENT

The author would like to thank one of the reviewers and Dr. J. Endresen for helpful suggestions and comments.

### REFERENCES


Comments on “Two-Dimensional Interpolation by Generalized Spline Filters Based on Partial Differential Equation Image Models”

N. B. Karayiannis and A. N. Venetsanopoulos

Abstract—The generalized spline formula corresponding to the separable semicausal Partial Differential Equation (PDE) image model is given in its correct form. In addition, the correct formula for the second-order B-spline is derived. Finally, it is shown that there is no reason to consider separable PDE image models only, at least when describing the concept of generalized splines.

In this correspondence we introduce corrections in some formulas of the above paper which, regardless of their origin, may cause problems in the understanding of the paper. For the convenience of future readers, we identify the errors first and subsequently provide the correct formulas.

The generalized spline formula for the semicausal PDE model, as given in the paper by (6), should be corrected. In this case

\[ L_z(D_z) L_z(D_y) s_z(z) = 0, \quad z = x, y \]

and \( L^* \) is the formal adjoint operator of \( L \).

The function \( s_z(\cdot) \) is correctly given by (18) of the paper.

However, \( s_z(\cdot) \), as given in (18), is incorrect. The adjoint operator for \( L_z(\cdot) \) is

\[ L_z^*(D_z) = L_z(D_z) = D_z^2 + \alpha_z^2. \]

Therefore, \( s_z(\cdot) \) is the general solution of

\[ (D_z^2 + \alpha_z^2)s_z(x) = 0. \]

That is,

\[ s_z(x) = k_1 \cos \alpha_z x + k_2 \sin \alpha_z x + k_3 x \cos \alpha_z x + k_4 x \sin \alpha_z x. \]

In addition, the second-order B-spline given in (31) of the paper is also incorrect. The first-order B-spline corresponding to the operator

\[ L(D_z) = D_z + \alpha \]

is given by (30), i.e.,

\[ B_z(\alpha, -\alpha) = \frac{1}{(e^{2\alpha R} - 1)} \left[ \exp \left[ \alpha(x + 2R) \right] - e^{-\alpha x} \right] \]

and

\[ B_z^*(\alpha, -\alpha) = \frac{2e^{\alpha R}}{e^{2\alpha R} - 1} \sinh \left[ \alpha(x + R) \right] \]

The second-order B-spline corresponding to the operator

\[ L(D_z) = D_z^2 - \alpha^2 = (D_z + \alpha)(D_z - \alpha) \]

is obtained by convolving \( B_z(\alpha, \alpha) \) with \( B_z(\alpha, -\alpha) \).

In the paper, the second-order B-spline is given by (31), i.e.,

\[ B_z(\alpha, \alpha, -\alpha) = \frac{1}{(e^{2\alpha R} - 1)} \left[ \exp \left[ \alpha(4R + x) \right] + e^{-\alpha x} \right] \]

\[ + \frac{1}{(1/\alpha)} \left( e^{-\alpha x} - \exp \left[ \alpha(4R + x) \right] \right) \]

and

\[ B_z^*(\alpha, \alpha, -\alpha) = \frac{1}{(e^{2\alpha R} - 1)} \left[ (1/\alpha) \exp \left[ \alpha(4R + x) \right] \right] \]

\[ + 2 \exp \left[ \alpha(2R - x) \right] \]

\[ - 2 \exp \left[ \alpha(2R + x) \right] \]

\[ - e^{-\alpha x} - (x + h) 2 \exp \left[ \alpha(2R + x) \right] \]

\[ + 2 \exp \left[ \alpha(2R - x) \right] \]

\[ - x \exp \left[ \alpha(4R + x) \right] + e^{-\alpha x} \right] \]

The second-order B-spline should be a continuous function of \( x \in [-2R, 2R] \) and, therefore, be continuous at point \( x = -R \). Hence, \( B_z(\cdot) \) should satisfy the equation

\[ B_z(\alpha, -\alpha) \bigg|_{x = -R} = B_z^*(\alpha, \alpha, -\alpha) \bigg|_{x = -R}. \]

It can be easily verified that (2) is not satisfied by (31) regardless of what constant \( h \), undefined in the paper, is.

In the sequel, the correct expression of \( B_z(\cdot) \) is obtained. For \( x \in [-2R, -R] \),

\[ B_z(\alpha, \alpha, -\alpha) = \frac{2e^{2\alpha R}}{(e^{2\alpha R} - 1)} \left[ (x + 2R) \cosh \alpha(x + 2R) \right] \]

\[ - (1/\alpha) \sinh \alpha(x + 2R) \right] \]

After some algebra

\[ B_z(\alpha, \alpha, -\alpha) = \frac{2e^{2\alpha R}}{(e^{2\alpha R} - 1)} \left[ (x + 2R) \cosh \alpha(x + 2R) \right] \]

\[ - (1/\alpha) \sinh \alpha(x + 2R) \right]. \]
subroutine prninp(m,zr,isign,kinit,kband)

Algorithm that takes the input array 'zr' of size N=2**m
with 'kband' non-zero values starting from index 'kinit'
and forward ('isign = 1') or inverse ('isign = -1') Fourier transforms it.
The N output samples are found in 'zr'

Number of complex multiplications is (N/2)log(2) kband
rather than (N/2)log(2) N

Written by:
Sverre Holm, Informasjonskontroll a.s, Asker, NORWAY
ref:
S. Holm, "FFT Pruning Applied to Time Domain Interpolation and

Transposed version of:
K. Nagai, "Pruning the Decimatation-in-Time FFT Algorithm with Frequency

2**m   : total number of data
zr()   : input and output data; (complex)
isign  :  1 for forward transform; exp(-jG)
         -1 for inverse transform; exp(+jG)
kinit  : index of the lowest non-zero input sample
kband  : number of non-zero input samples

c  complex zr(1), cdata
n = 2**m
nrep = n
arg0 = 8.*atan(1.0)
if (isign.eq.1) arg0 = -arg0

do 110 i = 1, m
   nbtf = nrep/2
   arg = arg0/float(nrep)
   
   -- pruning --
   do 100 j = 1, min0(nbtf,kband)
            
   -- frequency shift --
   twf = arg*float(j-1+kinit)
   c = cos(twf)
   s = sin(twf)
   do 100 k = j,n,nrep
       j2 = k+nbtf
       cdata = zr(k)+zr(j2)
       zr(j2) = (zr(k)-zr(j2))*cmplx(c,s)
       zr(k) = cdata
   100    continue
   nrep=nbtf
110    continue
   if (isign.eq.-1) then
do 200 j = 1, n
   zr(j) = zr(j)/n
200    continue
endif
   call revbits(zr,m,n)
return
end

c *********************************************************************
c
subroutine revbits(data,mexp,n)
c
subroutine for in-place bit-reversal

c  complex data(1), temp
c
c from IEEE Prog's for DSP ch 1.1
c
c This section puts data in bit-reversed order
c
  j = 1
  do 80 i = 1, n

  At this point, i and j are a bit reversed pair (except for the
c displacement of +1)
  if (i-j) 30, 40, 40

  Exchange data(i) with data(j) if i.lt.j
  temp = data(j)
  data(j) = data(i)
  data(i) = temp

  implement j=j+1, bit-reversed counter
  m = n/2
  if (j=m) 70, 70, 60
  j = j - m
  m = (m+1)/2
  go to 50
  j = j + m
80    continue
   return
end

c **********************************************************************
c