# Acoustic echo cancellation for human-robot communications

Jérôme Berclaz

# Acoustic echo cancellation for human-robot communications

Master thesis of **Jérôme Berclaz**

|  |  |
|---|---|
| Supervisor at NEC | **Akihiko Sugiyama** |
| Supervisor at EPFL | **Martin Vetterli** |
| Assistant at EPFL | **Harald Viste** |

March 7, 2004

NEC

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

*I would like to express many thanks to my supervisor at NEC Corporation, Dr. Akihiko Sugiyama, for guiding and inspiring me from the beginning through the end of this thesis with his intellectual advices and insightful suggestions. I truly appreciate and value his consistent feedback on my progress, which was always constructive and encouraging and ultimately drove me to the right direction.*

*I would also like to thank Harald Viste from EPFL, for remotely following the progress of my master thesis.*

*Finally, I wish to express my sincere gratitude to all the people, whose contribution made this thesis possible: my colleagues from the Robot Group at NEC Corporation and particularly Miki Sato and Thanh Phong Hua for their help and support, Dr. Yoshihiro Fujita from NEC Corporation for giving me the chance of doing my master thesis in his department and Professor Martin Vetterli from EPFL for organizing and supervising my thesis at EPFL.*

# Abstract

This master thesis presents a new efficient method of acoustic echo cancellation targeted at speech recognition for robots. The proposed algorithm features a new double-talk detector, an enhanced initialization and a new noise estimation method. The DTD algorithm is based on the normalized cross-correlation method, uses noise power estimation to be more robust in noisy environment and reacts more accurately to double-talk. The new initialization method switches between two different DTD algorithms to prevent problems during filter convergence. The simple, yet robust Geigel DTD is used during adaptive filter convergence, whereas the program switches to the newly developed DTD after convergence. Finally, the new noise estimation algorithm relies on the output auto-correlation to correctly estimate the noise.

To improve speech recognition performance, center clipping is applied on the output of the echo canceler, to further remove the residual echo. White noise is also added to the output signal, in order to make the signal power more stable, which helps the speech recognition engine.

Evaluation of the proposed algorithm has been done on a large set of sequences and results have shown that the new algorithm can increase the word recognition rate by up to 80%.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 The robot *PaPeRo*

*PaPeRo*[1], which stands for **Pa**rtner-type **Pe**rsonal **Ro**bot, is an autonomous robot prototype, targeted at home usage. The main goal of the *PaPeRo* project is to study human-machine interaction. Therefore, the robot's design has been focused on enhancing communication with humans rather than featuring complicated mechanics, like moving legs, and so on.

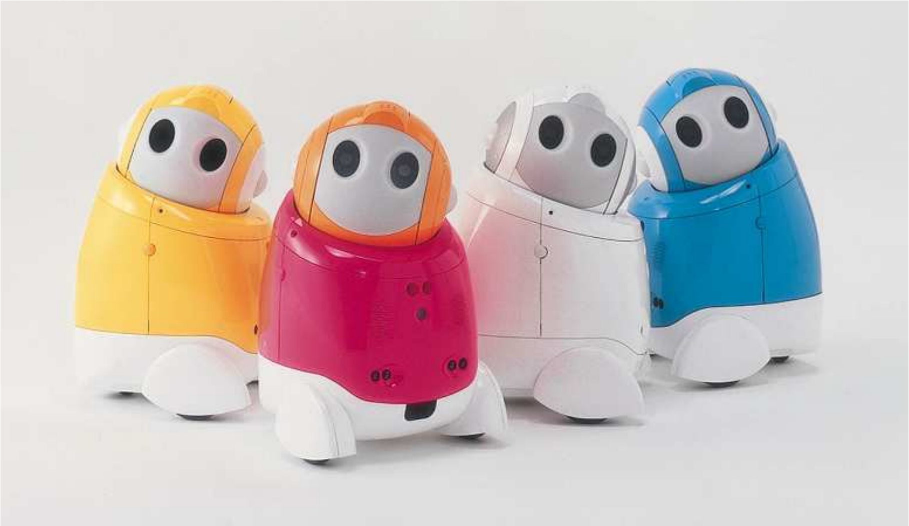


Figure 1.1: An external view of *PaPeRo*

Targeted essentially at home usage, the robot has a simple and smooth shape (c.f. figure 1.1), with a nice design and attractive colors. Its body contains the main part of the hardware and also includes three wheels, which allows it to move. On top of its body, its moving round head is the main component for interaction with people. *PaPeRo* is able to change its face appearance in order to express its “emotions” and “feelings”, or show that it is paying attention. The robot can also *see*, thanks to its two CCD cameras, placed in its eyes, *hear* with its microphone located on the forehead, *speak* using its two speakers located on its body and access the Internet using a IEEE 802.11 wireless access, among others.

From the hardware point of view, *PaPeRo* is, in fact, an enhanced PC, since its main hardware components are a Pentium 500MHz CPU, 192 MB of memory, a 6GB hard disk and a battery, all originating from laptop parts. The additional devices like cameras, or mechanisms for its head, are connected to the main-board using USB or IEEE1394. *PaPeRo* software runs on *Windows 98*, however a more recent version of the robot featuring *Windows 2000* is planned.

To integrate itself in a home environment, *PaPeRo* is able to perform the following functions, among others:

- estimate the direction of a sound source, using its three microphones around the neck

- avoid obstacles using its 2 CCD cameras and real-time visual recognition

- detect and recognize faces, as well as estimate the distance that separates it from a human

- recognize speech, even in a home and noisy environment

- communicate with other devices using, for example, a TV remote controller or an Internet access point, etc.

## 1.2 Speech recognition

One of *PaPeRo*’s most important components for enabling human interaction is speech recognition. Since the robot is targeted at use in the home environment, where the acoustic conditions are far from the ideal, it must feature a robust speech recognition engine that can achieve its task in difficult conditions.

Among the elements that might disturb speech recognition, the most prominent are the background noise and the echo. Since the robot’s microphones and speakers are located in a close range, the sound originating from the speaker is captured by the microphone as echo, as shown in Figure 1.2. This echo will reach the speech recognition engine, which will be disturbed in two ways. First, the engine might be lead to recognize words pronounced by the robot. Second, if the echo occurs when someone is speaking to *PaPeRo*, the echo will act as a strong noise that will seriously affect the speech recognition engine's performance. The main difference between the background noise and the echo of the robot's voice is that the latter is relatively predictable. It does indeed depend highly on how the voice is synthesized. Hence the idea of using an *acoustic echo cancellation* technique to remove this annoying echo and thereby improve the robustness of the speech recognition.

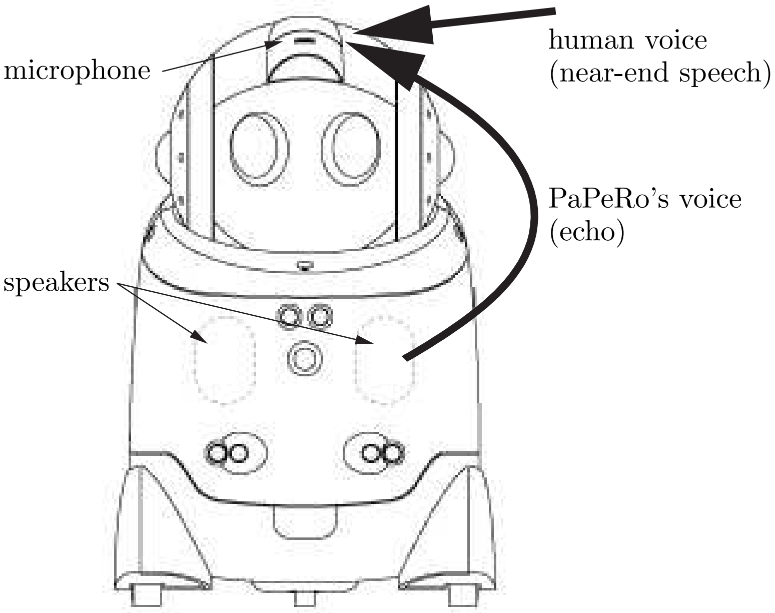


Figure 1.2: The echo problem for speech recognition

## 1.3 Echo cancellation

The goal of this master thesis is to develop a robust and reliable echo canceler to be used by the robot *PaPeRo*, in order to improve its voice recognition performance. It is based on the echo cancellation technique that is widely used in telephony or tele-conferencing, among others.

In this report, we describe the work done on the acoustic echo canceler and present the achieved outcomes. Chapter 2 introduces the theoretical aspects of echo cancellation. Then Chapter 3 describes the conventional echo cancellation algorithm, that was available at the beginning or this thesis, as well as its flaws. In Chapter 4, we present new improvements that we have brought to the original algorithm, in order to achieve better echo cancellation. Chapter 5 discusses some techniques that, combined with an echo canceler, help the speech recognition engine in its task. After this, Chapter 6 demonstrates the improvements brought by the new techniques developed during this master thesis. Finally, Chapter 7 concludes this report.

# Chapter 2

# Principle of acoustic echo cancellation

In this chapter we review the main principle and theoretical aspects behind acoustic echo cancellation. After a brief historical review, we introduce the NLMS algorithm, and the most recent *Affine Projection* algorithm, used to perform adaptive filtering. We then focus on the important function of *double-talk detection* and present some of the recent algorithms developed for this purpose, i.e. the *Geigel* DTD and the methods based on cross-correlation and normalized cross-correlation.

## 2.1 Prehistory - echo suppression

The need for echo suppression first appeared with long-distance telephony. Every telephone set in a given geographical area is connected to a central office by a two-wire line, which serves for communication in either direction. The use of the same two wires for both transmission and reception results in considerable saving of wires as well as of local switching equipment. However, for long-distance calls, a separate path is necessary for each direction of transmission. There are two reasons for this. First, long circuits require amplification, and amplifiers are one-way devices. Second, for reasons of economy, most long-distance calls are multiplexed, multiplexing requires that signals in the two directions be sent over different slots. Thus, on long-distance calls, it is necessary to connect the four-wire circuit to the two-wire circuit. A device that accomplishes this is called a *hybrid*. Figure 2.1 shows a typical long-distance telephone circuit. There are two hybrids involved in such a circuit, one at each end. In this configuration, an impedance mismatch appears most of the time at the hybrid level. Therefore, a portion of the signal gets reflected as an echo, as illustrated in Figure 2.1.

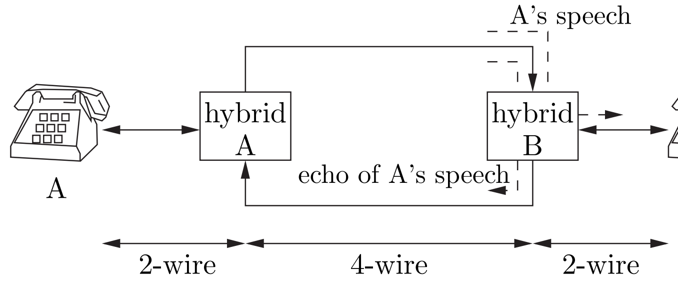


Figure 2.1: A typical long-distance telephone circuit

The echo generated by the hybrid is heard at the other end of the telephone circuit. This echo disturbs the conversation, because the users experience difficulties to talk while hearing a delayed echo of their own voice.

At the beginning, this problem has been addressed by inserting a loss in each direction of transmission. This attenuates the signal by $L$ dB, while at the same time attenuating the echo by $2L$ dB. However, for circuits exceeding about 3000 kilometers, this results in an unacceptably low signal level at the receiver. For those long telephone circuits, a device called *echo suppressor* was developed in the early 1920's. It consists basically of two voice-activated switches, placed at the 4-wire termination, at each end of the long-distance connection. Figure 2.2 shows a simplified view of an echo suppressor.

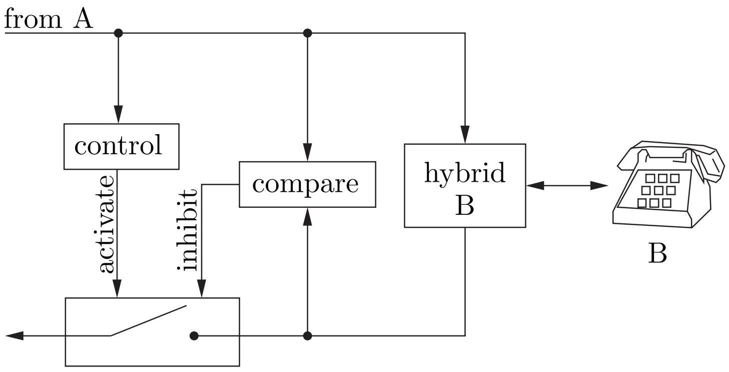


Figure 2.2: Schema of an echo suppressor

For the most part of the conversation, A and B speak separately, therefore A's echo can be prevented to go back to A by simply breaking the path after *hybrid B* when A is talking. To deal with *double-talk* situations (i.e.

when both talkers are speaking at the same time), the switch is prevented from opening whenever a comparator decides that the signal after *hybrid B* is mostly B's speech. However, the decisions that control the switches can not be made with perfect accuracy and even in the best echo suppressors, some amount of chopping of the initial portions of interrupter's speech is unavoidable. In addition, echo is not eliminated during double-talk sequences. Nevertheless, echo suppressors have been used for over 50 years on circuits with round-trip delay up to about 100 ms.

## 2.2 Adaptive echo cancellation

The year 1965 saw a new type of telephone circuit being invented with the advent of commercial communication satellites. Apart from making possible communications over a very long distance, this new system also introduced an additional delay of about 240ms. Combined with the delay of the terrestrial line, this could result in delays as long as about 600ms. In conversations happening with such long delays, the frequency of interruption becomes much higher. The old echo suppressor, with its poor performance during double-talk, was thus not adapted anymore for the new circuit. Therefore, since the early 1960's, more efforts have been made to find a new approach to echo removal. At that time, a major advance was made in echo suppression with the invention of the *adaptive echo canceler* [2]. The key idea of this technique is to artificially generate a copy of the echo, which will be subtracted from the real echo, in order to remove it. Figure 2.3 illustrates the principle of the echo canceler. Far-end speech $x(t)$ enters the hybrid, which produces an echo $y(t)$. This echo is coupled with the near-end signal in the return signal $m(t)$. The echo canceler is placed at the end of the four-wire line, just before the hybrid. It uses an adaptive filter $\hat{h}$ that tracks the real echo path $h$, using $x(t)$ as a reference signal, and produces an *echo replica* $\hat{y}(t)$. This echo replica is then subtracted from the signal $m(t)$, in order to cancel the echo. If the echo replica perfectly matches the real echo, the output signal $e(t)$ should then consist only of the near-end speech.

Since the echo path is unknown and can change during a conversation, the filter $\hat{h}$ has to be adaptive, so as to continuously track the echo path changes. This is usually done using an adaptive filter with the *Normalized least mean square* (NLMS) or *Affine projection algorithm* (APA), which will be discussed further in this chapter. The adaptive filter tries to build an echo replica by adjusting its coefficients, in such a way that the output signal $e(t)$ is driven to 0. However, this is only possible when the signal $m(t)$ consists only of echo. During *double-talk* periods (i.e. when far-end and near-end speeches

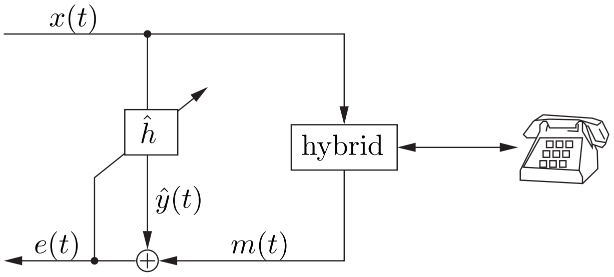


Figure 2.3: Adaptive echo canceler

occur at the same time), the near-end speech acts as a disturbing noise and will cause the adaptive filter to diverge. Therefore, the echo canceler must suspend filter adaptation when it senses double-talk. For that purpose, a function called *double-talk detector* (DTD) is added to the echo canceler. This will be discussed in more details in Section 2.5.

## 2.3 Acoustic echo cancellation

Historically, the first application of echo cancellation was the suppression of the echo created by the *hybrid* in telephone networks (called *network echo cancellation*). Nowadays, the increasing popularity of hands-free telephony and tele-conferencing (audio or video) have induced a new use of echo cancellation: *acoustic echo cancellation*. The latter application, indeed, incorporates a loudspeaker and a microphone, placed such that the microphone picks up the signal radiated by the loudspeaker and its reflections at the borders of the enclosure. As a result, the electroacoustic circuit may become unstable and produce howling. Moreover, users of the system are annoyed by listening to their own speech delayed by the round-trip time of the system, hence the need of attenuating the acoustic path between the loudspeaker and microphone. Figure 2.4 shows an acoustic echo canceler with a double-talk detector. The signal $n(t)$ represents the background noise and $v(t)$ the near-end speech, i.e. the voice of *PaPeRo*'s user.

Although the principle of acoustic echo cancellation stays basically the same as network echo cancellation, there are some significant differences that drive the problem even more complex. First of all, the echo path is much longer in the acoustic case. To have a good approximation of the acoustic echo path, a filter of about 1000 taps is needed, whereas a size of a few dozens taps is enough in the network case. Additionally, the acoustic echo path is much more subject to changes than the network one. Movement of objects,

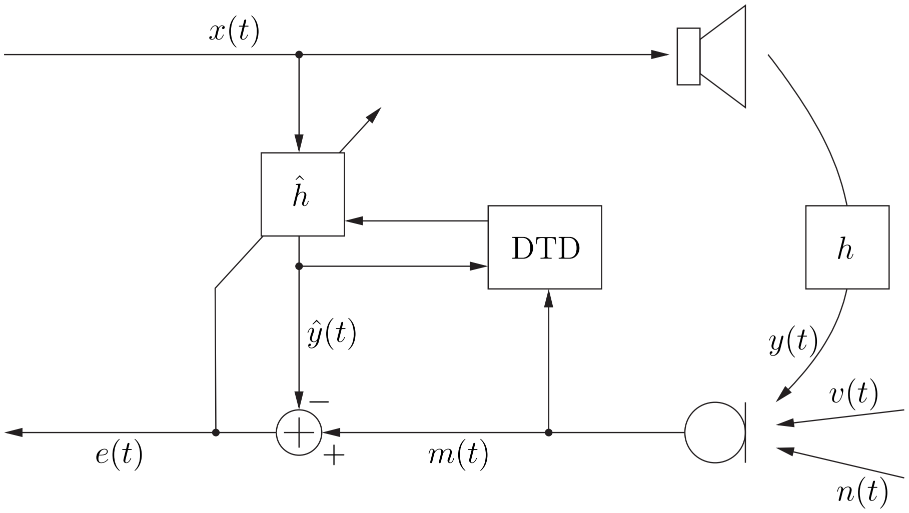


Figure 2.4: Acoustic echo canceler with double-talk detector

such as human bodies, doors, and the locations of the microphone and speaker can all modify the acoustic impulse response. Finally, the background noise in the near-end signal is usually stronger.

## 2.4 Coefficients adaptation algorithms

Adaptive filtering is the heart of echo cancellation. Several different algorithms are commonly used to carry out this task. Among them, the *Normalized least mean square* (NLMS) algorithm is the most widely employed, although the *Affine projection* (AP) and *Recursive least squares* algorithms are also used for their good convergence characteristics. In the rest of this section, we are going to introduce the basic principles behind the NLMS and AP algorithm.

### 2.4.1 NLMS

To introduce this algorithm, we will use figure 2.4 as a basis. Let us suppose that we use an N-tap FIR adaptive filter to model the real echo path. We define the filter coefficients as follows:

$$\hat{\mathbf{h}}^T = [\hat{h}_0, \hat{h}_1, \ldots, \hat{h}_{N-1}] \tag{2.1}$$

Then we also define the reference signal (i.e. the far-end speech) at time $i$ using a vector notation

$$\mathbf{x}_i^T = [x_i, x_{i-1}, \ldots, x_{i-N+1}] \tag{2.2}$$

Now, if we denote the echo path by an FIR filter with $h_i, 0 \leq i < \infty$, then we can define $\mathbf{h}$ as the $N$ first coefficients of the echo path.

$$\mathbf{h} = [h_0, h_1, \ldots, h_{N-1}] \tag{2.3}$$

The echo $y_i$ is the convolution of the reference signal $x$ with the echo path $h$ as

$$y_i = \sum_{m=0}^{\infty} h_m x_{i-m} \tag{2.4}$$

and in the same way, the echo replica is the convolution of the reference signal with the adaptive filter $\hat{h}$ as

$$\hat{y}_i = \sum_{m=0}^{N-1} \hat{h}_m x_{i-m} \tag{2.5}$$

Having defined all these, we can now express the error signal as

$$\begin{aligned} e_i &= y_i - \hat{y}_i + v_i + n_i = \sum_{m=0}^{\infty} h_m x_{i-m} - \sum_{m=0}^{N-1} \hat{h}_m x_{i-m} + v_i + n_i \\ &= (\mathbf{h} - \hat{\mathbf{h}})\mathbf{x} + z_i \end{aligned} \tag{2.6}$$

where

$$z_i = \sum_{m=N}^{\infty} h_m x_{i-m} + v_i + n_i \tag{2.7}$$

represents the unmodeled echo tail as well as the background noise $n_i$ and a possible near-end signal $v_i$. For the purpose of analyzing a given adaptation algorithm, it is often necessary to assume that the reference signal $x_i$ and echo $y_i$ are jointly wide-sense stationary. For this case, define

$$\mathbf{p} = E[\mathbf{x}_i y_i] \tag{2.8}$$

$$\mathbf{\Phi} = E[\mathbf{x}_i \mathbf{x}_i^T] = \begin{bmatrix} R_0 & R_1 & \ldots & R_{N-1} \\ R_1 & & \ddots & \vdots \\ \vdots & & & \\ R_{N-1} & & \ldots & R_0 \end{bmatrix} \tag{2.9}$$

where

$$R_j = E[x_i x_{i+j}] \tag{2.10}$$

are the autocorrelation coefficients of the reference signal.

Supposing that there is no other near-end signal than echo, the goal of the adaptive filter is to minimize the mean-square error $J = E[e_i^2]$. Since it is a quadratic function, we can find the minimum by canceling its derivative.

$$\begin{aligned}
\frac{\partial E[e_i^2]}{\partial \hat{h}_k} &= 2E\Big[e_i \frac{\partial e_i}{\partial \hat{h}_k}\Big] \\
&= 2E\Big[e_i \frac{\partial (y_i - \sum_{m=0}^{N-1} \hat{h}_m x_{i-m})}{\partial \hat{h}_k}\Big] \\
&= 2E[e_i \cdot (-x_{i-k})] \qquad (2.11)
\end{aligned}$$

We now need to set (2.11) to 0.

$$\begin{aligned}
\frac{\partial E[e_i^2]}{\partial \hat{h}_k} &= 2E[e_i \cdot (-x_{i-k})] \\
&= 2E[(y_i - \sum_{m=0}^{N-1} \hat{h}_m x_{i-m}) \cdot (-x_{i-k})] \\
&= 2E\Big[\sum_{m=0}^{N-1} \hat{h}_m x_{i-m} x_{i-k} - y_i x_{i-k}\Big] \\
&= 2\Big(\sum_{m=0}^{N-1} E[\hat{h}_m x_{i-m} x_{i-k}] - E[y_i x_{i-k}]\Big) \\
&= 2\Big(\sum_{m=0}^{N-1} \hat{h}_m R_{m-k} - p_k\Big) \qquad (2.12) \\
&= 0
\end{aligned}$$

Canceling this last equation leads to

$$\sum_{m=0}^{N-1} \hat{h}_m R_{m-k} = p_k \tag{2.13}$$

or

$$\hat{\mathbf{h}} = \mathbf{\Phi}^{-1} \mathbf{p} \tag{2.14}$$

Equation (2.14) defines the filter that best approximates the echo path, in order to remove the echo. However, the echo path is usually changing all the time, which implies that one needs to perform the costly matrix inversion at almost each iteration. This problem leads to the idea of adaptive filtering, which tracks the echo path by iteratively adapting the filter coefficients $\hat{h}_k$. Thus, the basic adaptive equation is

$$\hat{\mathbf{h}}_{n+1} = \hat{\mathbf{h}}_n + \alpha_n \mathbf{d}_n \tag{2.15}$$

where $\alpha_n$ is called the *step size* and $\mathbf{d}_n$ is the search direction.

Our goal is to minimize the mean square error $J = E[e_i^2]$. Since the gradient $\nabla J$ is in the direction of maximum rate of increase of $J$, the search direction should be $-\nabla J$. From (2.12), we know that $\frac{\partial J}{\partial \hat{h}_k} = 2(\sum_{m=0}^{N-1} \hat{h}_m R_{m-k} - p_k)$. Reordering terms and using matrix equation, we can write:

$$\nabla J = 2(\mathbf{\Phi}\hat{\mathbf{h}} - \mathbf{p}) \tag{2.16}$$

Thus, from equations (2.15) and (2.16), we can derive the following method

$$\hat{\mathbf{h}}_{n+1} = \hat{\mathbf{h}}_n - 2\alpha_n(\mathbf{\Phi}\hat{\mathbf{h}}_n - \mathbf{p}). \tag{2.17}$$

Yet we assumed the knowledge of $\mathbf{\Phi}$ and $\mathbf{p}$, however these statistics are not known and even changing. Thus, we should also update $\mathbf{\Phi}$ and $\mathbf{p}$ over time, which is still computationally expensive. Therefore, the idea of *least mean squares* (LMS) is to make a further approximation and replace the gradient of the mean square error by its instantaneous value:

$$\nabla J_i = \frac{\delta E[e_i^2]}{\delta \hat{\mathbf{h}}_i} \approx \frac{\delta e_i^2}{\delta \hat{\mathbf{h}}_i}. \tag{2.18}$$

Now, let us compute the following derivative

$$\begin{aligned} \frac{\delta e_i^2}{\delta \hat{\mathbf{h}}_i} &= 2e_i \frac{\delta(y_i - \hat{\mathbf{h}}_i^T \cdot \mathbf{x}_i)}{\delta \hat{\mathbf{h}}_i} \\ &= -2e_i \cdot \mathbf{x}_i. \end{aligned} \tag{2.19}$$

Thus, we obtain the LMS algorithm as

$$\hat{\mathbf{h}}_{n+1} = \hat{\mathbf{h}}_n + \alpha_n \cdot e_n \cdot \mathbf{x}_n \tag{2.20}$$

where the step size $\alpha_n$ has to be carefully chosen. When it is large, it increases the convergence speed, with possible instability. When set small, the convergence is slower, with less noisy $\hat{\mathbf{h}}_i$.

### From LMS to NLMS

A common modification to the LMS algorithm is to normalize the step-size to eliminate an undesirable dependence of convergence speed on the input signal power. Further, if the step-size $\alpha_n$ is kept constant and the signal power is increased, eventually, the algorithm becomes unstable. This is particularly a problem in speech applications, where the input signal power varies considerably. The step-size is therefore normalized by an estimate of the reference signal power

$$\alpha_n = \frac{a}{\sigma^2_{x_n} + b} \tag{2.21}$$

where $\alpha_n$ is the step-size at sample $n$, a and b are some well chosen constants, and $\sigma^2_{x_n}$ is an estimate of the reference signal power at time $n$. This modified version is commonly referred to as *normalized least mean square* (NLMS) algorithm.

## 2.4.2 Affine projection (AP) algorithm

Despite its low complexity, the NLMS algorithm suffers from slow convergence in case of a colored signal like speech. In order to improve convergence time, without adding too much complexity, the *affine projection* algorithm was developed [3]. The derivation of this algorithm is based on a geometrical interpretation of NLMS. Let us define the following notations, so as to introduce the principles of this algorithm:

1. For $\mathbf{a} = [a_1, \cdots, a_N]^T$ and $\mathbf{b} = [b_1, \cdots, b_N]^T$, $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{k=1}^{N} a_k b_k$
2. $\|\mathbf{a}\| = \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle}$
3. $\Pi_j = \{\mathbf{h} | \mathbf{h} \in \mathbb{R}^N, \langle \mathbf{h}, \mathbf{x}_j \rangle = y_j\}$

Definition (1.) is the scalar product of two vectors in $\mathbb{R}^N$ and $\Pi_j$, defined in (3.), is the set of all coefficient vectors which give the output equal to $y_j$ for the input vector $x_j$. It forms a hyperplane in the $N$-dimensional Euclidean space.

Using a geometrical interpretation, we can describe the NLMS algorithm as follows:

```
$\hat{\mathbf{h}}_0$ =arbitrary value
loop
  $\hat{y}_j = \langle \hat{\mathbf{h}}_j, \mathbf{x}_j \rangle$
  $e_j = y_j - \hat{y}_j$
  $\Delta\hat{\mathbf{h}}_j = \frac{e_j}{\|\mathbf{x}_j\|^2}\mathbf{x}_j$
```

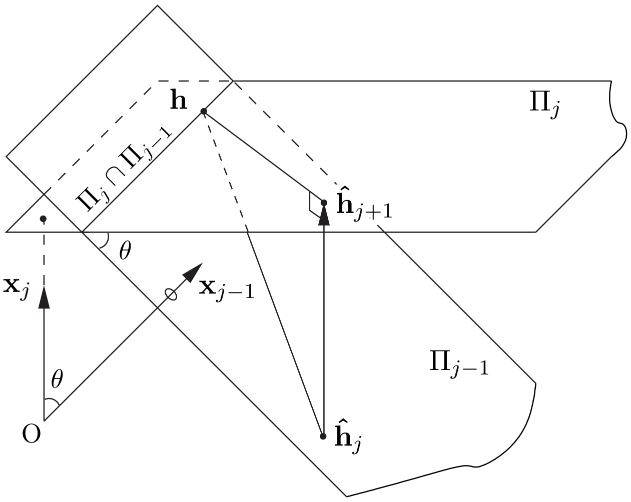


Figure 2.5: Geometrical interpretation of NLMS

```
    $\hat{\mathbf{h}}_{j+1} = \hat{\mathbf{h}}_j + \alpha\Delta\hat{\mathbf{h}}_j$
end loop
```

where $\alpha$ is called a step size and should be in the interval $]0;2[$. As we can see in Figure 2.5, the convergence speed of the coefficient vector highly depends on the angle between $\Pi_{j-1}$ and $\Pi_j$. If this angle is close to 0 or $\pi$, then the convergence speed is decreased. The angle $\theta$ between $x_{j-1}$ and $x_j$ is given by

$$\cos\theta = \frac{\langle \mathrm{x}_{j-1}, \mathrm{x}_j \rangle}{\|\mathrm{x}_{j-1}\| \cdot \|\mathrm{x}_j\|} \tag{2.22}$$

One could also notice that this is precisely the equation of the input signal auto-correlation. Thus, when $\theta$ is close to 0, $\cos\theta$ approaches 1. Therefore, for correlated signals like speech, NLMS will converge quite slowly. This is an undesirable property, since acoustic echo cancellation is used to suppress speech signals most of the time.

To address this problem of convergence, *affine projection* algorithm has been developed. The idea behind this new algorithm was to make the convergence speed independent of the angle $\theta$ between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$. This can be done by slightly modifying the NLMS algorithm. In Figure 2.5, the vertical line that goes from $\hat{\mathbf{h}}_{j-1}$ to $\hat{\mathbf{h}}_j$ should go to $\Pi_{j-1} \cap \Pi_j$ instead, to keep the convergence speed constant. Then, according to Figure 2.6, the following algorithm can be derived:

```
$\hat{\mathbf{h}}$ =arbitrary value
loop
```

```
    $\tilde{\mathbf{x}}_{j-1} = \frac{\langle \mathbf{x}_{j-1},\mathbf{x}_j\rangle}{\|\mathbf{x}_{j-1}\|^2}\mathbf{x}_{j-1}$
    $\mathbf{u}_j = \mathbf{x}_j - \tilde{\mathbf{x}}_{\mathbf{j-1}}$
    $\hat{y}_j = \langle\, \hat{\mathbf{h}}_j,\, \mathbf{x}_j\rangle$
    $e_j = y_j - \hat{y}_j$
    $\mathbf{\Delta}\hat{\mathbf{h}}_j = \frac{e_j}{\langle \mathbf{u}_j,\mathbf{x}_j\rangle}\mathbf{u}_j$
    $\hat{\mathbf{h}}_{j+1} = \hat{\mathbf{h}}_j + \mathbf{\Delta}\hat{\mathbf{h}}_j$
end loop
```

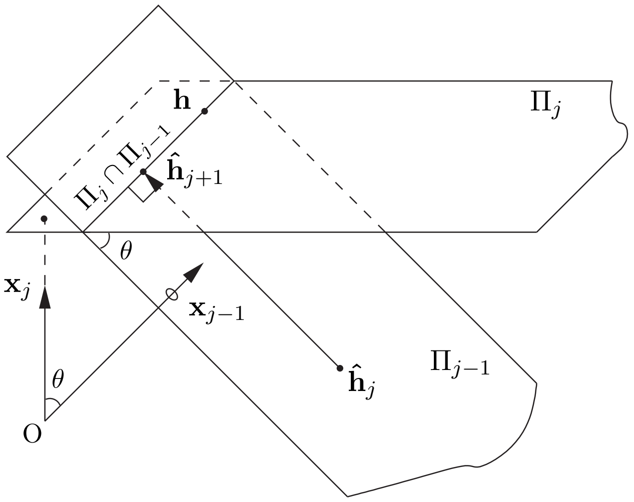


Figure 2.6: Geometrical interpretation of AP

Looking at the NLMS and the new algorithm, we can notice that $\hat{\mathbf{h}}_{j+1}$ is the orthogonal projection of $\hat{\mathbf{h}}_j$ on $\Pi_j$ in NLMS, and the orthogonal projection on $\Pi_j \cap \Pi_{j-1}$ in the new algorithm. From that point of view, the new algorithm can be extended the following way. Let us call $P_\Pi(\hat{\mathbf{h}})$ the orthogonal projection of $\hat{\mathbf{h}}$ on $\Pi$. Filter adaptation can be modified this way

$$\hat{\mathbf{h}}_{j+1} = P_{\Pi_j \cap \Pi_{j-1} \cap \cdots \cap \Pi_{j-p+1}}(\hat{\mathbf{h}}_j) \tag{2.23}$$

We then can see that the vector $\hat{\mathbf{h}}_{j+1}$ is the solution to a set of equations with $\hat{\mathbf{h}}$ as the unknown, which minimizes $\|\hat{\mathbf{h}} - \hat{\mathbf{h}}_j\|$.

$$\left\{ \begin{array}{lcl} \langle \mathbf{x}_j, \hat{\mathbf{h}} \rangle & = & y_j \\ \langle \mathbf{x}_{j-1}, \hat{\mathbf{h}} \rangle & = & y_{j-1} \\ & \vdots & \\ \langle \mathbf{x}_{j-p+1}, \hat{\mathbf{h}} \rangle & = & y_{j-p+1} \end{array} \right. \tag{2.24}$$

Defining

$$\begin{aligned}\mathbf{X}_j &= [\mathbf{x}_j, \mathbf{x}_{j-1}, \cdots, \mathbf{x}_{j-p+1}]^T \\ \mathbf{y}_j &= [y_j, y_{j-1}, \cdots, y_{j-p+1}]^T\end{aligned}$$

we can rewrite the set of equations using vector notation

$$\mathbf{X}_j \cdot \hat{\mathbf{h}} = \mathbf{y}_j \tag{2.25}$$

Letting $\mathbf{X}_j^+$ be the Moore-Penrose generalized inverse of $\mathbf{X}_j$, the system can be solved as

$$\hat{\mathbf{h}}_{j+1} = \hat{\mathbf{h}}_j + \mathbf{X}_j^+(\mathbf{x}_j - \mathbf{X}_j\hat{\mathbf{h}}_j) \tag{2.26}$$

Based on the above equation and introducing the step size $\alpha$, the following adaptive algorithm is developed.

```
$\hat{\mathbf{h}}_0$=initial value
loop
  $\Delta\hat{\mathbf{h}}_j = \mathbf{X}_j^+(\mathbf{x}_j - \mathbf{X}_j\hat{\mathbf{h}}_j)$
  $\hat{\mathbf{h}}_{j+1} = \hat{\mathbf{h}}_j + \alpha\Delta\hat{\mathbf{h}}_j$
end loop
```

This algorithm is called the *Affine projection algorithm* (APA) and $p$ is its order. According to this definition, NLMS is the first-order APA.

## 2.5 Double-talk detection

As already stated in Section 2.2, tracking of the echo path is possible only when the near-end signal consists solely of echo. During double-talk periods, the near-end speech $v(n)$ is also present in the microphone input signal and acts as an interference to the adaptive filter. If we go on adapting coefficients during double-talk, the adaptive filter will be disturbed and may diverge from its converged state. Therefore, a double-talk detector is used in adaptive echo cancelers to detect double-talk periods, and adaptive filter coefficient adjustment is stopped during these periods to prevent the echo canceler from being disturbed by the near-end speech signal.

Double-talk detection plays an important part in adaptive echo cancellation. The basic requirement for a double-talk detector is that it can detect double-talk quickly and accurately. Besides, it should also have the ability to distinguish double-talk from echo path variation and quickly track variations in the echo path.

### 2.5.1 Geigel DTD

*Geigel DTD* was the first method proposed for double-talk detection [4]. Its principle is based on a simple power comparison. Since the echo is supposed to be an attenuated version of the reference signal, if the near-end signal becomes higher than a certain level, we can assume that we are dealing with a double-talk situation. Therefore, the Geigel DTD algorithm is defined as follows: near-end speech is declared, and thus coefficients adaptation is stopped, when

$$m(k) > \frac{1}{2} max\{x(k-1), x(k-2), \ldots, x(k-N)\} \tag{2.27}$$

where $m(k)$ is the microphone input signal and $x(k)$ represents the reference signal. This method was specifically designed for echo cancellation in the telephone network and thus the factor of 1/2 is based on the assumption that there is at least a 6 dB loss through the hybrid. A so-called hangover time, $T_{hold}$ is also specified such that if double-talk is detected, then the adaptation is inhibited for this duration beyond the detected end of double-talk.

One of the main advantages of this method is its simplicity; it does not require a high computation power. However, although this detector works fairly well, detection errors do occur, and these result in frequent divergence of the adapted filter coefficients, which in turn gives rise to a large amount of uncanceled echo.

Moreover, it is not well adapted to *acoustic* echo cancellation, where the attenuation of the echo path may not be predicted as in the network case, and can be very different depending on the situation.

### 2.5.2 DTD based on cross-correlation

A more advanced technique for double-talk detection has been proposed by Ye and Wu [5]. This method uses the cross-correlation of the reference signal $x(n)$ and the error $e(n)$ to detect near-end speech. If we assume that the echo canceler has converged at time $n$, the echo will be perfectly canceled and it is then obvious that

$$E[e(n)\mathbf{x}(n)] = 0. \tag{2.28}$$

If we further assume that near-end speech $v(n)$ is uncorrelated with the reference signal, we can see that this previously computed expectation is not disturbed by double-talk.

$$E[e(n)\mathbf{x}(n)] = E[v(n)\mathbf{x}(n)] = 0 \tag{2.29}$$

However, in case of variations in the echo path, the echo will not be completely removed and $E[e(n)\mathbf{x}(n)]$ will differ from 0.

Based on the above considerations, this technique proposes to continuously compute $E[e(n)\mathbf{x}(n)]$ and to update the adaptive filter coefficients only when it is significantly different from 0[1]. If this solution has the advantage of accurately distinguishing double-talk from echo-path change, it is however not suited for acoustic echo cancellation purpose, since in the latter case, the echo-path is constantly changing.

A similar idea has been proposed in [6]. The cross-correlation of $x$ and $m$ is used to discriminate double-talk. The cross-correlation coefficients are defined as follows:

$$\gamma_{mx}(n) = \frac{|E[x(n)m(n)]|^2}{E[x(n)^2]E[m(n)^2]} \tag{2.30}$$

Based on this definition, a decision variable is then formed as

$$\xi^{(1)} \equiv \frac{1}{I+1}\sum_{i=0}^{I} \gamma_{mx}(i) \tag{2.31}$$

The decision rule for detecting double-talk is:

- if $\xi^{(1)} > T \Rightarrow$ single-talk
- if $\xi^{(1)} < T \Rightarrow$ double-talk

where $T$ is a threshold level.

This method yields better results than Geigel DTD. It is also better than the cross-correlation between $x$ and $e$, because it actually detects double-talk, as the former rather detects echo-path changes.

### 2.5.3 DTD based on normalized cross-correlation

Although the method based on cross-correlation of $x$ and $m$ is quite good, it suffers from a major drawback: the value of the threshold $T$ , used for decision, is hard to define and will strongly vary among experiences. In order to address this issue, a new DTD technique derived from the cross-correlation method was introduced [7]. The goal of this new method is to derive a new normalized cross-correlation vector between $\mathbf{x}$ and $m$, so as to use a decision variable with a fixed threshold.

Let us suppose that the echo-path is finite, therefore $y(n) = \mathbf{h}^T\mathbf{x}(n)$. Supposing that there is no other near-end signal than echo, we can derive

$$\sigma_m^2 = \mathbf{h}^T\mathbf{R}_{xx}\mathbf{h} \tag{2.32}$$

[1] It means that the echo-path has changed and filter adaptation is needed.

where $\sigma_m^2 = E[m(n)^2]$ and $\mathbf{R}_{xx} = E[\mathbf{x}(n)\mathbf{x}^T(n)]$. Defining

$$\mathbf{r}_{xm} = E[\mathbf{x}(n)m(n)] = \mathbf{R}_{xx}\mathbf{h} \tag{2.33}$$

we can rewrite equation (2.32) as

$$\sigma_m^2 = \mathbf{r}_{xm}^T \mathbf{R}_{xx}^{-1} \mathbf{r}_{xm}. \tag{2.34}$$

However, in the general case, the microphone signal can include some near-end speech ($v(n) \neq 0$) and we can thus rewrite the previous equation as

$$\sigma_m^2 = \mathbf{r}_{xm}^T \mathbf{R}_{xx}^{-1} \mathbf{r}_{xm} + \sigma_v^2. \tag{2.35}$$

The new decision variable is then obtained by dividing equation (2.34) by $\sigma_m^2$ and taking the square root:

$$\xi^{(2)} \equiv \sqrt{\mathbf{r}_{xm}^T (\sigma_m^2 \mathbf{R}_{xx})^{-1} \mathbf{r}_{xm}}. \tag{2.36}$$

Since the matrix inversion $R_{xx}^{-1}$ is a very CPU-consuming operation, we can make the following approximation in order to speed up computations:

$$\mathbf{R}_{xx}^{-1} \mathbf{r}_{xm} = \mathbf{h} \cong \hat{\mathbf{h}} \tag{2.37}$$

where $\mathbf{h}$ is the real echo path, and $\hat{\mathbf{h}}$ is the approximation of the echo path by the adaptive filter. Note that this approximation only holds if we suppose that the adaptive filter has converged, hence $\hat{\mathbf{h}} \cong \mathbf{h}$. Replacing (2.37) in (2.36), we obtain

$$\xi^{(2)} = \sqrt{\mathbf{r}_{xm}^T \sigma_m^{-2} \hat{\mathbf{h}}}. \tag{2.38}$$

Furthermore, knowing that $\mathbf{r}_{xm}^T = \mathbf{h}^T \cdot E[\mathbf{x}(n)\mathbf{x}^T(n)]$ and that $\hat{\mathbf{h}} \cong \mathbf{h}$, we can expand (2.38)

$$\begin{aligned} \xi^{(2)} &= \sqrt{\frac{\hat{\mathbf{h}}^T \cdot E[\mathbf{x}(n)\mathbf{x}^T(n)] \cdot \hat{\mathbf{h}}}{\sigma_m^2}} \\ &= \sqrt{\frac{E[\hat{y}(n)\mathbf{x}^T(n)] \cdot \hat{\mathbf{h}}}{\sigma_m^2}} \\ &= \sqrt{\frac{E[\hat{y}^2(n)]}{\sigma_m^2}} = \sqrt{\frac{\sigma_{\hat{y}}^2}{\sigma_m^2}}. \end{aligned} \tag{2.39}$$

From Figure 2.4, we know that the microphone signal is composed from echo, near-end speech and background noise. Therefore, we can finally write the DTD decision variable in this form

$$\xi^{(2)} = \sqrt{\frac{\sigma_{\hat{y}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_y^2}}. \tag{2.40}$$

If we consider the noise term as being negligible, it is easy to see that in absence of near-end speech ($\sigma_v^2 = 0$), $\xi^{(2)}$ will be close to 1. When near-end speech is present, $\xi^{(2)}$ is smaller than 1. Therefore, in order to discriminate double-talk, we just need to set a threshold $T$ and compare the decision variable $\xi^{(2)}$ with $T$. When $\xi^{(2)} < T$, we can assume that double-talk is taking place. Moreover, due to the normalized form of the decision variable $\xi^{(2)}$, the threshold $T$ is now independent of the data.

# Chapter 3

# The conventional algorithm

In this chapter, we describe the conventional acoustic echo cancellation algorithm that we started with, at the beginning of the project. We explain how it has been analyzed and what problems have been identified.

## 3.1 Program structure

At the beginning of our project, we had an echo cancellation program previously written by a former internship student at **NEC**. This program was written in C and was performing echo cancellation using a fast version [8] of the Affine Projection algorithm. In order to avoid the disturbing influence of background noise on coefficient adaptation, the algorithm also featured an adaptive step-size based on noise estimation [9] [10]. To control the adaptive step-size, the algorithm included a simple background noise estimation process. It also implemented a double-talk detector using the normalized cross-correlation method [7].

The program was designed as a standalone command line executable, that took two audio sequences in text format (one representing the reference signal, i.e. the robot's voice and the other the signal from the microphone). When invoked with correct arguments, the program performed echo cancellation on the microphone signal using the reference signal and output the result in a new file.

## 3.2 Program analysis

Although the program seemed to perform echo cancellation correctly, the performance was still far from expectation. Therefore, our first task in this project was to evaluate the initial program, so as to determine what was wrong and what could be added or modified in order to improve its performance.

In order to ease the analysis of the program, some scripts and makefiles were first developed to display the program's internal variables. All the plots of variables were made with *Gnuplot*[1], which was chosen for its quality and its capacity to be invoked by a script. A few audio sequences, originating from different environments, were provided to test the program.

## 3.3 Identified problems

Apart from some minor bugs, our analysis revealed a series of problems that were compromising the echo cancellation process. We describe them in details in the following sections.

### 3.3.1 Inappropriate initialization

One of the main problems of the initial program resulted from the fact that very little care had been given to initialization. When started, the program was forcing the double-talk detector output to 1 (no double-talk) during a few samples, hence forcing filter coefficients adaptation. Then, the program performed normal echo cancellation with the help of the normalized cross-correlation based DTD.

Two fundamental problems can occur due to this inadequate initialization phase. First of all, and most obviously, it is possible that a double-talk sequence takes place when the program is started. Since DTD is forced to 1 at the beginning, regardless of what happens in the input signals, filter coefficients would be adapted during double-talk, which would result in divergence, precisely at a time where fast convergence would be greatly needed.

Second, the use of the normalized cross-correlation based DTD is inadequate when the adaptive filter has not converged. One needs to recall that, in order to avoid computing the inverse of the matrix $\mathbf{R}_{xx}$, the following approximation has been used in the derivation of the DTD algorithm (see Section 2.5.3)

$$\mathbf{R}_{xx}^{-1}\mathbf{r}_{xm} = \mathbf{h} \cong \hat{\mathbf{h}} \tag{3.1}$$

This approximation implicitly assumes that the adaptive filter has converged and is valid at this condition only. It is obviously not the case during the initial convergence of the adaptive filter, and therefore, the output of the DTD will not be reliable at that time. As we will demonstrate later, when the

[1] `http://www.gnuplot.info`

approximation (3.1) is not valid, the value of the DTD decision variable $\xi^{(2)}$ will be lowered, resulting in false alarms. Although a few false alarms is not very disturbing during the echo cancellation process, it becomes more critical if it happens during the initial convergence phase. At the beginning, we need, indeed, to adapt the coefficients as fast as possible. However a wrong DTD decision would prevent coefficient adaptation and rather delay the filter convergence. In the worst case, the filter misadaptation could prevent the DTD from working correctly, which would in turn prevent coefficient adaptation. Thus, if no care is given to initialization, the program could enter a fatal loop that would ruin the complete echo cancellation process.

### 3.3.2 Inaccurate background noise estimation

In order not to be disturbed by background noise, the algorithm implements a variable step-size controlled by the noise level [9]. However, there is no easy way to accurately obtain the background noise level. The only data that the program can use are the reference ($x$) and the microphone ($m$) signals. Although the background noise is included in the microphone signal, it is most of the time mixed with echo or near-end speech. Therefore, the only way to estimate the noise level is to identify periods of the microphone signal that are free from echo and near-end speech and update the average noise-power during this time. In the worst case, if echo or near-end speech is present at all time, there is no way of estimating noise. Using the output signal instead of the microphone signal for noise estimation is possible. The advantage of doing so is that, since echo is supposed to be canceled, there should be more noise-only sequences. However, even when echo is canceled, there might still be some residual echo, due to the unmodeled tail of the echo path. Moreover, especially in *acoustic* echo cancellation, the echo path is changing all the time, which will increase the amount of residual echo present in the output signal.

The strategy used in the initial algorithm was to perform background noise estimation using the output signal. Estimation was carried out only when the reference signal power was under a defined threshold, hence trying to avoid the effect of residual echo on noise estimation. However, although this method is simple in implementation, it is obviously not good enough to ensure accurate noise estimation. No attention is given to near-end speech, and it is very likely that noise estimation occurs during a near-end speech period, whose power is normally much higher than that of the background noise. In such a case, noise power will be greatly over-estimated, resulting in considerably slow coefficient adaptation.

Moreover, in the next chapters, we will see that the enhancements that will be brought to the conventional algorithm make extensive use of noise estimation and especially require it to be precise. Therefore, it becomes obvious that the conventional noise estimation can not be used in its present form and must be improved.

### 3.3.3 The double-talk detector

Although the double talk detector was build upon recent and proved techniques, analysis showed that it did not perform as good as we expect. In the ideal case, the decision variable $\xi^{(2)}$ should be bigger than the threshold $T$ when no near-end speech is present, and smaller than $T$ in case of near-end speech. However, we were able to notice that this ideal behavior was biased in a few situations. In case of silence (i.e. no near-end speech neither echo) a false alarm was systematically happening, thus preventing coefficient adaptation.

It was also noticed that the quality of DTD was highly correlated with the echo power, or rather the power ratio of near-end speech to echo. During high power echo periods, double-talk detection quality is degraded and some near-end speech periods are chopped and even missed. This resulted in complete words being highly distorted and not recognizable by the speech recognition engine. Obviously this was not acceptable and needed to be fixed.

# Chapter 4

# New algorithms

In this chapter, we describe the new algorithms that have been developed in order to correct the flaws from the conventional echo canceler and improve its performance. These include an enhanced algorithm initialization, that allows the echo canceler to react properly to any input signal, a new DTD algorithm that is robust to the background noise and reacts more accurately to near-end speech, and a new noise estimation method, more precise and reliable. We focus on the theoretical aspects and will present the results in a later chapter.

## 4.1 Better initialization

In the previous chapter, we have seen that the simple initialization done in the original program was not good enough to ensure the correct functioning of the echo cancellation process. Although the algorithm is working on a few signals, we have also found some other signals on which the echo cancellation was completely failing, due to inadequate initialization. In this section, we investigate a better way to perform the algorithm initialization, such that echo cancellation is guaranteed to work with any input.

The double-talk detector based on normalized cross-correlation, as implemented, does not work well when the adaptive filter has not converged, due to the approximation (2.37) used to avoid a matrix inversion. Therefore, it is not advised to use this DTD algorithm during the initial convergence phase. One might think that a solution to this problem would be to suppress the approximation and to compute the original equation (2.36) instead. However, this would imply continuously computing a matrix inversion, which is not feasible with *PaPeRo*'s hardware, especially since acoustic echo cancellation requires a long filter with over 1000 taps.

Yet, one could not do without a double-talk detector during the convergence phase. Therefore, a solution would be to use another method for double-talk detection that does not depend on the convergence state of the adaptive filter. Geigel DTD typically fulfills these conditions. Thus using Geigel DTD instead of the normalized cross-correlation method would certainly reduce problems during the adaptive filter convergence, however its lower performance compared to the latter technique would alter the overall echo cancellation. Hence, a good trade-off can be to use Geigel DTD during the initialization phase, and then switch to the more powerful normalized cross-correlation DTD, once the adaptive filter has reached convergence. This way, we would avoid bad behavior of the DTD due to immature filter coefficients.

### 4.1.1 Geigel DTD for acoustic echo cancellation

The main advantage of Geigel DTD is its very easy implementation. We will remind here its original definition as in [4]

$$m(k) > \frac{1}{2} max\{x(k-1), x(k-2), \ldots, x(k-N)\} \tag{4.1}$$

This detector only relies on power comparison, and the powers that it needs are already computed for other parts of the algorithm. However, one problem of using Geigel DTD in *acoustic* echo cancellation as opposed to *network* echo cancellation, is that we can not predict the power loss from the reference signal to the echo. This can vary from one sequence to another, and even during a single sequence, if the room conditions are changing. Therefore, if we want to use this technique for acoustic echo cancellation, we need to be able to evaluate the power ratio of reference over echo signal. This can be recursively computed using a forgetting factor $\delta$, such that it is averaged over the last few samples, without using a large amount of memory

$$r_{j+1} = \delta r_j + (1-\delta)\frac{\sigma_x^2}{\sigma_y^2} \tag{4.2}$$

where $\sigma_y^2$ and $\sigma_x^2$ are the power of the echo and the reference signal respectively, and $\delta$ is the forgetting factor, usually defined in $[0.9, 1[$ depending on how smooth the averaging should be. Note that we use the $l_2$ norm instead of the $l_\infty$ norm found in the original Geigel definition. Now that we have an estimation of the power ratio of reference over echo signal, we can define a customized version of the Geigel DTD, adapted for acoustic echo cancellation. The decision variable for Geigel DTD would then be

$$\xi^{(G)} = \sigma_x^2 - \beta\sigma_m^2 \cdot r_j \tag{4.3}$$

where $r_j$ is the power ratio of reference over echo signal at time $j$, defined in (4.2) and $\sigma_m^2$ is the power of the microphone signal. The double-talk decision is then very easy: when $\xi^{(G)}$ is positive, we consider that there is no near-end speech and declare double-talk only when $\xi^{(G)}$ gets negative, which means that the microphone signal is stronger than the expected echo and that it probably contains near-end speech in addition to echo. The factor $\beta$ used in (4.3) is there to avoid oscillating between the two states, since the ratio $r_j$ is only an approximation. Thus small fluctuations of echo intensity will not trigger uselessly the double-talk detection.

The advantage of this simple double-talk detection is that it only depends on the input signal itself and does not rely on other mechanisms inside the algorithm. Therefore, it is independent and its performance will not be affected if other parts of the algorithm do not behave correctly.

### 4.1.2 Detecting adaptive filter convergence

Now that we have derived a simple double-talk detection that works even when the adaptive filter has not fully converged, we can integrate it into the echo cancellation algorithm, along with the normalized cross-correlation DTD. However, we still need to determine the right moment at which we should switch between the two DTD algorithms. The normalized cross-correlation DTD is reliable once the adaptive filter has converged to a reasonable approximation of the echo path. Therefore, we need a way to determine when the initial convergence phase is over.

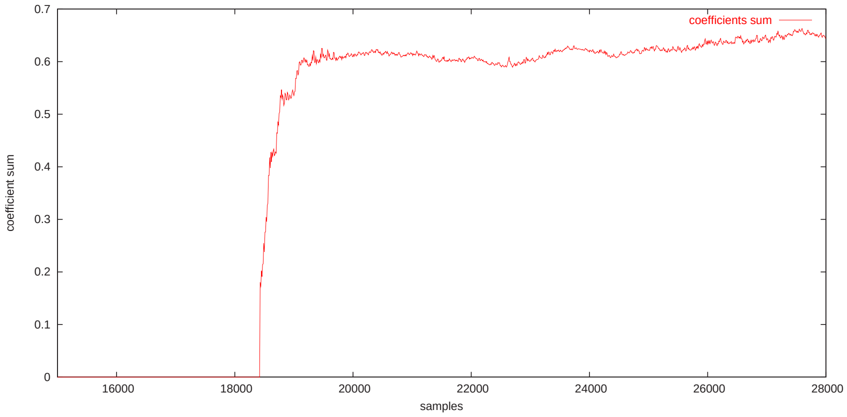


Figure 4.1: Evolution of the sum of filter coefficients. We can see that convergence is happening around sample 20000

To achieve this purpose, we have based our reasoning on the following idea. Before starting the echo cancellation algorithm, the adaptive filter coefficients are all initialized to 0. Then, they are iteratively updated until they reach a value close to the real echo path. Once the adaptive filter has finished its initial convergence, the coefficients update will be limited to the tracking of the echo path. Therefore, unless a sudden change occurs in the echo path, coefficient adaptation should be relatively small. Based on this discussion, we can design a method for detecting when the adaptive filter approximately converges. Let us define $\Sigma_n$ as the sum of the filter coefficients at time $n$.

$$\Sigma_n = \sum_{i=0}^{N} |\hat{h}_i(n)| \tag{4.4}$$

Once convergence is reached, the coefficients' sum $\Sigma$ should vary little, as shown in Figure 4.1. Thus, its gradient should approach 0. Hence the idea of comparing the gradient $\Delta\Sigma$ to a threshold: if $\Delta\Sigma$ stays below the threshold during a predefined number of samples $N_s$, then we can assume to have reached a stable situation, and declare convergence. To compute the gradient of the coefficient's sum $\Sigma$, it is enough to calculate the normalized difference between two consecutive sums and make an average over a few samples:

$$\Delta\Sigma_n = \sum_{i=0}^{L} (\frac{\Sigma_{n-i} - \Sigma_{n-i-1}}{\Sigma_{n-i}}) \tag{4.5}$$

By carefully choosing the comparison threshold as well as the averaging window $L$, we can thus determine with enough accuracy the time at which the adaptive filter has reached convergence. It is therefore possible to switch at the right time between Geigel and normalized cross-correlation DTD.

### 4.1.3 Overall initialization algorithm

Having found a robust DTD to be used during convergence and being able to determine when the convergence phase is over, we can now picture the overall initialization algorithm. This is illustrated in Algorithm 1.

The first `while` loop in lines 1-3 is there to avoid doing anything before the first reference sequence appears. In line 4 a variable called `cmpt` is initialized. This will be used to check that $\Delta\Sigma$ has been below the threshold $T$ long enough to assume convergence. Then the `repeat` loop from line 5 to 17 performs echo cancellation using Geigel DTD. Line 6 computes the power ratio of echo over reference signal and line 7 computes the decision variable of Geigel DTD. In line 9, adaptive filter's coefficients adaptation is carried out only if no double-talk is suspected. The `if` loop of lines 12 to 16 increments

**Algorithm 1** Initialization algorithm

```
 1: while $\hat{\sigma}_x^2 < T_1$ do
 2:     no adaptation
 3: end while
 4: cmpt $\Leftarrow 0$
 5: repeat
 6:     $r \Leftarrow \delta r + (1 - \delta)\frac{\sigma_y^2}{\sigma_x^2}$
 7:     $\xi \Leftarrow \sigma_x^2 - \beta \sigma_m^2 \cdot r$
 8:     if $\xi > 0$ then
 9:         coefficient adaptation
10:     end if
11:     update $\hat{\sigma}_n^2$, $\Sigma$ and $\Sigma\gamma$
12:     if $\Delta\Sigma < T$ then
13:         cmpt $\Leftarrow$ cmpt $+1$
14:     else
15:         cmpt $\Leftarrow 0$
16:     end if
17: until cmpt $> N_s$
18: AEC using normalized cross-correlation DTD
```

the variable `cmpt` if the gradient of $\Sigma$ is small enough or reset it to 0 if it is not the case. Once we get out of the repeat loop (line 17), the initialization is over, and we will then carry out AEC using normalized cross-correlation DTD.

## 4.2 Noise-robust double-talk detection

In the previous chapter, it has been noticed that the DTD algorithm, although built upon the reliable technique of normalized cross-correlation, was not giving satisfactory results, and had therefore to be modified. In this section, we will present the improvements that have been brought to the original algorithm.

### 4.2.1 Noise-robust DTD

In the paper that introduced the normalized cross-correlation method [7], developments have been mainly focused on the theoretical aspect, and background noise has been ignored, or at least considered as negligible. In normal operation mode, the noise power $\sigma_n^2$ should be much smaller than both echo and near-end speech powers ($\sigma_y^2$ and $\sigma_v^2$). If it is not the case, echo cancellation is simply impossible, because the noise would be much too disturbing to allow the adaptive filter to track the echo.

Therefore, during single talk and double-talk, the noise power is negligible compared to the echo and near-end speech, and equation (2.40) could be approximated as follows

$$\xi^{(2)} = \sqrt{\frac{\sigma_{\hat{y}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_y^2}} \approx \sqrt{\frac{\sigma_{\hat{y}}^2}{\sigma_v^2 + \sigma_y^2}} \tag{4.6}$$

However, during a silence (i.e. when there is neither echo nor near-end speech), the noise term becomes important in comparison to echo and near-end speech powers, and $\xi^{(2)}$ will move towards 0, hence detecting double-talk. One could argue that this is a minor problem, because during silences, the filter does not need to be adapted anyway. But if we consider that all the powers are averaged over at least 512 samples, it means that the double-talk detector reacts with a small delay to the input signal. Thus, there usually is a small latency at the beginning of echo, during which the filter adaptation is prevented. Figure 4.2 shows this problem in more practical terms. The signal on top represents the DTD decision variable. When it reaches its maximum, adaptation is carried out, and when it is at its lower value, the filter is not adapted. The other signal is the echo. We can see that, whenever there is no echo, adaptation is prevented. This is probably harmless in most situations, but in case the echo path has changed during the silence, it could stop adaptation and then lead to a wrong detection of double-talk.

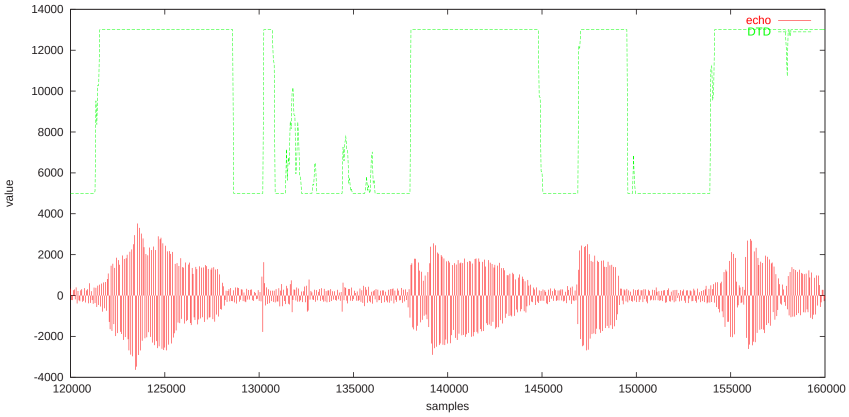


Figure 4.2: Unwanted behavior of the DTD decision variable due to background noise

In order to solve this problem, a rather simple approach has been chosen. It is not possible to remove the noise term appearing in the denominator of equation (2.40), because it is included in the microphone signal. However, we could add an estimation of the background noise power in the numerator, so that it balances the influence of the noise term in the denominator. This is shown in equation (4.7).

$$\xi^{(3)} \equiv \sqrt{\frac{\sigma_{\hat{y}}^2 + \sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_y^2}} \tag{4.7}$$

The term $\sigma_{\hat{n}}^2$ appearing in the numerator is the estimation of the noise power. Since the noise signal usually varies very slowly, it is possible to have a good estimation of its power.

In order for the echo cancellation to be realizable, the background noise must be significantly lower than the echo itself. In this conditions, the additional noise power term added to the numerator of equation (4.7) will not modify significantly the value of the decision variable $\xi^{(3)}$ in single-talk or double-talk situations. However, during silences, the new term brings a significant contribution, and prevent a false alarm from the DTD. In silence, there is neither echo nor near-end speech, therefore $v(t) = 0$, $\hat{y}(t) = 0$ and $y(t) = 0$. Thus equation (4.7) becomes

$$\xi^{(3)} \approx \sqrt{\frac{0 + \sigma_{\hat{n}}^2}{0 + \sigma_n^2 + 0}} \approx 1 \tag{4.8}$$

Hence, with this modified DTD equation, the decision variable will show a correct behavior as will be illustrated in a later chapter.

### 4.2.2 Undesired echo influence

In order to accurately and quickly detect double-talk, one would want the decision variable $\xi^{(3)}$ to move close to 0 as soon as near-end speech signal appears in the microphone. However, looking closely at equation (4.7), we find that when near-end speech is weak compared to echo, the value of $\xi^{(3)}$ is close to 1. Let us suppose that echo and near-end speech have the same power. If we ignore the noise, the value of $\xi^{(3)}$ would then become

$$\xi^{(3)} = \sqrt{\frac{x}{x+x}} = \sqrt{1/2} \approx 0.71 \tag{4.9}$$

If we want to detect double-talk in this case, we need to set the threshold $T$ to a value higher than 0.71. Figure 4.3 illustrates the influence of echo

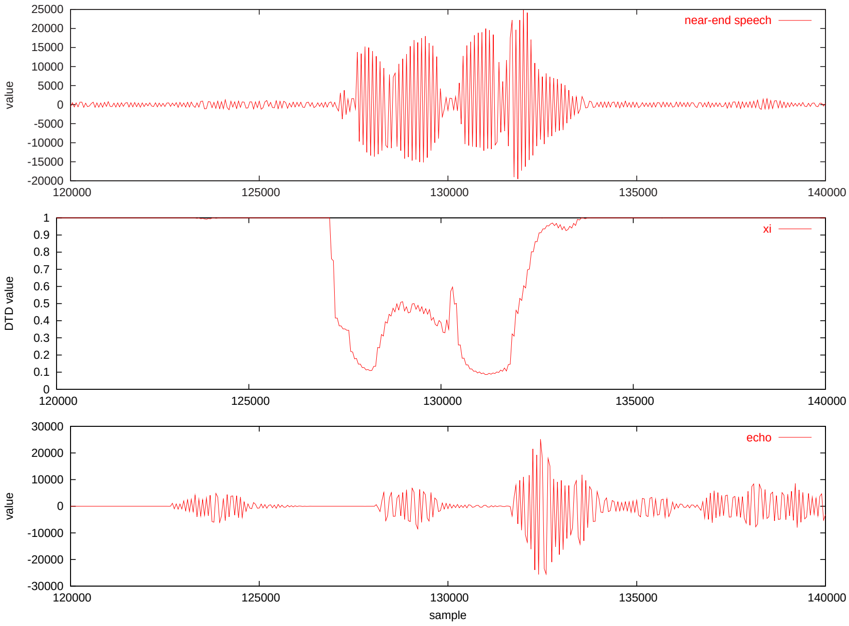


Figure 4.3: Influence of the echo power on $\xi^{(3)}$

power on the DTD decision variable. Note especially how $\xi^{(3)}$ is influenced by the echo around sample 129000 and 132000, causing partial coefficient adaptation too early.

### 4.2.3 Filter misadaptation influence

However, setting the threshold to a high value can also have unexpected consequences. One has to remind that we derived equation (4.7) by doing the assumption that the adaptive filter has fully converged, hence $\hat{\mathbf{h}} = \mathbf{h}$. In reality, though, this is never absolutely true, because $\hat{\mathbf{h}}$ is a finite length filter, and $\mathbf{h}$ is infinite. Moreover, the echo path is always changing a little, and the filter is therefore adapting all the time. Thus, rather than assuming the equality between $\hat{\mathbf{h}}$ and $\mathbf{h}$ (and respectively between $\hat{y}$ and $y$), we could write

$$\sigma_y^2 = \sigma_{\hat{y}}^2 + \Delta \tag{4.10}$$

If we replace equation (4.10) in (4.7) we obtain

$$\xi^{(3)} = \sqrt{\frac{\sigma_{\hat{y}}^2 + \sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_{\hat{y}}^2 + \Delta}} \tag{4.11}$$

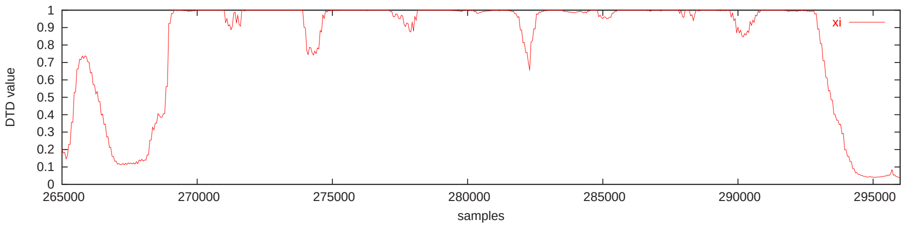


Figure 4.4: Adaptive filter misadaptation influences the DTD decision variable

Now, we can see that the misadaptation of the filter is contributing to change the value of $\xi^{(3)}$. Figure 4.4 shows the variable $\xi^{(3)}$ between two sections of double-talk. We can see that instead of staying all the time to 1, it lowers a little from time to time, due to filter misadaptation.

Therefore, setting the threshold too high would be a bad idea, because all the fluctuations of $\xi^{(3)}$ would be interpreted as double-talk. On the other hand, if we set it too low, we risk to miss some double-talk section, especially in the presence of large echo.

### 4.2.4 Suppressing echo influence

Double-talk detection would be much more efficient if we did not consider echo, but just near-end speech, as expressed in the following equation

$$\xi^{(3)} = \sqrt{\frac{\sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2}}. \tag{4.12}$$

Unfortunately, we do not have an access to the near-end speech signal separately, and the only signals we have are the reference signal $x(n)$ and the microphone signal $m(n)$. To have a close approximation to equation (4.12), we might want to subtract the echo power from both the numerator and the denominator of equation (4.7), using echo replica:

$$\xi^{(4)} \equiv \sqrt{\frac{\sigma_{\hat{y}}^2 + \sigma_{\hat{n}}^2 - \sigma_{\hat{y}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_y^2 - \sigma_{\hat{y}}^2}} = \sqrt{\frac{\sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_y^2 - \sigma_{\hat{y}}^2}} \tag{4.13}$$

We must yet recall that echo and echo replica are not equal. Using equation (4.10) we obtain

$$\xi^{(4)} = \sqrt{\frac{\sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_{\hat{y}}^2 + \Delta - \sigma_{\hat{y}}^2}} = \sqrt{\frac{\sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2 + \Delta}} \tag{4.14}$$

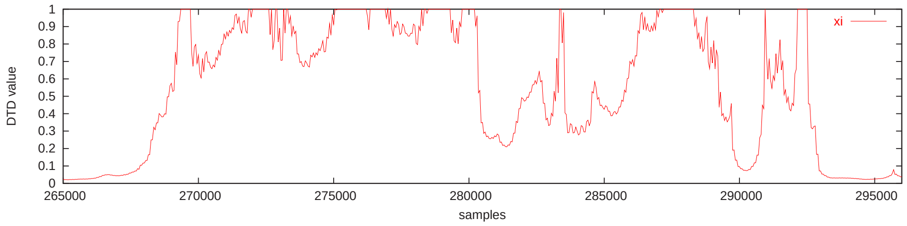


Figure 4.5: Subtracting the echo power can have unexpected results

which is close to equation (4.12), except for the misadaptation term $\Delta$. This additional term is though disturbing a lot the behavior of the DTD decision variable in case of single-talk. Since the noise power is supposed to be quite weak, the smallest echo-path change will seriously lower the value of $\xi^{(4)}$, resulting in double-talk detection. This fact is well illustrated in Figure 4.5, where double-talk occurs from sample 265000 to 270000 and from 293000 to the end of the graph. The rest of the graph contains only single-talk. Although the decision variable reacts quickly and accurately to double-talk sections, it is seriously disturbed by echo-path change during single-talk.

### 4.2.5 Weighted echo power

Considering all these facts, we wanted to find a method that would reduce the influence of the echo on the DTD decision variable, without having the undesirable consequences of equation (4.14). The solution is a compromise between the full subtraction of the echo power and the conventional DTD equation (4.7). The idea is to weight the echo power term, instead of trying to suppress it. This is obtained in the following way: starting from equation (4.7), we subtract $(1-\epsilon)\cdot\sigma_{\hat{y}}^2$ from the numerator and denominator, where $\epsilon$ is a well chosen constant in range $]0;1[$.

$$\xi^{(5)} \equiv \sqrt{\frac{\sigma_{\hat{y}}^2+\sigma_{\hat{n}}^2-(1-\epsilon)\cdot\sigma_{\hat{y}}^2}{\sigma_m^2-(1-\epsilon)\cdot\sigma_{\hat{y}}^2}} = \sqrt{\frac{\epsilon\cdot\sigma_{\hat{y}}^2+\sigma_{\hat{n}}^2}{\sigma_m^2-(1-\epsilon)\cdot\sigma_{\hat{y}}^2}} \tag{4.15}$$

Supposing that the adaptive filter has converged, we can further write

$$\xi^{(5)} \overset{\hat{\mathbf{h}}=\mathbf{h}}{\approx} \sqrt{\frac{\epsilon\cdot\sigma_{\hat{y}}^2+\sigma_{\hat{n}}^2}{\sigma_v^2+\sigma_n^2+\epsilon\cdot\sigma_{\hat{y}}^2}}. \tag{4.16}$$

Using equation (4.16), the DTD decision variable will be much more influenced by the near-end speech than before. If we suppose the echo power equal to the near-end speech power, and if we use $\epsilon = 0.1$, then the average value of $\xi^{(5)}$ will be

$$\xi^{(5)} = \sqrt{\frac{0.1x}{x + 0.1x}} = \sqrt{0.1/1.1} \approx 0.3 \tag{4.17}$$

Compared to the previous value of 0.71, this represent a good improvement in double-talk detection. Practical results on real signals will be shown in Chapter 6.

## 4.3 Robustness to echo

Although the newly introduced DTD equation is enhancing the double-talk detection accuracy, as we will show in Chapter 6, it also has an important drawback: it makes the DTD decision variable also more sensitive to echo-path changes. This is not a problem when those changes occur during double-talk or silence, however it could lead to a critical situation in case of single-talk. To illustrate this fact, we will proceed from the new weighted DTD equation (4.15) and introduce again the filter misadjustment from equation (4.10).

$$\begin{aligned}
\xi^{(5)} &= \sqrt{\frac{\epsilon \cdot \sigma_{\hat{y}}^2 + \sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_y^2 - (1-\epsilon)\cdot\sigma_{\hat{y}}^2}} \\
&= \sqrt{\frac{\epsilon \cdot \sigma_{\hat{y}}^2 + \sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2 + \sigma_{\hat{y}}^2 + \Delta - (1-\epsilon)\cdot\sigma_{\hat{y}}^2}} \\
&= \sqrt{\frac{\epsilon \cdot \sigma_{\hat{y}}^2 + \sigma_{\hat{n}}^2}{\sigma_v^2 + \sigma_n^2 + \epsilon\cdot\sigma_{\hat{y}}^2 + \Delta}}
\end{aligned} \tag{4.18}$$

If we consider the single-talk case, we can assume that there is no near-end speech ($\sigma_v^2 = 0$) and that the noise power is negligible compared to the echo power. Therefore, we can simplify equation (4.18) as follows

$$\xi^{(5)} = \sqrt{\frac{\epsilon \cdot \sigma_{\hat{y}}^2}{\epsilon \cdot \sigma_{\hat{y}}^2 + \Delta}} \tag{4.19}$$

If we do the same simplifications on equation (4.11) representing the old, non-weighted decision variable, we obtain

$$\xi^{(3)} = \sqrt{\frac{\sigma_{\hat{y}}^2}{\sigma_{\hat{y}}^2 + \Delta}} \tag{4.20}$$

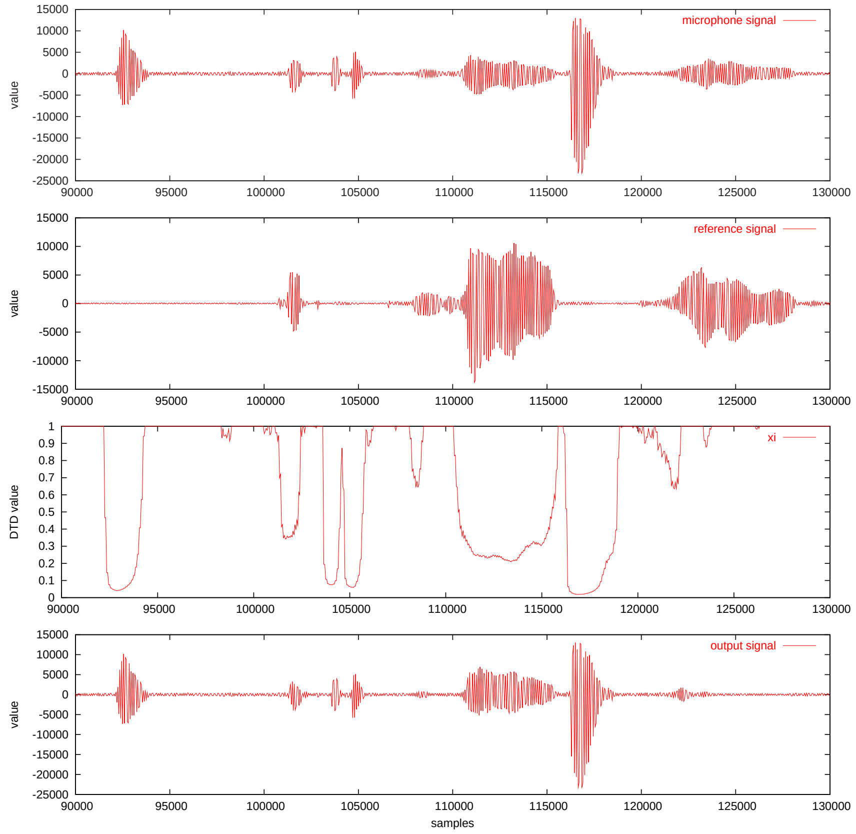


Figure 4.6: Misdetection of double-talk, caused by echo-path change

Observing equations (4.19) and (4.20), we see that the misadaptation term $\Delta$ is more disturbing in equation (4.19), especially if $\epsilon$ is small.

This theoretical observation can, in fact, lead to concrete problems in echo cancellation. Let us suppose that we have a change in the echo path that happens in a silence. At that time, the adaptive filter will not be able to track the echo path change, since we are missing the echo signal. When the echo starts to take non-zero values again, it will be a strong and sudden echo path change, that will influence the DTD decision variable. If this variable crosses the decision threshold, the algorithm will declare double-talk and will stop coefficient adaptation, precisely at a time when adaptation is highly needed. Figure 4.6 shows a signal which causes this problem. The echo between samples 110000 and 115000 is considered as double-talk and therefore not removed.

### 4.3.1 Variable $\epsilon$ parameter

To solve this problem, we have based our reasoning on two facts.

1. the echo-path influence is a real problem only during single-talk sections
2. during silence we do not need to weight the echo power (since it is 0 anyway)

Therefore, the idea to overcome this problem is to make the weighting parameter $\epsilon$ variable in $0 < \epsilon_{min} < \epsilon < \epsilon_{max} < 1$. During the silence, we will increase $\epsilon$ to its maximum value $\epsilon_{max}$, and during echo, we will decrease it up to $\epsilon_{min}$. This behavior is summarized in Algorithm 2, where $\kappa$ is a constant slightly higher than 1, used to smoothly increase or reduce $\epsilon$.

**Algorithm 2** Variation of the weighting factor $\epsilon$

```
if echo == 0 then
    if $\epsilon < \epsilon_{max}$ then
        $\epsilon \Leftarrow \epsilon \cdot \kappa$
    end if
else
    if $\epsilon > \epsilon_{min}$ then
        $\epsilon \Leftarrow \epsilon / \kappa$
    end if
end if
```

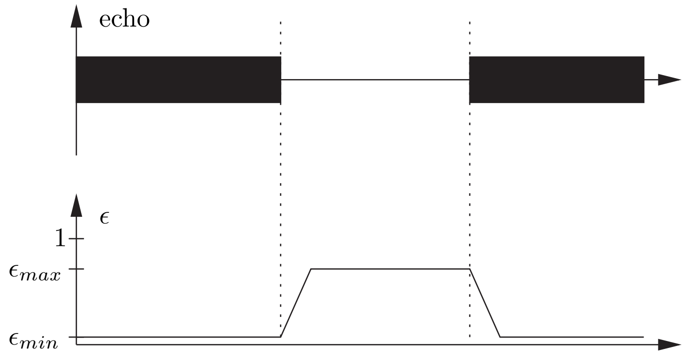


Figure 4.7: Variation of the parameter $\epsilon$ depending on the echo

Figure 4.7 illustrates clearly how the algorithm works. It ensures that at the beginning of an echo section, the variable $\epsilon$ will be close to its maximum value, and thus make the DTD decision variable rather insensitive to echo-path changes. In Chapter 6, we will show how this method is actually working on real signals, helping to avoid a misdetection of double-talk.

The choice of the two constants $\epsilon_{min}$ and $\epsilon_{max}$ is a delicate problem. It is clear that $\epsilon_{min}$ can not be made as small as we want, otherwise we will repeat the problems encountered when trying to subtract completely the echo power from the DTD equation. To select an appropriate value for both constants, we have relied on experiences. The evaluations have shown that the best echo cancellation was happening with $\epsilon_{min}$ around 0.07. A value of 0.5 for $\epsilon_{max}$ seems large enough to avoid too much sensitivity to echo path changes.

## 4.4 Accurate background noise estimation

Background noise estimation is not a very easy task, because we do not have access to the noise as a separate signal. It is rather mixed with echo and near-end speech in the microphone input signal $m(t)$. Nevertheless, several parts of our echo cancellation algorithm rely on noise estimation, and it is therefore important to execute this task with sufficient accuracy.

The conventional algorithm was using the output signal to compute an estimation of the noise power, trying to locate noise-only sequences with the help of a simple heuristic method. However, as we have already seen in Section 3.3.2, the rules were too simple and noise estimation was often computed while near-end speech was present in the microphone signal. This resulted in over-estimation of the noise, which disrupted correct working of other processes relying on noise estimation.

To improve noise estimation, we thus need to discriminate noise-only sections accurately. Instead of identifying noise-only periods, it is easier to detect the presence of echo and near-end speech and stop noise estimation during this period. Choosing the error signal $e(t)$ instead of the microphone signal $m(t)$ for noise estimation is generally better, since echo is attenuated in the former, and a misdetection of a noise section would disrupt less noise estimation.

In order to stop noise estimation in the presence of residual echo, we rely on the criterion defined in [9]. We stop estimation if the power of the output signal is higher than the power of the echo replica

$$\sigma_e^2 > \sigma_{\hat{y}}^2. \tag{4.21}$$

This condition ensures that the echo is weaker than the background noise, and thus the residual echo power will be negligible compared to the noise power. However, used alone, this condition does not prevent to compute

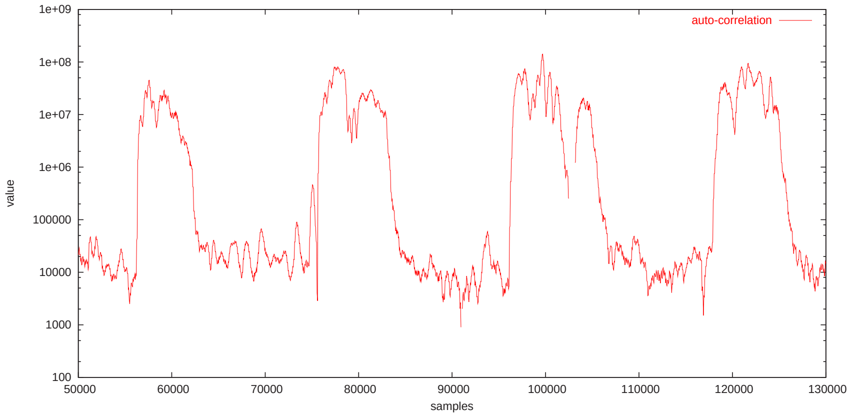


Figure 4.8: Auto-correlation of the output signal

noise estimation during near-end speech sections. Hence the need of an additional condition detecting the near-end speech.

For this purpose, an obvious way would be to rely on DTD and stop noise estimation when near-end speech is detected. We have discarded this technique, however, because it produces an additional loop in the program. The DTD would depend on the noise power estimation, that would in turn depend on the DTD. This kind of inter-dependence should be avoided, because it could lead to instability. Moreover, DTD tends to miss sometimes the beginning of a near-end speech section, when the power is small. While this is not a real problem for the echo cancellation, it is much disturbing for noise estimation. Indeed, it causes the noise power to be overestimated at the beginning of the near-end speech, and then be held at the wrong value during the whole near-end speech period, because the estimation is stopped.

Therefore, we have instead chosen to rely on the auto-correlation of the output signal. The near-end speech is a highly correlated signal whereas the background noise is mainly uncorrelated. Therefore, using the auto-correlation of the output signal allows to accurately distinguish near-end speech from background noise. The computation of output signal's auto-correlation is done by taking the average of the product of two consecutive samples over a chosen averaging window. This averaging window should not be defined too large, such that we have a quick enough response.

$$\rho_n = \sum_{i=0}^{W} e(n-i)e(n-i-1) \tag{4.22}$$

As shown in Figure 4.8, near-end speech and noise section have different

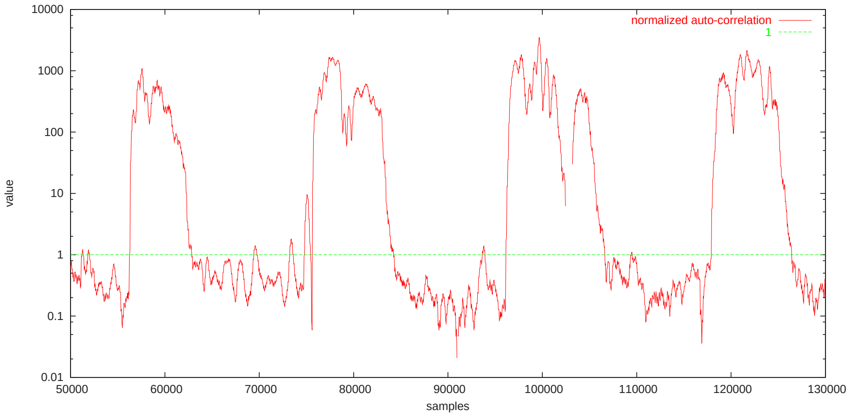


Figure 4.9: Normalized auto-correlation of the output signal

auto-correlation levels, allowing to easily distinguish them. A good threshold should be set just above the noise auto-correlation level, so as to detect near-end speech with a minimum delay. However, the overall auto-correlation level depends on near-end speech and echo power, which can vary much from one signal to another. Thus, setting a good threshold is difficult. Hence the need for normalization of auto-correlation.

Using Geigel DTD, already described in Section 4.1.1, we can roughly avoid near-end speech and compute an approximation of noise auto-correlation. This is done using a small forgetting factor to obtain a smooth approximation.

$$\lambda_n = (1 - \delta)\lambda_{n-1} + \delta \cdot e(n)e(n-1) \tag{4.23}$$

By dividing the output auto-correlation $\rho_n$ by the approximation of the noise auto-correlation $\lambda_n$, we obtain a normalized version of output auto-correlation. In this signal, the noise auto-correlation is very close to 1, as shown in Figure 4.9, independent of the overall signal power. Therefore, setting a threshold to a value slightly higher than 1 will allow to quickly detect near-end speech in the output signal.

One can wonder why not use the autocorrelation technique for DTD, if it has proved to detect near-end speech accurately. The answer is that this technique can not distinguish near-end speech from residual echo. Thus, this method can easily confuse residual echo with near-end speech. It is not a problem for noise estimation, since we also want to avoid residual echo, anyway. However, in the case of DTD, this would prevent the filter from convergence and disturb echo cancellation. Therefore, this method is not suited for double-talk detection.

### 4.4.1 Variable forgetting factor

Computing averages using a forgetting factor has the advantage of using a small amount of memory, compared to averaging with a sliding window. This advantage is especially important when we want an average over a large number of samples. Therefore, this method is particularly well suited for noise estimation, since we want a smoothed yet precise estimation. However, using a forgetting factor is always a trade-off between speed and smoothness. Using a small forgetting factor results in a smooth estimation with slow response to significant changes. A larger forgetting factor has exactly the opposite effect.

For noise estimation, however, both speed and smoothness would be needed, in order to obtain a good estimation. At the beginning of the process, it is necessary to quickly converge towards a reasonable value for noise power, whereas during the rest of the process, we would like a precise and slow varying estimation. Hence the idea of using a variable forgetting factor for noise estimation. This would be set to a value close to 1 at the beginning, and will then be progressively lowered as the estimation is approaching the true value of the noise power.

In order to know when to reduce the size of the forgetting factor, we rely on the variations of the gradient of noise estimation. Every time this gradient changes its sign, the forgetting factor is slightly reduced until it reaches its final value. Therefore, the estimation reaches quickly a value close to the real noise power and then continues adaptation smoothly. This technique is explained in Algorithm 3, in which $\tau$ represents a constant a little higher than 1 used to slowly reduce the forgetting factor $\delta$.

**Algorithm 3** Variable forgetting factor

```
gradient_sign $\Leftarrow$ 1
if sgn(gradient) $\neq$ gradient_sign then
    if $\delta > \delta_{min}$ then
        $\delta \Leftarrow \delta/\tau$
    end if
    gradient_sign $\Leftarrow$ sgn(gradient)
end if
```

# Chapter 5

# Enhancement on speech recognition

Our acoustic echo cancellation algorithm is specifically targeted at speech recognition. Therefore, we have also added several parts to our algorithm, that are not found in usual echo cancellation algorithms, whose goal is to help the speech recognition engine to perform its task. Center-clipping is performed to get rid of the residual echo. Moreover, white noise is added to the output signal to stabilize its power.

## 5.1 Non-linear residual echo removal

No matter how good an adaptive echo canceler is, the echo will never be perfectly canceled and some attenuated version of the echo, called *residual echo*, will still be present in the output signal. This is inherent to the principle of adaptive echo canceler, which generates an echo replica using an FIR filter. Since echo path is an FIR filter with infinite number of taps, even when the adaptive filter matches the echo at best, some echo components will still pass through the system, due to the unmodeled *echo tail*. This echo replica still has some speech characteristics and is therefore disturbing the speech recognition.

One common way to remove this undesired residual echo is to use a non-linear center clipping method on the output signal of the echo canceler. The principle of this method is fairly simple: it removes every sample that is below a given threshold $T_c$. Since the residual echo is generally a very low power signal, it should be removed by this method. During non-speech sections, the background noise is also suppressed by center-clipping. To make the method adapt to different signals, the threshold $T_c$ for center clipping is variable and depends on the noise power estimation.

However, applying uniformly center clipping on the output signal from echo cancellation is not conceivable, because it would be applied also to near-end speech sections, distorting them and making them unrecognizable by the speech recognition engine. Therefore, center-clipping should be applied selectively on the non-speech section, so as to avoid damaging near-end speech. To achieve this purpose, we use the DTD signal to prevent performing center clipping during double-talk sections. Unfortunately the double-talk detection is not always perfect and it happens that it misses some parts of the double-talk period, particularly the beginning and the end, when the signal power is weak. When such a miss happens, the corresponding near-end speech signal will be distorted, not to say removed if its power is weak.

To overcome this problem, we have designed a method that applies center-clipping selectively on different parts of the signal. Near-end speech power is generally much higher than residual echo power. Thus, most of its samples are higher in magnitude and power than the center clipping threshold. On the other hand, when it comes to the residual echo, the majority of its samples should be below the threshold $T_c$.

Therefore, we can base the decision whether clipping a sample or not, on the power of the previous samples. If the majority of them are higher than the threshold $T_c$, we are probably dealing with a near-end speech section, and it is better not to clip the current sample. The following algorithm summarizes the center-clipping decision method.

**Algorithm 4** Selective center-clipping

```
if majority of $n$ past samples $< T_c$ then
    clip current sample
else
    let current sample untouched
end if
```

With this algorithm, near-end speech that was not detected by DTD has little distortion, and at the same time most of the residual echo is canceled.

## 5.2 White noise addition

Center-clipping described in the previous section is a successful way to get rid of the annoying residual echo. To further boost the results of the voice recognition, we have also tried to add white noise to the output signal. Indeed, white noise addition in the non-speech section helps the speech detection of the voice recognition program, by making the signal power more stable. However, white noise addition should not be added uniformly on the whole output signal. Its power should depend on the output signal. We should add less white noise during near-end speech section, so as not to distort the speech signal. During non-speech section, the added white noise should be stronger, so as to cover the possible residual echo. For deciding whether to add strong or weak white noise, we rely again on the DTD decision variable. The overall white noise power depends on the logarithm of the output signal power. This technique is derived from a noise suppression program for *PaPeRo*.

One important issue about white noise addition was whether to place it before or after center-clipping, and whether both methods were efficient when used together. After some experiments, we found out that the best configuration was to first add white noise and then perform center clipping. Both methods used together in that configuration give better results than each of them separately.

# Chapter 6

# Evaluation

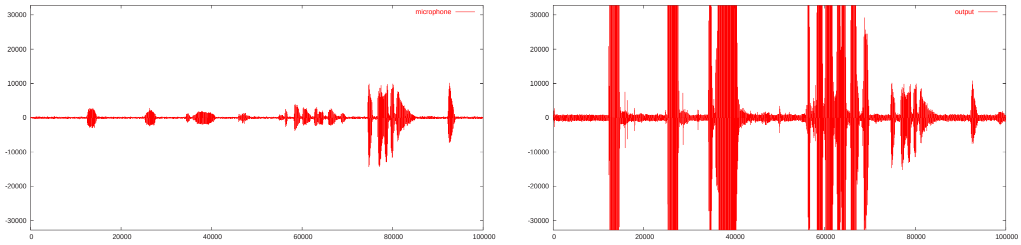
In this chapter, we show the results achieved by the new echo cancellation algorithm. In a first part, we illustrate by concrete examples, how the new developments can contribute to improve the quality of echo cancellation. We then present the method we have designed for evaluation of the algorithm. In the next section, the results of the evaluation are shown and analyzed. Finally, we present an overview of the final acoustic echo cancellation algorithm.

## 6.1 Echo cancellation

### 6.1.1 New initialization model

In chapter 3 we have seen how the too simple initialization performed in the initial program could lead to instability. Figure 6.1 illustrates more concretely this problem. The left signal (6.1(a)) represents the microphone input signal on which echo cancellation is performed. The bursts up to sample 70000 are echo, whereas the next are near-end speech. The signal on the right (6.1(b)) is the output signal after echo cancellation. We can notice that echo is not canceled but rather amplified. This phenomenon can be explained as follows: the original algorithm forced filter adaptation during the first samples of the signal. Since there is only noise, and since the noise estimation is initializing at this time, the variable step-size for coefficient adaptation will not prevent adaptation and the filter will adapt to the noise. Then, the DTD will be disturbed by the filter's wrong adaptation and will prevent further adaptation during echo sections.

In such a case, the new initialization method will avoid this problem. As we can see in Figure 6.2(b), the new program first stops filter adaptation until the first echo period is seen (around sample 12000). Then it uses Geigel DTD

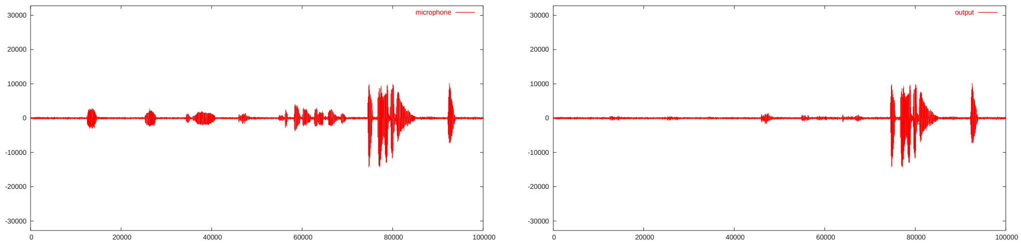


(a) The microphone input signal

(b) The output signal after echo cancellation

Figure 6.1: Results of a bad initialization

to carry out double-talk detection, until the adaptive filter has converged to a stable value (around sample 40000). Double-talk detection is then performed by the normalized cross-correlation method.



(a) The microphone input signal

(b) The output signal after echo cancellation

Figure 6.2: Echo cancellation with correct initialization

Thanks to this new initialization method, the echo cancellation program is more robust, and can be used in different environments with various signals, without requiring a special set-up from the user. There is no need of providing an echo-only signal at the beginning, so that the filter can adapt at the first time. The algorithm can rather deal with any kind of signals and ensure convergence.

### 6.1.2 New DTD

In Section 4.2.1, we saw that the original definition of the double-talk detector based on normalized cross-correlation was not designed to cope with background noise, and was reacting incorrectly during silences. Therefore we designed a new DTD equation using noise estimation. In Figure 6.3, we display the DTD decision variable $\xi^{(4)}$ applied to the same signal as figure 4.2, using the new DTD equation. The double-talk detector does not detect double-talk during silences, and reacts accurately to every near-end speech section.

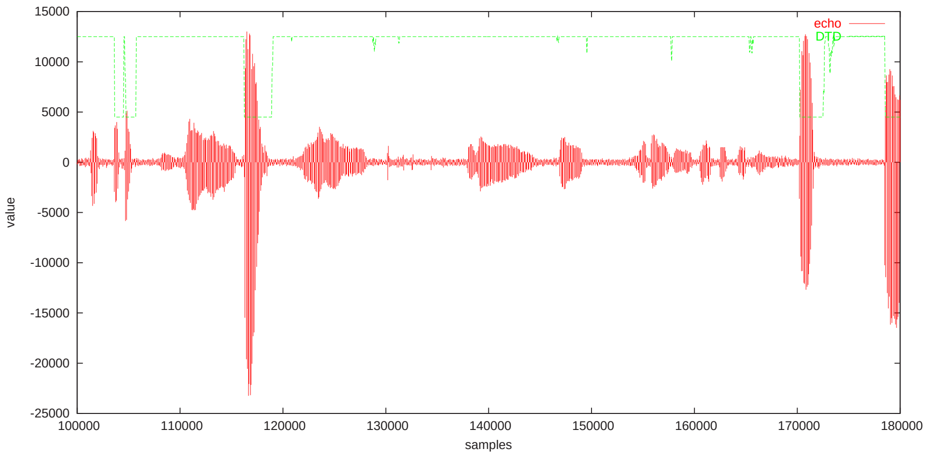


Figure 6.3: DTD decision variable using the new equation with estimated noise term

The second point in the new DTD algorithm was the weighting of the echo term, in order to limit its influence on the decision variable. Figure 4.3 showed how echo was disturbing double-talk detection when the original equation was used. Figure 6.4, displays the decision variable by the new DTD definition, applied on the same signal as in Figure 4.3.

The new decision variable does not suffer anymore from a too strong influence from the echo signal. Although, this influence is still noticeable, it does not disturb the double-talk detection. If we set the threshold for double-talk detection around 0.6 or 0.7, the double-talk period will be accurately detected from the beginning to the end.

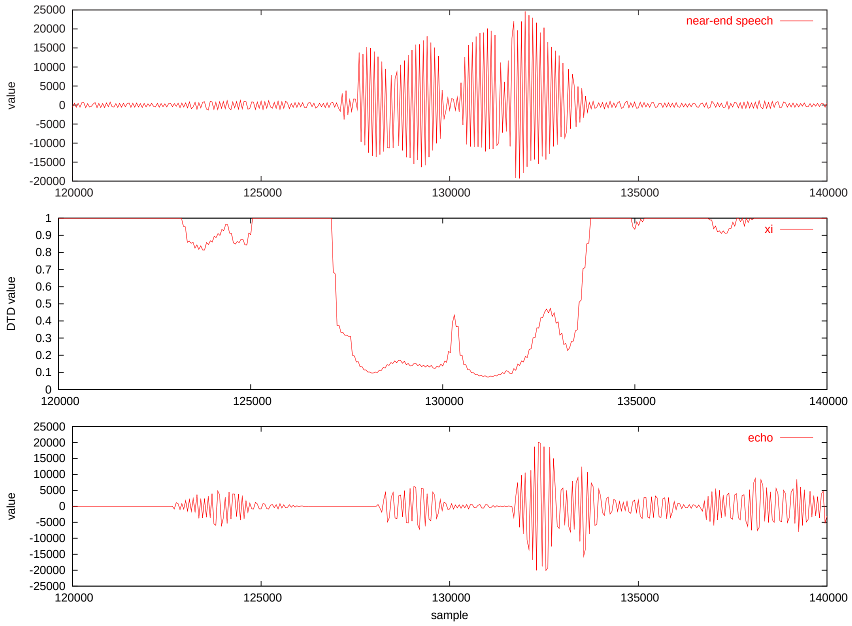
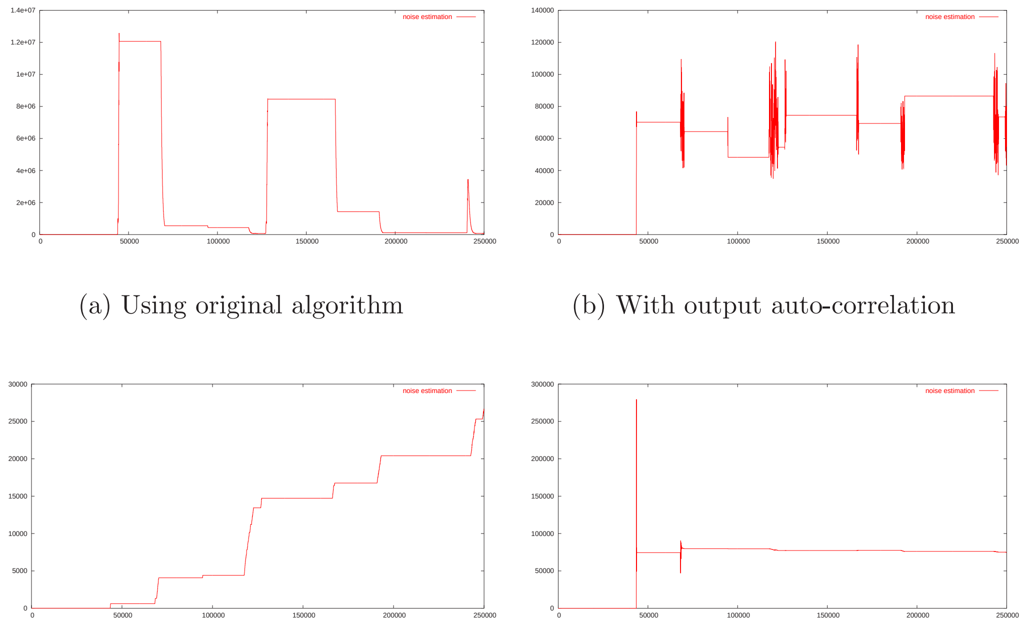


Figure 6.4: DTD decision variable using the weighted echo term is less disturbed by echo sequences

### 6.1.3 New noise estimation method

In Chapter 3, we discussed the simple noise estimation algorithm used by the original AEC algorithm and its consequences. We saw that the simple rule it was using was not good enough to avoid estimating the noise during near-end speech. Figure 6.5(a) concretely illustrates the consequences of estimating noise power at the wrong moment. Noise is much over-estimated all along the signal, and particularly around samples 50000 and 150000, where it reaches a high value. One should look at the scale to have a good measure of how wrong the estimation is, knowing that the true value of background noise power for this signal is about 70000.

To improve this result, we have relied on output auto-correlation to avoid estimating noise during near-end speech. This makes the estimation much more accurate, as shown in Figure 6.5(b). The estimation now always stays in the range of the true noise power and does not reach unrealistic values. However, although the estimation is now reasonable, it is not as smooth as we could expect. In order to get a smoother noise estimation, the easiest solution is to lower the forgetting factor. However, as already explained in Chapter 3, this also slows down the tracking of the estimation. Since the background noise is a rather stable signal, whose magnitude does not change



(a) Using original algorithm

(b) With output auto-correlation

(c) With small forgetting factor

(d) With variable forgetting factor

Figure 6.5: Noise estimation using different methods

fast, this is acceptable, except for the initial tracking, where the estimation will converge much too slowly towards the true value. This problem is shown in Figure 6.5(c), where noise estimation is done using output auto-correlation and a very small forgetting factor. Finally, Figure 6.5(d) shows the noise estimation on the same signal, using a variable forgetting factor. This time, we can combine both fast tracking and smooth estimation in the steady state.

## 6.2 Evaluation for speech recognition

The goal of the acoustic echo canceler for *PaPeRo* is to enhance speech recognition. Therefore, an important criterion to measure the performance of our algorithm is the word recognition rate of the speech recognition engine, after echo cancellation. For this evaluation purpose, a total of 240 sets of echo and reference signals have been recorded in real conditions with *PaPeRo*. The recording consists of *PaPeRo* continuously talking, and someone speaking words to the robot for recognition. The reference signal is the synthetic voice of the robot, and the echo is captured by *PaPeRo*'s microphones.

| | | No echo | AEC &<br>low echo | AEC &<br>loud echo | no AEC &<br>low echo | no AEC &<br>loud echo |
|---|---|---|---|---|---|---|
| Head mic. | 0.5m | 94.2% | 91.3% | 86.5% | 26.6% | 13.7% |
| | 1.5m | 93.1% | 88.7% | 81.4% | 19.6% | 10.0% |
| Neck mic. | 0.5m | 92.7% | 87.3% | 82.1% | 14.3% | 8.0% |
| | 1.5m | 91.7% | 84.5% | 76.4% | 9.1% | 5.6% |

Table 6.1: Results of the evaluation in word recognition rate

Each recorded file features 50 words pronounced at a regular interval. The sequences were recorded using different parameters to simulate the real-life conditions: several different speakers were used as near-end speech, males, females and children. The speaker was placed at 0.5m and 1.5m from the robot. Also the recordings were done using two different volume levels for *PaPeRo* voice, one high and another relatively low. The near-end signal was recorded by two different microphones, one located on the robot's head and the other one on its neck.

To evaluate the new algorithm, the following procedure has been taken. First, echo cancellation is performed on every evaluation sequence, thus producing 240 new output sequences. Then speech recognition is applied to every output sequence. The speech recognition engine produces then for each sequence a text file containing the recognized words. Finally, another C program was used to compare the results of the speech recognition with reference files comprising the true word sequences pronounced by the corresponding speakers.

To take care of the whole evaluation process automatically, a set of `bash`[1] scripts have been written. This was needed, since the process is relatively long and can take up to two hours.

## 6.3 Word recognition rate improvement

Figure 6.6 and Table 6.1 display the results in word recognition rate of the evaluation on the new algorithm.

Several things can be learned from the voice recognition results. First and most obvious, we can see that echo cancellation improves the speech recognition rate as much as 80% in different recording conditions.

Then, we can notice, as we could have expected, that the louder the echo, the lower the word recognition rate. This can be partly explained

[1] `http://www.gnu.org/software/bash/bash.html`

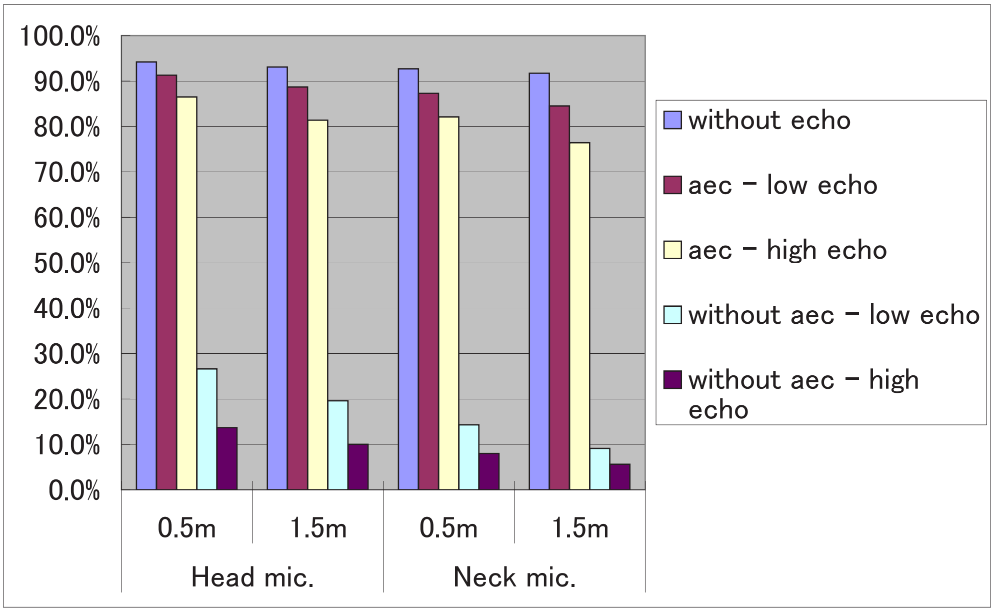


Figure 6.6: Results of the evaluation in word recognition rate

by the fact that double-talk detection is more difficult when both near-end speech and echo have almost the same power. Also, coefficient adaptation is stopped during double-talk, and therefore if the echo path changes during this time, a loud echo will produce a more disturbing residual echo. For the same reason, the word recognition rate is usually lowered as the speaker gets far from the robot. When the distance between the speaker and the robot increases, the near-end speech will get weaker compared to the echo, hence more difficulties in detecting near-end speech.

Finally, we can see that the overall results of voice recognition for the signal without echo is 92.95%, and 85% for the signal with echo cancellation. Therefore, there is less than a 10% difference between the clean speech and the noisy one. It means that the objective expressed in the introduction are reached, although there is still place for improvement.

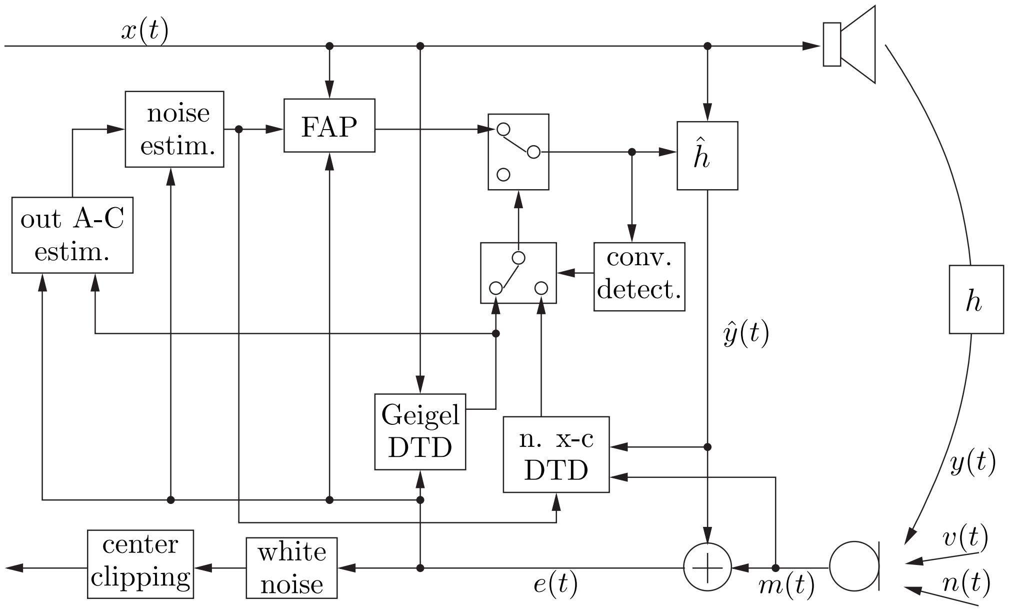


Figure 6.7: Overview of the echo cancellation algorithm

## 6.4 Algorithm overview

After having described individually the new developments and enhancements brought to the AEC algorithm in Chapters 4 and 5, we will briefly see here how all pieces are put together to form the echo cancellation program. Figure 6.7 depicts an overview of the new echo canceler.

Figure 6.8 illustrates the sequence of operations to perform echo cancellation. First an echo replica is generated using the current adaptive filter coefficients, and the output signal is created by subtracting the echo replica from the microphone signal. Then auto-correlation of the output signal is computed, using Geigel DTD. Based on output auto-correlation, noise-only sequences are then identified. If the microphone signal consists only of noise, then noise estimation is performed. After this, double-talk detection is realized using Geigel DTD during the convergence phase, or normalized cross-correlation DTD after convergence. If no near-end speech signal is present, the filter coefficients are then adapted. After that, convergence detection is done. Finally, white noise is added and center clipping is applied to the output signal.

This whole sequence is then repeated to process the next sample. This loop should continue as long as echo cancellation is desired.

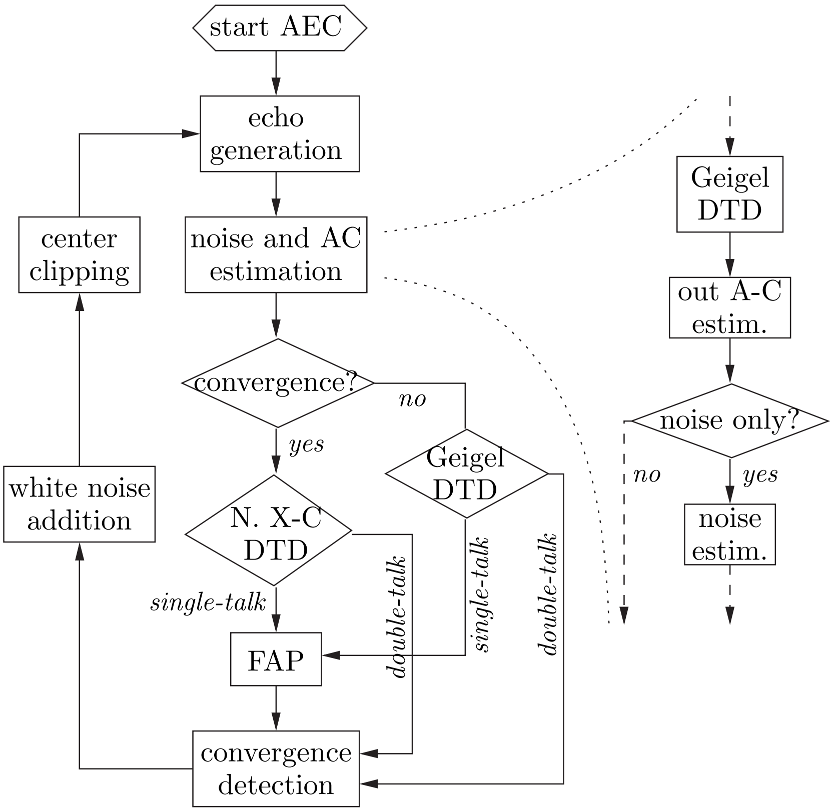


Figure 6.8: Flowchart of the echo cancellation process

# Chapter 7

# Conclusion

## 7.1 Conclusion

In this report, we have presented our master thesis work at **NEC** Corporation. During the 6 months of the internship, we have developed an acoustic echo canceler to improve the speech recognition of the robot *PaPeRo*. Starting from an acoustic echo cancellation algorithm written by a former internship student, we have analyzed it and identified its flaws, which were compromising its performance. We have then developed various techniques to correct and enhance the original program. Among the newly developed features, a new double-talk detection algorithm based on normalized cross-correlation has been developed. This noise-robust algorithm provides a fast and accurate response to double-talk, preventing near-end speech to disturb the echo cancellation process. A new background noise estimation algorithm has also been developed, providing a smooth yet precise noise power estimation for the other parts of the echo cancellation algorithm. Special care has been given to the algorithm initialization, in order to ensure that the echo cancellation can be performed on any signal in any conditions. Finally, a center clipping method has been implemented along with the AEC algorithm to further remove the residual echo that pass through echo cancellation. White noise has also been added to the output signal, in order to make the signal power more stable and thereby improve speech recognition results.

Validation of the developed algorithm has been done through a series of evaluations over a large set of echo sequences. The application of the newly developed echo cancellation algorithm on sequences corrupted by echo has proved to enhance word recognition rate of the speech recognition engine by more than 70%. The results of the speech recognition on noisy signals are therefore within 10% degradation from those for noiseless signal, which fulfills the objective of this master thesis work.

Thanks to the new methods developed during this master thesis, the echo cancellation algorithm for *PaPeRo* is now more robust in difficult conditions. Echo is removed better, with minimal distortion to the near-end speech, which is critical to reach good speech recognition performance.

## 7.2 Future work

Among the future work that can still be done about the echo cancellation algorithm, one interesting feature would be to automatically detect the optimal adaptive filter length for every signal, so as to optimize convergence speed without compromising echo cancellation.

Although, *PaPeRo* is soon moving to a new and more powerful hardware, some work could be done to optimize the resource consumption of the AEC algorithm, in order to be nice to the other processes running on the robot.

Despite the fact that the algorithm has been widely tested, a series of evaluations in which people and the robot move, in order to constantly change the echo path, could be of interest.

# Bibliography

[1] Yoshiro Fujita. Personal robot papero. *Journal of Robotics and Mechatronics*, 14(1):60–63, 2002.

[2] M. M. Sondhi. An adaptive echo canceller. *Bell System Technical Journal*, 46:497–511, 1967.

[3] Kazuhiko Ozeki and Tetsuo Umeda. An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties. *Electronics and Communications in Japan*, 67-A(5):126–132, February 1984.

[4] Donnald L. Duttweiler. A twelve-channel digital echo canceler. *Communications, IEEE Transactions on*, 26(5):647–653, May 1978.

[5] Hua Ye and Bo-Xiu Wu. A new double-talk detection algorithm based on the orthogonality theorem. *IEEE Transactions on Communications*, 39:1542–1545, November 1991.

[6] Tomas Gänsler, Maria Hansson, Carl-Johan Ivarsson, and Göran Salomonsson. A double-talk detector based on coherence. *IEEE Transactions on Communications*, 44:1421–1427, November 1996.

[7] Jacob Benesty, Dennis R. Morgan, and Jun H. Cho. A new class of doubletalk detectors based on cross-correlation. *IEEE Transactions on Speech and Audio Processing*, 8:168–172, March 2000.

[8] Yuisuke Maruyama. A fast method of projection algorithm. Technical report, NEC Corporation, 1990.

[9] Akihiro Hirano and Akihiko Sugiyama. A noise-robust stochastic gradient algorithm with an adaptive step-size suitable for mobile hands-free telephones. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1392–1395. IEEE, May 1995.

[10] Yutaka Hiratani, Akihiro Hirano, and Masaya Kanazawa. A noise-robust echo canceller on v830 multimedia risc processor integrated into a car navigation system. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1753–1756. IEEE, May 1998.