A SCALABLE FREQUENCY DOMAIN-BASED LINEAR CONVOLUTION ARCHITECTURE FOR SPEECH ENHANCEMENT

Christelle Yemdji, Moctar I. Mossi, Nicholas Evans

EURECOM
Sophia-Antipolis, France
{yemdji, mossi, evans}@eurecom.fr

Christophe Beaugeant

Intel Mobile Communications
06560 Sophia-Antipolis, France
christophe.beaugeant@intel.com

ABSTRACT

Most speech enhancement algorithms consist of a time-varying filter which is applied to the signal in the frequency domain. One of the motivations for filtering in the frequency domain compared to convolution in the time domain is to reduce the computational complexity. Filtering in the frequency domain ,however, can introduce distortion if the linear convolution condition is not fulfilled. Although there are standard approaches to linear convolution in the frequency domain, they tend to be computationally prohibitive for small terminals. In this paper, we present a new and more efficient approach. The proposed approach is derived from the equivalence of zero-padding and interpolation in time and frequency domains. A distinct advantage of the approach proposed in this paper relates to its scalability which is exploited to manage computational complexity with only moderate degradation in speech quality.

Index Terms— Linear convolution, circular convolution, speech enhancement, speech distortion

1. INTRODUCTION

Speech quality in telecommunication terminals is often degraded by artifacts such as noise, echo or non-linearities [1]. It is therefore common to use speech processing algorithms to improve speech quality. This is generally performed with a time-varying filter which is applied in either the time domain or the frequency domain [2, 3]. In this paper, we focus on frequency domain filtering.

Filtering in the frequency domain is advantageous for its simplicity as it consists in multiplying the discrete Fourier transform (DFT) bin values by a gain vector. However bin-by-bin multiplication is equivalent to circular convolution rather than linear convolution [4, 5] and signals processed this way suffer from time domain aliasing [5], which results in speech quality degradation.

Attempts to reduce distortions coming from circular convolution commonly involve the use of overlapping frames, impulse response truncation, windowing and/or zero-padding [6]. For example with the overlap approach, the signal is processed in frames, where each frame is composed of a block of new samples to which are appended its preceding samples and/or zeros. This approach approximates the linear convolution constraint and is successful in reducing distortion.

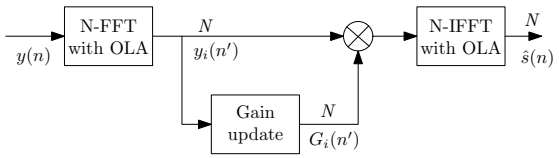
Existing approaches that achieve linear convolution in the frequency domain have been investigated mainly for use in frequency domain adaptive filters [4, 7] and are not suitable for real time systems in mobile or other small terminals since they are generally too computationally demanding.

A computationally efficient structure for linear convolution in the frequency domain is proposed by Marin-Hurtado *et al.* in [6]. In this paper, we propose a new, alternative approach that exploits the correspondence of zero-padding in the time domain to interpolation in the frequency domain (and vice-versa). The algorithm obtained is equivalent to that in [6] but has the advantage that it can be scaled in order to reduce computational complexity without significant degradation in speech quality.

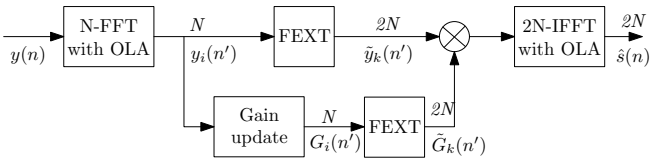
This paper is organized as follows. The next section reviews classic circular and linear convolution in the frequency domain. Section 3 reviews the algorithm proposed in [6] and introduces our new algorithm for linear convolution. In Section 3, we also show how low computational approaches can be derived from our algorithm. Section 4 presents our experimental work and results. Our conclusions are presented in Section 5.

2. BLOCK PROCESSING IN THE FREQUENCY DOMAIN

In this section, we describe the basic principles of circular and linear convolution in the frequency domain. In both cases, the input and output signals are converted from the time domain to the frequency domain (and vice-versa) through a fast Fourier transform (FFT) algorithm with an overlap add (OLA) method as illustrated in Figure 1.

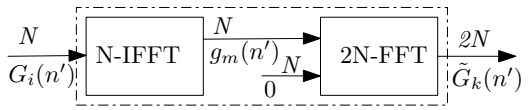


(a) Circular convolution

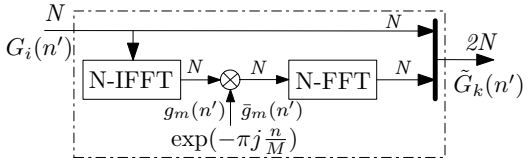


(b) Linear convolution

Fig. 1. Block processing in the frequency domain



(a) Frequency extension definition



(b) Efficient FEXT implementation

Fig. 2. FEXT implementations

2.1. Circular convolution

Figure 1(a) illustrates circular convolution which is the classical method used for filtering signals in the frequency domain. The filtered signal is obtained through a bin-by-bin multiplication of the N frequency domain components $y_i(n')$ with the gains $G_i(n')$ where n' is the frame index and i is the frequency bin number and ranges from 0 to $N-1$. As shown in Figure 1(a), filtering an N-point signal with an N-tap filter produces an N-point signal instead of an $(2N-1)$-point signal as would normally result from convolution in the time domain. This observation implies that the filtering of signals in the frequency domain as shown in Figure 1(a) introduces distortion in processed signals [5]. Distortions introduced by circular convolution result from time domain aliasing.

2.2. Linear convolution

Figure 1(b) depicts linear convolution in the frequency domain. Here, the frequency domain signals $y_i(n')$ and the gains $G_i(n')$ are processed through a frequency resolution extension (FEXT) block before the filtering operation. As illustrated in Figure 2(a), the FEXT block operates in two steps:

- The N frequency bins input signals $G_i(n')$ are converted into the time domain with an IFFT of length N (N-IFFT) to obtain $g_m(n')$ where m is the tap index of the impulse response and ranges from 0 to $N-1$.
- The time domain signal $g_m(n')$ is zero-padded with N zeros to obtain a signal of length $2N$ and reconverted into the frequency domain through a $2N$-FFT to obtain $\tilde{G}_k(n')$ with k being the "new" frequency bin and ranging from 0 to $2N-1$.

Returning Figure 1(b), the frequency resolution is now $2N$ instead of N and the filtered signal has $2N$ samples.

The merit of linear convolution over circular convolution is that it does not introduce distortion since it is equivalent to filtering in the time domain. Linear convolution, as described here, requires additional FFTs therefore its major disadvantage is its increased computational load.

3. FREQUENCY RESOLUTION EXTENSION (FEXT)

In practice linear convolution, as shown in Figure 1(b), is rarely used because of the additional computational load which prohibits its use in most mobile terminals. In this section, we focus on efficient implementations of the FEXT block. Section 3.1 reviews an existing approach to FEXT implementation [6] while in Section 3.2, we introduce our new approach. For brevity the frame index n' is omitted throughout the remainder of this paper (i.e. $\tilde{G}_k(n')$ becomes $\tilde{G}_k$).

3.1. Existing approach

This approach was introduced by Marin-Hurtado *et al.* in [6] and is illustrated in Figure 2(b). Here, the extended frequency bins $\tilde{G}_k$ are defined as follows:

$$\tilde{G}_k = \sum_{m=0}^{N-1} g_m \cdot \exp\left(-2\pi j \frac{mk}{2N}\right), \qquad (1)$$

where the summation terms between $m=N$ and $m=2N-1$ are omitted since $g_m = 0$ for this interval of m. By splitting Equation 1 into two for even and odd values of k, we obtain:

$$\tilde{G}_{k=2k'} = \sum_{m=0}^{N-1} g_m \cdot \exp\left(-2\pi j \frac{mk'}{N}\right) = G_{k/2} \qquad (2)$$

$$\tilde{G}_{k=2k'+1} = \sum_{m=0}^{N-1} g_m \cdot \exp\left(-2\pi j \frac{m(k'+1/2)}{N}\right), \qquad (3)$$

where k' is an integer ranging from 0 to $N-1$. Equation 2 shows that for even values of k, $\tilde{G}_k$ is equal to $G_{k/2}$. For odd values of k (Equation 3), $\tilde{G}_k$ is a discrete Fourier transform except for the "odd" exponential term. It is not specified in [6]

how Equation 3 can be implemented in a real time system. For this we suggest rewriting Equation 3 as:

$$\tilde{G}_{2k'+1} = \sum_{m=0}^{N-1} \bar{g}_m \cdot \exp\left(-2\pi j \frac{mk}{N}\right) \quad (4)$$
$$\text{where} \quad \bar{g}_m = g_m \cdot \exp\left(-2\pi j \frac{m}{2N}\right).$$

From Equation 4, it is apparent that $\tilde{G}_{2k'+1}$ can be computed through an FFT algorithm as shown in Figure 2(b).

3.2. Proposed approach

Our approach to FEXT computation is based on the fact that zero-padding in the time domain is equivalent to interpolation in the frequency domain and vice-versa. In the following we determine the direct relation between $\tilde{G}_k$ and G_i. For this, we use the FFT and IFFT definitions.

As the impulse response g_m is the IFFT of G_i Equation 1 can be rewritten as;

$$\tilde{G}_k = \sum_{m=0}^{N-1} \left[\frac{1}{N} \sum_{i=0}^{N-1} G_i \exp\left(2\pi j \frac{mi}{N}\right)\right] \exp\left(-2\pi j \frac{mk}{2N}\right)$$
$$= \frac{1}{N} \sum_{i=0}^{N-1} G_i \sum_{m=0}^{N-1} \left(\exp\left(\frac{2\pi}{2N} j\,(2i-k)\right)\right)^m. \quad (5)$$

The sum of exponentials in Equation 5 is equal to:

$$\sum_{m=0}^{N-1} \left(\exp\left(\frac{2\pi}{2N} j\,(2i-k)\right)\right)^m =$$
$$\begin{cases} N & \text{if } k \text{ is even and } k = 2i \\ 0 & \text{if } k \text{ is even and } k \neq 2i \\ \frac{1-\exp\left(\frac{2\pi}{2N} jN(2i-k)\right)}{1-\exp\left(\frac{2\pi}{2N} j(2i-k)\right)} = \frac{2}{1-\exp\left(\frac{2\pi}{2N} j(2i-k)\right)} & \text{if } k \text{ is odd.} \end{cases} \quad (6)$$

Inserting Equation 6 into Equation 5 leads to:

$$\tilde{G}_{k=2k'} = G_{k'} \quad (7)$$
$$\tilde{G}_{k=2k'+1} = \sum_{i=0}^{N-1} \frac{1}{N} \frac{2}{1-\exp\left(\frac{2\pi}{2N} j\,(2i-k)\right)} G_i. \quad (8)$$

We denote the weighting factor of G_i in Equation 8 by $w_{k,i}$. One can easily verify that $w_{k,i}$ is such that $w_{k+2,i} = w_{k,\lfloor i-1 \rfloor}$ where $\lfloor i-1 \rfloor$ is the remainder of the division of $i-1$ by N. This property is of interest if we write Equation 8 in matrix form:

$$\begin{bmatrix} \tilde{G}_1 \\ \tilde{G}_3 \\ \vdots \\ \tilde{G}_{2N-1} \end{bmatrix} = \begin{bmatrix} w_{1,0} & w_{1,1} & \cdots & w_{1,N-1} \\ w_{3,0} & w_{3,1} & \cdots & w_{3,N-1} \\ \vdots & & & \\ w_{2N-1,0} & w_{2N-1,1} & \cdots & w_{2N-1,N-1} \end{bmatrix} \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_{N-1} \end{bmatrix}. \quad (9)$$

Using the properties of $w(k,i)$ mentioned above, Equation 9 then becomes:

$$\begin{bmatrix} \tilde{G}_1 \\ \tilde{G}_3 \\ \vdots \\ \tilde{G}_{2N-1} \end{bmatrix} = \begin{bmatrix} w_{1,0} & w_{1,1} & \cdots & w_{1,N-1} \\ w_{1,N-1} & w_{1,0} & \cdots & w_{1,N-2} \\ \vdots & & & \\ w_{1,1} & w_{1,2} & \cdots & w_{1,0} \end{bmatrix} \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_{N-1} \end{bmatrix}. \quad (10)$$

Equation 10 shows that the matrix W formed from the weighting factors $w_{k,i}$, is circulant. A well known property of circulant matrices is that they are diagonalizable by Fourier matrices:

$$W = FDF^{-1}, \quad (11)$$

where D is a diagonal matrix such that $D = \text{diag}\,(F\mathbf{w})$, $\mathbf{w}$ is the vector formed by the elements of the first column of W and F is a Fourier matrix with elements $f_{ik} = e^{-2\pi jik/N}/\sqrt{N}$. Equation 10 can then be rewritten as:

$$\tilde{G}_{(k=2k'+1)} = WG = FDF^{-1}G. \quad (12)$$

This approach can also be computed through the scheme illustrated in Figure 2(b). Experiments verified that the diagonal matrix D is indeed composed of the same terms as the exponent term illustrated in Figure 2(b). Our approach thus corroborates the work in [6] but has the distinct advantage of scalability for managing computational complexity. Both the existing approach [6] and the new approach proposed here are equivalent and have reduced computational complexity compared to classic linear convolution. The FEXT implementation then requires $2N$-points FFTs and N complex multiplications.

3.3. Frequency resolution extension with reduced computational complexity

The approach presented in the previous section can be exploited to reduce the FEXT computational complexity even further. In this section we consider the case where $\tilde{G}_k$ is computed according to Equation 8 through multiplication of G_i by the weighting function $w(k,i)$.

The imaginary part of the weighting function $w(k,i)$ for $k = 127$ is depicted in Figure 3. The real part of $w(k,i)$ is not shown because it is constant for all i. The plot in Figure 3 shows that $w(k,i)$ does not equally weight the spectral gains G_i in the computation of $\tilde{G}_k$. More specifically, the closer the

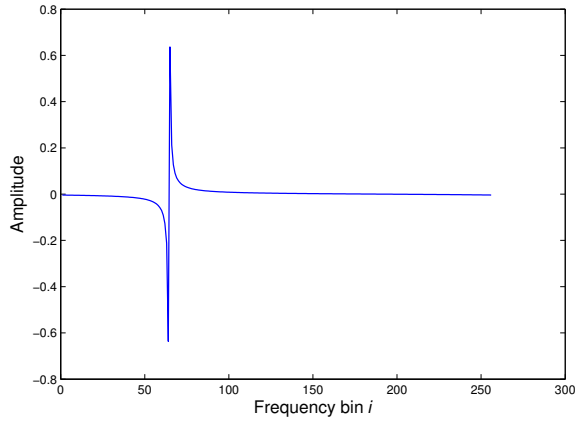


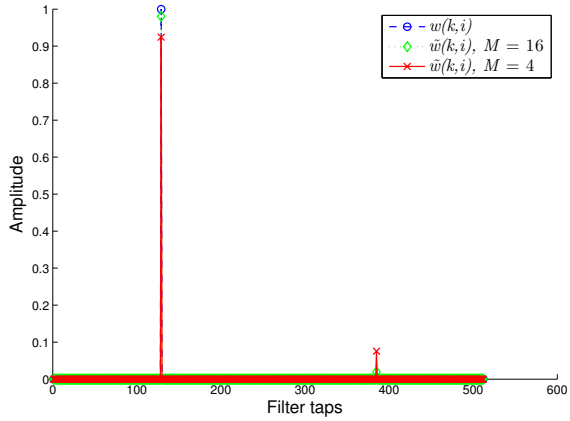
Fig. 3. Imaginary part of the weighting function $w(k,i)$ for $k = 127$ (i.e. normalized frequency $k/2N$) and $N = 256$

normalized frequency $\frac{i}{N}$ is to the normalized frequency $\frac{k}{2N}$, the more G_i influences the value of $\tilde{G}_k$ (and vice-versa). To reduce the computational complexity related to the computation of $\tilde{G}_k$, one can use a truncated approximation of $w(k,i)$ which we denote $\tilde{w}(k,i)$. The weighting function $\tilde{w}(k,i)$ is truncated to include only M points (with $M < N$) centered on the peak of $w(k,i)$. Therefore, the computation of $\tilde{G}_k$, as in Equation 8 requires less operations since $\tilde{w}(k,i)$ is equal to zero for some value of i. For all the spectral gains $\tilde{G}_k$, this computation requires $M \cdot N$ multiplications and $(M-1) \cdot N$ summations.

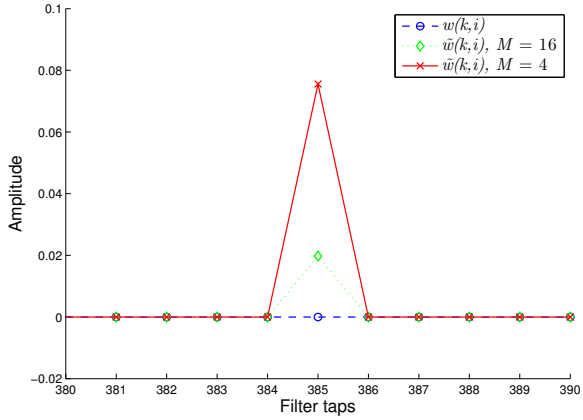
Compared to the optimum computation of Equation 12, the use of $\tilde{w}(k,i)$ is advantageous when $M \leq 2 \cdot log_2(N)$ if we assume that an N-point FFT has a computational complexity of $Nlog_2(N)$. The scalability comes from the fact that one can choose the value of M according to the computational load of the system.

4. EXPERIMENTS

The truncation of the weighting function $w(k,i)$ to reduce the computational complexity of the system may introduce some distortion since the resulting approximation $\tilde{w}(k,i)$ contains less information than the original function. The comparison with the approach in [6] is implicit as we showed in Section 3.2 that this system corresponds to the use of the complete weighting function $w(k,i)$. In order to evaluate the impact of such approximation, we performed assessments on filters and then on speech signals before more throughout experimentations in a noise reduction context. These assessments are reported in Sections 4.1, 4.2 and 4.3 respectively.



(a) Impulse response



(b) Zoom on the impulse response

Fig. 4. Impulse response for FEXT different weighting function configurations

4.1. Impact of FEXT optimization on filters

Here, we evaluate the impact of the approximation by comparing the impulse response and the frequency response of a filter obtained with the full weighting function to that obtained with a truncated weighting function. In the test reported here, spectral gains G_i are all set to 0dB and N is set to 256. These spectral gains are processed by the FEXT to obtain new spectral gains. Figure 4(a) shows the impulse responses obtained when FEXT uses the full weighting function $w(k,i)$ or two truncated weighting functions $\tilde{w}(k,i)$ (of length $M = 16$ and $M = 4$ respectively). We observe that impulse responses obtained with the truncated weighting functions do not exactly match the reference impulse response which, in this case, is composed of a solitary peak of unit amplitude. In the case of the approximated impulse responses, the peak amplitude is slightly lower than 1 (0.9802 for $M = 16$ and 0.9244 for $M = 4$ respectively). Moreover,

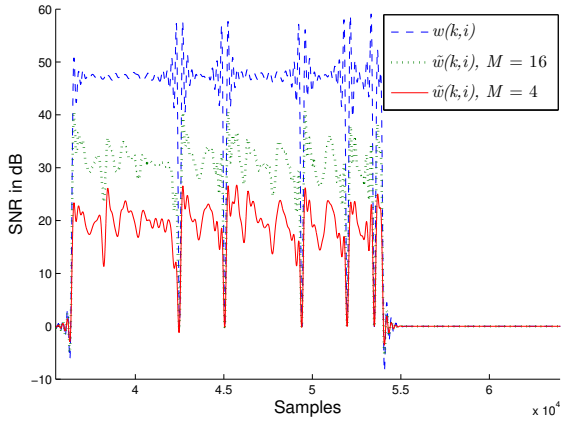


Fig. 5. Signal-to-noise-ratio (SNR) between input signal and reconstruction error between the input signal and the output of the FEXT module

as we can see on Figures 4(a) and 4(b), both approximations have a second peak which is small compared to the main peak but whose amplitude increases with decreasing M (0.02 for $M = 16$ and 0.039 for $M = 4$ respectively). When comparing the spectrum of the approximated impulse responses with that of the spectral gains, we observe that they contain a small ripple. Instead of all the spectral gains being equal to 0dB (as for the input spectral gains G_i), the spectral gains of the odd frequency bins are constant (i.e. -0.1751dB for $M = 16$ and -0.7716dB for $M = 4$) and the spectral gains of the even frequency bins are of course equal to 0dB as they do not require any computation. The effects observed on the filter can be judged as annoying or negligible depending on the application. With mobile communications for example, such artifacts may be irrelevant whereas in high quality speech enhancement systems, they may be very annoying.

4.2. Impact of FEXT optimization on speech signals

Here we report the impact of the optimized FEXT on speech signals. We undertook a similar experiment to that reported in Section 4.1 but this time using a speech signal as input to the FEXT module. An input speech signal is transformed into the frequency domain through an N-point FFT with overlapping frames of 256 samples (128 new samples and 128 samples from the previous frame) and with Hanning windowing. The number of frequency bins N is set to 256. The obtained spectrum is processed by the FEXT and transformed back into the time domain through a $2N$-point FFT. Except for the FEXT, no additional processing is performed in the frequency domain. Figure 5 shows the signal-to-noise-ratio (SNR) between the input speech and the reconstruction error observed between the input signal and the output of the FEXT module. The SNR was forced to zero during speech pauses so that it reflects the impact of the FEXT on the speech signal only. We observed that, without any approximation in the FEXT, the SNR is approximately 50dB during speech periods. The use of a truncated weighting function results in a modest degradation in SNR (about 20dB with $M = 4$). Moreover informal listening tests indicate that these degradations in SNR are not audible. This shows that truncating the weighting function $w(k, i)$ does not adversely affect speech quality in this instance.

4.3. Results on noise reduction

We now evaluate the impact of the use of $\tilde{w}(k, i)$ on speech signals within a noise reduction algorithm. The noise reduction spectral gains are updated through a Wiener rule [1]. The algorithm inputs are the noisy speech signal $y(n)$, the noise signal $b(n)$ and the clean speech signal $s(n)$ thus no estimation is required. This is done to avoid distortions from estimation errors so that observed distortions are directly attributed to the filtering approach under investigation.

Speech signals have a sampling frequency of 8 kHz and we performed our tests on a database of 48 signals. This database includes signals with signal to noise ratios ranging from 0 to 15dB. To avoid speech distortions in regions of locally low signal to noise ratio the noise reduction spectral gains are limited using a fixed spectral floor.

In our simulations, the input signal is processed in frames of 256 samples with an overlap between successive frames of 50% (128 new samples and 128 samples from the previous frame). The number of frequency bins N is set to 256. The reduced complexity system is assessed with two configurations: M is set to 16 then to 4 to assess the trade-off between reduced computational complexity and degradation in speech quality.

Distortions are assessed through the cepstral distance (CD) between the clean speech $s(n)$ and the processed speech signal $\hat{s}(n)$:

$$\begin{aligned} CD(n) &= \sqrt{\textstyle\sum_K [C_s(n) - C_{\hat{s}}(n)]^2} \\ C_s(n) &= \text{IDFT}\left\{ln|\text{DFT}(s(K))|\right\} \end{aligned} \tag{13}$$

where K spans 256 samples. We also conducted some informal listening tests to assess subjective speech quality.

Figure 6 shows an example CD profile for the different filtering approaches tested. Here there is only one curve for linear convolution because the systems illustrated in Figures 2(a) and 2(b) give exactly the same results. Of particular note are the profile for the low computational system with $M = 16$ which is very similar to that of the linear convolution. At the contrary, the low computational system with $M = 4$ which has high CD values compared to that of the linear convolution. These observations show that the low computational system do not necessarily increase distortions.

Informal listening tests revealed some differences bet-

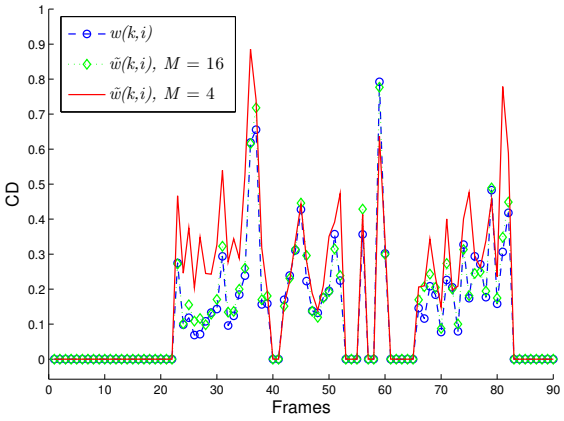


Fig. 6. Cepstral distance against signal frames

ween processed speech signals. Speech signals processed by the low computational complexity system with $M = 4$ have some distortions during speech periods compared to signals processed by linear convolution. These distortions can be described as crackling noise but still there were not perceived as annoying. For systems with $M = 16$, no distortions were perceived in comparison to linear convolution systems. This shows that low computational complexity systems can efficiently be used instead of linear convolution.

5. CONCLUSION

Linear convolution in the frequency domain necessitates the use of frequency resolution extension (FEXT) through zero-padding. This paper introduces the first computationally efficient and scalable approach to FEXT.

We introduce an implementation of FEXT which performs linear convolution in the frequency domain. The most appealing feature of the proposed approach is its scalability through which one can improve computational efficiency with only modest increases in speech distortion. The computationally efficient implementations do not correspond exactly to the linear convolution but our tests showed that they still yield processed speech of high quality. Tests performed with noise reduction showed that this approach is a suitable computationally efficient alternative to full linear convolution.

6. REFERENCES

[1] Eberhard Hänsler and Gerhard Schmidt, *Acoustic Echo and Noise Control: A Practical Approach*, Wiley-Interscience, 2004.

[2] Christelle Yemdji, Moctar Mossi Idrissa, Nicholas W. D. Evans, and Christophe Beaugeant, "Efficient low delay filtering for residual echo suppression," in *Proc. European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, Aug. 2010.

[3] Heinrich W. Löllmann and Peter Vary, "Uniform and warped low delay filter-banks for speech enhancement," *Speech Communications*, vol. 49, no. 7–8, pp. 574–587, 2007.

[4] P.C.W. Sommen and J.A.K.S. Jayasinghe, "On frequency domain adaptive filters using the overlap-add method," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Jun. 1988, pp. 27 –30.

[5] Alan V. Oppenheim and Ronald W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 1999.

[6] J.I. Marin-Hurtado and D.V. Anderson, "Distortions in speech enhancement due to block processing," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010, pp. 4774 –4777.

[7] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, 2002.