2019 EECS 352 Final Presentations

See more at guitabwebsite.github.io

GuiTabs: An Automatic Guitar Transcription Generator

David Hofferber, Joseph Eligon, Joshua Fields

Machine Perception of Audio and Music. Northwestern University, USA

NORTHWESTERN UNIVERSITY

1. Motivation

- People want to be able to play music, and the majority of the guitar repertoire is only available as audio
- Manual transcription can often take a long time and is also difficult for beginners

2. Problems

- Most transcriptions needs to be done based on the audio because transcriptions are not commonly available. However, pitch tracking is a difficult task in itself
- Even discounting the different positions your hand can be in while playing notes on the guitar, there are still five or six different ways to play every note
- Even though the notes are important, the fingerings are probably the most important part of the song in order to be viable to play. However, due to the complexity of the fingering, much of how the fingerings are determined depend upon complex distance metrics

Evaluation Metrics

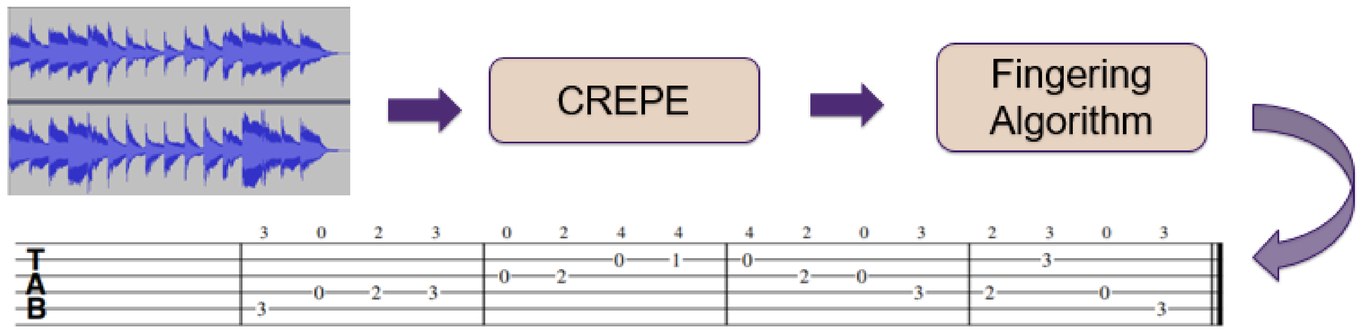
- We evaluated our results based on their pitch accuracy to the actual notes. This was done through human feedback, where the user told us whether or not the pitches sound right.
- We evaluated our results based on the fingerings ease of playability. This was done by having several guitar players play the generated tablature and give feedback on how natural it was for them to play it.

Related Work

- Bello and Monti in “Techniques for Automatic Music Transcription” give a rough blueprint of producing audio transcriptions from audio and some methods that we have not endeavored to use in this project.
- Dlabal and Wedeen in “Generating Sheet Music From Audio Files” give their approach for going from audio to notes and give some detailed note-detection information
- Barbancho, et. Al in “Automatic Transcription of Guitar Chords and Fingering” discusses a way of feature extraction from many guitar samples to glean useful metrics for potential fingerings.

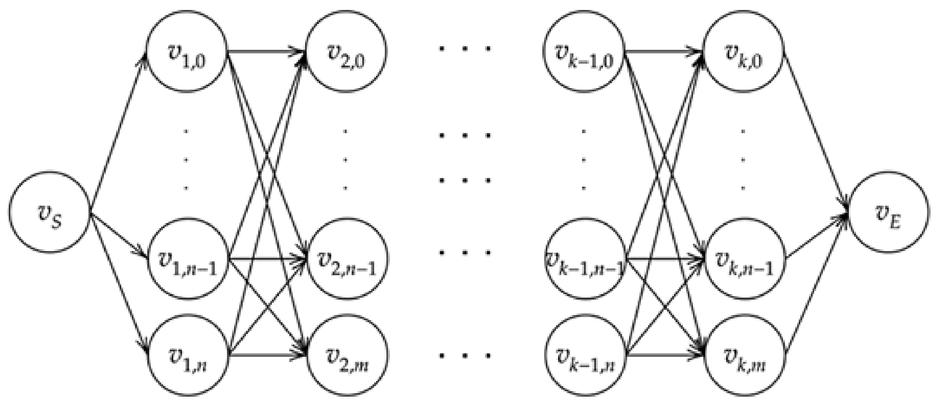
3. Pipeline

1. The user indicates an audio file to transcribe
2. We run CREPE on the audio file, determining new notes based off of having a sufficient change in Hz. If CREPE is not confident for too long, we also terminate the current note.
3. Using the note list generated by CREPE, we identify an optimal fingering by performing a shortest path algorithm on a graph representation of the finger pattern.
4. Using the optimal fingerings, we use Muse Score in order to transcribe the notes in tablature, and generate a pdf of it



4. Fingering Algorithm In Depth

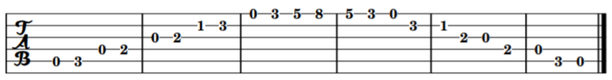
- We initially parse our song with k notes into a graph structure of $k + 2$ layers, where each vertex is a way to play the i^{th} note.
- We then connect each vertex layer to the next vertex layer with directed edges, where weights are assigned by a distance metric between note tuples
- Once we have our finalized graph, we perform Dijkstra’s Algorithm to find the shortest path through the graph based on the weighted edges, which returns the fingerings which require the least movement based on our distance metrics.



The vertex layers of our song, where v_S is a special node representing the start of our graph structure, v_E is a special node representing the end of our graph structure.

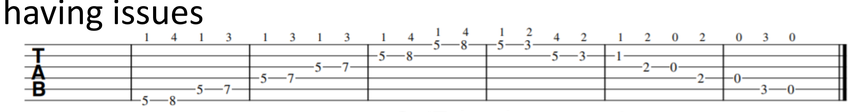
5. Results

- On monophonic audio without noise that it is 16 bit depth we are able to mostly successfully transcribe audio into tablature



LilyPond tab generation

- However, a few notes are sometimes an octave off due to CREPE (and pitch trackers in general) having issues determining octaves



GuiTab tab generation

- The fingerings will occasionally put more emphasis on shifting positions and repeatedly using your pinky than it does on using your ring finger and shifting at a later time

6. Future Work

- Fine tune weights of fingering graph or use an alternative machine learning approach
- Integrate beat onsets in pitch tracking
- Ideally, make capable of polyphonic input

Acknowledgements: This work used the CREPE pitch tracker made by Justin Salamon, Jong Wook Kim, Peter Li, and Juan Pablo Bello