A Human-Mimicking Music Practice **Companion**

Tim Nadolsky, Caasi Boakye, Rixin Chen, Nic Hornung, Temirlan Karataev, Nick Ping-Hung Ko, Minsoo Oh, Calvin Tseng, Shrish Senthilkumar
Dr. Kristen Yeon-Ji Yun (Music), Dr. Yung-Hsiang Lu (ECE)

Check us out online!

PURDUE UNIVERSITY College of Engineering

Vertically Integrated Projects

Problem Statement

When preparing to play ensemble pieces, it's imperative for musicians to practice on their own to maximize efficiency during often-limited group sessions, which is challenging because they can't reference their fellow musicians' playing styles during solo practice.

To solve this problem, we are creating a tool called the Companion, which is an AI-embedded tool for assisting musicians in string players' ensemble practice. Companion's goal is to be able to replicate a human player's ability to match the tempo/beats of other players, change their playing style, articulations, and expression on the fly, as well as respond to verbal and gestural commands from other players to remain in sync/take instructions.

Prior Work

Many music accompaniment apps such as SmartMusic, MyPianist, Music Plus One (IU), and Yamaha's AI music accompaniment systems exist with tempo/beat tracking and synchronization features. However, these apps usually have a significant learning curve, and their expression controls are limited to tempo and volume.

Future Work

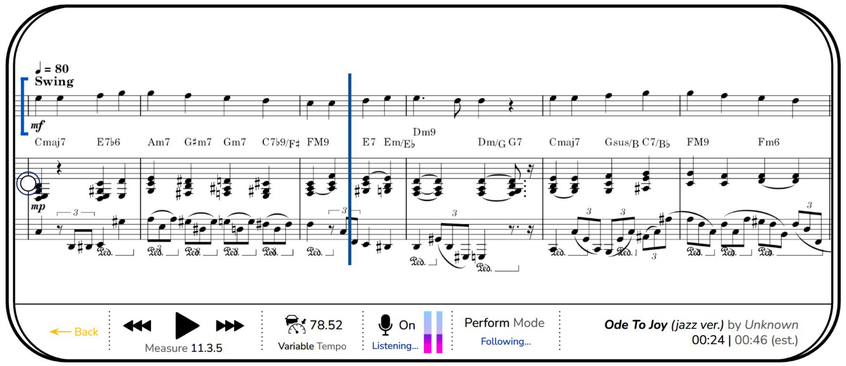
Further development is needed in accompaniment audio editing techniques - specifically, greater audio quality and matching the source audio's reverb is a goal, as well as the ability to "learn" to edit/change expression of more instruments without requiring explicit human software design.

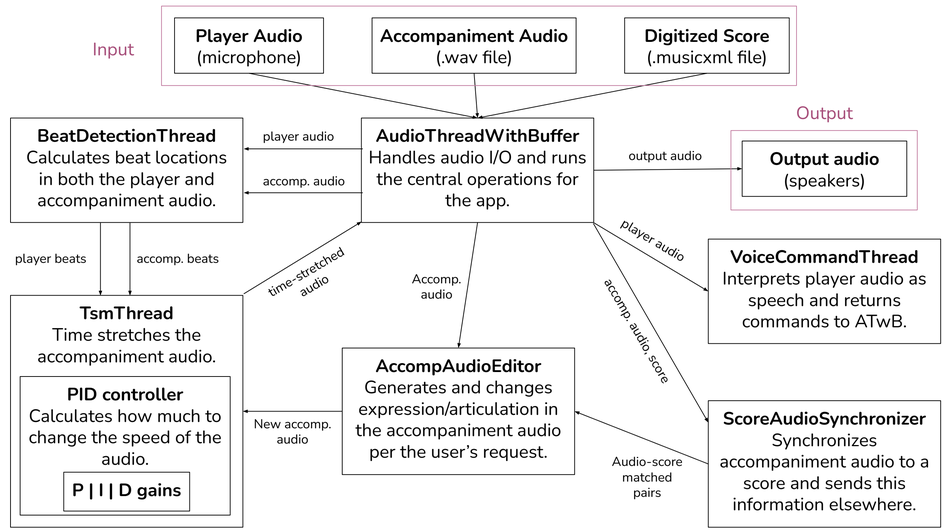
Also, once the current LLM voice commanding model is fully developed, we want to start incorporating face/pose tracking from our sister project Evaluator to provide gesture control for Companion.

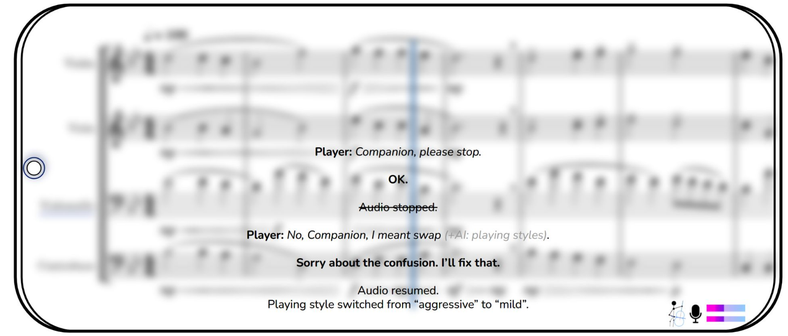
Acknowledgements

This research team would like to thank professors Dr. Kristen Yeon-Ji Yun (Music) and Dr. Yung-hsiang Lu (ECE) for mentoring and providing financial support for this project. This research was supported by NSF Proposal #2326198.

Overall system diagram







Interaction

To ensure that Companion is as easy to use and work with as a real human player, we've adopted a combination GUI-voice interaction model where a user can use both a GUI or voice commands to interact with Companion.

Voice commanding:

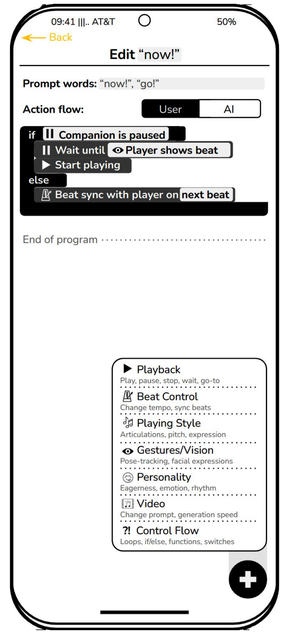
At first, we attempted to use syntax trees to parse user commands, but then one of our members had the idea to simply feed ask an LLM to parse the user's prompt instead. In this vein, we currently utilize the zero-shot classification model Roberta to process text-converted spoken prompts. For example, if the user says "it's too quiet", the LLM asks our Voice Dictionary class for a list of possible commands, and matches the prompt to commands from that list. The system not only translates direct commands but also interprets the intent behind natural language.

The user can also edit the dictionary of possible commands, increasing Companion's vocabulary and interaction possibilities.

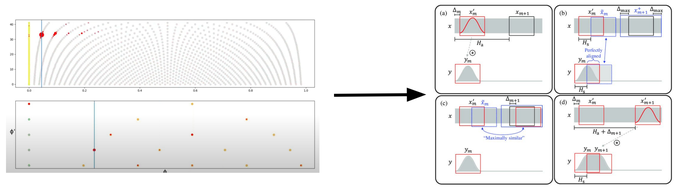
The system's capacity to recognize and process a broad vocabulary of terms related to music practice ensures that users can communicate naturally and effectively with the Companion.

GUI:

Although we haven't done much work yet, several possible examples of the GUI have been graphically designed and placed on this slide.



Beat detection and tempo tracking



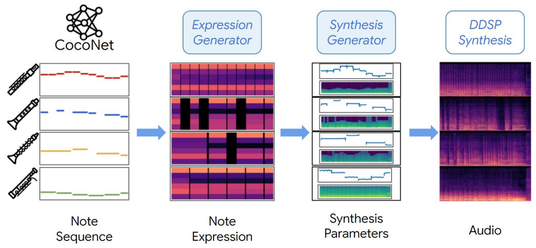
Our beat detection algorithm currently works by finding peaks/onsets in a spectrogram of the audio, but we are currently looking into using **BeatNet**, a CRNN network which can do online and efficient beat and tempo tracking.

We feed the detected beat positions in both the input and accompaniment audio into a **PID controller** to generate a speed gain for the accompaniment audio, with error calculated as the difference in number of beats between the input and accompaniment audio.

Time stretching algorithms such as Waveform Similarity Overlap-Add (**WSOLA**) are then used to accurately speed up/slow down the accompaniment with little artifacting.

These technologies enable Companion to follow the human player's tempo, and play the accompaniment in time with the player.

Accompaniment audio editing

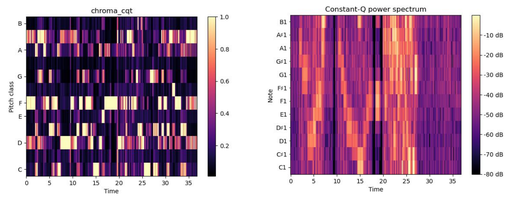


To edit accompaniment audio, we use **MIDI-DDSP**, developed by Google's Magenta team. MIDI-DDSP synthesizes string audio from MIDI by creating human-interpretable expression parameters using an autoregressive RNN, which are used to generate audio using a GAN.

We edit articulations/expression in the audio by generating the expression parameters from an articulation-annotated MusicXML, interpolating them to appropriate values based on the MusicXML, and synthesizing the audio from those new, edited parameters.

Being able to edit the accompaniment audio enables Companion to play in different styles at the player's request, enabling Companion to be more expressive than a simple beat tracking app.

Score-audio alignment



To accurately edit and understand the accompaniment, Companion needs to understand which samples of audio map to different notes in the score. The steps are:

1. Extract note names (frequency) from musical score.

The Python library Music21 is used to extract notes played in the musical score.

2. Extract chroma features from the audio.

We utilize either a Constant-Q transform or chroma feature to provide us with info on how the energy of different frequencies (corresponding to notes) changes over time in the accompaniment audio.

3. Apply techniques like dynamic time warping to align note timings.

Dynamic time warping is a dynamic programming approach to aligning a score with chroma audio features. The problem exhibits optimal substructure, so we can use a recurrence based on aligning one note w.r.t. all the previous notes.