Mobile Maestro: An Interactive Conducting Application for Android

Mobile Maestro, an app for Android, is intended to be a simple tool for conducting practice and self-accompaniment. It uses hand gestures and smartphone accelerometer capabilities to alter the playback dynamics of a piece of music in real time. The traditional approach for a tool like this uses sample-based instrument synthesis (such as that employed by SoundFonts\(^1\)) to play MIDI files generated from stored scores of specific pieces of classical, jazz or pop music, adjusting the volume (and/or the speed) of the music to match the gestures of the conductor/user. To distinguish my efforts from such previous applications, this app streams video directly from YouTube using the YouTube Android Player API in conjunction with a project-specific API key. Users can navigate to different YouTube videos within the app by specifying a video's unique ID (the sequence of numbers and upper- and lowercase letters after the last equal sign in a YouTube URL). The ratio of the absolute magnitude of the smartphone's acceleration in the x and z dimensions (while the phone is held sideways in one's hand - see Figure 1) to the maximum measurable acceleration then determines the playback volume (set proportionally to maximum device volume). An added benefit of this behavior is that, while

---

\(^1\) https://en.wikipedia.org/wiki/SoundFont
video advertisements are not blocked, they are very quiet if the device is at rest (assuming that users will not wish to conduct ads). Common YouTube features such as play/pause, seek (using the progress bar on the touchscreen), screen rotation, full-screen mode, and video playback quality options are supported. Inspiration came in part from this tutorial on embedding video with the YouTube API and from these two accelerometer usage examples.

This project could serve both as a device to practice conducting and as a self-accompaniment tool for vocalists who wish to change the volume of the orchestra/band/etc. track from moment to moment without needing to interact with individual buttons or sliders. The advantage of adding the interface to stream performance videos rather than using pre-stored MIDI tracks lies in the potential base of source material. YouTube offers a vast range of recordings to choose from that would be extremely difficult to match if each piece of music used in the app required a score to be stored on the device itself before it could be conducted. In addition, it creates further possibilities as to genres of music that can be conducted realistically: while some musical instruments can be simulated in a relatively convincing manner, the human voice often remains elusive in this regard, especially concerning the pronunciation of sung words. By using actual performances instead of simulations, more realistic opera and even pop music is available, hopefully resulting in a more satisfying conducting experience.

Similar existing conducting implementations include the Airconductor app, also for Android, and the interface vMaestro, among others, although these alter tempo rather than dynamics. Tempo alteration through metered hand gestures was my original goal, but my need
was to alter tempo without altering pitch -- a phenomenon properly known as time stretching\(^7\) -- and while YouTube offers this functionality on certain desktop browsers, it does not yet offer a similar service for mobile browsers or applications. I had also hoped to automatically locate beats and determine the original number of beats per minute (bpm) of any musical performance on YouTube in order to sync the piece with the conductor. The concept of beat detection is, of course, not new in itself; methods range from note onset autocorrelation (as described in this paper\(^8\) from 1993), which isolates a voice or voices for analysis and which can be applied either to scores or to performance recordings, to more sophisticated models that are capable of processing and resolving multiple conflicting tempo/meter hypotheses (note this evaluation of the interactive beat tracking system BeatRoot,\(^9\) introduced in 2001). When this proved infeasible due to the necessity to process the audio offline before playing it back -- which would involve directly downloading YouTube content onto devices, a practice that is highly discouraged if not illegal -- I planned to ask the user for an estimate of the song's bpm and perform my calculations using that information. This would allow the user to input manipulated bpm figures to allow conducting at half tempo, double tempo, or any other multiple or fraction that he or she wished. The bpm data could be obtained by using a manual tap-to-determine-tempo application, either on desktop\(^10\) or mobile,\(^11\) or could be retrieved from an online database.\(^12,13\) However, the nonexistence of time stretching capabilities unfortunately made this idea impracticable for the present.

---

\(^7\) https://0110.be/posts/Audio_Time_Stretching_-_Implementation_in_Pure_Java_Using_WSOLA
\(^8\) http://academics.wellesley.edu/Physics/brown/pubs/meterACv94P1953-P1957.pdf
\(^10\) http://www.beatsperminuteonline.com/
\(^12\) https://songbpm.com/
\(^13\) https://www.cs.ubc.ca/~Davet/music/bpm/index.html
In light of these considerations, I chose not to publish the Mobile Maestro app publicly on the Google Play store, at least not until there exists a method to change the tempo (and not the pitch) of streamed YouTube content, presumably without relying on the YouTube Android API. The entire body of Java code for the app is available from the home page of the Mobile Maestro website, however, if any are interested in adding to it or running it on their own Android devices. I would not recommend attempting to run Mobile Maestro on Android emulators, since these do not seem to behave well with the YouTube API. Simply follow the instructions\textsuperscript{14} for installing the latest version of Android Studio, uncompress the MobileMaestro.zip file, and open the project directly; or, if that does not work, try importing the project following these instructions.\textsuperscript{15} The Mobile Maestro app can then be run from Android Studio on any device running Android 4.0 (Ice Cream Sandwich) or later, connected to one of the desktop's USB ports. If you or your device are new to running apps from Android Studio with a USB connection and a debug key, these web pages\textsuperscript{16,17} are helpful.

\begin{footnotesize}

\textsuperscript{14}http://developer.android.com/sdk/installing/index.html
\textsuperscript{15}https://www.jetbrains.com/help/idea/2016.1/importing-an-existing-android-project.html?origin=old_help
\textsuperscript{16}http://developer.android.com/tools/device.html
\textsuperscript{17}http://android.stackexchange.com/questions/71761/how-to-regenerate-create-adbkey-and-adbkey-pub-from-the-command-line

\end{footnotesize}