An approach to assess loudness and dynamics with Web Audio native nodes

Sebastian Zimmer
Cologne Center for eHumanities
Albertus-Magnus-Platz, 50923 Köln, Germany
sebastian.zimmer@uni-koeln.de

ABSTRACT

Music content providers on the Internet like YouTube[1], Spotify[2] or Apple Music[3], as well as a range of software playback systems like the media player "foobar2000"[4] have a loudness normalization feature to match a series of diverse audio tracks in overall loudness. This is done to keep perceived volume differences between audio tracks as low as possible. Thus, it is important for music producers, especially mastering engineers, to master audio tracks with a particular amount of dynamic range, so that streaming services will not turn the playback volume of their tracks down. With their already low dynamic range, a listener would now even better be able to recognize their inferior sound compared to other tracks with higher dynamic range.

To correctly assess the dynamics of audio material, this paper introduces two web applications that compute and visualize the loudness and dynamic range of audio material, using a subset of the loudness units described in the recommendation R 128[5] by the European Broadcasting Union[6], and using only native notes by the W3C Web Audio API[7].

Keywords

Web Audio API, Loudness, Dynamic Range, EBU R 128

1. INTRODUCTION

In 2010, the European Broadcasting Union ("EBU") has issued their recommendation R 128, "LOUDNESS NORMALISATION AND PERMITTED MAXIMUM LEVEL OF AUDIO SIGNALS". In this recommendation, particularly the related EBU Tech document 3341 [1], the EBU uses new units to measure loudness that are defined ITU recommendation "ITU-R BS.1770" [2]. Two of these units, LU ("Loudness Unit") and LUFS ("Loudness Units, relative to Full Scale") use defined filters that provide a more accurate representation of human perception and thus making them superior compared to traditional methods to measure loudness like root mean square ("RMS").

The recommendations by EBU and ITU are mainly relevant to broadcast audio, but they are also useful for music producers. In this paper, an approach to assess loudness and dynamics is presented, using new units and only native nodes provided by the Web Audio API. The relatively unperformant ScriptProcessorNode[8] is not used.

In the 1990s and the 2000s, mastering engineers competed in making records louder and louder, desiring to make them more appealing when played back amongst other records without any loudness normalization in place. This so-called "Loudness War" resulted in records with very little dynamic range and, in extreme cases, even in severe distortion of the audio material. However, since more and more streaming services have implemented loudness normalization in the last few years, it has become more important for music producers, to find a sweet spot of dynamic range for their material. Audio material with too little dynamics is viewed as sounding unpleasant and can lead to ear fatigue [3]. Their playback volume is turned down by music services, whereas music with a dynamic range too large will not be turned up enough to "compete" with other tracks.

This paper introduces two web applications, LoudEv and LoudEv Live that provide an intuitive way to assess momentary dynamic range of audio material. To further illustrate this assessment, signal colors and emojis are used.

2. RELATED WORK

The Loudness War and its consequences have been thoroughly analyzed by Earl Vickers [4] and others.

Mastering engineer Ian Shepherd has published an infographic that shows the different loudness normalization techniques of different music content providers[9], and has helped creating a DAW plugin, that introduced the unit peak-to-short-term loudness ("PSR") for the first time[10].

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

© 2017 Copyright held by the owner/author(s).

[1] https://youtube.com
[2] http://spotify.com
[3] http://www.apple.com/de/music/
[4] http://www.foobar2000.org/
[5] https://tech.ebu.ch/loudness
[6] https://www.ebu.ch/home
[7] https://webaudio.github.io/web-audio-api/
[8] https://webaudio.github.io/web-audio-api/#idl-def-ScriptProcessorNode
[9] http://productionadvice.co.uk/online-loudness/
[10] http://www.meterplugs.com/dynameter

Within the last few years, several other loudness meters have been released as DAW plugins that incorporate short-term loudness as well as other new units defined by EBU R 128.

3. IMPLEMENTATION

3.1 Overview

The author of this paper has created two web applications: LoudEv[11] analyses a complete audio file and then provides an overall assessment of the dynamics.

LoudEv Live[12], by contrast, allows analyzing live input signals, either provided by a MediaElementAudioSourceNode[13] or by a MediaStreamAudioSourceNode[14] in combination with the browser function Navigator.getUserMedia()[15].

Both applications use the Skeleton CSS framework. The offline version of LoudEv also incorporates wavesurfer.js, a customizable audio waveform visualization generator, built on top of Web Audio API and HTML5 Canvas[16].

3.2 Measuring short-term loudness

The first thing both LoudEv and LoudEv Live do when analyzing a signal is to compute the short-term loudness, as defined in EBU R 128.

The short-term loudness "uses a sliding rectangular time window of length 3 s" [1]. Each channel of the input signal must be processed separately, thus a ChannelSplitterNode[17] splits the signal into its channels. The first step of the signal-processing algorithm is a two-stage weighting filter, as specified by the ITU. The two stages are realized with two BiquadFilterNodes. After that, each sample of the channel signal is squared, which is done by connecting the audio graph to a GainNode's input, as well as its AudioParam gain (see code sample 1).

```
highpass_filter_L.connect(ebu_square_gain_L);
highpass_filter_L.connect(ebu_square_gain_L.gain);
```

Code sample 1: Squaring a signal

The squared samples are then summed within a 3 second window. This is achieved with a ConvolverNode[18] whose impulse response is a direct current signal of length 3 seconds, of which each sample has the absolute value 1.0.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

© 2017 Copyright held by the owner/author(s).

After that, the signals of each channel are merged again into one, which is done by connecting each channel to one GainNode[19]. Finally, the absolute sample values are obtained with an AnalyserNode[20] and its method getFloatTimeDomainData(). Each sample is multiplied with 10*log10 and then the constant -0.691 is subtracted.

LoudEv Live additionally provides RMS meters for each channel. These values are computed similarly to the short-term-loudness, with the addition that the samples are rooted with a WaveShaperNode[21]. This node uses a value-mapping curve that maps each sample value to its square root (see code sample 2).

```
var curve = new Float32Array(amount);
var slope = 1 / ((amount - 1)/2);

for (var i = 0; i < amount; i++ ) {
    if (i > (amount/2)){
        var sample_value = slope * i - 1;
        var target_value =
Math.sqrt(sample_value);
        curve[i] = target_value;
    } else {
        curve[i] = 0;
    }
}
```

Code sample 2: Creating a wave shaping curve, where each sample value is mapped its square root

3.3 Measuring peak-to-short-term loudness ratio (PSR)

The peak-to-short-term loudness is computed by subtracting the momentary short-term loudness from the maximum absolute value of the last 3 seconds of the signal.

3.4 Visualization and assessment

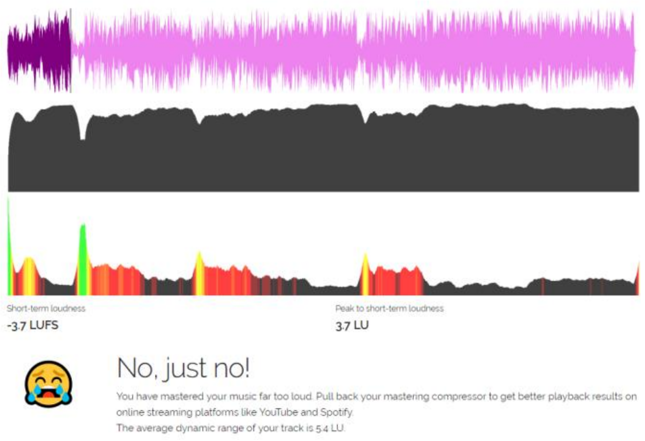
A function renders the current levels of loudness and dynamics for each frame. It is called by window.requestAnimationFrame()[22]. Under ideal conditions, the refresh rate is 60 fps.

PSR values are mapped to colors, where greener colors symbolize a more appropriate dynamic range and red color tones symbolize a dynamic range which is rather low.

PSR values are also mapped to emojis to provide a second intuitive illustration (see code sample 3).

LoudEv's and LoudEv Live's assessments are based on research by the author as well as the recommendation by Ian Shepherd to keep the minimum short-term peak to loudness (PSR) of the track above 8 LU, to make sure, that the track is not turned down too much by music content providers [5].

[11] https://webaudiotech.com/sites/loudev/
[12] https://webaudiotech.com/sites/loudev-live/
[13] https://webaudio.github.io/web-audio-api/#MediaElementAudioSourceNode
[14] https://webaudio.github.io/web-audio-api/#MediaStreamAudioSourceNode
[15] https://developer.mozilla.org/de/docs/Web/API/Navigator/getUserMedia
[16] https://www.w3schools.com/html/html5_canvas.asp
[17] https://webaudio.github.io/web-audio-api/#the-channelsplitternode-interface
[18] https://webaudio.github.io/web-audio-api/#idl-def-ConvolverNode
[19] https://webaudio.github.io/web-audio-api/#idl-def-GainNode
[20] https://webaudio.github.io/web-audio-api/#idl-def-AnalyserNode
[21] https://webaudio.github.io/web-audio-api/#WaveShaperNode
[22] https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame


Short-term loudness

-3.7 LUFS

Peak to short-term loudness

3.7 LU

No, just no!

You have mastered your music far too loud. Pull back your mastering compressor to get better playback results on online streaming platforms like YouTube and Spotify.

The average dynamic range of your track is 5.4 LU.

Figure 3: LoudEv loudness and dynamic visualization and assessment of the song "Around the World" by "Red Hot Chili Peppers", which is known for its low dynamic range.

```
if (psr_value < 5){
        emoji = '😵';
} else if (psr_value < 6){
        emoji = '😭';
} else if (psr_value < 7){
        emoji = '😢';
} else if (psr_value < 7.5){
        emoji = '😟';
} else if (psr_value < 8){
        emoji = '🙁';
} else if (psr_value < 8.5){
        emoji = '😥';
} else if (psr_value < 9.5){
        emoji = '😐';
} else if (psr_value < 11){
        emoji = '🙂';
} else {
        emoji = '😃';
}
```
Code sample 3: PSR value to emoji mapping

3.5 ISSUES

The internal mechanisms of Navigator.getUserMedia() seem to normalize the input signal, which renders the analyzing technique by LoudEv and LoudEv Live useless for streamed input via MediaStreamAudioSourceNode.

4. CONCLUSION

LoudEv and LoudEv Live provide an intuitive and easy-to-use way to assess loudness and dynamics. Because they are web applications, everyone can use them without much friction. As more and more amateurs are producing their own music, LoudEv and LoudEv Live fill the gap of simple, online and free loudness and dynamic assessment tools.

The flexibility of Web Audio API native nodes allows for interesting signal processing techniques, without having to rely on ScriptProcessorNode.

5. REFERENCES

[1] Loudness Metering: 'Ebu Mode' Metering To Supplement EBU R 128 Loudness Normalization. *Geneva 2016, European Broadcasting Union.* https://tech.ebu.ch/docs/tech/tech3341.pdf.

[2] Recommendation ITU-R BS.1770-4 (10/2015): Algorithms to measure audio programme loudness and true-peak audio level. *Geneva 2017, International Telecommunication Union.* https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.1770-4-201510-I!!PDF-E.pdf

[3] Sreedhar, Suhas: The Future of Music. *In: IEEE SPECTRUM, August 1, 2007.* http://spectrum.ieee.org/computing/software/the-future-of-music

[4] Vickers, Earl: The Loudness War: Background, Speculation and Recommendations. *Santa Clara 2010.* http://www.sfxmachine.com/docs/loudnesswar/loudness_war.pdf

[5] Shepherd, Ian: Loudness online – how loud is loud enough, and how loud is too loud ? In *Production Advice, September 28, 2015.* http://productionadvice.co.uk/online-loudness/.