JOINT ESTIMATION OF FADER AND EQUALIZER GAINS OF DJ MIXERS USING CONVEX OPTIMIZATION

Taejun Kim
Graduate School of Culture Technology
KAIST
Daejeon, South Korea
taejun@kaist.ac.kr

Yi-Hsuan Yang
Research Center for IT Innovation
Academia Sincia
Taipei, Taiwan
yang@citi.sinica.edu.tw

Juhan Nam
Graduate School of Culture Technology
KAIST
Daejeon, South Korea
juhan.nam@kaist.ac.kr

ABSTRACT

Disc jockeys (DJs) use audio effects to make a smooth transition from one song to another. There have been attempts to computationally analyze the creative process of seamless mixing. However, only a few studies estimated fader or equalizer (EQ) gains controlled by DJs. In this study, we propose a method that jointly estimates time-varying fader and EQ gains so as to reproduce the mix from individual source tracks. The method approximates the equalizer filters with a linear combination of a fixed equalizer filter and a constant gain to convert the joint estimation into a convex optimization problem. For the experiment, we collected a new DJ mix dataset that consists of 5,040 real-world DJ mixes with 50,742 transitions, and evaluated the proposed method with a mix reconstruction error. The result shows that the proposed method estimates the time-varying fader and equalizer gains more accurately than existing methods and simple baselines.

1. INTRODUCTION

DJs select musical tracks and play them without stopping music to enhance listening experiences. To make music flow continuous, DJs make a seamless transition from one track to another using various mixing techniques. The techniques can be summarized as three steps: 1) beat matching that synchronizes tempo in BPM (beats per minute) and beat positions of the previous track and the next track, 2) cueing that decides the positions to stop the previous track and start the next track, and 3) applying audio effects such as crossfader or equalizer.

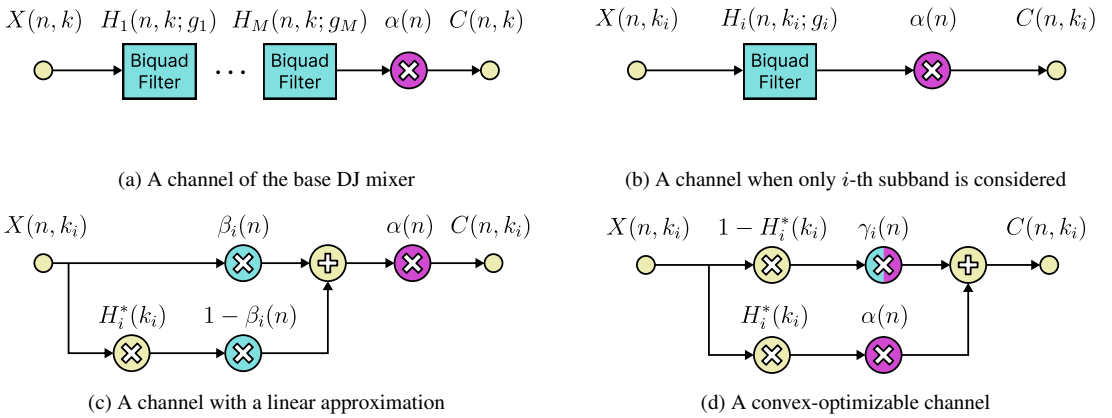
Analyzing the creative process using computational methods has drawn research interests. For example, mix-to-track alignment is a task that temporally synchronizes original tracks with the DJ mix. This enabled to find various features controlled by DJs such as cue points, transition length, tempo change, and key transpose [1]. In mix-to-track alignment, beat tracking was used as a crucial preprocessing step to summarize audio features in beat unit [2]. The cue points were also estimated with different methods and audio features [3, 4]. However, analyzing how DJs utilize audio effects has not yet been explored much.

There are a number of audio effects in DJ mixers such as fader, crossfader, equalizer, delay, and reverberation. DJs usually use multiple audio effects simultaneously to make seamless transitions. For example, they can increase the fader gain of one track while decreasing the low shelving filter (or low EQ) for the next track at the same time. Therefore, analysis of DJ mixer control requires estimating multiple audio effect gains simultaneously. While different brands of DJ mixers have different combinations of audio effects, they have faders and equalizers in common as they are the primary audio effects for DJing.

There are a few prior works that estimated the fader or EQ gains when a DJ mix and two source tracks are given. The two source tracks are the original tracks without any modification by the audio effects. One group of the study focused on estimating the fader gains in the track transition region [5, 6]. However, they used artificially generated datasets and did not consider EQ gains. In our previous work [7], we attempted to estimate the EQ gain curves using convex optimization using a real-world DJ mix dataset. However, we did not consider the simulatenous control of fader and EQ gain and the frequency responses of EQ filters in the EQ gain estimation. Nercessian [8] recently tackled EQ parameter estimation using a differentiable biquad filter in the EQ matching problem. They showed that the neural network approach outperforms a convex relaxation of the problem. However, they focused on estimating the fixed coefficients of the biquad EQ filters and did not consider time-varying control of the parameters. Furthermore, it makes the estimation more challenging when two input sources are involved as in the DJ mix. Recently, a method that generates fader and EQ gains has been proposed [9]. It uses differentiable faders and EQs with generative adversarial networks but the purpose is automatic mixing, not analyzing existing DJ mixes. Smooth transition between music tracks was also studied in the context of spatial audio effect using Head Related Transfer Function (HRTF) [10]. However, it was not for DJ mixing but for headphone listening.

In this study, we propose a method that can jointly estimate the fader and EQ gains of DJ mixers. The proposed method is designed to consider the frequency response of EQs and relax the joint estimation into a convex optimization problem. Although deep neural networks have become increasingly popular, we chose convex optimization instead of deep neural networks for its advantages such as better interpretability, flexibility to strong constraints, and optimization speed. For the experiment on a larger dataset, we collected a new DJ mix dataset that consists of 5,040 real-world DJ mixes with 50,742 transitions. We optimized and evaluated the proposed method with a mix reconstruction error. The result shows that our proposed method estimates the time-varying fader and equalizer gains more accurately than baselines and previous work. The code and dataset will be available at this link.[1]

[1] https://github.com/mir-aidj/djmixer-estimation



(a) A channel of the base DJ mixer

(b) A channel when only i-th subband is considered

(c) A channel with a linear approximation

(d) A convex-optimizable channel

Figure 1: A procedure that makes a EQ-Fader channel convex optimizable. Components involving parameters to be optimized are colored: *cyan* for EQ gains and *magenta* for fader gains.

2. DJ MIXER DESIGN

In this section, we define a transfer function for each channel of three types of DJ mixers. Then, we define the final mixed output of DJ mixers which have two input channels.

2.1. Fader

A transfer function of a fader channel H_{fd} parameterized by a gain α is defined as:

$$H_{\text{fd}}(n,k;\alpha) = \alpha(n) \tag{1}$$

where n is the time (or frame) index and k is the frequency bin index. Then, the output signal of the fader is written as:

$$C_{\text{fd}}(n,k;\alpha) = \alpha(n)X(n,k) \tag{2}$$

where X is the spectrogram of an audio track.

2.2. Equalizer (EQ)

An EQ channel consists of M cascaded peak and shelving filters implemented using biquad filters [11]. The transfer function of i-th biquad filter attenuating i-th subband is given as

$$H_i(n,k;g_i) = \frac{b_0^i(n) + b_1^i(n)z^{-1} + b_2^i(n)z^{-2}}{a_0^i(n) + a_1^i(n)z^{-1} + a_2^i(n)z^{-2}} \tag{3}$$

where $i = 1,\ldots,M$. Note that the filters coefficients are controlled by a time-varying gain g_i in decibel (dB). Accordingly, H_i also has a time-varying equalizing curve which is parameterized by g_i. The number and parameters of filters such as cutoff/center frequency and quality factor (Q) are different depending on the specification of DJ mixers. However, we assume that the DJ mixers used in our experiment have the same fixed settings except the gains to focus on estimating the gain parameter. Finally, the transfer function of the equalizer channel can be written as:

$$H_{\text{eq}}(n,k;g) = \prod_{i=1}^{M} H_i(n,k;g_i) \tag{4}$$

where $g = \{g_1,\ldots,g_M\}$ and the channel output is defined as:

$$C_{\text{eq}}(n,k;g) = H_{\text{eq}}(n,k;g)X(n,k). \tag{5}$$

2.3. EQ-Fader

An EQ-fader channel is simply a series of equalizers and faders as shown in Figure 1(a). The transfer function is a multiplication of the equalizer and fader transfer functions:

$$H_{\text{ef}}(n,k;\alpha,g) = H_{\text{fd}}(n,k)H_{\text{eq}}(n,k) \tag{6}$$

$$= \alpha(n)\prod_{i=1}^{M} H_i(n,k;g_i). \tag{7}$$

Then, the channel output of an EQ-fader mixer is written as:

$$C_{\text{ef}}(n,k;\alpha,g) = H_{\text{ef}}(n,k)X(n,k). \tag{8}$$

2.4. Mixer

The output signal of a DJ mixer Y in a transition region is simply a summation of two channels:

$$Y(n,k) = C^{\text{prev}}(n,k) + C^{\text{next}}(n,k) \tag{9}$$

where C^{prev} is a channel that plays the previous track in a transition region which fades away over time and C^{next} is a channel of the next track.

3. DJ MIXER GAIN ESTIMATION

DJ mixer gain estimation algorithms take three audio clips in a transition region as inputs: a previous track, a next track and a DJ mix. All the three input clips are aligned and time-scale modification is applied to them beforehand so that beats and time offsets between the tracks and the mix are exactly synchronized. Given the three clips, gain estimation algorithms derive the time-varying gain curves of faders and equalizers.

An objective function for the DJ mixer gain estimation can be defined using mean absolute errors (MAEs) as below:

$$\frac{1}{N}\sum_{n=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\left|\hat{Y}(n,k;\theta) - Y(n,k)\right|. \tag{10}$$

where Y is a spectrogram of the ground-truth DJ mix, $\hat{Y}$ is an estimated spectrogram, θ is any parameters to be optimized and

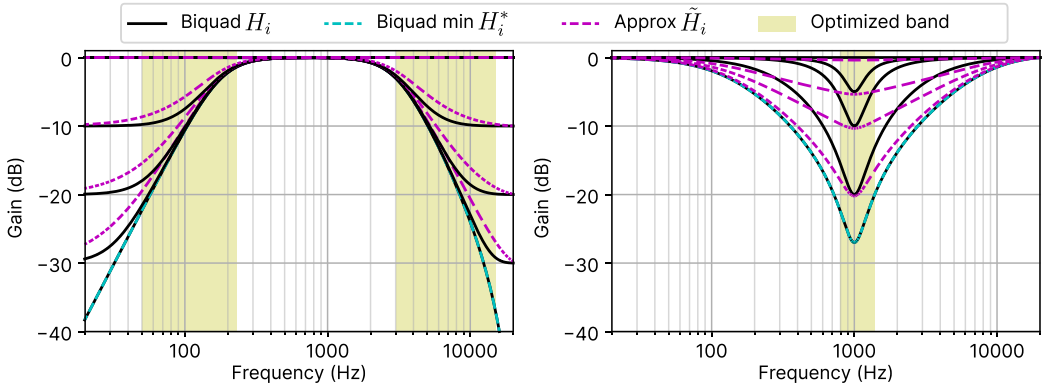


Figure 2: Gain differences in frequency responses between the actual EQ filters and the linearly approximated filters.

used for estimating $\hat{Y}$, N is the number of time frames, and K is the total number of frequency bins.

We could have used spectrograms in the decibel (dB) scale to emphasize small signal differences and to make it closer to the human perception of the intensity of sound. However, if we take the log on both sides of Eq. (9) to obtain dB-scaled spectrograms, the objective function becomes non-convex. We also considered the Log-Sum-Exp (LSE) trick, but the objective function based on LSE is still non-convex when MAE is used (it is non-convex even if mean squared error is used).

It is straightforward to apply this objective function to estimate gains of a mixer with two fader channels using convex optimization. However, the objective function cannot be used for mixers including equalizers because their transfer functions are not convex. The later part of this section explains a procedure that converts the transfer functions of EQ and EQ-Fader mixers to a convex form as illustrated in Figure 1.

3.1. Subband objective functions for EQ gain estimation

If we assume that an EQ filter does not affect other subbands, an EQ channel (Eq. (5)) for a specific i-th subband can be approximated as shown in Figure 1(b):

$$C_{\text{eq}}(n,k;g_i)|_{k\in\mathbb{K}_i} \approx H_i(n,k;g_i)X(n,k) \tag{11}$$

where $\mathbb{K}_i$ is a set of frequency bin indices of i-th subband. It removes the products of the EQ transfer functions in Eq. (4). Then, the MAEs can be computed for each subband and summed:

$$\mathcal{L}(\theta;\hat{Y},Y) \tag{12}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{M}\frac{1}{|\mathbb{K}_i|}\sum_{k_i\in\mathbb{K}_i}\left|\hat{Y}(n,k_i;\theta) - Y(n,k_i)\right| \tag{13}$$

where k_i is a frequency bin index in $\mathbb{K}_i$. However, due to H_i (Eq. (3)), it is still not a convex function.

3.2. Linear approximation of EQ filters

To relax the subband objective function into a convex form, we approximate the EQ transfer function H_i in a linear form which is parameterized by $\beta_i \in [0,1]$. β_i is a parameter that controls the ratio between the original signal and a fully filtered by an EQ filter as shown in Figure 1(c). It is formally written as:

$$H_i(n,k;g_i) \approx \tilde{H}_i(n,k;\beta_i) \tag{14}$$

$$= \beta_i(n) + (1-\beta_i(n))H_i^*(k) \tag{15}$$

where H_i^* is a fixed transfer function when H_i has a minimum gain g_i^*:

$$H_i^*(k) = H_i(n,k)|_{g(n)=g_i^*}. \tag{16}$$

A setting of a minimum gain for an EQ filter is different depending on DJ mixers but we assume all DJ mixers have the same g_i^*. β_i is related with g_i by:

$$\beta_i(n) = 10^{g(n)/20} - 10^{g_i^*/20} \tag{17}$$

Figure 2 illustrates H_i^* and compares the gain differences in frequency response between the actual biquad EQ filters and approximated filters. The left subplot shows the differences in low and high shelving filters, and the right subplot shows the differences in mid-peak filter. The three filters are used for our experiments. There are gaps between the actual and the approximated filters. Especially, the gaps are larger in mid-peak filters. To minimize optimization errors from the gaps, we limit the subband frequency range to the region colored in yellow. In other words, we choose a smaller number of frequency bins for $\mathbb{K}_i$.

3.3. Parameter substitutions for EQ-Fader channel

With the subband objective function and the linear approximation, convex optimization can be performed on a mixer with two EQ channels. However, the transfer function of EQ-Fader channel, H_{ef}, is approximated as:

$$\tilde{H}_{\text{ef}}(n,k_i;\alpha,\beta_i) = \alpha(n)\tilde{H}_i(n,k;\beta_i) \tag{18}$$

$$= \alpha(n)(\beta_i(n) + (1-\beta_i(n))H_i^*(k)) \tag{19}$$

$$= \alpha(n)H_i^*(k) + \alpha(n)\beta(n)(1-H_i^*(k)) \tag{20}$$

which is illustrated in Figure 1(c). However, it is impossible to apply convex optimization directly on the EQ-Fader channel due to the multiplication of α and β_i in Eq. (20). Therefore, we substitute $\alpha(n)\beta_i(n)$ as:

$$\gamma_i(n) = \alpha(n)\beta_i(n) \tag{21}$$

The number of mixes	5,040
The number of transitions	50,742
The number of unique tracks	52,910
The number of played tracks	67,514
The total length of mixes (in hours)	6,672

Table 1: Summary statistics of the *MixesDB* dataset.

Then, $\tilde{H}_{\text{ef}}$ becomes

$$\tilde{H}_{\text{ef}}(n, k_i; \alpha, \gamma_i) = \alpha(n) H_i^*(k) + \gamma(n)(1 - H_i^*(k)) \quad (22)$$

which is optimizable as a convex form and parameterized by α and γ_i.

3.4. Constraints

In most cases, DJs make a previous track fade out and a next track fade in in a transition region. From this observation, we assume that any fader or EQ gains of a previous track always decrease and gains of a next track always increase. This assumption is first introduced at [7] and gives the following constraints to optimization problems:

$$\Delta\alpha^{\text{prev}}(n) \leq 0 \qquad \Delta\alpha^{\text{next}}(n) \geq 0 \quad (23)$$

$$\Delta\beta_i^{\text{prev}}(n) \leq 0 \qquad \Delta\beta_i^{\text{next}}(n) \geq 0 \quad (24)$$

where Δ is a finite difference, i.e. $\Delta f(n) = f(n+1) - f(n)$. Also, we assume that a fader can amplify an input signal upto twice and set β_i to have a range of $[0, 1]$ by adding constraints below:

$$0 \leq \alpha(n) \leq 2 \qquad 0 \leq \beta_i(n) \leq 1 \quad (25)$$

3.5. Convex optimization problem

It is straightforward to establish a convex optimization problem for the fader and EQ mixers using the objective functions and the constraints. However, in case of EQ-Fader mixers, the constraints should be transformed into a domain of α and γ_i which gives

$$\underset{\alpha,\gamma}{\text{minimize}} \quad \mathcal{L}(\alpha, \gamma; \hat{Y}, Y) \quad (26)$$

$$\text{subject to} \quad 0 \leq \alpha(n) \leq 2 \quad (27)$$

$$0 \leq \gamma_i(n) \leq \alpha(n) \quad (28)$$

$$\Delta\alpha^{prev}(n) \leq 0 \quad (29)$$

$$\Delta\alpha^{next}(n) \geq 0 \quad (30)$$

$$\Delta\gamma_i^{prev}(n) \leq \Delta\alpha^{prev}(n) \quad (31)$$

$$\Delta\gamma_i^{next}(n) \geq \Delta\alpha^{next}(n) \quad (32)$$

where $\gamma = \{\gamma_1, \ldots, \gamma_M\}$. The detail of deriving the constraints is provided in the Appendix section.

4. EXPERIMENTAL SETUPS

4.1. Implementation Details

Input previous/next tracks and DJ mixes are aligned using mix-to-track subsequence alignment [1][2] that uses dynamic time warping (DTW) on audio features. After the alignment, time scale modification is applied to previous and next tracks. We used an implementation of waveform-similarity overlap-add (WSOLA) [12] in PyTSMod [13][3] for the time scaled modification. To efficiently run optimization, audio signals are clipped so that they only include parts where overlapping beats between the previous and the next tracks exist. We used CVXPY [14] and ECOS [15] to implement convex optimization.

All audio signals have a sampling rate of 44,100Hz. For X and Y, we used constant-Q transform (CQT) and mel spectrograms and computed them using Librosa [16]. Note that a direct STFT cannot be used for the optimization since it has too many frequency bins which may lead to failure of the optimization. Settings of mel spectrograms and CQT are chosen such that they are fairly compared. The hop size is set to be 4,096 samples (93 ms) and the frequency ranges are limited to $[50, 15000]$ Hz for both of mel spectrogram and CQT. We computed STFT with an 8,192-point FFT for mel spectrogram and the lengths of constant-Q filters are set to double for a better frequency resolution. The number of bins per octave of CQT is fixed to 12 which yields 100 frequency bins and thus the number of bins of mel spectrograms is set to be 100 as well.

We set the parameters of equalizers following the specification of a DJ mixer, Xone:96 by Allen & Heath. [4] The EQs consist of three subband filters: low shelving, mid-peak and high shelving filters. We implemented the EQ filters following the formulae in the link[5] with SciPy [17]. The low and high shelving filters have a quality factor (Q) of $1/\sqrt{2}$ and cutoff frequencies of 180 and 3,000 Hz, respectively. The mid-peak filter has a Q of 3 and a center frequency of 1,000 Hz. The minimum gain g_i^* is set to be -80, -27 and -80 dB for low, mid and high, respectively. The optimiza-

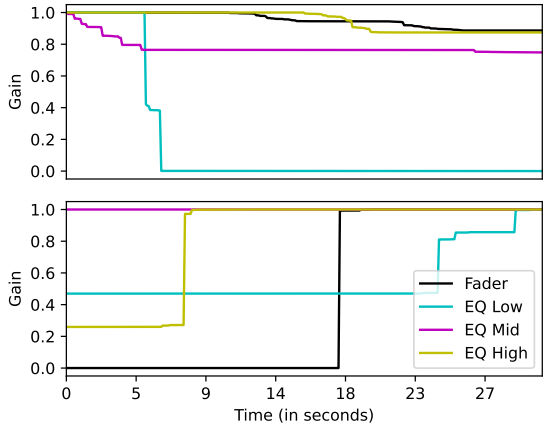


Figure 3: An example of generated gain curves for the synthesized dataset. (Top) Curves for a previous track. (Bottom) Curves for a next track.

[2] https://github.com/mir-aidj/djmix-analysis

[3] https://pytsmod.readthedocs.io

[4] Xone:96 has four EQs but we use three of them for simplicity. We selected Xone:96 because it was the only mixer that we could find the EQ frequency responses from the product manual. Some other mixer manuals have EQ cutoff/center frequencies, but they do not have frequency response curves, which makes it difficult to set EQ coefficients other than gains.

[5] https://webaudio.github.io/Audio-EQ-Cookbook/audio-eq-cookbook.html

tion frequency ranges for low, mid and high subbands are set to be $[50, 180]$, $[800, 1400]$ and $[3000, 15000]$ Hz, respectively, as shown in Figure 2.

4.2. Dataset

We collected the dataset from *MixesDB*[6]. The website is a database for DJ mixes that are added by users, and contains metadata and links to audios of DJ mixes and links to tracks played in the mixes. We processed the data so that they can be used for research purposes. Table 1 shows summary statistics of the dataset used for experiments. Note that the numbers of played and unique tracks are different because a track can be played in multiple mixes. However, the metadata of MixesDB dataset contain incorrectly annotated data by its nature. For example, a wrong track different from a track that a DJ actually played can be listed in tracklists. Also, since a single track often has various versions in dance music, for example, short or long versions and a remix, an annotated track in a mix can be a different version of a track that a DJ played. We filtered out the incorrect tracks based on the method in our previous work [1] but still some errors remain.

Therefore, we evaluate the models on a synthesized dataset as well to remove errors coming from the incorrect metadata. A thousand of transitions with the length of 64 beats (about 30–32 seconds) are randomly generated using tracks in MixesDB. A transition is a mix of two tracks: randomly selected previous and next tracks. Randomly generated gain curves for fader and EQ are applied to the tracks. Figure 3 shows an example of a set of generated random curves for a transition. Considering the practice in real-world DJing, we set that only a single audio effect gain can be changed in four seconds for each channel with two hands. EQ settings are the same as those in the proposed method. The gain curves of previous/next tracks monotonically decrease or increase.

4.3. Evaluation Metrics

We evaluated the models quantitatively from two perspectives. Each of them was motivated by the following questions: 1) How much does the reconstructed audio sound similar to the DJ mix? 2) How much are the extracted gain curves similar to the actual curves? Spectrogram reconstruction errors and gain estimation errors are used for the first and the second questions, respectively. All the errors are computed using MAE. As MixesDB does not have ground truth of gain curves that represent how DJs controlled the DJ mixers on performances, we report gain estimation errors only on the synthesized dataset. We computed the MAEs for 50,742 transitions in MixesDB and 1,000 synthesized transitions.

MAEs between a DJ mix and a reconstructed mix are used for the spectrogram reconstruction errors. The detailed procedure of the measuring is as follows:

1. Run a DJ mixer gain estimation algorithm and obtain gain curves.
2. Reconstruct a mix of two tracks applying extracted fader and EQ gains on time domain.
3. Extract two spectrograms of the reconstructed signal and a DJ mix.
4. Compute a mean absolute error (MAE) between the two spectrograms.

[6] `https://www.mixesdb.com`

Settings of spectrograms for evaluation are the same as those for optimization explained in subsection 4.1. As a low frequency band tends to have higher magnitudes than mid and high frequency bands, we also compute MAEs for the three subbands separately. Frequency ranges for the subbands are set to $[50, 180]$, $[180, 3000]$ and $[3000, 15000]$ Hz for low, mid and high subbands, respectively.

To measure the gain estimation errors, we computed absolute differences between the ground truth curves and the estimated gain curves and averaged them over time. We report the MAEs for EQ low, EQ mid, EQ high, and fader individually, and for all as well. However, when the errors are reported all together on faders and EQs, we used a weighted sum to give equal importance between faders and EQs. More specifically, we used weights of $1/6$, $1/6$, $1/6$, and $1/2$ to the errors of EQ low, EQ mid, EQ high, and fader, respectively. If a model estimates only a subset among the four audio effects, we assume that the model does not use other audio effects at all. Thus, gain curve MAEs are computed between the ground truth gain curves and a constant gain of one. For example, if a model only estimates fader gains, then the errors for three EQs are computed using a fixed gain of one because the model has an assumption that EQs are never used.

4.4. Models

We evaluate the proposed method using quantitative results compared to two simple baselines and two previous methods. The two baselines do not involve any optimization; they mix a previous track and a next track using the following simple rules:

- **Sum**: simply adds two tracks.
- **Crossfade**: applies a constant power crossfade linearly from the previous to the next track.

The models of the previous methods are as below:

- **Fader** [5]: runs convex optimization on a mixer that consists of two fader channels in Eq. (2) using the objective function in Eq. (10).
- **Subband-Fader** [7]: runs convex optimization on a mixer that has a fader for each subband using the subband objective function in Eq. (13). It also has three subbands and ranges of the subbands are same with the proposed methods.

Finally, the proposed methods include:

- **EQ3**: runs convex optimization on a mixer that consists of two EQ channels in Eq. (5) using the subband objective function in Eq. (13). The EQ channel has three EQ filters: low shelving, mid-peak and high shelving filters.
- **EQ3-Fader**: runs convex optimization on a mixer that consists of two EQ-Fader channels in Eq. (8) using the subband objective function in Eq. (13). The number and settings of EQ filters are same with EQ3.

All the methods involving convex optimization have constraints described in subsection 3.4. To find the best performing input representation for the models, we also compare CQT and mel spectrograms performances when they used as $X(n, k)$. For a fair comparison for the both of spectrograms, we report MAEs using both of the spectrograms. In other words, both of CQT and mel spectrograms are used as inputs for the estimation and also used for computing MAEs.

		CQT MAE				Mel MAE			
$X(n,k)$	Model	All	Low	Mid	High	All	Low	Mid	High
				MixesDB					
–	Sum	8.33	8.22	8.25	8.54	5.91	5.61	5.71	6.47
–	Crossfade	8.14	7.90	8.12	8.38	5.76	5.46	5.58	6.31
CQT	Fader [5]	7.68	**6.57**	7.94	8.14	5.70	5.29	5.57	6.23
	Subband-Fader [7]	7.53	7.07	7.56	7.85	5.04	4.70	4.68	5.92
	EQ3	7.46	7.03	7.38	7.93	5.13	4.53	4.46	6.73
	EQ3-Fader	7.21	6.65	7.22	7.63	4.83	**4.39**	4.40	5.90
Mel	Fader [5]	7.45	7.09	7.15	7.84	5.14	4.71	4.95	5.41
	Subband-Fader [7]	7.56	7.27	7.37	7.80	4.83	4.96	4.70	4.96
	EQ3	7.40	7.15	7.22	7.64	4.62	4.86	4.41	4.84
	EQ3-Fader	**7.12**	6.88	**6.91**	**7.39**	**4.24**	4.55	**4.14**	**4.33**
				Synthesized Dataset					
–	Sum	6.60	7.58	6.19	6.51	5.87	5.40	5.53	6.79
–	Crossfade	5.64	6.48	5.49	5.23	4.79	4.66	4.52	5.36
CQT	Fader [5]	3.38	2.10	3.58	4.06	3.35	2.79	3.07	4.24
	Subband-Fader [7]	1.61	2.06	1.27	1.81	1.17	1.01	0.87	1.81
	EQ3	2.81	3.67	2.43	2.74	2.10	2.08	1.77	2.67
	EQ3-Fader	**1.26**	**1.44**	**1.07**	**1.43**	**0.98**	**0.84**	**0.80**	**1.36**
Mel	Fader [5]	2.71	4.60	2.53	2.76	2.14	2.22	1.53	2.84
	Subband-Fader [7]	1.86	3.16	1.84	1.76	1.30	1.80	0.88	1.76
	EQ3	3.02	4.15	3.07	2.86	2.28	3.07	1.72	2.87
	EQ3-Fader	1.58	2.33	1.52	1.58	1.21	1.32	0.95	1.50

Table 2: Comparison of DJ mixer gain estimation methods: mean absolute errors (MAEs) on CQT and mel spectrogram.

5. EXPERIMENTAL RESULTS

5.1. Spectrogram Reconstruction Errors

Table 2 reports the spectrogram reconstruction errors of the six estimation models. In general, the proposed model EQ3-Fader shows the best performance regardless of input type, metrics, and datasets. The Fader model shows competitive results for low band MAEs in some cases, however, it tends to show inferior results on mid and high MAEs. We interpret this that the Fader model focuses on optimizing errors from the low band because the low band tends to have higher magnitudes. There are higher performance gaps between the baselines and the optimization involved methods in the synthesized dataset because the synthesized transitions have lower energy in the middle of transitions whereas real-world DJ mixes have relatively constant energy through transitions.

The overall errors are lower in the synthesized dataset than MixesDB because there are no errors from incorrect mix-to-track alignment, beat tracking errors, wrong annotations of tracks, and importantly, different settings of EQs. The synthesized dataset was generated using the same EQ parameters such as Q, cutoff or center frequency, and gain range, whereas MixesDB includes mix tracks rendered with a wide variety of DJ mixer models with different settings of EQ parameters. The proposed method has a limitation in that it assumes a fixed set of EQ parameters without considering the diversity. However, it would be possible to find the best-matching EQ parameters among a set of known EQ parameters from DJ mixer models.

5.2. Gain Estimation Errors

The gain estimation MAEs on the synthesized dataset are summarized in Table 3. The proposed EQ3-Fader model is the best-performing model in all metrics. Subband-Fader and EQ3 models show inferior results generally. They focus on EQ gain estimation but they even show degraded performances for EQ low, EQ mid, and EQ high gains. This is because they try to estimate EQ gains even though only a fader gain is changed and EQ gains are not changed. For example, if a fader gain decreases, the models will estimate that three EQ gains are decreased.

5.3. Visualization of DJ mixer gain estimation

Figure 4 visualizes an example of estimated gain curves extracted using an EQ3-Fader model with CQT on MixesDB. The mel spectrograms show better quatitative results, however, CQT is used for visualization as they show three subbands evenly whereas mel spectrograms allocate many frequency bins for the mid subband. From top to bottom, the spectrograms represent 1) an original previous track (without any audio effects), 2) a previous track filtered with estimated gains, 3) an original DJ mix, 4) a mix reconstructed using estimated gains, 5) an original next track and 6) a next track filtered with estimated gains. The DJ mix and the estimated mix show similar spectrograms. The estimated next track at the bottom shows that the algorithm estimates that the DJ increased the low shelving filter gain around at 1 min and 33 secs.

$X(n,k)$	Model	All	Fader	EQ Low	EQ Mid	EQ High
–	Sum	0.230	0.237	0.253	0.182	0.231
–	Crossfade	0.270	0.318	0.253	0.182	0.231
CQT	Fader [5]	0.229	0.236	0.253	0.182	0.231
	Subband-Fader [7]	0.261	0.237	0.279	0.275	0.303
	EQ3	0.288	0.237	0.349	0.293	0.377
	EQ3-Fader	**0.141**	**0.104**	0.177	**0.160**	**0.199**
Mel	Fader [5]	0.221	0.220	0.253	0.182	0.231
	Subband-Fader [7]	0.275	0.237	0.314	0.298	0.326
	EQ3	0.302	0.237	0.364	0.321	0.412
	EQ3-Fader	0.161	0.137	**0.175**	0.169	0.212

Table 3: Mean absolute errors (MAEs) between the ground truth and estimated gain curves on the synthesized dataset.

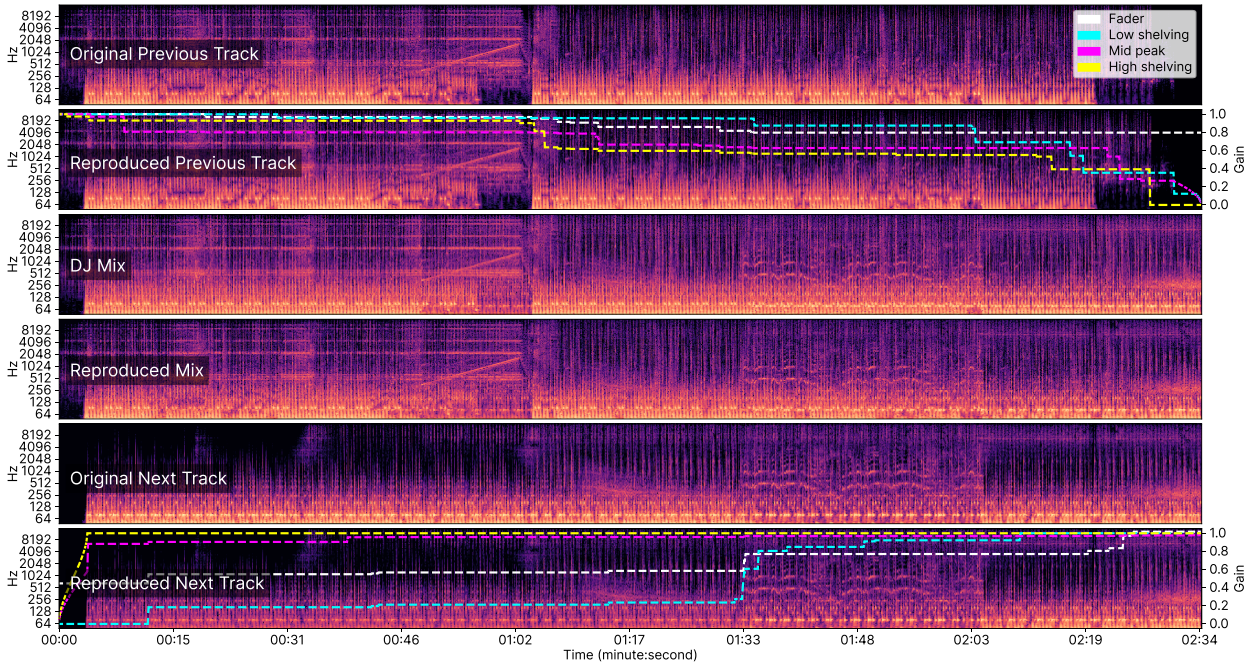


Figure 4: A example visualization of DJ mixer gain estimation using EQ3-Fader model with CQT from the MixesDB dataset.

6. CONCLUSIONS

We proposed an estimation algorithm that can estimate time-varying fader and EQ gains jointly in DJ music. To make a series of EQ filters and the fader optimizable in a convex form, we linearly approximated the transfer functions of EQ, introduced the subband objective function and substituted the parameters. We evaluated the proposed method on a real-world DJ music dataset and showed that the proposed method estimates the fader and EQ gains more precisely than compared methods.

There are a few limitations in this work: 1) the optimization is insensitive to signals with small magnitudes since it is optimized in a linear domain instead of a log domain and 2) the biquad coefficients are fixed except for gains, which can lead to degraded performance if the coefficients are too different from the actual ones. As future work, we plan to use neural networks with differentiable audio effects. For example, we might be able to translate the constraints in our formulation into regularization terms in training a neural network.

7. ACKNOWLEDGEMENT

This research is supported Year 2022 Copyright Technology R&D Program by Ministry of Culture, Sports and Tourism and Korea Copyright Commission (Project Name: Development of high-speed music search technology using deep learning, Project Number: 2021-hs-9500).

8. REFERENCES

[1] Taejun Kim, Minsuk Choi, Evan Sacks, Yi-Hsuan Yang, and Juhan Nam, "A computational analysis of real-world DJ mixes using mix-to-track subsequence alignment," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 764–770.

[2] Sebastian Böck, Florian Krebs, and Gerhard Widmer, "Joint beat and downbeat tracking with recurrent neural networks.," in *International Society for Music Information Retrieval Conference (ISMIR)*. New York City, 2016, pp. 255–261.

[3] Reinhard Sonnleitner, Andreas Arzt, and Gerhard Widmer, "Landmark-based audio fingerprinting for DJ mix monitoring.," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 185–191.

[4] Tim Scarfe, W Koolen, and Yuri Kalnishkan, "Segmentation of electronic dance music," *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 22, no. 3, pp. 4, 2014, Soure code available at [Online] `https://github.com/ecsplendid/DanceMusicSegmentation`.

[5] Lorin Werthen-Brabants, "Ground truth extraction & transition analysis of DJ mixes," M.S. thesis, Ghent University, Ghent, Belgium, 2018.

[6] Diemo Schwarz and Dominique Fourer, "Methods and datasets for DJ-mix reverse engineering," in *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2019, pp. 426–437.

[7] Taejun Kim, Yi-Hsuan Yang, and Juhan Nam, "Reverse-engineering the transition regions of real-world dj mixes using sub-band analysis with convex optimization," in *New Interfaces for Musical Expression (NIME)*, 2021.

[8] Shahan Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *International Conference on Digital Audio Effects (DAFx)*, 2020.

[9] Bo-Yu Chen, Wei-Han Hsu, Wei-Hsiang Liao, Marco A. Martínez Ramírez, Yuki Mitsufuji, and Yi-Hsuan Yang, "Automatic DJ transitions with differentiable audio effects and generative adversarial networks," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[10] Aki Härmä and Steven Van De Par, "Spatial track transition effects for headphone listening," in *International Conference on Digital Audio Effects (DAFx)*, 2007.

[11] Udo Zölzer, Xavier Amatriain, Daniel Arfib, Jordi Bonada, Giovanni De Poli, Pierre Dutilleux, Gianpaolo Evangelista, Florian Keiler, Alex Loscos, Davide Rocchesso, et al., *DAFX-Digital audio effects*, John Wiley & Sons, 2002.

[12] Werner Verhelst and Marc Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1993, vol. 2, pp. 554–557.

[13] Sangeon Yong, Soonbeom Choi, and Juhan Nam, "PyTSMod: A python implementation of time-scale modification algorithm," in *International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Paper*, 2020.

[14] Steven Diamond and Stephen Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[15] Alexander Domahidi, Eric Chu, and Stephen Boyd, "ECOS: An SOCP solver for embedded systems," in *European Control Conference (ECC)*. IEEE, 2013, pp. 3071–3076.

[16] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and music signal analysis in python," in *Python in Science Conference*, 2015, vol. 8, pp. 18–25.

[17] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

9. APPENDIX: PROOFS FOR THE CONSTRAINTS

Before the substitution of the problem in subsection 3.5, the original convex optimization problem is given as

$$\underset{\alpha,\beta}{\text{minimize}} \quad \mathcal{L}(\alpha, \beta; \hat{Y}, Y) \tag{33}$$

$$\text{subject to} \quad 0 \leq \alpha(n) \leq 2 \tag{34}$$

$$0 \leq \beta_i(n) \leq 1 \tag{35}$$

$$\Delta\alpha^{\text{prev}}(n) \leq 0 \tag{36}$$

$$\Delta\alpha^{\text{next}}(n) \geq 0 \tag{37}$$

$$\Delta\beta_i^{\text{prev}}(n) \leq 0 \tag{38}$$

$$\Delta\beta_i^{\text{next}}(n) \geq 0 \tag{39}$$

where $\beta = \{\beta_1, \ldots, \beta_M\}$. It is straightforward to transform the four constraints from the top into a domain of γ_i so we only provide a induction of the last two constraints. Only the induction of parameters of a previous track is provided since it is same for a next track. Due to the substitution,

$$\beta_i = \frac{\gamma_i}{\alpha} \tag{40}$$

is true. An intuitive way of understanding Eq. (31) is that β_i^{prev} will always decrease if a decreasing speed of γ_i^{prev} is slower than α^{prev}. Formally speaking, $\Delta\beta_i^{\text{prev}}(n) \leq 0$ is transformed into

$$\frac{\Delta\gamma_i^{\text{prev}}(n)}{\gamma_i^{\text{prev}}(n)} \leq \frac{\Delta\alpha^{\text{prev}}(n)}{\alpha^{\text{prev}}(n)} \tag{41}$$

by differentiation. However, it violates convex programming rules. But from $\gamma_i(n) \leq \alpha(n)$, we get

$$\frac{1}{\gamma_i^{\text{prev}}(n)} \geq \frac{1}{\alpha^{\text{prev}}(n)} \tag{42}$$

$$\frac{\Delta\alpha^{\text{prev}}(n)}{\gamma_i^{\text{prev}}(n)} \leq \frac{\Delta\alpha^{\text{prev}}(n)}{\alpha^{\text{prev}}(n)}. \tag{43}$$

Adding the constraint in Eq. (31), we can ensure that Eq. (41) is always satisfied since

$$\frac{\Delta\gamma_i^{\text{prev}}(n)}{\gamma_i^{\text{prev}}(n)} \leq \frac{\Delta\alpha^{\text{prev}}(n)}{\gamma_i^{\text{prev}}(n)} \leq \frac{\Delta\alpha^{\text{prev}}(n)}{\alpha^{\text{prev}}(n)}. \tag{44}$$

However, there is a limitation that this approach restricts parameter spaces.