VioScribe: A Music Notation App for Android

CSE 481i: Sound and Media Capstone, University of Washington

Lauren Martini
lmartini@uw.edu

Jack Eggleston
jegg13@cs.washington.edu

ABSTRACT

VioScribe is an Android app intended for use by violinists for music transcription. Audio from the violin is taken in through the microphone of a mobile phone and then processed to detect pitch. A small accelerometer/gyro unit attached to the end of the bow is also used to determine when the bow's direction changes. This information is combined to create a rough score, which is displayed on a music staff in the app. VioScribe vastly increases the ease of composition for violinists and other musicians.

1. INTRODUCTION

Composition can be an arduous task. Consider a musician playing their instrument who suddenly thinks of a melody they would like to write down. They would have to stop playing, run and grab some blank sheet music (or boot up their computer and open composition software), then painstakingly play a few notes, write them down, play the next few notes, write them down, etc. This time-consuming processes is hardly ideal, as each time the composer stops playing to write out the next few notes, they run the risk of forgetting the rest of the melody.

With these issues in mind, we set out to develop a mobile app that could generate a score from audio. This would allow the user to notate without having to stop playing, drastically decreasing the time required for composition. Additionally, we decided to narrow the focus of the project to violinists, adding a sensor to be attached to the bow that detects changes in bow direction. Combined with pitch detection, this system provides a way for violinists to quickly and painlessly obtain a rough score for a simple melody.

2. SOLUTION

Our solution utilized a Bluetooth LE-enabled Android device with a microphone and a Metawear sensor capable of connecting to that device. Interactions between hardware components are displayed in the architectural diagram in Figure 1. Pitch detection and notation display utilized pre-existing libraries.

2.1 Components and Architecture

The application was developed for an Android Mobile device, using Android Studio. All testing was performed using an Android Moto E4, running Android 7.1.1 (Nougat).

For detecting bow changes, we chose to use a MetawearC Streaming Sensor. This sensor streams data through Bluetooth LE (low energy) and contains an accelerometer and gyrometer, among other units. One of the major benefits of this sensor is its size at 2.5 cm in diameter. This makes it small and light enough to be attached to the tip of the violin bow without a significant impact on the balance of the bow. A button on the user interface is used to initiate connection with the sensor.

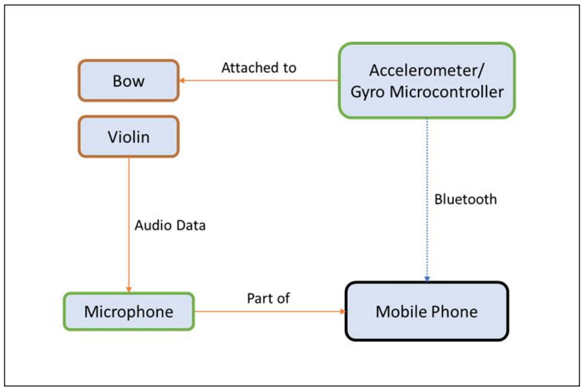

Figure 1: An architectural diagram demonstrating the interactions between components.

2.2 Pitch Identification

For pitch detection, we used the TarsosDSP audio processing library. TarsosDSP offered a selection of pitch detection algorithms, but we chose the McLeod Pitch Estimation Algorithm[1].

2.3 Notation Display

Our notation display was done using abcJS, a Javascript implementation of the popular abc music notation system. The score was displayed using a script embedded in an HTML page, which is then loaded into a WebView in the actual app.

2.4 Bow Change Detection

We used the accelerometer and gyrometer modules of the MetawearC sensor to detect bow changes simply by watching for large accelerations indicating a reversal of direction. In the future, an IMU with an accelerometer, gyrometer and magnetometer would probably be ideal for using sensor fusion to more precisely detect bow changes.

2.5 User Interface Design

The user interface has five main components that are currently functional, as seen in Figure 2. The note currently detected by the app is displayed in a textfield at the top of the screen. Below that is a field that displays the score, which has to be manually scrolled to the right to see new notes at this point in time. Below that is a button for initiating a bluetooth connection with the sensor, a button for saving the current score (as a PDF) that has not been implemented yet, a button for clearing the current note and score, and a button for recording audio that has not been implemented yet.

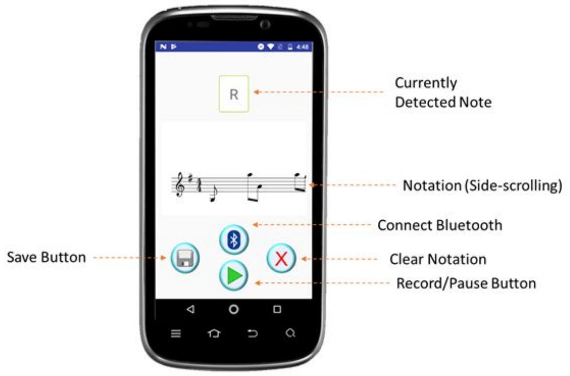

Figure 2: The user interface with some notation displayed. The currently detected note is “R”, indicating a rest.

3. PRIOR ART

There are a number of applications on both Android and iOS which seek to transcribe music to notation, including Accurate Music To Note Decoder, MusicTrans, and Music Transcribe. We sought to differentiate our application from these in two specific ways. The first was a focus on one particular instrument - the violin. This allowed us the freedom to implement violin specific features, such as using an accelerometer on the bow to increase the accuracy of the music transcription. The second was ease of use. Most of the apps we were able to find had ugly, complicated UIs, which can make them intimidating to inexperienced musicians. Our interface was much simpler. The user can see the current note, a transcription of the current song, and two control buttons. Due to this, our application is much more approachable to the novice, which is the audience we hoped to pursue.

4. CONCLUSION

For pitch detection, we evaluated success by two factors: accuracy and speed. TarsosDSP gave us high accuracy from the start, but with a large delay. As we wanted to be able to update the notation display in as close to real time as possible, we spent a lot of time refining the algorithm to reduce the delay between the note being played and the pitch being identified. This aspect of the project ended up being a success, as the delay has been reduced to a much more manageable level.

Our metric for the notation was much simpler - is the notation accurate? Unfortunately, we hit a major stumbling block in our project, which was finding music notation libraries that were compatible with Android. We only managed to get the score to show at all in the last week, leaving us little time to perfect it. As a result, the notation missing several features, such as measure bars and variable note lengths. However, while this part of the project was only a partial success, we believe it will be fairly easy to tune the notation display to be completely accurate for simple transcriptions, requiring maybe one or two more weeks of work.

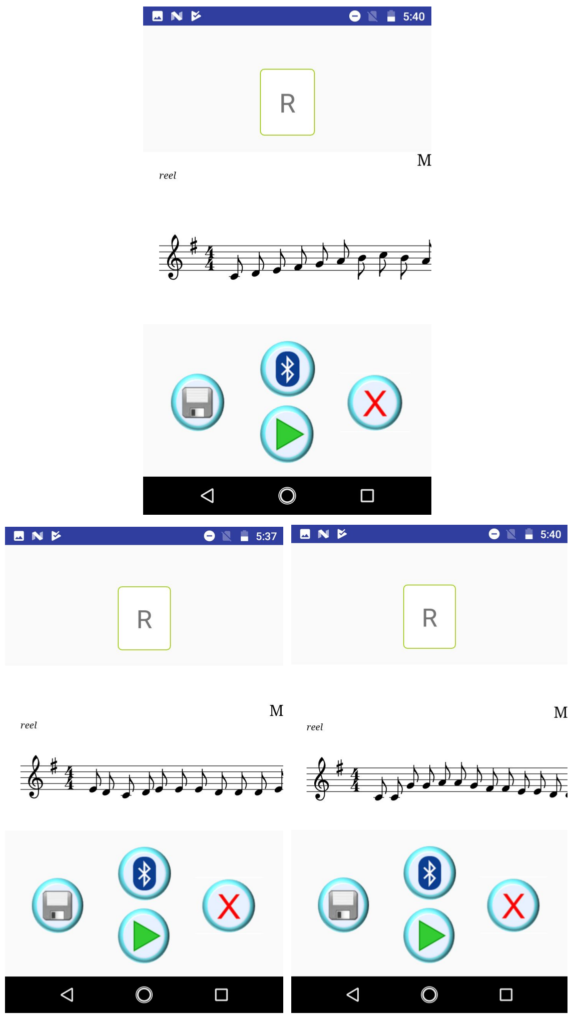

Figure 3: Example scores. The top score is a C major scale, played without the sensor. The bottom left score is "Mary had a Little Lamb" and the bottom right is "Twinkle Twinkle Little Star", both played with the sensor.

5. FUTURE WORK

Our ultimate goal is to make the app available for download on the Google Play store. Before doing so, we would like to increase the accuracy of pitch, rest and bow-change detection even further. We would like to add a tempo detection feature, where the app automatically determines the piece's tempo and time as the piece is played.

Another avenue to explore would be using machine learning to enhance the accuracy of note identification and rhythm determination. Additionally, the app currently works best in very quiet environments. In the future, we could add more sophisticated noise filters to improve the accuracy of pitch detection in areas with background noise. We would also like to port this app to iOS, as well as adding the ability to save scores not only as PDFs but also as files compatible with composition software, such as Musescore or Finale.

For an entirely separate future project, it may be interesting to try and develop a music notation display library in Java for Android, since surprisingly there does not currently seem to be a working library for that purpose that was easy to put into use. This would also help with the goal of making the scores generated by VioScribe compatible with other music composition software.

REFERENCES

[1] McLeod, P., & Wyvill, G. (2005, September). A Smarter Way to Find pitch. In ICMC.