



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



Nils Heine

Real-Time Automatic Gain Control for Singing Voice Applications

University of Hamburg
Department of Informatics

March 20, 2018

1. Motivation
2. Algorithm
3. Optimization
4. Results
5. Future Implementations



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



Signal Processing

Motivation

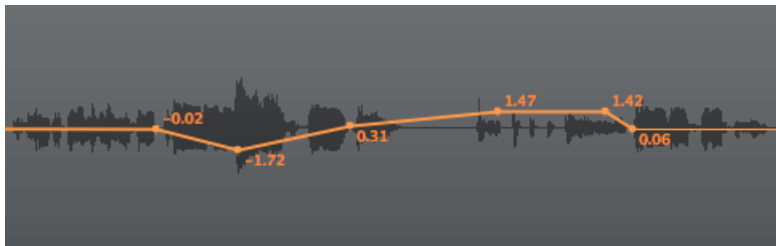




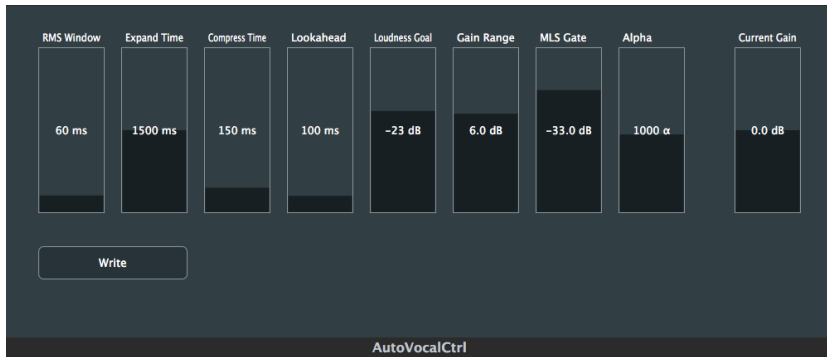
- compressor to fast



- compressor to fast ■ factor in human perception



- compressor to fast
- factor in human perception
- save time





Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



Signal Processing

Algorithm

2. Algorithm

2.1 Filter

2.2 RMS

2.2.1 Time Coefficients

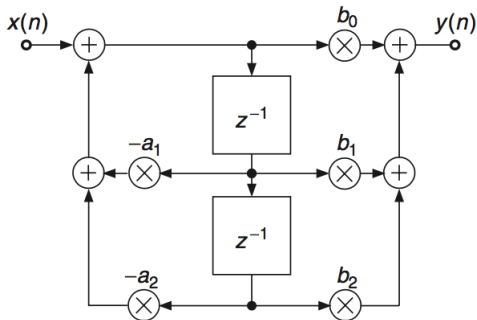
2.3 Gate

2.4 Gain

2.4.1 Loudness Goal Adaption

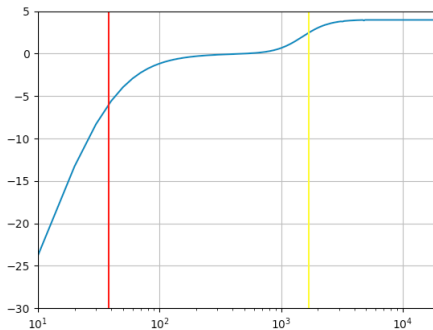
2.4.2 Write Automation

2.5 Lookahead



[1]

[1] Figure from DAFX: Digital Audio Effects by Udo Zolzer.



- lowcut (38 Hz), highshelf (1681 Hz)^[2]

^[2] from Recommendation ITU-R BS.1770-4.

Root Mean Square (RMS):

```
void AutoVocalCtrlAudioProcessor::updateRMS(int channel)
{
    rms[channel] = (1. - rmsCo) * rms[channel] +
                   rmsCo * (filterSample[channel] * filterSample[channel]);
}
```

[3]

Time Constants:

```
float AutoVocalCtrlAudioProcessor::getTimeConstant(float ms)
{
    if (ms > 0.f)
        return 1.f - exp(-2.2*(1./currentSampleRate)/(ms/1000.));
    else
        return 1.f;
}
```

[2]

[3] Based on Book: Digital Audio Signal Processing by Udo Zolzer.



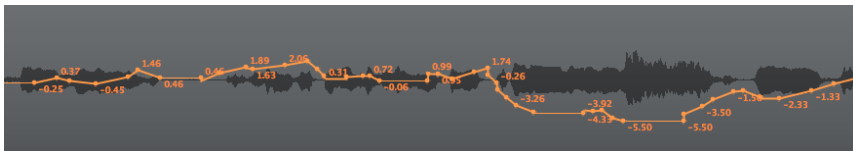
Algorithm: Gain

```
void AutoVocalCtrlAudioProcessor::updateGain(int channel)
{
    const double g = *loudnessGoal - mls[channel];
    const double co = g < gain[channel] ? compressTCo : expandTCo;
    gain[channel] = clipRange.clipValue((1 - co) * gain[channel] + co * g);
    updateAutomation();
    ...
}
```

Algorithm: Gain: Loudness Goal Adaption

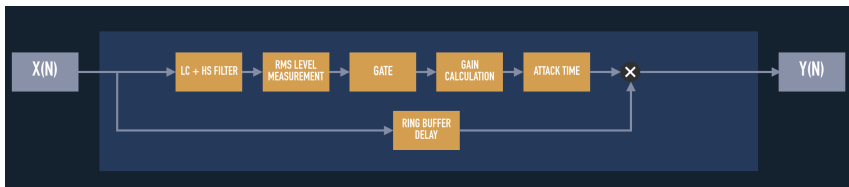
```
void AutoVocalCtrlAudioProcessor::updateGain(int channel)
{
    ...
    alphaGain[channel] = (1 - alphaCo) * alphaGain[channel] + alphaCo * gain[channel];
    if (channel == getTotalNumInputChannels() - 1)
    {
        ...
        updateLoudnessGoal();
    }
}
```


Algorithm: Gain: Write Automation



Algorithm: Lookahead

per ringbuffer





Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

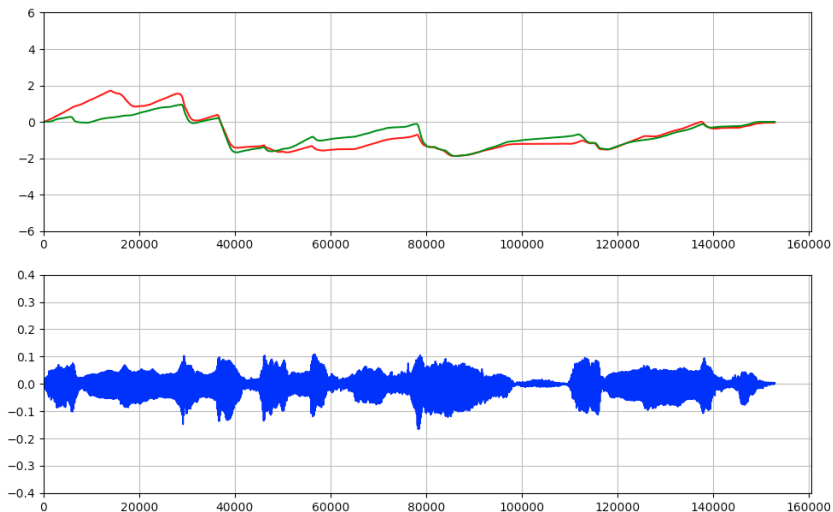


Signal Processing

Optimization

```
...
res = optmze.brute(compareGainCurve, bnds, full_output=True, finish=optmze.fmin)
...
res = optmze.minimize(compareGainCurve, x0, bounds=bnds, options={'disp': True, 'eps': 0.5})
...
x1 = np.array([-28.22, 5.65, 110.14, 2044.41, -33.94, 0.])
...
```

Optimization: Results






Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



Results



■ Original: 

■ Gain Control: 





Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



Signal Processing

Future Implementations

Future Implementations

- side chain backtrack
- offline loudness goal calculation
- set parameters → simplify UI
- improve writing of automation
- idle time?
- wet / dry?

1. Motivation
2. Algorithm
3. Optimization
4. Results
5. Future Implementations