Final Report Part II

Guitar Fretboard Mapping (Audio to Virtual Fretboard) and Chord Finder

COMP 4801 Final Year Project

Department of Computer Science

The University of Hong Kong

Hamza Siddiqui, 3035243106

Supervisor:

Prof. Dirk Schneiders

14th April, 2019

Table of Contents

a. List of Figures

b. List of Tables

b. Outline:

This report aims serves as part II to the exhaustive analysis of our final year project. Refer to part I of this report for a thorough background and introduction of the project, and discussions regarding previous works and existing applications in this domain. Chapter 3 of this paper, in conjunction with Chapter 3 of part I gives a thorough examination and justification of the methodology used. This chapter goes into detail regarding the image processing carried once the region of interest has been extracted from the video input, for the purpose of *Fretboard Mapping*. It further discusses the multiprocessing facet of the application, which is necessary for the concurrent analysis of video and audio required for fretboard mapping. Finally it discusses the application, *Chord Finder*, the building of the chord database required for its functionality and also, its GUI design. Success metrics and testing procedures for the project are included in Chapter 4, along with the results of our fretboard mapping algorithms. This is followed by a discussion of this project's limitations, implications and the possible future work. Finally, Chapter 5 concludes the report, and also summarises the successes, findings and insights of this project.

3. Methodology

3.1 Fretboard Mapping

3.1.5 Image Processing

3.1.5.2 Capturing the user's approximate position of the hand on the fretboard

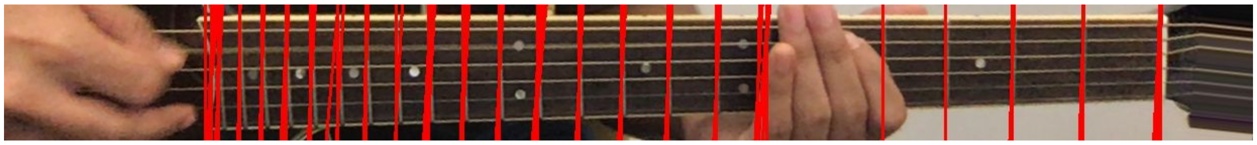


Fig. 3.12 Hough Transform applied after a horizontal sobel filter. The sobel filter ensures the detection of only vertical lines (the frets) on the fretboard.

Once the region of interest is isolated, the position of the hand is detected after another series of filters. A *Horizontal Sobel Filter* is applied to only detect vertical edges. Another *Hough Transform* is applied to get the coordinates of the vertical lines, which are the frets of the fretboard, and this is used to crop the headstock and bridge of the guitar, leaving us with just the fretboard.

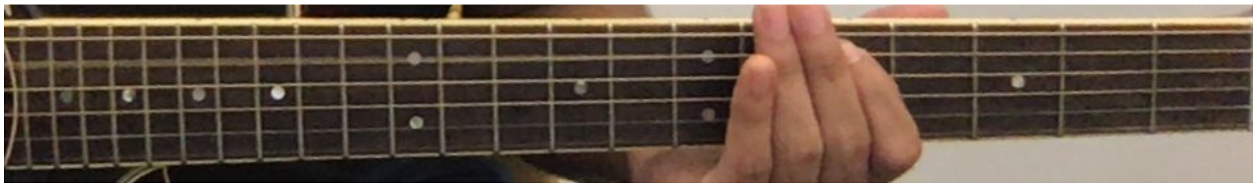


Fig. 3.13 The fretboard has been successfully extracted. The only part left is to detect the relative x coordinate of the hand.

For the final step of image processing, the histogram of the skin color described in 3.1.2 of part I of the report, will be used to extract the x coordinate of the top of the user's hand. OpenCV provides a method *cv2.calcBackProject,* that separates the regions which are in the range of the histogram values. After applying a couple of filters, the final thresholded image will then enable the detection of contours of the fretboard picture. The largest contour detected is taken to be the hand, and from the information of the contours, the x coordinate of the middle of the hand is returned.

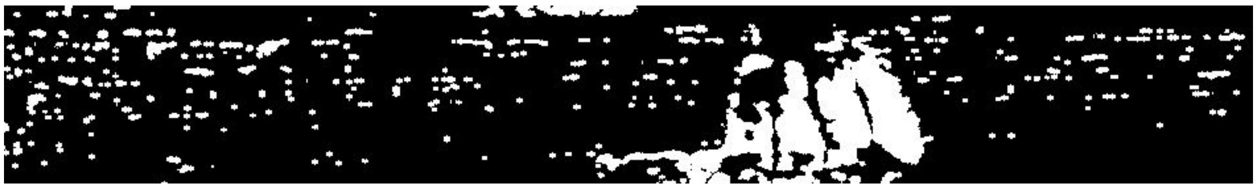


Fig. 3.14 cv2.calcBackProject is used to separate the features which are in the range of the skin-color histogram

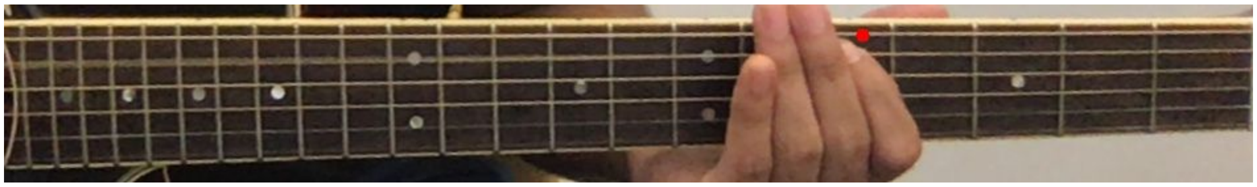


Fig. 3.15 The x coordinate of the top point of the largest contour is the approximate location of the guitarist's hand, which is what we need.

Since the dimensions of most acoustic guitar fretboards are roughly equivalent, we can use the x coordinate to determine the approximate fret position at which the user played the note. Then from a lookup table, the detected pitch and the relative hand position can be used to output the string and fret at which the user played the note.

3.1.6 Multiprocessing

Displaying the video (for the user to see where the guitar is being positioned), and capturing the audio at the same time is important for not only the functionality of the fretboard mapper, but also the user experience. To incorporate both video and audio processing simultaneously, the multiprocessing and multithreading libraries of Python were used.

Python has been designed in such a way, that only a single thread can access control of the Python interpreter. This is due to a feature in Python known as the Global Interpreter Lock (GIL), which is a mutex lock that allows a single thread access to Python objects. Due to this, even in a multi-threaded architecture with more than one core, the GIL only allows one thread to execute at a time. This rules out a multi-threaded approach for processing both video and audio since the frame rates of video capture is far lesser than the frame rate of audio capture.

A way to bypass the Global Interpreter Lock, and to get video frames and audio frames at the same time, would be to use multiprocessing. Each process would have its own Python interpreter and would be able to access the audio and video separately and simultaneously. Guitar Guide has two processes running together, the main process which handles GUI events and captures video frames from the user's webcam (running on a different thread), and the process which analyses the audio from the microphone. The multiprocessing library allows two types of communication channels between processes: a queue and a pipe. Guitar guide uses a queue to send the pitch from the audio process to the main process. The main process, which has been displaying the video feed on the Tkinter GUI, will then send the image and the pitch to an image processing function, which will then both process and return the fretboard position of the user's hand.

3.2 Chord Finder

3.2.1. Concept

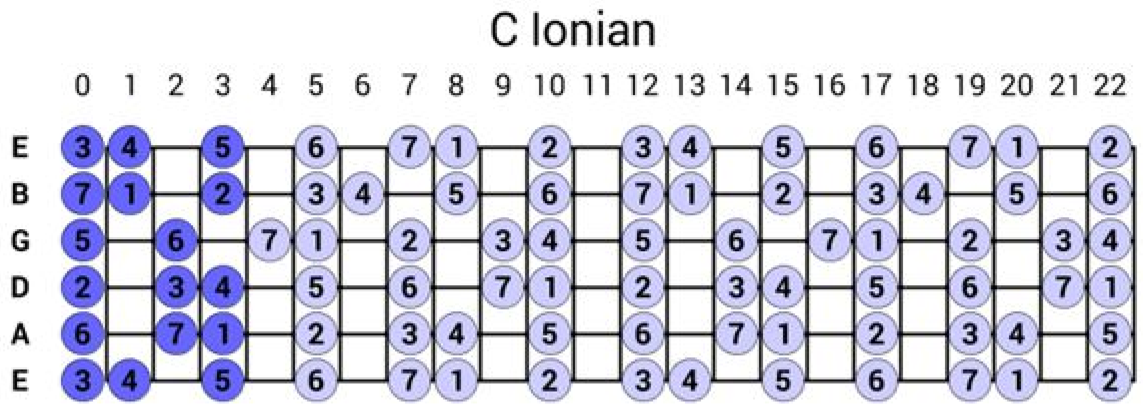


Figure 3.16: All notes of the C major scale on the fretboard labelled by scale intervals. The C major chord is constructed using any (realistically playable) combination of 1, 3 and 5 positions and the D minor chord is constructed using any (realistically playable) combination of 2, 4 and 6 positions found in this diagram [1].

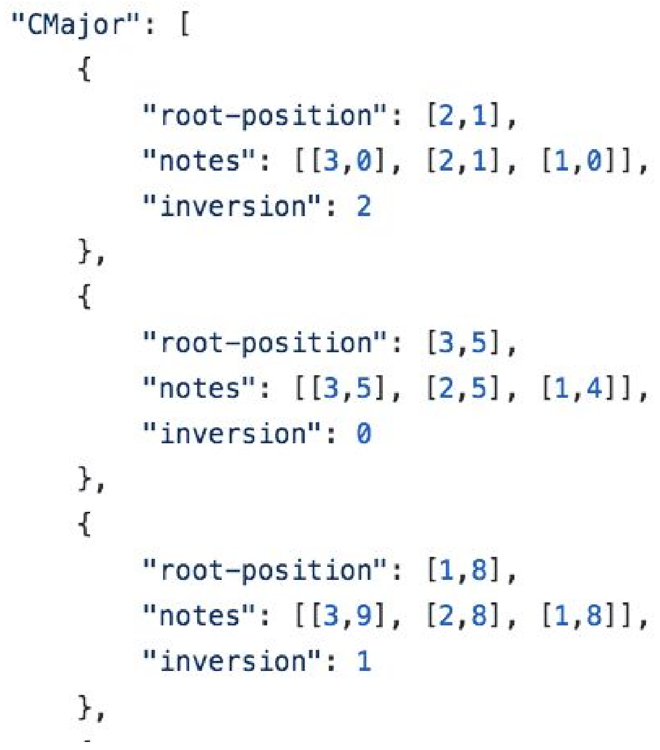
Harmony, one of the core facets of a musical piece (the other being melody), is composed of abstractions called ‘chords’. Chords are sets of three or more distinct ‘intervals’ of the ‘major scale’, a western music theoretical conception that orders specific notes by fundamental frequency/pitch. Chords are identified by their ‘root’ note; for instance, the C major chord has the root note, C. Figure 2 shows all the notes in the C major scale on the fretboard; the C major chord is constructed from the 1st 3rd and 5th intervals of the C major scale, for example. There are numerous chord types that share the same root note, such as C major, C minor, C7, C suspended etc. For a particular song, certain chords are said to be ‘in key’, meaning that the notes in the chord do not violate the key of the song (a key is a set of notes that sound great together; songs tend to be written in a particular key i.e. all of the notes played in the song are taken from the scale defined by the key). For each chord, there are a large number of possible voicings, or finger positions on the guitar. For beginners, it is often difficult to determine whether a chord is in key, and to recall and choose the best amongst the numerous voicings.

The user will play a root note on his guitar, the position of which will be identified using the fretboard mapping procedure described in the preceding section. The user will then be provided with a set of chord types that share the root note identified and fit the key of the song being written by the user (the key parameter will be provided by the user at the start of his songwriting session), and the most convenient voicings given the current hand position of the player, fetched from a chord voicing database. The user will pick his preferred voicing, which will be added to a 'chord map', a record of the chords played by the user in the song thus far. The user will then be given the option of playing the root note of the next chord he wishes to play. The user will be prompted to choose a different root note if the user's selection is out of key.

The voicings shown to the user will be filtered from a chord voicing database, with the system making use of the root note played and its position, and the song key. Since there are a large number of voicings for each chord, the query generated will also consider the spatial proximity of the voicing to the root note played on the fretboard, choosing only the voicings that are closest to the user's hand position.

3.2.2 Chord Voicing Database

A chord voicing database has been created in JSON format. The database has been built for major and minor chords in all keys. This was done using an algorithm that automated the creation of the database based on information of the C major and C minor chords, which were created manually. A snippet of the database has been attached in the figure below.



```
"CMajor": [
    {
        "root-position": [2,1],
        "notes": [[3,0], [2,1], [1,0]],
        "inversion": 2
    },
    {
        "root-position": [3,5],
        "notes": [[3,5], [2,5], [1,4]],
        "inversion": 0
    },
    {
        "root-position": [1,8],
        "notes": [[3,9], [2,8], [1,8]],
        "inversion": 1
    },
```

Fig 3.17. Chord Voicings Database

3.3 Design

The rationale behind using a desktop application as opposed to a smartphone application was the expected discomfort associated with using the latter for this purpose; a mobile phone would have to be placed at a distance to capture the fretboard, which would make it highly unintuitive both to provide input to the device and to look at the small screen. Desktop screens are larger, which will mitigate the reduced visibility of the app's GUI due to distance.

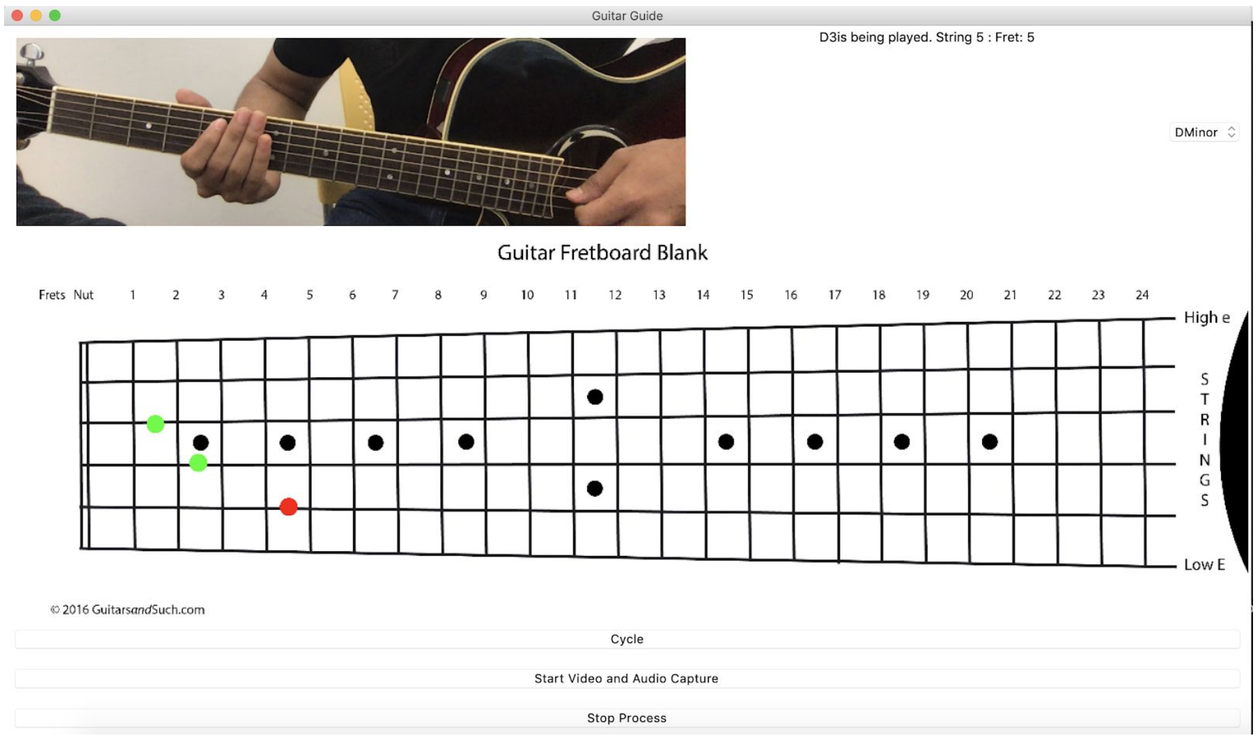

Guitar Guide
D3is being played. String 5 : Fret: 5
DMinor
Guitar Fretboard Blank
Frets Nut 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
High e
STRINGS
Low E
© 2016 GuitarsandSuch.com
Cycle
Start Video and Audio Capture
Stop Process

Fig. 3.18 GUI of Chord Finder, the application built on the foundation of the fretboard mapping algorithm. As you can see here, the user plays a note D3 on the guitar, the note is correctly mapped onto the virtual fretboard and the D minor chord is displayed for the user to play

The graphical user interface includes a visualization of a fretboard, with the root notes played by the user and the notes of the suggested chord voicings being highlighted in real-time. The user will have the option to cycle through and choose from various voicings of the chord rooted at the detected note.

It was decided that the application will be written in Python; this is due in part to the extensive existing work on Music Information Retrieval (MIR) done in Python, which was used in the implementation of the pitch detection aspect of the application. Other reasons included the existence of packages such as OpenCV, which is an industry standard within the domain of Computer Vision; this was used in the implementation of fretboard hand recognition. Tkinter, a Python binding to the Tk GUI toolkit was also used for developing the user interface. In short, Python provides an all-in-one solution to all of the facets of the project.

Moreover the application runs with standard and easily accessible components;the implementation has been carried out on a MacBook Pro 13-inch model, using its default webcam and microphone.

4. Testing

4.1 Evaluation Techniques

In order to test the accuracy of fretboard mapping, a controlled experiment was carried out. 200 notes were randomly generated and were played on guitar by a user positioned in front of a computer running the application. In order to limit the effect of external variables such as intensity and type of lighting, background colour, skin tone, ambient noise etc., the experiment was carried out in a single quiet room, at the same position, and by the same user. Half of the notes were played by a steel-string guitar, and the other half were played by a classical, nylon-string guitar. Two types were used for the test because these are two of the most popular types of acoustic guitars used today; this also allows for the testing of the difference in accuracy between the two types.

The random notes (string, fret combinations) generated were played sequentially by the user. For each note, the trial was deemed to be a success if the played note corresponded to the note displayed by the application. For failed attempts, it was noted whether the failure was due to errors in the audio mechanism, errors in the video mechanism, or errors in both mechanisms. This distinction would eventually prove vital in comparing the error rates of the two mechanisms.

A MacBook Pro 13-inch model with Touch Bar, 2.3GHz quad-core Intel Core i5, Turbo Boost up to 3.8GHz, with 128MB of eDRAM, was used to conduct the tests. The webcam resolution is 1280x720. The default sampling rate of the microphone is 48000 Hz.

4.2 Results:

4.2.1 Analysis of accuracy:

	Steel string	Nylon string	Overall
Accuracy	78.0%	91.0%	84.5%

Table 4.1 Accuracy of the fretboard mapping technique with different guitars and overall accuracy, calculated based on 100 random samples from steel string and 100 random samples from nylon string

As is depicted in the table above, the point estimate of the overall accuracy for fretboard mapping is **84.5%**; the 95% binomial proportion confidence interval of the overall accuracy is **[0.7884, 0.8886]**.

The table also suggests that nylon string guitars yield a better accuracy than steel string guitars (**91.0%** v **78%**). To compare the success rates (proportion of accurate results) for the two types of guitars, a two-sample Z-test was used, with the following hypotheses:

H0: The success rate for nylon string guitars and steel string guitars is equal
H1: The success rate for nylon string guitars is greater than that of steel string guitars

The test yielded a p-value of 0.0055; therefore, H1 is accepted at the 1% significance level i.e. we can say with 99% confidence that <u>the success rate for nylon strings is significantly greater than that of steel string guitars.</u> This can be explained by the shape of the fretboard of a nylon string guitar being a closer approximation of a rectangle than that of the fretboard of a steel string guitar; this means that the rectangular region-of-interest captured is less likely to capture the background, for a nylon string guitar. Furthermore, pitch detection for the nylon string guitar was significantly more accurate (0 errors for the 100 samples). This could be attributed to the cleaner tone of the nylon string guitar, resulting in fewer harmonics being detected.

4.2.2 Analysis of the error-types:

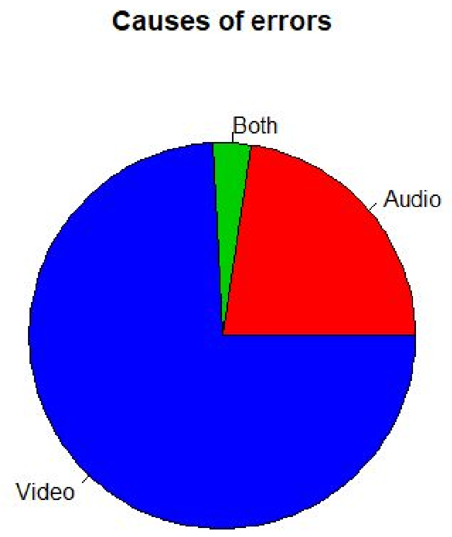


Fig 4.1 Pie chart demonstrating the proportions of the sources of errors encountered.

Next, we compare the error rates of the two mechanisms (audio and video). 25.8% of the errors were caused due to audio errors, and 74.2% of the errors were caused due to video errors. A two-sample z-test for the difference between error rate of the audio mechanism and the error rate of the video mechanism was carried out, with the following hypotheses:

H0: The error rates of the video and audio mechanisms are equal
H1: The error rate of the video mechanism is greater than that of the audio mechanism

The test yielded a p-value of 0.001; therefore, H1 is accepted at the 1% significance level i.e. we can say with 99% confidence that <u>the error rate of the video mechanism is significantly greater than that of the audio mechanism.</u> This is likely due to the video mechanism being a more complex process, having a significantly greater number of steps (and parameters to adjust) than the audio mechanism.

4.2.3 Correlation of fret/string number with the success rate:

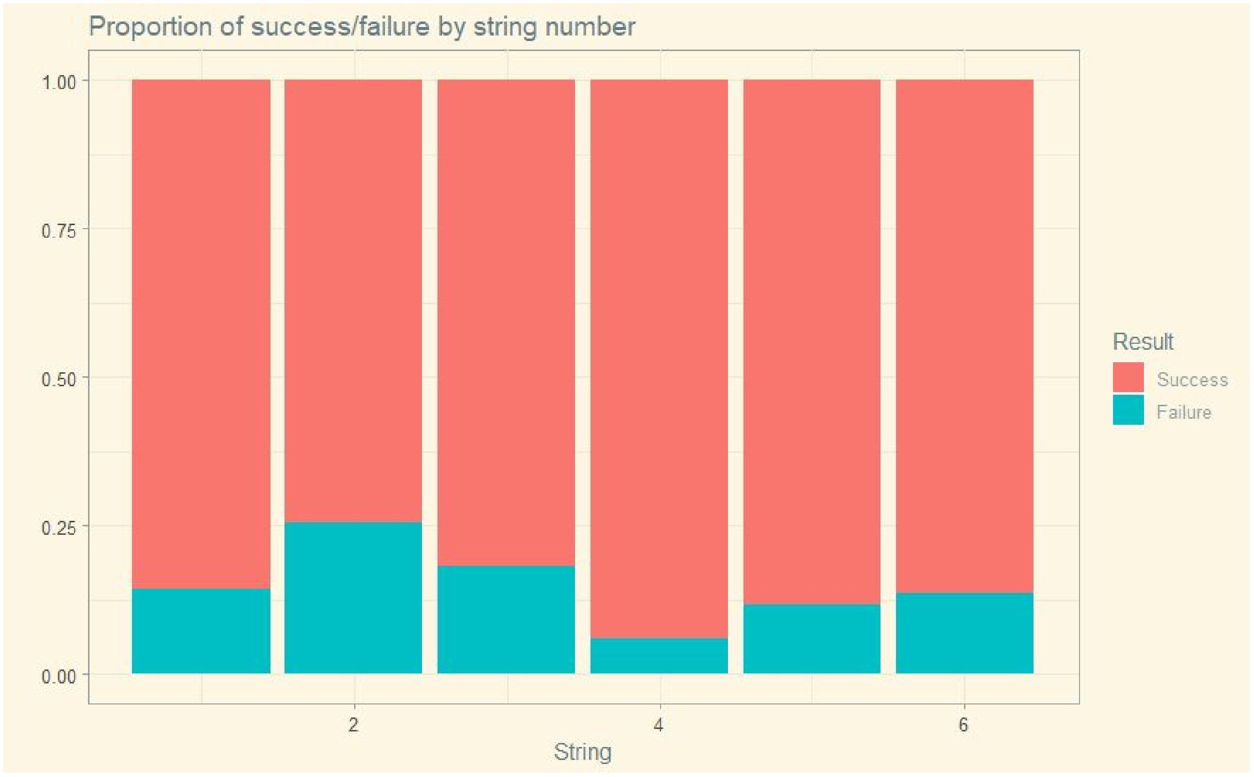


Fig 4.2 The higher success rate of strings 4, 5, and 6 is due to the hand being detected as the largest contour by the cv2.findContours method. For strings 1, 2, and 3, the errors are mostly due to the hand not being the largest part of the region of interest.

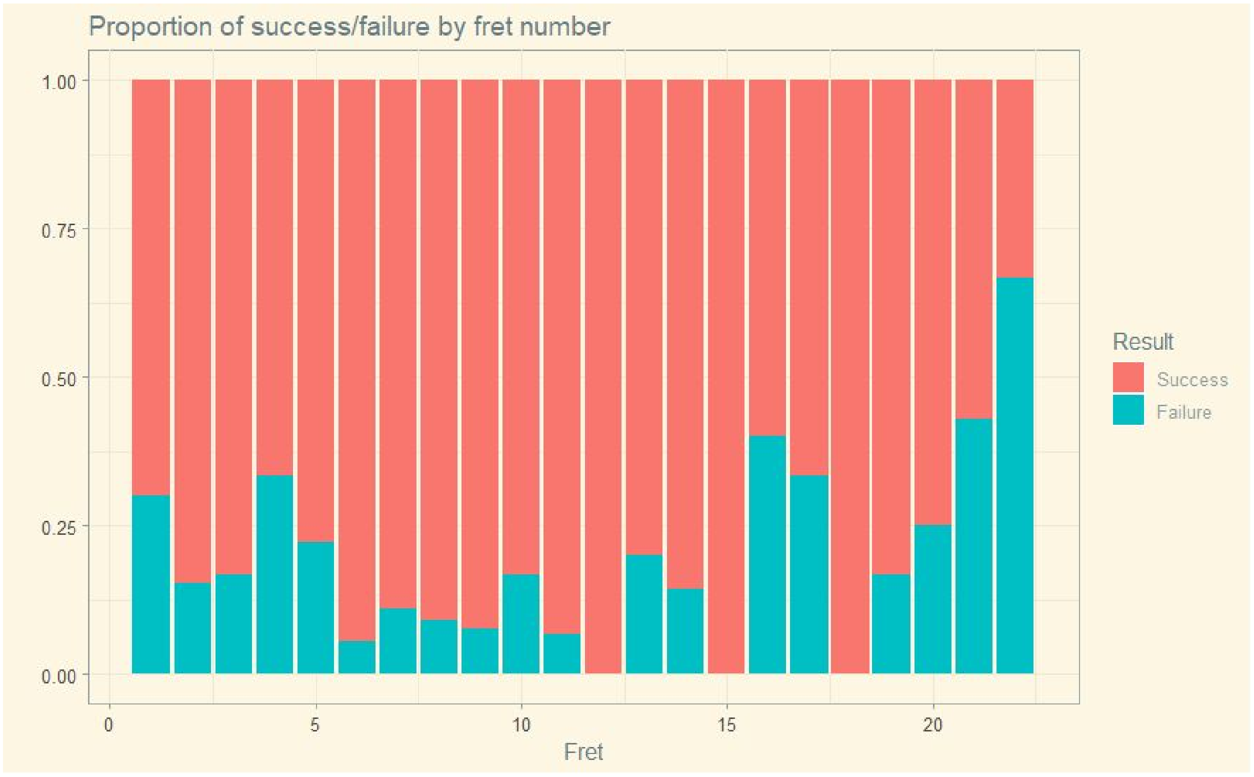


Fig. 4.3 The pitch detection algorithm that has been used is usually more error-prone in the extremely high pitches (i.e. > 20th fret) or extremely low pitches (1st fret). The mistakes are due to octave errors from the frequency autocorrelation technique.

4.3 Limitations

One of the limitations of the application is that if consecutive notes are played faster than a tempo of about 85 bpm (beats per minute), the application fails to accurately record all the notes played. Although this limitation has little consequence for our application, which is based around the idea of playing a singular note, and then observing the possible chord voicings rooted at that note, it does affect the performance of the transcribing a melody if it is played at a speed higher than 85 bpm, which usually is not the case.

Another limitation is that the output is sensitive to varying lighting conditions, and, to a lesser extent, to the ambient noise. Therefore, the use of this application is discouraged in dimly lit environments, in environments where the light is not uniformly distributed, and in noisy environments.

4.4 Problems Faced

The chord voicing database implementation required an enormous time dedicated to manual data entry, since there was no resource available that documented chord voicings in the representation required for the purpose of this application. In order to alleviate this problem, a Python script was written that helped automate this procedure: the script took an input of all the voicings of 1 of 12 tonal centers of a chord type, and outputted the rest. This immensely reduced the time needed to construct the database, as we only needed to input all the voicings on the guitar of a single tonal center for every chord type.

Regarding the computer vision algorithm implemented thus far, the determination of suitable values for the numerous parameters involved proved to be a demanding task. Parameters yielding positive results when the algorithm was run on a certain personal computer employing a certain camera build usually failed to yield the same output on a personal computer with different specifications. Moreover, even on the same personal computer, the results fluctuated immensely. This issue was alleviated by using only one computer for the purpose of implementation and evaluation, and by testing in well-lit and relatively quiet environments (thereby, making the assumption that the user will be in similar conditions in their use of the application). Furthermore, after sufficient trial and error, the parameters were optimised for that particular computer, yielding consistent results.

Initially, for step 2 of the image processing algorithm, we tried to detect the user's hand using the largest contour function. However, we realized that even after cropping the minimum area rectangle that covers the fretboard, there was still a part of the background that was visible which was not part of the guitar fretboard. This happened due to the fact that acoustic guitars are not perfect rectangles. Thus, in a lot of cases, the background used to get detected as the largest contour, instead of the hand, and this caused a lot of errors. The error was fixed by taking the skin color histogram of the skin to retrieve the approximate position of the hand as the background would be of a different color. The new method is described in detail in section 3.1.5.2.

An existing library function of Aubio was used for detecting the pitch of the wave at the onset previously, which made use of the *Yin* algorithm: an autocorrelation technique to determine the fundamental frequency of the note, which is used to evaluate its pitch. However, this function was not deterministic and was sufficiently inconsistent in its output. Other library functions and algorithms were then tested. Finally, the *frequency using autocorrelation* technique was used.

4.5 Future Work:

Although this project shows that fretboard mapping at a reasonable accuracy and minimal hindrance is feasible, it falls short of implementing a product that can run seamlessly and accurately across all personal computer types and builds. In this connection, a future project could implement a means to dynamically adjust the parameters, depending on the specifications of the computer on which this fretboard mapping process is carried out.

The self-education guitar software, *Chord Finder,* can be expanded to include more features, such as a *Harmony Recommender,* which will serve as an introductory guide to commonly used chord progressions and provide a good base for amateur guitarists to start writing simple harmony. The application will then have an emphasis on songwriting, a skill sought after by most guitar players.

Once the fretboard mapping works reasonably well on multiple different laptops, then a feature will be added to transcribe the melody the user plays and store it in an MusicXML file which can later be exported as sheet music.

5. Conclusion

This paper proposed an algorithm that maps the user’s guitar note playing to a virtual fretboard which works with an accuracy that is higher than existing methods (around 84.5%) . Using the fretboard mapper, a self-learning guitar education application, Chord Finder, was also created which focuses on finding chord voicings. It uses video input from the guitar fretboard and raw audio input from the note played to identify the exact position of the note on the fretboard; it uses this information to provide the user with the tools he needs to compose or even possible notate a compelling piece of music.

It is anticipated that the algorithm, Fretboard Mapper and the application, Chord Finder, will be highly beneficial to budding guitarists given the distinct lack of interactive songwriting education apps of this kind in the market. Amateur guitarists will quickly be able to both learn and apply music theoretical concepts, without having to aimlessly memorize the seemingly innumerable fretboard shapes without context.

6. References

1. JGuitar (n.d.). *C major scale notes (tones)*. [image] Available at: https://jguitar.com/scale?root=C&scale=Ionian&fret=0&labels=tone&notes=sharps [Accessed 24 Oct. 2018].