A NEW, EFFICIENT STRUCTURE FOR THE SHORT-TIME FOURIER TRANSFORM, WITH AN APPLICATION IN CODE-DIVISION SONAR IMAGING

Michele Covell, Acoustics & Radar Technology Laboratory, SRI International, Menlo Park, CA 94025

John Richardson, Digital Signal Processing Group, Massachusetts Institute of Technology, Cambridge, MA 02138

ABSTRACT

Although most applications which use the short-time Fourier transform (STFT) temporally downsample the output, some applications exploit a dense temporal sampling of the STFT. One example which will be discussed is code-division, multiple-beam sonar. Given a need for the densely-sampled STFT, the complexity of the computation can be reduced from $\mathcal{O}(N \log N)$ for the general short-time FFT structure to $\mathcal{O}(N)$ using the Goertzel algorithm. This paper introduces the pruned short-time FFT, a new computational structure for efficiently computing the STFT with dense temporal sampling. The pruned FFT achieves the same computational savings as the Goertzel algorithm and, unlike the Goertzel algorithm, is unconditionally stable.

I. INTRODUCTION

Although most applications which use the short-time Fourier transform (STFT) temporally downsample its output, some applications exploit a dense sampling of the time evolution of the STFT. Examples include time-scale modification of speech using the short-time spectral magnitude [1] and code-division multiple-beam sonar [2,3]. Given a need for the STFT at each point, the complexity of the computation can be reduced from $\mathcal{O}(N \log N)$ for the general short-time FFT to $\mathcal{O}(N)$ using the recursive formulation given by the Goertzel algorithm [4]. However, because this formulation is recursive, there are difficulties due to quantization effects. In particular, the variance of the computational noise grows linearly with time.

This paper introduces the pruned FFT, a new computational structure for efficiently computing the short-time Fourier transform with dense temporal sampling. This implementation of the STFT achieves the same computational savings as the Goertzel algorithm without relying on recursive computation. Since this structure does not rely on pole/zero cancellation, there is no instability in the computation.

Code-division, multiple-beam sonar is discussed in the next section as an example of an application which exploits a dense temporal sampling of the STFT. Section III outlines two classic implementations of the STFT: the short-time FFT structure and the Goertzel algorithm. Section IV introduces the pruned, short-time FFT, a new computational structure for efficiently computing the STFT. In Section V, the variance of the computational noise in the Goertzel algorithm is shown to grow linearly with time, while that of the pruned FFT is time-invariant.

II. CODE-DIVISION, MULTIPLE-BEAM SONAR

Conventional sonar imaging systems achieve resolution either through the use of a single, swept beam or through the use of multi-element arrays. These techniques, while highly successful, present some inherent difficulties. In the case of the single swept beam, the time required to scan through the desired aperture can result in the failure to detect transients. When multi-element arrays are used, the hardware requirements necessary to achieve high resolution can result in a large, costly system. An alternative approach is multiple-beam, code-division sonar [2,3].

Multiple-beam, code-division sonar provides resolution in azimuth and elevation using multiple, coded signals and resolution in range by the characteristics of the autocorrelation of each of the signals. N transducers each illuminate a different direction and each transmit a distinct signal, S_i for $i = 1, ..., N$. A single wide-beam hydrophone acts as a receiver. Multiple-hypothesis testing is then used to detect and discriminate the returns from the separate beams. Frequency-shift keyed (FSK) codes are used to provide good spatial resolution. FSK codes are made up of individual, uniformly-spaced frequency bursts (commonly referred to frequency chips). Each code within a set of N FSK codes transmits the N frequency chips in a unique order and, thus, is largely uncorrelated with the other codes in the set. The high signal-to-signal rejection of these codes provides good azimuth and elevation separation and their sharply peaked autocorrelation functions provide good range separation.

Upon reception of the reflected waveform, each signal is separately detected and the results are inverted to yield an estimate of the spatial location of the scattering centers in three dimensions. The minimum mean-square error receiver for these FSK codes is shown in Figure 1. In-phase and quadrature samples are taken of the received signal after demodulation by the carrier frequency, f_c. Matched filters are used to detect the individual frequency chips, $w(t)e^{j\frac{2\pi}{T}kt}$. The final section of the receiver combines the outputs from

This work was supported in part by: the Defense Advanced Research Projects Agency monitored by the Office of Naval Research under Grant No. N00014-89-J-1489; the National Science Foundation under Grant No. MIP 87-14969; Lockheed Sanders, Inc.; a National Science Foundation Graduate Fellowship; and SRI International. We wish to acknowledge the MIT-WHOI Joint Program, the Woods Hole Oceanographic Institution and the Scripps Institution of Oceanography, as well as the sponsorship of Mr. Richardson's participation by the U.S. Navy.

CH2977-7/91/0000-2041 $1.00 © 1991 IEEE

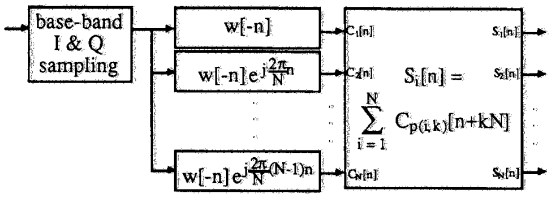


Figure 1: The minimum mean-square error detectors for a set of N FSK codes

these matched filters to form estimates for the backscattering of each of the FSK waveforms.

The receiver shown in Figure 1 uses the outputs from the frequency-chip matched filter bank at each point in time. This dense temporal sampling of the outputs avoids the highly non-linear problem of simultaneous estimation of the waveform identity and its time delay. Since the digitized frequency chips will be $w(\frac{T}{N}n)e^{j\frac{2\pi}{N}kn}$ for $k = 0, ..., N-1$, the bank of matched filters can be implemented as a modulated filter bank. The remainder of this paper discusses alternative implementations of modulated filter banks with dense temporal sampling. It is the efficiency of these implementations which makes FSK-code sonar imaging practical.

III. CLASSIC IMPLEMENTATIONS OF MODULATED FILTER BANKS

This section discusses two well known alternate implementations of modulated filter banks.

The Short-Time Fourier Transform

One well-known implementation of a modulated filter bank is the short-time Fourier transform (STFT). The output of the filter bank, $y_k[t]$, is given by

$$\begin{aligned} y_k[t] &= x[t] * (w(\frac{T}{N}t)e^{j\frac{2\pi}{N}kt}) = \sum_{n=-N+1}^{0} x[t-n]w(\frac{T}{N}n)e^{j\frac{2\pi}{N}kn} \\ &= \sum_{n=0}^{N-1} \left(x[t+n]w(-\frac{T}{N}n) \right) e^{-j\frac{2\pi}{N}kn} \end{aligned} \quad (1)$$

By defining $v_t[n] = x[t+n]w(-\frac{T}{N}n)$, Equation 1 can be seen to be the N-point discrete Fourier transform (DFT) of $v_t[n]$. Formulating the matched filters as a series of DFT's allows the use of the FFT. This approach requires $\left(\frac{N}{2} - 1\right)(\log_2 N - 1)$ multiplications and $N \log_2 N$ additions per time sample t.

The Goertzel Algorithm

Another implementation of a modulated filter bank can be derived by considering Equation 1. Define

$$x_t[n] = \begin{cases} x[t+n] & 0 \le n < N \\ 0 & otherwise \end{cases}$$

and $X_t[k]$ and $W[k]$ as the N-point, one-dimensional DFT's of $x_t[n]$ and $w(n\frac{T}{N})$, respectively. Then, $y_k[t] = X_t[k] * W^*[k]$. This approach imposes the shaping provided by $w(t)$ through frequency-domain convolution. The advantage of shaping $x_t[n]$ after computing the DFT is that the Goertzel algorithm can then be used to compute $X_t[k]$. In particular,

$$X_t[k] = e^{j\frac{2\pi}{N}k} \left(X_{t-1}[k] + x[t+N-1] - x[t-1] \right)$$

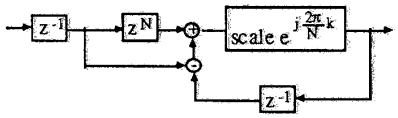


Figure 2: The Goertzel algorithm for the k^{th} channel of an N-point STFT

As shown by Figure 2 and the above equation, the Goertzel algorithm computes each short-time spectral sample independently, using an IIR filter with a pole at $e^{j\frac{2\pi}{N}k}$ and an FIR section with zeros at $e^{j\frac{2\pi}{N}m}$ for $m = 0, ..., N-1$. This recursive computation of $X_t[k]$ requires N multiplications and $2N$ additions per time sample t. The total computational cost of computing $y_k[t]$ depends on the form of $W[k]$. If $W[k]$ has only a few non-zero samples, the additional computational cost imposed by shaping may be low and the total cost of computing $y_k[t]$ will also be $O(N)$.

A disadvantage of the Goertzel algorithm is that it is marginally unstable: the reliance on pole/zero cancellation on the z-domain unit circle introduces this instability. This difficulty is discussed in Section V.

IV. A NEW STFT IMPLEMENTATION: THE PRUNED FFT

An innovative implementation of the STFT with dense temporal sampling, the pruned FFT, was recently derived [5].[1] Like the Goertzel algorithm, the pruned FFT computes the DFT of $x_t[n]$ and imposes the shaping provided by $w(t)$ through frequency-domain convolution. Figure 3 shows the use of the pruned FFT for an 8-point STFT. As can be seen from this figure, the pruned FFT has the same underlying structure as the classic FFT. The difference lies in the number of butterflies that are computed at each stage. The pruned FFT has only one butterfly in the first stage, two in the second, four in the third, eight in the fourth and so on, while the classic FFT has $\frac{N}{2}$ butterflies in each stage. Mathematically, the pruned FFT can be described as

$$\begin{aligned} X_t[k] &= X_t[k, \nu - 1] \quad \text{where } \nu = \log_2 N \\ &\quad \text{for } l = 0, ..., \nu - 1 \text{ and } k = 0, ..., 2^l - 1 \\ X_t[k, l] &= X_t[k, l-1] + e^{-j\frac{\pi}{2^l}k} X_{t-\frac{N}{2^{l+1}}}[k, l-1] \\ X_t[k + 2^l, l] &= X_t[k, l-1] - e^{-j\frac{\pi}{2^l}k} X_{t-\frac{N}{2^{l+1}}}[k, l-1] \\ X_t[0, -1] &= x[t] \end{aligned}$$

[1] The pruned FFT was originally derived by ADE, a signal-processing environment for computer-aided algorithm design [5]. An interesting mathematical description of the pruned FFT using generalized Kronecker products was recently given in [6]. This mathematical framework is particularly interesting because it provides a systematic way to derive fast, unitary transforms.

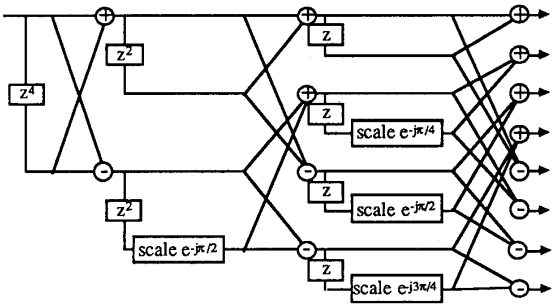


Figure 3: The pruned FFT implementation of the 8-point, rectangular-window STFT .

This "pruning" of butterflies reduces the order of the computational complexity from $\mathcal{O}(N \log N)$ for the classic FFT implementation to $\mathcal{O}(N)$ for the pruned FFT implementation. The amount of computation which is required for the pruned FFT is actually slightly less than that of the Goertzel algorithm at $N - \log_2 N - 1$ multiplications and $2N - 2$ additions per output sample.

Thus, the pruned FFT achieves the same computational savings as the Goertzel algorithm. The pruned FFT structure has the added advantage of being numerically stable while the Goertzel algorithm is marginally unstable.

V. QUANTIZATION EFFECTS

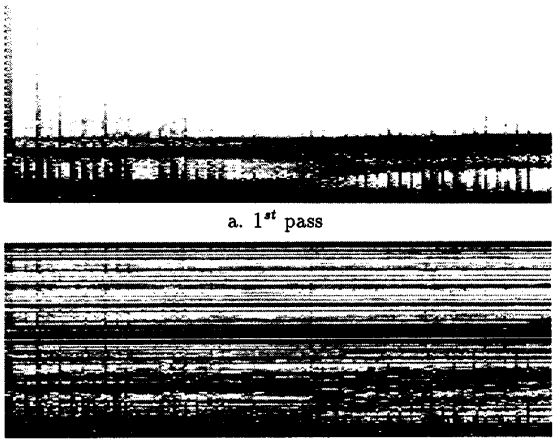
This section provides analytic expressions describing the output error and its variance for the Goertzel algorithm and for the pruned FFT.

The Goertzel Algorithm

The error in the output from the Goertzel algorithm depends on the error in the addition operation, the error in the value stored as the twiddle factor and the error in the multiplication operation. Let $\epsilon_{twdl}[k]$, $\epsilon_{mul}[t,k]$ and $\epsilon_{add}[t,k]$ respectively represent the errors in the twiddle factor, $e^{j\frac{2\pi}{N}k}$; the multiplication; and the summation. Then, $\epsilon_{tot}[t,k]$, the total error in the output from the Goertzel algorithm as a function of time, is

$$\begin{aligned}\epsilon_{tot}[t,k] &= \left(e^{j\frac{2\pi}{N}k} + \epsilon_{twdl}[k]\right)\epsilon_{tot}[t-1,k] + \epsilon_{mul}[t,k] \\ &\quad + \epsilon_{twdl}[k]e^{-j\frac{2\pi}{N}k}X_t[k] + \left(e^{j\frac{2\pi}{N}k} + \epsilon_{twdl}[k]\right)\epsilon_{add}[t,k]\end{aligned}$$

To determine the variance of the output error, assume that errors in the summation, the twiddle and the multiplication are all zero mean and uncorrelated and that the summation and multiplication errors are white. Let $\sigma^2_{twdl}[k]$, $\sigma^2_{mul}[k]$ and $\sigma^2_{add}[k]$ respectively represent the variances of the twiddle error ($\epsilon_{twdl}[k]$), the multiplication error ($\epsilon_{mul}[t,k]$) and the summation error ($\epsilon_{add}[t,k]$). Then, $\sigma^2_{tot}[t,k]$, the variance of the error in the output from the Goertzel algorithm as a function of time, is



a. 1^{st} pass

a. 1000^{th} pass

Figure 4: The output from the Goertzel algorithm on the first and the 1000^{th} loop through a single speech sequence

$$\begin{aligned}\sigma^2_{tot}[t,k] &= \left(1 + \sigma^2_{twdl}[k]\right)\sigma^2_{tot}[t-1,k] + \sigma^2_{mul}[k] \\ &\quad + \sigma^2_{twdl}[k]|X_t[k]|^2 + \left(1 + \sigma^2_{twdl}[k]\right)\sigma^2_{add}[k]\end{aligned}$$

The variance of error in the output from the Goertzel algorithm grows linearly with time. To see this, first assume that there is no twiddle error or $\sigma^2_{twdl}[k] = 0$. Then, the variance of the output error at time t is the variance of the output error at time $t-1$ plus some non-negative terms (in this case, the variances of the multiplication and addition errors). When the variance in the twiddle error is non-zero, the variance of the output error is the previous output-error variance, scaled by a number greater than one, plus some non-negative terms. Thus, the variance of the output error grows linearly.

To show the effect of this growing variance, a short section of speech was looped through the Goertzel algorithm a thousand times. That is, the speech sequence was replicated thousand times and this new repetitive sequence was put through the Goertzel algorithm as one long data stream. All of the quantities within the Goertzel algorithm were represented using a 24-bit mantissa: the quantization noise within this simulation would be similar to that seen in a floating-point DSP chip using a 24-bit mantissa. Figures 4-a and 4-b show the outputs from the first and the 1000^{th} loops through the speech sequence, respectively. As can be seen from this example and from the above equation, the computational error within the Goertzel algorithm is increasing across time.

The Pruned FFT

The error in the output from the pruned FFT depends on the error in each of the FFT stages. The error in the output from each stage depends on the error in the value stored as the twiddle factor, the error in the multiplication operation and the error in the addition or subtraction operation. Let

$\epsilon_{twdl}[k]$ represent the error in the twiddle factor and, within the l'th stage and the k'th frequency channel, let $\epsilon_{tot}[t,k,l]$, $\epsilon_{mul}[t,k,l]$ and $\epsilon_{add}[t,k,l]$ respectively represent the errors in the output of that stage, the multiplication and the addition. Then $\epsilon_{tot}[t,k]$, the total error in the output from the pruned FFT as a function of time, is

$$\begin{aligned}
\epsilon_{tot}[t,k] &= \epsilon_{tot}[t,k,\nu-1] \quad \text{where } \nu = \log_2 N \\
&\quad \text{for } l = 0,\ldots,\nu-1 \text{ and } k = 0,\ldots,2^l-1 \\
\epsilon_{tot}[t,k,l] &= \epsilon_{tot}[t,k,l-1] + \epsilon_{add}[t,k,l] \\
&\quad + \left(e^{-j\frac{\pi}{2^l}k} + \epsilon_{twdl}[k]\right)\epsilon_{tot}[t-\frac{N}{2^{l+1}},k,l-1] \\
&\quad + \epsilon_{twdl}[k]X_{t-\frac{N}{2^{l+1}}}[k,l] + \epsilon_{mul}[t,k,l] \\
\epsilon_{tot}[t,k+2^l,l] &= \epsilon_{tot}[t,k,l-1] + \epsilon_{add}[t,k+2^l,l] \\
&\quad - \left(e^{-j\frac{\pi}{2^l}k} + \epsilon_{twdl}[k]\right)\epsilon_{tot}[t-\frac{N}{2^{l+1}},k,l-1] \\
&\quad - \epsilon_{twdl}[k]X_{t-\frac{N}{2^{l+1}}}[k,l] - \epsilon_{mul}[t,k,l] \\
\epsilon_{tot}[t,0,-1] &= 0
\end{aligned}$$

To determine the variance of the output error, again assume that errors in the twiddle factor, the multiplication, and the addition are all zero mean and uncorrelated and that the multiplication and addition errors are white. Let $\sigma^2_{twdl}[k]$ represent the variance of the twiddle error ($\epsilon_{twdl}[k]$) and, within the l'th stage and the k'th frequency channel, let $\sigma^2_{tot}[t,k,l]$, $\sigma^2_{add}[k,l]$ and $\sigma^2_{mul}[k,l]$ respectively represent the variances of the output error for that stage ($\epsilon_{tot}[t,k,l]$), the addition error ($\epsilon_{add}[t,k,l]$) and the multiplication error ($\epsilon_{mul}[t,k,l]$). Then, $\sigma^2_{tot}[t,k]$, the variance of the error in the output from the pruned FFT as a function of time, is

$$\begin{aligned}
\sigma^2_{tot}[t,k] &= \sigma^2_{tot}[t,k,\nu-1] \\
&\quad \text{for } l = 0,\ldots,\nu-1 \text{ and } k = 0,\ldots,2^l-1 \\
\sigma^2_{tot}[t,k,l] &= \sigma^2_{tot}[t,k,l-1] + \sigma^2_{add}[k,l] \\
&\quad + \left(1+\sigma^2_{twdl}[k]\right)\sigma^2_{tot}[t-\frac{N}{2^{l+1}},k,l-1] \\
&\quad + \sigma^2_{twdl}[k]|X_{t-\frac{N}{2^{l+1}}}[k,l]|^2 + \sigma^2_{mul}[k,l] \\
\sigma^2_{tot}[t,k+2^l,l] &= \sigma^2_{tot}[t,k,l-1] + \sigma^2_{add}[k+2^l,l] \\
&\quad + \left(1+\sigma^2_{twdl}[k]\right)\sigma^2_{tot}[t-\frac{N}{2^{l+1}},k,l-1] \\
&\quad + \sigma^2_{twdl}[k]|X_{t-\frac{N}{2^{l+1}}}[k,l]|^2 + \sigma^2_{mul}[k,l]
\end{aligned}$$

Thus, the variance of the error in the computation approximately doubles at each successive stage but it does not increase as a function of time. To show the stability of the computational error within the pruned FFT, the same speech sequence used to test the Goertzel algorithm was run through the pruned FFT a thousand times. As with the Goertzel algorithm, all of the quantities within the pruned FFT were represented using a 24-bit mantissa. The first and 1000^{th} sections of the output from the pruned FFT are both indistinguishable from Figure 4-a. As can be seen from this example and from the above equations, the computational error within the pruned FFT is stable across time.

VI. CONCLUSIONS

The conclusions which can be drawn from this development are that both the pruned FFT and the Goertzel algorithm are computationally efficient implementations of the STFT, under the condition of dense temporal sampling, and that the pruned FFT has the advantage of stability with respect to computational noise. It is interesting to note that the topologies of the Goertzel algorithm and the pruned FFT can both be used to describe fast algorithms for a running sort with N as the number of data points to be sorted [7]. The Goertzel algorithm is transformed into a fast running sort by removing the multiplication operation and by replacing the addition operation by a merge/sort and the subtraction operation by a data remove. The pruned FFT can be transformed into a fast running sort by removing the multiplication operation and replacing both the addition and subtraction operations by a merge/sort. For the "pruned sort", all of the other butterfly outputs within a single stage degenerate into the same operation.[2]

ACKNOWLEDGEMENTS

Special thanks go to Bruce Musicus for pointing out References [6] and [7] and to Malcolm Slaney both for his help in creating Figure 4 and for his ideas for improving this article.

[1] S. H. Nawab, *Signal Estimation from Short-time Spectral Magnitude*. R.L.E. Technical Report 494, Massachusetts Institute of Technology, Cambridge, MA, 1982.

[2] J.S. Jaffe and J.M. Richardson, "Code-Division, Multiple-Beam Imaging," In *Proceedings OCEANS '89*, pp. 1015–1020, 1989.

[3] J.M. Richardson, *A Code-Division Multiple-Beam Sonar Imaging System*, Electrical Engineer's thesis, Massachusetts Institute of Technology, Cambridge MA, 1989.

[4] A.V. Oppenheim and R.W. Schafer, *Discrete-time Signal Processing* (pp. 585–587), Prentice-Hall, Inc., New Jersey, 1989.

[5] M.M. Covell, *An Algorithm Design Environment for Signal Processing*, R.L.E. Technical Report 549, Massachusetts Institute of Technology, Cambridge MA, 1989.

[6] P.A. Regalia and S.K. Mitra, "Kronecker Products, Unitary Matrices and Signal Processing Applications", *SIAM Review*, vol. 31, no. 4, pp. 586–613, December 1989.

[7] I. Pitas, "Fast Algorithms for Running Ordering and Max/Min Calculation", *IEEE Trans. on Circuits and Systems*, vol. 36, no. 6, pp. 795–804, June 1989.

[2] The "pruned sort" gotten from the pruned FFT is not quite the same as the running sort described in [7]. The sorting algorithm derived from the pruned FFT divides each sorting problem into two subproblems: sorting the even samples and sorting the odd samples. The sorting algorithm described in [7] divides each sorting problem into two different subproblems: sorting the first half and sorting the second half. This difference does not change the number of comparisons that are necessary. The change however does reduce the memory requirements from $O(N^2)$ for the algorithm described in [7] to $O(N \log N)$ for the algorithm derived from the pruned FFT.