AUTOMATIC MULTI-TRACK MIXING USING LINEAR DYNAMICAL SYSTEMS

Jeffrey Scott, Matthew Prockup, Erik M. Schmidt, Youngmoo E. Kim
Drexel University - Electrical and Computer Engineering
{jjscott mprockup, eschmidt, ykim}@drexel.edu

ABSTRACT

Over the past several decades music production has evolved from something that was only possible with multi-room, multi-million dollar studios into the province of the average person's living room. New tools for digital production have revolutionized the way we consume and interact with music on a daily basis. We propose a system based on a structured audio framework that can generate a basic mix-down of a set of multi-track audio files using parameters learned through supervised machine learning. Given the new surge of mobile content consumption, we extend this system to operate on a mobile device as an initial measure towards an integrated interactive mixing platform for multi-track music.

1. INTRODUCTION

The advent of digital audio and high-speed global communication has revolutionized the way people produce, distribute and consume music. It has become possible for individuals to make near professional recordings in their own home using a laptop and a microphone. While the technology has progressed enormously, a significant amount of skill and experience operating a digital audio workstation (DAW) is necessary to produce high quality results. Learning when and how to perform certain operations to transform a set of tracks into a polished product requires a large investment of time and training.

This paper targets the most basic studio production parameters, namely the mixing coefficients (fader levels) used to sum the tracks together to form a single audio output. Within the context of a standard rock/pop instrumentation (i.e. guitar, vocals, bass, and drums) the proportion of each track present in the mix is one of the most important factors determining the overall sound of a song. In this work we aim to predict time varying mixing coefficients for a set of multi-track *stems* that will produce a perceptually coherent and consistent final mix. Stem files are audio files that contain either a single instrument or a sub-mix of several instances of the same instrument or related instruments.

We describe a process for estimating the weighting coefficients when the source tracks and final mix are available but the actual gain parameters used in the final mix-down are unknown. We use the estimated fader values to train a linear dynamical system (LDS) that estimates the weighting coefficients using a set of acoustic features extracted from the audio. The mix attained from applying the predicted weights to the source tracks accurately represents the true version of the song. Currently, the system requires prior knowledge of the type of instrument present on each track and is limited (by the training data) to bass, drums, guitar, vocals and backing accompaniment. Here, backing accompaniment can be vocal harmonies, additional guitar, percussion or keyboards.

1.1 Structured Audio Integration

We introduce an initial implementation of an integrated system for interactive music mixing on a mobile platform for *structured audio* using Apple's iOS for the Apple iPhone, iPod Touch and iPad. In the broadest sense, structured audio is a representation of sound content using symbolic or semantic information as a means of encoding the data. Using parameters estimated by the automatic mixing system outlined herein, we can generate a mix of individual source tracks in real-time on the mobile device.

The application also facilitates the use of multi-track sessions exported directly from a producer's DAW using the Advanced Authoring Format (AAF). Unlike proprietary DAW formats, AAF is an open standard, and is accessible by an available set of APIs that allow developers to easily access information about a DAW session. This includes the source audio for each track and session parameters such as time sampled mixing coefficients and panning [1]. The session is uploaded to a user's mobile device where the mix can be re-automated and modified in a custom structured audio player. The mobile platform can alter the gain values in real time using the actual fader values for unknown source material or the estimated values computed using the automatic mixing system. An overview of the AutoMix iOS implementation, complete from producer to playback, is shown in Figure 1.

The remainder of the paper is organized as follows, Section 2 elaborates on the previous research in this area and Section 3 details the dataset and how it affected decisions in the system implementation. In Section 4 we describe the model used to extract the weights from the dataset as well as the training and testing of an LDS to perform prediction. Section 5 discusses our experiments and results. Section 6 describes the mobile application and Section 7 summarizes our findings and provides insight into the next

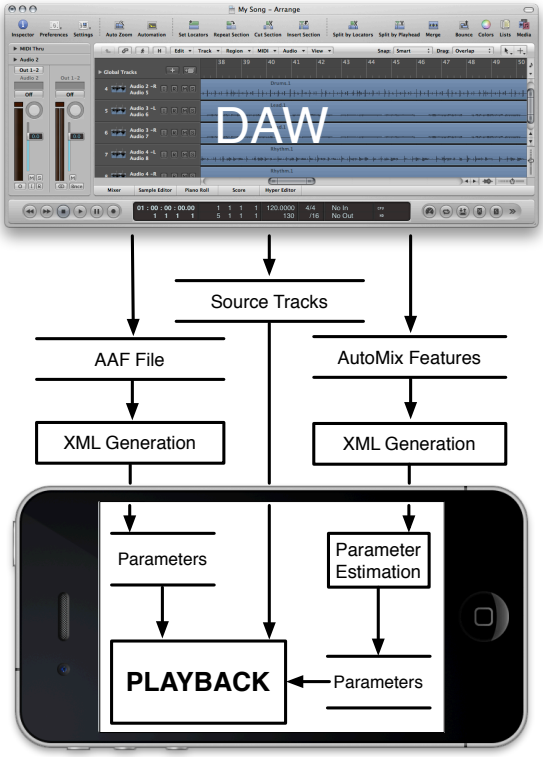

Figure 1. Diagram of iOS AutoMixing Implementation

avenue of research.

2. BACKGROUND

Much of the research in the area of automatic audio signal mixing is devoted to applications in the context of a live performance or event. Initial research on the subject was oriented toward broadcast, live panel discussion and similar environments dealing with the human voice as the primary audio source [2]. These systems analyze the amplitude of the audio signal and apply adaptive gating and thresholding to each input signal to create a coherent sound source mixture of the individual tracks in addition to feedback prevention.

More recent work incorporates perceptual features (loudness) into systems designed for live automatic gain control and cross-adaptive equalization [3, 4]. The implementation of the former focuses on adapting the fader level of each channel with the goal of achieving the same average loudness per channel. The latter is designed for use in live settings as a tool for inexperienced users or to reduce equipment setup time. The system attempts to dynamically filter various frequency bands in each channel so that all channels are heard equally well.

Structured audio is the representation of sound content with semantic information or algorithmic models [5]. This form of encoding allows for much higher data transmission rates as well as retrieval and manipulation of audio based on perceptual models. Currently, professional music post-production is performed by a highly skilled engineer with years of training. Using structured techniques, a parameterized, generative version of this process that is applicable to a variety of source audio is possible.

Other related work seeks to equalize an audio input based on a set of descriptive perceptual terms such as *bright* or *warm* [6]. Rather than attempt to navigate the complex network of sliders and knobs in an audio interface, a user can specify a high level term that describes the desired sound quality and an appropriate equalization curve will be applied. The system was developed through collecting user ratings for audio examples and performing linear regression to find a weighting function for a particular instrument/timbre pair.

3. DATASET

The dataset used to learn the mixing parameters is a set of multi-track source files from the RockBand® video game. A total of 48 songs were selected randomly from various pop/rock artists with each song belonging to a unique artist. The 'final mix' experienced during gameplay was acquired by recording the optical audio output of the game console onto a computer and aligning it to the source tracks. The game console mix was used, as opposed to the radio/album release, due to synchronization issues between the source files and the radio version. It was evident that time stretching/compression was performed on many of the RockBand® releases since the song from the commercial release was often not the same length as the version from the game console.

There were several inconsistencies in the dataset which we had to account for in order to make comparisons between songs more accurate. The number and type of sources varied between each song, with a minimum track count of eight and maximum of 14. For example, many songs had individual stereo (L and R) waveforms for each instrument, whereas other songs only had mono tracks for some instruments and stereo tracks for others. Additionally, not all songs had individual tracks for the kick drum, snare drum or overhead drum microphones.

To deal with this discrepancy, we opted to form five mono tracks for each song: bass, drums, guitar, vocals and backup. The instruments in the backup track vary from song to song and may contain vocal harmonies, synthesizers, percussion, guitar or a variety of other instruments, however the content of the backup track within a song is fairly consistent. Given the content of the dataset, this method created more uniformity between the content of each song.

To create a single mono track for each instrument class, we mixed all audio that belonged to the given instrument class according to the track weights computed using the method described in Section 4.1. A diagram of the preprocessing step is shown in Figure 2.

4. AUTOMATIC MIXING

With the current dataset of RockBand stems and the mixed output file, we do not have access to the exact fader values used to create the final output mix, therefore we must estimate these parameters in order to train our model. The weight estimation process is subject to several unknowns

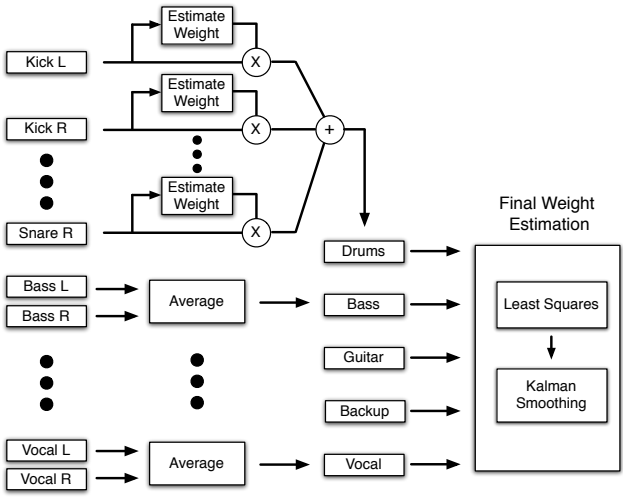

Figure 2. Diagram of dataset preprocessing for each track.

including additional compression and equalization of the stem tracks on the game console in producing the final mix. We use our estimated weights as ground truth for supervised machine learning and train an LDS to estimate a series of weighting coefficients for each track from a set of acoustic features extracted from the audio [7].

4.1 Weight Estimation

The process of mixing multi-track source files down to a single track is a linear combination of the audio sources in the time domain

$$\alpha_{1t}u_{1t} + \alpha_{2t}u_{2t} + \cdots + \alpha_{kt}u_{kt} = v_t \qquad (1)$$

where $\{\alpha_{1t}, \ldots \alpha_{kt}\}$ are the mixing coefficients of the k tracks at time t and $\{u_{1t}, \ldots u_{kt}\}$ are the time domain waveforms of each track.

Since the Fourier transform is a linear operator, we assume that the spectrum of the final mix at time t is a linear combination of the spectra of the source tracks at time t. Considering a single frame in time, we have

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} & \cdots & U_{1k} \\ U_{21} & U_{22} & U_{23} & \cdots & U_{2k} \\ \vdots & & \vdots & & \vdots \\ U_{N1} & U_{N2} & U_{N3} & \cdots & U_{Nk} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{bmatrix} \approx \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_N \end{bmatrix}$$

$$\mathbf{U}\alpha \approx \mathbf{V} \qquad (2)$$

where each column in $\mathbf{U}$ is the magnitude spectrum of the kth track and $\mathbf{V}$ is the spectrum of the final mix. We are careful here to note that Equation 2 is not an exact equality in the context of our real data. Small offsets due to misalignment of the stems with the reference track will introduce error as a phase offset.

Given a set of multi-track stems and the resulting audio produced by mixing the individual tracks, we can estimate the mixing coefficients, α_k, using non-negative least squares (NNLS) [8].

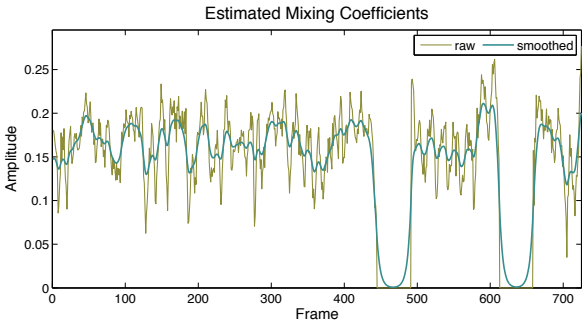

Figure 3. Extracted weights for bass guitar using NNLS, Kalman smoothing and normalization.

$$\hat{\alpha} = \min_{\alpha} ||\mathbf{U}\alpha - \mathbf{V}||_2^2 \qquad \alpha \geq 0 \qquad (3)$$

We select NNLS to estimate the weights since the mixing process is additive by definition. Using unconstrained least squares, we experience both very large values for some weights since the algorithm can increase the weight of tracks that contain very little energy to reduce the overall error.

We perform this analysis on a frame-by-frame basis using a 1 second rectangular window and overlap the frames by 0.75 seconds. In each frame, we compute the spectrogram of each individual track using a 1024 sample window with a 512 sample overlap. We vectorize and concatenate the spectrograms to attain the form given in Equation 2 then compute the weights. A resolution of 0.25 seconds for changing fader values is sufficient to capture the dynamic changes in each track.

To improve the initial estimate of the weights, we only include tracks that contain audio in the given frame. Assuming we have k tracks, if RMS$(u_{kt}) < 0.01$, then we negate the track in the estimate of the weight vector for the current frame and use $k - p$ tracks, where p is the number of inactive tracks. Removing these tracks prevents very large weight coefficients from being calculated for tracks that have very little energy. The value of 0.01 was empirically determined to provide good peak suppression in the weight estimates.

We then process the weight vector using Kalman smoothing to reduce the noise that still remains in the signal [9]. The initial weight estimates as well as the smoothed weights are depicted in Figure 3. In the following section, we assume that the mixing coefficients are Gaussian when modeling the data. A histogram showing the distributions of mixing coefficients for multiple instruments is shown in Figure 4.

It is significant to note that while these coefficients produce a mix that is perceptually very similar to the original track, they are not the actual ground truth weights. We provide online audio examples of the original song and the mix using the estimated weights[1].

[1] http://music.ece.drexel.edu/research/AutoMix

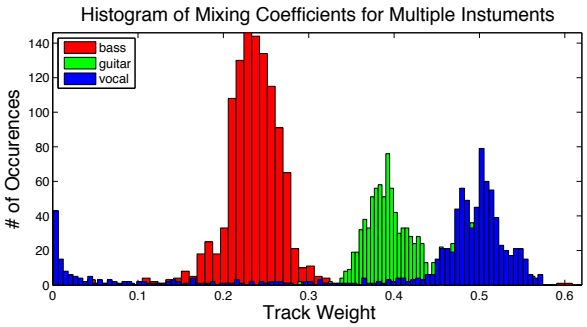

Figure 4. Histogram of mixing coefficients.

4.2 Feature Extraction

The features we extract to train the LDS are a combination of time domain and spectral domain values. The following features were used to train the model:

- Spectral Centroid
- Root Mean Square (RMS) Energy
- Slope/Intercept from fitting a line to the spectrum

4.3 Modeling

We train two different models using acoustic features to predict the time-varying mixing coefficients for an unknown input song. We first use multiple linear regression (MLR) to find the projection from features to weights that minimizes error in the least squares sense. To model time dependence between the mixing coefficients of a given track, we use a linear dynamical system (LDS) and compute the latent states using Kalman filtering.

4.4 Multiple Linear Regression

We assume that each weight vector α is a linear combination of our features $\{y_1, \ldots, y_m\}$

$$\boldsymbol{\alpha} = \mathbf{Y}\boldsymbol{\beta} \tag{4}$$

where Y is an $N \times M$ matrix, M is the number of features we have per frame, N is the number of frames and k indexes the track. We compute the projection matrix used to find the weighting coefficients of a new song,

$$\hat{\boldsymbol{\beta}} = \min_{\beta} ||\mathbf{Y}\boldsymbol{\beta} - \boldsymbol{\alpha}||_2^2 \tag{5}$$

and estimate the mixing coefficients of an unknown song by applying the projection to the feature data Y.

$$\hat{\boldsymbol{\alpha}} = \mathbf{Y}\hat{\boldsymbol{\beta}} \tag{6}$$

This model assumes that the mixing coefficients are independent with respect to time. In the next section we describe a model that considers the time dependence of the data.

4.5 Linear Dynamical System

We treat the time-varying mixing coefficients α as the latent states resulting from some noisy process and our features, y as noisy observations of the output of our model.

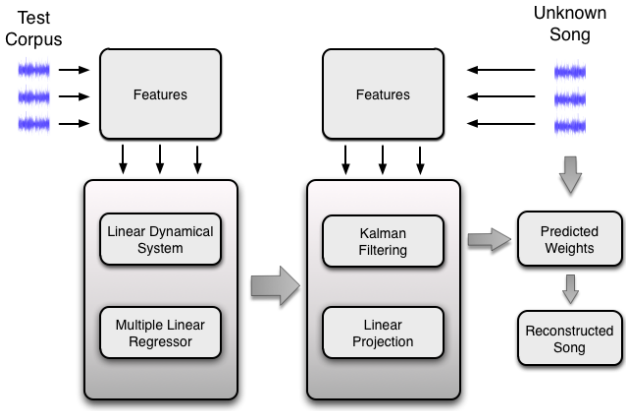

Figure 5. Supervised machine learning of gain coefficients using LDS and MLR.

We formulate the linear dynamical system as follows

$$\boldsymbol{\alpha}_t = \mathbf{A}\boldsymbol{\alpha}_{t-1} + \mathbf{w}_t, \tag{7}$$

$$\mathbf{y}_t = \mathbf{C}\boldsymbol{\alpha}_t + \mathbf{v}_t \tag{8}$$

Here, w_t and v_t are zero mean Gaussian noise sources

$$\mathbf{w} \sim \mathcal{N}(0, Q) \tag{9}$$

$$\mathbf{v} \sim \mathcal{N}(0, R) \tag{10}$$

The dynamics matrix $\mathbf{A}$ models the evolution of the weights as a linear transformation in each time step and $\mathbf{C}$ translates the α values into our observation space $y \in Y^R$.

To train the model we estimate $\mathbf{A}$ and $\mathbf{C}$ through constraint generation and least squares, respectively. We opt for a constraint generation approach to estimate $\mathbf{C}$ since a stable solution is guaranteed [10]. The covariances $\mathbf{Q}$ and $\mathbf{R}$ are computed from the residuals of $\mathbf{A}$ and $\mathbf{C}$. Prior to training, we remove the means of the features and weights since our model assumes that the process is Gaussian and zero mean. The feature $\bar{\mathbf{y}}$ and weight $\bar{\boldsymbol{\alpha}}$ means are retained for the testing phase.

For an unknown set of stems, we compute our acoustic features for each track and remove the training feature bias, $\bar{\mathbf{y}}$. We then perform the forward Kalman recursions using the $\mathbf{A}$, $\mathbf{C}$, $\mathbf{Q}$ and $\mathbf{R}$ parameters learned during training to get an estimate of the weighting coefficients. Adding the weight bias $\bar{\boldsymbol{\alpha}}$ to this result yields our final estimate of the mixing coefficients. A diagram of the feature extraction, training and estimation/prediction is shown in Figure 5.

5. RESULTS

Training and testing is performed in a typical manner for a supervised machine learning task. Given the relatively small size ($N = 48$) of the dataset we opt to use leave-one-out cross-validation, training on $N - 1$ songs and testing on the remaining song. This process is repeated for all N songs such that each is a test song only once.

We define $\mathbf{Y}_{train}$ as a matrix formed by concatenating the features of all songs, and $\boldsymbol{\alpha}_{train}$ as the matrix formed by concatenating all weighting coefficients for all songs.

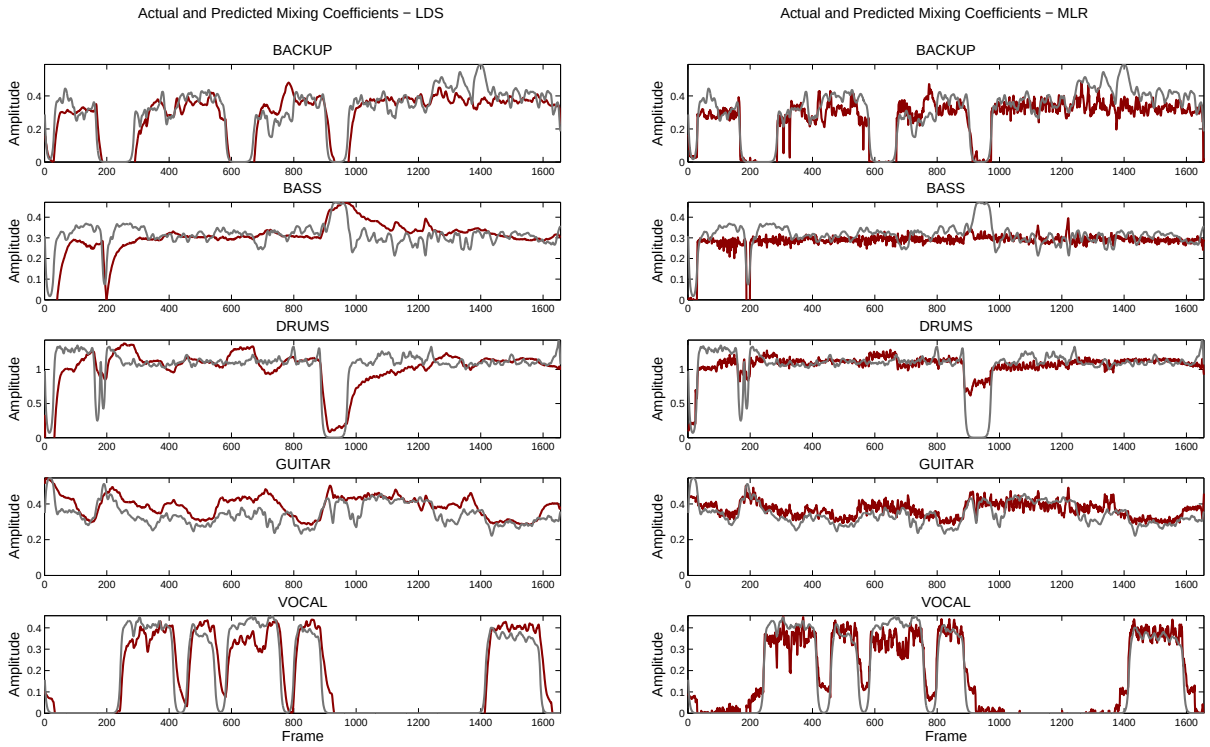

Figure 6. Results for weighting coefficient prediction using LDS (left) and MLR (right). The estimated ground truth weights are shown in gray and the predicted coefficients are depicted in red.

Track	LDS	MLR
backup	0.0126 ± 0.0076	0.0091 ± 0.0075
bass	0.0191 ± 0.0183	0.0086 ± 0.0102
drums	0.1452 ± 0.1237	0.0590 ± 0.0444
guitar	0.0158 ± 0.0169	0.0075 ± 0.0077
vocal	0.0188 ± 0.0107	0.0149 ± 0.0124

Table 1. Average mean squared error across all songs between ground truth weights and predicted weights for MLR and LDS.

These quantities are then used to train the parameters of an LDS. We perform Kalman filtering on the remaining test song using the parameters learned in the training phase to estimate the time-varying weights for the song.

Figure 6 shows the predicted and actual weights plotted on the same axis for each instrument in the song "Constant Motion" by Dream Theatre. The resulting weights from MLR fit the data better and result in a lower error and the weights computed through Kalman filtering are much smoother yet sometimes exhibit bias or offset from the actual values. Table 1 shows the average mean squared error for all songs in the database for both algorithms.

Using a small dimensional feature set, we are able to generate a mix that is comparable to the desired result. Audio examples of the original mix, the drum sub-mixes and the reconstructed mix using the predicted weights can be found online at the previously specified link. A listening analysis performed by the authors finds that the LDS and MLR models yield very similar perceptual results. For comparison, we generated audio mixes using a simple averaging of all tracks. The result of this oversimplified model is hardly comparable to the results from the automatic mixing system.

Although these results are good, we note that the weights estimated in Section 4.1 are not the true parameters. In order to have a cleaner data set we need a large collection of multi-track session files and access to the actual parameters. This idea is explored further in the next section as we introduce a mobile device application that can faithfully reproduce a mix using the actual parameters as well as the estimated model parameters from our system.

6. INTEGRATION INTO STRUCTURED AUDIO FRAMEWORK

The system detailed in Section 4 essentially describes a structured audio representation of music content. A song is represented by its component tracks and the output of a model, which, when applied to these component tracks produces a song. We incorporate the auto-mixing system into a larger framework for mixing and post-production of audio stem files. Within this framework we can potentially incorporate the actual parameters from DAW production sessions into a platform that allows the user to interact with the multi-track session in a variety of ways.

6.1 Automatic Mixing Implementation

The wide use of mobile technology for digital content consumption is the motivation to implement the automatic

mixing system for use on the iOS series of mobile devices. We created a set of tools that import audio features onto a mobile device then use these features to estimate the parameters using a hardware accelerated linear algebra library. The estimation is computed efficiently in real-time, mixing the output audio appropriately.

The source audio, feature data, and parameters of each model are computed offline, then uploaded to the mobile device in an XML wrapper to be parsed by the mixing platform. The LDS model includes the following data:

- features - $\mathbf{Y}$
- dynamics - $\mathbf{A}$
- translation - $\mathbf{C}$
- covariance - $\mathbf{Q}$
- covariance - $\mathbf{R}$

The MLR model is simpler, and only requires the features $\mathbf{Y}$ and projection matrix $\hat{\beta}$.

As of iOS 4, Apple added functionality of the Basic Linear Algebra Subprograms Library (BLAS) as part of their Accelerate Framework [2]. This allows the device to perform hardware accelerated linear algebra calculations that are crucial to the MLR and LDS approaches to gain parameter estimation. The calculations are performed on a frame by frame basis and applied to the individual audio channels in real-time. The LDS approach is more computationally intensive in its operation, whereas MLR is a simple projection of the feature data. Both, however, run in real-time on the device.

6.2 Future Work: Collaboration with Producers

The current system works well for a specific subset (rock instrumentation) of source track content on the device due to the RockBand® dataset used for testing and training. While the current model may extrapolate to a variety of source content, the overall goal is to move beyond the RockBand® stems and collaborate directly with producers in the music industry in order to obtain data.

Most professional mixing is performed on a Digital Audio Workstation (DAW) such as AVID's Protools, Apple's Logic, MOTU's Digital Performer, and Steinberg's Cubase. All of these platforms, as well as many others allow the user to export an AAF file containing metadata of the parameters used to combine and process the individual source tracks. The metadata is the automation of parameters such as the gain and pan of a mixer channel. When the gain of a certain track is adjusted over time, the DAW records this alteration and reproduces it on playback. Using the source audio and the mixer automation information, a simple DAW session can be recreated outside of the platform it was created in. Collecting data in this format will facilitate integration and comparison between the metadata from divergent platforms.

We plan to use the data in AAF files to generate a more robust model of the parameter space in Section 4. We hope to increase the accuracy of the current predictions in regards to the standard rock instrumentation as well as enable reliable modeling for a larger class of instruments. Incorporating these ideas into the current framework will allow novice users to interact in a novel way with available structured audio content with their own user generated content.

[2] http://developer.apple.com/performance/accelerateframework.html

7. CONCLUSIONS

Leveraging a structured audio representation of music content, we developed a system that can estimate the original mixing coefficients of a multi-track recording session through least squares estimation. Using this result as ground truth, we trained a system to predict the time-varying parameters that produce a perceptually coherent mixture of unknown source content using minimal prior information. We deployed this system on a popular mobile device to show the creative potential for developing interactive music production applications that facilitate various modalities of personalization and customization for consuming music content.

8. REFERENCES

[1] B. Gilmer, "AAF-the advanced authoring format," White paper, AAF Association, July 2002, http://www.aafassociation.org.

[2] D. Dugan, "Automatic microphone mixing," *J. Audio Eng. Soc*, vol. 23, no. 6, pp. 442–449, 1975.

[3] E. Perez-Gonzalez and J. Reiss, "Automatic gain and fader control for live mixing," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2009, pp. 1–4.

[4] ——, "Automatic equalization of multichannel audio using cross-adaptive methods," in *Audio Engineering Society Convention 127*, 10 2009.

[5] B. Vercoe, W. Gardner, and E. Scheirer, "Structured audio: Creation, transmission, and rendering of parametric sound representations," in *Proceedings of the IEEE*, 1998, pp. 922–940.

[6] A. T. Sabin and B. Pardo, "A method for rapid personalization of audio equalization parameters," *Proceedings of ACM Multimedia*, pp. 769–772, 2009.

[7] E. M. Schmidt and Y. E. Kim, "Prediction of time-varying musical mood distributions using kalman filtering," in *Proceedings of the 2010 IEEE International Conference on Machine Learning and Applications*, 2010.

[8] C. L. Lawson and R. J. Hanson, *Solving least squares problems*, 3rd ed., C. L. Lawson and R. J. Hanson, Eds. Prentice-Hall, 1995.

[9] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[10] S. Siddiqi, B. Boots, and G. Gordon, "A constraint generation approach to learning stable linear dynamical systems," in *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press, 2008, pp. 1329–1336.