ADAPTIVE AUDIO REPRODUCTION USING PERSONALISED COMPRESSION

ANDREW MASON[1], NICK JILLINGS[2], ZHENG MA[3], JOSHUA D. REISS[3], AND FRANK MELCHIOR[1]

[1] *British Broadcasting Corporation, London, UK*
[2] *Birmingham City University, Birmingham, UK*
[3] *Queen Mary, University of London, London, UK*

Correspondence should be addressed to
andrew.mason@bbc.co.uk

Audio quality is very important to broadcasters' audiences, and unwanted loudness variations do compromise the quality of experience for the listener. Dynamic range control applied by the broadcaster can go some way to avoiding problems but can never take the individual environment of the listener into account. The listening conditions are a significant factor to be taken into account when dynamic range control is applied. The web audio API provided by HTML5 offers the possibility of performing dynamic range control under the control of the listener, tailoring it optimally for their individual situation. We have developed a system that demonstrates that this is achievable in a modern web browser. The implementation controls the compressor based on environmental noise level measured using the microphone present in most mobile device audio players.

INTRODUCTION

Variability in loudness is a major concern in broadcast audio ([1], and references therein), affecting audibility, intelligibility, comfort and overall satisfaction with the programme material and its delivery. This variability is often addressed through the use of dynamic range compression, a nonlinear audio effect that maps the dynamic range of an audio signal to a smaller range [2].

Up to now, attempts have been made to increase audience satisfaction with the dynamic range of programmes. Analogue radio often applies dynamic range compression in the studio and at transmission processing. Whilst this addresses the audibility/comfort problem, it leads to a reduction in fidelity for some.

DAB and DTV both attempt to solve the problem by including a dynamic range control mechanism in which compression control data is sent to the receiver [3]. However, particularly in DAB, not all receivers support that part of the standard. There is no standard for calculating the compression control data, and the systems are under-used.

An alternative approach to broadcasting, gaining support amongst broadcasters because of its simplicity, is to move dynamic range control firmly towards the listener. It is the listener who knows what they want, based on where they are, what they are doing, and what is happening around them. As the broadcaster knows none of this, it makes sense to give control to the listener (or to their content player).

Personalised compression, where the parameters are tuned to the listener and his or her environment, is a relatively new field. However, it has its roots in automatic dynamic range compression research, which has a rich history [4]. Compressors with partly automated parameters (such as 'autorelease') have already found their way to production both as analogue and digital designs [5]. In [6] the time constants were automated by observing the difference between the peak and RMS levels of the signal fed into the side-chain. An RMS measurement was used to scale the release time constant in [7]. The concept of replacing a user-controlled ratio and knee width with an infinite ratio and single, user controlled knee width has been used previously in both analogue and digital compressors, albeit with a static knee width [6]. Similarly, automatic make-up gain can be found in some compressor designs [8], but only as signal-independent static compensations that do not take into account loudness, even though the main purpose of make-up gain is to achieve the same loudness between the uncompressed and compressed signals. More relevant research can be found in [9] where a series of methods to automate most of the parameters of a digital dynamic range compressor based on side-chain feature extraction from the input signal were presented. However, in this system, the threshold was still manually chosen.

In [10] a new class of adaptive digital audio effects that mapped semantic metadata to control parameters was proposed. Automation of dynamic range compression was performed using a simple mapping between different selection of metadata and static compression presets. However, the system assumed that the metadata already exists either from a prior process or manual configuration and might be invoked on demand.

Several techniques have been proposed to perform dynamics processing based on multiple concurrent sound streams. Notably, [11] described an off-line method for automating multi-track compression based on loudness and loudness range. The control strategy was to reduce the difference between the highest and lowest loudness range of the multi-tracks and sound sources where a higher loudness range requires greater amounts of compression. However, evaluation results were inconclusive regarding the sonic improvement of the mixes. [12] proposed an algorithm to automatically adjust the background music levels based on the activity and energy of foreground audio contained in a video file. Such time varying level changes are closely related to dynamic range compression. Though this approach distinguishes between background and foreground audio, in this context, both background and foreground content are played from the same device, and external sounds are not considered.

None of these automation approaches were 'environment aware.' That is, automation of parameters was made solely based on the audio to be compressed, independent of listening level and independent of any additional sounds in the environment. Techniques that dynamically adjust signal level based on the background noise are primarily for automation of volume control, e.g., [13-14], and thus provide coarse changes to the signal which may be easily perceived by the listener.

Some car manufacturers include speed-dependent and background noise-dependent signal processing in the radio output, such that dynamic range compression, equalisation, stereo and surround processing may be adapted to changing vehicle speed and noise levels [15-16]. This significantly improves audio sound quality, but requires multiple sensors and is specific to automotive applications.

Perhaps closest to the work presented herein, at least in spirit, is [17], which described a method for adapting a dynamic range compressor's output gain to account for environmental noise. However, this technique does not describe automation of any compressor parameters. Furthermore, it is specific to speech signals, and no evaluation is described.

The work presented here describes personalised compression that adapts the dynamic range of the audio being played according to the environmental noise around the listener, and offers simple control of the process to the listener. Environmental noise is picked up by the microphone in or attached to, the phone, tablet, laptop, or PC being used, and a graphical user interface provides information and control.

The web audio API was used as the basis of a player implemented in a web browser. Internet delivery of content allows much easier experimentation, and potentially quicker and cheaper deployment of this type of adaptation. The web audio API allowed deployment of new techniques for audio processing without requiring software installation, and with independence of the platform being used.

1 USING THE WEB AUDIO API FOR PERSONALISED REPLAY

Web audio is part of the HTML5 web standard being drafted by the W3C [18]. The web audio API enables audio processing to be done by the browser. This includes scheduling of audio for accurately-timed playback, live controls, feedback to the user, and comparatively easy implementation.

The fundamental structure is that an audio source is connected through a series of processing "nodes" before being passed to the destination. The chain of nodes is called an "audio graph". Sources can be a page object, such as the HTML5 video or audio element, a buffer loaded in JavaScript with sample values, or a local source such as a microphone. The destination is the audio output of the browser (which typically finds its way to the headphones or loudspeakers).

The nodes used by this project are:

- *buffer source* - to provide the programme audio;
- *gain* - to apply gain, but also to provide a simple way to connect and disconnect parts of the chain;
- *compressor* - to apply the single-band dynamics processing (with automated variation of its parameters);
- *script* - to allow building custom audio nodes written in JavaScript.

The *media stream* element is used to access the microphone, which is not part of the web audio standard, but it captures data from the microphone and can stream it into the web audio graph in a similar way to streaming from an HTML5 audio element.

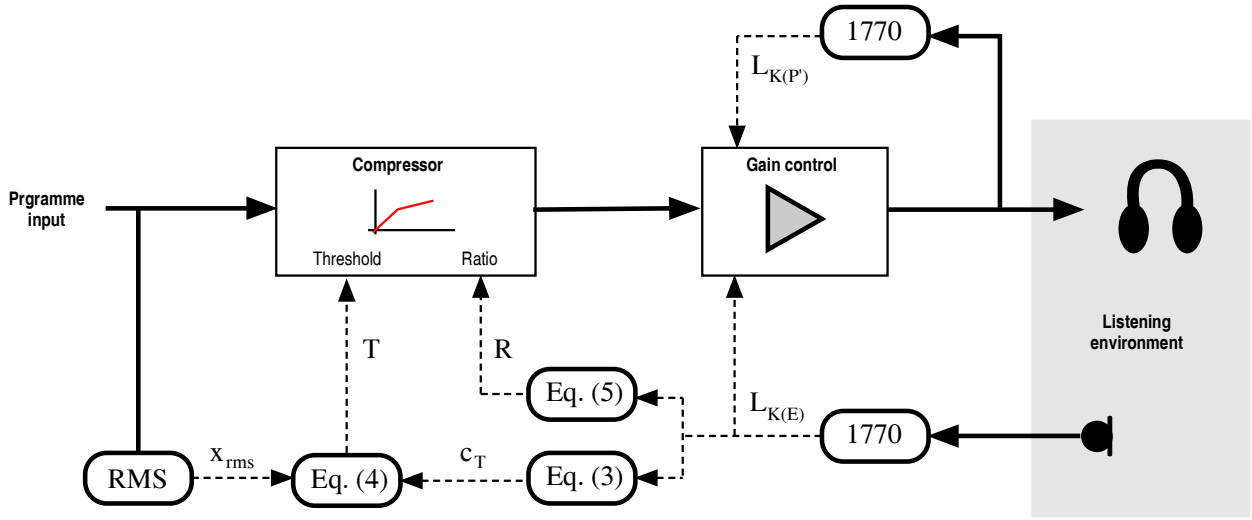

Figure 1 Signal flow of personalised compression system

HTML sliders and buttons provide a mechanism whereby the user can control the system or be given information about its state. The intent of this project is to provide only the sufficient and necessary control, but controls are present at the moment for development purposes.

In essence, the programme and environmental noise (picked up by the microphone) are analysed, and the results used to control the gain and compressor nodes to maintain an adequate programme-to-noise ratio and to reduce the dynamic range as the environmental noise increases. A simplified signal flow graph is shown in Figure 1.

2 AUTOMATIC VOLUME CONTROL

A first level of adaptation is provided by an automatic adjustment of gain applied to the programme sound. The system measures the loudness of the programme sound and of the environmental noise and adjusts the gain applied to the programme sound to maintain a 6 LU (loudness unit) programme-to-noise ratio.

The loudnesses $L_{K(E)}$ of the environment noise, and $L_{K(P')}$ of the programme (after compression) are measured according to equations (1) and (2) of Recommendation ITU-R BS.1770 [19] with a 3 second integration time, as for a "short term" measurement according to Recommendation ITU-R BS.1771-1 [20]. The BS.1770 algorithm applies a K-weighting filter to each input channel, and then calculates a weighted combination of the mean-square of the filtered samples. The loudness meter is implemented in a "script" node.

As a result of trying to minimise computational load, this implementation mixes stereo programme sound to mono before calculating the loudnesses.

The algorithm makes the measurements every block of 4096 samples (at a sampling rate of 48kHz). The gain being applied is updated every 3ms to adapt to changes in measured values. The changes in gain are smoothed using an exponential moving average (EMA) to avoid jumps in response, but not make the adaptation to slow. This EMA filter is given in Equation (1),

$$y[n] = (1-a)x[n] + ay[n-1] \qquad (1)$$

where $x[n]$ is the most recent loudness value $L_K[n]$, and $y[n]$ and $y[n-1]$ are the new smoothed value and the previous smoothed value, respectively. The value of the 'forgetting factor' a is set to 0.998 and to 0.9 for gain increases and decreases, respectively, with corresponding time constants of 1.5s and 20ms.

The maximum gain applied is limited to 10dB, in order to minimise the risk of damaging the hearing of the listener when the environmental noise is very loud.

The current implementation relies on microphone calibration using a white noise source at 65dBA. It is anticipated that this explicit requirement will be engineered out of the system, either by finding reasonable assumptions, or by learning from the listener's use of any controls provided.

To adapt the automatic gain control further the listener may indicate that they are using a particular style of headphone, with a corresponding typical attenuation of environmental noise. Measurements made on a small

selection of headphones suggest that attenuation of 10dB might be expected for circum-aural closed-back headphones, about 8dB for supra-aural closed-back ones, and less than 1dB for open-backed ones. Again, manual intervention by the listener might be engineered out, for example in future generation of devices which potentially will automatically detect the type of headphone being used.

3 AUTOMATIC DYNAMIC RANGE COMPRESSION

The web audio API provides a dynamics compression node with threshold, ratio, knee width, attack time, and release time properties [18].

Threshold defines the level above which compression starts. Signals overshooting the threshold will be reduced in level. Ratio controls the amount of compression applied. It defines a drop in level above the threshold. For example, if a signal is above the threshold by 10dB, with a ratio of 2:1, the signal will be attenuated by 5dB.

Figure 2 shows the basic transfer characteristic of the compressor.

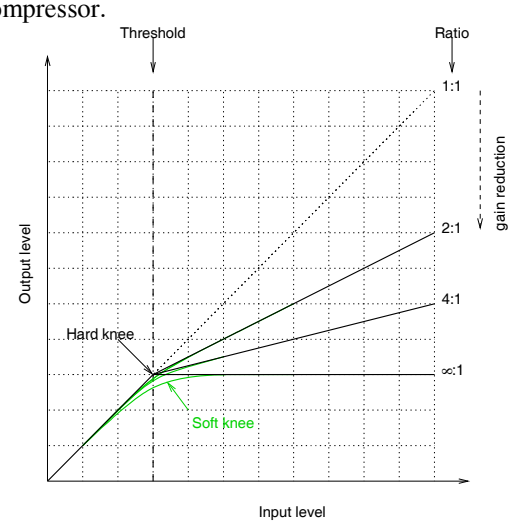

Figure 2: General form of compressor transfer function

Note that in this application, only downward compression is used.

The *knee width* property controls whether the bend in the transfer characteristic has a sharp angle or is rounded. A sharp transition is called a “hard knee”, and a smooth one, where the ratio gradually grows from 1:1 to a final value over a transition region spanning both sides of the threshold, is called a “soft knee”. Softening the knee reduces the production of audible artefacts. The soft knee is shown in green/grey in Figure 2.

The attack and release times define how long it takes for the compressor to change its gain by 10dB towards the level determined by the ratio when the signal exceeds the threshold, and back again when it has stopped doing so.

Some of the properties of the compressor were set to optimal values that have been defined by informal listening, and some are changed automatically as the programme sound and environment noise change. Those that are fixed are as follows:

- *knee width* = 15dB .. a soft knee allowing smooth transition at the threshold level
- *attack time* = 8ms .. very short attack time to catch the transients in the audio signal
- *release time* = 80ms .. moderate release time to give a smooth compression recovery

The compressor's threshold and ratio are continuously adjusted to adapt to changing programmes and environments.

3.1 Compressor threshold automation

Previous research on "intelligent compression" suggests that the RMS value of the programme sound (in dBFS) could be used as a good starting point for automating the threshold [9]. In this project, a simple RMS calculation is performed on blocks of 4096 samples, at a sampling rate of 48kHz, on a mono down-mix (L+R)/2 of the programme audio, as shown in Equation (2),

$$x_{rms} = \sqrt{\frac{\sum_T \left(\frac{x_l(t)+x_r(t)}{2}\right)^2}{4096}} \qquad (2)$$

where $x_l(t)$ and $x_r(t)$ are the left and right channel sample values at time t, respectively, and T is the time interval of the 4096 samples.

Due to the short block-based processing in the algorithm, an efficient and reliable long-term averaging process is needed to produce smoothly varying data, removing rapid changes that would lead to artefacts being introduced. An EMA filter according to equation (1) is used to smooth the x_{rms} values, with a = 0.98 (a time-constant of approximately 0.8s).

Listening in an environment with a high level of environmental noise requires more compression to make the quieter parts of the audio audible whilst not making the louder parts too loud. When the environmental noise level is very low less compression is needed, and therefore listeners can enjoy a wider dynamic range.

This implies that the compressor threshold should be lower than the RMS value when environment noise is high, and vice versa. Based on this, the threshold value is adapted by weighting the RMS programme sound with a value that is a function of the environment noise level. The threshold weighting factor, c_T is an altered Gaussian function of the environment noise level, as shown in Equation 3.

$$c_T = e^{-\frac{\left(\frac{(L_{K(E)}(t)}{60}-b\right)^2}{2\cdot c^2}} \quad \text{for } L_{K(E)}(t) \leq 60dB\ (SPL)$$

$$c_T = -e^{-\frac{\left(\frac{(L_{K(E)}(t)}{60}-b\right)^2}{2\cdot c^2}} \quad \text{for } L_{K(E)}(t) > 60dB\ (SPL) \qquad (3)$$

The ideal shape of the function was determined by informal listening and is shown in Figure 3, where *b* has a value of 1 and *c* a value of 0.7.

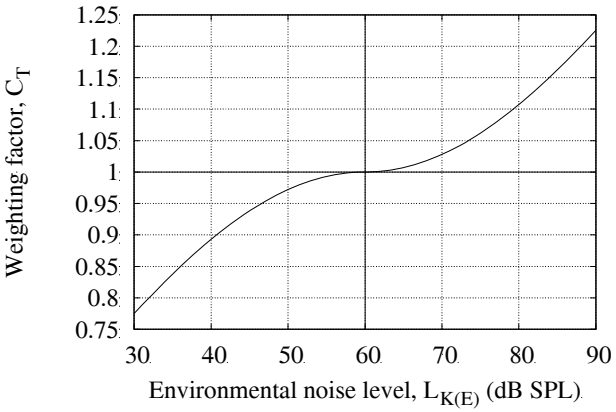

Figure 3: Weighting function applied to compressor threshold

The threshold is weighted as shown in Equation 4.

$$T(t) = x_{rms(dB)} \cdot C_T \qquad (4)$$

where $x_{rms(dB)}$ is the RMS value from Equation 2 converted to dBFS.

The result of applying this weighting is that the threshold is slightly lower than the RMS when the environment noise is higher than 60 dB (SPL), and slightly larger when it is less than 60 dB (SPL). When, for example, the RMS value of the programme sound is -25dBFS, the threshold varies as a function of environment noise level as shown in Figure 4.

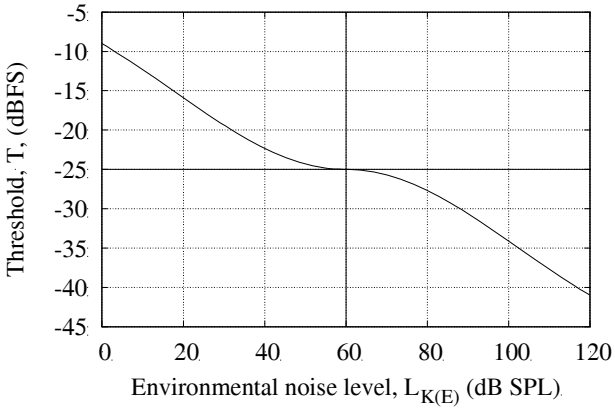

Figure 4: Compressor threshold as a function of environment noise level

In the real world, the environment noise typically ranges from 30dB(SPL) to 90dB (SPL), being representative of a quiet room and traffic on a busy road. As shown in Figure 4, within that range, the threshold value is set close to the RMS. The purpose of the Gaussian curve is so that the threshold varies slowly around the RMS value within the anticipated environmental noise level range.

When compression is being applied with a time-varying threshold, intensive variation of the threshold in a short time causes audible artefacts, so another EMA smoothing, with *a*=0.95, is used prior to the actual setting of the compressor threshold. This is done every 3ms, so the time constant is approximately 60ms.

3.2 Compressor ratio automation

The adaptation of the ratio is similar to that of the threshold, but based only on the environment noise level. In general, higher environment noise level demands a higher ratio. No compression is applied when the noise level is less than 30dB(SPL), and the compression increases monotonically, but non-linearly, in a way that matches human perception of the compression effect. Here, the ratio, *R*, is calculated as shown in Equation 5,

$$R = c \cdot (L_{K(E)} - 30)^2 + 1 \qquad (5)$$

where *c* has been chosen through informal listening experiments to be 0.003265. This gives the curve shown in Figure 5.

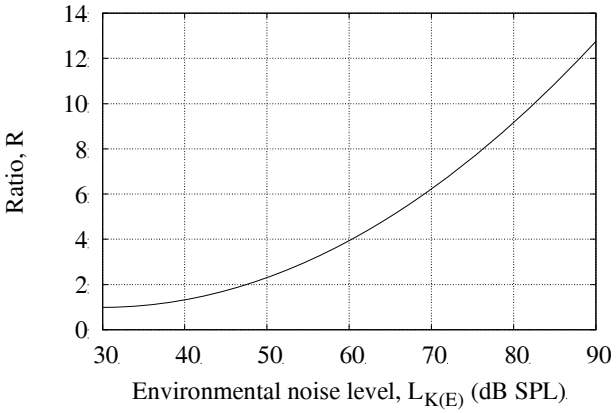

Figure 5: Compressor ratio as a function of environmental noise level

As with the other parameters, an EMA filter, with a=0.95 (a time-constant of 60ms), is used prior to setting the compressor ratio. Although very large values of ratio would not normally be used, no limit is applied. Above a ratio of 10:1, the effect is that of a limiter, and, furthermore, the physiological effects of dangerously high environmental noise levels might become a problem before one needs to worry about the audible effects of extremely large values of ratio.

4 USER INTERFACE FOR EVALUATION

The experimental user interface developed to demonstrate the personalised compressor is shown in Figure 6. It was designed with as few controls as are required to operate it. In the figure, optional controls that are available, but not required, are displayed to the right of the main panel.

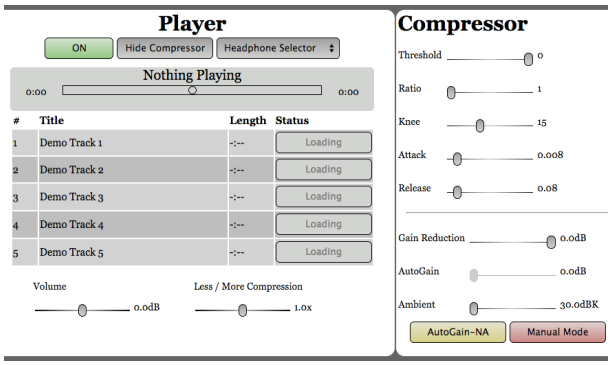

Figure 2: User interface of the development version of the personalised compressor

The interface was designed with both "traditional" computers and touch-sensitive devices in mind because consistency across devices is important for the user's experience. It is styled using CSS and animated using a mixture of JavaScript and JQuery. JavaScript enables the calling of functions at set intervals, and this is used to update on-page elements with information at regular intervals. Crucially it is also used to calculate the compressor values automatically.

The first task of the user interface is to perform the microphone calibration, which it does using a pop-up window telling the user what to do. Following that, the only controls offered are a volume control and a "Less/More" control.

The volume control has a -20dB to +20dB range, preventing complete muting or too much boost.

The "Less/More" control abstracts the multiplicity of controls of the compressor and presents them as one user control: less compression or more compression. Its value, in the range of 0 to 1, is used as a multiplier to scale the values of ratio and threshold.

As stated, the default controls are very minimal, but additional indicators and controls can be shown. Manual control can be taken of the compressor parameters, and they can be varied at will, but this usually only makes things worse.

5 EVALUATION

Feedback from listeners in an informal listening test conducted in the lab using open-backed headphones, a set of test programme material, and environmental noise from the BBC sound effects library played back over loudspeakers, suggested that the system was working quite well already: listeners reported that the system was doing very much what they wanted, and that its operation was unobtrusive.

The choice of operating parameters appeared to have been made well, and listeners sometimes did not realise just what the processing had been doing until it was turned off. A few comments about excessive compression being apparent on one of the items could be addressed by simple adjustment of the "More/Less" slider.

6 CONCLUSIONS AND FUTURE WORK

This paper has described a demonstration system that has shown that personalised dynamic range control can easily be done in a web browser, responding to the environment around the listener. Demonstrations to listeners showed that the processing was unobtrusive and very effective at adapting to changes in environment noise. Future work will include the formal evaluation of the system.

7 ACKNOWLEDGMENTS

This work has been carried out as part of the BBC Audio Research Partnership. The authors would like to thank Chris Lowis (formerly of the BBC) for his invaluable guidance during the development of the personalised compressor. The initial stages of the work were made possible by QMUL's Innovation Fund.

REFERENCES

[1] E. Skovenborg and T. Lund, "Loudness Descriptors and Wide Loudness-Range," 127th AES Convention, New York, 2009.

[2] D. Giannoulis, et al., "Digital Dynamic Range Compressor Design - A Tutorial and Analysis," J. Audio Eng. Soc., vol. 60, June 2012.

[3] W. Hoeg, et al., "Dynamic Range Control (DRC) and Music/Speech Control (MSC)," EBU Technical Review Autumn 1994.

[4] L. Tyler, "An above threshold compressor with one control," 63rd AES Convention, May 1979.

[5] U. Zolzer, Digital Audio Signal Processing, 2nd ed.: John Wiley and Sons, Ltd., 2008.

[6] A. T. Schneider and J. V. Hanson, "An adaptive dynamic range controller for digital audio," IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, Canada, 1991, p. 339 - 342.

[7] G. W. McNally, "Dynamic Range Control of Digital Audio Signals," J. Audio Eng. Soc., v. 32, pp. 316-327, May 1984.

[8] R. Izhaki, Mixing Audio: Concepts, Practices and Tools: Focal Press, 2008.

[9] D. Giannoulis, et al., "Automating a Dynamic Range Compressor," J. Audio Eng. Soc., v. 61, 2013.

[10] T. Wilmering, et al., "High level semantic metadata for the control of multitrack adaptive audio effects," 133rd AES Convention, San Francisco, 2012.

[11] J. Maddams, et al., "An autonomous method for multi-track dynamic range compression," Digital Audio Effects (DAFx), York, U.K., 2012.

[12] J. A. Kang, et al., "A smart background music mixing algorithm for portable digital imaging devices," IEEE Trans. Speech and Audio Processing, v. 57, pp. 1258-1263, Aug. 2011.

[13] M. Urup, "Communication Device with Motion Dependent Auto Volume Control," Patent US20140140539 A1, 2014.

[14] F. J. O. De Beek and J. W. Kemna, "Noise-dependent volume control having a reduced sensitivity to speech signals," US4677389 A, 1987.

[15] M. Christoph, "Speed dependent equalizing control system," US 20120308036 A1, 2012.

[16] J. Kontro, et al., "Digital car audio system," IEEE Trans. Consumer Electronics, v. 39, Aug. 1993.

[17] A. Erell, "Enhancing the intelligibility of received speech in a noisy environment," US 8407045 B2, 2013.

[18] W3C. (2012), Web Audio API. Available: http://www.w3.org/TR/webaudio/

[19] ITU, "ITU-R Recommendation BS.1770-3, Algorithms to measure audio programme loudness and true-peak audio level," 2012.

[20] ITU, "ITU-R Recommendation BS.1771-1, Requirements for loudness and true-peak indicating meters", 2012