FIFTY SHADES OF COMPRESSION

Diemo SCHWARZ[1], **Rémi MIGNOT**[1], **Côme PELADEAU**[2], **Haoran SUN**[3], **Étienne DEMOULIN**[1], **Geoffroy PEETERS**[2], **Hichem MAAREF**[3], and **Dominique FOURER**[3]

[1]**Ircam STMS**, Paris, France
[2]*LTCI*, **Télécom-Paris, IP-Paris**, Paris, France
[3]*IBISC*, **Univ. Evry Paris-Saclay**, Evry, France

ABSTRACT

Audio Dynamic Range Compression (DRC) is one of the most widely used audio effects for controlling the dynamic range of an audio signal and shaping perceived audio quality. In the field of music production, particularly during mixing and mastering, efforts have been made to standardise engineering practices. However, many plugins have idiosyncratic parameters and implementations, and parameter choices remain largely empirical and depend on the individual experience and auditory perception of sound engineers. This work aims to fill this gap by introducing a proof-of-concept application that enables users to explore different sets of DRC profiles. These profiles are projected into a feature space designed to capture the perceptually relevant characteristics of the output audio. By navigating this space, users can compare plugins and compression settings based on features related to perceived audio quality. We believe that this contribution can support sound engineers in selecting appropriate DRC parameters before applying compression to an audio mixture. In addition, the tool may serve pedagogical purposes by helping students and practitioners understand best practices in music production and dynamic range processing.

1. INTRODUCTION

Dynamic range compression (DRC) is a ubiquitous audio effect in music production that is applied to a large proportion of source tracks during mixing, as well as to the final stereo mix during mastering [1, 2]. Since the introduction of hardware compressors in the 1930s, a variety of technologies have been developed to achieve dynamic range control, including tube, optical, and transistor-based circuits [3]. These different designs exhibit distinctive nonlinear behaviours and time responses that shape the perceived sound character of the processed signal. Many of these classic devices are now emulated in modern digital audio plugins, leading to an ever-growing ecosystem of software compressors and limiters available to sound engineers.

Despite the widespread use of DRC, selecting the appropriate compressor settings remains a challenging and largely empirical task. In practice, engineers rely on their listening experience and subjective judgment to tune parameters such as threshold, ratio, attack, and release. As the number of available plugins and parameter configurations continues to increase, it becomes increasingly difficult to systematically compare their sonic characteristics and understand the perceptual impact of different compression strategies.

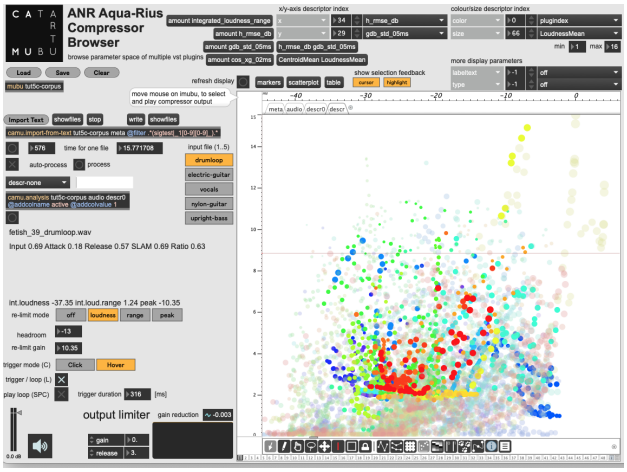


Figure 1. Example of the navigation interface in MAX showing a corpus of compressor outputs according to spectral centroid and loudness.

Several studies have investigated the perceptual and signal-processing aspects of dynamic range compression. In particular, the studies proposed in [4,5] explore the characterisation of dynamic range compression through audio descriptors and analysis of the resulting signal transformations. Such approaches highlight the potential of feature-based representations to describe the perceptual effects of compression processes.

Pursuing this work on DRC, we establish a collection of free DRC plugins, and systematically sample their parameter spaces. For each parameter configuration, the compressor is applied to a set of test audio signals. The resulting processed signals are then analysed using perceptually relevant audio descriptors that capture both general sonic characteristics (e.g., spectral centroid reflecting changes in frequency distribution) and dynamic attributes (e.g., output dynamic range).

To explore the variations in sonic character produced by different compression settings, we propose an interactive navigation approach based on a two-dimensional projection of the descriptor space (Fig. 1). This paradigm is in-

Copyright: © 2026. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

spired by corpus-based concatenative synthesis (CBCS), which shows its efficiency for exploring large sound collections through perceptually meaningful audio features [6]. By projecting compressor configurations into such a feature space, users can browse and compare different compression profiles and plugins according to their resulting audio characteristics.

As a proof of concept, we present an implementation in MAX that allows interactive 2D browsing of the sonic characteristics associated with different parameter sets across a collection of DRC plugins. [1] This interface enables intuitive exploration of compression behaviours and may support both professional sound engineering workflows and pedagogical applications in music production. This idea was previously applied to the interactive descriptor-based browsing of collections of room impulse responses (RIRs) for convolution reverbs [7], and is here adapted to browsing the outputs of DRC plugins.

The paper is organised as follows. After an overview of the background of this work in section 2, where we describe in detail a conventional DRC algorithm, section 3 introduces the dataset, including the proposed audio features, and numerical results. Practical implementation considerations are presented in section 4. Our contributions, including future work and extensions, are finally discussed in section 6.

2. BACKGROUND

Although commercial dynamic range compressors rely on diverse implementations, they generally expose a common set of user-adjustable parameters. In this paper, we use the conventional feed-forward DRC model as a generic mathematical description of these controls, without implying that commercial plugins exactly follow this model.

2.1 Dynamic Range Compression in a Nutshell

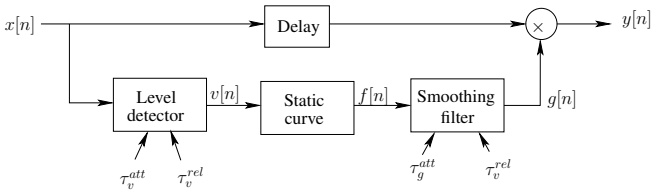


Figure 2. Conventional dynamic range compressor model.

Dynamic range compression is a nonlinear audio effect that modifies the level of a one-dimensional discrete-time real-valued signal x sampled at frequency F_s by applying a time-varying gain g such that:

$$y[n] = x[n]\, g_{x,\theta}[n], \tag{1}$$

where θ gathers the level detector, static curve, and smoothing parameters. For the sake of clarity, $g_{x,\theta}$ is expressed g in the remainder of this paper. For the conventional feed-forward DRC model [8] in Fig. 2:

$$\theta = \{L, R, \tau_v^{\text{att}}, \tau_v^{\text{rel}}, \tau_g^{\text{att}}, \tau_g^{\text{rel}}, p\},$$

with threshold L, static ratio $R = S_{\text{in}}/S_{\text{out}}$, detector attack/release times $(\tau_v^{\text{att}}, \tau_v^{\text{rel}})$, gain attack/release times $(\tau_g^{\text{att}}, \tau_g^{\text{rel}})$, and detector type p (peak if $p = 1$, RMS if $p = 2$). The computation of $g_{x,\theta}$ proceeds as follows:

1. Level detector The detection envelope $v[n]$ is obtained from the input signal $x[n]$ using the parameters p, τ_v^{att} and τ_v^{rel}:

$$v[n] = \sqrt[p]{\beta|x[n]|^p + \bar{\beta} v[n-1]^p}, \quad \text{with } \bar{\beta} = 1 - \beta, \tag{2}$$

where $\beta = 1 - e^{-2.2/(F_s \tau_v)}$, and

$$\tau_v = \begin{cases} \tau_v^{\text{att}} & \text{if } |x[n]| > v[n-1], \\ \tau_v^{\text{rel}} & \text{otherwise.} \end{cases}.$$

2. Static curve Let $l = 10^{L/20}$ denote the threshold in linear scale. The static gain factor is defined as

$$f[n] = \begin{cases} \left(\dfrac{l}{v[n]}\right)^{1-\frac{1}{R}}, & R > 1,\ v[n] > l, \\ \left(\dfrac{v[n]}{l}\right)^{\frac{1}{R}-1}, & 0 < R < 1,\ v[n] < l, \\ 1, & \text{otherwise,} \end{cases} \tag{3}$$

where $R > 1$ corresponds to downward compression and $0 < R < 1$ to downward expansion. The limiting cases $R \to \infty$ and $R \to 0^+$ yield an ideal limiter and an ideal noise gate, respectively.

3. Smoothing filter Finally, the DRC gain $g[n]$ is obtained by smoothing $f[n]$:

$$g[n] = \gamma f[n] + (1-\gamma) g[n-1], \quad \gamma = 1 - e^{-2.2/(F_s \tau_g)}, \tag{4}$$

with:

$$\tau_g = \begin{cases} \tau_g^{\text{att}} & \text{if } f[n] > g[n-1], \\ \tau_g^{\text{rel}} & \text{otherwise.} \end{cases}$$

The processed signal y is finally obtained from Eq. (1). Corresponding software implementations can be found in popular audio tools such as SoX [2].

2.2 Blind DRC Parameter Identification and Effect Cancellation

Blind DRC parameter identification aims to estimate the the DRC parameters from its output signal in order to restore the original dynamic range. Applications include audio mix reverse engineering, remixing, and audio restoration. Recent approaches combine deep neural networks for DRC parameter estimation with model-based DRC inversion [9, 10], using modified versions of the Audio Spectrogram Transformer [11] and the Music Effect Encoder [12]. A Python implementation of the inversion method described in [10] is publicly available. [3]

[1] A demo video is available at https://www.youtube.com/playlist?list=PLt5gV5YpSJ0z1t3bllYfvGUTR1eDpiLGG.

[2] SoX Sound eXchange: https://sox.sourceforge.net

[3] https://github.com/SunHaoRanCN/Inversion-de-la-Compression-Dynamique

Plugin Name	Num Params	Selected Params
molotok [4]	5 of 21	**Thresh.** (inv), Attack, Release, Ratio, Knee
kotelnikov [5]	7 of 22	**Threshold** (inv), Peak Crest, Soft Knee, Ratio, Attack, Release Peak, Release RMS
mcompressor [6]	7 of 22	**Threshold** (inv), Attack, Release, RMS length, Ratio, Knee mode, Knee size
bxmasterdeskc [7]	6 of 12	**Volume**, Foundation, THD, Tone Adjust, Tone Mode, TMT Compressor Mode
dc1a3 [8]	5 of 10	**Input**, Deep, Relaxed, DualMono, Negative
mjucjr [9]	2 of 6	**Compress**, Timing
edcomp [10]	1 of 5	**Amount**
attractor [10]	8 of 14	**Release Compression**, Attack Compression, Attack Low, Attack High, Attack Saturation, Release Low, Release High, Release Saturation
fetcb [10]	4 of 11	**Threshold** (inv), Attack, Release, Ratio
kolin [10]	3 of 11	**Input**, Attack, Release
fetish [10]	5 of 17	**Input**, Attack, Release, SLAM, Ratio
comper [10]	15 of 22	**Threshold Int** (inv), Threshold Int/Ext, Ratio Int, Ratio Int/Ext, Attack Int, Attack Int/Ext, Release Int, Release Int/Ext, Input Int, Input Int/Ext, VCA Int, FET Int, Opto Int, VCA Int/Ext, FET Int/Ext
dbcomp [10]	2 of 10	**Threshold** (inv), Compression
busterse [10]	6 of 20	**Threshold** (inv), Attack Time, Ratio, Release Time, Turbo, XFormer
britpressor [10]	3 of 16	**Compressor Threshold** (inv), Ratio, Compressor Recovery Time
yala [10]	3 of 10	**Volume**, HPF, Release
lala [10]	2 of 13	**Peak Reduction** (inv), HF
oss [10]	5 of 16	**Input**, Attack, Release, Ratio, Interstage
varimoon [10]	4 of 12	**Input**, AC Threshold, DC Threshold, Release
pressure5 [11]	4 of 6	**Pressre**, Speed, Mewines, PawClaw
w1limiter [12]	3 of 5	**Threshold** (inv), Release, A.Release

Table 1. List of VST DRC plugins in our collection and the selected parameters, the first of which was identified as the main compression amount parameter. Parameters with an inverted influence on compression amount are marked "(inv)".

2.3 Perceptual Timbre Spaces and Corpus-based Synthesis

The concept of timbre space was introduced by Grey, Wessel, and McAdams [13–15] in psychoacoustics research as a model of human perception of timbre, or *sound colour*, where relative distances between sounds in a low-dimensional space correspond to relative dissimilarity of timbre perception. Wessel already envisioned this representation as a control interface for musical composition by "taking a walk in timbre space" [14]. This concept was realised in the work on corpus-based concatenative synthesis [16, 17], using navigation through a (usually 2D-projected) vector space of audio descriptors as the main interface for playing and composing music [18].

3. DATASET

In order to compare DRC plugins and their parameters, we constituted a publicly available dataset [13] of 5 input sound files of varying character, 21 DRC plugins, with a regular sampling of their parameter spaces, and the 9,975 output files of the plugins.

[4] https://www.tokyodawn.net/tdr-molotok
[5] https://www.tokyodawn.net/tdr-kotelnikov
[6] https://www.meldaproduction.com/MCompressor
[7] https://www.plugin-alliance.com/en/products/bx_masterdesk_classic.html
[8] https://klanghelm.com/contents/products/DC1A
[9] https://klanghelm.com/contents/products/MJUCjr.html
[10] https://www.analogobsession.com/dynamicprocessing
[11] https://www.airwindows.com/pressure5
[12] https://www.yohng.com/software/w1limit
[13] https://zenodo.org/records/21383068

3.1 Plugin Collection

Our collection of 21 freely downloadable DRC VST plugins (mostly single-band compressors and some limiters) is listed in Table 1. For each plugin, we identified the core parameters relevant to the compressor (typically compression threshold and ratio, attack and release time, knee) and sound character, removing parameters related to the side-chain input. We also removed the output gain or wet/dry mix parameters, if present, because they apply linear gain changes that are at the discretion of the mix engineer (and would be cancelled by the normalisation of the output file for the dataset anyway). We then identify the parameter that determines the main unified compression *amount*. This is often the compression threshold, but some plugins already offer a *volume* or *amount* parameter. However, there are two different types of this main parameter leading to opposite semantics: If it is the compression threshold, the lower the parameter, the stronger is the compression. If the parameter is of type *volume*, *input* (gain) or *compression amount*, higher parameter values lead to stronger compression. Therefore, we inverted the axis for the main parameter when its name matches "threshold" or abbreviations thereof.

We included some plugins' ideosyncratic parameters regarding the simulation of the electronic components, such as switching between tubes, opto-couplers, and solid state. Moreover, the two mastering limiters/compressors in the collection offer additional parameters like *tone* or *total harmonic distortion (THD)* to influence the tonal balance and cleanness. See Fig. 3 for the distribution of the number of selected parameters.

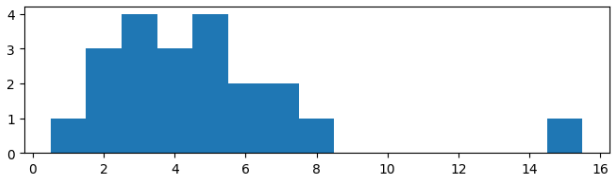


Figure 3. Histogram of the number of selected parameters of the DRC plugin collection.

3.2 Parameter Space Sampling

We performed a systematic sampling of the resulting parameter space of each plugin using Latin hypercube sampling (LHS), which creates irregular but evenly distributed grids across all dimensions for 64 parameter set samples. This sampling method allows us to generate combinations with regular density across all dimensions of the parameter space, without resorting to computationally intractable dense regular grids of hypercubes.

We generated a second set of parameter samples where only the main unified amount parameter is sampled at 31 regular steps while all other parameters are left at their default value. This set is used for most of the figures showing the behaviour of the plugins.

3.3 Input Files and Processing

We chose 5 example input audio clips of diverse character with durations around 3 s, shown in Table 2. These were peak-normalised and used as input to be processed in batch mode by the 21 compressors, under each of the $64+31$ parameter sets. Each output signal's peak loudness is measured and recorded in the dataset metadata before the signal is normalised and written to disk in 24 bit FLAC-compressed files in order to save storage space. The process results in $6720+3255=9975$ output audio files. The whole dataset takes up 3.8 GB. There is also a variant of the dataset with uncompressed and non-normalised 32 bit float files, taking up 8.5 GB.

Fig. 4 shows the sample histograms of three of the input files and the outputs of all 21 compressor plugins and 95 parameter sets. Following Tardieu et al. [4], we calculate the histograms on the absolute sample values converted to dB, not on RMS values, in order to precisely detect the peak-maximizing effect of compression. Note that, depending on the polarity of the main compression parameter (amount or threshold), and the absence of make-up gain, the signal can be reduced to very low values.

	Source File	d [s]	l [LUFS]	r [LU]
●	drumloop	3.5	-21.1	2.0
●	electric-guitar	7.4	-19.1	4.6
●	male-backing-vocals	4.4	-14.1	2.5
●	nylon-guitar	4.0	-16.4	3.4
●	upright-bass	4.9	-15.0	4.8

Table 2. Source input files with their colour code, duration d, integrated loudness l and range r.

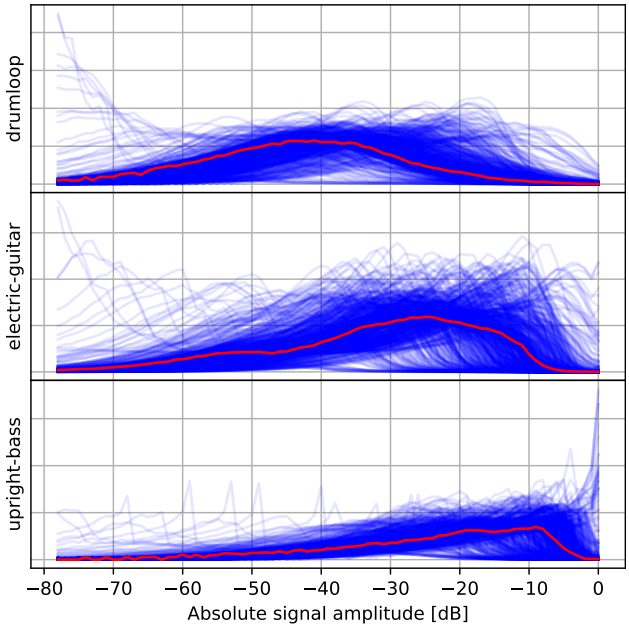


Figure 4. Histogram of the absolute sample values in dB of the input signal $x[n]$ (red) and the outputs $y[n]$ (blue) of all compressor plugins and parameter sets and three of the input files.

3.4 Audio Descriptor Analysis

The input files and the resulting processed output files are then analysed using perceptual audio descriptors that capture both general sonic characteristics (e.g., spectral moments reflecting frequency distribution), audio quality attributes (distortion) and dynamics attributes (e.g., integrated loudness, output dynamic range). Some descriptors are calculated using the original audio as reference, some only on the output (blind descriptors).

3.4.1 Perceptual Audio Descriptors

We calculate the following 8 descriptors [19]: fundamental frequency (pitch), periodicity (corresponding to the continuum between tonal and noisy sounds), and first order autocorrelation coefficient (expressing spectral tilt), estimated by the time-based *yin* algorithm [20]; the loudness, ITU-R 468 weighted to respect the frequency-specific sensitivity of the human ear, and the spectral descriptors centroid (the center of gravity of the spectrum, corresponding to the perceived brightness of a sound), spread, skewness, kurtosis (describing the overall shape and distribution of the spectrum). These time-varying instantaneous descriptors calculated at a period of 6 ms are condensed to summarise each of the audio files by calculating the mean value of each descriptor over each file.

3.4.2 Dynamics Attributes

We follow the ITU-R BS.1770 recommendation for measuring the perceived loudness of audio signals, widely adopted in broadcasting and implemented in the Python package *pyloudnorm* [21] that enables the measurement of integrated loudness l in absolute loudness units LUFS and integrated loudness range r in relative LU.

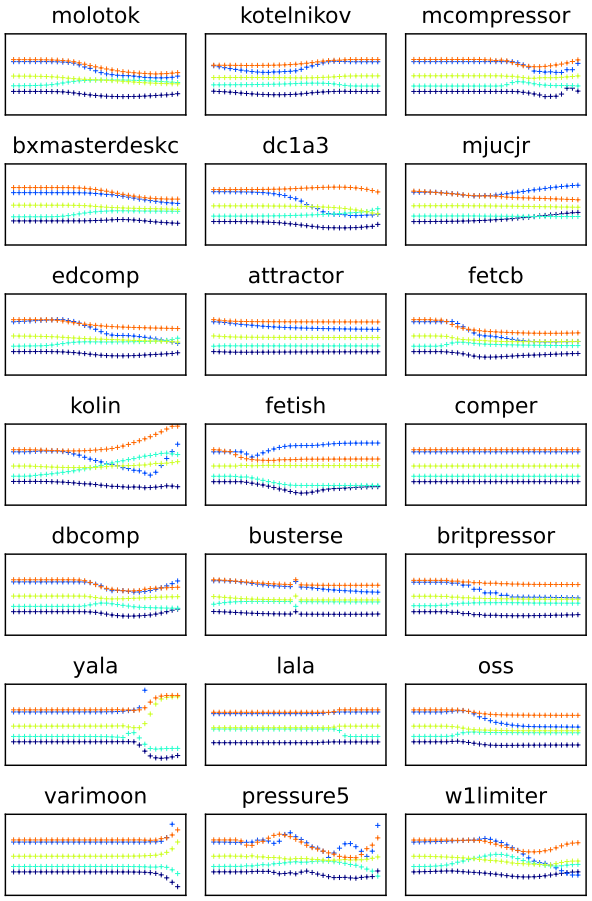


Figure 5. Integrated loudness range r [0–7 LU] vs. unified compression *amount* over a linear sweep.

For each input and output file, l and r are measured, as well as the loudness and loudness range increase δ_l and δ_r, i.e. the difference of these values with the values of their respective input files:

$$\delta_l = l_{out} - l_{in} \quad (5)$$

$$\delta_r = r_{out} - r_{in} \quad (6)$$

The integrated loudness range is shown in Fig. 5 for all plugins. We can see that the mappings between the compression amount parameter and its effect on loudness range are highly diverse between the plugins and also depend on the loudness and spectral content of the input. Further loudness measures are shown in Fig. 6 for a subset of the plugins (the full set is available in the supplementary material[13]). Fig. 6 also shows that the analog technology emulations of some plugins (*yala* and *varimoon*[10]) produce extreme output peak values of up to $80\,\mathrm{dB}$ when the parameters are at their maximum.

We developed an additional dynamics descriptor g_0^{std} based on the input/output pair: for a given segment $\mathbf{x}$ of the input signal and the corresponding output segment $\mathbf{y}$, the gain that matches the input to the output is given by $\tilde{g} = \arg\min_g \|\mathbf{y} - g\mathbf{x}\|_2 = \langle \mathbf{x}, \mathbf{y} \rangle / \|\mathbf{x}\|_2^2$. To predict the time variation of the gain, an exponential window is simulated by a *one-pole* filter leading to $\tilde{g}[n] = (w * xy)_n / (w * x^2)_n$, where $w[n]$ is the impulse response parametrised by a relaxation time (e.g. 5 ms). Note that x is preprocessed to align with y in time (in case of delay) and in frequency (to cancel a possible global equalization). The descriptor g_0^{std} is then expressed as the standard deviation of $\tilde{g}[n]$ over the full sound sample.

Even if this process has been directly inspired by the conventional DRC model given in section 2.1, it is not intended to estimate the true gain as in section 2.2. Instead, g_0^{std} informs about the total amount of dynamic changes. Note that the relaxation time has an impact on the value of g_0^{std}, because it determines the temporal resolution of the gain estimate. For our application, the value of 5 ms is a good trade-off between slow and fast attack/release times.

Fig. 8 shows the same descriptors under the hypercube sampling scheme, which varies all parameters of the plugins, not only the main compression amount. We can see that they strongly influence the output character in a non-linear way, showing that content-based navigation introduced here is a necessary alternative to linear parameter exploration.

3.4.3 Audio Quality Descriptors

In order to estimate the distortion introduced by a DRC plugin depending on its settings, the predicted gain is reapplied to the input. The resulting residual $r[n] = y[n] - \tilde{g}[n]x[n]$ contains all signal and spectral components introduced by the plugin, notably added harmonics from distortion. The average amplitude of the residual h in dB is therefore a good indicator of the amount of distortion introduced. Fig. 7 and 8 show this distortion measure over all parameter settings.

4. INTERACTIVE BROWSING

In order to explore the various compression plugins and parameter sets, while hearing the effect they have on the example signals, the dataset is visualised in 2D and serves as a navigation interface, as explained in detail in the following.

4.1 Navigation Interface

The visualisation and navigation interface, depicted in Fig. 1, is a 2D scatterplot where each point represents one output file produced by a plugin and parameter set from the dataset. The horizontal and vertical coordinates of the points are given by a projection to two user-selected axes out of the descriptors of the output signals. The colour and size of the points can be used to visualise two additional descriptor dimensions.

In the 2D navigation space, one can use the mouse pointer to select the closest points' compressor settings for auralisation (see 4.2 below). Selection happens either on click, or simply by hovering, depending on a mode switch. Note that the scatterplot is filtered to display the outputs of only one of the input files at a time, because, first, as shown in Fig. 6, the input file content highly influences the output descriptors and generates rather disjoint output ranges, and second, the auralisation would need to switch instantly

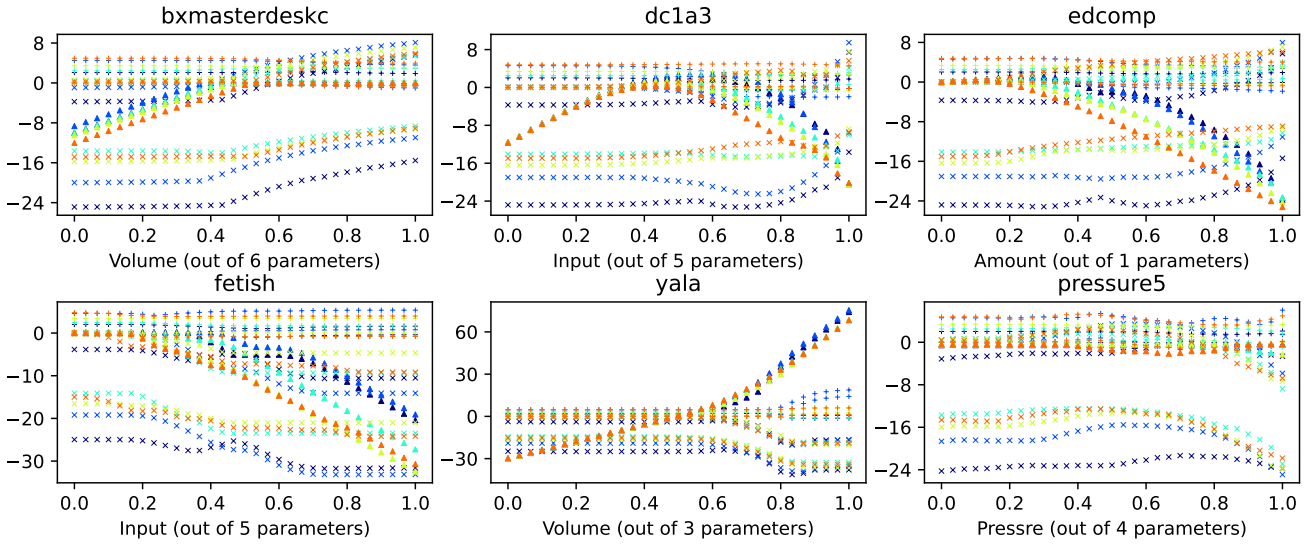


Figure 6. Absolute peak signal value [dB] (▲), integrated loudness [LUFS] (×), and integrated loudness range [LU] (+) vs. unified compression *amount* over a linear sweep.

between different sound files, creating discontinuities that hinder its readability and comparability.

4.2 Auralisation

To listen to the effect of the compressor plugin and parameter set selected in the navigation interface, the output signal produced by the selected plugin and parameter set is played in a loop, or triggered whenever a new point is selected, which allows to compare the different plugins and settings in a unified interface.

When another point is selected, simply switching the looped playback to the corresponding new output file, while keeping the playback position within it, is no problem, as the output files are all sample-aligned and differ only in their output volume and characteristics. We do apply a quick cross fade with a 20 ms ramp time to avoid click artifacts in that case.

4.3 Loudness Handling

The highly varying peak values produced by some compressors (see Fig. 6) make the auralisation of peak normalised output files impractical (for extreme settings, the bulk of the signal can be almost inaudible at −80 dB). What's more, the ecologically-valid use of a compressor in the production chain will entail consciously allowing peaks over 0 dBFS in the mix, to be reduced by the final master limiter, possibly introducing desired saturation or clipping.

Therefore, the auralisation will apply a gain calculated from each file's integrated loudness to bring it to −14 dB (the recommendation for mastering) and reduce the overshooting peaks by a transparent lookahead limiter (which does not introduce distortion) and visually indicate the limiting action. This in fact cancels out the absolute loudness changes induced by the plugins and allows one to focus on the *loudness range* and the sonic quality produced.

4.4 Implementation

The dataset is generated in python using the *dawdreamer* [14] [22] and *pyloudnorm* [21] packages. The list of plugins and original input files are configurable, so that it can be used with other collections of sounds and plugins. The code is freely available. [15] Some of the dynamics and audio quality descriptors are calculated by Matlab scripts and joined with the database metadata for import into the 2D compressor browser.

The compressor browser is implemented in the MAX [16] graphical programming environment for interactive media applications, with the MUBU [23] and CATART [17] [18] extension libraries. The selection, visualisation and auralisation parts of the patch are presented in Fig. 1.

We were able to use the CATART modules and their underlying MUBU container for time-based media and data as-is to handle the collection of compressor outputs and their descriptors, instead of general audio files. CATART's underlying architecture for importing, analyzing, and selecting audio units can be reused. The internal timbre descriptors are calculated by the PIPO [24] framework, part of the MUBU package. Especially the scalable kNN unit selection [25] is completely agnostic of the audio type, as it finds nearest neighbours in a Euclidean descriptor space and outputs the ID of the closest projected parameter set(s) (encompassing the whole compressor output file, since there is no segmentation for this usage). The only difference is that the rendering is not playing back segments of a sound file as in CATART, but the compressor output is played in a loop using the `mubu.play~` object. This object allows switching the source buffer while playing, while maintaining the playback position, such that we can auralise different compressors and parameters on the fly.

[14] https://github.com/DBraun/DawDreamer
[15] https://github.com/AQUA-RIUS/fifty-shades-of-compression
[16] http://cycling74.com/max
[17] https://ircam-ismm.github.io/max-msp

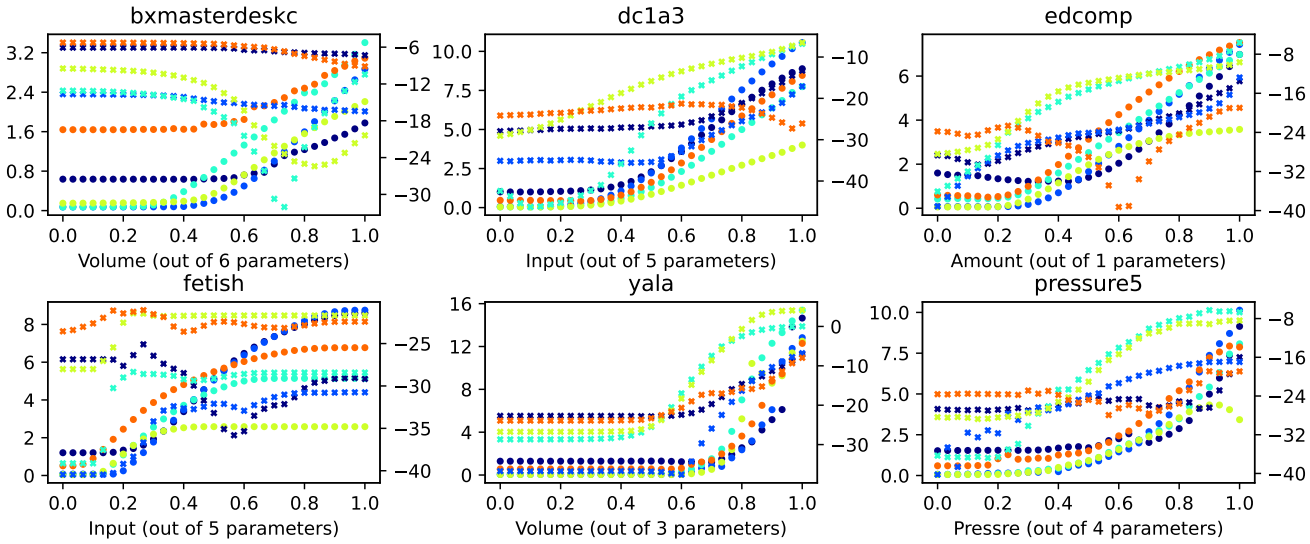


Figure 7. "Squashiness" descriptor g_0^{std} [dB, left axis] (•) and distortion measure h as residual energy [dB, right axis] (×) vs. unified compression *amount* over a linear sweep.

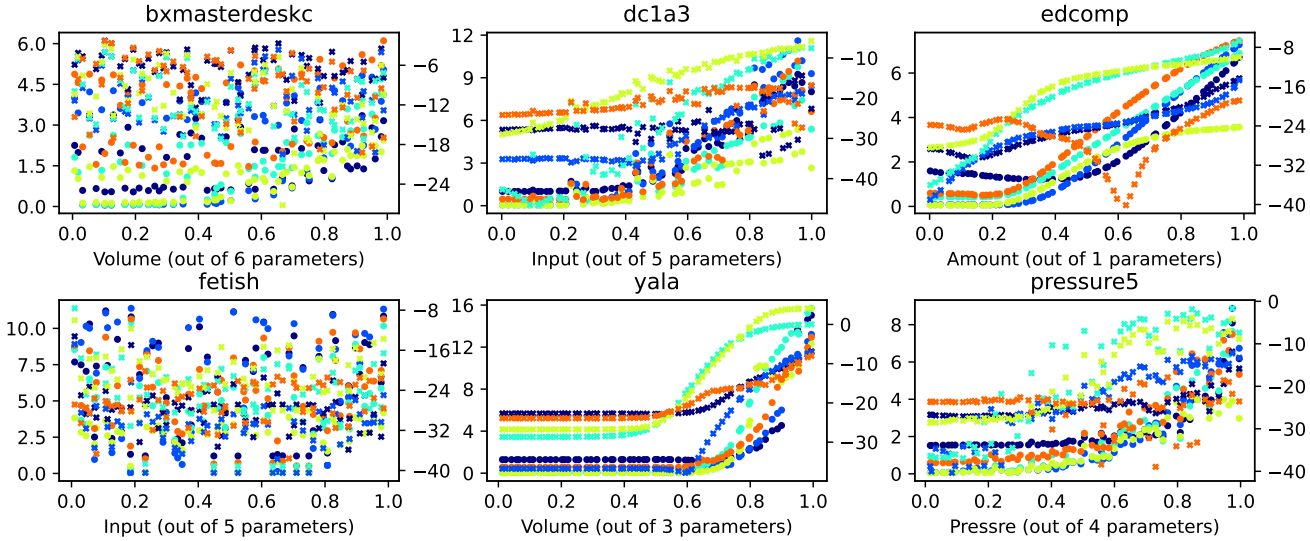


Figure 8. "Squashiness" descriptor g_0^{std} (•) and distortion measure h as residual energy in dB (×) vs. unified compression *amount* over all LHS parameter sets.

5. RESULTS

When browsing our dataset, we can immediately hear the very diverse sound characteristics produced by the 21 DRC plugins, depending on their settings and idiosyncratic implementation. They produce a wide range of dynamic profiles, some of them introduce distortion, others change subtle timbral characteristics, that are not necessarily captured by the theoretical signal processing model from section 2.1, but are clearly perceptible.

The descriptor g_0^{std} (3.4.2) very clearly expresses the "squashiness" of a plugin, i.e. the perceived effect of most of the output audio being squashed to the same loudness level, as can be seen in Fig. 7, where these values go up with rising unified compression amount, highly dependent on the input signal. De facto, higher squashiness most often correlates with higher levels of distortion h (3.4.3).

Regarding the perceptual audio descriptors (3.4.1), only little effects related to the action of the DRC plugins can be noted by listening to the outputs, except of course loudness. In those plugins that have a tonal balance control, we can see that it affects the spectral centroid, as expected. In extreme cases of distortion, we see a decrease in periodicity, without this being on a useful scale, and no effect on spectral centroid.

6. CONCLUSION AND FUTURE WORK

The high level of interactivity afforded by the corpus-based concatenative synthesis paradigm can give important insights into the influence of the varying parameters of specific plugins on the output. It also allows to perceptively validate the various measures and descriptors available against the acoustic ground truth. Notably, we could observe that the timbre descriptors (3.4.1) can not mean-

ingfully distinguish the nuances of different plugins and settings, and that the specific dynamics and audio quality descriptors (3.4.2 and 3.4.3) are needed to capture these.

We developed and informally validated two novel descriptors g_0^{std} and h expressing the action of a compressor in terms of compression strength (*squashiness*) and the introduced distortion, based on comparison of the output with the input signal. Formal listening tests are left for future work.

The auralisation is calibrated to integrated loudness, such that all output files are auditioned on the same level. While seemingly counter-intuitive, this actually allows to focus on the details of the *dynamics* and audio quality that a compressor produces, abstracting from the absolute loudness, which is a downstream decision dependent on the rest of the mix.

We hope to extend the dataset by other commercial DRC plugins through collaborative generation and collection of their outputs. We're also planning statistical evaluation of the correlations of the descriptors and further existing audio quality measures [5].

In future work, it would be possible to probe previously unseen plugins with analytic test signals, in order to predict the influence they would have on real input audio. This could then place the new plugins on the 2D map together with existing known plugins, to add them to the browsing experience by deriving the mapping from parameter to output descriptor (independent of the input audio).

In principle, the parameter browsing could work on real-time input, or as a meta-compressor in a DAW. For this, the plugins would need to be hosted and fed with the input signal in parallel, while the interactive browsing would choose the plugin to listen to and set its parameters. Moreover, this would allow interpolation of the parameters within a plugin, and different plugins could possibly be crossfaded, as we explored in our previous work [7].

Acknowledgments

This research was supported by the French ANR project AQUARIUS (ANR-22-CE23-0022).

7. REFERENCES

[1] M. Senior, *Mixing Secrets for the Small Studio*, 2nd ed., ser. Sound on Sound Presents. New York, USA: Focal Press, 2011.

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

[3] T. Wilmering, D. Moffat, A. Milo, and M. B. Sandler, "A history of audio effects," *Applied Sciences*, vol. 10, no. 3, p. 791, Jan. 2020.

[4] D. Tardieu, E. Deruty, C. Charbuillet, and G. Peeters, "Production effect: audio features for recording techniques description and decade prediction," in *Digital Audio Effects (DAFx)*, 2011.

[5] D. Fourer and G. Peeters, "Objective characterization of audio signal quality: applications to music collection description," in *Proc. IEEE ICASSP*, 2017, pp. 711–715.

[6] D. Schwarz and N. Schnell, "Sound search by content-based navigation in large databases," in *Sound and Music Computing (SMC)*, Porto, Portugal, Jul. 2009.

[7] D. Schwarz, E. Démoulin, B. Alary, and T. Carpentier, "Spaces of Spaces — Interactive Navigation in Corpora of Room Impulse Responses for Musical Production and Creation," in *Sound and Music Computing (SMC)*, 2025.

[8] U. Zölzer, *DAFX: Digital Audio Effects*. Wiley Online Library, 2011.

[9] H. Sun, D. Fourer, and H. Maaref, "Neural-enhanced dynamic range compression inversion: A hybrid approach for restoring audio dynamics," *preprint arXiv:2411.04337*, 2024.

[10] ——, "Approche hybride pour l'inversion de la compression dynamique en traitement audio à l'aide de l'apprentissage profond," in *Proc. GRETSI*, 2025.

[11] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *preprint arXiv:2104.01778*, 2021.

[12] C. Peladeau and G. Peeters, "Blind estimation of audio effects using an auto-encoder approach and differentiable digital signal processing," in *Proc. IEEE ICASSP*. IEEE, 2024, pp. 856–860.

[13] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *Journal of the Acoustical Society of America*, vol. 61, pp. 1270–1277, 1977.

[14] D. Wessel, "Timbre space as a musical control structure," *Computer Music Journal*, vol. 3, no. 2, pp. 45–52, 1979.

[15] S. MacAdams, "Perspectives on the contribution of timbre to musical structure," *Computer Music Journal*, vol. 23(3), pp. 85–102, 1999.

[16] D. Schwarz, "Concatenative sound synthesis: The early years," *Journal of New Music Research*, vol. 35, no. 1, pp. 3–22, Mar. 2006, special Issue on Audio Mosaicing.

[17] ——, "Corpus-based concatenative synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 92–104, Mar. 2007, special Section: Signal Processing for Sound Synthesis.

[18] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-Time Corpus-Based Concatenative Synthesis with CataRT," in *Digital Audio Effects (DAFx)*, 2006, pp. 279–282.

[19] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, "The timbre toolbox: Extracting audio descriptors from musical signals," *The Journal of the Acoustical Society of America*, vol. 130, no. 5, pp. 2902–2916, 2011.

[20] A. de Cheveigné and N. Henrich, "Fundamental Frequency Estimation of Musical Sounds," *Journal of the Acoustical Society of America (JASA)*, vol. 111, p. 2416, 2002.

[21] C. J. Steinmetz and J. Reiss, "pyloudnorm: A simple yet flexible loudness meter in python," in *Audio Engineering Society Convention 150*. Audio Engineering Society, 2021.

[22] D. Braun, "Dawdreamer: Bridging the gap between digital audio workstations and python interfaces," in *ISMIR Late Breaking Demos*, 2021.

[23] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, and R. Borghesi, "MuBu & friends – assembling tools for content based real-time interactive audio processing in Max/MSP," in *International Computer Music Conference (ICMC)*, 2009.

[24] N. Schnell, D. Schwarz, and J. Larralde, "PiPo, a plugin interface for afferent data stream processing modules," in *International Symposium on Music Information Retrieval (ISMIR)*, Suzou, PRC, Oct. 2017.

[25] D. Schwarz, N. Schnell, and S. Gulluni, "Scalability in content-based navigation of sound databases," in *International Computer Music Conference (ICMC)*, 2009.