ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Signal and Information Processing Laboratory
Institut für Signal- und Informationsverarbeitung

Spring Semester 2020 Prof. H.-A. Loeliger

Master's Thesis

Robust Time Delay Estimation via Onset Detection Filter Bank

Gustavo Cid Ornelas

Advisor: Elizabeth Ren

Abstract

Humans can perform sound source localization in a seemingly effortless manner. To do so, one of the main cues we rely on is the interaural time difference (ITD), which is related to the perceived time delay between the versions of the audio signal received by the right and left ears. There are many computational methods that strive to estimate the ITD from binaural audio signals and in [1], we proposed an algorithm based on fitting autonomous linear state space models (LSSM). The algorithm has three main modules, namely an onset detection module, which receives the binaural audio signals as an input and produces the local cost ratio (LCR) pairs, a local polynomial fitting module, where the LCR pairs are locally approximated by 3^{rd} degree polynomials, and a delay estimation module, which produces delay estimates from the local polynomial approximations. The formulation in [1] can be seen as a first implementation of the algorithm. This work takes the next step, where we extend the algorithm to robustly estimate the ITD for a wider class of audio signals. The extensions include the use of a filter bank structure comprised of multiple onset models with different frequencies, producing one LCR pair per frequency, locally approximating the LCR pairs with 3^{rd} degree polynomials in parallel and experimenting with different delay estimation modules, namely the vanilla and the confidence versions. We asses the algorithm's performance on a constructed binaural speech corpus and we compare it to the expected theoretical time delay as well as to some of the classical approaches for ITD estimation.

Contents

Chapter 1

Introduction

1.1 Sound source localization

Humans are capable of performing sound source localization even in complex environments in a seemingly effortless manner. Lord Rayleigh in his duplex theory [2], identified that there are two dominant cues that are used by humans to perform localization. They are the interaural level differences (ILD) and the interaural time differences (ITD), both of which are binaural cues. The former is related to the difference in the intensity of the sound perceived by the left and the right ears of a listener, while the latter is a measure of the time difference between the arriving sound waves at each ear.

Despite it being straightforward to intuitively grasp what it entails, there is no universally accepted definition of ITD. As a consequence, each computational method that aims to estimate it, produces slightly different results depending on the set of assumptions underlying them [3].

In general, the starting point of most computational methods is the so called head related transfer function (HRTF) or its time-domain counterpart, the head related impulse response (HRIR). The HRTF represents the relationship between the original sound signal produced at the source and the signal perceived by each ear, taking into account the physical propagation effects around the head and ears [3]. It is important to note that the HRTF considers all the effects in the frequency domain that might incur from the listener's torso, head, and pinnae on the sound wave. Therefore, the HRTF is unique to each person. Moreover, the HRTF is a function of the relative elevation between the sound source and the listener and of the sound source's azimuth. On the other hand, the HRTF does not take into account the

physical environment in which the listener is present.

When performing sound source localization in the horizontal plane (i.e., when there is no elevation difference between the sound source and the listener) the HRTF essentially encodes all the information needed about the ILD and ITD. The ILD information is present in the power of the filter, while the ITD is in the phase spectrum. This is why the HRTF is the working object of the classical approaches for sound source localization explored in the literature.

1.2 Classical methods for ITD estimation

As discussed in greater detail in [3], one can categorize the classical approaches for ITD estimation into three families of methods. The first one relies on the precedence effect and strives to estimate the ITD from the sound onsets. These are called threshold methods. The second family is comprised of cross-correlation methods and they estimate the ITD by comparing the information received by the left and right ears. Finally, there is the family of group delay methods, which estimate the ITD by calculating the group delay.

1.2.1 Threshold methods

The threshold methods require the HRIR as an input. First, a threshold value is defined and more often than not, this value depends on the peak level present in the HRIR (in [3], it is shown that a threshold of between 20 and 30 dB below the peak level of the HRIR produces more robust estimates of the ITD). Then, one processes the HRIR of the right and left ear sample by sample and checks if the HRIR is above the previously defined threshold. Once the values of one of the HRIRs surpasses the threshold, it becomes the point of reference. The ITD is estimated by computing the difference between the instant in which the other HRIR reaches the threshold level and the point of reference. There are other details that might be used to increase the robustness of this family of approaches, such as first filtering the HRIR with a lowpass filter. The interested reader is pointed to [3] for a more comprehensive description.

1.2.2 Cross-correlation methods

Cross-correlation methods rely on the comparison of the HRIR from the left and from the right. The ITD is, then, defined as the delay that maximizes the cross-correlation between the two signals. As with the threshold methods, there are many variations that can be applied to arrive at more robust results, such as using the centroid of the cross-correlation instead of its maximum value or computing the cross-correlation between the energy envelopes instead of the HRIRs, but in it's simplest form, the problem is comprised of

$$\mathrm{ITD}(\theta) = \underset{\tau}{\operatorname{argmax}} \frac{\int_{t_1}^{t_2} h_L(\theta, t) h_R(\theta, t+\tau) dt}{\sqrt{\int_{t_1}^{t_2} h_L^2(\theta, t) dt \int_{t_1}^{t_2} h_R^2(\theta, t) dt}}, \tag{1.1}$$

where t_1 and t_2 are the integration limits to compute the cross-correlation and $h_L(\theta, t)$ and $h_R(\theta, t)$ are the HRIRs from the left and right channels, respectively, for the azimuth θ. Again, the interested reader is pointed to [3] for further details.

1.2.3 Group delay methods

The group delay methods are based on a systems view of the HRTF. First, one notes that the HRTF can be factored as the product of its minimum phase and excess phase components. The excess phase component can be further factored into a linear and an allpass components. It was empirically shown in [4, 5] that the allpass component can be neglected without affecting human spatial perception. Additionally, it was also shown in [6] that, under certain conditions, the linear component can be replaced by a time delay without perceptible consequences. In [4], various approaches are described, but they are generally concerned with working with the linear component in ways to estimate the group delay and then map it to the ITD.

1.3 Limitations of the classical approaches

The classical approaches, presented in the previous section, have limitations that can be problematic if we think of estimating the ITD in some real world applications. The first issue is that they rely on the HRTF. The HRTF is unique to each individual and measuring it for a listener is cumbersome.

Thus, in a real world application where the ITD estimation is required, we would like to work with the raw audio signals. The second issue is that the classical methods are offline and for some applications, we might need to estimate the ITD in an online fashion and not have to wait for the complete signal to arrive before being able to produce an estimate.

There are ways to extend some of the classical methods so that they, in a sense, overcome these two limitations. To address the first issue, the threshold and cross-correlation methods have straightforward extensions to work with raw audio signals. For both methods, one could simply apply exactly the same algorithm, but replace the HRIRs with the audio signals. As for the second issue, instead of working with the full audio signals, one could work with smaller frames containing a limited amount of samples. Then, perform the computations defined in each method on a frame-by-frame basis, producing one ITD estimate per frame.

Evidently, the performance of both algorithms would suffer with these modifications. For instance, the threshold method would be much less robust, as the raw audio signal is significantly more noisy and varies more than the measured HRIRs. The cross-correlation would be expected to work reasonably well, but in real world scenarios, especially in reverberant environments, it would be prone to producing wrong ITD estimates as a consequence of comparing versions of the audio signal that do not match but that are contained in the same frame.

This work improves on the algorithm developed in [1]. We propose variations and assess the performance of an ITD estimation algorithm based on autonomous linear state space models (LSSMs) that work for a wide range of speech signals. The proposed algorithm estimates the ITD in an online manner and relies on the precedence effect to extract accurate timing information, thus overcoming the limitations of the classical approaches.

1.4 Proposed time delay estimation algorithm

The work in [1], developed by the same authors, can be seen as a first implementation of and as a proof of concept for the algorithm further developed here. We extend the basic formulation and modify the algorithm so that it robustly estimates the ITD for a wider class of speech signals.

Figure 1.1 illustrates the block diagram of the proposed algorithm. The

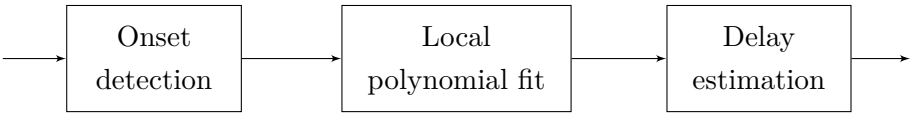


Figure 1.1: Block diagram of the proposed algorithm.

input is a binaural audio signal and the output is the estimated difference in the time of arrival between the two audio signals.

The algorithm starts by detecting sound onsets, which indicate the first-arriving sound. The use of the sound onsets for the task at hand is justified by the precedence effect [7]. The onsets correspond to the sound signal that travelled the direct path from the sound source and can, thus, give accurate timing information related to the localization of the source. The output of the onset detection stage are local cost ratio (LCR) signals, which are sparse signals that accurately represent if a sound onset was detected at each audio sample. The LCRs are, then, locally approximated by polynomials, to facilitate the last stage of processing, which corresponds to the delay estimation.

1.5 Validation procedure

A central part of this work is comprised of designing and running a validation procedure to assess the capabilities of the proposed time delay estimation algorithm. In order to achieve this goal, the first step is to think about the objectives that we wish to achieve with such a procedure. The three main objectives for the validation procedure are to show that

- our algorithm is producing reasonable ITD estimates if compared to the simplified theoretical ITD and to the ITD produced by the other methods;
- our algorithm is invariant and robust to different speech signals, i.e., it works well independent of the speaker, of the content of the speech and of the accent of the speaker;
- we can learn a mapping from the ITD to the azimuths and, with that map, localize the speaker in the horizontal plane.

With the previous goals in mind, we can design a validation procedure that encompasses all the objectives. The simplest way to test the previous

goals is to test the algorithm for multiple speech signals (from different speakers speaking different words and sentences), but that are all at the same azimuth. What we wish to show, then, is that the ITD estimates for all of those signals are similar to each other and are not too far from either the simplified theoretical delay or the ITD estimates produced by the other methods. Furthermore, if we move those speakers around, changing their azimuths, we can clearly see a trend in the magnitude of the ITD estimates and, from the estimates we could, in principle, learn a mapping from the ITDs to the sound source's azimuth.

In order to achieve the validation objectives, the first step is having a binaural speech signal corpus. This corpus should contain binaural speech signals from different speakers saying different things. We optimize our algorithm's parameters so that it properly works for the corpus.

As a reference, we compute the simplified theoretical delay expected for each azimuth we investigate. The theoretical delay is computed via a simplification of the physical model of the situation. We consider the sound source to be a point source and we assume it is far enough so that we can consider that the wave fronts arriving at the microphones are flat. To obtain the absolute value of the theoretical delay, it suffices to apply simple trigonometric identities to arrive at Equation (1.2)

$$\text{ITD} = \frac{d}{c}\sin(\theta), \tag{1.2}$$

where d is the separation between the microphones, c is the speed of sound and θ the azimuth of the sound source in radians.

We use the frame-wise threshold and cross correlation methods for comparison with the different versions of our algorithm.

1.6 Outline of thesis

This work is organized as follows: in Chapter 2, we discuss the two first modules in our algorithm's block diagram, namely the onset detection and local polynomial fitting modules; in Chapter 3, the delay estimation module is extensively explained; in Chapter 4, the experimental results are presented and analyzed; finally, in Chapter 5, the conclusions are drawn and we make suggestions for the direction of future work.

Chapter 2

Onset Detection and Local Polynomial Fitting

The first two modules in the proposed time delay estimation algorithm are the onset detection and the local polynomial fitting modules. Both of them are based on locally fitting LSSMs to the signals in their input. For the onset detection module, the inputs are binaural audio signals and we locally fit onset models to such signal. With the onset model fits, we are able to compute the local cost ratio (LCR), which is a local measure of the goodness of fit of the onset model to the different parts of the sound signal if compared to the zero-signal hypothesis, which can be used as a proxy to detect the presence of sound onsets. Then, for the local polynomial fitting module, the input are the LCR pairs, which are locally approximated by 3^{rd} degree polynomials, making them more tractable for the further processing steps done afterwards to extract the time delay information.

In this chapter, we briefly present the LSSM representation of the models that we fit and we quickly go through the recursive procedure followed to fit them. All the details and derivations of the equations presented in this chapter can be found in [1].

2.1 Fitting a LSSM

LSSMs are a powerful toolbox to perform many of the relevant tasks in signal processing. Besides their flexibility and representational power, it is possible to fit LSSMs in a computationally efficient way, using recursive

cost computations. In this section, we define LSSMs and formulate the optimization problem that is solved. A more extensive description of some of the families of signals that have LSSM representations as well as some of their important properties can be found in Chapter 2 of [1].

We work with LSSMs that run backwards in time, from sample k. The m-channel output signal $\tilde{y}_i$ at instant $i \leq k$ of a LSSM of order n is determined by

$$x_i = Ax_{i+1} \tag{2.1}$$

$$\tilde{y}_i = Cx_i, \tag{2.2}$$

where $A \in \mathbb{R}^{n\times n}$ is the *state transition matrix*, $C \in \mathbb{R}^{m\times n}$ is the *output matrix*, $x_i \in \mathbb{R}^n$ is the *state vector* and $\tilde{y}_i \in \mathbb{R}^m$ is the *output signal*. In the formulation followed in this work, we fix the initial state $x_k = s$ and vary C. The output signal at instant i is given by (2.3), which represents the trajectory defined by the LSSM.

$$\tilde{y}_i = CA^{k-i}s \tag{2.3}$$

Let y_i be a given m-channel signal at instant i and suppose we want to locally approximate it with $\tilde{y}_i$. To do so, we minimize the cost function defined in (2.4), which is the squared error weighted by the window $w_i \in \mathbb{R}$, which is responsible for localizing the fit around sample i.

$$\begin{aligned} J_k(C) &= \sum_{i=1}^{k} w_i \|y_i - \tilde{y}_i\|_2^2 \\ &= \sum_{i=1}^{k} w_i \left\|y_i - CA^{k-i}s\right\|_2^2 \end{aligned} \tag{2.4}$$

Therefore, fitting a LSSM means finding $\hat{C}$ in the feasible set $\mathcal{C}$, such that

$$\hat{C} = \underset{\hat{C}\in\mathcal{C}}{\operatorname{argmin}}\, J_k(C). \tag{2.5}$$

As previously mentioned, one can solve this optimization problem efficiently using recursions, via message passing.

2.2 Onset Detection

According to the precedence effect, mentioned in Chapter 1, the accurate timing information needed to perform sound source localization, can be found in the version of the audio signal that travelled the direct path from the source to the listener. Therefore, it is important to be able to detect such events, which is the role of the onset detection module.

A sound onset is characterized by a sudden rise in the signal value after a period of low energy. We need onset models that reflect this characteristic and are, thus, good representations of what the audio signal looks like in these moments. In [1], we explored two different onset models, namely the decaying sinusoid and the gammatone filter. We produced better results with the decaying sinusoid models and the fitting procedure of such models is simpler than the procedure for the gammatone filter. For these reasons, in this work, we make use of solely decaying sinusoids.

2.2.1 Onset model LSSM

A m-channel decaying sinusoid $\tilde{y}_i$ is defined by

$$\tilde{y}_i = \alpha\gamma^{\ell}\cos(\Omega\ell + \phi), \tag{2.6}$$

where $\Omega \in (0, 2\pi) \setminus \pi$ is the frequency, ϕ is the phase, $\gamma \in \mathbb{R}$ is the decay and $\alpha \in \mathbb{R}^m$ is the amplitude.

Therefore, a decaying sinusoid is defined as the product of a sinusoid and an exponential function. Applying the product of two LSSMs property, presented in the Section 2.3.2 of [1], we arrive at the following resulting LSSM:

$$A = \gamma\begin{pmatrix}\cos(\Omega) & -\sin(\Omega)\\ \sin(\Omega) & \cos(\Omega)\end{pmatrix} \tag{2.7}$$

$$s = \begin{pmatrix}1 & 0\end{pmatrix}^{\mathsf{T}} \tag{2.8}$$

$$C = \alpha\begin{pmatrix}\sin(\phi) & -\cos(\phi)\end{pmatrix}. \tag{2.9}$$

Note, from equations (2.7) and (2.9), that the decay and the frequency of the decaying sinusoid are defined in the matrix A, while the amplitude and the phase are encoded in C.

To localize the onset model fit, in this work, we use a gamma window, defined by

$$w_\ell = \begin{cases} \beta|\ell|^{\nu-1}\gamma^{|\ell|} & \text{if } \ell \leq 0 \\ 0 & \text{if } \ell > 0, \end{cases} \tag{2.10}$$

where ν is a positive integer and $\beta > 0$ is a scale factor.

2.2.2 Fitting a decaying sinusoid

One of the key advantages of using LSSMs as a modeling framework is the fact that the fitting procedure can be done recursively, via message passing. In the Section 3.2 of [1], the cost function, the messages and the optimal solution for the onset model fitting procedure are discussed in detail.

It is important to note that fitting a decaying sinusoid consists in finding the best phase and amplitude of the sinusoid, both of which are encoded in C.

The window functions we consider in this chapter can also be represented as LSSMs $\{A_w, C_w, s_w\}$ of order n_w, hence, the cost can be written as

$$J_k(C) = \sum_{i=1}^{k} C_w A_w^{k-i} s_w \left\| y_i - CA^{k-i}s \right\|_2^2. \tag{2.11}$$

Furthermore, Equation (2.11) can be re-written as

$$J_k(C) = C_w \overrightarrow{\chi}_k - 2\mathrm{tr}((C \otimes C_w)\overrightarrow{\zeta}_k) + \mathrm{tr}((C \otimes C_w)\overrightarrow{s}_k C^\intercal), \tag{2.12}$$

where $\overrightarrow{\chi}_k \in \mathbb{R}^{n_w}$, $\overrightarrow{\zeta}_k \in \mathbb{R}^{nn_w \times m}$ and $\overrightarrow{s}_k \in \mathbb{R}^{nn_w \times n}$ can be efficiently with the following forward recursions

$$\begin{aligned} \overrightarrow{\chi}_k &= \sum_{i=1}^{k} A_w^{k-i} s_w \|y_i\|_2^2 \\ &= A_w \overrightarrow{\chi}_{k-1} + s_w \|y_k\|_2^2 \end{aligned} \tag{2.13}$$

$$\begin{aligned} \overrightarrow{\zeta}_k &= \sum_{i=1}^{k} (A \otimes A_w)^{k-i}(s \otimes s_w) y_i^\intercal \\ &= (A \otimes A_w)\overrightarrow{\zeta}_{k-1} + (s \otimes s_w) y_k^\intercal \end{aligned} \tag{2.14}$$

$$\begin{aligned} \overrightarrow{s}_k &= \sum_{i=1}^{k} (A \otimes A_w)^{k-i}(s \otimes s_w) s^\intercal (A^{k-i})^\intercal \\ &= (A \otimes A_w)\overrightarrow{s}_{k-1} A^\intercal + (s \otimes s_w) s^\intercal. \end{aligned} \tag{2.15}$$

The cost function can be further simplified and re-written as

$$J_k(C) = \vec{\kappa}_k - 2\mathrm{tr}(C\vec{\xi}_k) + \mathrm{tr}(C\vec{W}_k C^\intercal), \tag{2.16}$$

where $\vec{\kappa}_k \in \mathbb{R}$, $\vec{\xi}_k \in \mathbb{R}^{n\times m}$ and $\vec{W}_k \in \mathbb{R}^{n\times n}$ are defined as

$$\vec{\kappa}_k = C_w \vec{\chi}_k \tag{2.17}$$

$$\{\vec{\xi}_k\}_{j,p} = \mathrm{tr}((P_{p,j}^{(m,n)} \otimes C_w)\vec{\zeta}_k) \tag{2.18}$$

$$\{\vec{W}_k\}_{j',p'} = \mathrm{tr}((P_{p',j'}^{(n,n)} \otimes C_w)\vec{s}_k), \tag{2.19}$$

where $P_{i,j}^{(m,r)}$ is the $m \times r$-matrix with a one at index (i, j) and zeros everywhere else.

The scenario we encounter when fitting LSSMs in this work is that of an unconstrained optimization problem, where the feasible set $\mathcal{C}$ is equal to $\mathbb{R}^{m\times n}$. Therefore, we can simply derive (2.16) with respect to C and set it equal to zero, which results in

$$\hat{C} = (\vec{W}_k^{-1}\vec{\xi}_k)^\intercal. \tag{2.20}$$

2.2.3 Local cost ratio (LCR)

With the onset detection module, we would like to detect the instants when sound onsets are occurring. To do so, we define the local cost ratio (LCR), which is a local measure of the goodness of fit of our onset model if compared to the zero-signal model. The underlying idea is that in the moments when a sound onset is present, our onset model fit will be significantly better than fitting the zero-signal, while in the moments where there are no sound onsets, the zero-signal will likely be better than fitting the onset model. Therefore, we can see the LCR as a sparse signal, which serves as a proxy for the detection of sound onsets: when the LCR is high, an onset is being detected and when it is low and almost null, there are none.

For every sample k, as we perform the onset model fits, we compute the LCR as

$$\mathrm{LCR}_k = -\frac{1}{2}\log\frac{\min_{C\in\mathcal{C}} J_k(C)}{J_k(0)}. \tag{2.21}$$

Note that in the way the LCR is defined, it is always non-negative.

2.2.4 Multiple frequencies

In [1], we worked with onset models that contained a single frequency, but, in order to make sure that we robustly detect the occurrence of sound onsets in a wider set of audio signals, we extend the basic modeling framework and work with onset models that contain multiple frequencies.

There are two basic approaches to incorporate the use of multiple frequencies into our modeling framework. The first is by fitting a single onset model to the audio signal, but this onset model is comprised of a linear combination of multiple onset models, each with a different frequency and, possibly, different decays. The second is by fitting multiple onset models in parallel, each with a distinct frequency, forming a filter bank, producing one LCR per frequency used.

Let's start by considering the linear combination of onset models with different frequencies. In this scenario, we can use the sum of LSSMs property, presented in the Section 2.3.1 of [1], and simply stack the models of each frequency in the correct manner. The fit for decaying sinusoid models is unconstrained, so the fitting procedure remains exactly the same as before.

For the filter bank approach, no modification to the fitting procedure is required. We simply fit each onset model frequency in parallel, producing one LCR pair per frequency in the filter bank. The differences in our algorithm to support the multiple LCR pairs are done in the delay estimation module, as discussed in Chapter 3.

2.3 Local Polynomial Fitting

The LCRs indicate the presence of sound onsets. The underlying idea of the proposed algorithm is estimating the delay from them when the sound onsets are detected. To do so, we introduce an intermediate step, where we locally approximate the LCRs with 3^{rd} degree polynomials, to facilitate the next steps in the signal processing algorithm.

2.3.1 Polynomial LSSM

The 3^{rd} degree polynomials that we use to locally approximate the LCR signals have a LSSM representation given by

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.22}$$

$$s = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}}. \tag{2.23}$$

The output vector C contains the polynomial coefficients.

Note that using the LSSM trajectory as defined in Equation (2.2), the polynomial coefficients are given in the binomial basis $(\binom{k}{0}, \binom{k}{1}, \binom{k}{2}, \dots)$. If we wish to work with the canonical basis $(1, k, k^2, \dots)$, the output vector found with the fitting procedure must be multiplied with a transformation matrix. The justification for this can be found in [8] and, the transformation matrix used in this work can be found in Equation (4.3) of [1].

2.3.2 Locally Fitting Polynomials

In this work, to locally fit the polynomials, we use a rectangular window. Let $a < 0$ and $b > 0$ be the window borders for a rectangular window centered at sample k. Let $\text{LCR}_k \in \mathbb{R}^2$ be comprised of the LCR samples at instant k. We process a LCR pair at the same time, considering it a two channel signal. We also consider the signal to be generated by a LSSM moving forward in time. The value of the cost function at sample k is, given by

$$J_k(C) = \sum_{i=a}^{b} \left\| \text{LCR}_{i+k} - CA^i s \right\|_2^2. \tag{2.24}$$

The cost can then be re-written as in (2.16), with $\overrightarrow{\kappa}_k$ and $\overrightarrow{\xi}_k$ defined by Equations (2.25) and (2.26), that are easily calculated recursively. The message $\overrightarrow{W}_k$, given by (2.27), can be pre-computed, as it does not depend on k.

$$\begin{aligned} \overrightarrow{\kappa}_k &= \sum_{i=a}^{b} \|\text{LCR}_{i+k}\|_2^2 \\ &= \overrightarrow{\kappa}_{k-1} - \|\text{LCR}_{k+a-1}\|_2^2 + \|\text{LCR}_{k+b}\|_2^2 \end{aligned} \tag{2.25}$$

$$\begin{aligned}\vec{\xi}_k =& \sum_{i=a}^{b} A^i s(\mathrm{LCR}_{i+k})^\intercal \\ =& A^{-1}(\vec{\xi}_{k-1} - A^a s(\mathrm{LCR}_{k+a-1})^\intercal + A^{b+1} s(\mathrm{LCR}_{k+b})^\intercal)\end{aligned} \tag{2.26}$$

$$\vec{W}_k = \sum_{i=a}^{b} A^i s s^\intercal (A^i)^\intercal, \tag{2.27}$$

The optimal value $\hat{C}$ of C is given by (2.20), since the optimization problem we are solving is unconstrained.

Chapter 3

Delay Estimation

The last block in the proposed algorithm's chain is the delay estimation module. This module is the one responsible for receiving the local polynomial fit coefficients and using them to output the delay estimates. If compared to the version of the algorithm presented in [1], the delay estimation module is the one that changed the most.

In this chapter, we present the vanilla version of the module, which is similar to the version presented in [1], except for the extension to work with the fits from multiple onset model frequencies in parallel. Then, we discuss its limitations that led to the development of the confidence version of the algorithm, which produced significantly better results on a multitude of speech signals, as discussed in Chapter 4.

3.1 Vanilla version

The algorithm proposed in this work relies on the precedence effect to produce the delay estimates. In practical terms, this means that the delay estimates are produced only at specific instants, when a sound onset is detected, since these are the parts of the audio signal that contain accurate timing information. The LCRs indicate the presence of a sound onset in each signal, so the objective is to extract the delay information from them.

The LCRs are sparse signals and we wish to trigger the delay estimation module when both LCRs (i.e., from the left and right channels) are peaking, signaling that sound onsets are present in both channels. What we chose to do in [1] is to produce the delay estimates when both LCRs are rising to form a spike.

Since we locally approximate the LCRs by 3rd degree polynomials, determining if the LCRs are rising is a straightforward task. At instant k, a polynomial is rising if its first derivative evaluated at k is large, which indicates that the signal level is rapidly increasing, and its second derivative evaluated at k is small, meaning that the rise is almost linear. For each time step k, we get the polynomials $f_k^{(L)}(\ell)$ and $f_k^{(R)}(\ell)$, centered at sample k, which approximate the LCRs from the left and from the right channels, respectively, in the form of (3.1) and (3.2).

$$f_L^{(k)}(\ell) = \alpha_0^{(k)} + \alpha_1^{(k)}(\ell - k) + \alpha_2^{(k)}(\ell - k)^2 + \alpha_3^{(k)}(\ell - k)^3 \tag{3.1}$$

$$f_R^{(k)}(\ell) = \beta_0^{(k)} + \beta_1^{(k)}(\ell - k) + \beta_2^{(k)}(\ell - k)^2 + \beta_3^{(k)}(\ell - k)^3 \tag{3.2}$$

Thus, to determine whether or not the polynomials are rising at instant k, for every sample, we check if $\alpha_1^{(k)}$ and $\beta_1^{(k)}$ (i.e., the first derivatives) are above a threshold τ_1 and if $2\alpha_2^{(k)}$ and $2\beta_2^{(k)}$ (i.e., the second derivatives) are below a threshold τ_2. If both conditions are satisfied, that means that both LCRs are rising at instant k and, hence, we can proceed with the delay estimation procedure.

So far, we know the instant when both LCRs are rising, meaning that we are at the beginning of a peak of the LCRs and, thus, at a good instant to estimate the delay, since this indicates the presence of a sound onset. What we need to do now is extract the delay information from the LCRs.

To do so, we locally approximate the 3rd degree polynomials around the sample k by 2nd degree polynomials $g_L^{(k)}(\ell)$, with coefficients $a_0^{(k)}, a_1^{(k)}$ and $a_2^{(k)}$, and $g_R^{(k)}(\ell)$, with coefficients $b_0^{(k)}, b_1^{(k)}$ and $b_2^{(k)}$. The 2nd degree polynomial coefficients are found by solving a system of linear equations. We simply choose three points on the 3rd degree polynomial, evaluate them there and solve the linear system (more specifically, we choose k, $k - w$ and $k + w$, where w is smaller than the window length used for the 3rd degree polynomial fit). The coefficients for the 2nd degree polynomial to the left found by solving Equation (3.3). Analogously, one can find the values of the coefficients for the 2nd degree polynomial to the right.

$$\begin{pmatrix} a_0^{(k)} \\ a_1^{(k)} \\ a_2^{(k)} \end{pmatrix} = \begin{pmatrix} 1 & -w & w^2 \\ 1 & 0 & 0 \\ 1 & w & w^2 \end{pmatrix}^{-1} \begin{pmatrix} f_L^{(k)}(k-w) \\ f_L^{(k)}(k) \\ f_L^{(k)}(k+w) \end{pmatrix} \tag{3.3}$$

To estimate the delay from the 2nd degree polynomials, we check which polynomial arrived first at sample k and then solve the equation that tells us when the polynomial that arrived later reaches the same level as the one that arrived first, as shown in Equation (3.4). This equation has at most one solution that makes sense and this is the delay estimate.

$$g_{\text{first}}^{(k)}(k) = g_{\text{last}}^{(k)}(\ell) \tag{3.4}$$

The pseudo-code for the vanilla version of the algorithm can be seen in Algorithm 1.

Algorithm 1: Delay estimation algorithm.

for *every sample* k **do**
  if $\min(\alpha_1^{(k)}, \beta_1^{(k)}) > \tau_1$ *and* $2\max(\alpha_2^{(k)}, \beta_2^{(k)}) < \tau_2$ **then**
    Approximate $f_L^{(k)}(\ell)$ and $f_R^{(k)}(\ell)$ with $g_L^{(k)}(\ell)$ and $g_R^{(k)}(\ell)$, respectively ;
    Check which 2^{nd} degree polynomial arrived first and name it $g_{\text{first}}^{(k)}(\ell)$. The other polynomial is named $g_{\text{last}}^{(k)}(\ell)$;
    Solve $g_{\text{last}}^{(k)}(\ell) = g_{\text{first}}^{(k)}(k)$, resulting in roots ℓ_1 and ℓ_2 ;
    if $\ell_1 < 0$ *and* $\ell_2 > 0$ **then**
      $delay[k] = \ell_2$;
    else if $\ell_1 > 0$ *and* $\ell_2 < 0$ **then**
      $delay[k] = \ell_1$;
    else if $\ell_1 > 0$ *and* $\ell_2 > 0$ **then**
      $delay[k] = \min(\ell_1, \ell_2)$;
  else
    $delay[k] = delay[k-1]$;

In Figure 3.1, taken from [1], the steps of the vanilla version of the algorithm are visualized. Figures 3.1a and b show an instant when both LCRs are rising and the local polynomial fit performed to locally approximate them. We notice that the two LCRs are rising by checking the threshold conditions on the first and second derivatives of the 3rd degree polynomials. Then, as shown in Figure 3.1c, the 3rd degree polynomials are locally approximated by 2nd degree polynomials using a smaller window. Note that for the 3rd degree fits, a rectangular window of length equal to 201 samples was used, while for the 2nd degree fits, a window length of 81 samples was used. The 2nd degree polynomials, shown in Figure 3.1c are the ones used to output

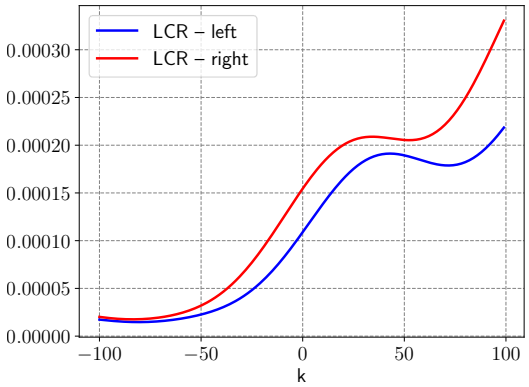
the delay estimate. Note that the polynomial fit to the right channel arrives first and that the 2^{nd} degree polynomial of the left channel reaches the same level as the one from the right in roughly 20 samples, if we pick $k = 0$ as the reference point.

3.1.1 Multiple frequencies

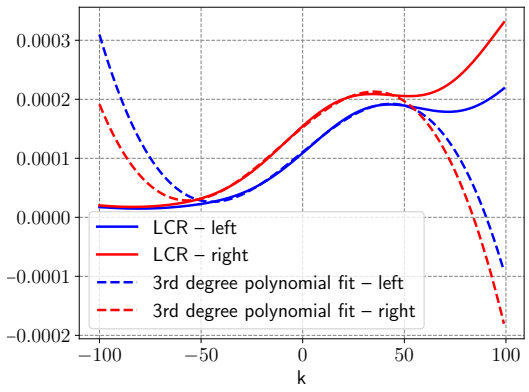
The vanilla version previously presented is the same as the one first presented in [1]. The algorithm, in the way that it is defined in Algorithm 1, works with the 3^{rd} degree polynomial fits from a single LCR pair. This is the case when we work with a single onset model fit (with one or multiple frequencies). Thus, we need to make a few modifications on the algorithm presented to work with multiple LCR pairs, which is the case when we fit multiple onset models with different frequencies in parallel, forming a filter bank.

Fortunately, the extension of the vanilla version to work with multiple frequencies is straightforward. The underlying idea is the following: when we have a filter bank resulting in multiple LCR pairs, we are more likely to have good LCRs to estimate the delay from, since each LCR pair was obtained from fitting onset models with different frequencies and we are more likely to have a fit that matches the base frequency of the audio signal we encounter. We therefore need a method to select a "good" LCR pair to estimate the delay from. What we can do is simply check the first and second derivative conditions for all the LCR pairs that we have available and once it is satisfied, proceed with the delay estimation procedure normally, but working with the LCR pair selected at that instant.

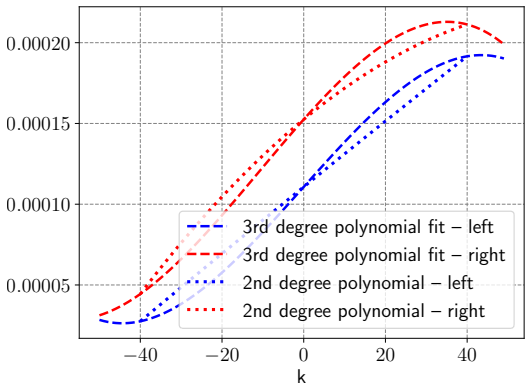
There are two important considerations to be made with such an approach. The first one is that a selected LCR pair that fulfilled the derivative conditions is not necessarily always the most reliable pair to estimate the delay from. In fact, in a real-world scenario, where multiple audio signals arrive at different times, we expect the best LCR to change over time. Therefore, it is important to define a forgetting period, after which we go back to monitor all the fits. The second consideration is that in case more than one LCR pair satisfies the threshold criteria at the same time. We solve this issue by adding a third criteria and we give preference to the LCR with the highest magnitude at that instant.



(a) LCRs from the left and from the right starting to rise as an onset is detected.



(b) Local 3rd degree polynomial fits to the LCRs.



(c) 2nd degree approximation for delay estimation.

Figure 3.1: Delay estimation algorithm steps.

3.1.2 Limitations

We started this project working with the vanilla version with multiple frequencies, but soon found severe limitations with it when working with a variety of speech signals, as described in Chapter 4.

The first issue with the vanilla version of the algorithm is that we require the LCR pair to be rising at the same instant. For the order of magnitude of the delays we are trying to estimate, this is a reasonable requirement, but for larger delays, the vanilla version would clearly not work, due to the fact that the LCR peaks in each channel might be more distant from each other and there would be no single instant when both of them are rising.

The second major issue is that the conditions that we set to trigger the delay estimation module are too loose. We set hard thresholds for the first and second derivatives, which corresponds to constraining the coefficients that multiply the first and second power terms and we do not have any conditions for the remaining coefficients. There are infinitely many polynomials that satisfy the two criteria that we set, but look very different from each other. This implies that we potentially end up estimating the delay from two completely different polynomials, which is not the right thing to do. Aspiring to overcome the limitations of the vanilla version of the algorithm, we propose the confidence version, presented in the next section.

Note however, that despite its limitations, the vanilla version of the algorithm works reasonably well most of the times, as the delay estimates produced by it are not too far from the simplified theoretical delays for the scenario we investigate.

3.2 Confidence version

The first thing we notice is that the main issue with the vanilla version has to do with the fact that we are potentially estimating a delay from two different polynomials instead of actually estimating a delay from two versions of the same polynomial (or at least polynomials that are very similar to each other and can be thought out of as being equal).

The 3^{rd} degree polynomials that locally approximate the LCRs have four parameters, namely the four coefficients. Polynomials that are similar to each other have similar coefficients. Therefore, if we think about each polynomial as a vector defined by its coefficients in $\mathbb{R}^4$, polynomials that are

similar to each other are points close to each other in $\mathbb{R}^4$.

With our prior knowledge about the shape of the LCRs, we know that the polynomials that live in certain regions of $\mathbb{R}^4$ are better candidates to estimate the delay from than polynomials that live in other regions. For example, polynomials that are close to the origin do not seem to be good candidates, as they are, for the most part, approximating the regions of the LCR that are almost constant, close to zero. On the other hand, the region in $\mathbb{R}^4$ that correspond to polynomials with a positive intercept, large first derivative and small second derivative seem to be a better region to search for candidate polynomials to estimate the delay from, as they are the polynomials locally approximating the LCR spikes.

There is an additional advantage of thinking of the polynomials as points in $\mathbb{R}^4$. When we choose two polynomials, we can quantify their similarity with a metric such as the inverse of their Euclidean distance. This measure of similarity can serve as a notion of confidence that we have when estimating the delay from two polynomials, to make sure that we are comparing two similar polynomials and not two arbitrarily dissimilar polynomials that satisfied a loose set of criteria.

To gain a bit more of intuition of what is really going on, we can plot some of the polynomial coefficients approximating a LCR pair, shown in Figure 3.2. We obtain Figure 3.2 by first generating a LCR pair by fitting an onset model with a single frequency to a binaural speech signal, then, locally approximating the LCR signals by 3^{rd} degree polynomials and by noticing that for the whole duration of the signal, the coefficient that multiplies the third power term has a variance significantly smaller than the other coefficients. Hence, we plot each polynomial as a point in $\mathbb{R}^3$, getting rid of this last coefficient. For visualization purposes, we also standardized each coefficient to have zero mean and unit variance.

Figure 3.2 has many interesting interpretations that are valuable for the design of a better delay estimation procedure. The first noticeable feature is that the coefficients define intricate trajectories in this space of coefficients. The trajectories defined by the local polynomial fits to the left and right LCRs clearly correspond, especially in the regions farther away from the origin, which accounts for the rising and falling of the major peaks in the LCR signals. Another noticeable characteristic is that the trajectories are for the most part of the signal, defined around the origin. This is due to the fact

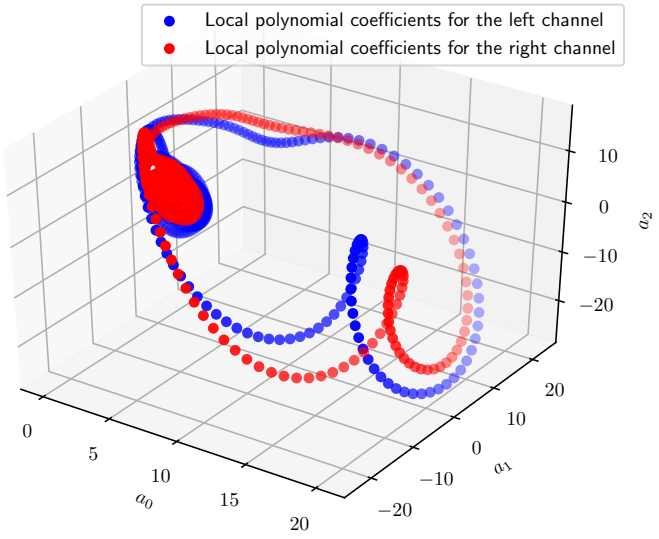


Figure 3.2: Trajectories defined by the polynomial coefficients in $\mathbb{R}^3$.

that the LCRs are sparse, so for the parts of the LCR that are mostly close to zero, the local polynomial fits also have small coefficients. The observation that inspired the confidence version of the delay estimation module is the fact that sometimes, the trajectories defined by the left and right signals intercept, meaning that they have very similar polynomials and those are the polynomials that we wish to estimate the delay from.

With the previous ideas as building blocks, we can formulate a new version of the delay estimation module that overcomes the limitations of the vanilla version. The idea is that we constantly monitor the coefficients inside a window that gets updated as new coefficients arrive. Then, we want to possibly find two polynomials (one from the left and one from the right signal) to compare and estimate the delay from. As we motivated previously, we want to compare polynomials that are similar. Therefore, what we do is compute the Euclidean distance between the coefficients from the left and from the right, resulting in a square matrix $D \in \mathbb{R}^{w \times w}$, where w is the window length, where the element $D_{i,j}$ corresponds to the Euclidean distance between the i^{th} polynomial in the left to the j^{th} polynomial in the right. We want to compare the two most similar polynomials, so we pick the two

polynomials that produce the smallest entry in D.

We do not want to compare any two polynomials, since we believe, *a priori* that some regions of $\mathbb{R}^4$ seem to be better than others to estimate the delay. We check if both polynomials belong to a good region in $\mathbb{R}^4$. In our case, we simply checked the same conditions that we were checking for the vanilla version, namely, if both polynomial's first derivative was large enough and if their second derivatives were small enough.

If both polynomials satisfy the above condition, we proceed with the delay estimation. Otherwise, we simply move on to the next sample and update the window.

In this version of the algorithm, we estimate the time delay between the two chosen polynomials using the method presented in [cite]. To compute a delay estimate, we minimize the squared error between the two polynomials we compare while allowing a time shift. The delay is computed as the time shift that minimizes the squared error between the polynomials.

Let k_L and k_R be the center samples for the polynomials from the right and from the left and let $f_L^{(k_L)}(\ell)$ and $f_R^{(k_R)}(\ell)$ be the polynomials from the left and from the right, respectively, that we wish to estimate the delay from. First, we have to take into account the fact that those two polynomials are probably not defined over the same time axis. Thus, we need to first adjust the two polynomials' coefficients so that we can properly compare them, namely

$$\begin{aligned} & f_L^{(k_L)}(\ell) = \alpha_0^{(k_L)} + \alpha_1^{(k_L)}(\ell - k_L) + \alpha_2^{(k_L)}(\ell - k_L)^2 + \alpha_3^{(k_L)}(\ell - k_L)^3 \\ \implies & f_L(\ell) = \alpha_0 + \alpha_1 \ell + \alpha_2 \ell^2 + \alpha_3 \ell^3, \end{aligned} \tag{3.5}$$

where

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} 1 & -k_L & k_L^2 & -k_L^3 \\ 0 & 1 & -2k_L & 3k_L^3 \\ 0 & 0 & 1 & -3k_L \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_0^{(k_L)} \\ \alpha_1^{(k_L)} \\ \alpha_2^{(k_L)} \\ \alpha_3^{(k_L)} \end{pmatrix}. \tag{3.6}$$

The analogous transformation is applied for the coefficients from the polynomial from the right.

With the polynomials' coefficients properly transformed, we can compare them and estimate the delay by minimizing Equation (3.7), which represents the squared error between the two polynomials across the interval fro a to b when a time shift of s is applied.

$$J(s) = \int_a^b \left[f_L\left(\ell - \frac{s}{2}\right) - f_R\left(\ell + \frac{s}{2}\right) \right]^2 d\ell \tag{3.7}$$

In [9], it is shown that the cost defined by Equation (3.7) can be transformed into an univariate polynomial as well, so the problem of finding

$$s^* = \underset{s}{\operatorname{argmin}}\, J(s) \tag{3.8}$$

reduces to finding the argmin of a polynomial. The full derivation of such result can be found in [9, 10].

Alternatively, to estimate the delay estimate, one could also simply define the delay as $k_L - k_R$. This choice of delay estimate produces very similar results to the case when the delay is found via (3.8).

We not only compute the delay estimate, but we also record the confidence we have in such an estimate. We define the confidence as being the inverse of the Euclidean distance between the polynomials we compare. This metric is justified by the fact that we are more confident that we are producing a good delay estimate if we are comparing similar polynomials, i.e., polynomials that are close to each other in $\mathbb{R}^4$.

The pseudo-code for this version of the algorithm can be seen in Algorithm 2.

3.2.1 Multiple frequencies

With the procedure outlined in Algorithm 2, we are able to produce delay estimates for a single LCR pair. Now, we present the modifications needed to work with multiple LCR pairs, such as in the filter bank scenario. In Algorithm 2, we produce independent delay estimates for each LCR pair. What is particularly convenient, though, is the fact that besides the delay estimates produced from each LCR pair, we also have a measure of confidence in each estimate. Therefore, to use multiple frequencies, we can simply switch between the delay estimates produced based on their confidence. After we estimate the delay once, we only change our estimate if the new estimate has a higher confidence than the one that was associated with the one we produced previously.

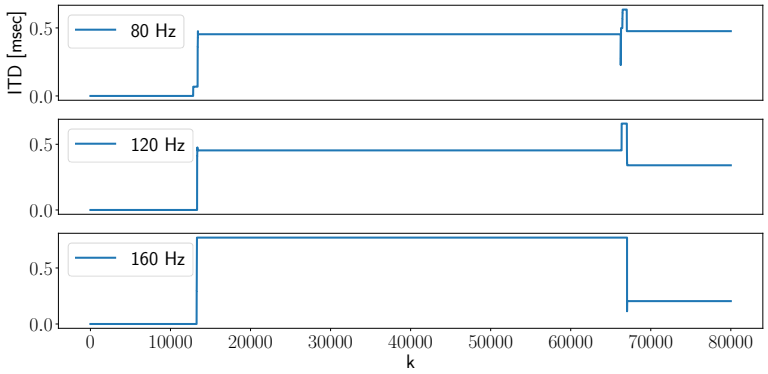
What we observed empirically is that although the first delay estimate produced by the algorithm might not be particularly good, as new samples

Algorithm 2: Confidence version of the delay estimation algorithm.

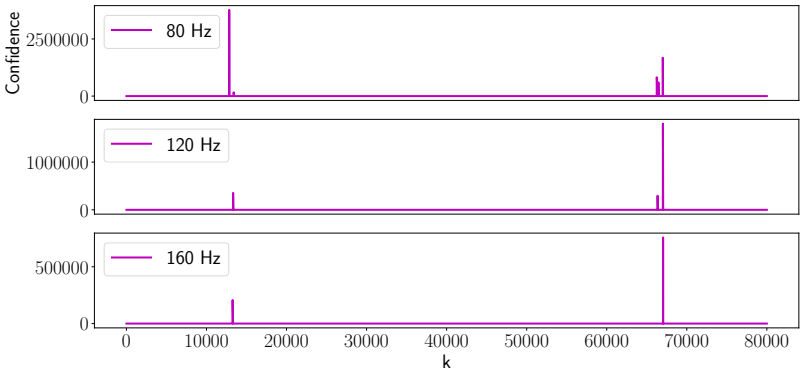
Let w be the window length that we chose to monitor;
for *every* k **do**
  Update the window with the last $w-1$ coefficients and with the current coefficients for the polynomials from the left and from the right ;
  Calculate the matrix of pairwise Euclidean distances between the coefficients from the left and from the right windows ;
  Get the two polynomials that resulted in the minimum distance in the matrix ;
  if *Both polynomials' coefficients satisfy the first and second derivative criteria* **then**
    Estimate the delay as $delay[k] = k_L - k_R$ or as $delay[k] = \text{argmin}_s\, J(s)$;
    Compute the confidence on the delay estimate as the inverse of the Euclidean distance between the chosen polynomials' coefficients;
  else
    $delay[k] = delay[k-1]$;

arrive, we are capable of replacing those estimates with better ones in the future.

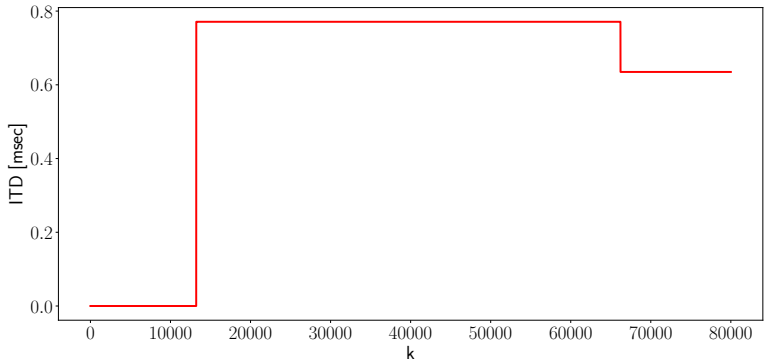
In Figure 3.3, it is possible to visualize the delays estimates produced by each frequency in the filter bank with the confidences associates with each estimate. Note that the final delay estimates produced by the algorithm are much more stable than the individual estimates produced at each frequency due to the use of the confidence. It is important to mention that the plots in Figure 3.3 serve the purpose of illustrating the working principle of the algorithm when dealing with multiple frequencies. In the actual simulation used to generate the delay estimates shown, more frequencies were present in the filter bank, so the final delay estimates might not correspond to the estimates produced by one of the three frequencies shown in Figure 3.3.



(a) Delay estimates for each frequency in the filter bank.



(b) Confidences associated with each delay estimate produced.



(c) Final delay estimates outputed by the confidence version of the algorithm.

Figure 3.3: Delay estimates with associated confidences.

3.2.2 Limitations

We have empirically found that using the confidence version improved significantly the results produced by the algorithm if compared to the vanilla version, as it is extensively discussed in Chapter 4, but the confidence version still has one important drawback.

In this version of the algorithm, we measure the similarity between two polynomials by computing the Euclidean distance between their coefficients. The coefficients define the polynomials across their whole domain, but we do not necessarily want to compare the two polynomials globally, on the full domain. The polynomials were used to locally fit the LCR signals, thus, it makes sense to compare them only locally with each other, within the window in which they were defined.

Two polynomials can be very similar within their respective windows, but significantly different outside it. In this case, their Euclidean distance would be large, even though they might be good candidates to estimate the delay from.

To overcome this limitation, we can re-define the measure of distance in the confidence version of the algorithm from a global measure of distance, such as the Euclidean distance, to a more localized measure. One possibility of local measure would be integrating the squared difference between the two polynomials over a defined window of length comparable to the original window used to fit the polynomials. We have

$$d = \int_a^b [f_L(\ell) - f_R(\ell)]^2 d\ell. \tag{3.9}$$

Note that Equation (3.9) corresponds to (3.7), evaluated at $s = 0$. Therefore, the integral defined in (3.9) can also be computed efficiently.

Chapter 4

Experimental results

In [1], we conducted experiments with two speech signals, one from a female and one from a male speaker, but this is not enough to guarantee that the algorithm properly works in real-world situations, where speech signals from a multitude of speakers are encountered. Thus, suggestions for future work included applying modifications that would make the algorithm more robust to a wider class of speech signals and properly assess its capabilities. This implements the suggested modifications, as presented in the previous chapters. In this chapter, we propose a validation procedure that is used to assess and optimize the algorithm's parameters so that it works well for multiple speech signals. Our algorithm's performance is also compared to the performance of the classical ITD estimation procedures, discussed in Chapter 1, as well as the theoretical estimate of the ITD based on a simplified physical model of the scenario.

4.1 Binaural speech signal corpus

Having a corpus of speech signals that follow a distribution similar to the distribution expected to be encountered in the real-world is of utmost importance to assess our algorithm's capabilities. Therefore, this work explored a range of popular HRIR and speech corpora to validate the proposed algorithm.

4.1.1 CIPIC HRTF database

The CIPIC HRTF database [11] contains high-resolution HRTFs for multiple subjects, including the KEMAR mannequin. For each subject, there are HRIR pairs available for 25 different azimuths and 50 elevations.

In the present work, we use the HRIR pairs with the single purpose of simulating binaural audio signals from single channel signals, to construct the binaural speech signal corpus. Therefore, among all the 45 different subjects for which the HRIRs are made available, we pick a single one, namely, the HRIRs for the KEMAR mannequin with small pinnae. Furthermore, we are interested in performing sound source localization in the horizontal plane, so among the HRIRs pairs for the chosen subject, we only pick the ones corresponding to the $0°$ elevation.

In the end, that leaves us with 25 HRIRs pairs for the KEMAR mannequin with small pinnae. The 25 HRIRs pairs correspond to the azimuths considered in the horizontal plane. The azimuth ranges from $-80°$ to $80°$, sampled at $-80°, -65°, -55°$, from $-45°$ to $45°$ in increments of $5°$ and at $55°, 65°, 80°$. Additionally, note that the HRIRs from the CIPIC database are sampled at 44.1 kHz.

By convolving the HRIR pairs with single channel speech signals, we are able to simulate binaural speech signals, which are the input to the algorithm proposed in this work.

4.1.2 TIMIT acoustic-phonetic continuous speech corpus

The TIMIT speech corpus [12] is an extensive dataset containing ten phonetically rich sentences read by 630 different speakers. All the speakers read the sentences in English, spanning the eight major American English dialects. The speech signals are sampled at 16 kHz.

The TIMIT corpus was the first candidate considered by the authors of the present work to serve as a basis to construct the binaural speech signal corpus. It was seen as a strong corpus candidate due to the variety of speakers and sentences, leading to the potential of building a corpus that represents well the distribution of speech signals that can be encountered in the real world, at least when it comes to American English speakers.

To select from the total of 6300 speech signals to build a balanced corpus, we designed a set of selection criteria to construct the binaural speech corpus from the TIMIT corpus. We have 25 HRIR pairs, representing the 25 different

azimuths in the horizontal plane considered. Therefore, for each azimuth, we wanted to select the speech signals from the TIMIT corpora that we would convolve the HRIR pair with to generate the binaural signals. To do so, ensuring a balance of male and female speakers within each azimuth, we proceeded according to Algorithm 3.

Algorithm 3: Building the binaural speech dataset from the TIMIT speech signals

for *each azimuth in* $-80°, -55° \ldots, 80°$ **do**
  Select one of the 8 dialect regions uniformly at random;
  Pick uniformly at random 5 male speech signals from the selected dialect region;
  Pick uniformly at random 5 female speech signals from the selected dialect region;
  Convolve each selected speech signal with the azimuth's HRIR pair;
end

Following Algorithm 3 results in a binaural speech corpus with 10 binaural speech signals for each azimuth, 5 from female and 5 from male speakers. The sentences contained in each azimuth are selected at random as well. It is important to consider that while the HRIRs are sampled at 44.1 kHz, the speech signals from the TIMIT corpora are sampled at 16 kHz. Therefore, before convolving the HRIR pairs with the speech signals, we downsampled the HRIRs by an integer factor to match the sampling rate of the TIMIT speech signals.

We performed preliminary tests of our ITD estimation algorithm with the binaural speech corpus built over the TIMIT corpus, but we noticed that while we were able to perceive the trend followed by the ITD estimates for each azimuth, their values did not match the theoretical ITD estimates. After investigating if the issue lied in our ITD estimation algorithm or in the speech signals that we chose, we found out that the issue had to do with the downsampling that we were performing with the HRIRs prior to the convolution with the speech signals. By downsampling the HRIRs from 44.1 kHz to 16 kHz, we were losing the the accurate information about the first arriving signal. Our algorithm is constructed to take advantage of the precedence effect and by losing the information about the first arriving version of the signal, the ITD estimates produced did not match the theoretical ones.

This lead us to search for another speech corpus where the speech signals were available at a higher sampling rate.

4.1.3 GRID audiovisual sentence corpus

The GRID corpus [13] contains high-resolution audio and video recordings of 18 male and 16 female speakers, being comprised of a total of 34000 sentences read. The sentences read are, overall, shorter than the sentences read in the TIMIT corpus, but the speech signals are available at a sampling rate of 50 kHz. Since the GRID corpus contained speech signals from multiple female and male speakers and the signals had a higher resolution – so that we would not need to downsample the HRIR from the CIPIC database –, it presented itself as a strong candidate to serve as the basis to build the binaural speech corpus.

The procedure followed to construct the binaural speech signal corpus from the GRID corpus is very similar to Algorithm 3. We are still interested in selecting, for each azimuth, 5 female and 5 male speech signals to convolve the HRIR pairs with. The only modification in Algorithm 3 is that for the signals from the GRID corpus, there is no need to select a dialect region. Thus, what we do is simply select uniformly at random 5 female and 5 male speech signals among all the 34000 signals available, for each azimuth. It is also important to note that due to the fact that the speech signals from the GRID corpus are sampled at 50 kHz, prior to the convolution with the HRIR pairs, we downsampled them to 44.1 kHz, to match the sampling rate of the HRIR pairs from the CIPIC database. Since the downsampling now is done on the speech signal prior to the convolution and not on the HRIR, the accurate timing information of the first arriving sound is not lost.

We also performed numerous experiments running our ITD estimation algorithm with the binaural speech corpus built over the signals from the GRID corpus and, again, we encountered similar issues matching the delay estimates produced by our algorithm and the theoretical ones. Again, we extensively investigated the underlying problem and we found out that the LCRs were peaking right before the actual sound onsets. These LCR peaks were strong enough to trigger the delay estimation module and, as a consequence, we were producing delay estimates based on these regions of the signal right before the sound onsets.

What we noticed was that virtually all the speech signals that we picked

for the corpus had a brief period of noise present immediately before the sound onsets and this noise was responsible for making the LCRs peak. The noise before the sound onsets is known as fricative and is produced due to a burst of air released when pronouncing some consonants. This issue was severe in our case because all the GRID corpus' sentences are of the form "put red at G9 now" [13], where the sentence starts with a fricative inducing consonant.

Although fricatives are indeed encountered in real-world speech signals, their occurrence is not ubiquitous such as it was in our corpus. The process of investigating the issue with the fricatives was extremely useful to identify the limitations of the first version of the algorithm. What was revealed during this process was that the conditions that we were applying to check whether or not we were at a good moment to estimate the delay were too loose and this is partly what made us estimate the delay from those fricatives, which is clearly not the correct thing to do. The LCRs are prone to peaking in instants that are not exactly correct and we should not simply use the first peak of the LCR to estimate the delay, like we were doing in the vanilla version of the algorithm. Additionally, just because the LCR peaks, it does not mean that it contains the timing information that we seek. This realization lead to the development of the further versions of the delay estimation modules (described in Chapter 3), where we further constrain the conditions that trigger the delay estimation module, use a notion of confidence to assess whether or not the delay we estimated is correct and search for good LCR pairs to estimate the delay from.

4.1.4 Segmenting the ISTS

In the search for a corpus that contained speech signals that not always had fricatives, we came across the International Speech Test Signal (ISTS). The ISTS is a test signal that is widely used for the technical assessment of hearing instruments [14]. The speech signal is comprised of multiple speakers reading a passage in six different languages. The speech signal as a whole is unintelligible, due to the segmenting and mixing applied to meld all of the speakers' voices.

The whole signal duration is of 60 seconds and it contains multiple brief moments of pause. By inspecting the audio wave as a whole, we noticed that during some of those moments of pause, there were fricatives, but in others,

there were none. Therefore, we decided to, in a sense, construct a corpus by manually segmenting the ISTS, splitting the full 60 seconds signal into 22 shorter signals, as it can be seen in Figure 4.1.

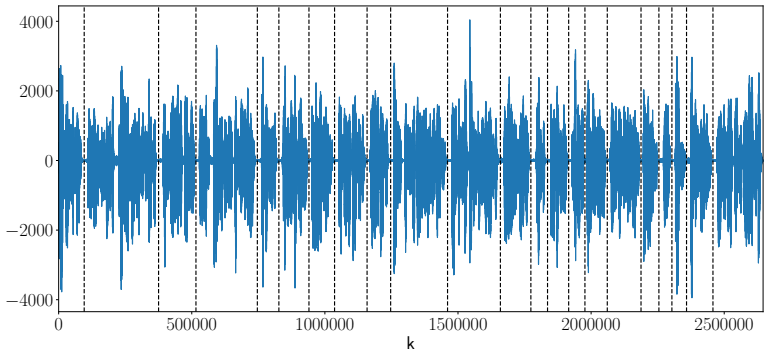


Figure 4.1: Segmenting the ISTS to construct the speech corpus.

The ISTS is already made available at a 44.1 kHz sampling rate, which already matches the sampling rate of the HRIRs from the CIPIC database, so no further processing is required prior to the convolution. We convolved all 22 signals with all the HRIRs pairs, resulting in the same 22 signals for each azimuth.

4.2 Algorithm setup

In [1], we performed experiments with the onset model and local polynomial fits and empirically found parameters that were shown to work well. Moreover, we noted that for different speech signals, a critical parameter to adjust was the frequency of the onset model and the best performance in terms of a sparse LCR signal was found when the frequency of the onset model used matched the base frequency of the speech signal we were fitting. Therefore, in order to have an algorithm that is robust to a wider class of speech signals, we decided to work with multiple frequencies in parallel, forming a filter bank, as described in Chapter 2.

For the onset models in parallel, we used the best parameters found in [1], namely a decaying sinusoid with a gamma window with the parameters shown in Table 4.1, but each model in the filter bank had a distinct frequency. Since we are working with speech signals and the base frequency of the

human voice ranges from 80 to 255 Hz [15], we decided to uniformly partition the base frequency range in 5 parts. Thus, the frequencies present in the filter bank are 80, 120, 160, 200, 240 Hz. All the onset detection parameters can be seen in Table 4.1.

Parameter	**Value**
Filter-bank frequencies	80, 120, 160, 200 and 240 Hz
Onset models	Decaying sinusoids
Onset decays (value/in time)	0.99/1.56 ms
Window models	Gamma
Window decays (value/in time)	0.999/56.32 ms

Table 4.1: Onset detection module parameters.

For the local polynomial fits for each LCR, we use exactly the same parameters as the ones used in [1], namely a rectangular window of length equal to 201 samples and a 3^{rd} degree polynomial fit. The local polynomial fit parameters can be seen on Table 4.2.

Parameter	**Value**
Local polynomial fit	3^{rd} degree polynomial
Window model	Rectangular
Window length	201 samples

Table 4.2: Local polynomial fit module parameters.

Finally, for the delay estimation module, we perform experiments with the three versions presented in Chapter 3. For the vanilla version, which is the one similar to the one presented in [1], we use exactly the same parameters for the first and second derivative thresholds. For the confidence version of the delay estimation module, we picked the Euclidean distance as a measure of similarity between the polynomials and we monitored the polynomials within a rectangular window of length equal to 100 samples. For the integration limits in Equation (3.7), we picked $a = -80$ and $b = 80$, to compute the squared difference between the polynomials we compare on a window smaller than the one used to fit them. All the parameter values are summarized in Table 4.3

Figure 4.2 shows an overview of the block diagram of the algorithm.

Parameter	Value
Thresholds (vanilla version)	$\tau_1 = 1.2\ 10^{-6}$ and $\tau_2 = 10^{-13}$
Window length for the 2nd deg. poly. fit (vanilla version)	81 samples
Distance metric (confidence version)	Euclidean distance
Window length (confidence version)	100 samples
Integration limits (confidence version)	$a = -80$ and $b = 80$

Table 4.3: Delay estimation module parameters.

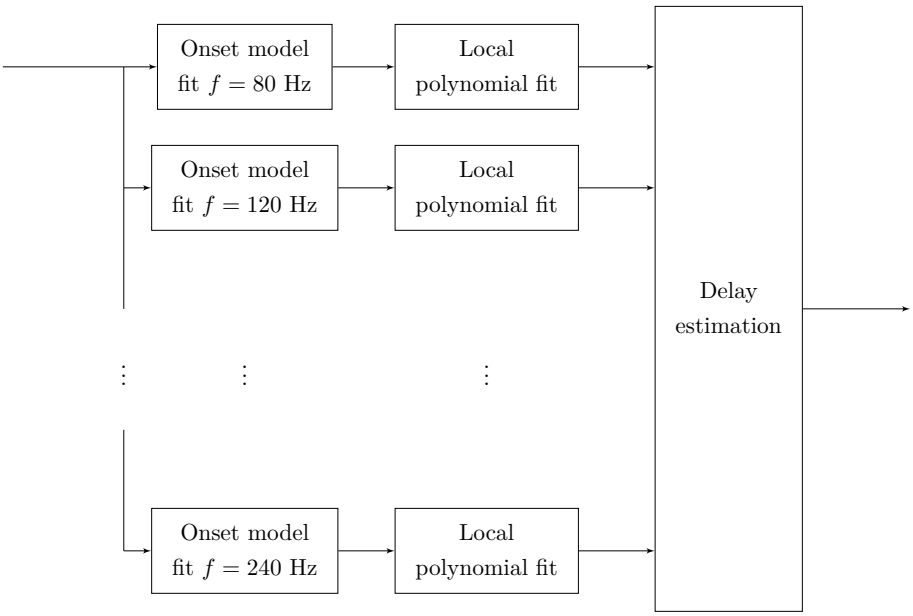


Figure 4.2: Proposed algorithm block diagram.

The algorithm was fully implemented in Python 3.7 using the NumPy package (version 1.19.1) [16].

4.3 Experiments

In this work, we perform a series of experiments with the proposed algorithm with the goal of illustrating its working principles and evaluating its capabilities and limitations against the classical methods of performing ITD estimation. In this section, we first illustrate the algorithm at work, going step by step from the binaural speech signals to the final delay estimates. Then, we run the full algorithm on the binaural speech corpus and compare its performance to the performance of the classical ITD estimation methods,

with a special focus on the comparison with the cross-correlation with frames method.

4.3.1 Algorithm at work

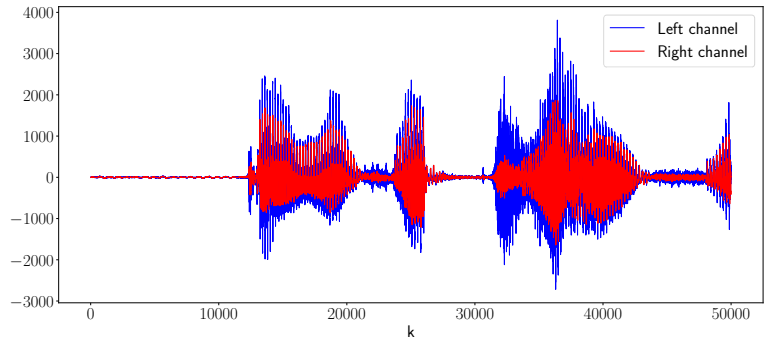
The first two blocks in our algorithm are the same for all the versions presented. These modules are responsible for, first, taking the binaural speech signal samples as an input and producing the multiple LCR pairs in parallel (one pair per onset model frequency) and then locally fitting 3rd degree polynomials to each LCR. Figure 4.3 illustrates the first step.

It is worth noting that not all the LCR pairs shown in Figure 4.3 are equally good to estimate the time delay from. From its definition, the LCR measures the goodness of fit of the onset model to the speech signal and evidently, some onset models are more similar to the actual sound onset of the signal we are currently fitting. For the signal shown, the onset model with frequency equal to 240 Hz seems to be the most appropriate LCR pair to estimate the delay from, while the onset models with lower frequencies, namely 80 and 120 Hz, seem to be more noisy. This fact justifies the use of a filter bank of onset models with different frequencies to make the algorithm more robust to a wider class of speech signal since, for each signal we encounter, an onset model of a different frequency might produce the most appropriate fit.

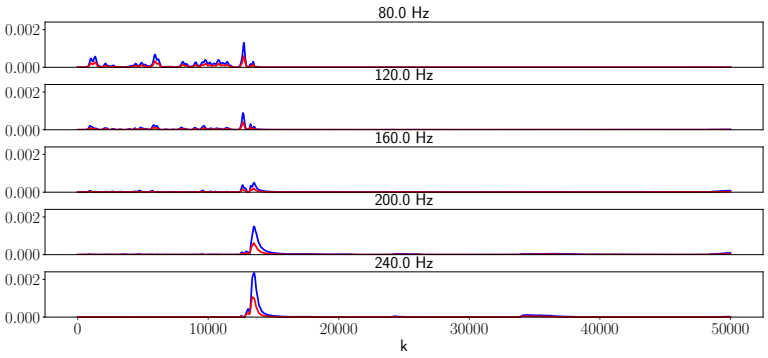
With the local polynomial fits approximating each LCR pair, we perform the delay estimation. The vanilla version of the delay estimation module checks if the polynomial coefficients satisfy the first and second derivative threshold criteria and then, once they are satisfied, perform the delay estimation by further approximating the 3rd degree polynomial with 2nd degree polynomials, as discussed in Chapter 3. The confidence version, on the other hand, produces delay estimates for all the LCR pairs, if they satisfy a set of criteria, and records the confidence in each estimate. To produce the final delay estimate, the algorithm switches between each frequency's estimates based on the confidence recorded.

4.3.2 Validation results

To perform the validation procedure described earlier in Chapter 1, we run both versions of our time delay estimation algorithm – with the parameters specified in Section 4.2 – on the binaural speech corpus. For comparison



(a) Binaural speech signal.



(b) LCRs obtained from each onset model frequency in the filter-bank.

Figure 4.3: Binaural speech signal and resulting LCR pairs.

purposes, we also run the classical ITD estimation methods on the whole corpus and compute the simplified theoretical delay for each azimuth.

The first ITD estimation method we run is the threshold method. We start with the threshold method because it is the simplest procedure one can perform to estimate the ITD, so it can provide a loose lower bound to our algorithm's performance. As described in Chapter 1, we run the threshold procedure on the raw speech signal and we set the threshold level to 20 dB below the peak level of the speech signal. Note that this method cannot be executed in an online manner, as the threshold level depends on the maximum amplitude of the audio signal.

The second method that we run is the cross-correlation with frames. We set the frame size to 100 samples (so that it is comparable to the buffer needed for our algorithm) and we compute the cross-correlation of the raw speech signals on a frame-by-frame basis.

Then, we run both versions of our algorithm, first with the vanilla version and then with the confidence version of the delay estimation module. The parameters we use are the ones described in Section 4.2.

In Figure 4.4, it is possible to see the results of the threshold method for the binaural speech signal corpus. The error bars for this method are very large and, for some azimuths, the results are far from the simplified theoretical delay. Even though the threshold method works reasonably well with the HRIR, the raw audio signals are much more noisy and in a rough sense, less well behaved, than the impulse responses. Therefore, the simplicity of the procedure hinders its performance with speech signals that might be encountered in the real-world.

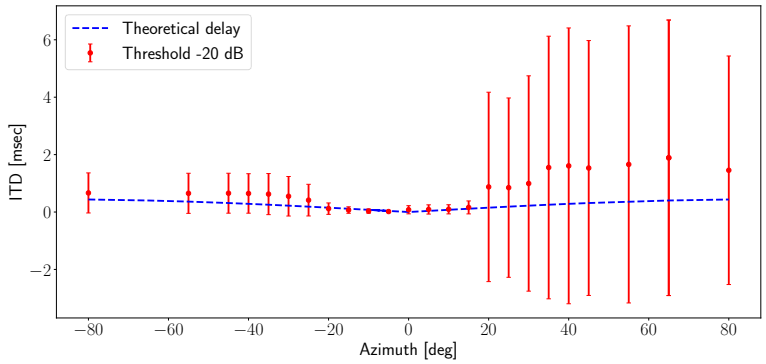


Figure 4.4: Time delay estimates for the threshold method.

Figure 4.5 shows the comparison of the results of both versions of our algorithm and for the cross-correlation with frames. First, it is worth noting that the confidence version performed significantly better than the vanilla version, both in terms of the mean value of the time delay estimate as well as on the length of the error bars. This results illustrates that we indeed addressed some of the major limitations of the vanilla version of the algorithm.

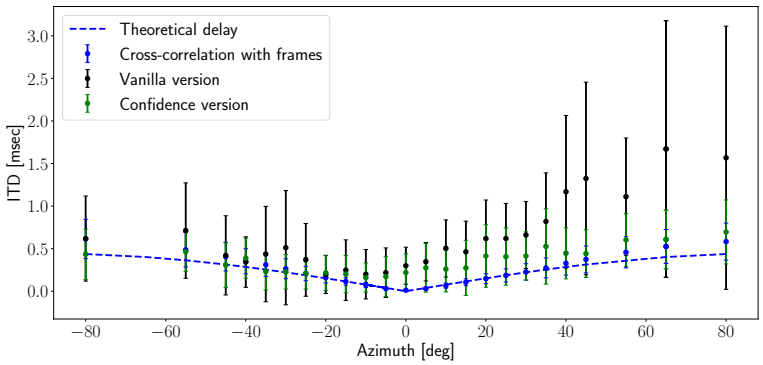


Figure 4.5: Comparison of time delay estimates between the cross-correlation with frames method and our versions of the proposed algorithm.

If we compare the performance of the cross-correlation with frames with the one for the confidence version of our algorithm, we can see that they are very comparable for the larger azimuths, but the cross-correlation with frames outperforms our algorithm on the azimuths closer to 0°.

The issue with the delay estimates produced for the azimuths close to 0° is that for most of the times, we have LCR peaks with different amplitudes in each channel. As a consequence, the polynomials that approximate the LCRs are similar, but slightly different. When we select two polynomials that are not exactly equal to estimate a very small delay from (which is the case for the azimuths close to 0°), we generally end up with a estimate that, although small, is not exactly equal to zero. In fact, this small artifact of estimating the delay from slightly different polynomials is present for all azimuths, but the delay estimates for the small azimuths are dominated by it, so it is more evident. There are ways to overcome this issue. The most straightforward one would be to increase the window length of the 3rd degree polynomial fit.

The binaural speech signals we work with were simulated from the convolution of single channel speech signals with the corresponding HRIRs. The HRIRs do not take into account the environment in which the subject is present, it just considers the effects of the torso and pinnae on the incoming sound. Therefore, there are no reflections or reverberation effects from the environment. We expect our algorithm to perform better than the cross-correlation with frames method the more these performance degrading effects are present in the environment. The cross-correlation with frames compares the two channels of the binaural signal on a frame-by-frame basis, but not all the frames should be compared if we think about the possibility of reverberation and reflection. By doing so, the method might be comparing two versions of the signal that do not match, for instance, in one channel we might have the version of the signal that travelled the direct path from the sound source to the microphone while on the other, we might have a reflected version. The timing information to perform sound source localization is lost in this case and this fact justifies the use of the precedence effect to perform sound source localization.

Our algorithm estimates the time delay between the signals only when the sound onsets are detected and, physically, this is the most appropriate way to extract the timing information from the signals. Even though the environment effects are not present in the binaural speech signals that we work with, we can look at the delay estimates produced by the confidence version of the algorithm and the estimates produced by the cross-correlation with frames method. The cross-correlation with frames method outputs one estimate per frame and, as can be observed in Figure 4.6, these estimates, despite clearly fluctuating around a mean value, oscillate a lot as a result of the effect previously described. On the other hand, our algorithm produces the time delay estimates only at the specific instants when a sound onset is detected. After a while, if another sound onset is detected, we might re-estimate the time delay, based on the confidence that we have in the first estimate, producing, thus, significantly more stable time delay estimates.

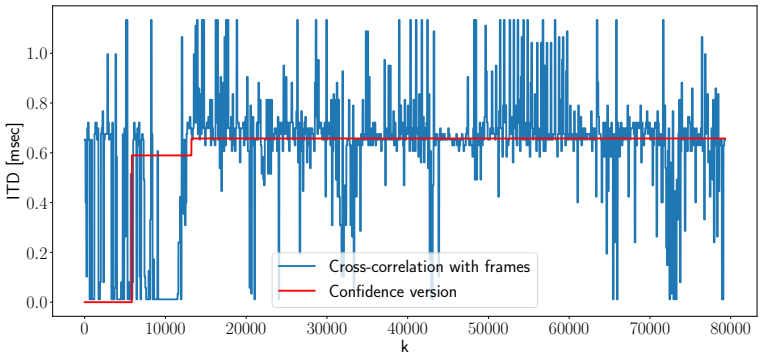


Figure 4.6: Comparison of delay estimates from the cross-correlation with frames method and the confidence version of our algorithm.

Chapter 5

Conclusion

In this work, we have extended the formulation started in [1] and explored an algorithm for ITD estimation based on fitting LSSMs. The algorithm performs time delay estimation from raw binaural audio signals in an online fashion and produces those estimates only when the sound onsets are detected, using the precedence effect as a physical justification. We tested the algorithm on a binaural speech signal corpus, constructed to contain a multitude of speech signals from different speaker at each azimuth. The versions of the presented algorithm are compared against the simplified expected theoretical delay as well as some of the classical methods used for ITD estimation.

The results suggest that the presented algorithm is producing reasonable estimates of the time delay for a wide range of speech signals. When looking at the results from individual signals, our algorithm produces much more accurate time delay estimates than the cross-correlation with frames method. We expect that the more reflection and reverberation is present in the environment, the better our performance will be if compared to the cross-correlation with frames method, due to the reliance on the precedence effect.

Future work should address the issues pointed out as limitations of the latest version of the algorithm, namely using a local measure of similarity instead of the Euclidean distance as well as the issue related to the difference in the amplitude of the LCRs from the two channels.

Bibliography

[1] G. C. Ornelas, "Sound source localization via onset detection," semester thesis, ETH Zurich, Zürich, Switzerland, 2019.

[2] L. Rayleigh, "Xii. on our perception of sound direction," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 13, no. 74, pp. 214–232, 1907.

[3] B. F. Katz and M. Noisternig, "A comparative study of interaural time delay estimation methods," *The Journal of the Acoustical Society of America*, vol. 135, no. 6, pp. 3530–3540, 2014.

[4] P. Minnaar, J. Plogsties, S. K. Olesen, F. Christensen, and H. Møller, "The interaural time difference in binaural synthesis," in *Audio Engineering Society Convention 108*. Audio Engineering Society, 2000.

[5] D. Preis, "Phase distortion and phase equalization in audio signal processing-a tutorial review," *J. Audio Eng. Soc*, vol. 30, no. 11, pp. 774–794, 1982. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=3810

[6] A. Kulkarni, S. Isabelle, and H. Colburn, "Sensitivity of human subjects to head-related transfer-function phase spectra," *The Journal of the Acoustical Society of America*, vol. 105, no. 5, pp. 2821–2840, 1999.

[7] A. Brown, G. C. Stecker, and D. Tollin, "The precedence effect in sound localization," *Journal of the Association for Research in Otolaryngology : JARO*, vol. 16, 12 2014.

[8] N. Zalmai, "A state space world for detecting and estimating events and learning sparse signal decompositions," Ph.D. dissertation, ETH Zurich, Zürich, Switzerland, 2017. [Online]. Available: http://d-nb.info/1138847720

[9] R. A. Wildhaber, "Localized state space and polynomial filters with applications in electrocardiography," Ph.D. dissertation, ETH Zurich, Zurich, 2019.

[10] F. W. R. A. Wildhaber, E. Ren and H.-A. Loeliger, "Signal analysis using local polynomial approximations," in *28th European Signal Processing Conference (EUSIPCO)*, 2020.

[11] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, "The cipic hrtf database," in *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*, 2001, pp. 99–102.

[12] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, "Timit acoustic-phonetic continuous speech corpus ldc93s1," 1993.

[13] M. Cooke, J. Barker, S. Cunningham, and X. Shao, "An audio-visual corpus for speech perception and automatic speech recognition," *The Journal of the Acoustical Society of America*, vol. 120, no. 5, pp. 2421–2424, 2006. [Online]. Available: https://doi.org/10.1121/1.2229005

[14] I. Holube, S. Fredelake, M. Vlaming, and B. Kollmeier, "Development and analysis of an international speech test signal (ists)," *International journal of audiology*, vol. 49, pp. 891–903, 12 2010.

[15] I. R. Titze, *Principles of Voice Production*. Prentice Hall, 1994.

[16] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

List of Figures

List of Tables