University of Brescia
Departement of Information Engeneering

M.Sc. Communication Technologies and Multimedia

- ADVANCED METHODS FOR INFORMATION REPRESENTATION -

Final Project

A Matlab implementation for speech enhancement based on adaptive wavelet denoising on multitaper spectrum

Fierro Leonardo MAT **704495**

Bertozzi Iacopo MAT **91828**

Bianciardi Alessandro MAT **91725**

Nodari Stefano MAT **80338**

Academic year 2017/2018

1 Algorithm implementation

1.1 Audio acquisition, additive noise and filtering

A wave audio signal is imported using Matlab function *audioread*. Data used was chosen from *TIMIT* database, which contains broadband recordings of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences. The TIMIT corpus includes time-aligned orthographic, phonetic and word transcriptions as well as a 16-bit, 16-kHz speech waveform file for each utterance.

The audio file is then normalized by its absolute maximum value, since Matlab range for audio signals takes values from -1 to +1.

The sample frequency *Fs* is 16 kHz, as reported above. The frame size N can be chosen: 1024 samples with 50% overlap is a fair trade-off between temporal and frequency localization. The overlapped windowing is performed by means of the function *buffer*.

Since the statistical features of the noise are going to be extracted from the noised signal, a zero-padding at the beginning of the clean audio file is done, in order to produce an unvoiced first frame inside the overall speech.

A random gaussian noise vector is then generated with the same duration of the clean audio file and added to this one in order to generate the noisy speech, which is later filtered by means of a 8-kHz LPF (for further details, see function *mylowpass*).

1.2 Multi-taper spectrum estimation

Choosing a MTS estimator, the variance of the error introduced during the estimation can be reduced by a factor of L compared to traditional windowing methods, where L is the number of tapers used (in this case, $L = 5$ is a fair value).

The tapers are designed to be orthonormal, and a popular choice resides in the sine tapers (l is the taper number):

$$a_l(m) = \sqrt{\frac{2}{N+1}} \sin \frac{\pi(m+1)}{N+1} \tag{1}$$

The MTS estimator for each audio frame is defined as:

$$\hat{S}_y(k) = \frac{1}{L} \sum_{k=1}^{L} \hat{s}_l(k) \tag{2}$$

$$where\ \hat{s}_l(k) = \Big| \sum_{m=0}^{N-1} a_l(m) x(m) e^{-i2\pi km} \Big|^2$$

Once the tapers has been obtained, it is possible to perform the MTS estimation on each single audio frame. Since the algorithm imposes that the noise is known, but this implementation is designed to work also in zero-knowledge situations, the statistical description $\hat{S}_n$ of the noise can be approximated by computing the MTS of the first audio frame, which is unvoiced, as it was reported above.

The logarithm of the spectrum estimation is then computed, since it is the one that is going to be thresholded.

The MT spectrum estimation of the original clean audio signal can also be obtained, in order to perform a graphical comparison with the noisy signal spectrum and, later, of the denoised thresholded one.

1.3 Discrete Wavelet Transform

A DWT is performed both on the MTS and the log MTS of the noisy speech, using function *wavedec* and 4^{th} order Daubechies wavelets up to level $J = 5$.

For each level, that is, for each decomposition, a scaling component c_j and a detail component w_j are obtained, where j indicates the scale/decomposition:

$$\{c_{j,k}^{mt}, w_{j,k}^{mt}\} \qquad \{c_{j,k}^{logmt}, w_{j,k}^{logmt}\}$$

1.4 Thresholding

Computing the threshold

As reported in the attached presentation, it is possible to model the noisy signal as the distrbution of noise n plus the clean signal in the log MTS domain q:

$$z = log(n + q) \tag{3}$$

A good estimation of q for each scale j is obtained from the scaling coefficient of the linear MTS of the noisy speech as follows:

$$q_{j,k} = 2^{-j}\big(\frac{1}{2R+1}\sum_{n=-R}^{R} c_{j,n+k}^{mt} - q_j^{noise}\big) \tag{4}$$

where j is the scale parameter, k is the axis and $R = 2$ specifies the local window size. q_j^{noise} is the DC shift of the scaling coefficients due to the noise and can be approximated by the mean of all scaling coefficients at the current scale j.

After retrieving all q_j, the adaptive threshold δ for each scale j is computed as a function of the standard deviation of the z distribution:

$$\delta_{j,k} = \sigma_{z\ j,k}\sqrt{2\ logN} \tag{5}$$

Applying the threshold

The denoising shrinkage must be performed using the just obtained thresholds δ_j on the detail coefficients of the log MTS at each scale j. There are two different possible thresholding approaches:

- hard thresholding:

$$w_{j,k}^{logmt} = \begin{cases} w_{j,k} & if\ w_{j,k} \geq \delta_{j,k} \\ 0 & if\ w_{j,k} < \delta_{j,k} \end{cases}$$

- soft thresholding:

$$w_{j,k}^{logmt} = \begin{cases} w_{j,k} - \delta_{j,k} & if\ w_{j,k} \geq \delta_{j,k} \\ 0 & if\ |w_{j,k}| < \delta_{j,k} \\ w_{j,k} + \delta_{j,k} & if\ w_{j,k} \leq -\delta_{j,k} \end{cases}$$

Both of them work fine, even if the soft threshold provides slightly better results.

IDWT, Wiener filter and reconstruction

The refined spectrum is retrieved by performing the inverse discrete wavelet transform with function *waverec* and then the exponentiation to get back to the linear MTS. Since it is not possible to get back from the PSD without knowing the phase, the enhanced denoised spectrum is used instead to determine the gain function of a Wiener filter:

$$g(k) = \frac{\hat{S}_x}{\hat{S}_x + \mu \cdot \hat{S}_n} \qquad X(k) = g(k)\ Y(k)$$

$$SNR = \frac{\sum_{i=0}^{N-1} \hat{S}_x(i)}{\sum_{i=0}^{N-1} \hat{S}_n(i)} \qquad \mu = \begin{cases} \mu_0 - b \cdot SNR_{dB} & -5 < SNR_{dB} < 20 \\ 1 & SNR_{dB} \geq 20 \\ \mu_{max} & SNR_{dB} \leq -5 \end{cases}$$

In this way, the Fourier transform estimation of the speech is obtained. By taking the ISTFT of it and then reconstructing the signal waveform from the overlapped parts, the final enhanced and denoised version of the original audio signal is retrieved.

The algorithm can be iterated, by using the output signal as the input noisy signal, obtaining better results in case of noise with high variance.