Audio processing: lab sessions

Session 2: Binaural synthesis and 3D audio: OLA and WOLA frameworks

Alexander Bertrand, Jesper Brunnström and Julian Schott.[1]

October 2022

Introduction

In Session 1, the required time domain filters were obtained in order to spatially resynthesize a particular acoustic environment. One of the issues that can arise is the computational complexity of the filtering operations in the time domain. As an alternative, such filtering can be accomplished using frequency domain techniques. In this lab, two such techniques, namely the overlap-add (OLA) and the weighted overlap-add (WOLA), are investigated.

Practical information

The code you have created to solve the exercises in the lab should be submitted by 12.00 (midday) the day before the evaluation session. Make sure any dependencies (functions or data) are included in the submission, such that it will run on other computers. For the same purpose, avoid absolute file paths in the scripts.

Please include a script called main.m which solves the selected exercise, Task 7 of Section 2-2: *Implementation of the WOLA method*. The script should convey how well the approach of WOLA & OLA performs the task of generating the binaural signals. The demonstration script does not necessarily have to answer questions about the number of FFT points and computational complexity stated in the task description.

Exercise 2.1: Implementation of the OLA method

The OLA method offers an efficient Fast Fourier Transform (FFT) based way to compute the convolution of a (long) signal with a finite (but long) impulse response filter, and has been discussed in your first DSP course.

- A short and basic discussion is available here: `https://en.wikipedia.org/wiki/Overlap-add_method`
- A more extensive discussion is available here (page 3-26): `https://homes.esat.kuleuven.be/~dspuser/DSP-CIS/2022-2023/material/chapter13.pdf`

[1]For questions and remarks: `jesper.brunnstroem@kuleuven.be` and `jschott@esat.kuleuven.be`.

From Session 1, you already have access to the following:

- The speech source signal
- The filters, $\mathbf{g}_j[n]$, for $j = 1, \ldots, J$, where J is the number of loudspeakers.
- The room impulse responses (RIRs) from each of the loudspeakers to the left ear, $\mathbf{h}_{jL}[n]$, and from each of the loudspeakers to the right ear, $\mathbf{h}_{jR}[n]$.

In the time domain, to obtain the binaural signal, you firstly filter the speech source signal with $\mathbf{g}_j[n]$ for each of the loudspeakers to obtain the J loudspeaker signals. These loudspeaker signals are then filtered by the $\mathbf{h}_{jL}[n]$ for the left ear, and by $\mathbf{h}_{jR}[n]$ for the right ear. As this procedure can be quite computationally intensive in the time-domain, a frequency-domain based framework is used instead. Therefore, your first task is:

1. Complete the missing sections in the Matlab code provided, *OLA_skeleton.m*, which implements the OLA method. This function will take arguments of the signal to be filtered, the filter, and the number of points for the FFT.

Perform the following experiments using OLA to ensure its correct implementation.

2. Create a filter that has a Dirac impulse response, and use it to filter the speech source signal. For the Dirac impulse response, you can create a vector with a one in the first position and the rest as zeros. The dimension of the vector should be small than the number of FFT points. Plot the output signal and check whether it looks as expected, given the nature of the Dirac impulse response.
3. To test the speed of the OLA function, firstly re-run the code from Session 1 to obtain the filters, $\mathbf{g}_j[n]$, $\mathbf{h}_{jL}[n]$, and $\mathbf{h}_{jR}[n]$ for the HRTF case. Create the binaural signals using the *conv* function in Matlab. Create the binaural signals using your OLA function. Is there a difference in speed? Are the two signals perceptually different?

Exercise 2.2: Implementation of the WOLA method

The WOLA method offers a popular way to perform frequency domain processing, i.e. perform various processing tasks (filtering, noise reduction, interference cancellation, etc.) in the frequency domain. It combines FFTs of windowed input signal segments (using a so-called analysis window), frequency domain processing, and output signal synthesis using Inverse Fast Fourier Transforms (IFFTs) in an overlap-add configuration and adopting a second window function (a so-called synthesis window). The analysis and synthesis operations effectively correspond to a specific DFT-modulated filter bank operation.

- A short and basic discussion is available here (page 13-14): `https://homes.esat.kuleuven.be/~dspuser/dasp/material/Slides_2022_2023/Chapter-2%20Noise%20Reduction-pp40.pdf`

- A more extensive discussion is available here (page 4-19): `https://homes.esat.kuleuven.be/~dspuser/DSP-CIS/2022-2023/material/chapter14.pdf`

In this section you will implement the WOLA method and create functions for the analysis stage as well as for the synthesis stage. You will also be introduced to the spectrogram and Short-Time Fourier Transform (STFT) processing, which will be the framework used in Session 3.

1. Complete the missing sections in the Matlab code provided, *WOLA_analysis_skeleton.m*, which implements the analysis stage of the WOLA method. For this function, the arguments will include the source signal, the number of points for the FFT, the window, the sampling frequency, and the amount of overlap between successive frames. Applying this function to a particular source signal transforms it into the STFT domain.

2. Complete the missing sections in the Matlab code provided, *WOLA_synthesis_skeleton.m*, which implements the synthesis stage of the WOLA method. For this function, the arguments will include the STFT representation of the signal from the analysis stage, the number of points for the FFT, the window, and the amount of overlap between successive frames. Applying this function to the STFT domain signal results in the original time domain signal, provided that the windows used satisfy the perfect reconstruction condition.

In order to gain some additional insight into the analysis and synthesis stages, perform the following:

3. Apply the WOLA analysis stage function to the speech source signal. Observe the spectrogam (i.e the magnitude-squared of the complex coefficients over time and frequency). How does the speech source signal vary over time and frequency? Is it a stationary signal?

4. Apply the WOLA synthesis stage to the STFT domain signal after the analysis stage of the WOLA. Is the signal the same as the input speech source signal? What is the effect of different analysis and synthesis windows? What is the condition to have perfect reconstruction?

The final step is to do the filtering with the WOLA method. Typically, as will be seen in Session 3, it is common to perform both estimation and filtering within the WOLA framework. However, in this session, the WOLA method will only be used to filter the speech source signal with the filters $\mathbf{g}_j[n]$ to generate the loudspeaker signals. The OLA method will then be used for filtering these loudspeaker signals with the room impulse responses to the left and right ear.

5. Modify the *WOLA_analysis_skeleton.m* to perform the filtering with $\mathbf{g}_j[n]$

6. Apply the WOLA analysis and WOLA synthesis functions to obtain the loudspeaker signals. What is the effect of the number of FFT points in relation to the length of the filters?

7. Using these loudspeaker signals, apply the OLA filtering method to obtain the binaural resynthesized signals. What is the effect of the number of FFT points in relation to the length of the filters? How do these methods compare in terms of computational complexity with filtering in the time-domain?