MATLAB IMPLEMENTATION OF THE COUNT-COMPARISON LSO MODEL

J. Bouše, V. Vencovský

Department of Radioelectronics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

Abstract

Hearing system allows us to localize the sound sources in the open space. Sound is entering auditory system by ears placed on contralateral sides of the head. This generally causes that the sounds coming from the different angles in a space are not exactly the same in both ears. Interaural time and level differences can occure. These differences allow us to localize the sound in space. Binaural model allowing to detect interaural level differences between left and right ear was described in the paper of Ville Pulkki and Toni Hirvonen [5]. Implementation of the LSO part of model in MATLAB is desribed in this paper.

1 Introduction

The ability to localize sound sources in the space is one of the fundamental properties of the hearing system. Since we have two ears placed on the contralateral sides of the head, sound signals entering both ears are not exactly the same. Beside the other factors there are interaural level differences and interaural time differences between the left and right signal [2]. Interaural level differences are beleived to be coded in the Lateral Superior Olive (LSO) placed in the brainstem. There are two LSO each in one hemisphere. Numerous of physiological experiments revealed that the neural activity in the LSO is higher when the sound signal in the ipsilateral ear has higher level in comparison to the sound signal in the contralateral ear [6].

An overview of existing LSO models can be found here [4]. This paper describes MATLAB implementation of the LSO part of the count-comparison binaural model designed by Ville Pulkki and Toni Hirvonen [5]. The binaural part of the model allowing detection of the interaural time differences will be added into the model in the future and the model will be used for simulations of localization of sound sources placed in the horizontal plane.

2 Proposed model

A schematic diagram of the model is presented in Figure 1. The model can be divided in two parts: monaural processing and binaural processing which simulates roughly the human cochlea and lateral superior olive (LSO) respectively.

2.1 Monaural processing

The monaural processing simulates the cochlear and the nerve excitation in human ear. The text bellow describes the processing of only one ear since the second ear channel is processed in the same way.

2.1.1 Gammatone filter bank

The first stage of monaural processing is the 4th grade gammatone filter bank which divides the input signal into 115 frequency bands (400 Hz–14 kHz) spaced with 1/4 ERB. This process tries to simulate the distance-to-frequency transformation in the cochlea.

The general way to compute gammatone filter's impulse response is defined by equation[3]

$$h[n] = An^{g-1}e^{-2\pi bn}\cos\left(2\pi f_c n + \phi\right), \tag{1}$$

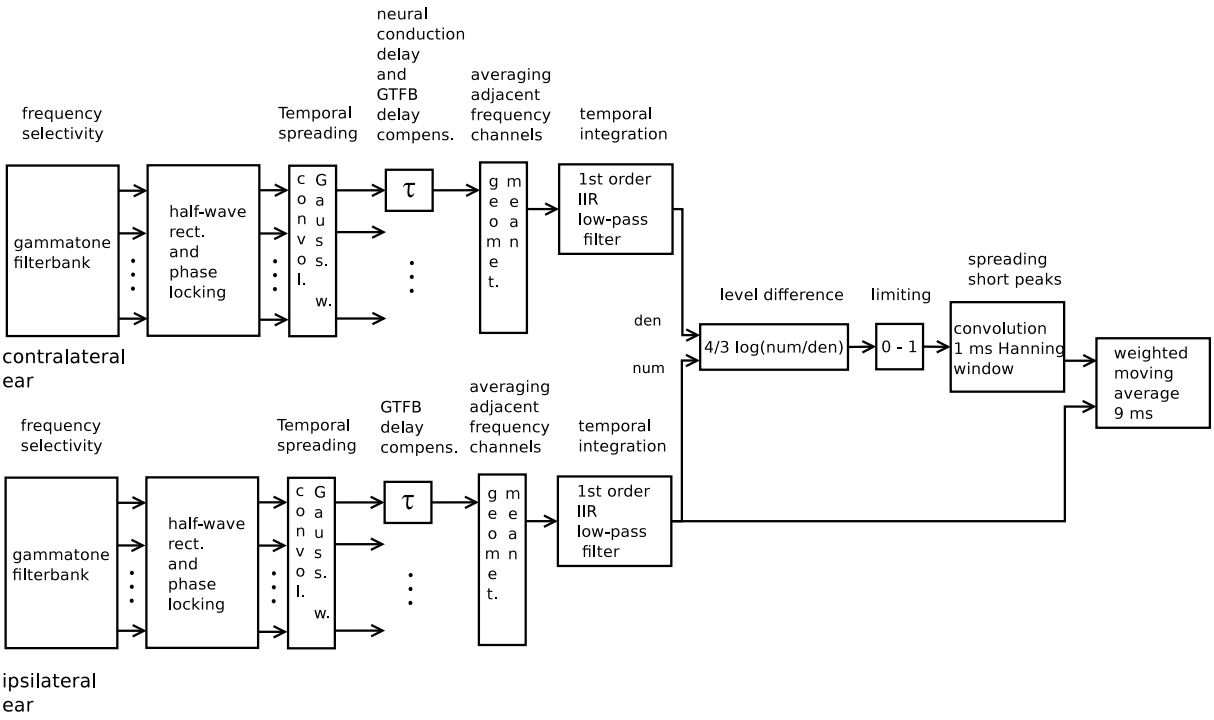

Figure 1: The model diagram

where A is the amplitude, n is number of sample, g is grade of the filter, b is the bandwidth of the filter (for 4th grade filter it's equal to 1.019), f_c is central frequency of the band and ϕ is the initial phase of the filter.

Implementation in MATLAB: The implementation of the gammatone filter bank in MATLAB is done by two functions. The first one is `[N cERB]=ERBSpaceII(lowf, highf, density)` where `N` is scalar representing number of ERB central frequencies, `cERB` is the vector of the ERB central frequencies, `lowf` and `highf` is the lowest and the highest frequency of the ERB space respectively and `density`(0.25 in our case) is the density of the ERB central frequencies across the ERB band.

The second function `[out]=gamma_filt(fs,in,cERB,downS)` firstly creates gammatone filters' coefficients and then these coefficients are used to filter the intput signal by MATLAB function `filter`, where `out` is output matrix where columns represent the frequency bands and rows the time samples, `fs` is the sampling frequency, `in` the input signal, `cERB` is the vector of the ERB central frequencies and `downS` allows us to use downsampling during the processing (1 means no downsampling used).

2.1.2 Half-wave rectification and phase locking

The signal is half-wave rectified and phase locked after the gammatone filterbank block. The phase locking in the cochlear nucleus is implemented by replacing each half-wave by an impulse at local maximum with the magnitude which equals to the RMS of the half-wave (Fig. 2).

Implementation in MATLAB: For the half-wave rectification is used MATLAB function `find` which gives us the indexes of samples smaller than zero. These samples are replaced by zero values. Then each frequency band of the half-rectified signal goes through the modified (the indexes of the local extremes are not ordered by size) function `[xmax,imax,xmin,imin] = extrema(x)` [1] where `xmax`, `xmin` is the vector with local maximas, minimas values respectively

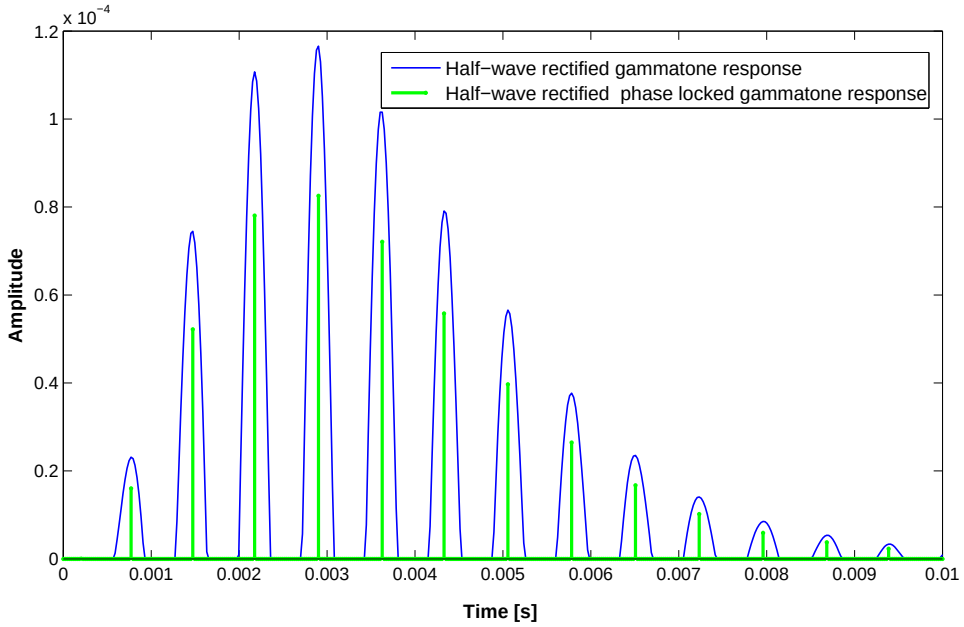

Figure 2: The response of the half-wave rectified gammatone filter and half-wave rectified phase locked gammatone filter with central frequency of 1381 Hz to sinusoidal signal with frequency 3000 Hz

and `imax`, `imin` is the vector with indexes of local maximas,minimas respectively. The pair of local minimas indexes minus zero values between them represent the borders of one half-wave in processed frequency band from which the RMS value is computed. The impulse with amplitude equals to the RMS is then placed at the position of half-wave's local maxima.

2.1.3 Temporal spreading

In reality the impulses in the trapezoid body are spread about 1/10 of cycle period.[5] This spreading is modeled in adjacent block by convolving the impulses with a Gaussian window w[5]

$$w(n) = e^{-\frac{1}{2}(\frac{\alpha n}{N/2})^2}, \tag{2}$$

where n is sample index between $N/2$ and $-N/2$. Constant α is set to value of 20. The window length N depends on frequency f[5]

$$N = \begin{cases} \frac{2}{f} f_s, & f < 800Hz \\ 0.0024(0.6 + 0.4\frac{f}{800}) f_s, & f \in \langle 800, 2800 \rangle \, Hz \\ 0.0048 f_s, & f > 2800Hz, \end{cases} \tag{3}$$

where f_s is sampling frequency.

Implementation in MATLAB: In MATLAB we use Eq. 2 and with respect to central frequency of the band Eq.3 to compute the impulse response of the Gaussian filter. After that the convolution between input signal and Gaussian windows is made by MATLAB function `out=conv(in,w, 'same')` where `out` is output vector, `in` is input signal, `w` is the impulse response of the Gaussian filter and parameter `'same'` sets the length of the output vector `out` equals to input vector `in`.

2.1.4 Neural conduction delay and gammatone filter bank delay compensation

The compensation of non-constant gammatone filter time delay is numerically computed for each frequency band individually so that the impulse responses of each frequency band show the highest peaks at the same time constant. Then the neural conduction delay of 0.4 ms is added because of the length of neuron to LSO.

Implementation in MATLAB: `[delay]=gammatone_sync(density,fs)` function is used for the delay compensation, where `delay` is the vector which represents numerical computed delay of the gammatone filter bands, `density` is the density of the ERB central frequencies across the ERB band and `fs` is sampling frequency. Inside the `gammatone_sync` function is generated the same gammatone filter bank as in the monaural process and the unity impulse is fed to the input. In the final impulse responses are found maximum peaks using the `max` function. Desired compensation delays are computed as every channel is moved by adding zero vector which length is equal to the difference between the maximal delay and delay of the processed band, to each frequency band besides the frequency band with maximum delay and to ensure the same vector sizes the ends of the input vectors are cut.
Neural conduction delay is provided by adding zero vector with length equal to 0.4 ms in time samples.

2.1.5 Averaging adjacent frequency channels and temporal integration

The geometric mean implements a coincidence counting function, or AND-like function which results in the active output only if a pulse arrives from each frequency channel at the same time.[5]

$$y_j = \sqrt[2W+1]{\prod_{i=j-W}^{j+W} x_i}, \tag{4}$$

where y_j is the result of averaging of the jth frequency band, W is a parameter which controls the width of the frequency window to averaging (in our case it's set to 3), and x_i is the signal of the ith frequency band from the cochlear model. The output vector is then summed across the rows.
In the adjacent block the averaged signal from geom. mean is temporally integrated by first-order IIR filter with time constant of 1ms.

Implementation in MATLAB: The geometric mean is implemented only by rewriting the Eq. 4 in MATLAB. The temporal integration is then made by designing the filter coefficients with function `[b,a]=butter(1,wn, 'low')` where parameter `1` is an order of the filter, `wn` is an angular frequency derived from the time constant of the filter and parameter `'low'` sets the filter the to be a low-pass. The computed filter coefficients are then used in filtering by a function `filter`.

2.2 Binaural processing

2.2.1 Level difference computation and limiting

The binaural interaction is implemented by dividing the ipsalateral input by the contralateral input sample by sample and by taking a logarithm of the result. Multiplication of the logarithm by a factor 4/3 is chosen to provide unity output when the level difference between the ipsalateral and the contralateral input is equal to 15 dB. [5]

$$ILD = \frac{4}{3} log_{10}(\frac{num}{den}) \tag{5}$$

The output of the calculation is limited between 0-1.

Implementation in MATLAB: We use the Eq. 5. Because of the logarithm in the formula the output is not defined for zero input values so we have to use `ILD(find(isnan(ILD)))=0` command to replace them by zero in the output vector. With similar commands the signal between 0-1 is limited, `ILD(find(ILD<0)))=0` and `ILD(find(ILD>1)))=1` respectively.

2.2.2 Spreading short peaks

In this stage the signal is convolved with a 1 ms Hanning window. This operation smooths short peaks in the output which can occur, e.q., during signal onsets.

Implementation in MATLAB: The `han=hann( round(1e-3*fs))` function generate Hanning window with the length equal to number of samples in 1 ms. The window is convolved with input signal. The output is then divided by a direct component of the Hanning window.

2.2.3 Peak following

The last stage is weighted moving average(see Figure 3) which follows the maximum value of the peaked output. The ipsilateral signal is used as a weight since it was found as most appropriate in this case.[5]

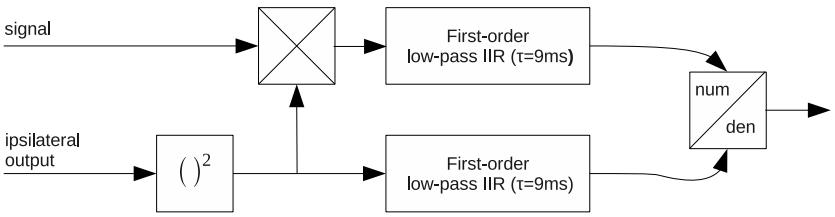

Figure 3: The weighted moving average diagram

Implementation in MATLAB: The block is implemented in accordance to scheme (Figure 3). The IIR filters' coefficients are designed by the `[b,a]=butter(1,wn, 'low')` function and then the appropriate signals are filtered by this filter. Small intervention into output signal from weighted moving average is needed because the denominator signal in division subblock might be equal to zero which means that the output signal might rise to infinity.This is done by the same command as in the section 2.2.1, `ILD(find(isnan(ILD)))=0`.

3 Results

The output of the model was measured by sinusoidal signal for several interchannel level differences. As can be seen on the Figure 4 the output of the implemented mode is similar to the Figure 5 from the paper[5].

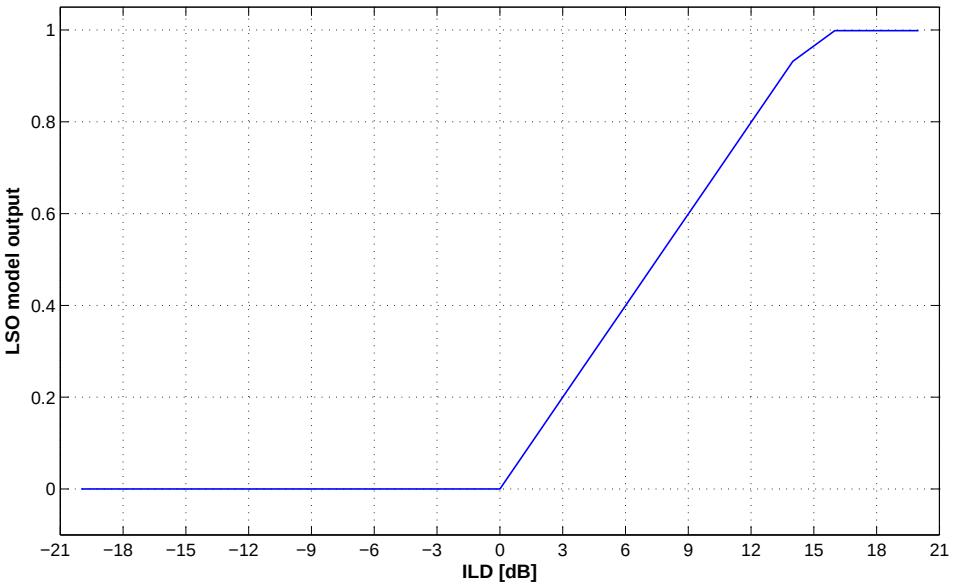

Figure 4: The dependence of ILD to the interchannel level differences (step 2 dB) for sinusoidal signal with frequency of 3.8 kHz.

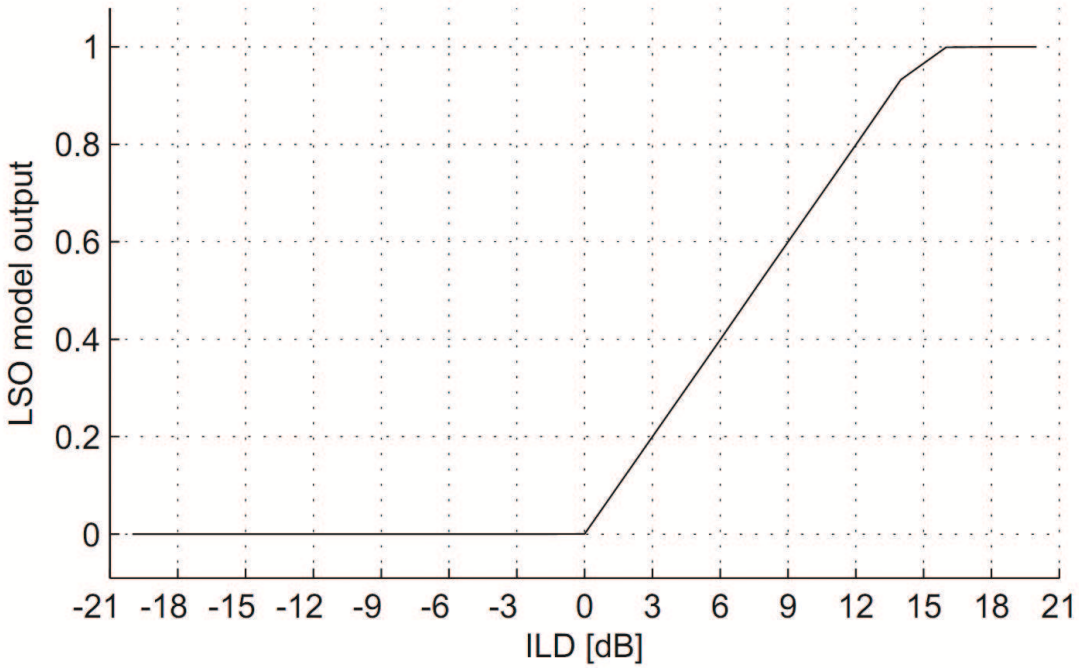

Figure 5: The dependence of ILD to the interchannel level differences for sinusoidal signal with frequency of 3.8 kHz, taken from [5]

4 Conclusion

The LSO part of the count-comparison binaural model designed by Ville Pulkki and Toni Hirvonen [5] was implemented in MATLAB. MSO part will be added to the model in the future and the model will be used for simulations of localization of sound sources placed in the horizontal plane.

Acknowledgements

This paper was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS11/159/OHK3/3T/13 and the research program MSM 6840770014 of Ministry of Education of the Czech Republic.

References

[1] Carlos Adrian, Vargas Auilera *Extrema matlab function* MATLAB CENTRAL, url:"http://www.mathworks.com/matlabcentral/fileexchange/12275", 2007

[2] Jens Blauert *Spatial Hearing* The MIT Press, 1996, ISBN 0-262-02413-6

[3] Marek Drápal *Model prostoroveho slyseni*, Diploma Thesis, CTU in Prague, 2006.

[4] T.R. Jennings, H.S. Colburn, Models of the Superior Olivary Complex, In: *Springer Handbook of Auditory Research: Computational Models of the Auditory System*, edited by R. Meddis,E. A. Lopez-Poveda, R. R. Fay, A. N. Popper New York: Springer, 2006, 277 p. ISBN 978-1-4419-1370-8

[5] Ville Pulkki, Toni Hirvonen Functional Count-Comparison Model for Binaural Decoding. *Acta Acoustica united with Acoustica*, 2009, vol 95. pp. 883–899.

[6] J.O. Pickles, *An Introduction to the Physiology of Hearing, Third Edition* Academic Press, 3 edition, 2008, 400 p. ISBN 978-0120885213

Jaroslav Bouše
Department of Radio Engineering, FEE, CTU in Prague, Technická 2, 166 27, Praha 6
bousejar@fel.cvut.cz

Václav Vencovský
Department of Radio Engineering, FEE, CTU in Prague, Technická 2, 166 27, Praha 6
vencovac@fel.cvut.cz