Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Real-Time Automatic Gain Control for Singing Voice Applications

Bachelor Thesis in the course Informatik

Author:
Nils Heine

Matriculation Number: 6703759

Signal Processing / Signalverarbeitung
Department of Computer Science, MIN Faculty

First Reviewer:	Prof. Dr.-Ing. Timo Gerkmann
Second Reviewer:	Dr.-Ing. Martin Krawczyk-Becker

Hamburg, October 19, 2017

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

When a sound engineer edits a song he wants all the recorded audio tracks to be perceptible in the final mix (apart from some special cases). Most important is this for audio tracks with notably significance for the musical piece. In this thesis we will deal with vocal tracks which often have a significance for the meaning, main melody or recognition value of the song. The problem with vocal tracks in the mix is the wide dynamic range that singers often use, unlike for example an distorted electric guitar which mainly stays on the same loudness level and is therefore easy to mix with great presence. Almost in every mix the vocals path through an compressor to reduce their dynamic range. But this is rarely sufficient as compressors are working comparatively fast. To fast to compensate hole song parts with a different vocal level or even some seconds. For example when a singer is changing his singing style or he sings instinctively quieter during an instrumental break which may not fit the mix. Because of this it is an common procedure to automate a applied gain for every vocal track in the digital audio workstation (DAW) via sketching a gain curve by hand. Obviously this is a time consuming and monotonous task but perfect to hand over to a machine.

1.2 Idea

The idea was to write a DAW plug-in that handles the former described problem. A plugin that will sketch a gain curve for a vocal track in real time. This will save the engineer time at every mixing session and the outcome will probably be more accurate as he could do it.
As the recorded sound pressure level is not transferable to the perceived loudness, the plan was to adapt some algorithmic features of the ITU-R BS.1770-4 [FOOTNOTE pls] “algorithm to measure audio loudness”. This algorithm is an up-to-date standard for adjusting audio files to a same humanly perceived loudness level. This should help the gained audio track to stand out in the mix at every point.
There was published one plug-in with similar purpose called “Vocal Rider” by Waves [fooznotr]. I will build my plug-in for this thesis with features i consider important and

useful for its purpose, while i do not know about the algorithm behind the Waves plug-in. Hence i will compare the resulting audio files after adapting the gain individually through the “Vocal Rider” and my own plug-in. Further i will evaluate the outcome and verify it with a listening test on independent participates.

Chapter 2
Background

In this chapter,

Chapter 3

Approaches

3.1 Overview

The basic approach works in five steps: filter, root mean square (RMS) calculation, gate, gain adaption, delay.
It starts with a low-cut and a high-shelf filter which will be applied sample wise on the incoming audio. Those filters are a simplified mapping of the perception of sound pressure for human beings. In this way (as done in FOOTNTE) it is the first step to adjust the plug-in to loudness instead of sound pressure. Although in the basic approach the plug-in will not be working with the full loudness detection algorithm (FUTNOT) due to calculation time and real time capability (see SPÄTER? oder hier?).
After the filter section every sample is passed to the RMS calculation. This will output the squared average of a previously specified period of time. The determination of the square root is not necessary because it will happen in the following calculation of the equivalent dB value. The plug-in needs to convert the linear audio samples into the logarithmic dB scale because it will display gain values and loudness goal (see gain part?) in dB at the user interface (UI) as it is the standard scale of DAWs. Thereby the executive sound engineer will intuitively know how to interpret and interact with the UI (see design part?).
When dB conversion is done, all the samples path through an initially specified gate. The gate will set all samples with lower dB value as itself to the current loudness goal (see gain, see improve). In this way the plug-in will not operate when it is fed with silence or irrelevant noise.
Next step after the gate is the gain adaption. In this step the gated RMS value is compared to the current loudness goal. Depending on the difference of both values there results an preliminary gain. The gain variations per sample are smoothed comparable to the RMS calculation. This leads to the final gain value.
Lastly the new gain is multiplied with the current sample. Because the plugins behaviour is smoothed as it shall sound natural, it will not react instantly to the input. To compensate the reaction time it delays the input signal before multiplying the calculated gain. This delay is finally offset by the DAW.
During development most of the parameters described in the following sections were settable in a basic dummy UI. This was realised with the standard JUCE slider and button objects and used for tests and fast adjustments.

3.2 Filter

While gathering information on how to improve my idea of the plug-in I got interested in the ITU-R BS.1770-4 [FOOTNOTE pls] algorithm. It classifies an audio file for its humanly perceived loudness. The main use of this algorithm is in television and music streaming services as they can keep the program loudness while switching content. As the human perception is also interesting for mixing a song, I examined how it was done. Because there was a good documentation about how to implement the algorithm I build it in python and decided which elements could be useful for my plug-in. The first element were the filters. Like I described above, their use ist to mimic the human perception of sound pressure at different frequencies. Realised in a greatly simplified but cost effective version with one low cut and one high shelf filter. The low cut filter has a cutoff frequency at 38 Hz, the high shelf around 1681 Hz. They are initialised at every plug-in startup in the JUCE method "prepareToPlay" with the current sample rate of the integrating DAW:

```
lowcut.setCoefficients(38.0, sampleRate, (1.0/2.0));
highshelf.setCoefficientsShelf(1681.0, sampleRate, 4.0);
```

The implementation is based on the biquad filter from the Book BLA (BLA NOTE auch im code). I have chosen the biquat filter architecture because it is a very flexible and simple solution which is real time capable. The calculation of filter coefficients is adopted as follows:

berechnung in Math Latex für beide filter wo dran steht wie sie heißen.

$$a = 2$$

Because the loudness algorithm uses second order filters (with two delay memories) it works like this:

Biquatfilter Bild

The same as my implementation in the Filter class:

```
double AutoVocalCtrlFilter::process(double sample)
{
    const double mid = sample - a1 * z_1 - a2 * z_2;
    const double out = b0 * mid + b1 * z_1 + b2 * z_2;

    z_2 = z_1;
    z_1 = mid;

    return out;
}
```

Chapter 4
Evaluation

The evaluation...

Chapter 5

Results

Chapter 6

Discussion

In the gggg

Chapter 7

Conclusion and Future Work

In this thesis, we yadaaaaa

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Ort, Datum	Unterschrift

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum	Unterschrift