Algorithms in Signal Processors EITN80
Adaptive Echo Canceller

Kalle Friman Eriksson, Axel Jonsson, Måns Åhlander

2017/03/13

1 Introduction

Echo cancellation is a feature that is widely used in technological devices that both can play and record audio. Often it is desirable that the sound played is not audible in the recorded sound. Example of this are the speakerphone feature in mobile phones, it would be awkward if you would hear yourself as clearly as the person you are talking in phone with if the person is using the speakerphone function. What is happening in the phone is that it adjusts a filter which the sound signal that is played also is filtered through. The sound that enters the microphone is converted to a signal which is subtracted by the signal that has been filtered in the telephone. What remains is the incoming sound to the microphone which the phone has not played.

To get a deeper understanding of signals, algorithms and how to implement them on hardware, has a project about echo cancellation been done. In the report, will a solution for how to implement algorithms on a DSP device that can cancel sound which itself have played will be presented. The report consists of the following chapters: theory, implementation, result, discussion and appendix. The chapter on theory describes the LMS-algorithm which we use to adapt a filter and how we used it. Implementation will focus on the DPS hardware, the C-programming needed and the problems we got during the implementation. In the chapters results and discussion will the result be presented and discussed.

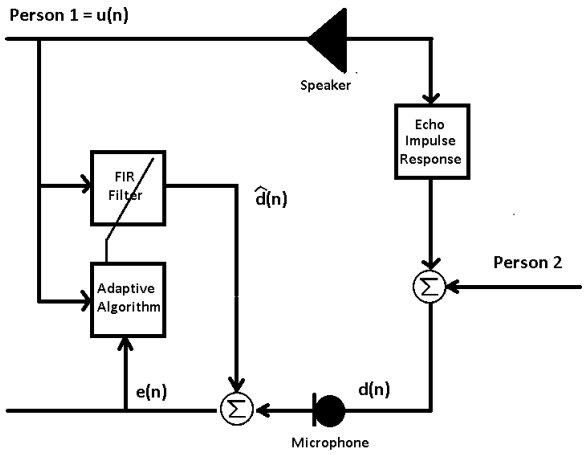

Figure 1: Block diagram of the system.

In order to create a successful echo cancellation an algorithm that design an adaptive filter must be implemented. The filter must match the filter that cause the change of the audio signal as it went through the room from the speaker to the microphone. By using an adaptive filter a model of the echo can be created.

If the real signal is subtracted by the model of the echo we get the difference of the signal which is suppose to be the sound entering the microphone that hasn't been played in the speaker. In figure 1 we can see a block diagram of the filter. The signal $u(n)$ represent the audio that will be played in the speaker, $d(n)$ is the signal which consist of person 2 speaking and the echoed $u(n)$ signal. $\hat{d}(n)$ is the model of the echo and $e(n)$ is the difference $e(n)$=$d(n)$-$\hat{d}(n)$ which is only suppose to be the sound from person 2. Because the audio signal entered the DPS in 32 bits at a time it is required that the algorithm used 32 bits at a time. Due to that the filter which cause change of the real signal was unknown, the length of filter which the algorithm create will also be unknown but it must be set to be able to match the real filter. It also required understanding of C programming and the framework for the DSP to successfully implement the algorithm.

2 Theory

2.1 The Wiener Solution

The problem illustrated in figure 1 is on the form of a general Wiener filter. The Wiener filter consists of an input signal $u(n)$, a desired signal $d(n)$, the convolved input signal $\hat{d}(n)$ and the signal $e(n)$, which is difference between the desired signal and the convolved input. The adaptive filter to be modeled will be a FIR-filter of an order which is preset before the algorithm is used. We will refer to this order as M. The filter will thus have M different coefficients, $\hat{w}$, which will have to be estimated. The reconstructed echo will then become: $\hat{d}(n) = \hat{w}^H \bar{u}(n)$, where $\bar{u}(n) = [u(n) \quad u(n-1) \quad ... \quad u(n-M-1)]$.

If the statistics about the input and the desired signal is known and the signal is assumed to be wide-sense stationary there exists thus an optimal Wiener solution which will minimize the cost function:

$$J = E\{|e(n)|^2\} \tag{1}$$

where $E\{\cdot\}$ denotes expectation.

If R is the known correlation matrix of $u(n)$ and p is the cross-correlation vector between $u(n)$ and $d(n)$ we can find the optimal Wiener solution w_o as:

$$w_o = R^{-1}p \tag{2}$$

2.2 The LMS Algorithm

The problem with finding the Wiener solution is that the correlation matrix R and the cross-correlation vector p must be known to be able to find the Wiener solution w_o. By instead using estimates of R and p, namely $\hat{R}$ and $\hat{p}$ respectively. These estimates are calculated using only the current samples $\bar{u}(n)$ and $\bar{d}(n)$. We can write the cost function to the form as if the current value of $|e(n)|$ is the mean of $|e(n)|$:

$$J = |e(n)|^2 \tag{3}$$

For this cost function a steepest-descent method is used to find the approximate Wiener solution. By taking the gradient of the cost function together with $\hat{R}$ and $\hat{p}$, the following update algorithm for the filter coefficients $w(n)$ can be derived:

$$\hat{w}(n+1) = \hat{w}(n) + \mu\bar{u}(n)e^*(n) \tag{4}$$

where μ is a user-defined parameter to determine the step-size of the algorithm. Because the LMS algorithm uses the current value of $|e(n)|$ instead of the mean, the algorithm will never converge to the exact wiener solution w_o. Instead the algorithm will converge to a point near the wiener solution which is dependent

on the current value of $|e(n)|$. This point is determined similar to the wiener solution as:

$$w_o = \hat{R}^{-1}\hat{p} \tag{5}$$

The algorithm will converge if the chosen step size μ is chosen to fulfill the following relation:

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{6}$$

where λ_{max} is the largest eigenvalue of the correlation matrix R.

3 Implementation

3.1 Preparations for coding

A flow chart of the desired signal processor was given by the supervisor, as shown in figure 2. The next step was to derive a block diagram which is shown in figure 1. From this the LMS-algorithm could be determined.

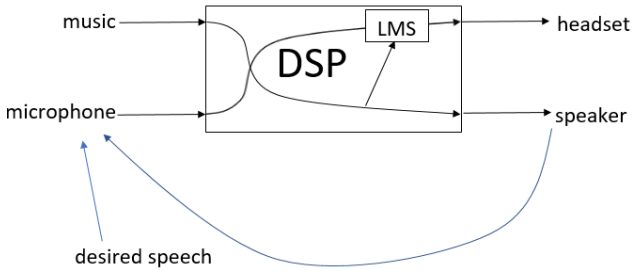

Figure 2: Flow chart of the desired signal processor.

3.2 Coding

The LMS-algorithm was then translated into matlab code in order to see if it worked. The algorithm was tested in matlab with a string of randomized values as input. When the algorithm was working satisfactory the input was divided into blocks. This was done since it needs to be this way when the algorithm is applied on the DSP. When the algorithm and the signal was handled in blocks in matlab the code needed to be rewritten in C. Just as in matlab the first step was to get the LMS-algorithm to work without the input being split into blocks. When this worked the division of the input into blocks was implemented in C as well. The inputs were still the same as in matlab to be able to see if the filter coefficients that got generated were the same in matlab and C. When the generated filter coefficients were the same in matlab and C the input was changed. The input was now from the microphones. Flash programming was implemented in order to run the algorithm without having the DSP connected to a computer.

3.3 Hardware

The processor that's used is a SHARC ADSP-21262. Four microphones, two input ports and two output ports are connected to the processor. For the echo canceller to work the two output ports needed to be able to output different signals. This was done by changing the framework of the processor. The processor is connected to a computer with a HPUSB-ICE emulator platform. This allows the code to be transfered to the processor and make it possible to run an

emulate session. Music is played by a cellphone and is connected to one of the input ports with an AUX-cable. A headset is connected to one of the output ports and a loudspeaker is connected to the other output port. A microphone is connected to the input port. The microphone records the music that is being played from the speaker and the voice of the person speaking. The desired signal is sent out through the headset.

4 Result

A few experiments were done to evaluate the performance of the echo cancellation. In figure 3 a signal's amplitude is shown where some music is played. On the first half of the test the algorithm is turned on and such that the amplitude of the music picked by the microphone is reduced. On the next half of the signal a person is talking in to the microphone without interfering with the echo cancellation. These results were also confirmed by listening to the signal.

The next test which was performed tested the convergence speed by setting different step sizes. The step size value is referred to as mu. In figure 4 it is seen that the larger step size $mu = 0.02$ is converging much faster whilst the setting $mu = 0.005$ has not performed a better cancellation even after 20 seconds of runtime.

In the last test the convergence was tested after the environment has been changed. Ten seconds into the recording while the algorithm is turned on, the speaker was quickly moved 30 cm away from the microphone. Figure 5 shows the adaptation to the new environment for two different step sizes. In the result plots the vertical axis represents the amplitude and the horizontal axis the current time.

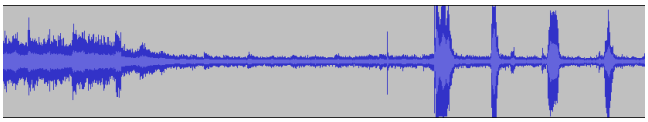

Figure 3: Amplitude plot for testing convergence and speech interference.

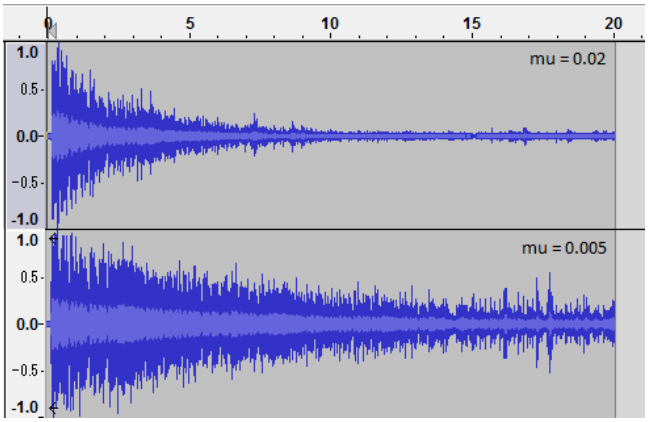

Figure 4: Amplitude plot for testing convergence for different step sizes mu.

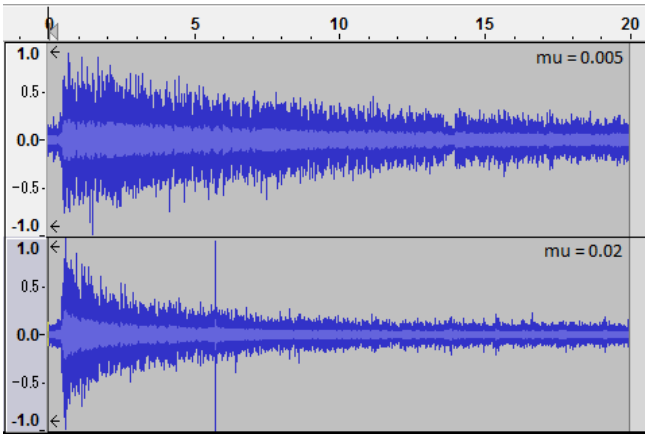

Figure 5: Amplitude plot for testing convergence for different step sizes after speaker is moved mu.

5 Discussion

We did not have any problems with the implementation of the LMS-algorithm. It was easy to use and easy to implement, both in MATLAB and in C. We had some issues with deriving a suitable filter length because of the performance of the DSP. A faster DSP with a bigger memory would allow a use of filters with more coefficients which would boost the performance greatly because the algorithm is linearly dependent of the number of coefficients.

The DPS we have still works well and the music in the background gets canceled satisfactorily. We had to reduce the sample rate in order to get a filter that had enough coefficients. This lowers the sound quality but the cancellation is improved. The choice of step size has shown to greatly change the performance of the algorithm. A larger step size will make the filter converge faster to the solution but with a greater gradient noise. This will result in a solution with less music being canceled. If a smaller step size is chosen the algorithm will converge slower but with less gradient noise.

The choice of step size is therefore very dependent on the application. For example if the environment is very static a low step size can be used in order to get as low gradient noise without any drawbacks compared to if the environment would be changing. In that case a larger step size would be needed. Otherwise the algorithm would not be fast enough to converge to the solution, since the solution would change faster then the convergence of the filter. There is also another risk of increasing the step size as the system may become unstable, which is completely dependent on the input signal. Therefore one must be cautious when increasing the step size. This problem could be avoided by for example implementing the normalized LMS algorithm, which also considers the norm of the input signal in the update algorithm.

White noise is still present, to get rid of this another filter is needed. Another

way to improve the performance is to improve the algorithm, with fast LMS for example. Our algorithm depends on two convolutions which is the biggest contribution to the time complexity of the implementation. Fast LMS would make require less computational power and therefor be faster.

6 Appendix

6.1 Meetings

Date	Subject
2017-01-18	handledarmöte: fick echo tilldelat. Grundläggande genomgång
2017-01-19	grupparbete: LMS teori och matlab
2017-01-25	handledarmöte: introducering av block
2017-01-26	grupparbete: implementerade block
2017-01-30	handledarmöte: utdledning och genomgång av hårdvara
2017-02-01	skrivit C kod
2017-02-06	Handledarmöte: Nästa moment är att få C-kod som funger för samplingsblock samt testköra koden på hårdvara.
2017-02-08	Arbete med att få fungerande C-kod för samplingsblock. Svårigheter med cirkulärminneshantering. Även småproblem med LMS-algoritmen. Görs just nu på ett element i blocket men måste göras på alla.
2017-02-13	handledarmöte. bekräftning av konvergens
2017-02-16	inläsning av data, ramverk
2017-02-17	försökt få genom ljud
2017-02-20	handledarmöte. fick igång ljudet. fixade framework
2017-02-21	måns fixade!
2017-02-22	uppspaltning av rapport
2017-02-27	handledarmöte samt fixat flash