Audio Player in Human-Computer Music Performance

Guanqiao Wang

ABSTRACT

Human-Computer Music Performance is a project for coordination and synchronization of different media to live human performances. Media include MIDI, music notation and audio. My work focuses on an audio player (and its communication with a "conductor" process). To be more specific, I create an interactive conductor-controlled real-time audio player based on predefined protocols and structures in HCMP such as time synchronization and arrangement, and other play modes independent of the conductor such as override, adjust tempo freely, and "tempo track," which scales tempo relative to the original tempo fluctuations.

Keywords

Real-time, computer accompaniment, Sound Processing, Interactive, Arrangement, Time Synchronization, Override

1.INTRODUCTION

Human-Computer Music Performance (HCMP) is a system designed to help musicians communicate with and perform with computers. Features include facilities to quickly restructure music (called "arranging"), tempo estimation and beat tracking based on foot tapping or other real-time indications of the beat, and the ability to control multiple "players" running on multiple computers (to allow multiple screens

of music notation and multiple channels of audio.) For synchronization, the conductor and players synchronize to a shared global clock. Components communicate in a simple way (using ZeroMQ, a communication protocol based on TCP/IP and sockets), using protocols that minimizes the impact of network latency. One of the key design elements is a clear definition of different "coordinate systems," e.g. there are score beats, reference beats, arrangement beats, and media beats, as well as clearly defined functions of mappings that establish relationships among all these different beat systems. To support decision-making and communication, HCMP systems will interact with musicians through a visual interface based on music notation, allowing users to point to measures to indicate locations and give cues before and during a musical performance.

There are currently three types of players in HCMP: MIDI player, score player and audio player. MIDI and audio players share similar functionality in HCMP in that they both produce musical sound. As a music accompaniment system, HCMP has to be compatible with a wide range of sounds, but MIDI can only represent musical sounds that can be synthesized. Audio, capable of playing every sound, including recordings of acoustic instruments, is expected to enhance the range of applications of HCMP by overcoming many of the limitations of MIDI.

Though it is similar to the MIDI player, the audio player has its unique parts, such as processing data with a phase vocoder and the use of a "label track," which provides the location of beats within the audio file (essentially providing the same information as a tempo track in a MIDI file). A phase vocoder is a tool for stretching time in (music) audio without changing the pitch. To maintain the quality of sound, which means no difference aside from the tempo or speed, the pitch of sound is of great importance here. If we were to simply speed up the number of sample played per second, the frequencies represented in the audio would change in proportion to the speed. This is much like changing the speed up a tape player: If you speed up the tape, the pitch is raised and the sound is unsuited to a musical performance. To solve the problem, the phase vocoder analyzes the frequencies in the sound using the Fast Fourier Transfrom (FFT) and reconstructs a sound using an inverse FFT. Time

stretching is accomplished by stepping through the input sound by smaller time steps, thus producing more frames of spectral data (results from the FFT) which in turn leads to a longer or stretched output. This kind of processing could actually be done without the FFT just by breaking up the input sound into tiny overlapping grains of sound and reassembling those grains with different overlap to stretch or compress the sound in time. The advantage of the FFT and the phase vocoder is that when the spacing of these overlapping grains is changed, there are artifacts caused by cancellation. E.g. if the overlap of two adjacent grains is changed by 1ms, a 500Hz frequency component, with a period of 2ms, will be shifted exactly half a cycle so that the frequency in one grain will tend to cancel the frequency in the next grain at the points where they overlap. By converting to the frequency domain, the phases within each grain can be adjusted to be in-phase with the previous grain. This greatly reduces artifacts due to time stretching.

The tempo track is an assistant file recording beats and their related time in the music file. The tempo track is used to map from the live performance position, measured in beats, to the correct audio playback position measured in seconds or samples. Every beat is indicated in the file (rather than simply specifying tempo) so that the audio player can handle music with non-constant tempo, including classical music and music recorded live without a click-track.

2. RELATED WORK

There are several HCMP components that have been previously implemented. Researchers at Carnegie Mellon and University College of London designed the whole HCMP framework and architecture that integrates different parts. Dannenberg and Russell have created the arrangement formalism and algorithms and implemented them in the score player, conductor, and MIDI player. The current Live Score Display component supports an interactive graphical interface in which new arrangements are made and sent to the conductor. Dannenberg and Ziheng have implemented a phase

vocoder that can stretch sounds with high quality. Russell has worked on communication between the score player, MIDI player and conductor, and documented the communication protocol. I am going to implement the audio player to operate much like the MIDI player, except that the sound representation will be sampled audio rather than MIDI.

3. Real-Time Playing

Real-time performance imposes a number of technical challenges. Sound cannot be processed and output immediately. Instead, samples must be computed ahead of real time and delivered to an output queue or double buffer to allow for a smooth uninterrupted flow of samples to the digital-to-analog converter. Otherwise, there would be audible artifacts caused by dropped samples. In the audio player, there are two threads, main thread and associate thread. The main thread manages the graphical user interface and periodically reads data from the music file into buffers so that the audio processing can access input samples with very low latency. The associate thread works to process and play data. We process a few data (approximate 64 floats) at a time. In the Macintosh OS X audio implementation, these 64 floats are double-buffered in the audio driver, so audio latency ranges from 64 to 128 samples. At a sample rate of 44,100 Hz, this is about 1.5 to 3 ms, and there is probably some additional latency imposed by the digital-to-analog system.

As for real-time accompaniment, computers synchronize by sensing the tempo and current beat, then adjusting the output to match. Currently, there are no known algorithms to sense beats reliably by “listening” to audio, so we create an assistant label file for each music file to record each beat and related time by ear. For example, we can listen to the music and manually tap a key to indicate timing. With a label file, we can check the current tempo (normal tempo), calculate the ratio based on required tempo (live tempo) and change the playback speed to match it using the phase

vocoder.

There are three play modes within audio player: tempo track, override and conductor. In tempo track (this name is based on the behavior of the MIDI Player, where the MIDI file tempo track is used to determine the tempo), we simply scale the playback speed according to a slider. In override, we pick the tempo as we want with a slider. In tempo track mode, the slider actually represent tempo *scaling*, whereas tempo in override is directly equal to the value of slide bar. Tempo track mode retains expressive tempo deviations in the audio file while override removes tempo deviations, producing a constant tempo. The third mode is conductor, in which the player is under the control of a conductor process through the predefined protocols.

4. AUDIO PLAYER

In HCMP, when we are talking about synchronization and coordination, timing is expressed as a mapping from real time to beats. Therefore, the role of conductor in real-time accompaniment is more like a sensor, feeling and telling the current live beat and tempo. However, before we talk about communication, the prerequisite lies in how players and conductor can share the same time feeling.

4.1 Time Synchronization

In order to work together, the conductor and players rely on a shared global clock. Because the conductor and players may be running on different computers separated by some communication delay, it is not always possible for players to read the true global clock (maintained by the conductor) with any accuracy. The solution is for players to synchronize local clocks to the global clock. This can be done in spite of network latency, and once clocks are synchronized, all processes have access to the same time base with low latency. To synchronize clocks, players send synchronizing message every few seconds (suppose at t1) and receive the message carrying time

(suppose t3) back from conductor (at t2). The round trip time is t2 – t1. The one-way time can be estimated by (t2 – t1)/2. Thus we can assume that t2, measured on the player's clock, corresponds to t3 + (t2 – t1)/2 on the conductor's clock.

This would be all we need if we could assume that messages were never delayed. In practice, messages can be dropped or delayed, so the clock time estimates may have a large error. Here, we assume in situation with higher latency, the time to and from conductor may not be the same, causing the estimate to be wrong. To eliminate this problem, we assume that in any sequence of, say, 10 clock estimates, at least one estimate is good. The best estimate will be the one where the round trip time t2 – t1 was the smallest. So we record the history of clock synchronization messages and choose the smallest round trip from the 10 or so recent round trips. In practice, minimum round trip times in our system are a millisecond or two and clock synchronization is accurate to within 1 ms. This is about an order of magnitude smaller than the requirement for music synchronization. After all, sound only travels about 1 foot per millisecond (0.3m/ms), so simply by standing a meter apart, musicians experience delays and asynchronies of several milliseconds.

4.2 Beats Communication Protocol

Another concern is the method of beat synchronization. All the players and conductor in HCMP are designed to share the current tempo and beat using a linear function method to avoid problems of communication latency. In other words, the conductor does not broadcast the information of every beat, which would then arrive late at the players due to network latency. Instead, the conductor broadcasts a time-to-beat mapping function that contains not only the current beat but also the tempo. This mapping function can be used to compute the beat for the foreseeable future. Furthermore, the computation of the current beat uses the local synchronized clock, so as long as the mapping function is up-to-date, communication latency will not have any impact on synchronization. Imagine a bad connection between one player and conductor: The player can still move on with the last mapping until the new one

comes. In contrast, if players relied on periodic and frequent clock ticks from the conductor, it would be impossible for players to keep playing in the event of a bad connection or even a momentary failure such as a lost packet. The assumption that networks are low-latency and always reliable should definitely be avoided in real-time music performance.

4.3 Beat Layers

From tempo sensing to media processing and delivery in the players, different contexts require different “kinds” of beats. For example, beats in a midi file may not be numbered the same as corresponding beats in music notation or counted from a foot pedal tapping the beat. Thus there are four beat layers made in three phases to account for all the different contexts. This makes the HCMP system more extensible and able to synchronize between different players. The top layer is called performance beat; it is the actual beat in live performance. The performance beat begins when a regular beat is established, e.g. by tapping on a foot pedal. The second layer is the arrangement beat, which is designed for flexibly arranging or restructuring the score (this beat is optional and can be ignored if the music is played according to the original notation). The third layer is the reference beat or flattened score beat in the score player. The reference beat represents a nominal performance of the score. Typically, you have to “unroll” the repeats and loops in the score to create the reference beat form of the score, and there is a mapping from reference beats to score position. This mapping is many-to-one in the case of repeats. The fourth layer is media beats, which is the actual beat in the audio file. The purpose of media beats is to account for the fact that audio recordings and MIDI files may not have the same beats as the score or reference beats. Typically, MIDI files have a measure or two of silence in the beginning, thus you might have to offset the reference beat by a measure or two to find the corresponding beat in a MIDI file.

Imagine a situation, in which a live singer might enter a bar late and the band adjusts by repeating a bar. Alternatively, imagine that the bandleader decides to start a

piece of music by repeating the first measure twice. These cases can be represented by an arrangement that maps the second measure of arrangement back to the first reference beat. If each measure has 4 beats, the arrangement beat is currently at 8 beat after playing it twice, whereas the reference beat is currently 4. Moreover, in the audio player scope, the audio file might have two measures (4 beats in each measures) of silence at the beginning, so the actual sounds may come out at media beat 8. In this case, when it received the command to play the first and second measures (each from reference beat 0 to 4), it should play from media beat 8 to 12 twice. The existence of these different beats is necessary, because conductor can uniformly represent beats (reference beats and arrangement beats) and each player can play their individual media beats accordingly.

4.4 Widgets for user

Except the basic functionality like pause and stop. The software also provides some conveniences like starting music wherever beats you want by passing first beats you want at the “Start Beats blank”. Display the current tempo beat in the GUI for debugging. Four different modes for different purposes, Tempo track indicates playing the music locally according to the music file (solely on media beats). Override modes even gives user the power to create their real own tempo which is not a ratio of original tempo. Conductor is the mode for playing according to the conductor by ZeroMQ wrapping packet. Label mode for creaking label files without typing time stamp and index of beats manually, with this mode user can tap space bar to mark for the according beats while listening to the sound and this will create a label file with detailed beat information for actually play.

5.CONCLUSIONS

Basically, I have implemented basic parts of the audio player, such as client-server

communication, real-time audio stretching with multithread, label creating/reading, beats mapping from four different layers, four different playing modes, real-time tempo calculating and so on.

6. ACKNOWLEDGMENTS

Thanks to Roger who gives me the chance to taking part in such an interesting music accompaniment system and a brilliant experience working in the top college as CMU. I really learn a lot from him. His eagerness in music along with accurate and wide knowledge in computer inspires me to be a geek on details. Thanks for his help and guidance during the work and hopefully I can finish the remaining part with him for another one month and a half. Also thanks to my college Beihang University, which gives me the freedom and support to do some research I am interested in.

7. REFERENCE

[1] Roger B. Dannenberg, Guangyu Xia and Dawen Liang: A Framework for Coordination and Synchronization of Media. *Proceedings of the International Conference on New Interfaces for Musical Expression, 30 May - 1 June 2011, Oslo, Norway*

[2] Mark Dolson: The Phase Vocoder. Computer music Journal, Vol. 10, No.4 (Winter, 1986), pp. 14-27

[3] Roger B. Dannenberg, Andrew Russell: Arrangements: Flexibly Adapting Music Data for Live Performance.

[4] Dannenberg, R. Real-time scheduling and computer accompaniment. In Current Directions in Computer Music Research, edited by Max. V. Mathews & John R. Pierce, MIT Press, Cambridge, MA, 1989, pp. 225-261.

[5] Dannenberg, R. New interfaces for popular music performance. Seventh International Conference on New Interfaces for Musical Expression: NIME 2007, New York, NY, 2007, 130-135.

[6] Connick, H. Jr. System and method for coordinating music display among players in an orchestra. US Patent #6348648 (2002).