A REAL-TIME SIGNAL PROCESSING FRAMEWORK OF MUSICAL EXPRESSIVE FEATURE EXTRACTION USING MATLAB

Ren Gang[1], Gregory Bocko[1], Justin Lundberg[2], Stephen Roessner[1], Dave Headlam[1,2], Mark F. Bocko[1,2]

[1]Dept. of Electrical and Computer Engineering, Edmund A. Hajim School of Engineering and Applied Sciences, University of Rochester; [2]Dept. of Music Theory, Eastman School of Music, University of Rochester

g.ren@rochester.edu,gregory.bocko@rochester.edu,justin.lundberg@rochester.edu, stephen.roessner@rochester.edu,dheadlam@esm.rochester.edu, mark.bocko@rochester.edu

ABSTRACT

In this paper we propose a real-time signal processing framework for musical audio that 1) aligns the audio with an existing music score or creates a musical score by automated music transcription algorithms; and 2) obtains the expressive feature descriptors of music performance by comparing the score with the audio. Real-time audio segmentation algorithms are implemented to identify the onset points of music notes in the incoming audio stream. The score related features and musical expressive features are extracted based on these segmentation results. In a real-time setting, these audio segmentation and feature extraction operations have to be accomplished at (or shortly after) the note onset points, when an incomplete length of audio signal is captured. To satisfy real-time processing requirements while maintaining feature accuracy, our proposed framework combines the processing stages of prediction, estimation, and updating in both audio segmentation and feature extraction algorithms in an integrated refinement process. The proposed framework is implemented in a MATLAB real-time signal processing framework.

1. INTRODUCTION

Music performance adds interpretative information to the shorthand representation in a music score [1]. These performance dimensions can be extracted from performance audio as musical expressive features using signal processing algorithms as in [2]. These features quantitatively model the performance dimensions that reflect both the interpretation of performance musicians and the artistic intention of composers [1] and are important for various music signal processing [2] and semantic musical data analysis [3,4] applications.

Existing automatic music transcription [5] and musical expressive feature extraction algorithms [2] are designed in post-processing frameworks. These existing algorithms are essentially multimedia file process systems, which assume that the entire duration of the audio performance is already recorded. However, various real-time signal processing applications, such as visualization, automatic music mixing, stage lighting control, interactive music media, and electronic games, require that musical expressive features be extracted and synchronized with the ongoing audio. In such a real-time signal processing framework, the musical expressive features have to be obtained from the audio signal that is still in progression to facilitate simultaneous interactions with external applications. Thus, the complete music event is not observed at the "decision point" since the music transcription and expressive features have to be obtained at (or shortly after) the onset of each music event. In this paper we extend the feature extraction and recognition functionalities of conventional music transcription and musical expressive feature extraction algorithms and establish a real-time processing framework which includes the processing stages of prediction, estimation and updating. First, signal features (signal features here include segmentation features, score-level features and musical expressive features) are predicted using generative probabilistic graphical models [6,7] based on a "history" of these features (or other available information, e.g., features extracted from a synchronized rehearsal track). Then we estimate these signal features when a short audio segment in the beginning part of a music event is available. When additional audio frames are captured, we refine the estimations and make necessary updating.

"True" real-time methods can only be achieved in a feature prediction framework: the signal features are obtained before the actual music event. For example, in an automatic music mixing system, we are expected to adjust the fader settings according to the "past" signal features *before* a loud section begins. That is, the expressive loudness feature and its related fader instruction must be generated at a

time point when the music event of the "future" loud section is not observed at all! In our proposed processing framework, a generative probabilistic graphical model [6] is employed to enable such predictions. A probabilistic graphical model depicts the causality (or statistical) relations between signal features [7]. A prediction of "future" signal features is inferred from these statistical relations and a finite length of an observed "history". Such predictions might fail, as any prediction that peeks into an unknown "future". To improve the reliability of our proposed system, several levels of relaxation are applied. These pseudo-real-time processing frameworks are essentially buffer and post-processing frameworks that allow us to take glimpses at the music event and be more "confident". If the signal processing delays they introduce are kept within the perceptual limit (about 25ms [8]), the live performance, audio and the feature processing results would appear to be perceptually well synchronized for the audience.

A pseudo-real-time processing framework allows a short audio frame to be captured near the predicted music note onset. The signal features extracted from this short audio frame confirms or rejects the predicted onset location and other signal feature dimensions. If the pseudo-real-time constrains, including the perceptual delay limit and/or the audio reinforcement delay limit, are satisfied, a short signal capturing and processing delay would be effectively concealed from the audience. The perceptual delay limit is the limit of human perceptual capabilities of discerning the time sequence of two perceptual events [8,9]. For application scenarios such as visualization, a short delay such as 10ms in the visualization interface is not perceptible since human visual perception is a relatively slow responding process [9]. However, a processing delay that exceeds 20ms results in a sloppy "thunder first, lightning second" effect. An audio reinforcement delay can be utilized in application scenarios where sound reinforcement systems are employed to further enhance synchronizations. The reinforced sound is briefly delayed to compensate for the signal processing delays so the reinforced sound is still synchronized with the feature extraction and processing results[1]. Because the signal features extracted usually trigger the most dramatic visual and aural events and the reinforced audio carries the most prominent aural event, this audio reinforcement delay effectuates the most critical synchronizations and is thus strongly recommended whenever applicable. The sound reinforcement delay must be kept low (less than 20ms, with a typical value of 10ms) to maintain the perceptual synchronizations of other aural and visual events. On the aural aspect, the audio reinforcement delay limit ensures that the direct sound from actors can blend seamlessly with the reinforced sound for front-row audiences. On the visual side, the reinforced audio lags behind the stage scenes so this limit insures that the time lag is perceptually tolerable.

The proposed system architecture as detailed in Sec. 2 utilizes both real-time music event prediction and pseudo-real-time processing, with an emphasis on pseudo-real-time processing. Key processing components are introduced in Sec. 3. Sec. 4 discusses the MATLAB implementation issues and Sec. 5 provides a brief summary.

2. SYSTEM ARCHITECTURE

The system architecture of our proposed system is illustrated in Figure 1. Figure 1(a) is the system architecture for application scenarios when a music score database is available and a matching music score is retrieved. In the initialization phase, a short audio segment (5-20 seconds) is first captured as the audio query for finding the matching music score using score-audio matching algorithms [10]. The feature estimation blocks include audio segmentation and features extraction algorithm. The real-time score-audio alignment algorithm segments the audio by identifying the onset points based on the music score and the segmentation features extracted from the audio. If a music onset is detected, the following short audio frame is captured and passed on to the musical expressive feature extraction algorithm to obtain an initial estimation of musical expressive features. These musical expressive features are then formatted as a control data stream for external applications. Figure 1(b) presents alternative system architecture for the application scenarios when a music score is not available. In this system we implement a real-time music transcription framework parallel with the real-time musical expressive feature extraction process. For both systems music event prediction and feature updating algorithms are implemented to further improve performance. The music event prediction algorithm predicts the "future" feature values based on a "history" and use the prediction values as priors for the "current" music event segmentation and feature estimation process. The alignment/feature updating algorithm refines signal features when additional audio frames are captured and submits essential corrections. The refined features also improve subsequent probabilistic predictions.

[1] In a staged music setting, for instance, the music expressive features and the aural-visual events controlled by these features are delayed behind the onset of stage scenes because a short audio frame have to be captured and processed. Taking the stage light control application as an example, the light controlled by loudness feature turns on 10ms after an actor begin to sing a music phrase. In this "precious" 10ms, a short audio frame is captured and analyzed so the "light on" stage lighting instruction could be inferred. The reinforced audio is also delayed 10ms to compensate for the delay of the lighting effect. For the audience the reinforced audio onset is perfectly synchronized with the lighting effect since they are both delayed 10ms behind the actor, while the 10ms delay between stage scene and audio/lighting is still imperceptible.

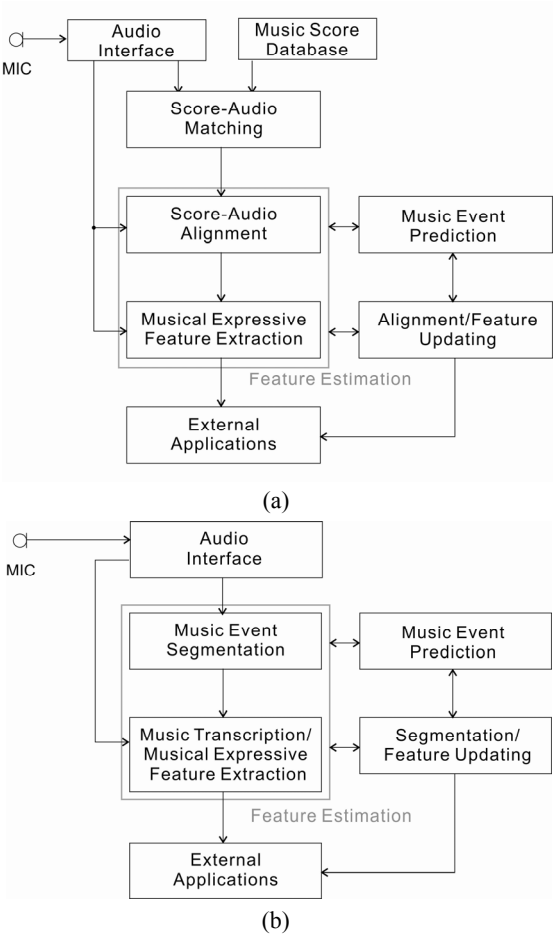

Figure 1. **System Architecture.** (a) is the system architecture when a music score database is available. (b) is the system architecture when the music score is not available.

3. REAL-TIME PROCESSING ALGORITHMS

Real-time processing algorithms for key functional blocks are introduced in this section. Algorithms both for application scenarios with and without a music score are introduced.

3.1 Audio Segmentation

If a music score is available, the note boundaries are identified using score-audio alignment or score following algorithms based on real-time dynamic time warping as detailed in [10]. These algorithms optimally align a music score to the dynamic performance timeline of an audio file by searching-and-finding an optimal alignment path corresponding to the alignment features extracted from the score and the audio.

If music score is not available, the conventional onset detection and music event segmentation algorithms [11] are extended to fit in our proposed real-time processing framework. These onset detection algorithms compare the audio features (for example, energy value or spectrographic content) and track their variations. The magnitude of the variations is encoded as an onset detection function $D(t)$ and the time points correspond to significant variation are selected as onsets or segmentation points. In our proposed real-time framework only the 'past' part of the detection function $D(t), t \leq t_c$ is available, where t_c is the current time. To ensure real-time processing performance, we cannot delay the segmentation decision until a downward slope of $D(t)$ is observed. Instead of peak-picking [11], the segmentation decisions have to be generated using a threshold detection method, which do not guarantee that a $D(t)$ peak is reached.

Our proposed real-time processing framework is implemented by providing two types of threshold for onset detection. An initial detection threshold is set as th_1. If $D(t_c) > th_1$ and no segmentation decisions have be generated in a time interval of t_c, an initial segmentation point is identified. A 'regretting' threshold is set as th_2. If $D(t_c) > th_2$ and the time distance t_r to the previous segmentation decision satisfies $t_{r1} \leq t_r \leq t_{r2}$, a forward updating of the segmentation point is performed to erase an existing segmentation point and substitute the current time point. Here t_{r1} is the segmentation error tolerance. If the previous segmentation point is within this range, a correction is not necessary. t_{r2} is the maximum correction range. If the time interval to the previous segmentation point is greater than t_{r2}, another segmentation point is generated using threshold th_1. These thresholds are time varying with the current beat tracking result obtained using the algorithms in [12]. The rhythmically significant locations are assigned a lower detection threshold as in Fig. 2 to push detected onsets towards these interpolated locations, as a combined process of prediction and real-time detection.

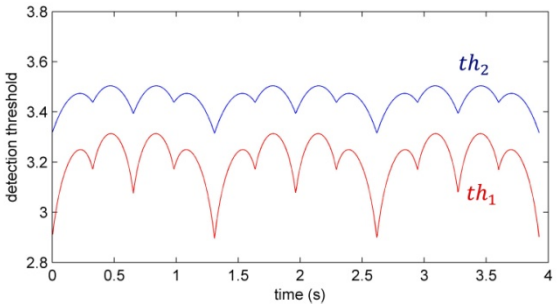

Figure 2. **A typical profile of segmentation detection thresholds.** The lower detection threshold at predicted rhythmic locations pushes the segmentation point towards a predicted rhythmic grid.

3.2 Feature Extraction

The musical expressive features we implemented include feature dimensions of the relatively small but continually changing adjustments in pitch, timing, auditory loudness, timbre, articulation and vibrato that performers use to create expression [1,2]. Definitions of these feature dimensions are briefly summarized in Table 1 and more details can be found in [2]. In this section the real-time extraction process of symbolic pitch and expressive feature dimension of pitch deviation is detailed in an application scenario when a music score is not available. Pitch deviation measures the difference between performance pitch and the score specified pitch [2]. The expressive pitch processing is more sophisticated compared to other feature dimensions since the quantized score pitch, the expressive pitch deviations, and the calibration of a temperament grid[1] have to be updated simultaneously. The other feature dimensions are briefly summarized in Table 1 and their feature extraction algorithms are similar extensions based on [2].

For estimation of pitch deviation an accurate mapping between symbolic pitch and fundamental frequency (F0) has to be established since the expressive pitch deviation is just a small fraction of the fundamental frequency. The fundamental frequency is first obtained from the audio frames captured at the segmentation point using a pitch detection algorithm as in [13]. Suppose that the fundamental frequency is detected from the first short audio frame of music note m and denoted as $f_{(1)}$ and the initial temperament grid we implemented as $[\check{f}_m, \hat{f}_m; \bar{f}_m, p_m], m = 1, \ldots, M$. Here $\check{f}_m$ and $\hat{f}_m$ is the decision boundary of the pitch quantization grid. $\bar{f}_m$ is the quantized frequency value that would be selected if $\check{f}_m \le f_m \le \hat{f}_m$ and p_m is its symbolic value. For equal temperament scale, the quantized value $\bar{f}_m$s form a temperament grid which is derived from a reference frequency point $\bar{f}_R$ with symbolic pitch value p_R as:

$$\bar{f}_m = \mathrm{tpa_e}(\bar{f}_R, p_R; p_m) = 2^{\frac{p_m - p_R}{12}} \cdot \bar{f}_R \qquad (1)$$

where p_m is the symbolic pitch value of quantization interval $[\check{f}_m, \hat{f}_m)$, here p_m and p_R is specified in MIDI value. Since human frequency discernment is most acute at mid-frequency region, the frequency reference point $[p_R: \bar{f}_R]$ could be selected at this frequency region. In our implementation the reference point [69:440Hz] is selected. Using this initial temperament grid, we obtain the initial symbolic pitch value as $p_{(1)}$. When additional audio frames are captured from the audio stream, we might revise our estimation of the $f_{(1)}$ and $p_{(1)}$ values within a music note based on the pitch detected in the extended musical note duration. To ensure a smooth updating process we only update the F0 estimation after a time interval. We also only update the estimated value of fundamental frequency and pitch deviation if the difference of two adjacent estimated F0 values will exceed the detection grid of one semitone.

When an adequate number of music notes are captured, the temperament grid is updated by fitting a temperament grid to the detected F0 values in a calibration process. Suppose the F0 sequence we obtained is represented as $f_1 \ldots f_M$, these frequency points find their quantized values $\bar{f}_1 \ldots \bar{f}_M$ as the nearest neighbors in an initial quantization grid with frequency reference point $[p_R: \bar{f}_R]$. The residual values of this quantization process are denoted as $d_1 \ldots d_M$. Then we shift the frequency reference point within 1/6 of a semitone interval and find the best reference frequency point $\bar{f}_R + \Delta\bar{f}_R$ where the sum of the residual values $\sum_{m=1}^{M} |d_m|$ is minimized. After this calibration process the residual frequency value $d_1 \ldots d_M$ is calculated as pitch deviation values. The pitch deviation in the units of cents is calculated as $1200 \cdot log_2(d/\bar{f})$. An example of pitch feature extraction and feature updating process is illustrated in Figure 3.

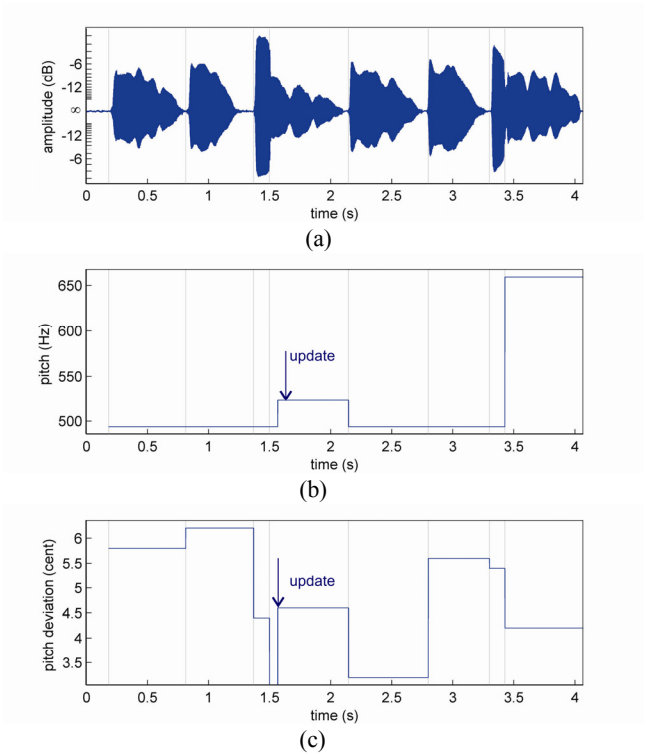

Figure 3. **Estimation and Updating Process of Musical Pitch Related Features.** (a) audio waveform; (b) quantized musical pitch; (c) expressive pitch deviation.

[1] For expressive feature extraction this calibration is crucial because the calibration level is within the same range of pitch deviation value.

Feature	Definition	Real-Time Musical Expressive Feature Extraction Algorithms	Typical Value
Pitch Deviation	The difference between performance pitch and score pitch	(1) The fundamental frequency of an audio segment is detected using a pitch analysis algorithm as described in [13]. (2) A temperament grid is initialized and fit to the fundamental frequency sequence as the music note number increase. The deviation of the optimum temperament grid is utilized as the pitch calibration value. (3) The pitch deviation is calculated by comparing the audio pitch f and with score pitch p.	-15 cents to 15 cents
Auditory Loudness	The perceptual intensity of sound	Calculate the strength of auditory response [2] of an short audio segment of 20ms based on its energy distribution in the frequency domain, using a computational auditory model	30 dB dynamic range
Timing	The time difference of music events between the score and the audio.	The time deviation of onset n is calculated the normalized onset time deviation as: $F_T(n) = \frac{t(n+1) - t(n)}{\hat{t}(n+1) - \hat{t}(n)}$ where $t(n)$ is the audio onset timing and $\hat{t}(n)$ is the interpolated score timing. $t(n+1)$ denotes the next onset location. $F_T(n)$ can be viewed as an indicator of timing extension ($F_T(n) > 1$) or compression ($F_T(n) < 1$).	From 0.6 (compression) to 1.5 (extension)
Timbre	The energy distribution pattern of the frequency domain	(1) The short time Fourier analysis result is $S_M(i,k)$ is calaulated, where k is the frequency bin index. i is the time frame index. (2) The timbre centroid is calculated as the "weight center" of the frequency spectrum of a analysis segment as: $c(i) = \sum_{k=1}^{K} kS_M^2(i,k) / k_F \sum_{k=1}^{K} S_M^2(i,k)$ where k_F is the frequency bin index of fundamental sonic partial. (3) Timbre width is defined as the frequency width $b(i)$ required to include a pre-defined portion η (with a typical value of 90%) of the total energy.	Timbre centroid from 1.2 to 4. Timbre width from 1.5 to 3.
Attack	The transient characteristics of music onset	The attack feature [2] is calculated as the ratio of the energy content of the first 1/3 of the note duration.	from 0.5 to 3.
Vibrato	The amplitude and frequency modulation inside a musical note	(1) A band-pass filter is implemented to extract a single sonic partial from the complex harmonic sound for analysis. (2) A musical vibrato recognition algorithm is implemented as in [14]. The modulation components of a vibrato note is extracted using analytic signal methods [2].	Amplitude modulation depth from 0.1 to 0.4.

Table 1. The definitions and real-time feature extraction algorithms of musical expressive features

3.3 Music Event Prediction

Certain aspects of music event prediction have been introduced in the real-time audio segmentation algorithm as in Sec. 3.1, where we perform a beat detection algorithm and interpolate the beat detection results as predictors of "future" rhythmic structure. The statistical relations within a time series of audio features are codified using probabilistic graphical models [7] as a prediction framework to infer the "future" feature values based on available observations. Complete learning and inference algorithms of a music event prediction framework are detailed in [6]. Most real-time applications require an early "decision point", where the available audio segment is still insufficient for unambiguously estimating most feature dimensions. Thus in our proposed frameworks these probabilistic predictions are integrated into the audio segmentation and feature extraction process. The signal features are predicted before the onset of an actual music event as prior information for feature estimations. Additional reference feature tracks including a music score or a matching expressive music transcription obtained from a rehearsal track can be further incorporated in this prediction framework, as an extension to the alignment process that assigns reference features as the prediction values to real-time music events. The integration of prediction and estimation also allows the prediction point to be closer to the "decision point", as the shortened prediction distance enhances the prediction accuracy [6].

3.4 Feature Updating

The real-time segmentation decision process here is essentially a hit-or-miss process: once a segmentation decision is made based on the audio signal features of the "current" audio frame (we may also utilize the "past" audio frames deposited in the captured signal stream and some prediction) any audio frames captured later will not count even if the 'hit' (the attack point) is at the wrong place. If we "miss" a segmentation point due to a stringent detection threshold, we may find that the subsequently captured audio frames are inappropriate for allocating a segmentation point. The design of real-time feature extraction algorithms also have to balance these requirements of real-time performance and feature accuracy. To reconcile these conflicting real-time performance criteria we implement an updating mechanism which enables the system to "regret" previous prediction/estimation when subsequent events in the audio stream are captured and processed. These refinements are buffered for improving future predictions and essential updates are submitted to the external applications. Although for some application scenarios a real-time decision is irreversible, certain minor corrections can still be effectively disguised using perceptual models [9]. Because frequent revisions give the system user an unstable impression, the number of segmentation point modifications must be restricted. An example of a feature updating process is illustrated in Figure 3.

4. MATLAB IMPLEMENTATION

In a MATLAB real-time signal processing framework a *timer* object [15] is implemented to handle the looping operation and schedule the subsequent processing operations. In a *timer* object loop a block of main code is executed iteratively in a prescribed short time slot until an error or user interruption is detected. In our implementation the audio capturing and processing functionalities are programmed within the main *timer* loop so for every *timer* slot an audio frame is captured, analyzed and the feature data is submitted to the external application. If the *timer* slot is short enough (i.e., 10ms), the buffering and processing delay is negligible. If the capturing and processing time exceeds the allocated *timer* object slot, the error handling function of the *timer* object is implemented. The error handling code contains the same processing steps as in a regular processing *timer* slot and the code to resumes regular *timer* cycles after error processing. This mechanism allows extra processing time when necessary. The audio capturing functionality is implemented by programming two *audiorecorder* objects in each processing cycle to make sure that there is no missing audio segment due to the processing delays. For the odd-numbered processing loops (including *timer* loops and error handling loops), we capture the recorded audio segment from *audiorecorder1,* read the time location, clear and restart the recorder, and then append the audio segment to the corresponding time location of the main audio stream for subsequent processing. For the even-numbered loops, we perform the same instructions on *audiorecorder2*. In MATLAB, multiple *audiorecorder* objects are run-time independent so their functionalities are performed simultaneously without interference.

5. SUMMARY

Our proposed real-time signal processing framework of musical expressive feature extraction obtains musical features from an incoming audio stream and provides important music data for various multimedia applications such as visualization, electronic games, interactive media and automatic music production. By implementing a processing framework that combines prediction, estimation and updating, musical features are obtained at the music note onset. This capability effectively synchronizes the musical expressive features with interactive content and avoids the delay effect of conventional post-processing frameworks. The proposed updating processing enables important feature modifications to be updated to the user interface when additional lengths of audio signal are captured. In a performance evaluation the performance of our proposed real-time processing framework and an automatic post-processing framework [2] is compared with a benchmark dataset of manually annotated musical feature analysis. If any feature dimension of automatic processing is different from the benchmark dataset, the music note is considered an error. The error rate is then calculated as the proportion of notes with errors. The test dataset is composed of oboe performance recordings that contain 162 music notes. The error rate of real-time processing without music score, real-time processing with music score, post-processing without music score, and post-processing with music score is 19.75% (14.81% after update), 3.70% (1.23% after updates), 13.58%, and 1.23% respectively. These performances prove to be adequate for our proposed applications.

6. REFERENCES

[1] H. Schenker and H. Esser (Ed.), and I. S. Scott (Trans.): *The Art of Performance*, Oxford University Press, New York, NY, 2000, pp. 3-6.

[2] G. Ren, J. Lundberg, G. Bocko, D. Headlam, and M. F. Bocko: "What Makes Music Musical? A Framework for Extracting Performance Expression and Emotion in Musical Sound," *Proceedings of the IEEE Digital Signal Processing Workshop*, pp. 301–306, 2011.

[3] M. Balaban, K. Ebcioglu, and O. Laske (Ed.): *Understanding music with AI : perspectives on music cognition*, AAAI Press, Menlo Park, CA, 1992.

[4] C. Raphael: "Representation and Synthesis of Melodic Expression," *Proceedings of IJCAI09*, pp. 1474-1480, 2009.

[5] A. Klapuri: "Introduction to Music Transcription". In A. Klapuri and M. Davy (Ed.): *Signal Processing Methods for Music Transcription*, Springer, New York, NY, 2006, pp. 3-20.

[6] G. Ren, J. Lundberg, G. Bocko, D. Headlam, and M. F. Bocko: "Generative modeling of temporal signal features using hierarchical probabilistic graphical models," *Proceedings of the IEEE Digital Signal Processing Workshop*, pp. 307–312, 2011.

[7] D. Koller and N. Friedman: *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, Boston, MA, 2009, pp.1-14.

[8] B. Moore: *An Introduction of the Psychology of Hearing, 5th ed.*, Academic Press, London, UK, 2000, pp. 160-165.

[9] E. B. Goldstein: *Sensation and Perception, 8th ed.*, Wadsworth Publishing, Belmont, CA, 2009.

[10] M. Müller: *Information Retrieval for Music and Motion*, Springer, New York, NY, 2007, pp. 85-139.

[11] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davis, and M. B. Sandler: "A Tutorial on Onset Detection in Music Signals", *IEEE Trans. Speech Audio Process.*, Vol. 13, No. 5, pp.1035 – 1046, 2005.

[12] M. Goto: "An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sound", *Journal of New Music Research*, Vol. 30, No. 2, pp.159-171, 2001.

[13] A. Klapuri: "Auditory Model-Based Methods for Multiple Fundamental Frequency Estimation". In A. Klapuri and M. Davy (Ed.): *Signal Processing Methods for Music Transcription*, Springer, New York, NY, 2006, pp. 229-265.

[14] H. Pang, D. Yoon: "Automatic Detection of Vibrato in Monophonic Music", *Pattern Recognition*, Vol. 38, pp.1135 – 1138, 2005.

[15] S. T. Smith: *MATLAB Advanced GUI Development*, Dog Ear Publishing, Indianapolis, IN, 2006, pp. 241-278.