AUTOMATIC CONTROL OF THE DYNAMIC RANGE COMPRESSOR USING A REGRESSION MODEL AND A REFERENCE SOUND

Di Sheng

Centre for Digital Music (C4DM),
Queen Mary University of London
London, UK
d.sheng@qmul.ac.uk

György Fazekas

Centre for Digital Music (C4DM),
Queen Mary University of London
London, UK
g.fazekas@qmul.ac.uk

ABSTRACT

Practical experience with audio effects as well as knowledge of their parameters and how they change the sound is crucial when controlling digital audio effects. This often presents barriers for musicians and casual users in the application of effects. These users are more accustomed to describing the desired sound verbally or using examples, rather than understanding and configuring low-level signal processing parameters. This paper addresses this issue by providing a novel control method for audio effects. While a significant body of works focus on the use of semantic descriptors and visual interfaces, little attention has been given to an important modality, the use of sound examples to control effects. We use a set of acoustic features to capture important characteristics of sound examples and evaluate different regression models that map these features to effect control parameters. Focusing on dynamic range compression, results show that our approach provides a promising first step in this direction.

1. INTRODUCTION

The invention of recording in the late nineteenth century democratised music listening and gave rise to a new industry concerning the production, distribution and reproduction of music. Due to technical and aesthetic requirements, the industry developed an increasingly large number of tools for the manipulation of audio content to achieve desired sound qualities. Changing the dynamic range, timbre or frequency balance of recordings have first become widely possible with the introduction of analogue signal processing techniques, for instance, linear filters and non-linear effects like the compressor. Digital technologies such as software plugins or audio effects embedded in Digital Audio Workstations have significantly extended and, to some extent, replaced analogue effects. However, from the users' point of view, they rarely go beyond mimicking the operation of analogue counterparts.

Controlling effects requires significant experience and knowhow, especially when used for aesthetic purposes during music production [1]. This often involves mapping a concept or idea concerning sound qualities to low-level signal processing parameters with limited meaning from a musical perspective. Knowledge of signal processing, which was requisite for engineers in early studios, as well as good understanding of their control parameters and function constitute the skills of sound engineers and producers. Acquring these skills however present a high barrier to musicians and casual users in applying today's production tools. Consequently, the development of intelligent tools, as it has been done in other industries, may greatly benefit music production.

Substantial amount of works in this area are concerned with automating the mixing and mastering process (see e.g. [2] or [3]). Our work is significantly different from previous studies in that it does not directly target multitrack mixing and mastering, or attempt to use high-level semantic descriptors to control effects. Our focus is on the novel task of estimating the parameters of audio effects given a sound example, such that the processed audio sounds similar in some relevant perceptual attributes (e.g. timbre or dynamics) to the reference sound. This has applications in various stages of music production. For instance, while creating an initial rough mix of a track, artists may describe how they would like an instrument to sound using an actual sound example [1]. An intelligent tool that provides audio effects settings based on a reference audio track is useful to meet this requirement. It may also help hobbyists and amateur to make their own music or create remixes, an activity encouraged by well-known bands such as Radiohead, by releasing stems and multitrack recordings.

It may be a considerable effort to develop an intelligent tool that estimates the parameters of different types of effects using complex audio material. To assess the feasibility of solving this problem, we first simplify the task and focus on a single effect: dynamic range compression, and simple audio material: monotimbral notes and loops. The proposed solution consists of 1) an audio feature extractor that generates features corresponding to each parameter, 2) a regression model that maps audio features to audio effect parameters and 3) a music similarity measure to compare the processed and the reference audio.

The rest of the paper describes the proposed algorithm in detail. It is structured as follows: Section 2 outlines related work, Section 3 shows the workflow of our system, the evaluation and analysis is discussed in Section 4, followed by future work and conclusion in Section 5.

2. RELATED WORK

In this section, we provide a brief overview of intelligent audio production technologies with and emphasis on the dynamic range compressor (DRC), a non-linear time dependent audio effect. The area of intelligent audio production has become a burgeoning field over the last decade, with solutions ranging from automatic mixing systems [2] to intelligent audio editors [4][5]. These systems typically rely on audio feature extraction to analyse one or more channels and embed expert knowledge into procedural algorithms to automate certain aspects of the production workflow. The goal in many cases is the delivery of a technically correct mix by controlling the gain or loudness balance of sources in multitrack recordings. Finding the optimal dynamic range for each instrument[3] or reducing the number of user configurable parameters of an effect [6] have also been considered.

Other approaches aim at providing alternative control mecha-

nisms, such as the use of semantic descriptors or simplified graphical interfaces. Loviscach [7] for instance describes a control method for equalisation that allows drawing points or using free hand curves instead of setting parameters. Subsequent work [8] provides a creative method to map features to shapes. However, the shapes of this system do not link directly with the "meaning" of settings, rather they are classifications of the settings. Cartwright and Pardo [9] outline a control strategy using description terms such as warm or muddy, and demonstrate a method of applying high-level semantic controls to audio effects. Wilmering et al. [10] describe a semantic audio compressor that learns to associate control parameters with musical features such as the occurrence of chord patterns. This system provides intelligent recall functionality.

The idea of cross adaptive audio effects [2] was introduced in the context of automatic multitrack mixing. In this scenario, audio features extracted from multiple sources are utilised and automatic control is applied to optimise inter-channel relations and the coherence of the whole mix. These systems often follow expert knowledge obtained from the literature about music production practice, or interviews with skilled mixing engineers. In [3], the authors explored how different audio engineers use the compressor on the same material to train an intelligent compressor. An automatic multitrack compressor based on side-chain feature extraction is presented in [6] providing automatic settings for attack/release time, knee, and make up gain based on low-level features and heuristics. An alternative implementation of DRC using non-negative matrix factorisation (NMF) is proposed in [11]. Here, the authors consider raising the NMF activation matrix to $\frac{1}{R}$ to obtain a compressed signal with ratio R after re-synthesis. This is viewed as a compressor without a threshold parameter.

The goal of our work is substantially different from these solutions. At this initial stage, we focus on individual track compression, rather than trying to fit the signal into a mix, therefore we do not assume the presence of other channels. We do not aim to incorporate expert knowledge into the system or automate control parameters in a time varying manner, instead, we assume an audio example, and aim to configure the compressor to yield an output that sounds similar to the example. This requires the prediction of each parameter.

To estimate the settings of a compressor with hidden parameter values, Bitzer et. al. [12] proposes the use of purposefully designed input signals. Although this provides insight into predicting DRC parameters, in most cases, including ours, only the audio signal is available and the original compressor and its settings are not available for black-box testing. A reverse engineering process is proposed for the entire mix in [13]. The authors estimate a wide range of mixing parameters including those of audio effects. This work however focusses on the estimation of time-varying gain envelopes associated with dynamic non-linear effects rather than their individual parameters. A recent work proposes the use of deep neural networks (DNN) for the estimation of gain reduction [14]. The use of DNN in intelligent control of audio effects is quite novel, but this work targets only the mastering process and only considers the ratio factor.

Our work focuses on comparing linear and non-linear regression models to map low-level audio features discussed in Section 3 to common parameters of the dynamic range compressor. We propose using a reference audio example as target and evaluate the regression models in terms of how close the processed sounds get to the target in overall dynamic range and audio similarity. This is motivated by the need to analyse and compare changes in the dynamic and spectral characteristic of the processed sounds, since both are affected by DRC. To this end, we measure peak-to-RMS ratio and also use a simple baseline model of audio similarity consisting of a Gaussian Mixture Model (GMM) trained on Mel Frequency Cepstrum Coefficients(MFCCs) [15]. The Kullback-Leibler (KL) divergence is a robust method to measure the similarity between single Gaussian distributions [16][17]. However, the divergence between multiple Gaussian models is not analytically tractable, therefore we use the approach proposed in [18] based on variational Bayes approximation. In the next section, we discuss regression model training and DRC parameter estimation.

3. METHODS

3.1. Training procedure

This study uses a single-channel, open-source dynamic range compressor developed in the SAFE project [19]. In the interest of brevity, we do not discuss the operation of the compressor and assume the reader is familiar with relevant principles[20, 6]. We consider the estimation of the most common parameters: threshold, ratio, attack and release time from reference audio and leave other parameters e.g. make up gain and knee width for future work. To form an efficient regression model, we need to choose the most relevant features first. Section 3.1.1 describes the features derived from a series of experiments. The system is then discussed in Section 3.1.2 outlining the data flow and system structure.

3.1.1. Feature extraction

Audio features are selected or designed for each specific effect parameter. Since DRC affects perceptual attributes in terms of loudness and timbre, six statistical features are selected for all four parameters. The RMS features reflect energy, which is related to loudness, while the spectral features reflect the spectral envelope, which is related to timbre. The statistical features are calculated frame-wise, with a frame size of 1024 samples and a 50% overlap. For spectral features, we use 40 frequency bins up to 11kHz. We assume this bandwidth is sufficient for the control of selected DRC parameters. We define the magnitude spectrogram $Y(n,k) = |X(n,k)|$ with $n \in [0 : N-1]$ and $k \in [0 : K]$ where N is the number of frames and k is the frequency index of the STFT of the input audio signal with a window length of $M = 2(K+1)$. We extract the spectral features described in Equations 1 - 4 as follows:

$$\mathsf{SC}_{\mathsf{mean}} = E[\frac{\sum_{k=0}^{K-1} k \cdot Y(n,k)}{\sum_{k=0}^{K-1} Y(n,k)}], \tag{1}$$

$$\mathsf{SC}_{\mathsf{var}} = Var[\frac{\sum_{k=0}^{K-1} k \cdot Y(n,k)}{\sum_{k=0}^{K-1} Y(n,k)}], \tag{2}$$

$$\mathsf{SV}_{\mathsf{mean}} = E[(E[Y(n,k)^2] - (E[Y(n,k)])^2)^{1/2}], \tag{3}$$

$$\mathsf{SV}_{\mathsf{var}} = Var[(E[Y(n,k)^2] - (E[Y(n,k)])^2)^{1/2}], \tag{4}$$

where SC stands for spectral centroid, and SV stands for spectral variance. The mean and variance in the equations are calculated across all M length frames.

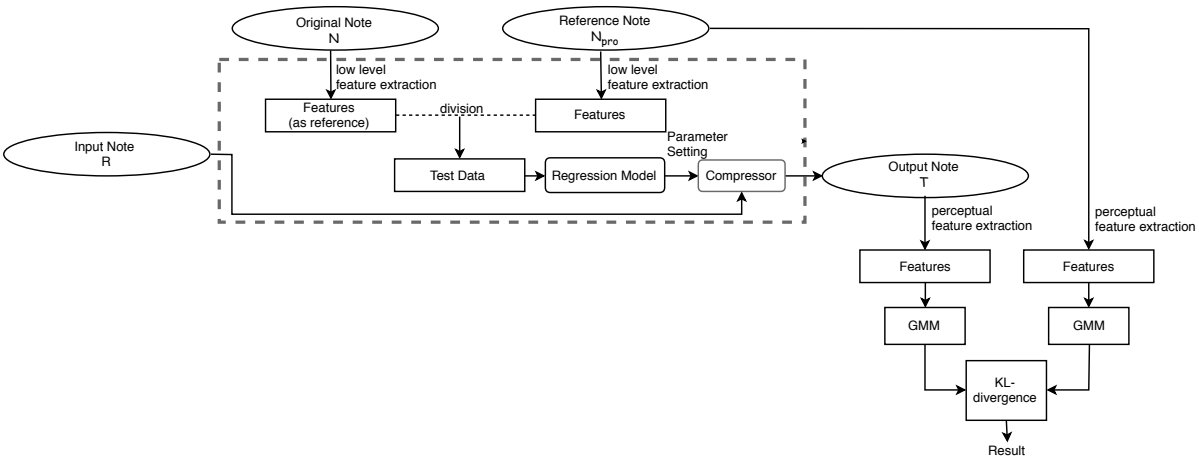

Figure 1: *Workflow of initial system with access to the reference sound and its corresponding unprocessed version (see Section 3.2.2)*

We also extract the following temporal features described in Equations 5 - 6:

$$\mathsf{RMS}_{\mathsf{mean}} = E[(\frac{1}{M}\sum_{m=0}^{M-1} x(m)^2)^{1/2}], \tag{5}$$

$$\mathsf{RMS}_{\mathsf{var}} = Var[(\frac{1}{N}\sum_{m=0}^{N-1} x(m)^2)^{1/2}], \tag{6}$$

where $x(m)$ represents the magnitude of audio sample m within each M length frame, and the mean and variance are calculated across all the N time frames as with the previous spectral features.

We designed four types of time domain features related to the attack and release of the notes as well as the speed of the compressor. The attack and release times $T_A = T_{endA} - T_{startA}$ and $T_R = T_{endR} - T_{startR}$ are calculated using the RMS envelope through a fixed threshold method (c.f. [21]) that determines start and end times of the attack and release parts of the sound. The end of the attack is considered to be the first peak that exceed 50% of the maximum RMS energy. The RMS curve is smoothed by a low-pass filter with a normalised cut-off frequency of 0.47 rad/s. We also extract the RMS amplitude at the end of the attack and the start of the release $rms(T_{endA})$ and $rms(T_{startR})$ respectively, as well as the mean amplitude during the attack and release parts of the sound.

$$A_{att} = \frac{1}{T_A}\sum_{n=T_{startA}}^{T_{endA}} rms(n), \tag{7}$$

$$A_{rel} = \frac{1}{T_R}\sum_{n=T_{startR}}^{T_{endR}} rms(n), \tag{8}$$

where T_{start} and T_{end} are indices of the start and end of the attack or release. Finally, we compute a feature related to how fast the compressor operates. We first calculate the ratio between the time-varying amplitudes of input or original sound and the reference sound $s(n) = rms_{ref}(n)/rms_{orig}(n)$. We then calculate the amount of time for $s(n)$ to reach a certain value using a fixed threshold. This relates to the speed of compressor to reach the desired compression ratio, which is controlled primarily by its attack time. The same process is applied at the end of the note to extract how fast the ratio curve drops back to one. The design of these features was motivated by visual inspection of the signal. They are shown to improve the ability of the regression model to predict the parameters (see Section 4).

3.1.2. Regression model training

This section outlines the datasets and training procedure for the regression models that are used to map features to effect parameter settings. In the first stage of our research, we consider two types of instruments: snare drum and violin. The former is one of the most common instruments that requires at least a light compression to even out dynamics. The drum samples are typically short and, considering a typical energy envelope, exhibit only the attack and release (AR) part. The violin recordings typically consist of a long note with fairly clear attack, decay, sustain and release (ADSR) phase. All audio samples in our work are taken from the RWC isolated note database [22].

Table 1 describes the four violin note datasets denoted $A, ..., D$ that are used for training. In each dataset, one parameter of the effect is varied while the others are kept constant. The number of training samples in each dataset equals to the number of notes, i.e., 60 in case of the violin dataset, times the number of grid points (subdivisions) for each changing parameter. In this study, we use 50 settings for threshold and ratio, and 100 settings for attack and release time as it is shown in the first column of Table 1. The same process is applied to 12 snare drum samples to form the drum dataset. Each training set $A, ..., D$ is used for predicting a specific parameter.

Training sets (size)	Conditions			
	Thr(dB)	Ratio	Att(ms)	Rel(ms)
A (60*50)	0:1:49	2	5	200
B (60*50)	37.5	1:0.4:20	5	200
C (60*100)	37.5	2	1:1:100	200
D (60*100)	37.5	2	5	50:10:1000

Table 1: *Training set generation*

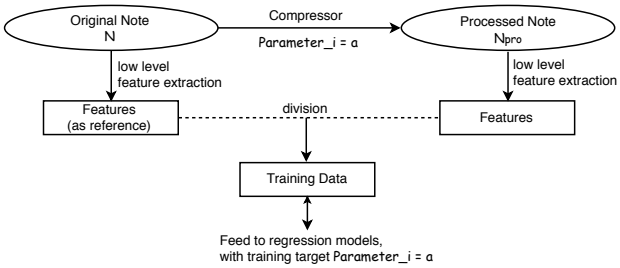

Figure 2: *Flowchart of training data generation*

Figure 2 describes the workflow. Taking training set A as an example, the original notes N are the recorded violin notes. The processed notes denoted N_{pro} on the right hand side of the figure are generated from N, which are processed by the compressor with different threshold values. There are 60 different notes N, and each N generates 50 processed notes N_{pro}. This yields $60 \times 50 = 3000$ N_{pro} in the training dataset for threshold. Because the features from N_{pro} are highly correlated with the original note N, the ratio between them is used to focus on how the features change as a result of dynamic compression. Therefore, it is the difference we actually used to train the regression model. There are 6 features related to the threshold extracted from each note, therefore training data A comprises 3000 N_{pro} $\times 6$ feature vectors. In the following, this training data is used to generate the regression model. The same principle applies to training sets B, C and D.

In our work, two regression models are compared and evaluated (see Section 4), simple linear regression, as well as random forest regression [23]. Random forest uses averaging over subsamples from the dataset to improve the predictive accuracy of the model as well as to mitigate over-fitting problems. The use of this latter model is motivated by the hypothesis that the relationship between the audio features and the compressor parameters may not be modelled accurately enough with simple linear regression due to the non-linearities in the process. In the evaluation, we use the implementations available in [24].

3.2. System design and testing procedure

3.2.1. Numerical test at the isolated note level

In this paper, we propose two system designs. The first aims only at verifying the basic idea behind the use of a reference sound (or note) to be approximated. This is not a realistic scenario, because it assumes we have access to both the processed and unprocessed (original) version of the sound used as reference. This is needed because the predictor variables, i.e., the input to the regression model is calculated as the ratio of the audio feature data extracted from these recordings. This requires access to all pairs of (N, N_{pro}) in the training data. This scenario is presented in Figure 1 consisting of two parts. The components within the dashed line box represent the actual control system for the compressor with three inputs: the input note R to be processed, the reference note N_{pro} to be approximated, and its corresponding original note N from the training set. The output note T outside of the dashed line box will be used in the evaluation, where we compare the similarity of the output and the reference. As it is mentioned in Section 3.1.2, the regression model is trained on the ratio, therefore, we need to provide the same data for the model to predict the compressor parameters. The original note N is used only in the process of generating feature vectors.

Before we use the similarity model depicted on the right hand side of Figure 1, we first evaluate the regression model accuracy. This first study compares predicted parameter values with the actual ones. The workflow is the same as Figure 2, providing a standard testing step for regression models. In this study, we use repeated random sub-sampling validation (Monte Carlo variation). 10% of each feature vectors are used for testing, while the remaining 90% are used as training data. This experiment is repeated 100 times and the average results are reported in Section 4.

3.2.2. Similarity assessment at the isolated note level

Considering the motivations and use cases described in Section 1, the desired output of this algorithm is to make an unrelated note R sound similar to the reference note N_{pro}, where N_{pro} is generated through a compressor with e.g. its threshold set to xdB. However, even if the prediction is perfect with $x_p = x$, the same compressor for note R and note N can give different perceptual results. Therefore, a similarity model which takes this into account is used to evaluate the similarity between N_{pro} and the algorithm output T. The processing and evaluation workflow is represented in Figure 1 with the structure within the dashed line box used to control the system while the components on the right hand side are used in the similarity test.

In the similarity assessment, we use a simple and frequently used model of audio similarity [15] as well as a simple feature which is a good (although partial) indicator of the overall dynamics of the signal. First, we consider the crest factor and report the difference between the reference and the processed sound. Second, we follow the similarity model using a Gaussian Mixture Model trained on Mel Frequency Cepstrum Coefficients. Accordingly, the feature extraction in the workflow indicates the calculation of the divergence between two multiple Gaussian models, which provides the similarity information. We use the symmetrised Kullback-Leibler (KL) divergence, which is commonly used for Gaussian models, and since we use a GMM, an approximation of the KL divergence is calculated using the approach presented in [18]. The results of this test and analyses are provided in Section 4.

3.2.3. Note level similarity assessment in a realistic scenario

In a real world scenario, if the reference N_{pro} is a commercial audio track, its corresponding unprocessed original sound N is not likely to be available. In this case, we propose to use the system design outlined in Figure 3, where the input of the system are limited to the input note R to be processed and the reference note N_{pro}. In the feature computation workflow, the original note N is replaced by the input note R, because the features capture the difference between the reference note and the original note. Measuring the difference between the reference note N and the input R can be seen more reasonable and closer to a real world scenario.

3.2.4. Loop level similarity assessment in a realistic scenario

This study extends the objective of the experiment from using mono-timbral notes to longer mono-timbral loops. The loops we used are approximately 5 seconds long, consisting of violin loops taken from the RWC Instrument Sound Database [22]. Under the

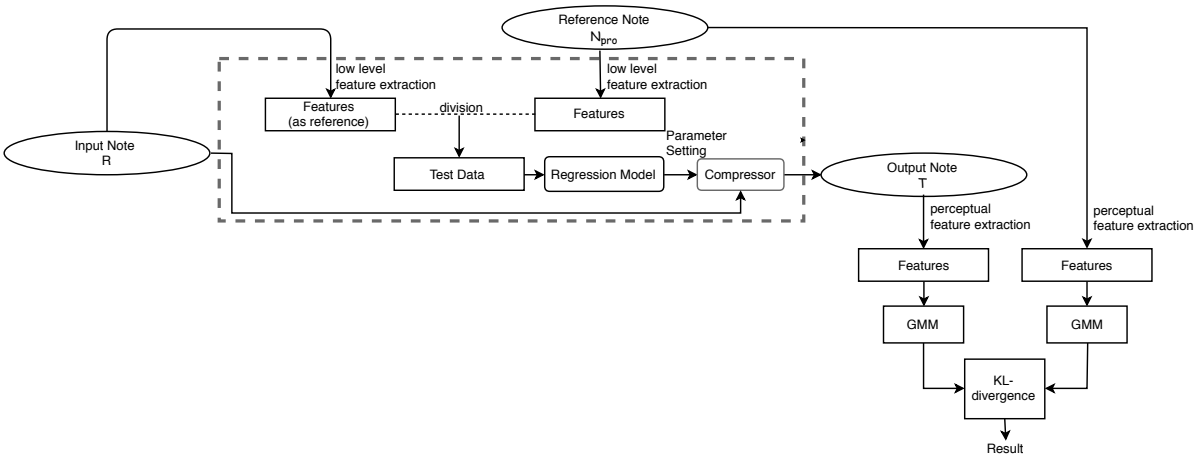

Figure 3: *Realistic workflow with only the input and reference sound available (see Section 3.2.3)*

constraint of mono-timbre, i.e., only a single instrument is sounding at a time, we assume that the statistical properties of the underlying features remain constant. Therefore, the rest of the algorithm remains the same. A possible method to apply our algorithm to loops is to consider them as notes, with the attack of the first note in the loop and release of the last. It is a reasonable simplification in practice, noting that the rest of the attack and release parts of notes are likely to be overlapping. As before, the results are reported and discussed in Section 4.3.

4. EVALUATION

4.1. Direct assessment of parameter estimation

Using the test procedure outlined in Section 3.2.1, here we report the accuracy of direct parameter estimation using random subsampling validation. Two regression models are compared and evaluated: simple Linear Regression (LR) and Random Forest regression (RF) [23] available in [24]. Table 2 & 3 show the absolute errors for both instruments and regression models. Since the observed feature values are relatively small, we linearly scale the feature values to $[0, 1]$ and compare the errors. The highlighted values in the table show that the smallest error is always observed when using the scaled features and the random forest regression model. For completeness, the range for the four parameters are (0,50] dB for threshold, [1,20] for ratio, (0,100] ms for attack time, and (0,1000] ms for release time. Scaling is reasonable in this pilot experiment, because the test data (reference) is selected randomly from the database mentioned above, which means all N_{pro} has its original N available. However, in a real world scenario, the reference sound is not taken from the database prepared to train the regression models. It is more likely to be a produced sound or track without access to its unprocessed version. Therefore scaling will not be used in the subsequent studies. Please note that we do not overfit the models, because in the evaluation the reference note and the corresponding original is excluded from the training set of the regression model.

The results also illustrate that the prediction accuracy for drums is better than for violins in all cases. One reason is that drum samples are shorter and exhibit a simpler structure - short sustain, followed by release and there is no pitched content. It shows that the system can predict the compressor parameters from drums better than for violins. In subsequent studies, we will therefore focus on the more complex case of similarity measurement for violin notes and loops.

Violin		LR	RF
Threshold(dB)	error	3.756	2.601
	scaled error	1.860	**1.731**
Ratio	error	2.065	1.583
	scaled error	0.110	**0.091**
Attack(ms)	error	15.503	0.719
	scaled error	1.012	**0.686**
Release(ms)	error	210.43	13.973
	scaled error	78.913	**10.583**

Table 2: *Numerical test using linear and random forest regression model for violin notes*

Snare Drum		LR	RF
Threshold(dB)	error	1.185	0.800
	scaled error	0.408	**0.345**
Ratio	error	1.571	0.999
	scaled error	0.669	**0.305**
Attack(ms)	error	6.867	0.860
	scaled error	2.260	**0.017**
Release(ms)	error	23.045	0.999
	scaled error	40.960	**6.851**

Table 3: *Numerical test using linear and random forest regression model for snare drum samples*

4.2. Results of similarity assessment between notes

In this section, we evaluate the changes in estimated similarity using the system outlined in the right hand side of Figure 1 and Figure 3. Firstly, we extract the crest factor, i.e., peak-to-RMS ratio as the similarity feature because it is correlated with the overall dynamic range of the signal. Based on our design, the crest factor of the reference note N_{pro} should be closer to the output note T than the input note R. An example of this test is given in Figure 4 with 25 test cases. The crest factor of the input signal is represented by

the constant at the top of the figure and the crest factor of a series of reference notes are depicted by the blue curve at the bottom. The crest factor of the output signal from the system is shown in the middle (green curve). It is consistently brought closer to the reference which fits our expectation here.

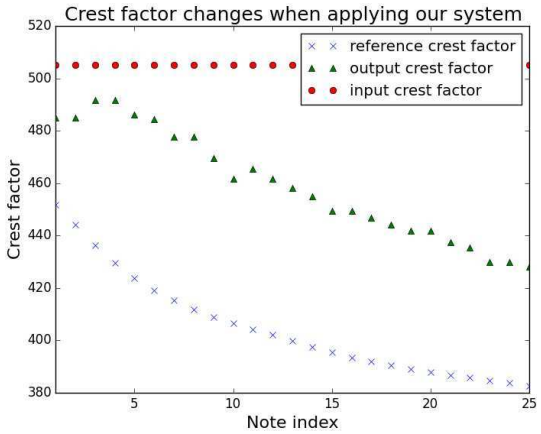

Figure 4: *Example of changing crest factor with a fixed input and decaying reference sound*

We test 50 reference notes and present the results in Table 4 for violin notes and Table 5 for snare drum, where $D_{Crest}(A, B) = mean(|Crest(A) - CrestB|)$. The results indicate that the system manages to bring the output closer to the reference using both regression models. In all parameters except threshold, the random forest model performs better than simple linear regression.

Violin	Threshold	Ratio	Attack	Release
$D_{Crest}(N_{pro}, R)$	60.31	94.13	104.93	85.31
$D_{Crest}(N_{pro}, T)_{LR}$	**12.53**	39.72	46.76	48.62
$D_{Crest}(N_{pro}, T)_{RF}$	15.27	**38.24**	**45.23**	**47.19**

Table 4: *Average of Crest factor difference - Violin*

Snare Drum	Threshold	Ratio	Attack	Release
$D_{Crest}(N_{pro}, R)$	49.14	70.04	50.47	70.68
$D_{Crest}(N_{pro}, T)_{LR}$	**27.99**	43.85	27.28	44.63
$D_{Crest}(N_{pro}, T)_{RF}$	29.33	**43.45**	**27.20**	**43.44**

Table 5: *Average of Crest factor difference - Snare drum*

Next, we discuss the resuts of similarity assessment as described in Section 3.2.2. An this stage, we use a simple audio similarity model to test the efficiency of the system. We use MFCC coefficients as features and fit a GMM on the MFCC vectors. An approximation of the symmetrised KL divergence is then calculated and used as a distance measure. Using the same procedure as in the previous part, the compressor settings provided by this algorithm should bring the output note T closer to the N_{pro} compared to the input note R. Thus it is reasonable to assume that $D(N_{pro}, R) > D(N_{pro}, T)$ holds and the performance of the regression models can be tested. In this experiment, we select 50 N_{pro} and change one parameter at a time. Table 6 & 7 indicate the efficiency of the algorithm. Since the similarity algorithm theoretically captures the timbre information as well, it will yield different results on different instruments. In this test, the distance reduction achieved by the system is larger for violins, i.e., the violin notes exhibit better results than the snare drums. This is possibly due to the fact that the MFCC features used here do not model the drum sounds well enough. Finding a better feature representation for percussive instruments constitutes future work.

Violin	Threshold	Ratio	Attack	Release
$D(N_{pro}, R)$	38.122	53.187	44.018	55.206
$D(N_{pro}, T)_{LR}$	19.799	20.911	22.852	20.994
$D(N_{pro}, T)_{RF}$	**19.742**	**20.856**	**22.213**	**20.807**

Table 6: *KL Divergences for the first workflow in 3.2.2 - Violin*

Snare Drum	Threshold	Ratio	Attack	Release
$D(N_{pro}, R)$	77.497	112.368	73.559	91.487
$D(N_{pro}, T)_{LR}$	73.749	88.574	73.307	**85.022**
$D(N_{pro}, T)_{RF}$	**73.696**	**88.487**	**73.238**	86.081

Table 7: *KL Divergences for the first workflow in 3.2.2 - Drum*

Finally, we investigate how the proposed algorithm works in a more realistic scenario. When the original note N is not available, it is reasonable to use the input note R in place of N. Under this condition, we repeat the same test for both crest factor and the MFCC-based similarity model. The results for crest factor are provided in Table 8 for violin and in Table 9 for snare drum samples. The system is still able to bring the crest factor of the output T closer to the reference N_{pro}, but the efficiency is worse compared to the case when the original note is available. Random forest regression still yields better performance in almost all cases.

Violin	Threshold	Ratio	Attack	Release
$D_{Crest}(N_{pro}, R)$	60.31	94.13	104.93	85.31
$D_{Crest}(N_{pro}, T)_{LR}$	34.86	29.37	68.74	75.94
$D_{Crest}(N_{pro}, T)_{RF}$	**30.11**	**25.36**	**67.82**	**54.02**

Table 8: *Average of Crest factor difference - Violin*

Snare Drum	Threshold	Ratio	Attack	Release
$D_{Crest}(N_{pro}, R)$	49.14	70.04	50.47	70.68
$D_{Crest}(N_{pro}, T)_{LR}$	**38.01**	66.18	**21.37**	44.05
$D_{Crest}(N_{pro}, T)_{RF}$	42.90	**45.24**	27.37	**42.75**

Table 9: *Average of Crest factor difference - Drum*

The result using the MFCC-based similarity model is illustrated in Figure 5 & 6, with $D(N_{pro}, R)$ on the left, $D(N_{pro}, T)_{LR}$ in the middle and $D(N_{pro}, T)_{RF}$ on the right of each subplot. In Figure 5 for violin notes, the average divergence is not as promising, especially when comparing with the results in Table 6, but it is clear that even if the given reference sounds have a large diversity, the algorithm reduces this significantly, and shows a very stable improvement in the similarity result. On average, the random forest regression performs better than linear regression in all cases except when predicting threshold. This shows the benefit of modelling non-linearities. Therefore we will build our system using random forest regression. Figure 6 shows that the output of the system did not manage to achieve a dramatic reduction in the similarity distance in case of the snare drum. As explained before, we

need to further investigate the influence of timbre on this similarity algorithm. Furthermore, due to the size of the snare drum dataset, we have a limitation in the choice of test data.

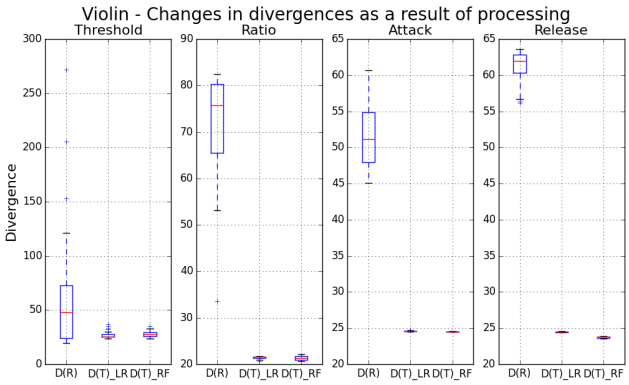

Figure 5: *Similarity for four parameters in the second workflow, assuming the origin note N is not available. - Violin*

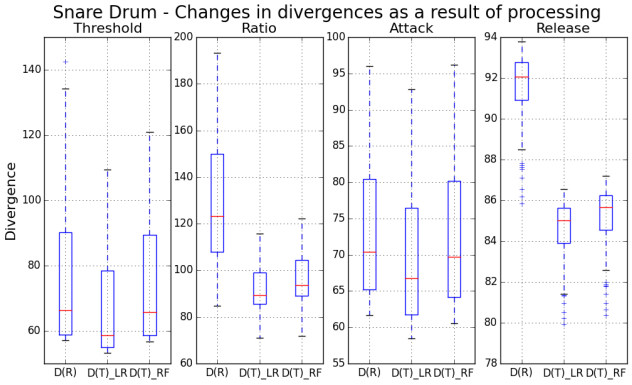

Figure 6: *Similarity for four parameters in the second workflow, assuming the origin note N is not available. - Snare drum*

4.3. Results of similarity between loops

We can extend the study by changing the audio material from mono-timbral notes to mono-timbral loops. The final experiment tests the efficiency of the workflow in Figure 3 without using the original notes, and replaces both R and N_{pro} with violin loops. The results displayed in Table 10 and Figure 7 correspond to 50 violin loops which have the duration of 3-5 seconds. A promising trend is observed using the average divergence. However, unlike in the previous studies, the divergence does not drop very significantly. These results indicate that the algorithm works better for attack/release time, but the performance is not yet satisfactory for threshold. A possible reason is that we selected six features for threshold/ratio but ten for attack/release while the features used may not be sufficient or accurate enough in this more generic case.

	Threshold	Ratio	Attack	Release
$D_{Crest}(N_{pro}, R)$	545.48	580.10	574.33	569.69
$D_{Crest}(N_{pro}, T)_{LR}$	**301.59**	237.24	326.68	**314.11**
$D_{Crest}(N_{pro}, T)_{RF}$	301.75	**209.06**	**325.08**	321.23

Table 10: *Average of Crest factor difference - Loops*

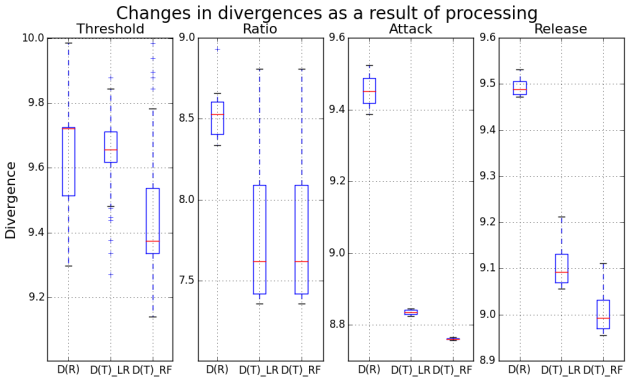

Figure 7: *Similarity for four parameters in the second workflow, assuming the origin note N is not available. - Loops*

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed a method to estimate dynamic range compressor settings using a reference sound. We demonstrate the first steps towards a system to configure audio effects using sound examples, with the potential to democratise the music production workflow. We discussed progress from using a linear regression model to a random forest model and from a designated test case to a real world scenario. The evaluation shows a promising trend in most cases and provides an initial indication of the utility of our system. Our current research focuses on simple audio material, notes and mono-timbral loops. The study discussed in Section 3.2.4 considers loops as signal notes, which ignores a lot of information. In future work, we will assess the use of an onset event detection to identify notes from loops or more complex recordings and measure their attributes using the method applied to individual notes. A useful intelligent audio production tool will be devised for audio tracks that are more complex, while polyphonic tracks will also be considered in future research.

The algorithm itself may also need improvement. The features we chose to train the regression model can be extended. Thorough research on how to design audio features specific to compressor and other audio effects parameters would be beneficial. At the same time, the regression models employed in this work may be improved using optimisation techniques that take the similarity features into account. An improved similarity model may be used as an objective function, rather than being used only during evaluation. We note however that the currently employed technique in the assessment of similarity is only considered a starting point. More realistic and complex auditory models will be applied in future work. Additionally, we plan to evaluate the system using real human perceptual evaluation, i.e., using a listening test. In conclusion, this paper provides an innovative and feasible method for intelligent control of dynamic range compressor. However, this research is still in early phase and further considerations are needed to optimise feature design and the regression model.

6. ACKNOWLEDGEMENTS

This work has been part funded by the FAST IMPACt EPSRC Grant EP/L019981/1 and the European Commission H2020 research and innovation grant AudioCommons 688382.

7. REFERENCES

[1] Sean McGrath, Alan Chamberlain, and Steve Benford, "Making music together: An exploration of amateur and pro-am grime music production," in *Proceedings of the Audio Mostly 2016*. ACM, 2016, pp. 186–193.

[2] Joshua D Reiss, "Intelligent systems for mixing multichannel audio," in *17th International Conference on Digital Signal Processing (DSP)*. IEEE, 2011, pp. 1–6.

[3] Zheng Ma, Brecht De Man, Pedro DL Pestana, Dawn AA Black, and Joshua D Reiss, "Intelligent multitrack dynamic range compression," *Journal of the Audio Engineering Society*, vol. 63, no. 6, pp. 412–426, 2015.

[4] Roger B Dannenberg, "An intelligent multi-track audio editor," in *Proceedings of international computer music conference (ICMC)*, 2007, vol. 2, pp. 89–94.

[5] Yuxiang Liu, Roger B Dannenberg, and Lianhong Cai, "The intelligent music editor: towards an automated platform for music analysis and editing," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, pp. 123–131. Springer, 2010.

[6] Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss, "Parameter automation in a dynamic range compressor," *Journal of the Audio Engineering Society*, vol. 61, no. 10, pp. 716–726, 2013.

[7] Jörn Loviscach, "Graphical control of a parametric equalizer," in *Audio Engineering Society Convention 124*. Audio Engineering Society, 2008.

[8] Philipp Kolhoff, Jacqueline Preub, and Jorn Loviscach, "Music icons: procedural glyphs for audio files," in *2006 19th Brazilian Symposium on Computer Graphics and Image Processing*. IEEE, 2006, pp. 289–296.

[9] Mark Brozier Cartwright and Bryan Pardo, "Social-eq: Crowdsourcing an equalization descriptor map.," in *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR)*, 2013.

[10] Thomas Wilmering, György Fazekas, and Mark B Sandler, "High-level semantic metadata for the control of multitrack adaptive digital audio effects," in *Audio Engineering Society Convention 133*. Audio Engineering Society, 2012.

[11] Ryan Sarver and Anssi Klapuri, "Application of nonnegative matrix factorization to signal-adaptive audio effects," in *Proc. DAFx*, 2011, pp. 249–252.

[12] Joerg Bitzer, Denny Schmidt, and Uwe Simmer, "Parameter estimation of dynamic range compressors: models, procedures and test signals," in *Audio Engineering Society Convention 120*. Audio Engineering Society, 2006.

[13] Daniele Barchiesi and Joshua Reiss, "Reverse engineering of a mix," *Journal of the Audio Engineering Society*, vol. 58, no. 7/8, pp. 563–576, 2010.

[14] Drossos K. Virtanen T. Mimilakis, S.I. and G. Schuller, "Deep neural networks for dynamic range compression in mastering applications," in *Audio Engineering Society Convention 140*. Audio Engineering Society, 2016.

[15] Jean Julien Aucouturier and Francois Pachet, "Music similarity measures: What's the use?," in *Proceedings of the 3th International Conference on Music Information Retrieval (ISMIR)*, 2002, pp. 13–17.

[16] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[17] Beth Logan and Ariel Salomon, "A music similarity function based on signal analysis.," in *ICME*, 2001.

[18] John R Hershey and Peder A Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. IEEE, 2007, vol. 4, pp. IV–317.

[19] Ryan Stables, Sean Enderby, BD Man, György Fazekas, Joshua D Reiss, et al., "Safe: A system for the extraction and retrieval of semantic audio descriptors," *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*.

[20] Udo Zolzer, *DAFX: Digital Audio Effects*, Wiley Publishing, 2nd edition, 2011.

[21] Geoffroy Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," 2004.

[22] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka, "Rwc music database: Music genre database and musical instrument sound database," *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pp. 229–230, 2003.

[23] Leo Breiman, "Random forests," *Journal of Machine Learning Research*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] Danilo P Mandic and Vanessa Su Lee Goh, *Complex valued nonlinear adaptive filters: noncircularity, widely linear and neural models*, vol. 59, John Wiley & Sons, 2009.

[26] Xavier Serra and Julius Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," vol. 14, no. 4, pp. 12–24, 1990.

[27] Sebastian Heise, Michael Hlatky, and Jörn Loviscach, "A computer-aided audio effect setup procedure for untrained users," in *Audio Engineering Society Convention 128*. Audio Engineering Society, 2010.

[28] Roger B Dannenberg, "Music representation issues, techniques, and systems," in *Computer Music Journal*. 1993, vol. 17, pp. 20–30, JSTOR.

[29] Olivier Lartillot, Petri Toiviainen, and Tuomas Eerola, "A matlab toolbox for music information retrieval," in *Data analysis, machine learning and applications*, pp. 261–268. Springer, 2008.

[30] Petri Toiviainen and Carol L Krumhansl, "Measuring and modeling real-time responses to music: The dynamics of tonality induction," vol. 32, no. 6, pp. 741–766, 2003.