ECE 1778 Creative Applications for Mobile Devices

Final Report- Acoustica

Akshay Gill
Amanjot Kaur
Nitin Guleria

Contents

1 Introduction

Acoustica is an android app that aims to help amateur musicians in composing music. Music composition requires extensive ear training and knowledge of music theory. Acoustica attempts to reduce this steep learning curve by assisting the user in making a transition from a melody to a complete song composition. The app also helps solo musicians by giving them access to more instruments for their composition. The scope of the app for this course was to provide backing percussion based on an input melody provided by the user.

2 Overall Design

Figure 1 sums up the design on Acoustica succinctly. The melody input by the user goes through 4 stages namely, Recording Threshold stage, Onset Detection stage, Pattern Matching stage and Stitching stage before finally being output to the user.

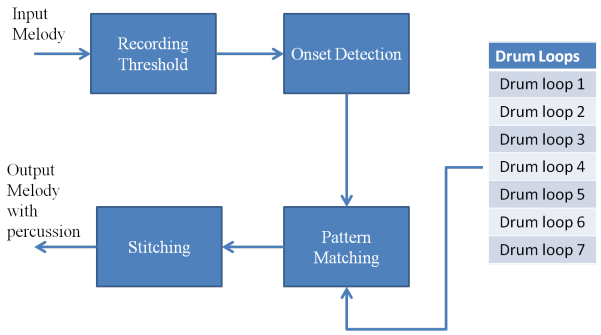

Figure 1: Diagram showing the four processing stages of the app

2.1 Initial setup

The user plugs his/her headset into the phone. It should be noted that once the headset is plugged in, the microphone on the headset will be used for recording. The user can now start recording by pressing the "Start Recording" button. A click track is played into the headphones at 120 BPM[1]. The app functionality has been tested to work well with a guitar input at this point. Melodies with higher frequency notes are also observed to perform better.

[1]Percy Scholes, et al., "tempo," The Oxford Companion to Music. Oxford Music Online, Oxford University Press,`http://www.oxfordmusiconline.com.myaccess.library.utoronto.ca/subscriber/article/opr/t114/e6699` (Accessed 8 April 2014).

2.2 Recording of the Input Melody

The recording should be done in a noise free environment for ideal functioning of the app. In order to give the time to user for set up, the recording starts when input sound crosses a threshold. This helps in synchronized recording for the application.

2.3 Onset Detection

The human brain interprets a beat as a sudden change in energy of the sound signal received by the ear. There are two common techniques of beat detection, namely, frequency based beat detection and sound energy beat detection. Both methods work on the principle described above, but the difference lies in the fact that the frequency energy mode splits the spectrum into different frequency bands, each representing a part of a drum beat. This is helpful in differentiating between different parts of a drum beat, for example, kick, snare and cymbal[2].

In Acoustica, the onset detection is done on the input melody which is a single instrument, like a guitar or piano playing single notes. As such, there is no drum component to the incoming sound, so there is no real benefit of using the frequency beat detection technique. Instead, the algorithm uses the sound energy method which detects a sudden change in sound energy irrespective of its frequency.

In Figure 2, the circles represent the onsets or beats that were detected in an input sound sample. Each one of the sections separated by the red lines is a bar[3] in a sound sample played at 120 BPM.

In this example, the onset detection algorithm detects 4 beats in the first bar, 4 in the second, 3 in the third and 4 in the fourth bar. Therefore, it decodes the pattern of the input melody as 4434 and passes that to the pattern matching algorithm to find the closest drum beat that matches this beat pattern.

2.4 Pattern Matching with Drum Loops

The pattern of the input melody is matched with the pattern of drum loops that the app has in its database. This is done by an algorithm which picks the closest available drum loop which would sound good with the melody based on its beat pattern. It was not possible to provide all combinations of drum loops because the size of the app would become too large. Thus, we decided to provide the most commonly used drum patterns.

The pattern matching algorithm works on a principle that a drum loop will sound good with the input melody if the beat pattern for the drum loop is either the same or smaller than that of the input melody in each part of a 4/4 time signature[4]. The

[2] "Drum kit." Princeton University. `https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Drum_kit.html` (accessed April 10, 2014).

[3] David Hiley. "Bar." Grove Music Online. Oxford Music Online. Oxford University Press, accessed April 10, 2014, `http://www.oxfordmusiconline.com/subscriber/article/grove/music/01972`

[4] Richard Rastall. "Time signature." Grove Music Online. Oxford Music Online. Oxford University Press, accessed April 10, 2014, `http://www.oxfordmusiconline.com/subscriber/article/grove/music/27980`.

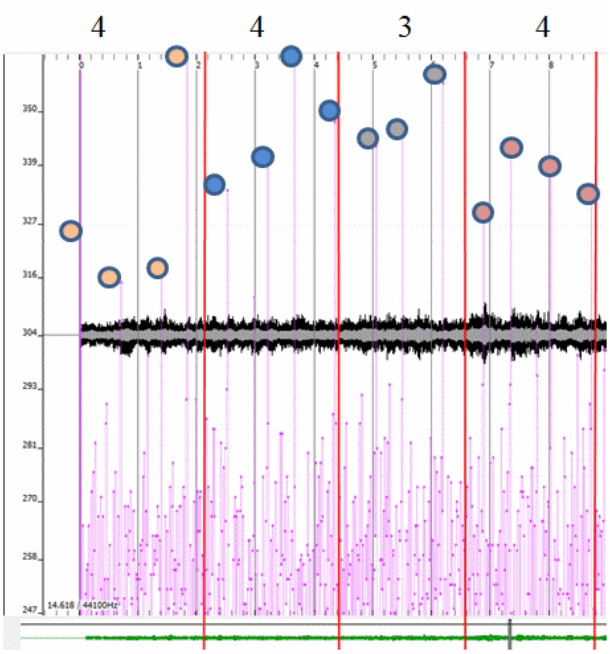

Figure 2: Input melody sample played at 120BPM showing onset pattern in first 4 bars of 4/4 time. Horizontal axis depicts time and vertical axis represents sound energy.

algorithm can be detailed as follows:

Step 1: Check for an exact match for the beat pattern of the melody in the drum loops.
Step 2: Reduce a digit by 1 and try to find a match. Do this for each of the digits one at a time.
Step 3: Reduce a digit by 2 and try to find a match. Do this for each of the digits one at a time.
Step 4: Reduce 2 digits by 1 and try to find a match. Do this for each pair of digits.
Step 5: Repeat Step 4 by reducing 2 digits by 2.
Step 6: Repeat Step 4 by reducing 2 digits by 3.
Step : If none of the above find a match, return the default loops as a match.

Another alternative approach we tried was to choose the drum loop pattern which has least absolute difference with the beat pattern of the melody. This method works well but it doesnt ensure that the drum will not take over the composition and make the melody sound like backing track.

Once a match is found, the two files, namely the input melody and the selected drum loop are stitched together.

2.5 Stitching

The stitching algorithm takes the melody and drum loop as '.raw' audio files and provides an output file in '.wav' format .This is done by attaching the '.wav' format header to the summation of two input raw files.

The drawback of summation is that the output audio may have clipping [5] if the input audio files are loud. Currently the app does not check for clipping, so the output music may sound louder at certain parts in comparison to the single input melody.

2.6 Music Library

The app provides the user the ability to play and retrieve any previous output recordings in .wav audio format.

3 Statement of Functionality

The following are the working features of Acoustica:

- Recording the input melody.
  - Current Limitation: The input melody must be recorded at 120BPM.
- Playing a click track while the user is recording the melody to set the tempo for recording. This click track has been commented out in the code currently to allow the user to test the app without needing a headset.
- Synchronization between drums and melody by starting the recording only after a minimum threshold of sound has been crossed.
  - Current Limitation: The synchronization might not perform optimally in a noisy environment.
- Visualizer to depict the sound being recorded.
- Extraction of beat pattern from the melody.
  - Current Limitation: The onset detection shows a +/- 1 beat error per bar of music.
- Pattern matching of beat pattern to a suitable drum loop.
  - Current Limitation: The pattern matching algorithm will need to be tweaked to ensure that all possible combinations of commonly used drum loop patterns are included.
- Stitching the melody and a drum loop.

[5] "Extensive compressor reviews and FAQ." Distortion, Clipping, and Square Waves. `http://www.ovnilab.com/articles/clipping.shtml` (accessed April 10, 2014).

- Adding percussion track to the melody input by the user.
- Playback of the composed music as well as from the library.
- Writing output song in '.wav' format.

4 Graphical User Interface

4.1 Home Screen

Figure 3: Home Screen

The first page allows the user to start recording a melody or to exit the app. The background and font are chosen to create a fun experience. (Figure 3)

4.2 Music Recording

This screen follows after the user presses Start Recording. The visualizer appears on the screen once the recording threshold has been crossed. The horizontal axis shows frequency in Hertz and the vertical axis shows audio levels in decibels. The user can stop the recording at any time by pressing the Stop Recording button.(Figure 4,5).

4.3 Processing Dialog

Once the user hits the Stop Recording button, the onset detection algorithm detects the pattern of drum beats in the input recording. This usually takes a few seconds

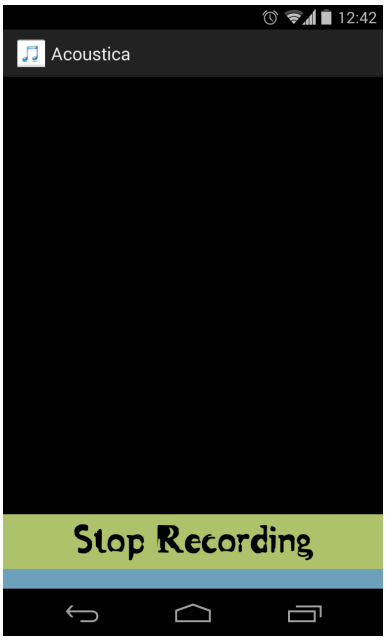

Figure 4: Recording Screen when sound threshold has not yet been crossed.

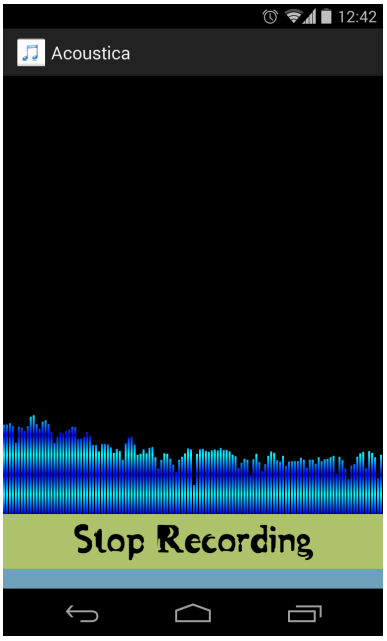

Figure 5: Recording Screen after sound threshold has been crossed and recording has started.

depending on the length of the recording and we see a processing dialog at that time. Once this process is complete, the user is shown the Playback Screen.(Figure 6)

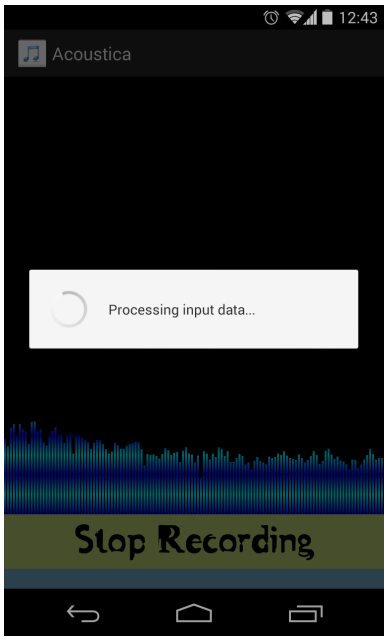

Figure 6: Processing dialog when the beat detection is in play.

4.4 Playback Screen

Figure 7: Playback Screen before stitching has been done. Play button is inactive.

This screen shows three buttons: Create, Play and Music Library. Initially, the Play button is inactive(Figure 7). When the user hits the Create button, the pattern matching

Figure 8: Playback screen after output has been created. Play button is active.

algorithm finds the closest drum loop to the input melody, stitches them together and stores the output in wave audio format. Once the output file has been produced, the Play button becomes active and the user can listen to the composed song by hitting the Play button(Figure 8). Also, by pressing the Music Library button, the user gets a list of their previous compositions.

4.5 Music Library

The Music Library button on the Playback Screen shows a list of previous songs that the user has composed using this app. Any of the entries in the list can be played by pressing on the item in the list. The playback stops on pressing the back button or by selecting another recording (Figure 9).

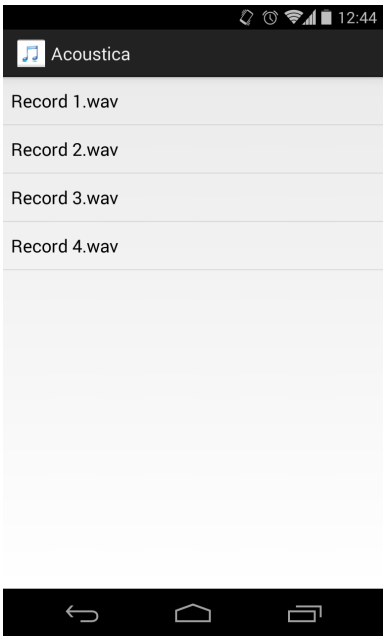

Figure 9: List of previous compositions to play.

5 Key Learning

We have learnt a few lessons during this project and they are as follows:

5.1 Working on Existing Code Base

- If basing the app on an existing code base, it should be tested for robustness before building on top of it.
- Trade off between using the open source libraries and developing algorithms from scratch should be taken into consideration.

We started with understanding and decoding the Mozarts Ear[6] app code base which took up a lot of time before we understood that their methodology would not work properly in the context of our app. The same time could have been used to add more functionality to our app had we realized this earlier. Therefore, the final version of the app is not based on Mozarts Ear.

5.2 Collaboration on the Project

- It is better to have the scope of the project well defined as early as possible. Getting feedback from people in the industry who are currently in similar fields helped in defining a realizable scope based on time constraints.

[6]"Mozarts Ear." Mozarts Ear. `http://www.eecg.utoronto.ca/~jayar/ece1778.2013/Reports/MozartsEar.pdf` (accessed April 10, 2014).

- The initial learning curve for understanding the underlying music theory terminology is considerably high for a non musician. Visually representing the underlying music terminology helped in understanding it properly.
- Separation of the code work into small individual tasks and using github version control helped us in working collaboratively on the project.

6 Contribution by Individual Members

6.1 Akshay

- Providing the overall idea and direction.
- Porting the Onset Detection algorithm based on the Minim Library initially designed for Processing to work on the Android platform.
- Developing stitching algorithm from scratch.
- Collecting feedback and guidance from an organization in Toronto working in a similar space.
- Conducting regular team meetings and organizing tasks among team members.

6.2 Amanjot

- Decoding of Mozarts Ear code to understand Key and Note detection. This was not used in the final version of Acoustica.
- Developing the beat pattern to drum loops mapping algorithm.
- GUI design and custom font for the app.
- Media Player functionality in the app.

6.3 Nitin

- Decoding Mozarts Ear key detection algorithm and using the same for Acoustica. This was not used in the final version of Acoustica.
- '.raw' format to '.wav' format conversion.
- Input recording threshold detection.
- Music library feature.

7 Apper Context

Music composition is an art that is mastered over many years and requires in-depth knowledge of music theory and ear training. Amateur musicians find inspiration as they begin composing music by creating a melody, a short arrangement of notes that sound good together. However, transitioning from a melody to a complete song can be a long and tedious process especially without the right tools and knowledge. Acoustica bridges this gap by analysing the melody and augmenting other instruments to it to give it a fuller sound.

The scope of the problem addressed by Acoustica is large and with careful consideration our team has taken the first step forward by adding percussion to the input melody provided by the user. Percussion instruments form the foundation of any music composition and therefore it is a good starting point for this app.

Music composition happens on the go, could be on top of a mountain while enjoying the beauty of nature or in a car while being stuck in a traffic jam, or on the side of a street, soaking in the rain getting over a heartbreak. A mobile app platform allows the user to compose music at the time when the emotion can be captured at its best.

This app also helps solo musicians by giving them a fuller sound on their music compositions. Currently, solo musicians use expensive software to digitally create other instruments which takes time and music knowledge. The user requires a lot of patience to piece together the small musical notes and beats from different instruments to make a composition. Few solo musicians also choose to collaborate on online platforms like Youtube for their compositions. Acoustica leverages the advancement in mobile technologies to automate part of this process to make music composition a fun and easy task.

Technological advancement has also allowed bands with as few as two members to perform concerts in large arenas. These bands cover up the lack of musical instruments by playing backing soundtracks from their laptop. These soundtracks include a plethora of digital instruments to give a unique sound to their music. With the future development of Acoustica, our goal is to allow the user to achieve this using their mobile phone which would provide the necessary backing instruments in real time based on what the user is currently playing.

8 Future Work

The following developments can be made to Acoustica in the future :

- Adding tonal backing instruments such as piano, rhythm guitar and violin.
- Extracting more features from the input melody to use for better composition. This can include features like key[7] and harmony[8] detection.

[7]Brian Hyer. "Key (i)." Grove Music Online. Oxford Music Online. Oxford University Press, accessed April 10, 2014, `http://www.oxfordmusiconline.com/subscriber/article/grove/music/14942`.

[8]Steven Strunk. "Harmony (i)." The New Grove Dictionary of Jazz, 2nd ed.. Grove Mu-

- Add more user customization options to define the type of song the user wants. For example, instead of limiting the user to the best match drum beat to their input melody, allowing them to choose from the top three matches would let the user bring a more personal touch to their compositions.
- Implement variable tempo customizable by the user.
- Improve the GUI to bring all functionality to one page.

9 Code Base

Acoustica team agrees to uploading of our video and making our code open source. Source code can be found at: `https://github.com/akshaygill/Acoustica`

10 References

1. "Compartmental." Compartmental. `http://code.compartmental.net/tools/minim/` (accessed April 10, 2014).
2. "Managing Audio Playback." Android Developers. `http://developer.android.com/training/managing-audio/index.html` (accessed April 10, 2014).
3. "public ece1778github/Mozarts-Ear." GitHub. `https://github.com/ece1778github/Mozarts-Ear` (accessed April 10, 2014).
4. "public neopixl/PixlUI." GitHub. `https://github.com/neopixl/PixlUI` (accessed April 10, 2014).

sic Online. Oxford Music Online. Oxford University Press, accessed April 10, 2014, `http://www.oxfordmusiconline.com/subscriber/article/grove/music/J990085`.