PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	A Computer Vision-Based Automatic Transcription of Guitar Music from RGBD Videos
Auteur: Author:	Mark Asmar
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Asmar, M. (2022). A Computer Vision-Based Automatic Transcription of Guitar Music from RGBD Videos [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10470/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10470/
Directeurs de recherche: Advisors:	Lama Séoud, & Guillaume-Alexandre Bilodeau
Programme: Program:	Génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

A Computer Vision-Based Automatic Transcription of Guitar Music from RGBD Videos

MARK ASMAR
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2022

© Mark Asmar, 2022.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

A Computer Vision-Based Automatic Transcription of Guitar Music from RGBD Videos

présenté par **Mark ASMAR**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Giovanni BELTRAME, président
Lama SÉOUD, membre et directrice de recherche
Guillaume-Alexandre BILODEAU, membre et codirecteur de recherche
Caroline TRAUBE, membre externe

ACKNOWLEDGEMENTS

Two years ago I decided to embark on a new journey in pursuit of a master's in Computer Engineering. It was a long and a big journey to come to Montreal from Lebanon in the midst of a pandemic. Despite all the challenges and complications, I can say that it was another challenge I overcame. However, this would not have been possible without the help of others whom I want to thank.

First of all, I want to express my thankfulness and gratitude to my supervisor Dr. Lama Seoud. Dr. Seoud was a key factor in helping me completing this work; her advises and supervision were helpful. She always maintained a positive attitude, she was genuinely interested in the subject and motivated about getting it done which pushed to keep me going. She felt more like a friend than a supervisor, her humbleness and personality made communication with her very easy and very friendly. She was always there for her students, ready to help, motivate them and make sure that they achieve their potential.

Likewise, I also would like to thank Dr. Guillaume-Alexandre Bilodeau, who became my co-supervisor somewhere around halfway of my master's. He immediately was up-to-date with all the needs of project, understood the context and became of help right away. He shared his knowledge and experience in the field of computer vision by suggesting ideas and different helpful ways to achieve certain tasks. Dr. Bilodeau was always present for any need and saw certain points from a different perspective which helped to improve the project in many ways.

Additionally, all of this would have never been possible without Polytechnique Montréal and the "Département de génie informatique et génie logiciel". I sincerely would like to convey my gratefulness to them for letting me pursue my master's degree at the university department.

Furthermore, I would like to thank my friends and colleagues with whom I shared good times, knowledge, experiences and thoughts. Whether these things were technical, cultural or just for fun, they definitely were of assistance during my master's. I thank: Youmna, Mohammad, Antonin, Olivier, and Ghassen. In addition, I extend my thanks to all the people who submitted videos during the data collection for the fretboard segmentation task.

A big thank you goes to my family, friends and different people I met throughout my life and studies. I cannot thank my parents and my brother enough for their help, their support was not limited to this master's but to everything that I did since my childhood. Another amazing person that was always supportive and ready to help is Elissa Lahoud, she was

always ready to listen and help whenever she can, a big thanks goes to her. At last, I want to thank Mr. Joe Sfeir; I met Mr. Sfeir in Montreal and he also expressed an interest in the project and provided me with different guitars whenever there was a need.

RÉSUMÉ

La musique est un art d'expression auditive, beaucoup de musiciens écoutent et jouent de la musique. Ces derniers partagent leurs musiques en forme écrite (notation) grâce à la transcription. Plusieurs notations existent, comme la notation musicale standard (la portée) et les tablatures de la guitare.

La transcription ou l'écriture d'une pièce musicale en cours de composition est faite généralement à la main, toutefois le processus est long, fastidieux et prone aux erreurs. Pour cela, plusieurs travaux ont porté sur l'automatisation de cette tâche. Les approches proposées utilisent la vision par ordinateur, le traitement du signal audio ou bien une combinaison des deux.

Dans ce travail, nous nous intéressons à un instrument en particulier, commun et utilisé presque partout aujourd'hui, la guitare. Pour partager des pièces musicales destinées à être jouées sur une guitare, les guitaristes peuvent utiliser la tablature ou bien la notation standard. Cette dernière est plus généraliste (c'est-à-dire peut être utilisée par des instruments différents) et se concentre sur la note et la durée de cette dernière. Hors, dans le cas de la guitare, une même note peut etre jouée sur différentes cordes et frettes de la guitare. Ainsi, la tablature offre une notation plus directe puisqu'elle indique exactement la corde à gratter et la frette à presser. Ces informations sont absentes dans la notation musicale standard.

Puisque le traitement par audio nécessite une connaissance a priori sur le réglage des cordes ainsi que leurs timbres, en plus de ne pas pouvoir savoir la frette et la corde jouée sans savoir les timbres des cordes, une approche basée sur le traitement du signal audio semble complexe. Par conséquent, dans ce travail, nous proposons une méthode de transcription se basant uniquement sur des vidéos.

Dans l'objectif de développer un système de transcription automatique basée sur la vision par ordinateur, nous avons utilisé une camera RGBD de grade commercial, qui offre à la fois des trames RGB ainsi que des cartes de profondeur. Les trames RGB sont utilisées pour la segmentation de la guitare, l'estimation de la pose de la main, le calcul du flux optique et le post-traitement; tandis que les cartes de profondeur sont utilisées pour l'estimation de la pose de la main et le post-traitement.

Dans le but d'entrainer un modèle d'apprentissage profond pour la segmentation automatique du manche de la guitare dans les trames RGB, nous commençons par construire notre propre base de données. Après entrainement, ce modèle prédit un masque binaire indiquant la

position du manche, notre région d'intérêt sur la guitare.

Ensuite, nous utilisons un modèle d'estimation de la pose de la main, MediaPipe Hands, pour prédire 21 articulations de la main. Ce modèle nous donne en sortie les coordonnées 3D de chaque articulation.

Le fait de presser simplement une corde sur une frette ne produit pas de la musique. Ainsi, l'étape suivante consiste à calculer le flux optique entre deux trames successives dans le but d'identifier si une corde a été grattée. Cette étape est importante parce que la transcription ne doit se faire que lorsqu'une corde est grattée.

La dernière étape de notre approche consiste en le post-traitement des sorties de chacune des étapes précédentes. On utilise le masque binaire de la segmentation pour isoler le manche, pour ensuite détecter les cordes ainsi que les frettes. Le flux optique est utilisé pour savoir quelle corde est jouée ; et finalement on utilise le résultat de l'estimation de la pose de la main pour savoir quelles articulations pressent quelle corde et sur quelle frette avant de générer la tablature correspondante.

Afin de valider notre approche, on a testé chaque étape séparement avant d'expérimenter avec l'approche toute entière. Le modèle de segmentation a été évalué sur nos données de test, ce dernier atteint un score de 0.9230 pour Dice. Pour l'estimation de la pose de la main, des images RGBD capturées avec notre camera RGBD ont été utilisées pour mesurer les erreurs de profondeur en millimètres. Des vidéos RGBD de deux guitaristes ont été enregistrées (en ayant leur vérité terrain, c'est à dire la tablature de référence) pour évaluer la méthode de détection de la gratte ainsi que le système complet.

Les résultats obtenus démontrent que la segmentation est la partie la plus satisfaisante du système, elle est précise et robuste. Le jeu de données collectées pour cette tache semble très prometteur puisqu'il offre des images fortement diversifiées ce qui augmente la capacité de généralisation du modèle de segmentation. La détection de la gratte quant à elle nécessiterait une meilleure résolution au niveau des images, puisque certaines cordes sont difficilement visibles dans certaines vidéos. L'estimation de la pose de la main constitue le maillon faible de notre système étant données les erreurs de profondeur rapportées par nos expérimentations. Enfin, le post-traitement implique des méthodes de traitements d'images et de clustering qui reposent fortement sur des seuils et des valeurs statistiques nécessitant un réglage adéquat.

ABSTRACT

Music has been an artful way of expression, and since it is based on sounds, musicians transcribed their music so it can be shared and used by others. Several notation systems exist for music, such as the standard music notation and guitar tablature.

When musicians transcribe an existing musical piece or write down a song they are composing, they notate the music manually. This makes the process long, tedious and error-prone mostly for amateur guitarists. Thus, several studies were conducted on automating transcription in many contexts and for different instruments. Some of these methods are based on audio signal processing, while others are based on computer vision or a combination of both.

The instrument of interest in this work is the guitar. The guitar is a musical instrument that has been existing for a long time and it is widely used across a wide spectrum of music styles across the globe, and it is one of the instruments used for research towards developing more robust automatic transcription of tablature. Unlike the standard music notation, which is not specific to any instrument, guitar tablature is. The latter is a notation created specifically for stringed instruments such as the guitar or the bass. Tablature displays which fret to press and on which string, which is a piece of information that the standard music notation lacks, and this is important since the same pitch exists in many different places on a guitar fretboard. Thus, the aim of this work was to develop an automatic transcription system producing a guitar tablature from the analysis of video images of a guitarist playing the instrument.

Furthermore, our system relies only on visual information to perform the task of transcription since we believe that a computer vision approach is more suited for the tablature transcription problem. An approach based on audio processing would increase the complexity of the task, especially when dealing with chords (multiple notes at the same time), frets and strings. In the case of processing audio signal, the pitch can be recognized, however, a string timbre is a must-know since the same pitch can exist on different string/fret combinations; in addition to requiring to know the tuning of the guitar beforehand.

The system developed in this work, uses videos that encodes both color and depth information through the use of a RGBD camera. The RGB frames are used in all stages of the project while depth maps are only used during hand pose estimation and post-processing. Indeed, the approach adopted in this work is comprised of many stages that include segmentation, hand pose estimation, optical flow and post-processing.

The first stage, segmentation, is made of a deep learning model that is trained on a custom built dataset for fretboard segmentation. This model outputs a binary mask representing the pixels pertaining to the fretboard since our region of interest and the main guitar part from where our information are drawn, is the fretboard.

Next, hand pose estimation is performed using MediaPipe Hands which predicts the 3D coordinates of 21 hand joints. The results of this step are used later during post-processing to know which joints are pressing what frets and what strings.

Since pressing a string on a particular fret is not enough to transcribe, optical flow had to be used. The optical flow is computed since we need to detect if a string was picked in order to transcribe. Pressing strings while not picking them is equivalent to not playing anything thus the need for optical flow. The result of this step is further processed later to detect which string was picked.

The last step of our approach, is post-processing. Post-processing uses the segmentation result to isolate the fretboard from the rest of the image, then uses that to detect strings and frets, before using both the optical flow and the detected strings to recognize the picked string. The last task of this step, is to decide which joint is pressing on which string and on what fret, which is done by comparing a joint depth to that of the fretboard while also finding the closest string and the fret on which it lies.

After developing our methodology, we validated it. The segmentation step was validated by testing on the test set of our custom built dataset, presenting a score of 0.9230 for Dice. The hand pose estimation was evaluated on captured images of the hand and the errors were presented and measured in millimeters. Then, we recorded videos of people playing the guitar and assessed our ability of picking detection and tablature extraction (entire approach) by comparing our method output to the ground-truth tablature.

Our results show that our segmentation model is robust and results are satisfactory. The data collected to train our model, helped achieving a better generalization due to a varied dataset. Picking detection was unable to perform well with thin strings which may require using a higher resolution in the future to address this. The hand pose estimation part proved to require improvements the most, due to its errors in terms of depth and instability. Finally, the post-processing part relies on traditional image processing and clustering algorithms which is not ideal since they rely mostly on static parameters and thresholds.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

2D	Two dimensions
3D	Three dimensions
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CRNN	Convolutional Recurrent Neural Network
CS1	Coordinate System 1
CS2	Coordinate System 2
fps	Frames Per Second
GB	Gigabytes
GPU	Graphics Processing Unit
HPC	High Performance Computer
HSV	Hue Saturation Value
Hz	Hertz
IoU	Intersection over Union
mIoU	Mean Intersection Over Union
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RGBD	Red Green Blue Depth
RMSprop	Root Mean Squared Propagation
RNN	Recurrent Neural Network
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
VGG	Visual Geometry Group

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

One of the most widely played instruments is the guitar. People often play the guitar by ear, listening to the music and then trying to reproduce, or by reading a certain written form. This written form is called a notation. Notations are common in the music community and different instruments require different notations. Musicians transcribe existing musical pieces or write down music under composition. When doing so, the process is usually long and tedious; it can also be error-prone especially for amateur guitarists. In addition, being able to extract tablature out of guitar playing videos would help amateur guitarists finding transcriptions[1], or transcribe easier. It can also benefit professional guitarists, easing their lives by automating and making the task shorter. It can also be used for learning[2] or as a tutor for evaluating performances of guitarists.

Before proceeding, we need to present the guitar terminology to better understand the rest of this work.

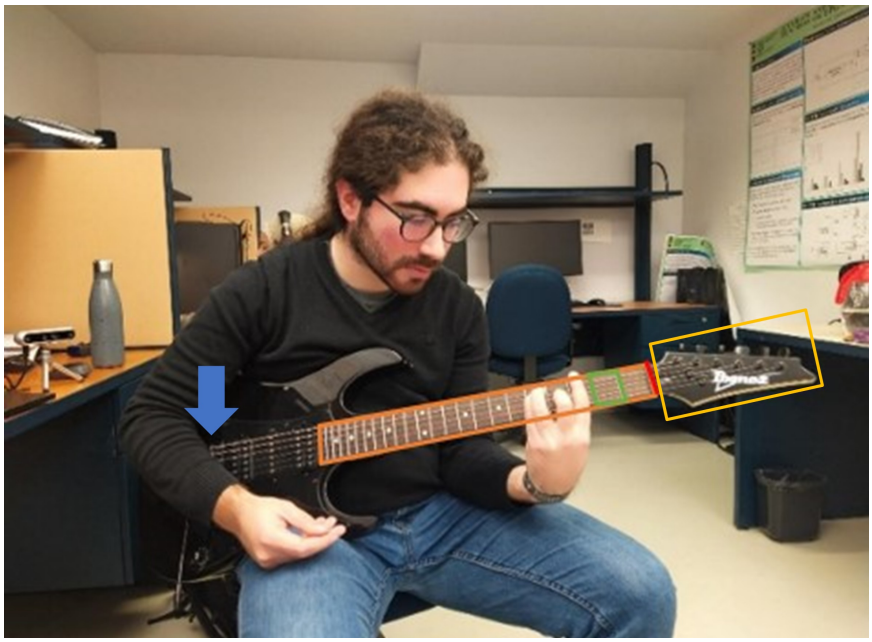


Figure 1.1 The different parts of a guitar. Blue arrow: points at the bridge of the guitar; in orange: the fretboard; in green: a fret (in this case the second fret); in red: the nut; yellow box: the head of the guitar.

[1]Many tablature databases exist such as Guitar Pro Tabs (https://guitarprotabs.org/) and Ultimate Guitar (https://www.ultimate-guitar.com/).

[2]An example application is Yousician: https://yousician.com/guitar

The guitar has different parts, each one with a different purpose. However, we are only interested in few parts that will be listed and explained here (see figure 1.1).

- String: a string on a guitar can be either made of metal or nylon. Regardless of its make, a string is what a guitarist hits to produce a certain note.
- Fretboard: the fretboard is the most essential part of the guitar and the one we are most interested in for visual processing. The fretboard (or neck) highlighted in orange in figure 1.1 is a rectangular wooden piece that holds some metallic bars over which the strings are also positioned.
- Fret: a set of frets constitute the fretboard. A fret is a small piece of the fretboard that ends with a metallic bar (fret bar). Each fret is a semitone, and thus each fret adds one semitone to the note of the open strings on which it is pressed (see green region in Figure 1.1).
- Nut: the nut is the small line highlighted in red in figure 1.1. The nut is the point from which the fretboard starts.
- Bridge: the bridge (pointed at with a blue arrow in Figure 1.1) is a piece that holds the strings from one end. The bridge is located on the body of the guitar.
- Head: the head of the guitar (within the yellow box in Figure 1.1) is a small piece that comes before the nut. On the head we can find tuning keys (the number of tuning keys is always equal to the number of strings); these keys hold the strings from the other end. By doing so, each string is held by both the bridge and its corresponding tuning key.
- Body: the body of the guitar serves as the part that holds everything together, it connects the bridge with the fretboard and amplifies the sound in the case of acoustic or classical guitars or has the volume and tone knob in the case of electric guitars. It is the section of the guitar colored in black.

This project main goal is to transcribe videos of guitarists playing the guitar into tablature. Tablature, is one of several notations that exist to represent music in a written format. Some of those notations are instrument-specific while others are more general. The standard music notation, as shown on the top row in Figure 1.2, is a notation that is widely adopted across the music community and can work with nearly any instrument. Despite the fact that it displays the pitch and octave of each note, this notation is not the first choice for many guitarists.

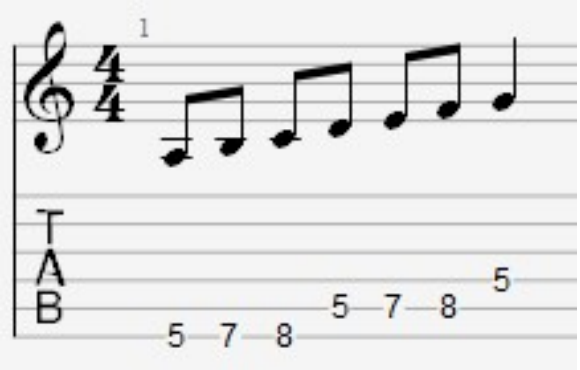


Figure 1.2 The same notes in both notations, standard music notation (top) and tablature (bottom).

The standard music notation may be tricky for some guitarists as it does not display the position as to where the fingers should press. In other words, it does not tell the guitarist what string to pick and what fret to press. This information is important since the same note can exist in multiple different places on the guitar. Figure 1.3 highlights the E_4 (the E note, fourth octave) note across the six different strings on a guitar. A guitar with standard tuning is tuned with the following notes starting from string 6: E_2 (82.41 Hz), A_2 (110 Hz), D_3 (146.83 Hz), G_3 (196 Hz), B_3 (246.96 Hz) and E_4 (329.63 Hz) [5].

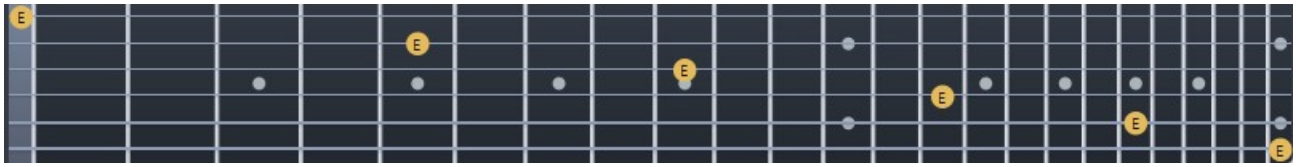


Figure 1.3 The E_4 note highlighted on all strings, showing that the same note can exist on different strings and different frets thus the importance of the tablature notation.

The tablature notation (or "tabs" for short) is a fairly simple notation to read and understand; it contains lines each representing a guitar string (in an upside down manner) on which numbers can be found. Each number represents a fret's index, indexing starts from the nut. Figure 1.4 provides a visual explanation on how tabs can be read.

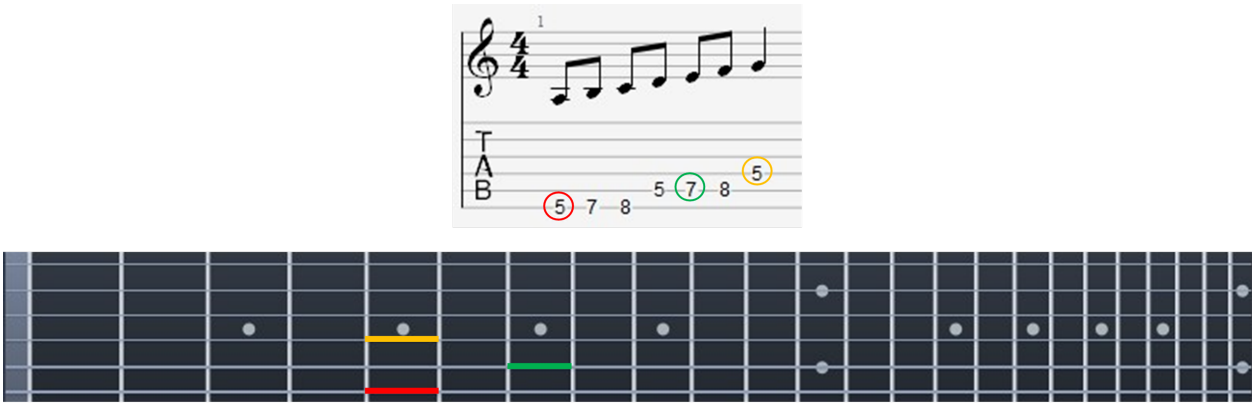


Figure 1.4 The tablature and their corresponding positions on an inverted fretboard (the lines represent the fretboard as if it was held upside down).

A single note at a certain point in time is represented by a single value on one line (guitar string) in terms of tablature. However, sometimes guitar players play multiple notes at the same time, these notes are referred to as chords. A chord is represented in with something like what is shown in Figure 1.5.

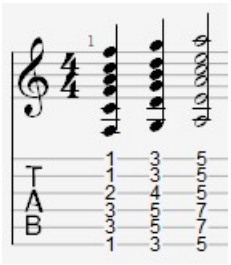


Figure 1.5 Three different chords tabs.

As shown in Figure 1.5, a chord will have multiple fret numbers across different lines placed at the same time position.

Automating the transcription process can be done in a computer vision-based, audio signal processing-based or multimodal fashion. However, we hypothesize that computer vision is enough for the transcription. To this day, despite all the technological advancements, no automatic transcription exists yet. If such a system exists, it would be helpful for guitarists to write down their music automatically instead of spending long hours of transcribing manually. A study on how to develop a robust transcription system may lead to the ability of the automatic transcription of songs and online videos that are often seen on YouTube. Also, a

solid system would help make sure people are learning and playing correctly by comparing the played tabs with respect to the original ones.

Many attempts in the literature addressed the problematic of automatic transcription; however, it is still an open research question. Some approaches relied on computer vision, while others processed audio signals or a combination of both video and audio information for the automatic transcription task. In this work, only visual information will be used for the transcription task as it requires no other equipment than a camera.

Despite the equipment or setup simplicity for a computer vision-based transcription system, several steps have to be followed to achieve such a task. Thus, a full transcription system requires to:

- Locate the fretboard and detect the strings and frets in the image. We will address this step using **segmentation**.
- Detect which fret is pressed by the fingers of the fretting hand. We will address this step using **hand pose estimation**.
- Detect which strings are picked by the other hand (the picking hand). We will adress this step using **optical flow**.

This thesis describes the approach developed and experimented on for the automatic transcription of tablature. This thesis goes as follows, we first start by reviewing the literature for the transcription, segmentation and hand pose estimation. Next, we explain the rationale behind the project, before moving on to the materials and methods, a chapter that explains in details the different steps of our approach. After that we detail the experiments performed to validate our approach before finally concluding the work while stating its limitations and possible future work.

CHAPTER 2 LITERATURE REVIEW

Several attempts in the literature were made to build an automatic transcription system for guitars. Some of them rely on computer vision, others on audio signal processing while a third category rely on using both together. In this section, we present a literature review of the different existing methods who attempt to transcribe/aid guitar learning or are guitar-related. In addition, our method makes use of segmentation and hand pose estimation. Each of those are also briefly reviewed.

2.1 Existing Methods

As mentioned earlier some of the existing methods rely on audio signal processing, computer vision or both. Since in this work, we focus on computer vision, most of the works presented here are computer vision-based approaches.

2.1.1 RGB-based Approaches

We will first start with RGB-based methods. The approaches mentioned here use only the RGB modality in their methodologies.

In [6], the authors mounted a camera on the guitar head that looks down only on the first five frets of the guitar. To isolate the fretboard, background subtraction was performed; then they localize fingertips through the use of the Canny edge detector and a circular Hough transform [6]. After both of these steps are done, the fingertips positions are known by identifying their positions on a grid of strings and frets [6].

Another approach that can be found in [7] relies on using different markers for localization of fingers and the guitar neck. The authors developed mainly as a guitar learning aid system that helps people to learn the guitar, however, it focused on chords [7]. They use two calibrated cameras to compute the position of the fingers in the guitar coordinate system and an ARTag marker to locate the fretboard [7]. Colour segmentation is used for finger tracking, then each channel in the HSV color space is thresholded to keep the coloured fingers markers [7]. Then, by computing the projection matrix on a per-frame basis, the authors were capable of retrieving the 3D positions of the fingers (through triangulation) in the guitar coordinate system where the origin is the corner of the fretboard below the ARTag marker placed near the head of the guitar [7]. Finally using these results the authors can compare the obtained results to a database of chords and check whether the guitar player is

playing a chord correctly [7]. This method, does not transcribe.

In 2010, the authors of [4], similarly to the previous approaches, relied on image processing to implement a tablature transcription system. By using the initial-frame of a desk-mounted camera, an initial frame (without the guitar) is selected for background subtraction [4]. Using Canny edge detector and clustering for strings, they proceed to isolate the fretboard [4]. Next, they use some image processing algorithms and operations like Sobel filter, thresholding, median filter and morphological closure before detecting frets as vertical lines through the Hough transform and finding the edges of the fretboard [4]. Since the fret detection can have some missing frets, the authors use luthier's formula to fill in the gaps, before normalizing the fretboard view [4]. Finally, the authors proceed to detect fingers using RGB thresholds and other conditions, before detecting fingertips and then computing their positions in terms of strings and frets [4].

The authors in [8] also relied on some image processing algorithms like Canny edge detection and a series of others to automatically transcribe a video of a guitarist playing. Using a camera facing the guitar, an initial frame is chosen to initialize tracking of strings and frets [8]. Strings and frets are detected on a per frame basis, however, tracking helps filling the gaps caused by the hand being on top of strings or frets [8]. Once the fretboard localization is finished, the finger contours are found, then the fingertips are found through detecting the local maxima in the upper part of the contours [8]. Finally, each fingertip string and fret are computed and then transcribed [8].

Building upon their previous work in [9], the authors in [10] developed a method for tracking fingertips as well as recognizing the pressed chord on a guitar fretboard. Relying on the guitar neck detection in [9], the authors extended their work in [10] to include chord recognition and fingertip tracking. For the former the approach relies on extracting key frames through finding the biggest differences when using frames subtraction in the RGB color space, key frames are frames where major fretting hand movements happen [10]. By doing so, the authors ensure that they would not need to process all of the frames of a given video, they would only focus on the frames where the chords change [10]. After performing edge detection using the Canny algorithm and morphological dilation, fingers are extracted by checking the pixels against a set of criteria including RGB thresholds [10]. The top left of each finger is considered as the tip and so after computing strings and frets (frets are computed based on the so-called rule of 18 [11]) it is possible to recognize the chord by comparing the tips positions with those of the strings and frets [10]. To perform hand tracking, the authors use a Bayesian pixel classifier that returns only skin related pixels before removing outliers in a later noise removal step in an attempt to remove unwanted or incorrect points, keeping only

the points with the highest probabilities [10].

Other approaches analyze visual cues that are different from fingers detection to extract the played tabs. For example, in [12], the authors fixed a GoPro camera on the guitar running at 240 fps which captures the strings vibrations. By analyzing those vibrations the authors were able to identify each string while also knowing the played note [12]. However, this method requires a priori knowledge and it may be unable to detect thin strings [12], in addition to the need to highly expensive cameras since the frequencies on some strings (especially the thinner ones) can reach 1000+ Hz.

In [13] the authors trained a network to classify five chords. They created their own dataset, used MobileNet v2 [14] for hand segmentation and cropping, before comparing GoogLeNet and ResNet18 for the classification task [13]. GoogeLeNet achieved a 100% accuracy beating ResNet18 [13]. This method only classifies the chord being played, and so it is not useful for transcription especially of single notes. In [15], the authors also used a pre-trained GoogLeNet to classify chords after cropping the hand region to finally generate a musical score.

2.1.2 RGB and Depth based Approaches

Despite the ability of some approaches on relying only on RGB, however, the depth was introduced in other works for reasons such as easier fretboard localization and 3D hand pose estimation.

The authors in [16] attempted developing a guitar teaching aid system that would assess the performance of people learning the guitar. The method first proceed by extracting the hand region from a depth map [16]. Then, a dataset is introduced that was specifically built for hands on guitars, with 16 annotated joints, 4 on each finger except the thumb [16].

Unlike the previous methods, in [17], the authors used a Microsoft Kinect V2 that helps capturing depth information. The authors detect the guitar by applying a depth filter, retaining pixels whose depth values are greater than a certain threshold [17]. The leftmost object is considered as the head area, from there a scan is performed rightwards until the fretboard is reached; after that the head is removed by using a color filter [17]; then they proceed to identify both hands [17]. The authors follow the right hand in order to recognize pickings, however, only tracking the right hand in that manner would make it vulnerable to detecting fake pickings (pretending to pick a string while actually not picking anything).

Similarly to [17], the authors in [18] used a Microsoft Kinect V2. They also applied a depth filter taking into account everything that is closer than the right shoulder depth value, then detecting the guitar using the HSV color space. Once the fretboard is horiztontal [18], the

neck image is used to apply a skin color filter to detect the guitarist's left hand [18]. Later on, a dataset was created to train a CNN on recognizing the pressing finger [18]. This system would not execute unless a pick was detected, this is done by installing a MIDI pickup beneath the strings on the guitar [18]. This method takes into consideration picking and addresses the inability of detecting fake pickings like in [17]. However, this method did not attempt to transcribe, does not consider the pinky finger in the training data and nothing was mentioned regarding chords or technique other than simply pressing using one fingertip [18].

2.1.3 Discussion

Table 2.1 compares the different computer vision-based transcription in terms of modalities used, fretboard localization method, the joints detected and picking monitoring.

The modalities help knowing whether a method has a perception of depth which is important to know whether a finger is pressing or just hovering over a fretboard.

As for the fretboard localization, many methods could be used, however we think deep learning segmentation methods would be very suitable for our case.

Since many parts of a finger can press strings, it is also important to differentiate between methods that detect only the fingertips and methods that considers a finer hand model.

While all the previous points are important, we cannot proceed without checking for picked strings. Indeed, knowing whether a string was picked or no is essential since only pressing a string is not sufficient. It has to be picked to produce a sound, thus it would be the only case where a transcription should happen.

Table 2.1 Comparison table between different computer vision-based transcription methods.

Method	Modality	Fretboard Localization	Finger Detection	Picking
[6]	RGB	Background Subtraction	Tips	Not Monitored
[7]	RGB	ARTag	Tips	Not Monitored
[4]	RGB	Canny, clustering, etc.	Tips	Not Monitored
[8]	RGB	Canny + PPHT + Tracking	Tips	Not Monitored
[10]	RGB	Neck detection and tracking	Tips	Not Monitored
[12]	RGB	None	None	Monitored
[13]	RGB	None	Whole hand	Not Monitored
[16]	RGBD	None	16 joints	Not Monitored
[17]	RGBD	Depth filter + color filter	Whole hand	Monitored
[18]	RGBD	Depth filter + color filter	Whole hand	Monitored

Many approaches in the literature are unable to detect whether a finger is hovering over a string or actually pressing it, such as [8], [4] and [6], etc.

Furthermore, Table 2.1 indicates that all previously mentioned methods have no way of monitoring picking except for [18] and [12]. To achieve this goal, there are several options, such as using a MIDI pickup like in [18], a camera mounted on the guitar, dedicated for string vibrations detection and processing like in [12], or through audio processing. The use of a MIDI pickup is not a practical way of monitoring for picking since it will have to be installed on the guitar being played, and it may also not be comfortable for the player/guitarist. Using a camera dedicated for vibrations, in addition to a camera that monitors the fretting hand would be a bulky solution. Moreover, the two cameras would need to work synchronously, in addition to the difficulty of matching the strings from one point of view to the other. Finally, on a standard tuned guitar the lowest note is E_2 with a frequency of 82.4 Hz [5], with the highest note being E_6 (1294.54 Hz [5]) for a 24 frets guitar, which requires having a high speed camera running at very high frame rates. Processing audio signals for the sole purpose of knowing when a string was played would add an audio module to the project which does not seem like beneficial enough to do an entire audio setup for such a small task. Also, timbers of each strings would need to be pre-recorded or known beforehand so a detected frequency can be matched to a string.

Finally, most methods perform fretboard localization using image processing algorithms that can be faulty and not robust, since relying on edge and line detections (like Canny edge detector and the Hough line transform) may lead to unwanted results especially if the background has horizontal and vertical lines similar to strings and frets, and relying on predefined thresholds that may not be suitable for all lighting conditions.

2.1.4 Computer Vision and Audio Signal Processing Based Approaches

While the previous methods rely on pure visual data to achieve their goals, others used a combination of auditive and visual information.

The authors in [19] used a 2D representation of audio signals and two CRNNs to identify the root note of each chord and its mode. While in [20], a 2D representation of the GuitarSet dataset [21] was used alongside a CNN to predict the corresponding tablature.

In [22], the authors adopted a multimodal approach where they rely on both visual and auditive information to extract tablature information. By detecting the fretboard first through a series of image processing algorithms on the first frame (Hough transform, Canny edge detector, etc.), tracking the fretboard and detecting the hand, the fret-string combination

can be found [22]. Using both audio signal processing (for extracting the pitch played) and computer vision would reduce ambiguity as per [22].

2.2 Segmentation

Despite this project being about the automatic transcription of guitar videos, segmentation is used to isolate the fretboard pixels from background pixels in a frame. The segmentation is needed since when processing images, we are only interested in the fretboard and everything else in that frame can be discarded. Several approaches exist that can be used for the fretboard segmentation. Apart from the classical image processing methods used in previous work on guitar tab transcription (see previous section), we will focus here on deep learning based segmentation techniques. U-Net [23] is a CNN that performs per-pixel classification. U-Net is a network composed of two paths, a downward and an upward path [23]. The former is composed out of convolution layers, ReLU layers and max pooling while the latter is identical, replacing max pooling with upsampling; these two paths are connected by skip connections [23].

Another well-known segmentation network is Mask R-CNN [24]. Mask R-CNN is an instance segmentation model that is built on top of a detection model called Faster R-CNN [24]. Faster R-CNN is composed of a CNN that outputs a feature map of the input image, which is later passed through a region-proposal network (RPN) that proposes objects to the rest of the network [25]. A RPN is a binary classification network that classifies regions in an image as objects or non-objects, in case of the former, these proposals are passed on to the rest of the network where we have a classification and a regression head that classifies the proposed object and determines the bounding box offsets respectively [25]. Mask R-CNN extends Faster R-CNN by adding a mask head that predicts a mask for each detected class among the classes it is trained on [24]. In addition to the classification and bounding box regression losses used in training, to train for accurate mask prediction, Mask R-CNN adopts the binary cross entropy to train the segmentation head [24].

Besides the two previously mentioned models, several other models exist for semantic segmentation. One particular paper conducted a comparative study between three models for the segmentation of guitars [26]. The authors in [26] created a dataset composed of 10400 images of people holding a guitar alongside their masks as the groundtruth data. Between the models compared in this study, HRNet [27] outperformed DeeplabV3+ [28] and DenseNet [29] [26]. They used the mean intersection over union (mIoU) metric, and HRNet had the highest mIoU despite being slower in training time [26].

2.3 Hand Pose Estimation

After isolating the fretboard, we need to know where is the fretting hand relatively to the fretboard and which fret it is pressing. This can be achieved using hand pose estimation.

Before proceeding, we present some definitions relating to the task of "Hand Pose Estimation" to promote a better understanding of the project.

Hand pose estimation aims at identifying and locating the different joints in a human hand. The position of these joints can serve as reference points for multiple applications, like gesture recognition. In this section we will explain the hand joints that we consider in this project for a better understanding of later parts or sections.

In our project, we are interested in 21 joints of the hand, 1 joint representing the wrist and 4 joints for each finger. The following figure illustrates the different joints that we will be using in our project.

Figure 2.1 The 21 joints on the hand. Based on [2], these joints are: CMC: Carpometacarpal joint, MCP: Metacarpolphalangeal joint, PIP: Proximal Phalangeal joint, DIP: Distal Phalangeal joint, IP: Interphalangeal joint and the tip © Jake , Dan, Public domain, via Wikimedia Commons. Image taken and modified from [3]

Based on [2] and as shown in the figure these joints are the following:

- CMC: Carpometacarpal joint. This joint will only be predicted for the thumb thus the Figure above does not show this joint for other fingers.

- MCP: Metacarpolphalangeal joint. This joint will be considered on all fingers.
- PIP: Proximal Phalangeal joint. This joint exists on all fingers except the thumb.
- DIP: Distal Phalangeal joint. Similar to the previous joint, it exists on all fingers except the thumb and we will be looking for predicting its positions across all fingers on which it exists.
- IP: Interphalangeal joint. This joint exists only on the thumb.
- Tip: this is the tip of the fingers.

Several methods exist in the literature that address the problem of hand pose estimation. Some of those methods are presented below.

Being one of the top performers in the Hands In the Million Challenge (HIM2017), V2V-PoseNet is a 3D detection-based hand pose estimation CNN [30] [31]. Before predicting the coordinates of the joints, the the center-of-mass is computed and a crop is performed around it [31]. However, since this method is not robust enough, the authors trained the hand center estimation network from DeepPrior++ [32] to output correcting offsets from the previously computed center-of-mass to the correct one [31]. After that, the 2D depth map is voxelized and given as input to the network which is composed of several blocks [31]. Since V2V-PoseNet is a detection-based hand pose estimation model, the model outputs for each joint the per-voxel likelihood, and so the voxel with the highest likelihood is the voxel selected for each joint [31].

A method was proposed in [33], where the authors combined the advantages of both detection and regression-based models. AWR (or Adaptive Weighting Regression) extracts dense representations for the hand image given as input and then proceeds to weigh the different regions in feature maps before predicting the coordinates of each joint [33]. This was particularly helpful in the case of occlusion, where in order to infer the positions of different hand joints, the weight was more distributed (bigger standard deviation) helping the model to predict the positions based on their neighbourhood. This approach helped AWR perform among the state-of-the-art [33].

Another hand pose estimation network is MediaPipe Hands [34]. MediaPipe Hands is a hand pose estimation and tracking model developed by Google that estimates 21 hand joints (the 21 hand joints mentioned in the introduction) [34]. Unlike the previously mentioned approaches, MediaPipe Hands expects an RGB image as an input and outputs 2.5D landmarks of the hand joints, where each joint has its own x, y and z coordinate [34]. These coordinates are

relative to an origin defined in the image, and this origin could either be the wrist joint itself or the centroid of the hand [34] [35].

MediaPipe Hands consists of two stages, the first being a palm detector and the second being the hand landmark model [34]. The latter is the part that predicts the coordinates of the joints, in addition to a confidence score that indicates the model confidence about a hand presence and a flag entitled "Handedness" that tells us whether the detected hand is a left or a right hand, the model predicts 21 joints 3D coordinates [34]. Google had to create their own dataset to train this model using the focal loss [34].

MediaPipe Hands proceeds by detecting the hand in a certain frame and continues by tracking it, the detector will not be used again until the tracking confidence of the model falls below a predefined threshold [34]. MediaPipe Hands provides the ability to obtain the 3D coordinates of the joints in two different ways; the first using the wrist as the origin [34] [35]. In this case, the joints x and y are ratios $\frac{r}{h}$ and $\frac{c}{w}$ where r and c are the row and the column of the joint in the image (w.r.t. to the image origin; top left pixel) while h and w are the image's height and width respectively [34] [35]. However, for the z coordinate, it is relative to the wrist, with the latter being the origin in terms of depth [34] [35]. The small the number the closer the joint is to the camera [35]. The second method consists in returning the 3D coordinates in meters while considering the hand centroid as the origin [35].

CHAPTER 3 PROJECT RATIONALE

In this chapter, the project rationale is presented. Before enumerating the different challenges and objectives of this project, we discuss what can be used from the previous work.

3.1 Rationale

As shown in the literature review, several papers have addressed, at least partly, the automatic transcription from visual data. However, a good number of these methods do not monitor picking, and for the ones who do, picking is either used as the main source of prediction such as in [12] or for monitoring hand movement by using a MIDI pickup as in [18]. Thus, the need for a new way of detecting picking that will not add extra hardware on the guitar in addition to not completely relying on it.

Furthermore, in addition to the many fretboard localization methods proposed in previous work, we believe that a deep learning-based approach is the most suited to the problem because of its well-demonstrated performance on similar segmentation tasks.

While others have considered only the detection of the fingertips for tabs transcription, we will be using in this project hand pose estimation since other parts of the hand can be used for pressing strings, a barre chord for example.

Existing hand pose estimation models estimate the depth of each joint and this is particularly important to our case since a hand over the fretboard does not necessarily means that it is pressing. Omitting the depth information and processing videos or images in 2D can lead to errors, since 2D models can be tricked into thinking that a finger or joint is pressing while in terms of depth the hand is considerably closer to the camera than the guitar and does not even touch the fretboard.

A fretting hand on a guitar can have multiple self-occlusions resulting in some fingers or joints not being visible to the camera. This point raises the need to have a model that would predict joints depth instead of just relying on 2D coordinates of a joint and retrieving its third coordinate from the depthmap.

3.2 Requirements and Challenges

The task of transcribing tablature from videos includes several requirements and challenges that have to be addressed to have a good transcription system able to transcribe as accurately

as possible. We first start by listing the requirements for our projects, then we list the challenges that we need to address in order to fulfill those requirements.

3.2.1 Requirements

The first requirement is having a depth modality. Depth is needed since not having a perception of depth could make a system fall in misinterpretation. This means that if a hand is placed in front of the guitar while not actually touching it (hovering over the fretboard), in a third person perspective it may be considered as pressing or touching the fretboard since depth is not available.

A second requirement is to detect the frets and strings as accurately as possible. This is very important so we can clearly predict the played strings as well as the frets. In addition, it implies the importance of choosing the best model possible for the fretboard segmentation to segment it as accurately as possible.

A third requirement is to correctly identify the vibrating string. Without monitoring the string vibration, the idea of transcription would not be complete. Indeed, placing the fingers on the fretboard is not sufficient to look at, since if no music is played, hence no strings were picked then nothing should be transcribed.

3.2.2 Challenges

Based on the aforementioned requirements, several challenges can be highlighted. These challenges are listed below.

The first challenge is to deal with the possible presence of visual occlusions. The occlusions may include a hand not pressing on a guitar (in other words, just hovering) or a finger on top of another, thus the importance of hand pose estimation and depth perception.

Second, the lack of data. Developing a deep learning based method for the fretboard segmentation requires a considerable amount of labeled data. When we first started this project, no public dataset existed for this task (before the dataset in [1] was released); thus we had to build a new dataset from scratch. This involves data collection and data annotation. However, the data collection has to be as diverse as possible; meaning that the data must include as much different guitars as possible since they all differ in shapes, looks, dimensions and some of them differ in type (classical, acoustic and electric).

Third, strings can be very thin especially when dealing with metal strings on electric guitars. Higher strings are also thinner than others, hence proving their difficulty to be detected. For

the frets, however, some of them may be hidden by the hand being on top of the fretboard, hence the need to find a way of addressing that.

3.3 Objectives

The main goal of this project is to develop an automatic way of transcribing a video of a guitarist into tablature. To do this, several steps need to be taken and several objectives need to be satisfied; these objectives are the following:

1. To develop a model to localize strings and frets: as explained in Section 1, this notation has lines representing strings and numbers indicating frets indices; and to have as much accurate tablature as possible as an output, a correct and precise detection of the fretboard, its strings and frets is required;

2. To estimate and to track the hand pose in 3D: locating fingertips in a hand is not enough, as other parts of the finger may be used for pressing a string on a fret. Thus, the need for an estimation of different joints of a hand. We also require 3D information to know whether a finger is pressing on the fretboard or not. The position or the placement of capture is important as it can greatly affect the results. Different views can help us improve the accuracy as we maximize the number of visible joints. However, for this project we used only one viewpoint by placing the camera in front of the guitarist to assess how well it would perform;

3. To extract the corresponding tablature of an RGBD video using the fretboard, strings, frets and hand data;

4. To evaluate the proposed approach: to know whether our approach performs well, an evaluation must be conducted.

With the objectives listed above, our approach consists of several steps to satisfy those objectives. These steps include:

- Recording videos in 3D so we can have a depth perception of the scene and be able to use that information later.

- Collecting and annotating images of subjects playing the guitar to populate our own dataset.

- Training and validating a CNN model for the fretboard segmentation.

- Estimating the pose of the fretting hand in 3D. This part involves predicting the coordinates in 3D (x, y and z) for the different joints of a hand, which we can use later to identify where each joint is located on the fretboard.
- Detecting vibrating strings. Indeed, we need to check for picked/plucked strings before transcribing; pressing a string on a particular fret without actually picking the string does not imply that a transcription has to be made.
- Integrating the fretboard segmentation, the hand pose estimation as well as the picking detection with some additional image processing techniques in order to compute the tablature before writing it down in a simple ".txt" file.

CHAPTER 4 MATERIAL AND METHODS

In this chapter, the different steps of the proposed solution are expanded and explained in more details. First, we start by an overview, then we go over each step in details. A thorough explanation of how they work and how they were implemented is given.

4.1 Overview

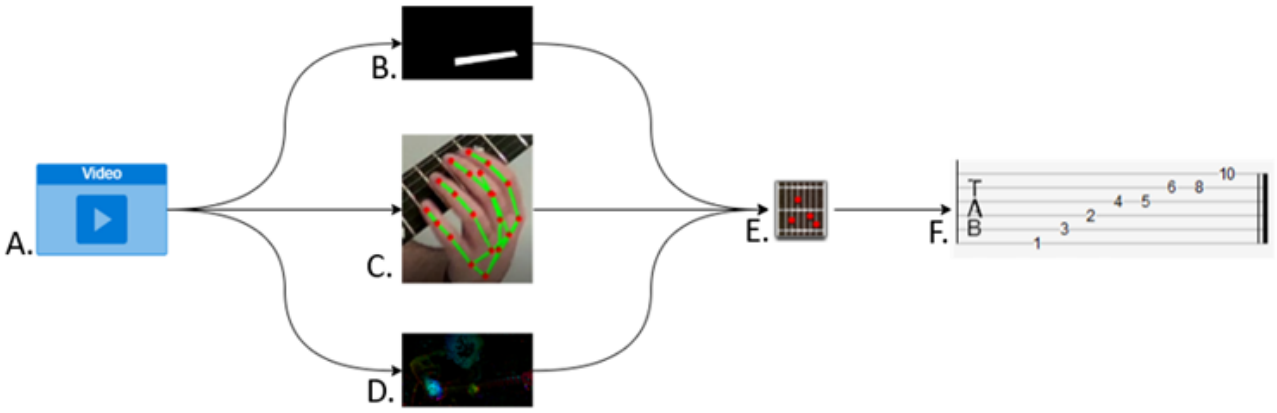


Figure 4.1 The different steps of the proposed method. A: RGBD video input. B: Fretboard segmentation. C: Hand pose estimation. D: Optical Flow. E: Post-processing. F: tablature output.

Figure 4.1 illustrates the entire methodology adopted for the project on a high level. The methodology contains four main steps (B to E), an input (A) and an output (F). In this section, we will explain the different parts separately and the reason or the need behind every single step in the project and desired or expected outcome of each one of them.

4.1.1 Input A: RGBD Videos

For our approach to work, the need for depth information is inevitable as we seek to know the position of the hand in 3D space with respect to the guitar fretboard. In addition, the RGB modality is still needed to perform the segmentation task and some post-processing.

To capture RGBD videos, we need to use RGBD cameras or otherwise known as depth cameras, these cameras can capture both modalities, RGB and depth.

Intel offers a series of depth cameras known as Intel RealSense, these cameras adopt technologies such as time-of-flight or active stereo vision to compute the depth map of a scene.

Despite that these cameras are widely available for anyone to buy on the market, they are very unlikely to be found with guitarists. However, they are relatively inexpensive, in the range of 300 to 500$ [1].

Among the several RGBD consumer-grade cameras that exist today, we can find the Intel RealSense D400 series. In this series, we took a look at two cameras for this project; these two cameras are the Intel RealSense D435 and the Intel RealSense D455. They both use the concept of active stereo vision [36].

These two cameras are similar in terms of components and the way they operate. They both have two depth sensors, one infrared light projector and one RGB sensor. However, these components are aligned differently between the two cameras. For the D435[2], these components are as follows from left to right: right image sensor, infrared light projector, left image sensor and the RGB sensor. While for the D455[3] the order is: right image sensor, RGB sensor, infrared light project and left image sensor, as shown in figures 4.2 and 4.3.

Figure 4.2 The Intel RealSense D435 RGBD camera.

Figure 4.3 The Intel RealSense D455 RGBD camera.

In addition, the ideal range for operating for the D435 is between 0.2 to 3 meters while the D455 operates best in a 0.4 to 6 meters range [37]. The D455 has a wider baseline of 95

[1] https://store.intelrealsense.com/
[2] https://www.intelrealsense.com/depth-camera-d435/
[3] https://www.intelrealsense.com/depth-camera-d455/

millimeters, while the D435 has a baseline of 50 millimeters [37].

Although both cameras are highly similar, the D455 was chosen for this project for two reasons: it has a wider baseline resulting in less noisy depth estimation and it also has higher RGB image quality.

4.1.2 Step B: Fretboard Segmentation

A segmentation of the fretboard is essential for any automatic guitar transcription that is based on computer vision, since the fretboard is the part of the guitar based on which we can know the played strings and frets (in other words notes). More precisely, in this part of the project we are looking for a segmentation model that can predict a mask for a given input image, and as an output we would get a binary mask where white pixels would stand for the pixels that pertain to the detected fretboard in the given image.

This step is inevitable since in the post-processing, the need for a clear fretboard localization is crucial. Locating the fretboard accurately would boost the string detection and fret detection and allows the used image processing algorithms in the post-processing step to work better.

4.1.3 Step C: Hand Pose Estimation

Most existing methods rely on just detecting the fingertips, however, that is not enough for guitars. The fingertips is not the only part of a finger that can press or be used to press on a guitar, thus the need for something that would get us different parts of the hand where we can know if a part of the finger (other than the tip) is pressing.

Some techniques played on the guitar (like bar chords) involve pressing not just with the fingertip but with the entire finger. In addition, other techniques such as the perfect fourth interval, for example, an A perfect fourth can be the fifth fret on both the sixth and fifth string both pressed with the same finger (we consider here that the tuning is E standard).

4.1.4 Step D: Optical Flow

The optical flow allows us to know whether something has moved between two consecutive frames. In our case, we are interested in knowing whether a string has vibrated or no. This is important, since if a guitar player presses some strings but never picks any string then no transcription should happen. Thus, monitoring for vibrating strings before transcribing is actually needed and important to transcribe the played frets only.

4.1.5 Step E: Post-Processing

After finishing the three previous tasks, a post-processing step is needed to perform needed operations on their outputs and find the corresponding tablature.

This step consists of performing a series of algorithms (clustering, image processing, linear regression, etc.) to process the segmented fretboard, locate the strings and the frets, find the pressing joints and their positions on the fretboard; while also checking if a string has vibrated by using the optical flow output. Figure 4.4 illustrates a high-level outline of this step.

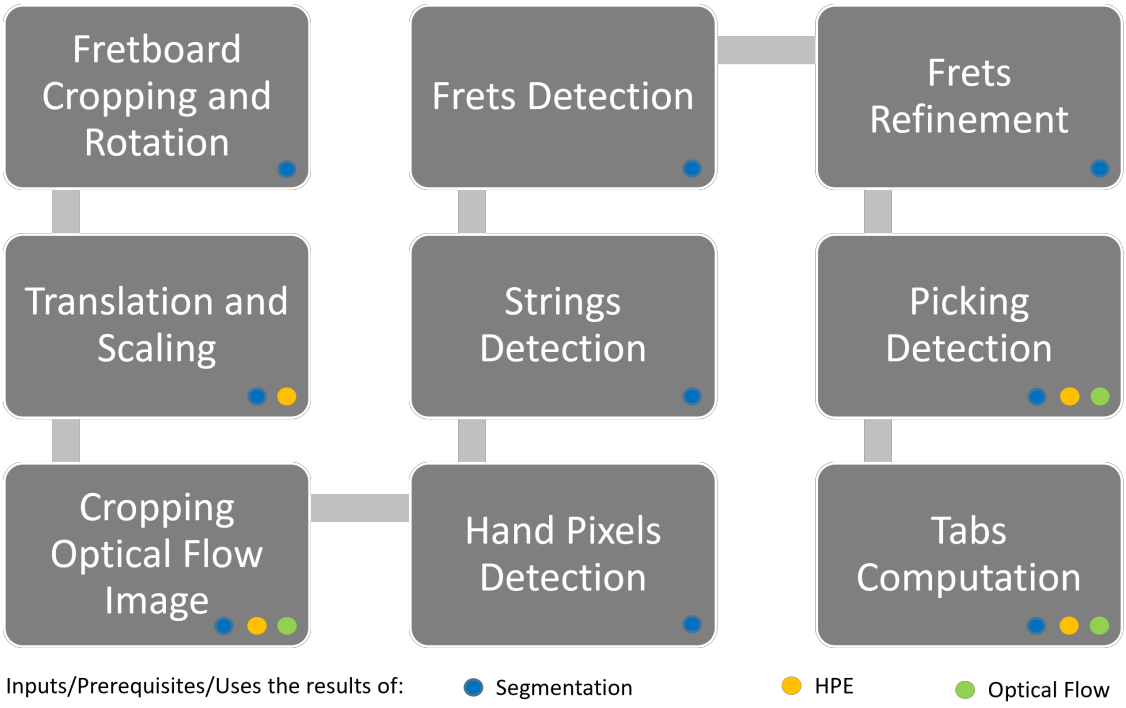


Figure 4.4 Post-processing high-level outline.

4.1.6 Step F: Output

The output given by our approach will be stored in text format. Every two columns of tablature will be separated by an empty column to ease up processing when computing the metrics especially when dealing with two digits fret numbers.

4.2 Fretboard Segmentation

One important part of the project is to segment the fretboard from the rest of the image. This is important due to the fact that most important features that we are looking for are on the fretboard (strings, frets and fingers). The segmentation part is also useful for the picking detection as well, since picking happens right next to the fretboard, right where the picking hand is placed.

4.2.1 Segmentation network

For the segmentation of the guitar fretboard in RGB images, we used a Mask R-CNN model.

To implement Mask R-CNN, we used the Python programming language and the PyTorch and Torchvision libraries. Torchvision already has an implementation of Mask R-CNN[4] with a ResNet50-FPN (ResNet50 with Feature Pyramid Network) backbone. This model was pre-trained on COCO2017 training set[4].

4.2.2 Model training

The dataset used in [26] was never made public. In addition, the annotations include the entire guitar instead of the fretboard only. Thus, we had to collect our own data to be able to perform transfer learning.

Having a pre-trained network allows us to perform transfer learning instead of training the model from scratch; this will lead to a better and faster convergence especially since our dataset is relatively small. To train our model we developed our own dataset in PyTorch which loads the images and preprocesses them before training Mask R-CNN. This dataset class can accept PyTorch image transforms to apply data augmentation.

While loading data, our implementation loads the data from the location specified for the dataset and resizes them. Both the images and their corresponding masks are resized to 512×512 pixels using a bicubic interpolation. The binary masks in the dataset are normalized to the range of 0 and 1.

If the data is being loaded for training purposes, our dataset implementation will perform data augmentation. We adopted an on-the-fly data augmentation in PyTorch, which performs data augmentation once an image is loaded and so the dataset size does not change. Our data augmentation involves horizontal and vertical flipping and rotation; we also input other transforms such as Gaussian blurring and color jittering.

[4] https://pytorch.org/vision/0.8/models.html

During preprocessing, it is necessary to provide the information required for each of the Mask R-CNN heads to train the model. During training, the corresponding labels, masks and boxes coordinates are computed during data preprocessing and returned by our loader.

4.3 Hand Pose Estimation

MediaPipe [34] is a hand pose estimation prediction model that can predict the joints coordinates, predict their "handedness" (left or right hand) alongside a confidence score and track the joints unless the confidence score of the model falls below a certain threshold, forcing the detection to reinitialize.

MediaPipe was chosen over other hand pose estimation models due to its lightness, and ability to work in real-time and little resources requirement. In addition, to our knowledge, no datasets exist for the hand pose estimation on a guitar and no models were trained for such a specific task before.

MediaPipe can output the coordinates of a hand joint on three axes (x, y and z). There are two coordinates systems in MediaPipe that would help us retrieve 3D coordinates of a joint, these are the following:

- The first system (CS1) is the one described in [34] and [35]. This coordinate system considers the top left of the image as being the origin for the x and y axes. These two coordinates are not returned in pixels but in ratios, where both x and y are the pixels positions normalized by the image's width and height respectively. As for the z (or depth) coordinate, the origin (z=0) is the wrist. Indeed, the wrist joint is not a static origin or point, it is dynamic and can differ from one image to the other. In this case, the wrist has a z equal to 0 and for the other joints they have either positive or negative values. The smaller the value the closer a joint is to the camera.

  These values (the z values) do not have a particular interpretation, they do not represent a distance in meters, centimetres or anything similar they are just values that tell how close is a joint to the wrist and the camera.

- The second coordinate system (CS2) has for origin the centroid of the hand for all three axes (x, y and z). Same as the previous system, the smaller the value, the closer the joint associated with that value is to the camera [35]. However, this coordinate system is in meters; the values obtained using this method gives the relative distance of a joint across all axes in meters [35].

An example of hand pose estimation is visualized in Figure 4.5.

(a) RGB image for a hand in the air on which we perform hand pose estimation.

(b) The visualization of the hand pose estimation performed on the image in Figure 4.5a.

Figure 4.5 Same image of a hand held in the air, Figure 4.5a shows the original image and Figure 4.5b shows the same image with the hand pose estimation result displayed on top of the hand

Since we use an Intel RealSense D455, we already have a depth map, however, a hand on a guitar in a lot of scenarios will have self-occlusions (fingers occluding other fingers) and so there is a need to predict the depth of each joint without completely relying on the depth map provided by the camera. To experiment and identify the best method to predict depth, 10 RGBD images were recorded where all joints of the hand are completely visible. By having all joints visible would ensure that we always have the ground-truth for all joints, these ground-truths can be retrieved from the depth map.

Since we need to know the depth with respect to the camera for each joint, we developed the following methods to learn a mapping from MediaPipe coordinate systems to the camera coordinate system:

- Method 1(LR): this method relies on using linear regression to learn a mapping from the depth given by MediaPipe to the real depth (the depth contained in the depth map, or the depth in the camera coordinate system). Using CS1, we attempt to learn a function that gives as an output the true value of depth z_{real} by giving it as input the depth value predicted by MediaPipe; thus, $z_{real} = f(z_{mp})$. Only the closest 8 joints based on this coordinate system are used for learning.

- Method 2 (LR2): this method is exactly the same method as the one described above, however, there is only one difference; instead of using the CS1 coordinate system, we will be using CS2 as input.

- Method 3 (Centroid): this last method is based on the CS2 coordinate system. This system, returns in meters the relative distance of each joint to the centroid of the hand. Thus, by using the 8 closest joints, we can compute the depth value of the centroid. This depth is computed by applying the following formula:

$$C_d = \frac{1}{n}\sum_{i=1}^{n} z_{real}^{(i)} - z_{mp}^{(i)} \tag{4.1}$$

  C_d represents the centroid depth, n is the number of joints taken for the computation (8 in this case), $z_{real}^{(i)}$ is the real depth of the i^{th} joint, while $z_{mp}^{(i)}$ is the depth predicted by MediaPipe relative to the centroid for that same joint. To compute the centroid depth, we retrieve the real depth of the 8 joints from the depth map returned by the camera using their x and y coordinates. Next, we subtract each of these joints MediaPipe depth from the real depth and sum them before taking the mean. By doing so, the centroid depth is calculated and then we can infer the depth of the remaining joints. To do this, for each joint we add the centroid depth to its MediaPipe depth prediction,

$$z_{pred}^{(i)} = C_d + z_{mp}^{(i)}, \tag{4.2}$$

  where $z_{pred}^{(i)}$ is the predicted depth after adding the computed centroid depth with MediaPipe relative depth prediction.

Before we can test, outliers have to be removed from the data. In other words, since the real depth values are retrieved from the depth map, we may have some incorrect or missing values, to remove those outliers, we check the depth of each joint after retrieval from the depth map. A joint i is removed only when $z_{real}^{(i)} - m > 0.3$ (m stands for the median depth of all joints ($m = median(\{z_{real}^{(1)}, z_{real}^{(2)}, ..., z_{real}^{(21)}\})$, 0.3 is for 0.3 meters).

Figure 4.6 illustrates a depth map obtained through the D455 camera.

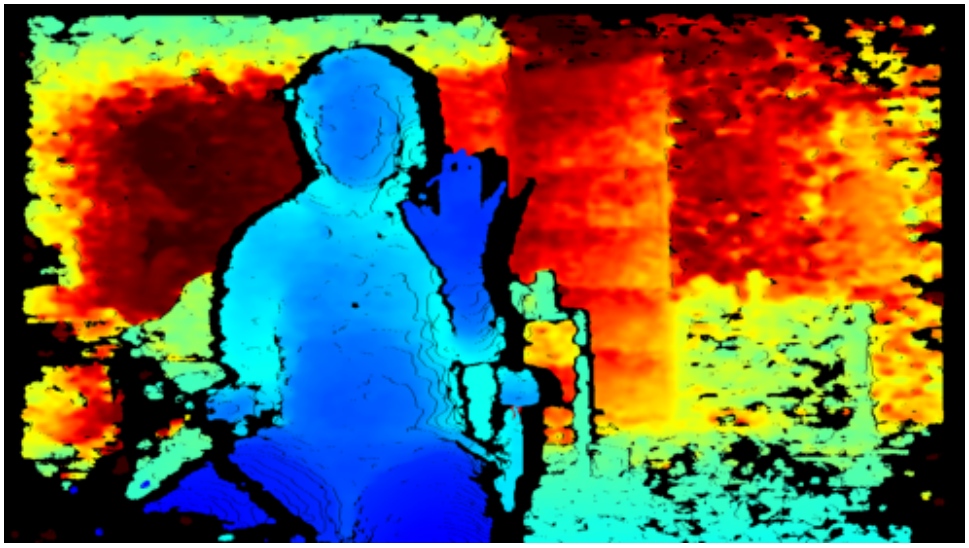


Figure 4.6 Colorized depth map for Figure 4.5a.

4.4 Optical Flow

While playing the guitar, it is not enough to press a string to consider that it is being played, it has to be picked/strummed. This is true due to the fact that holding a string without plucking it would not produce any sounds, thus no pitch and no music. Hence, this highlights the need for a picking detection method that would help us know what string was picked.

The solution adopted for this problem is using the optical flow. Indeed, the optical flow can detect movements in a video or two different frames/images and can detect string picking. On a guitar with an E standard tuning, the lowest frequency (6^{th} string played openly) is 82.4 Hz [5](E_2 note) while the highest could be 1318.51 Hz in the case of a 24 frets guitar (1st string, 24^{th} fret, E_6 note) [5]. With such frequencies, the optical flow may not detect the vibrations or the movement since they are subtle and very fast compared to a camera that works around 30 fps. However, when a string is being picked, the latter is deformed at the moment of picking and that is where the optical flow could come in handy.

(a) Frame showing no picking of the 6th string.

(b) Frame showing the slight deformation of upper string while being picked.

(c) Frame showing bigger deformation of the string from the same motion as Figure 4.7b.

(d) String form coming back to normal after picking end.

Figure 4.7 Video images displaying the deformation of the 6th (thickest) string on a guitar while being picked. All these figures are from the same motion. a) the last frame before the actual picking begins. Figure b) the frame right after the one displayed in Figure 4.7a. c) the frame right after Figure 4.7b, showing even more deformation during picking. d) the frame right after 4.7c, showing the form of the string coming back to normal after the picking ends.

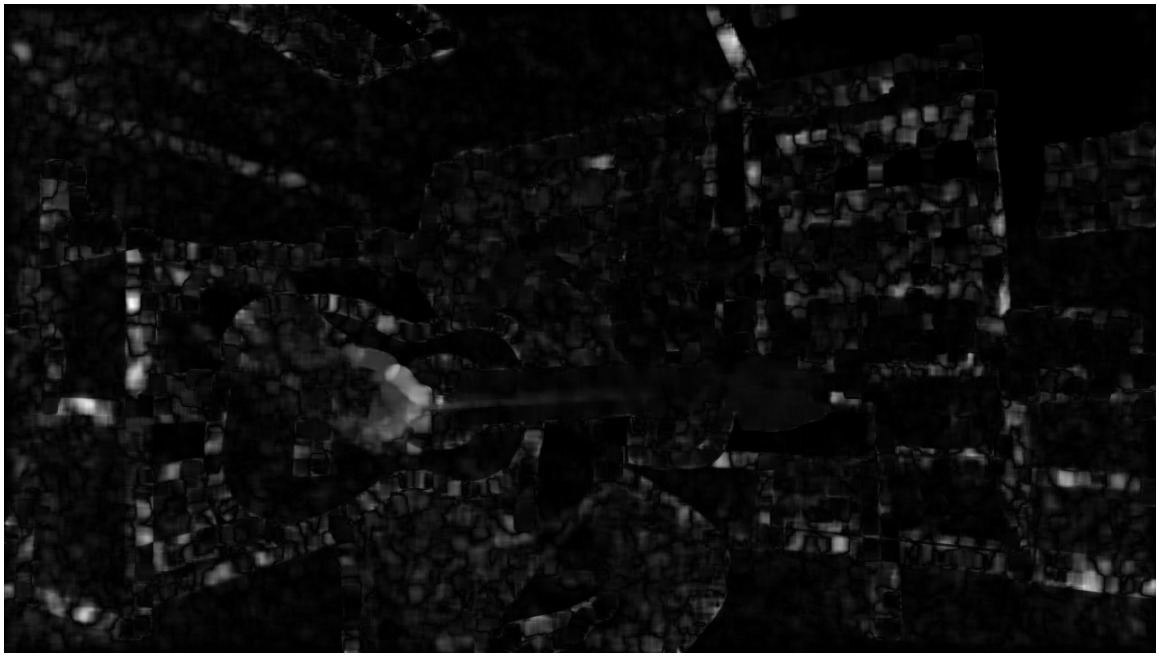
Figure 4.7 illustrates how the shape of a string changes during picking and this is what the optical flow could use for the detection of movement on a string. Indeed, it is not the vibration itself that it is detected as a movement, but the action of deforming a string while being picked since the latter moves during the picking action.

To compute the optical flow, we use OpenCV implementation of the Farneback[5] dense optical flow [38]. This computes the optical flow of two subsequent frames on the entire image and give results like the one shown in Figure 4.8.

(a) RGB frame where the third string is being picked from one of the RGBD videos recorded for the experiments.

(b) RGB frame during picking that comes after 4.8a.

[5] https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html



(c) The computed optical flow between the two successive frame shown in Figures 4.8a and 4.8b

Figure 4.8 Three figures where a) shows the moment a string is being picked, b) is the frame right after a) where the string is still being picked however the deformation is greater. c) is the optical flow result computed for the aforementioned two images, clearly showing a line representing the picked string.

As shown in Figure 4.8c, we can see a line that is a bit brighter than its neighborhood, that goes in the same direction as the fretboard. This line indicates a string was picked, and based on the position of that line we can in post-processing know which string it is.

4.5 Post-Processing

The need for all these different modules (fretboard segmentation, hand pose estimation and optical flow) is to allow our post-processing to use them to extract tablatures based on the outputs given by these three modules.

This step consists of different sub-steps and different image processing algorithms and techniques (like Sobel filter, thresholding, line detection using the Hough transform), in addition to other algorithms and methods like K-Means and Mean-Shift clustering, etc.

After outputting the fretboard segmentation mask using Mask R-CNN, the 3D coordinates of all hand joints using MediaPipe Hands and the optical flow using OpenCV Farneback dense

optical flow implementation, these three outputs are passed to the post-processing step of the project to further process these information and extract the corresponding tablature.

The first step is to crop the fretboard and rotate it.

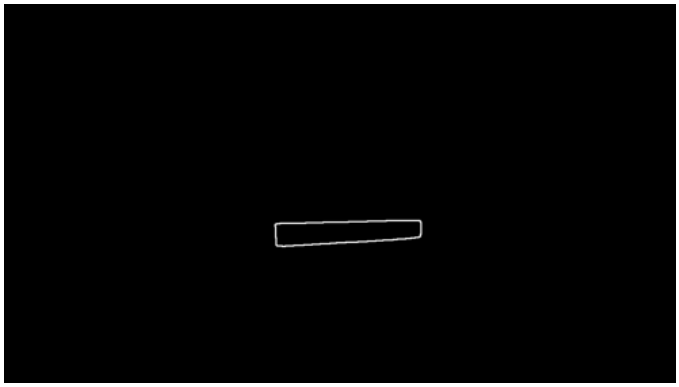
4.5.1 Fretboard Cropping and Rotation

The mask outputted by Mask R-CNN may not be a perfect rectangle thus it would not allow us to define four points around with which we can crop the corresponding frame to only keep the fretboard isolated from the rest of the image. Therefore, to normalize the appearance of the fretboard, we rotate it to be horizontal and make it rectangular. To achieve this, we first extract the contour and find the bounding rectangle around the fretboard contour.

To be more specific, if we have a mask like the one shown in Figure 4.9b for the frame displayed in Figure 4.9a, the contours obtained would be the one in Figure 4.9c.

(a) The RGB image used for the segmentation of the fretboard.

(b) The output of the segmentation model, a binary mask representing the pixels pertaining to the fretboard in white.



(c) The detected contours for the binary mask shown in Figure 4.9b.

Figure 4.9 Figures displaying the RGB frame used in a), the segmentation binary mask predicted by Mask R-CNN in b) and the detected contours of the mask in c).

The bounding rectangle of this contour is shown in Figure 4.10.

Figure 4.10 The fretboard mask after fitting to it the smallest possible rectangle.

Next, we rotate the mask, RGB image, optical flow and the depth frame obtained through the RGBD camera so that they are horizontal. The result is displayed in Figure 4.11.

Figure 4.11 The rotated fretboard contours to make it horizontal to ease later processing.

The result shown in Figure 4.11, needs to be processed to remove some noise. To do so, we find the four lines that define this rectangular area. This is simply done through defining horizontal and vertical line equations and fill everything in between to avoid unwanted noise, helping us obtain a mask like in Figure 4.12.

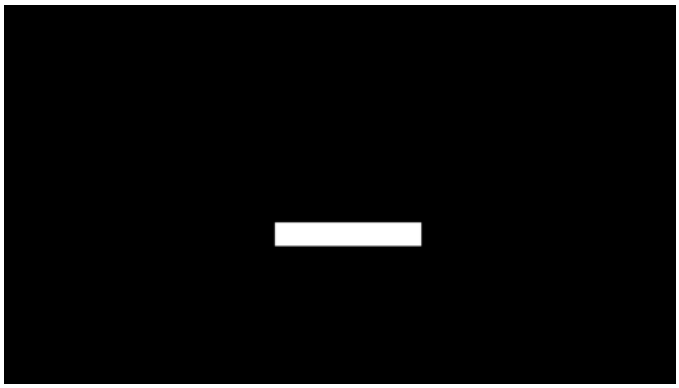


Figure 4.12 Final mask after shaping a perfect rectangle without any noise or unwanted pixels.

Before cropping the fretboard, the latter has to be completely isolated. In other words, the mask shown in Figure 4.12, has to be multiplied by the RGB image, pixel-wise product. By doing so, we make sure that only the pixels that are in white in the final mask, are taken into account in the RGB image, and everything else is removed due to irrelevance to the fretboard. After this multiplication occurs, the result is as in Figure 4.13.

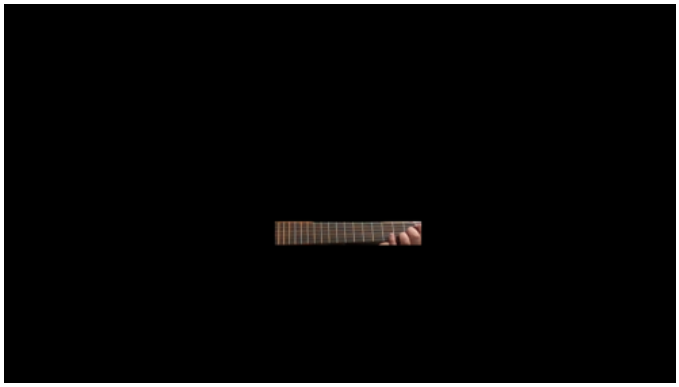


Figure 4.13 The RGB image after the pixel-wise multiplication with the mask.

After obtaining the image shown in the Figure 4.13, the coordinates of the four corners of the mask can be used to crop this image and remove all the pixels in black around the remaining colored pixels. We can use the top left and the bottom left corners y coordinates in addition to the bridge's and nut's ends x coordinates for cropping. The result is displayed in the figure 4.14.

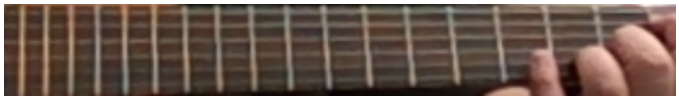


Figure 4.14 The RGB image of the fretboard after cropping.

The depth image is cropped the same way the RGB image is; pixel-wise multiplication and then using the four corners to crop the depth map. Both the depth and the cropped RGB images are resized to 500x68 pixels.

4.5.2 Translation and Scaling

After cropping the fretboard in the RGB image, we need to find a translation vector and a scaling vector. These vectors are important since after performing the hand pose estimation, we rotate, crop and resize the fretboard, and so the same operations have to be applied on the predicted hand joints coordinates to retrieve their coordinates in the coordinate system of the image shown in Figure 4.14.

In this case, the rotation matrix is the same rotation matrix as the one mentioned in the previous step. For the translation vector $\vec{T}$ this would simply be the coordinates of the top left corner of the mask shown in Figure 4.12 since the latter will become the origin after cropping. The translation vector $\vec{T}$ is given by:

$$\vec{T} = \begin{bmatrix} x_o \\ y_o \end{bmatrix} \tag{4.3}$$

It is a column vector having two elements. x_o and y_o are the x and y coordinates of the top left corner of the rectangle shown in Figure 4.12.

For the scaling vector $\vec{S}$, the need to find the ratio between the original size of the cropped fretboard (before resizing) and after resizing. Before resizing the cropped fretboard image, we can compute the size of the fretboard in terms of pixels. Hence, the $\vec{S}$ vector would contain the ratio between the new width (500) and the old width; in addition to the ratio between the new height (68) and the old height. $\vec{S}$ is given by:

$$\vec{S} = \begin{bmatrix} \frac{new\_width}{old\_width} \\ \frac{new\_height}{old\_height} \end{bmatrix} \tag{4.4}$$

After obtaining those two vectors, we can now rotate, translate and scale the 2D coordinates (x and y) of all the predicted hand joints by our hand pose estimation step. To do this, the following equation should be applied:

$$(R \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - \vec{T}) \odot \vec{S} = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{4.5}$$

where the R matrix is the 2x3 rotation matrix from earlier. This matrix would be dotted with a 3x1 column vector representing the hand joint homogeneous 2D coordinates. After that, a 2x1 vector is obtained (representing the same coordinates after rotation), this vector will be subtracted from the $\vec{T}$ vector to translate it, before multiplying the result, element-wise, with the scaling vector to scale the coordinates. Finally, we obtain a new vector with *x'* and *y'*, representing the new coordinates of the hand joint in the coordinate system of the cropped fretboard image.

4.5.3 Cropping the Optical Flow Image

As the cropping performed on the RGB image for the fretboard, a similar task needs to be performed on the image resulting from the optical flow step. This is needed since we are only interested in any action that happens on the guitar strings, in other words, we are only interested if there is a string that was picked. Furthermore, the optical flow is expected to return something that resembles a line as output inside the fretboard in the image; something

similar to the image shown in Figure 4.8c.

To begin, the y values used for the cropping are the y coordinates of the top corners and bottom corners of the mask shown in Figure 4.12. However, for the x coordinates used the procedure is slightly different. Instead of returning only the fretting hand from our hand pose estimation step, we return both hands detected by MediaPipe. The rotation, translation and scaling explained earlier, applies to both hands. However, sometimes no picking hand may be detected, hence we will explain next how the cropping works in both settings.

- No picking hand detected: the width of the fretboard is calculated by subtracting the x coordinate of the nut from the bridge. Then this value is multiplied by -0.15, added to the bridge x, all while adding 10 pixels as well.

$$x_1 = -0.15 \times (x_{nut} - x_{bridge}) + x_{bridge} + 10 \tag{4.6}$$

- Picking hand detected: the only difference in regards to the previous case, is the first part. Instead computing the width and extend by 15%, we select the highest x coordinate from the picking hand joints.

$$x_1 = max(\{x^{(1)}, x^{(2)}, ..., x^{(i)}\}) + x_{bridge} + 10 \tag{4.7}$$

  The equation above represents the process for this case. Also, the ensemble $x^{(1)}, x^{(2)}, ..., x^{(i)}$ represents all the x coordinates of the detected joints of the picking hand.

It is good to note that the 10 pixels added at the end in both cases, is a small value just to avoid having any hand-related optical flow after cropping.

The x coordinate at which the cropping ends, is simply the x_{nut}. Next, the width w and height h are computed by simply subtracting the two x and y coordinates respectively. To keep the cropped optical flow in the same scale as the cropped fretboard image, the image is resized to a new size based on:

$$\begin{bmatrix} w' \\ h' \end{bmatrix} = \vec{S} \odot \begin{bmatrix} w \\ h \end{bmatrix} \tag{4.8}$$

The elements w' and h' are the new width and height of the cropped optical flow image respectively. After obtaining them, the image is resized to have a shape of (w', h').

4.5.4 Hand Pixels Detection

In the cropped fretboard image as the one shown in Figure 4.14, we need to remove the hand pixels as they may affect the string and fret detection later. To do so we adopted the same thresholds as the ones in [8], originally taken from [39]. These thresholds are the following:

$$\begin{aligned} R &\geq 95 \\ G &\geq 40 \\ B &\geq 20 \\ R - G &> 15 \end{aligned} \tag{4.9}$$

The R, G and B represent the three channels of a pixel in the RGB color space; red, green and blue.

All pixels satisfying these conditions are considered as hand pixels, and all the rest are non-hand pixels. By identifying hand pixels we can later remove them and keep only the other pixels, so the hand does not interfere or affect the strings and frets detection steps.

4.5.5 Strings Detection

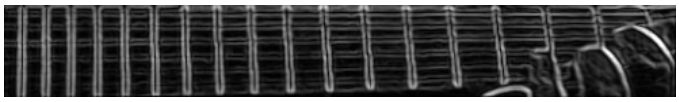
After finishing the last four steps, we can proceed to detect the guitar strings. Indeed, the detection of the strings is an important and essential step for the transcription since its helps in telling us the string being played or pressed. The results of this step can be used to compare detected lines in the optical flow to recognize the picked string; or, it can be used to be compared with any pressing joint in order to find the closest string to that joint, thus the string the joint is pressing.

This step not only is important, but it should also be accurate, since wrong results can make our model predict that a joint is pressing another string than the actual one due to poor results in this step.

The approach adopted for the strings detection is inspired by the fretboard localization and refining (Sections 3.2.1 and 3.2.2) in [4]. First, we transform the cropped fretboard image (as the one shown in Figure 4.14) into a grayscale image. Then, we convolve the image with two Sobel filters (horizontal and vertical) with a kernel size of 3. The two results obtained from these two filters, are then added and equally weighted (weight of 0.5 each).

Next, we threshold the image, turning it into a binary image. The threshold is computed as

the mean of a pixel neighborhood (size 5), minus 2. Figures 4.15a and 4.15b in 4.15 show example results of the Sobel filter and adaptive thresholding respectively.



(a) The obtained image after applying both horizontal and vertical Sobel filters and adding their results.

(b) The resulting image after using adaptive thresholding.

Figure 4.15 Images displaying the obtained result of applying the Sobel filter on the cropped fretboard image in Figure 4.15a; while the second image, Figure 4.15b displays the obtained binary image after thresholding the previous image.

After thresholding, a lot of noise would be present in the image, hence the need for morphological operations. To remove the noise, first we perform an opening operation with a kernel of shape 1x5 (1 row, 5 columns; in other words a line of length 5 pixels). After that, a closure morphological operation is performed with a 1x7 kernel to remove holes if there are any.

The subfigures 4.16a and 4.16b in Figure 4.16 display an example results of the opening and closure respectively.

(a) The resulting image after performing a morphological opening on the binary image.

(b) The resulting image after performing a morphological closure on the image shown in Figure 4.16a.

Figure 4.16 The results obtained after performing the morphological opening and closure respectively. The result in Figure 4.16a is when applied on the binary image shown in Figure 4.15b. The image in Figure 4.16b is the result of the closure operation applied on the result obtained after opening.

Next, we use the Probabilistic Hough Lines transform [40] to detect horizontal lines that are supposed to be the strings. With a ρ equal to 1 and angle intervals of $\frac{\pi}{180}$, a vote threshold and a minimum length of $0.2 \times width$ (width of the cropped fretboard; in other words, the width of Figure 4.14) while having a maximum line gap of $0.1 \times width$.

While going over each line returned by the Probabilistic Hough Transform [40], and as we compute lines, if a line orientation is horizontal, we compute $f(x_c)$ where f is the line equation and x_c is the width of the cropped fretboard divided by 2 (central x). Once the value of $f(x_c)$ is retrieved, it is appended to a data array that is later used for clustering.

After this array is filled, we use K-Means clustering algorithm to cluster lines detected and come out with one single line for each cluster. The number of clusters used is 6, since we have 6 strings. Next, for each cluster we need to find the line that minimizes the error to all other lines within the cluster; this is done through linear regression.

After clustering, for each cluster we collect its corresponding lines and use each line detected points to separate the x and y coordinates. The former are used as training data for the linear regression while the latter serve as targets.

Once all linear regressions are found (in this case 6), we sort those lines by their biases in a reverse order. The higher the bias, the higher the string; meaning, the line with the highest bias will be the high E string, E_4 string or the first string.

The figure 4.17 displays a result of strings detection, with detected strings being color-coded.

Figure 4.17 Image showing the detected fretboard strings with color-coding. The color codes are as follows: red for the first string, green for the second string, blue for the third string, cyan for the fourth string, magenta for the fifth string and yellow for the sixth string.

4.5.6 Frets Detection

After detecting the strings, we need to detect the frets so we can know on which fret a joint is positioned, also inspired by the work in [4]. To do so, we first start by taking the gradient image obtained in the previous step (Figure 4.15).

Next, we apply morphological opening with a kernel of size 11×1, before proceeding with a Probabilistic Hough transform [40] for lines detection (ρ 1, angle $\frac{\pi}{180}$, minimum votes $0.15 \times height$, minimum length $0.25 \times height$ and maximum line gap $0.05 \times height$).

After line detection, we only use the vertical lines (having an angle between 85 and 90 degrees) for clustering. This time a line center x coordinate is used for clustering. The clustering algorithm used is the Mean Shift algorithm with a bandwidth of 5. Once all clusters are found, we find the line that minimizes the errors to all point of a certain cluster through linear regression. These linear regressions are the detected frets (these linear regressions are vertical, in the form of $x = ay + b$).

4.5.7 Frets Refinement

To refine the detected frets, we start from the bridge end, meaning that we start from the last two frets, or the two closest frets within the detected frets.

Inspired by [4], first, we compute the x coordinate at $y = \frac{height}{2}$ for every detected fret and then these values are used for the refinement. We start with an accepted error margin of 5 pixels (meaning if a fret is within 5 pixels of the expected position then we accept it, otherwise it will be replaced). Since we are starting from the last fret, this means we can go up until the width of the cropped fretboard image (Figure 4.14).

To start the refinement, we select the two previous frets, compute the expected x coordinates (x_{i+1}) using the following formula:

$$x_{i+1} = \frac{x_i - x_{i-1}}{0.94387} + x_i \tag{4.10}$$

Equation 4.10 is a rearranged version of Equation 2 in [4]. Next, x_{i+1} is compared with the corresponding detect fret (x_{det}), if the difference is greater than the accepted error margin, then two scenarios exist:

- $x_{i+1} > x_{det}$: This means it was a wrong detection and it is too close to the previous detected fret, thus the detected fret is removed.
- Else: This means that the difference is too big, thus there is a missed detection. It means that x_{det} is far greater than what we expect thus a fret (or frets) is missing. In this case, a fret is inserted with a slope equal to the mean of the previous two frets, and a bias equal to x_{i+1}.

It should also be noted that the accepted error margin is multiplied by $\frac{1}{0.94387}$ at every iteration, however, when $x_{i+1} > x_{det}$ then it is multiplied by 0.94387 to take it one step back.

4.5.8 Picking Detection

After detecting the strings and frets, we need to process the output of the optical flow to know which string was picked. This is important so only the transcription would happen on the strings that were picked. To do this, a series of steps need to be done on the cropped optical flow image, in summary we need to find a line in the optical flow and compare that line to the detected strings to know which strings vibrated.

To check for played strings, from the fretting hand we select the x and y coordinates of the joints pertaining to the index, middle, ring or pinky finger. Then from those joints whose coordinates are within the image, we select the smallest x and subtract 10 from it to avoid having optical flow of the hand later, we will call this x_2. To select x_1, we do the following:

$$x_1 = \begin{cases} 0 & x_2 \leq 100 \\ width_{cropped\_flow\_image} - width_{cropped\_fretboard\_image} & \text{else} \end{cases} \tag{4.11}$$

After getting those two values, the cropped flow image is cropped again, to consider only the parts included inside x_1 and x_2.

Considering the new width of the flow as w ($w = x_2 - x_1$), we first create a kernel for the computation of the second derivative (Laplacian) of the image vertically. This kernel is shown in Equation 4.12

$$K = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \tag{4.12}$$

The image obtained from the previous step is padded with zeros and then convolved with the kernel K. The reason for computing the horizontal laplacian of the image is for the fact that if we take a horizontal cut of the fretboard, the intensities would form a gaussian where the peak is located at the place as the string. A gaussian's second derivative minimum is located at the same location as its peak.

Since the picked string will have negative values in its position when computing the second derivative, we change the pixel intensities of the image based on Equation 4.13.

$$f(p_{i,j}) = \begin{cases} p_{i,j} & p_{i,j} \geq 0 \\ 256 - p_{i,j} + min - 1 & \text{else} \end{cases} \tag{4.13}$$

In the equation above, $p_{i,j}$ is the intensity of the pixel at position (i, j) and min is the minimum value within the image. Doing so would help us have two groups of value, some values in the lower end of the 0 to 255 range, while the negative values obtained after the second derivative would be higher in the higher of that range. The reason for this, is that whenever a string is picked the pixels intensities in a column would look like a Gaussian thus its second derivative would have a negative dip when the Gaussian is going up or down around its peak. By separating the values like that, we would be transforming the image into grayscale as well as helping the thresholding method Otsu to better select a threshold.

Thus, the next step is to binarize the image by using the Otsu method, then we perform a morphological closure with a 3×3 kernel before opening the image with a $3 \times \frac{w}{3}$ kernel (w stands for the cropped optical flow image width).

The next step consists of performing line detection using the Probabilistic Hough Transform [40] (with ρ 1, angle $\frac{\pi}{180}$, minimum votes $\frac{width}{3}$, minimum length $\frac{width}{3}$ and maximum line gap $\frac{width}{10}$). Only horizontal lines are kept, then we fit a linear regression to the ends of those lines (the coordinates are expressed in the coordinate system of the cropped fretboard image, Figure 4.14), and compare the linear regression's bias to all the strings when $x = 0$. The closest string in terms of the Euclidean distance is considered to be the picked string.

4.5.9 Tabs Computation

The last step consists of computing the tabs based on the information computed in the previous steps. However, this step is executed only when there is at least one string that was detected as vibrating or was picked.

In case of picking, we first start by removing unwanted joints of the fretting hand. The wrist and all thumb joints are removed since the former cannot be used for pressing and the latter will be behind the fretboard, hence no need to process them. For the remaining 16 joints, a joint is removed only when its coordinates do not fall within the image. After rotating, translating and scaling the joints coordinates, and express them in the cropped fretboard image coordinate system, a joint is kept if it satisfies the following conditions $0 \leq x < width$, $0 \leq y < height$.

Next, all the vibrating strings will have their own tabs with the fret number being 0[6], then using the criteria explained in Section 4.5.4, we remove the pixels pertaining to the hand from the depth map, since they are not needed.

Following that, we have to check if a joint is pressing. To do so, we take a patch of $height \times 50$

[6] A vibrating string means there is a tab to transcribe.

pixels around the joint under processing and multiply by Gaussian centered at the joint under processing. This Gaussian is centered at the joint under processing to give more weight to the depth values closer to that joint. After that all positive values of depth while the values that are farther than 10 centimeters from the median in both directions.

Finally to decide whether that joint is pressing, we compare its depth with the mean of the Gaussian weighted patch, is it is less or equal to 2 centimeters, then we consider it as pressing. If a joint is pressing, then we find the two fret bars which it lies in between to know the fret number and find the nearest string to it. The transcribed joints are only the ones who lie on a vibrating string.

The former is done by comparing the x values of the fret bars and the joint's; while the latter, is done by using the following point to line distance formula.

$$d = \frac{|a \times x + y + c|}{\sqrt{a^2 + 1}} \tag{4.14}$$

x and y are the joint's coordinates, a and c are the negative values of slope and bias of the line (or detected string) respectively.

4.5.10 Tabs Grouping

Once a note is played on the guitar we may have multiple detections of it, some would have a different string or fret sometimes. We cannot know for sure how many strings were picked in reality, since in a series of frames many strings can be detected and a guitarist may play a chord which involves many strings. However, what we know is that one fret can be played per string; in the case pressing many frets, the highest fret is the fret considered as pressed. While we know this as a fact, however, our approach may have incorrect detections hence we adopt a voting strategy.

Since we play one note/chord per second in our recorded videos (as explained in Section 5.1.3), we group tabs every 1 second. Among the picked strings, the fret with the highest votes is selected for that string, alongside the finger with the highest votes for that fret.

Once the previous is done, we go on to refine the tabs. This step is needed since for a barre chord not all strings may be detected for example. After taking all the grouped tabs from the previous step, all the tabs who have the same fret and finger while have different strings, all strings in between would have the same fret except if there was already a detection on one of those strings with a higher fret.

For example, if we had the tabs shown in Figure 4.18, if the tab on string 1 has the same

finger as the one on string 6, then we consider them as a barre.

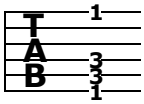


Figure 4.18 Example of tabs that may be detected by our approach with some values missing.

By considering this as a barre, this means that strings 2 and 3 should have a fret equal to 1. Same applies if these strings had any fret number less than 1 (0 or open string for example). The tabs become like shown in Figure 4.19 below.

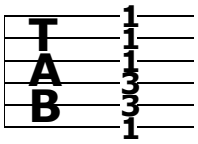


Figure 4.19 Example of tabs after refinement.

CHAPTER 5 EXPERIMENTS, RESULTS, DISCUSSION AND FUTURE WORK

This section presents the experiments and their results; in addition to discussing the obtained outcome.

5.1 Data Collection

It would not be possible to evaluate our different parts without data collection first. Data had to be collected for the segmentation, hand pose estimation, optical flow and the entire system evaluation.

5.1.1 Dataset for fretboard segmentation

To build our own dataset, we collected some data consisting of videos of people playing the guitar. Our approach for data collection was to build an online Google Form where volunteers can submit videos of themselves playing the guitar. For these videos, tablatures were not required. Also, the level of playing was completely irrelevant to our needs since it has nothing to do with the fretboard segmentation.

After approval of our research protocol by Polytechnique's ethics review board, 40 videos were collected through our online form, 34 of those videos were selected to construct the training set while 6 videos were used to build the test set. From each video, 25 frames were sampled and added to the corresponding set (training or test). Also, 61 images were chosen from Pixabay to add to our dataset. This gives 1061 images in total for our dataset, 911 of those images belong to the training set while 150 images are in the test set.

This dataset[1] is very diverse as well, different types of guitars exist within our dataset; classical, acoustic and electrical guitars. In addition to the different guitar types, different shapes exist within the same type, especially with electric guitars. Also, the images used were taken in different environments/places with different people making the dataset more diverse as well.

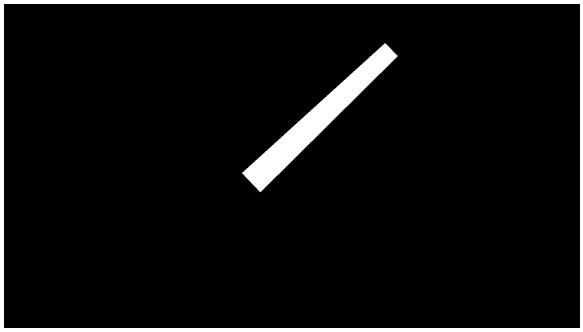
After building the dataset, the images had to be annotated. The ground-truth would be a binary mask where the white pixels represent the fretboard while the rest being black. This was done manually. In addition, the faces of the people shown in the images were anonymized

[1] Our dataset can be found at: https://bit.ly/3zgpYY9

through blurring to make sure they were unidentifiable.

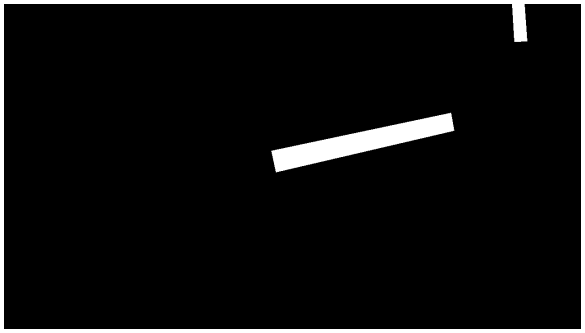
Figure 5.1 shows three examples of images in our dataset alongside their respective masks.

(a) Image from the training set showing a person playing a guitar.



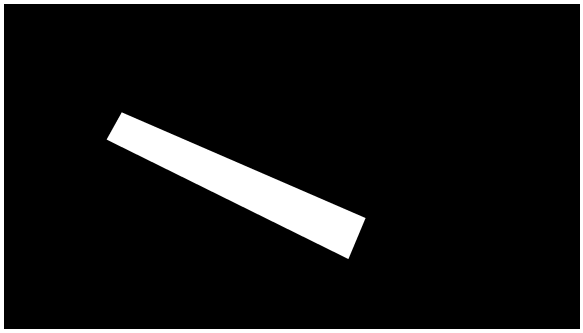
(b) The corresponding mask for the Figure 5.1a

(c) Image from the training set showing two guitars.



(d) The corresponding mask for Figure 5.1c with two guitars present.

(e) Image showing a left-handed guitar player from the test set.



(f) The corresponding mask for the left-handed guitar in Figure 5.1e

Figure 5.1 Images from our dataset for fretboard segmentation (first column) alongside their corresponding masks (second column).

Another dataset exists that could be used for fretboard segmentation, a guitar transcription dataset on Kaggle [1]. Despite running our segmentation training on that dataset as well, we

used the model trained on our dataset in the overall transcription approach since our dataset has more variance and more images.

5.1.2 Dataset for the Evaluation of Hand Pose Estimation

To validate the depth coordinates provided by MediaPipe for the hand pose estimation, 10 pairs of RGB and depth images were recorded. The RGB can be used with MediaPipe to predict the 3D coordinates of the different 21 joints; while the depth image can be used to check whether the predicted z coordinates are accurate or not.

5.1.3 Dataset for Transcription

To evaluate our entire approach, videos of people playing the guitar had to be collected. The musical pieces to be used for evaluation had to be defined first. We decided to use the same 3 musical pieces used in [4] alongside 6 others (see Appendix A). Using 3 different guitars (1 classical and 2 electric) with two different players, these different pieces test different scenarios for the system and thus would help us better assess our system in each different case.

To record these videos, the Intel RealSense Viewer recording feature was used with RGB and depth modalities having a resolution 1280×720 with 30 fps. Each note within any piece used for the recording was played every second, in other words, only one note (or one chord) was played every second. This was done due to the fact that in our approach we do not take timing and notes duration into consideration, since our main focus is detecting the correct tabs.

In addition, ground-truth text files for the tabs were created, since there is no need for a specialized file type or software. The tablature written in those files were separated by spaces, where each column with hyphens and numbers represents what should be detected. The first column represents the tabs played during first second, the second column represents the tabs played during the second second, and so on.

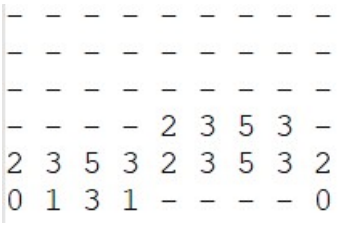


Figure 5.2 A ground-truth example of the "5th 4th" piece. The equivalent is shown in Figure A.3.

The spaces introduced in the ground-truth annotation (such as in Figure 5.2) helps us separate seconds, and that can be useful in cases where we have a two-digits number like 10 for example. This is illustrated in the figure 5.3.

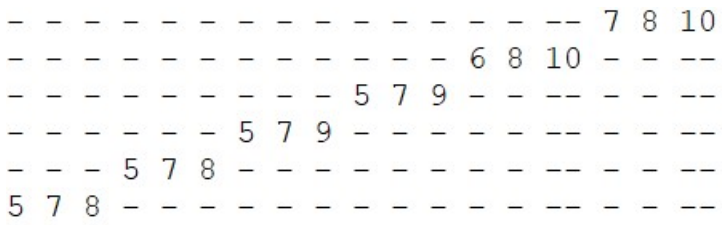


Figure 5.3 A ground-truth example of the "A minor 3 per string" piece. The equivalent is shown in Figure A.4.

5.2 Experiments

5.2.1 Segmentation

To assess how Mask R-CNN performs on our fretboard segmentation dataset, training had to be completed, then the model had to be tested on our test set. To speed up the training of Mask R-CNN, the training was run on Digital Research Alliance of Canada HPC. We used a NVIDIA V100-SXM2 GPU[2] with 32GB of memory[2]. We used 1 CPU core for the training, with a 32 GB of memory and a maximum time of two days.

The model is first deployed on the GPU to achieve maximum performance. Then, the data is loaded, split into five folds, four of those will be used for training and one for validation. Although K-Fold was implemented, we only trained the model for one fold.

In addition, the code used for training tunes the model using SigOpt[3] python library. By

[2]https://docs.alliancecan.ca/wiki/Using_GPUs_with_Slurm
[3]https://sigopt.com/

using SigOpt we can find the best combination of hyperparameters we want to tune; these hyperparameters are:

- Optimizer: three optimizers were tested, Adam, RMSprop and SGD.
- Learning rate: the learning rate could be any value, so instead of having a wide range from 0.000001 to 0.001 for example we used the log learning rate. If the log learning rate is -2 then the learning rate would be 10^{-2} (or 0.01). The range used for this hyperparameter was from -6 to -3, with a step of 1.
- Epochs: The number of epochs is important when training a model to make sure reaching the peak performance of a model. In our case, since Mask R-CNN is pre-trained on COCO2017 train dataset[4], and since we are applying transfer learning, our range of epoch is relatively small. From 5 to 10 inclusively.
- Batch size: since the fretboard segmentaton training set consists of only 911 images, it would be hurtful to use a huge number for the batch sizes since the weights of the model will be rarely updated. So, our batch size range is from 1 to 8 inclusively.

The metrics used for the evaluation of our system are Dice and IoU. These metrics are also known as F1 score and Jaccard score respectively. Suppose that we have a ground-truth mask where the white area pertaining to the fretboard is called A and a predicted mask with the white area called B. The dice score is computed as follows:

$$Dice = \frac{2 \times |A \cap B|}{|A| + |B|} \tag{5.1}$$

As shown in equation 5.1, dice is simply twice the intersection of A and B divided by their union. On the other hand, the IoU metric is as its name indicates (Intersection over Union), the intersection of both regions divided by their union.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{5.2}$$

This evaluation and computation is done after each epoch on the validation set. When predicting masks, detections with a confidence score equal to or higher than 0.9 are taken into account and used to build the mask. Other detections and predictions are dismissed. After finishing the training, the best hyperparameters determined were the following:

- Optimizer: RMSprop.

- Learning rate: 10^{-5} or 0.00001.
- Epochs: 10.
- Batch size: 1.

Quantitative Results

These parameters yielded a Dice score of 0.9231 and an IoU of 0.8571 on the test set. The metrics were computed for each image in the training set and then the average was taken for both Dice and IoU.

Using the same procedure for training Mask R-CNN, we trained the latter on Kaggle's Guitar Transcription Dataset [1].

Since the dataset follows COCO and VGG annotations, a small script was written to generate the ground-truth masks based on those annotations for each image in the training and test set. After that the images were placed in same hierarchy as our dataset before actually launching the training on this dataset.

After training, the best hyperparameters chosen by SigOpt were the following:

- Batch size: 4.
- Epochs: 7.
- Log learning rate: -4 (learning rate 10^{-4} or 0.0001).
- Optimizer: Adam.

The results obtained based on their test set were 0.9252 and 0.8609 for Dice and IoU respectively.

To compare our dataset to the one found in [1], we ran the model trained on our dataset on the Guitar Transcription Dataset and vice-versa.

Table 5.1 Dice and IoU scores reported on both dataset and the Guitar Transcription Dataset [1] with Mask R-CNN trained on either datasets.

	Dataset Tested On	
Dataset Trained On	Ours	Guitar Transcription Dataset [1]
Ours	Dice: 0.9231, IoU: 0.8571	Dice: 0.9634, IoU: 0.9294
Guitar Transcription Dataset [1]	Dice: 0.6841, IoU: 0.5199	Dice: 0.9252, IoU: 0.8609

Qualitative Results

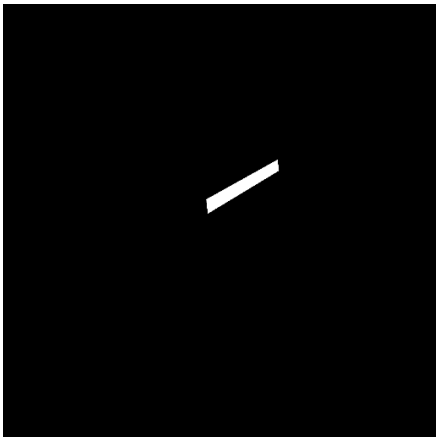
Qualitative results will be presented, showing successful as well as failure cases or scenarios. The images shown in this section are images taken from our fretboard segmentation test set alongside their ground-truth masks, and the prediction made by Mask R-CNN after training.

Success Cases In this section, cases of correct and good predictions from Mask R-CNN on the test set will be displayed. The images and ground-truth masks are from the test set, they are presented in Figures 5.4, 5.5 and 5.6.

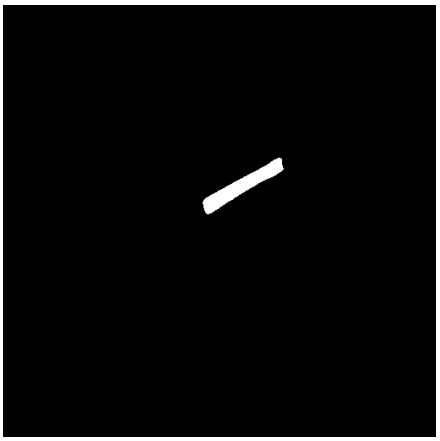
As shown, the model predictions are close to the ground-truth. Figure 5.5e displays an overlay of the predicted mask shown in Figure 5.5c, and it clearly shows us that there are some differences around the corners of the fretboard compared to the ground-truth (shown in Figures 5.5b and 5.5d), as they are round in the prediction.

As for Figures 5.4 and 5.6 the differences are even less significant relative to Figure 5.5.

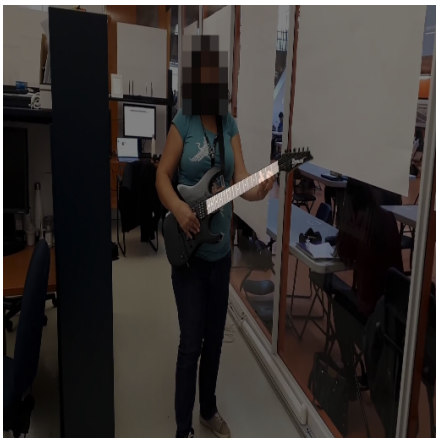
(a) Image from the test set used during Mask R-CNN evaluation.



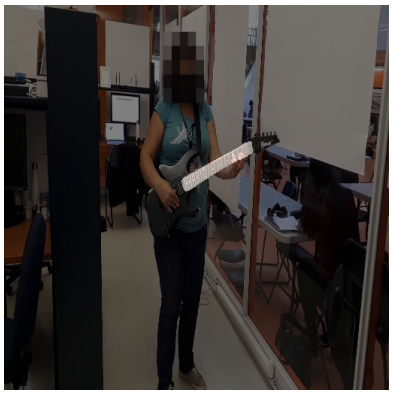
(b) The corresponding ground-truth mask for the Figure 5.4a



(c) The mask predicted by Mask R-CNN for the image shown in 5.4a



(d) Overlay of the ground-truth mask (Figure 5.4b) with the RGB image.

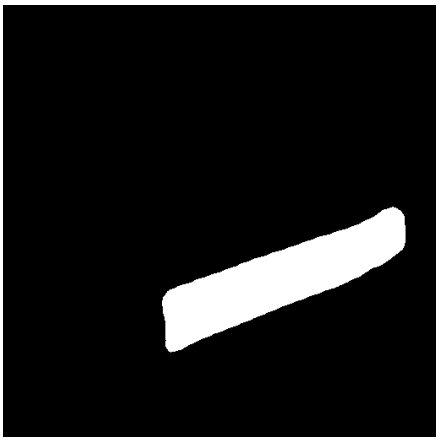


(e) Overlay of the predicted mask (Figure 5.4c) with the RGB image.

Figure 5.4 Figure showing an image from our test, alongside their ground-truth mask, predicted mask , overlay with the ground-truth, and overlay with the corresponding prediction. The image chosen demonstrates the ability of our model to predict even when the fretboard is relatively small and at a distance from the camera (Dice: 0.8545, IoU: 0.7460).

(a) Image from the test set used during Mask R-CNN evaluation.

(b) The corresponding ground-truth mask for the Figure 5.5a

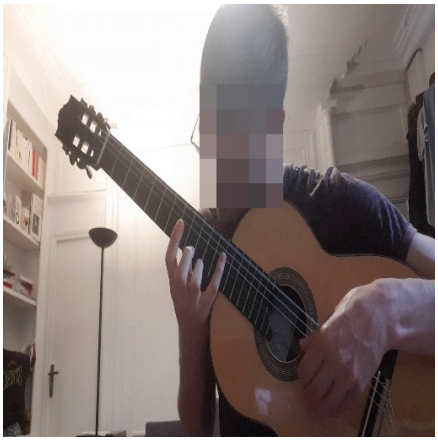


(c) The mask predicted by Mask R-CNN for the image shown in Figure 5.5a

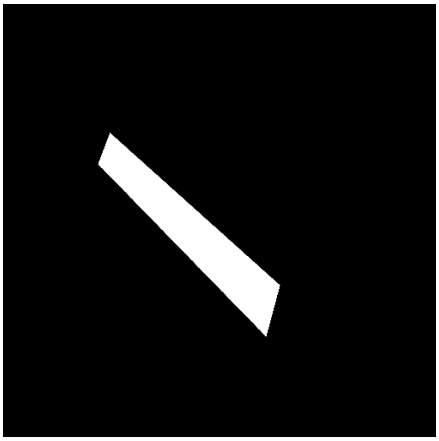
(d) Overlay of the ground-truth mask (Figure 5.5b) with the RGB image.

(e) Overlay of the predicted mask (Figure 5.5c) with the RGB image.

Figure 5.5 Figure showing an image from our test, alongside their ground-truth mask, predicted mask , overlay with the ground-truth, and overlay with the corresponding prediction (Dice: 0.9695, IoU: 0.9408).

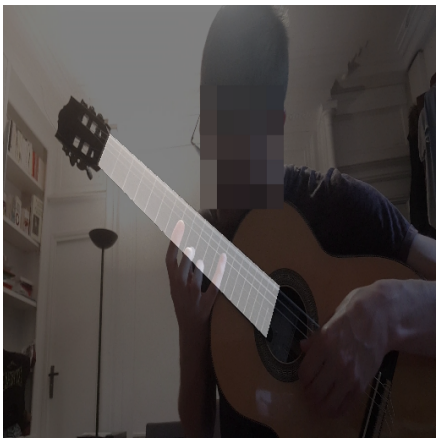


(a) Image from the test set used during Mask R-CNN evaluation.

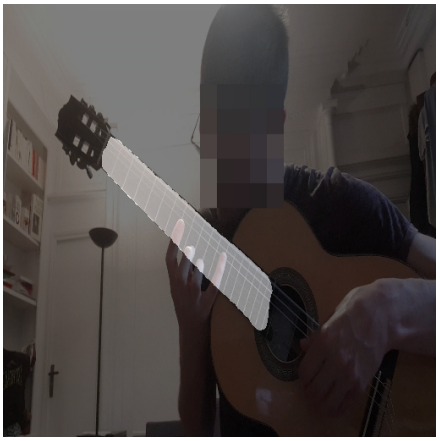


(b) The corresponding ground-truth mask for the Figure 5.6a

(c) The mask predicted by Mask R-CNN for the image shown in Figure 5.6a



(d) Overlay of the ground-truth mask (Figure 5.6b) with the RGB image.



(e) Overlay of the predicted mask (Figure 5.6c) with the RGB image.

Figure 5.6 Figure showing an image from our test, alongside their ground-truth mask, predicted mask , overlay with the ground-truth, and overlay with the corresponding prediction. The image chosen demonstrates the ability of our model to work with classic guitar which happens to be also left-handed (Dice: 0.9635, IoU: 0.9295).

Failure Cases In this section, the scenarios where Mask R-CNN was unable to predict the fretboard correctly will be displayed. In some cases, extra parts (around the fretboard) were incorrectly classified as fretboard, sometimes it is the other case; only a part of the fretboard is detected. Other cases contained both or just the inability to detect the fretboard at all.

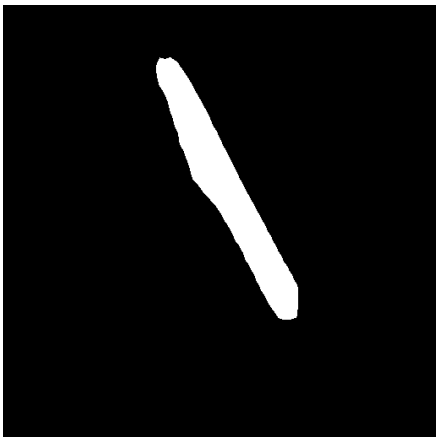
For Figure 5.7 the overlay sub-figure 5.7e show that the model predicted the fretboard incorrectly. A shift or translation exists, the model was unable to delimit the fretboard correctly.

Another incorrect delimitation of the fretboard exists in Figure 5.8, however, this time the model was able to stop at the bridge end of the fretboard unlike the results shown in Figure 5.7.

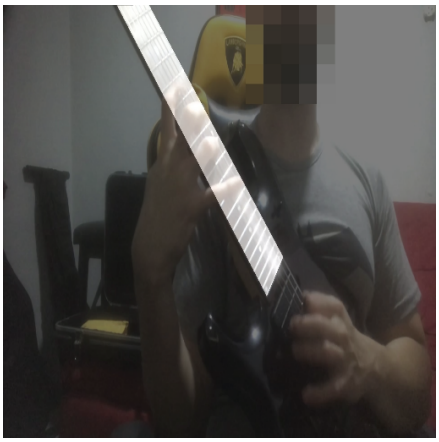
Additionally, Figure 5.9 illustrates the inability of the model to detect the fretboard despite it being there. This is probably caused by the low lighting condition.

(a) Image from the test set used during Mask R-CNN evaluation.

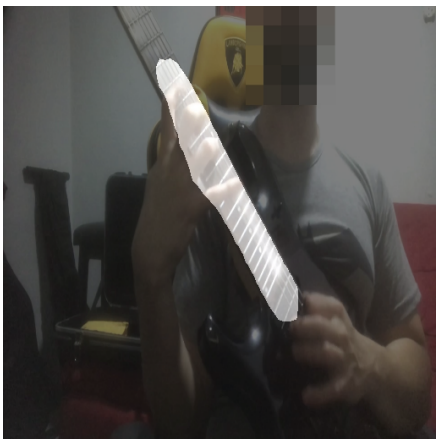
(b) The corresponding ground-truth mask for the Figure 5.7a



(c) The mask predicted by Mask R-CNN for the image shown in 5.7a

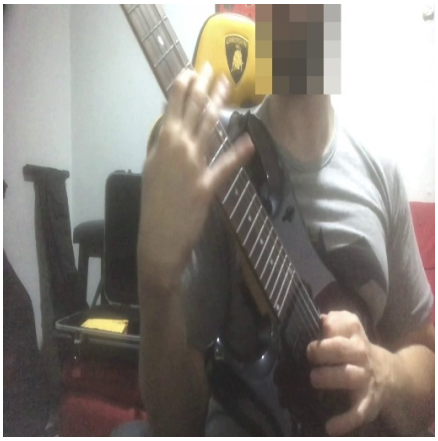


(d) Overlay of the ground-truth mask (Figure 5.7b) with the RGB image.



(e) Overlay of the predicted mask (Figure 5.7c) with the RGB image.

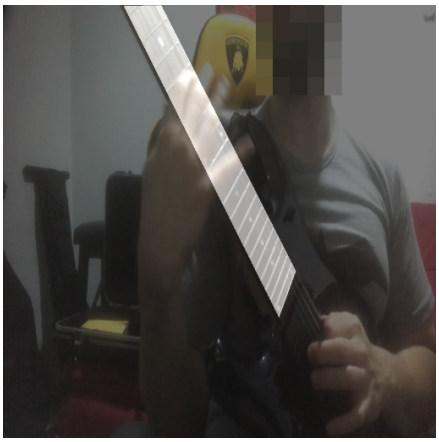
Figure 5.7 Figure showing an image from our test set, alongside its ground-truth mask, predicted mask , overlay with the ground-truth, and overlay with the corresponding prediction. This figure shows that the model classified some incorrect pixels of the fretboard (Dice: 0.7621, IoU: 0.6157).



(a) Image from the test set used during Mask R-CNN evaluation.

(b) The corresponding ground-truth mask for the Figure 5.8a

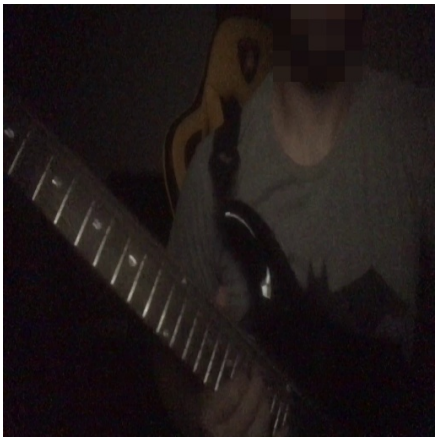
(c) The mask predicted by Mask R-CNN for the image shown in 5.8a



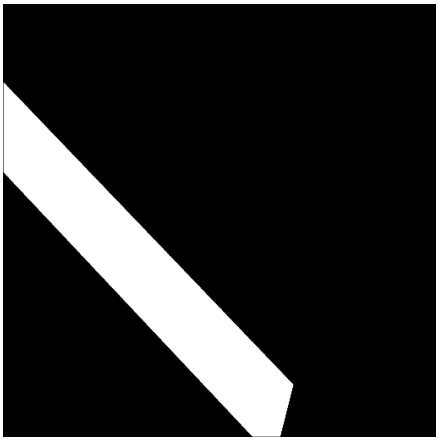
(d) Overlay of the ground-truth mask (Figure 5.8b) with the RGB image.

(e) Overlay of the predicted mask (Figure 5.8c) with the RGB image.

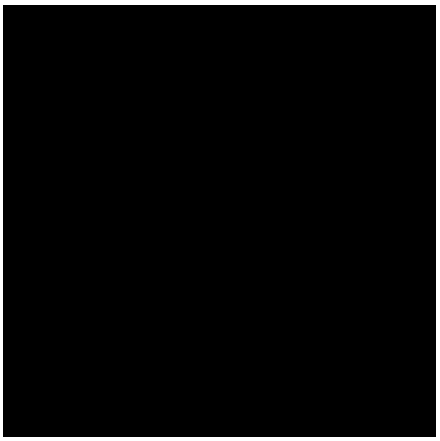
Figure 5.8 Figure showing an image from our test set, alongside its ground-truth mask, predicted mask , overlay with the ground-truth, and overlay with the corresponding prediction. This figure highlight the inability of our model to detect the fretboard entirely (Dice: 0.6978, IoU: 0.5359).



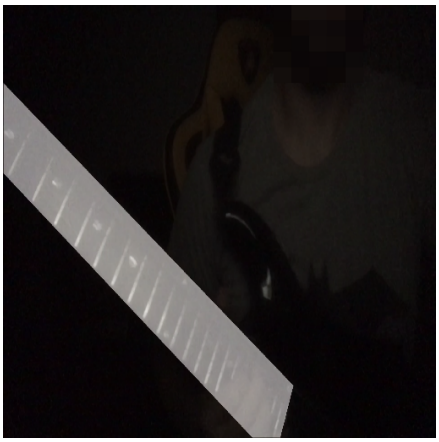
(a) Image from the test set used during Mask R-CNN evaluation.



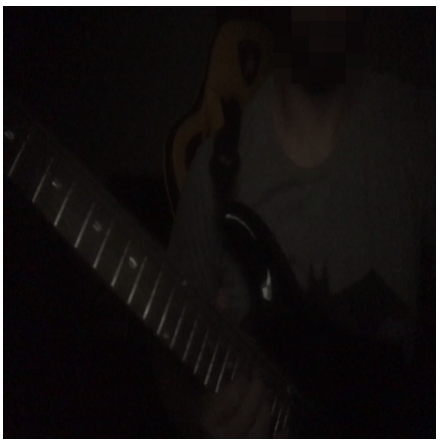
(b) The corresponding ground-truth mask for the Figure 5.9a



(c) The mask predicted by Mask R-CNN for the image shown in 5.9a



(d) Overlay of the ground-truth mask (Figure 5.9b) with the RGB image.



(e) Overlay of the predicted mask (Figure 5.9c) with the RGB image.

Figure 5.9 Figure showing an image from our test set, alongside its ground-truth mask, predicted mask , overlay with the ground-truth, and overlay with the corresponding prediction. This figure highlight the inability of our model to detect the fretboard in low light conditions (Dice: 0, IoU: 0).

5.2.2 Discussion

As indicated by our quantitative results and qualitative results shown in Figures 5.4, 5.5 and 5.6, Mask R-CNN was capable to predict fretboard location and related pixels in a relatively accurate manner. However, Figures 5.7, 5.8 and 5.9 show that the model can still make mistakes, false predictions or no predictions as it is the case in Figure 5.9c.

Based on the shown results, there are several points that can be improved in the segmentation, which are the following:

- Size of the dataset: to train a model well enough in a supervised manner (just like it was done with Mask R-CNN in this project), large quantities of data are required. Despite the fact that a pretrained model was used and then finetuned on our dataset, the dataset may still be small even for transfer learning.
- Hyperparameters: the range of hyperparameters can also be a factor. Since not all hyperparameter combinations were tested, new untested parameters could yield better results than the already obtained ones. By using SigOpt we tested 40 combinations of hyperparameters as suggested by SigOpt algorithm, however, it could deem useful to expand the range of some parameters (going up to 50 epochs instead of 10, for example), in addition to adding new ones (such as whether to add a learning rate scheduler or no, for instance).
- Architecture: although Mask R-CNN works well, other architectures could also be more useful to our use case. The different architectures could be variations of Mask R-CNN (changing the backbone, or slight modifications elsewhere within the model).

Despite the aforementioned points, the model is precise enough to be used for fretboard segmentation. Small errors can be tolerated, in addition, the testing videos for the end to end testing will be mainly videos where the fretboard is almost parallel to the camera and with good enough lighting conditions. The other cases were added to the dataset solely to improve the model robustness, however, in our guitar playing videos we do not have such cases. As shown in Figures 5.4, 5.5 and 5.6, the model is precise enough.

Finally, 5.1 proves the usefulness of our dataset. When training Mask R-CNN on our dataset, the model performed decently when tested on either datasets, even performing the model trained on the other dataset when tested on its test set. However, when training on the Guitar Transcription Dataset [1], the model performed well on its test set; but that was not the case when tested on our test set. This comparison proves that our dataset helps models

generalize better and avoids overfitting such as what happened with the Guitar Transcription Dataset [1].

5.3 Hand Pose Estimation

The hand pose estimation is an essential task in this project, and hence the need to evaluate it, validate it and understand its behaviour. To do this, experiments were conducted where we assess MediaPipe accuracy in terms of joints depth estimation.

Since both CS1 and CS2 are different from the D455 coordinate system, a mapping should happen. We should be able to go from one coordinate system to another (either from CS1 or CS2, to the D455 depth map coordinate system in terms of depth).

Three different methods for retrieving, in the camera coordinate system, the depth coordinate of all the joints predicted by MediaPipe, were tested and compared to the ground-truth depth (retrieved from the depth map). For each method, we measured the errors in terms of millimeters along the z-axis in the camera coordinate system.

To experiment, for LR and LR2 a training has to be done on an image basis, while for the centroid it is just a simple computation. Hence, learning, outliers removal and predictions are performed on a per-frame basis. Next, we compute the average mean error per joint and per method. For the former, the mean error for each joint is simply the sum of errors across all frames divided by the number of times that joint was not considered as an outlier. As for the latter, it is the weighted mean of the errors; each joint is weighted by the number of times it is used (that it is not an outlier) divided by the sum of the usage of all joints. Also, this experiment was run twice, once with and once without the hole filling filter previously mentioned.

5.3.1 Results

In this subsection, we present the results obtained for each method described above. Table 5.2 shows that the linear regression methods work better than the Centroid method, however the differences are subtle.

Table 5.2 Mean error for each tested method considering a number of scenarios, where the hole filling filter is either included or excluded and when including or excluding the wrist and thumb in the calculation.

Method	Mean over all joints	Mean excluding wrist and thumb
LR	18.62 mm	16.23 mm
LR2	18.99 mm	15.27 mm
Centroid	21.19 mm	17.96 mm

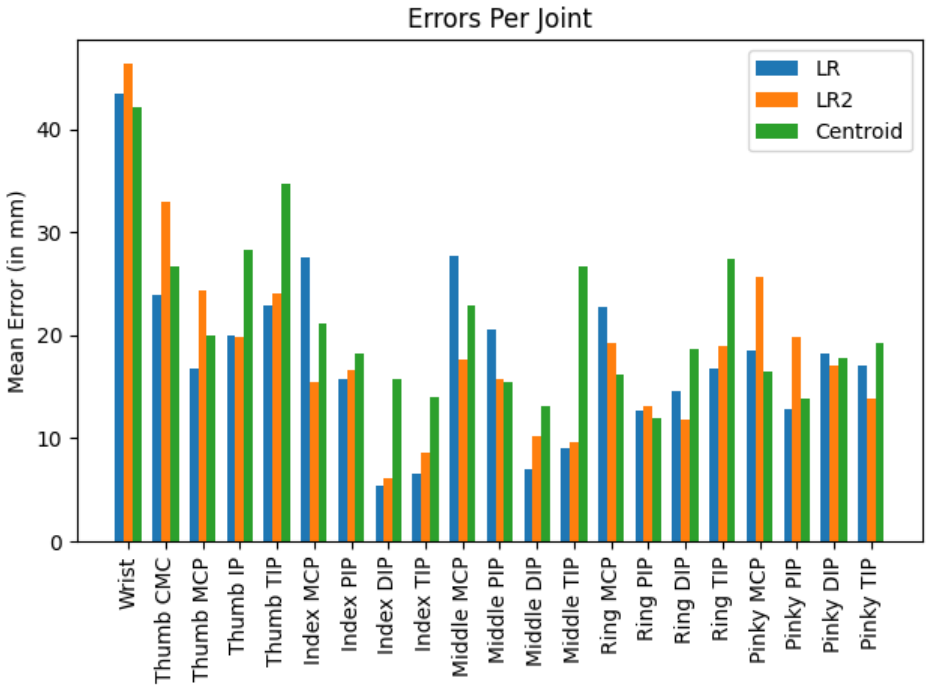


Figure 5.10 Mean error per joint in millimetres.

5.3.2 Discussion

As shown in Table 5.2, the difference between the three tested methods is less then 4 millimeters. One pattern we can notice is that the mean error is reduced when we exclude the wrist and the thumb from our error calculation. The reason for not taking the wrist and thumb joints into account is that when playing the guitar, the player only uses his index, middle, ring or pinky fingers in his/her fretting hand.

Figure 5.10 also clearly show that the mean error for the wrist and thumb joints (wrist, thumb CMC, thumb MCP, thumb IP and thumb TIP) are generally higher than others.

Despite these error drops, the differences do not exceed 5 millimeters in the best case (from Centroid with filter to Centroid without a filter while excluding the thumb and the wrist). The calculated errors are caused by many reasons, these errors can be caused by MediaPipe itself, the camera or even the methods used.

The source of errors could be that MediaPipe itself makes faulty predictions, causing either the linear regression to learn those errors or returning incorrect values that are used in the centroid computation. Also the depth map returns the depth value at the surface of the hand, while MediaPipe could be returning the depth of a joint. Thus, there would be a half a finger thickness error, which could be adding up to the error rate.

5.4 Optical Flow and Picking Detection

As explained earlier, our approach comprises an optical flow computing element whose result is used later for picking detection and matching (in case of detection) the detected line of flow with the corresponding string. Since the picking detection affects whether a transcription would happen or not (no tabs will be transcribed without any detected picking), it is important to evaluate this step and see how well it performs.

This evaluation helps us understand whether there are missing detections or incorrect string associations whenever picking is detected. First, we selected some optical flow outputs from the videos recorded, and set the expected results for these frames. Then, we ran our picking detection method (preceded by fretboard segmentation and strings detection of course) and compared the predicted output to our expectations. For each expected string we selected 9 frames, thus 63 frames were selected (9 for each string, and 9 for no picking represented by "X").

In another way of assessment, we used the same RGBD videos that were recorded for transcription. However, when evaluating we are not interested in providing the correct fret number, we only check whether it is the correct string and compute a "no picking" vs "picking" confusion matrix.

Finally, evaluating picking this way means that we are not only evaluating the picking detection method, by itself; this includes the tabs grouping and refinement steps as well.

5.4.1 Results

Three confusion matrices were computed for the results of this evaluation, the first one being a multi-classes confusion matrix having 7 classes, string 1 to 6 alongside another class

represented by 'X' to signify the absence of picking (no strings played). The second confusion matrix is similar to the previous one, however, this one is based on the output of the videos, and as explained earlier we are not interested in the fret numbers. The third confusion matrix, is a two-class matrix, with two classes: 'no picking' and 'picking'.

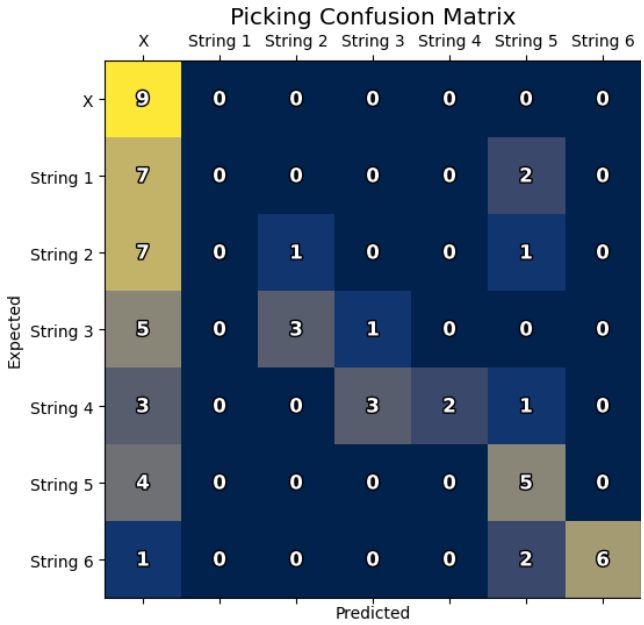


Figure 5.11 Multi-strings confusion matrix for picking detection on selected frames.

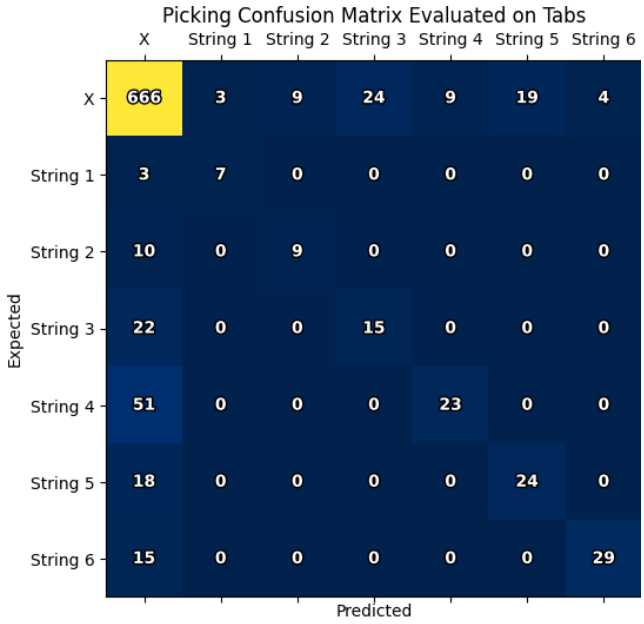


Figure 5.12 Multi-strings confusion matrix for picking detection for the predicted tabs on our recorded videos. This confusion matrix evaluates our picking detection + tabs grouping + tabs refinement.

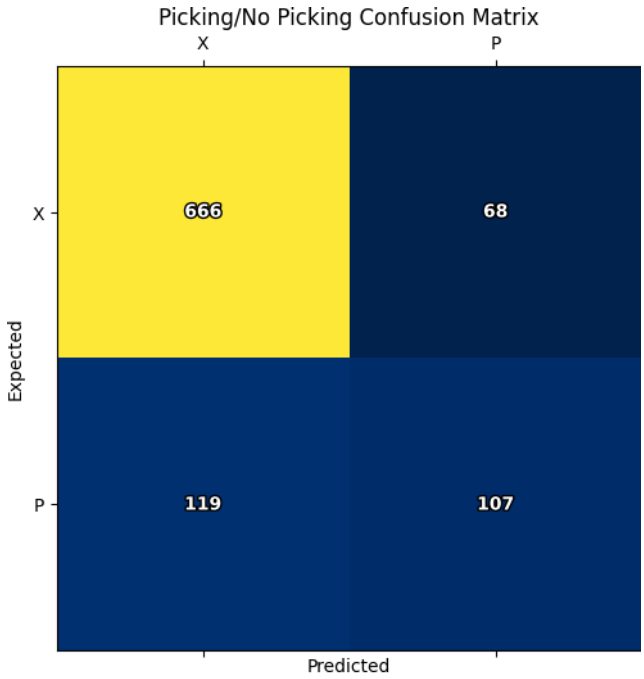


Figure 5.13 Picking/No picking confusion matrix for picking detection.

We also present bar charts that show the picking detection on a musical piece (Figure 5.14)

and guitar type (Figure 5.15) basis.

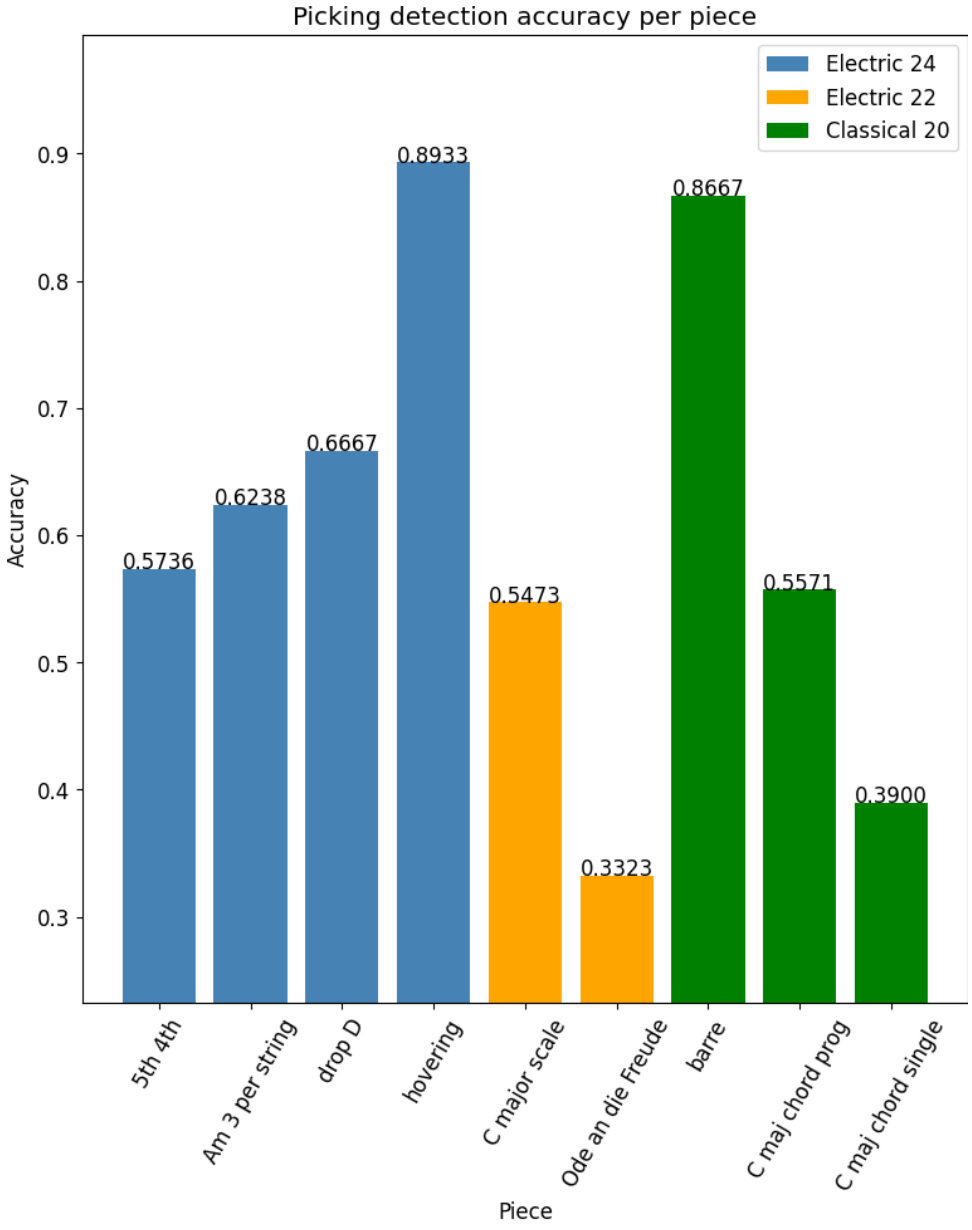


Figure 5.14 Bar chart representing the picking accuracy per musical piece.

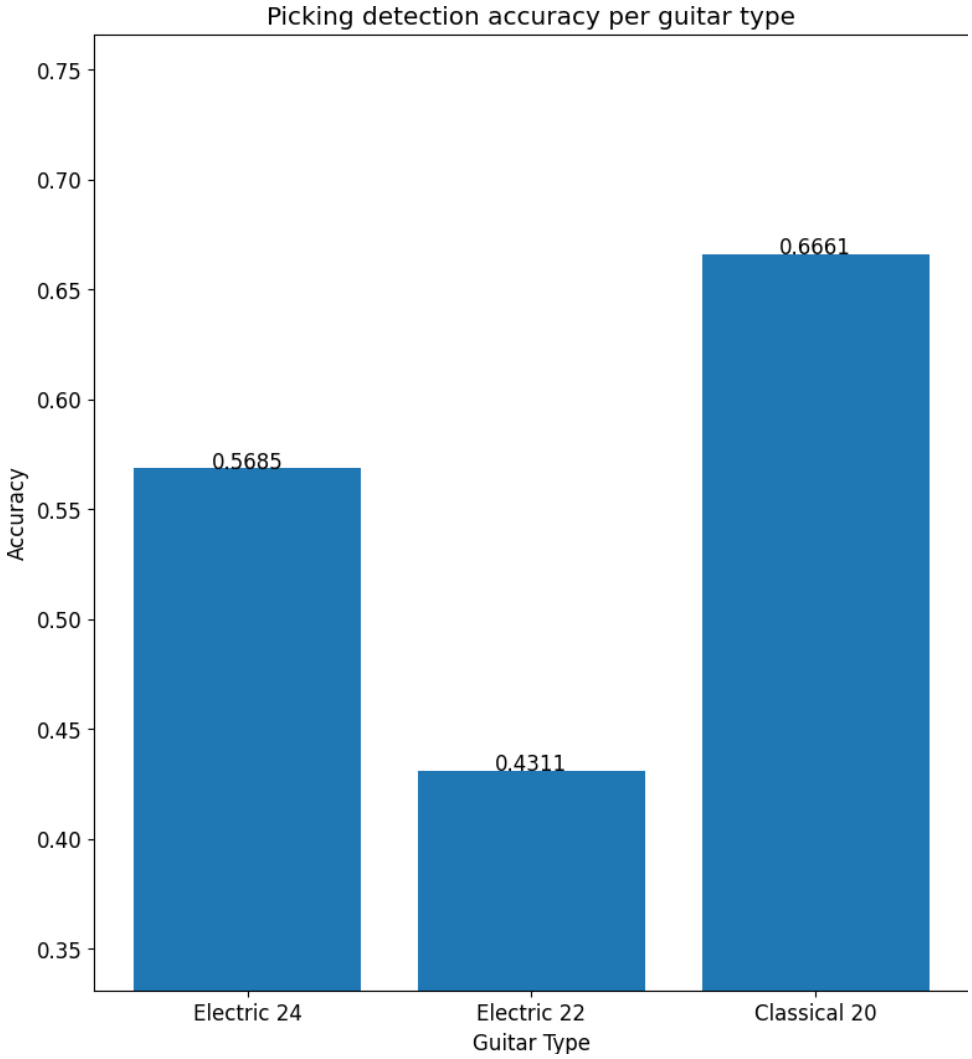


Figure 5.15 Bar chart representing the picking accuracy per guitar type used.

The difference between Figures 5.11 and 5.12 is that the former is an evaluation of the picking detection method that recognizes the played string. The ground-truth was a string number and thus we compare the prediction to the expected index. While the latter, is filled based on the predicted tabs. If the ground-truth consists of string 6 fret 1 and we predicted string 5 fret 2 thus in the confusion matrix we will add that we expected "X" but found string 5 in addition to expecting string 6 and finding "X". The reason we cannot place it in the cell of string 6 (expected) string 5 (predicted) is that for chords we cannot know which string was confused with which other string. Thus, our evaluation method for this part is not optimal.

Finally, we measured the accuracy per string (based on Figure 5.12) as well as the mean accuracy across all of them, and we got the following:

- No string played: 90.74%.
- String 1: 70%.
- String 2: 47.36%.
- String 3: 40.54%.
- String 4: 31.08%.
- String 5: 57.14%.
- String 6: 65.90%.
- Mean: 57.54%.

5.4.2 Discussion

As presented in the previous section, our method performs averagely.

For the confusion matrix shown in 5.11, we can see that the thicker the string, the less errors our method makes. String 6 has the least number of misclassifications (3 out of 9) while that number keeps on increasing with every thinner string (4 out of 9 for string 5, 7 out of 9 for string 4 and so on). Our method also correctly classified all frames where we expect to have no picking as a prediction. The worst result came for string 1; being the thinnest string, it is hard for the optical flow to detect a vibrating line since it is not very visible in the resolution used (1280×720). Another important point to notice is that the thinner the string the bigger the probability for our method to classify it as no picking rather than another string. This is again due to the strings thickness and some discontinuities in the optical flow.

Based on the confusion matrix shown in Figure 5.13 our model presents 80.52% accuracy for picking detection meaning that our method is capable of detecting the presence or absence of picking nearly 81% of the time.

In addition, both Figures 5.12 and 5.13 show that our method has a significant number of false positives and false negatives. The first row in Figure 5.12 displays how false positives were classified, while expecting an absence of picking, showing that the third string had the most false positives. However, looking at false negatives (first column in Figure 5.12), it is obvious that the fourth string is picked the most in our recorded videos, thus it is the one with the most incorrect classifications.

On another note, Figures 5.14 and 5.15 compare the accuracy of picking detection on a musical piece and guitar type level respectively. In the former, two pieces stand out, "hovering" and

the "barre". The first one has a lot of open strings picking on the sixth string (the thickest) on an electric guitar proving that unlike other existing approaches ([8], [4], [6], etc.) , ours is capable of detecting played open strings thus the importance of keeping the picking detection part in future research. While the former mostly consisted of an open 6^{th} string, the "barre" piece consisted of barre chords. The chords used in this piece go through all the strings, meaning that all strings are played per chord. Our method was able to correctly detect pickings almost 87% of the time.

We can also see that the results were particularly bad for "Ode an die Freude" and "C major chord singles". For the former, our picking detection method was unable to detect the picked string despite the good optical flow images. This was mainly due to the fact that some lines in the optical flow had discontinuities, in addition to string misclassifications as well. For the latter, the optical flow failed to detect picking in some frames in addition to incorrect string associations, the inability of correctly identifying pressing joints as well as picking even though a line exists within the optical flow output.

Moreover, the results in Figure 5.14 show that our picking method works almost equally well on both notes and chords-based pieces (see Appendix A). The picking detection on different pieces is around the same values (close to 0.5), however, the two pieces standing out the most are "hovering" and "barre" where the first is note-based and the second is chord-based.

As for guitar types, it is clear that the "Electric 22" which is an electric guitar with 22 frets and the thinnest strings (gauge 8) was the hardest to be precise with. While the classical guitar results were better due to thicker strings while also having nylon strings for the highest 3 strings (instead of metal strings). The other electric guitar (electric 24), was in the middle since it has metal strings but with a thicker gauge (gauge 9) than the aforementioned electric one. Thus, our method does not work equally on different types of guitars, it works better on guitar with thicker strings and with less metal strings.

Finally, considering the accuracy per string; we can notice that our method correctly classifies optical flow images with no picking around 91% of the time. However, looking at the strings, the best performing strings were strings 1 and 6 while the others fluctuated around 50%. Despite the relatively good numbers for strings 1 and 6, our picking detection is still inconsistent for the most part and is a contributor to the incorrect tablature outputted by our approach. We also hypothesize that for string 1 and 6 it could be that an optical flow detecting the movement of the guitar (or fretboard) can confuse our method make it predict vibrating strings close to the extremities, thus strings 1 or 6.

5.5 Transcription System

After testing all the different parts of our approach (segmentation, hand pose estimation, picking detection), the evaluation of the entire approach was conducted. As shown in Figure 4.1, the overall approach comprises the four different parts (segmentation, hand pose estimation, optical flow and post-processing). By evaluating the whole approach, we are assessing the system performance and how the different parts work together.

5.5.1 Results

In this section we present the results obtained for the predicted tabs by our approach when compared to the ground-truth.

Our code was run on an Intel Core i7-10700 CPU, NVIDIA GeForce RTX 3080 GPU with 16 GB of RAM. The system ran for 1 hour 59 minutes and 15.2035 seconds for 4878 frames, and the average time taken per frame was 1.4668 seconds. The segmentation had an average time of 1.005 seconds per frame, while the hand pose estimation, optical flow computation and the post-processing averaged 0.03277, 0.1675 and 0.0727 seconds respectively.

Figure 5.16 shows the confusion matrix for our transcription model. This confusion has 12 classes, 0 for an open string, 1 to 10 for frets and "X" for no tablature. Despite the fact that the guitars used in our experiments have at least 20 frets (classical guitar), we limited our confusion matrix to 12 classes only for visualization purposes and since the highest fret we use in our evaluation is the 10^{th} fret.

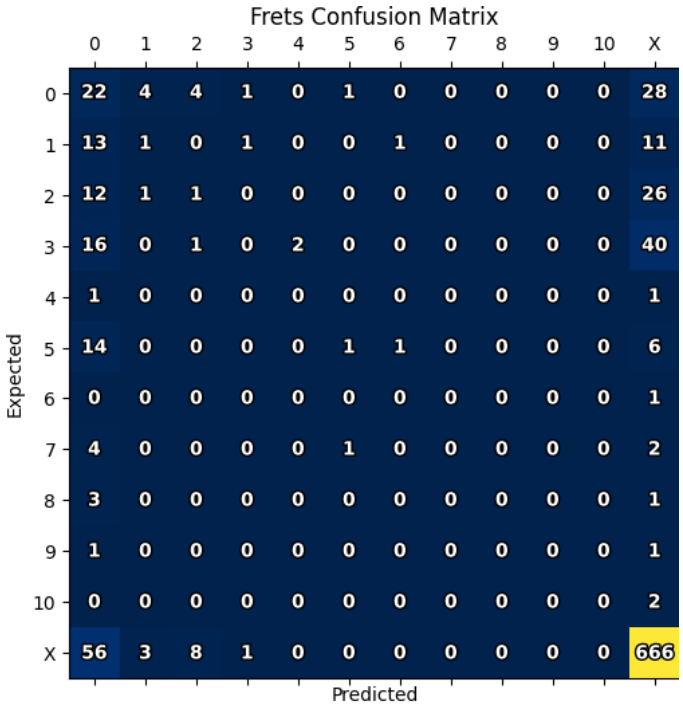


Figure 5.16 Frets or tabs confusion matrix for all the tested videos.

5.5.2 Discussion

While the picking proved to be working quite well, the frets or tabs prediction of our system is not as significant. As shown in Figure 5.16, our method predicts a good number of incorrect fret numbers in the output tablature.

In the last row of the confusion matrix, where we expect to have no tabs, a lot of false positives exist, this is due to the fact that whenever a picking is detected our system is forced to make a tab prediction even if in reality no transcription should happen. Similarly, in the last column, our model cannot predict tabs when our picking method does not detect any picking, thus this confusion matrix last row and last column can be dismissed during evaluation as these mistakes can be attributed to failure in picking detection, however, we kept them in the confusion matrix for completeness.

While excluding the aforementioned row and column, the accuracy of our method is 23.36%. These results are due to the accumulation of errors. Since our method is comprised of many steps that can affect each others results, one error can the following steps to fail. A missed picking detection, can impact the score for the entire system; thus the accuracy for the entire system is expected to be lower than that of picking.

It is also clear based on Figure 5.16 that our model misclassified a good number of tabs

since the values within the confusion matrix are very widely distributed and not very much focused around the diagonal. These misclassifications can be the result of incorrect hand pose estimation, especially in terms of depth or incorrect frets detection.

The hand pose estimation step of our project predicts the 3D coordinates of 21 joints of the hand. The depth coordinate (z coordinate) of a joint is used to compare it with the depth of the fretboard so we can assess whether that joint is pressing or no. Any incorrect depth estimation can cause either for a joint to be incorrectly labelled as pressing or not pressing thus lowering the accuracy of our approach and causing more misclassifications.

Another main reason that leads to such results is the incorrect detection of frets. Indeed, sometimes our method incorrectly detects frets since it is based on image processing, meaning that it is hard to find dynamic thresholds or thresholds that can work for different videos. This hinders our accuracy as well.

CHAPTER 6 CONCLUSION

In this last chapter, we conclude our work. We start with a summary of the works presented in this thesis, in addition to the limitations of our approach including the different parts of it before finally discussing the future work that can be done to improve our approach and further increase the performance.

6.1 Summary of Works

In this work, multiple parts were developed to complete an automatic guitar tablature transcription system. In the beginning, a RGBD camera had to be chosen which was later used to collect videos of guitarists playing different guitars. Both modalities RGB and depth are required since our approach uses the former for the segmentation, optical flow computation, strings and frets detection; while the latter is used alongside the hand pose estimation step to know whether a joint is pressing on a string or no.

Furthermore, we developed three main tasks which results will be used later, these different tasks are the segmentation of the fretboard, the fretting hand pose estimation and optical flow computation. The first one, the segmentation was achieved through fine-tuning a pre-trained Mask R-CNN with a ResNet50-FPN backbone on our own custom dataset for fretboard segmentation. For the second task, we used MediaPipe Hands for the hand pose estimation task. MediaPipe Hands is a lightweight model developed by Google who returns 3D coordinates for 21 joints of the hand [34]. Since the returned depth coordinates are not absolute values, we tested our own method to retrieve absolute depth values using the output of MediaPipe and the depthmap from the camera.

While for the previous two tasks deep learning was involved, for the optical flow computation the solution was simply using the Farneback algorithm [38].

To use the results of all the previous tasks, a post-processing step was developed. This step uses the segmentation result to isolate the fretboard from the rest of the image for strings and frets detection, the hand pose estimation result alongside the depth map to detect pressing joints after using the optical flow for detecting the picked or plucked string or strings.

After developing our approach, the different parts of our approach had to be tested to measure the performance of every task on its own, in addition to the entire methodology from end to end. For the segmentation, we built our own fretboard segmentation dataset on which our model scored 0.9230 for Dice; while for the hand pose estimation we experimented with

different ways of predicting depth, which proved that through linear regression we achieved the lowest error rate. Furthermore, to test both picking detection and our transcription method in general, videos of people playing different guitars had to be collected. For the former the mean accuracy was at 57.05%, while for the latter the accuracy was 22.11%.

From our experiments we can draw several contributions and findings, in addition to identifying multiple limitations that help us pinpointing future work.

6.2 Contributions and Findings

Despite the results obtained, several contributions and findings can be pointed out from this work.

To our knowledge, this project is the first attempt for a complete automatic transcription system that uses only a RGBD camera without any other special equipment. It is also the first work, where depth maps, hand pose estimation and optical flow are used for transcription reasons; and where picking detection is actually attempted for the same reason.

Moreover, a fretboard segmentation dataset was built to train our Mask R-CNN model for the localization of the fretboard in RGB images which helped us score 0.9230 for Dice on its test set. Compared to the Guitar Transcription Dataset found on Kaggle [1], our dataset seems to be much more diverse and leading to better generalization of the Mask R-CNN model.

Furthermore, the picking detection, despite the errors, seemed to work well with thicker strings, such as string 5 and string 6. Our method relies heavily on picking detection, thus making it able to detect hovering. For example, if a guitar player plays an open string on a guitar our method is able to detect open strings through the use of picking detection and hand pose estimation in 3D.

6.3 Limitations

As seen earlier, our results lack accuracy and our approach is significantly prone to errors and mistakes. This can be attributed to many factors and limitations.

The first limitation we can identify is the camera. Indeed, despite being the most suitable consumer-grade RGBD camera we could use, however, it is still not ideal. This is mainly because of its resolution, a 1280×720 resolution is not perfect especially when trying to detect the thin strings as they may not be clearly visible sometimes. This greatly affects our optical flow computation as it will be unable to detect picking on those strings, in addition to being

undetectable by the strings detection step. We also did not use multiple cameras or different views to capture as much as possible the needed information. This would particularly help the hand pose estimation where we can attempt to make sure the maximum number of joints is visible.

Another limitation is the hand pose estimation used. Indeed MediaPipe Hands is not suited for our case (a fretting hand on a guitar) and probably was never trained on similar data for the prediction of hand joints. Even if a model exists for this task, predicting only joints is arguably not ideal for such task, as sometimes guitarists press with parts even between joints; for example, playing a perfect fourth on a standard tuned guitar. Also, the poor depth estimation affected our results and depth estimation for such task has to be the most precise possible since it can greatly impact the outcome.

Despite that the Farneback algorithm [38] works pretty well, however, it remains vulnerable towards noise in addition to being an old method and not as precise as newer methods.

Finally, for the post-processing step using image processing algorithms may not be the ideal solution since a lot of them rely on thresholds and non-dynamic values such as the Probabilistic Hough Line transform and the different filters used (morphological, Sobel, etc.).

6.4 Future Work

In this section we present the future work that can be carried for our project. The future work presented here includes all different parts of the project; segmentation, hand pose estimation, optical flow and post-processing.

6.4.1 Camera Placement

The camera placement itself can be investigated and improved. For example, multiple cameras could be used to cover as much as possible the biggest number of hand joints. In addition, viewpoints like "The Magnet" cellphone holder [1] or something similar to the Novaxe platform[2].

6.4.2 Fretboard Segmentation

Several improvements can be made to further improve the fretboard segmentation. The suggestions in this section or the future work that can be carried out to improve this task,

[1] https://troygrady.com/
[2] https://www.novaxe.com/

are mainly based on the issues identified in Section 5.2.2.

Keeping the same dataset may mean the need for testing new hyperparameters or changing the confidence threshold. To do the former, more combinations need to be tested since 576 combinations exist within the parameters we chose to test and their ranges (3 for the optimizer, 4 for the learning rate, 6 for the epochs and 8 for the batch size, hence $3 \times 4 \times 6 \times 8 = 576$). Another avenue, could be adding new parameters within the parameters list, such as whether using a learning scheduler or no, the backbone, etc. or extending the range of the existing hyperparameters.

Changing the confidence threshold could also be another solution. To know the perfect threshold is a matter of testing, and could be determined empirically. A good way of doing it, can also be by considering a hyperparameter to be tested with the rest of hyperparameters. However, adding too many hyperparameters can cause the training to run for a very long time and hence a compromise or balance must be found.

Another improvement that can be introduced in the future, is adding more data. Many datasets have thousands and millions of images that can be used to train a model to perform a certain task, it would be highly beneficial to enlarge our dataset through adding many more images. Another version of the dataset could be built, where new additions would be introduced to the already existing data. This would be greatly profitable since adding more data means more variety for the model if the images are not similar to each other. One important note on this, is that the new images to add should be varied enough to help the model better predict and improve its robustness instead of being homogeneous and making the model fall into overfitting of a certain type of images or cases.

Finally, we think that many architectures can be tested. These architectures can be other architectures in the literature or architectures that we can design. Either way, Mask R-CNN works well, but other architectures may perform better. This would be a great improvement especially if a model with fewer parameters (weights) can perform at the same level as Mask R-CNN or better, resulting in a faster training, inference, less memory and GPU consumption and lighter model.

6.4.3 Hand Pose Estimation

As explained earlier, the part of hand pose estimation in this work is essential in order to know what strings are pressed. However, hand pose estimation can be much improved for the task of predicting the joints of a fretting hand on a guitar. Also, other approaches than hand pose estimation can be used for this task, all those options will be explained below.

- Create a guitar-specific hand pose estimation dataset: To our knowledge, no dataset exists for such use case. Hence, creating a new dataset specifically tailored for hand pose estimation of a fretting hand would be highly beneficial and would help increase the accuracy since this task can be very challenging for deep learning models because a fretting hand can have many self-occlusions.

- Train a model for press detection: Instead of going through hand pose estimation, we can directly try to train a deep learning model on the press detection using 3D data, and if good enough, may be it could be done using 2D data only. In other words, a model can be trained on learning how to know which strings are pressed, regardless of the joint. By doing so, we would avoid performing the hand pose estimation task, and we would need to use or create datasets such as the Guitar Transcription Dataset on Kaggle (Released in December 2021) [1].

6.4.4 Optical Flow and Picking Detection

Detecting picked/plucked strings proved to be tricky task, however, future work can be carried out to improve the outcomes of optical flow-based picking detection.

Despite this method being relatively able to detect picked strings, however it remains not ideal. This is mainly due to the fact that the optical flow is unable to return flow images containing lines representing picked thin strings. Thin strings are hard to detect, and hard to be seen with low resolutions, despite the D455 camera used having a 1280×720 resolution it still was not enough to visualize thin strings very well. This problem is particularly noticeable on metal thing strings on electric guitars, especially on the electric guitar with 22 frets on which the thinnest strings gauge was used.

In the future, it would be interesting to see how the optical flow and our picking detection method would behave with a resolution sufficiently high for capturing thin strings (especially the first, second and third strings). Another aspect that can be worked on in the future, is the lighting. The lighting used in for the videos of this project is similar to the image shown in 4.8a, regular room lighting. Different lighting conditions may prove helpful however that remains to be investigated.

Furthermore, the optical flow is vulnerable to noise, meaning if noise is introduced between two frames the optical flow computation method used would still show some flow. This point can backfire on our results sometimes, thus it would also be beneficial to try to develop a method for detecting strings optical flow or a way of detecting picked strings directly.

To do this, a CNN would have to be developed to achieve strings optical flow computation

or string picking detection immediately without passing through optical flow. Either way, a dataset has to be built through data collection and annotation in addition to experimenting with the best architecture that suits such a problem all while testing on different types of guitar and strings.

6.4.5 Post-Processing

Improving other components of our approach does not necessarily mean that our post-processing does not require improvements as well. The post-processing adopted in this project is highly dependent on predefined thresholds like in the Hough transform and morphological operations, in addition to not being robust to noise.

Defining these thresholds or values statically and manually is not ideal, as well as not being desirable. One way of addressing these shortcomings is to replace the post-processing by a small CNN that takes mixed input data, meaning that it would take the binary segmentation mask, optical flow image and the hand joint coordinates and predicts tabs as output.

This would also require data collection and annotation, however, it would probably be a great improvement as a neural network can learn to identify strings, match them to picking and learn how to recognize pressing joints.

6.4.6 Transcription System

All the previous future work will possibly improve the process of the automatic transcription, however, the methodology or the approach itself can be redesigned as well. Instead of using such a system where data and outputs are passed on from one module to the next, a deep learning method can be designed for the visual transcription of tablature.

Such a model can be trained to learn the task of transcription, where different sub-networks of the model would learn different tasks, for example, one would learn to detect pressed strings, another for localizing the fretboard and a third detecting picked strings with a few layers at the end to use the outputs of these sub-networks to finally output tablature. The training of such model would require many steps, such as training each sub-network separately to make sure they extract the features and predict what they are supposed to.

Another possibility could be Multi-Task Learning. A dataset such as the Guitar Transcription Dataset on Kaggle [1] can be used for that, where the network learns for each finger to detect whether it pressing or no, the tab index, as well as the string index.

In addition to the aforementioned, another suggestion would also be building a dataset where

inputs would be videos of people playing the guitar while the ground-truth would be text files of the corresponding tablature for each video. This dataset can later be used to train an RNN to learn the task of transcription from end-to-end. An interesting comparison can be conducted between this methodology and the one proposed previously as well.

Furthermore, since the field of computer vision-based automatic tablature transcription is not widely studied, a lot of research and experiments can still be conducted. In other words, a lot of uncertainties are still present, and a lot of improvement opportunities exist; thus, for the future, multiple works can still be conducted to further understand and improve the task of visual transcription of guitars.

Finally, the use of audio in our project can be a good addition that can be helpful especially in detecting picking. Similarly to [22], pitch detection can be performed to disambiguate or help correct tablature.

REFERENCES

[1] J. Lightfoot and D. Wijesinghe, "Guitar transcription dataset," 2021. [Online]. Available: https://www.kaggle.com/dsv/2873933

[2] "Body anatomy: Upper extremity joints | the hand society," WebPage, Jun. 2021. [Online]. Available: https://www.assh.org/handcare/safety/joints

[3] "File:hand001.jpg - wikimedia commons," WebPage, Dec. 2021. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/f/f3/Hand001.jpg

[4] J. Scarr and R. Green, "Retrieval of guitarist fingering information using computer vision," in *2010 25th International Conference of Image and Vision Computing New Zealand*, Nov 2010, p. 1–7.

[5] "Frequencies of Musical Notes, A4 = 440 Hz." [Online]. Available: https://pages.mtu.edu/~suits/notefreqs.html

[6] A.-M. Burns and M. Wanderley, "Visual methods for the retrieval of guitarist fingering." in *NIME '06: Proceedings of the 2006 Conference on New Interfaces for Musical Expression*, 2006, p. 196–199.

[7] C. Kerdvibulvech and H. Saito, "Real-time guitar chord estimation by stereo cameras for supporting guitarists," in *Proceedings of the International Workshop on Advanced Image Technology (IWAIT'07)*, 2007, p. 256–261.

[8] B. Duke and A. Salgian, "Guitar tablature generation using computer vision," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11845 LNCS, 2019, p. 247–257.

[9] Z. Wang and J. Ohya, "Detecting and tracking the guitar neck towards the actualization of a guitar teaching-aid system," *The Abstracts of the international conference on advanced mechatronics : toward evolutionary fusion of IT and mechatronics : ICAM*, vol. 2015.6, p. 187–188, 2015.

[10] ——, "Tracking the guitarists fingers as well as recognizing pressed chords from a video sequence," in *IS and T International Symposium on Electronic Imaging Science and Technology*, San Francisco, CA, United states, 2016, iSSN: 24701173.

[11] R. M. Mottola, "Calculating fret positions." [Online]. Available: https://www.liutaiomottola.com/formulae/fret.htm

[12] S. Goldstein and Y. Moses, "Guitar music transcription from silent video." in *BMVC*, 2018, p. 309.

[13] L. Tran, S. Zhang, and E. Zhou, "Cnn transfer learning for visual guitar chord classification," p. 6.

[14] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[15] T. Ooaku *et al.*, "Guitar chord recognition based on finger patterns with deep learning," in *Proceedings of the 4th International Conference on Communication and Information Processing*, ser. ICCIP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 54–57.

[16] Z. Wang and J. Ohya, "A 3d guitar fingering assessing system based on cnn-hand pose estimation and svr-assessment," in *IS and T International Symposium on Electronic Imaging Science and Technology*, 2018, p. 2781–2785.

[17] Y. Kashiwagi *et al.*, "A study of performance detection method for a guitar skill learning using kinect sensor," in *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*, Oct 2015, p. 343–345.

[18] Y. Kashiwagi and Y. Ochi, "A study of left fingering detection using cnn for guitar learning," in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, Mar 2018, p. 14–17.

[19] G. Byambatsogt, L. Choimaa, and G. Koutaki, "Guitar chord sensing and recognition using multi-task learning and physical data augmentation with robotics," *Sensors (Switzerland)*, vol. 20, no. 21, p. 1–17, 2020.

[20] A. Wiggins and Y. Kim, "Guitar tablature estimation with a convolutional neural network," Delft, Netherlands, 2019, pp. 284 – 291.

[21] Q. Xi *et al.*, "Guitarset: A dataset for guitar transcription," Paris, France, 2018, pp. 453 – 460.

[22] M. Paleari *et al.*, "A multimodal approach to music transcription," in *2008 15th IEEE International Conference on Image Processing*, Oct 2008, p. 93–96.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[24] K. He *et al.*, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, p. 2980–2988.

[25] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[26] I. Tono *et al.*, "Guitar segmentation in rgb images using convolutional neural networks," in *2020 IEEE 21st International Conference on Computational Problems of Electrical Engineering (CPEE)*, Sep 2020, p. 1–4.

[27] K. Sun *et al.*, "High-resolution representations for labeling pixels and regions," 2019.

[28] L.-C. Chen *et al.*, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[29] S. Jegou *et al.*, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[30] S. Yuan *et al.*, "Depth-based 3d hand pose estimation: From current achievements to future goals," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, p. 2636–2645.

[31] J. Y. Chang, G. Moon, and K. M. Lee, "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Proceedings*, 2018, p. 5079–88.

[32] M. Oberweger and V. Lepetit, "Deepprior++: Improving fast and accurate 3d hand pose estimation," 2017.

[33] W. Huang *et al.*, "Awr: Adaptive weighting regression for 3d hand pose estimation," New York, NY, United states, 2020, pp. 11 061 – 11 068, 3D hand pose estimations;Adaptive weighting;Adaptive weights;Aggregation process;Input modalities;Joint coordinates;State-of-the-art methods;Weight maps;.

[34] F. Zhang *et al.*, "Mediapipe hands: On-device real-time hand tracking," 2020.

[35] Google, "Hands." [Online]. Available: https://google.github.io/mediapipe/solutions/hands.html

[36] M.-A. Drouin and L. Seoud, *Consumer-Grade RGB-D Cameras*. Cham: Springer International Publishing, 2020, pp. 215–264.

[37] Intel, "Intel realsense product family d400 series datasheet," Aug 2021. [Online]. Available: https://www.intelrealsense.com/download/15409/?_ga=2.262235084.1383688231.1639433804-1908508730.1612538883

[38] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *Image Analysis*, J. Bigun and T. Gustavsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370.

[39] J. Kovac, P. Peer, and F. Solina, "Human skin color clustering for face detection," in *The IEEE Region 8 EUROCON 2003. Computer as a Tool.*, vol. 2, 2003, pp. 144–148 vol.2.

[40] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119 – 137, 2000, progressive probabilistic Hough transform;Robust line detection;.

[41] A.-M. Burns, "Computer vision methods for guitarist left-hand fingering recognition," p. 103, 2007.

APPENDIX A MUSICAL PIECES FOR TESTING

To record our testing videos several musical pieces were chosen in order to assess our system performance in certain scenarios and in the best way possible.

In this appendix, we list the different musical pieces we chose to record and the reasons they were chosen.

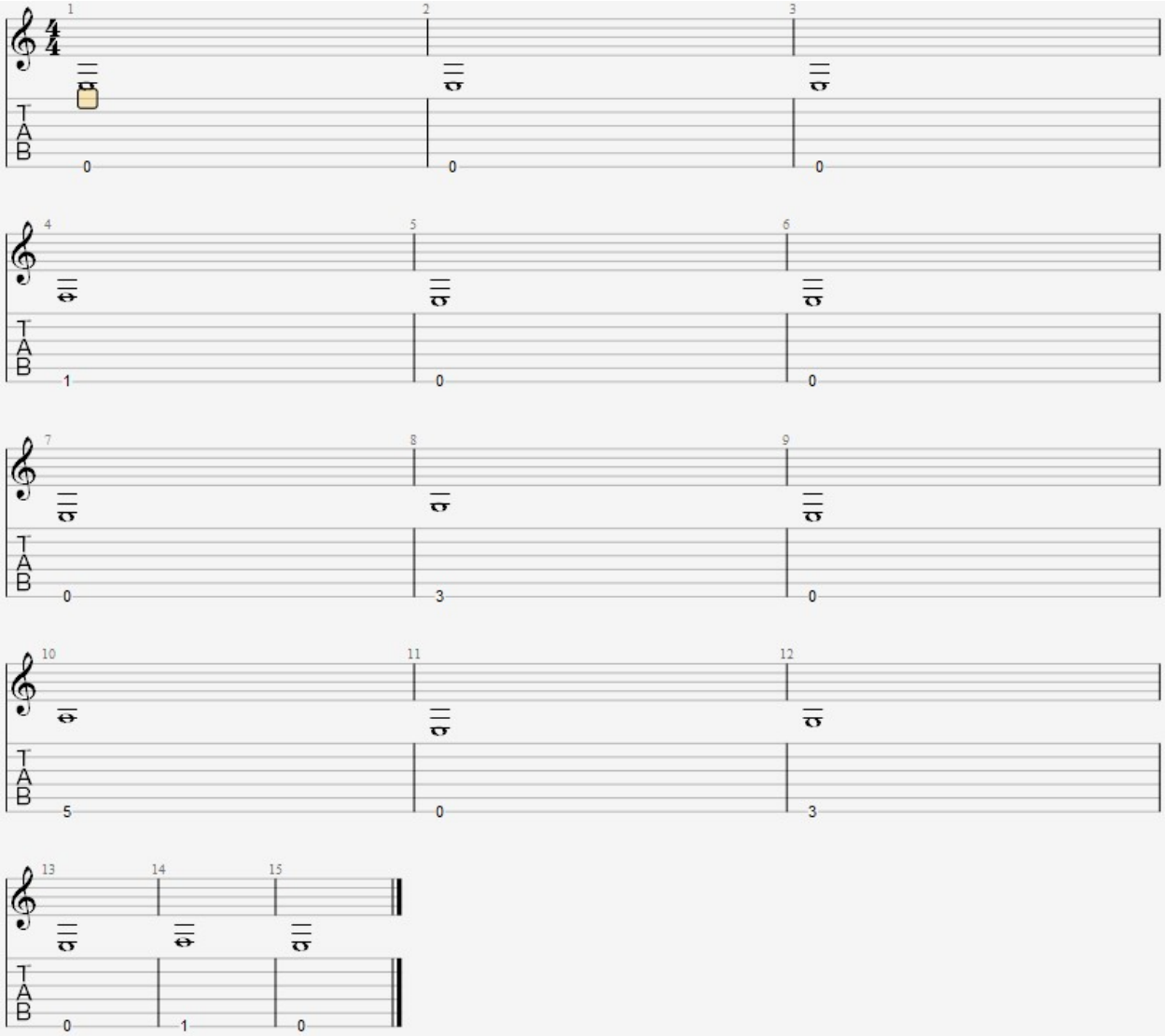
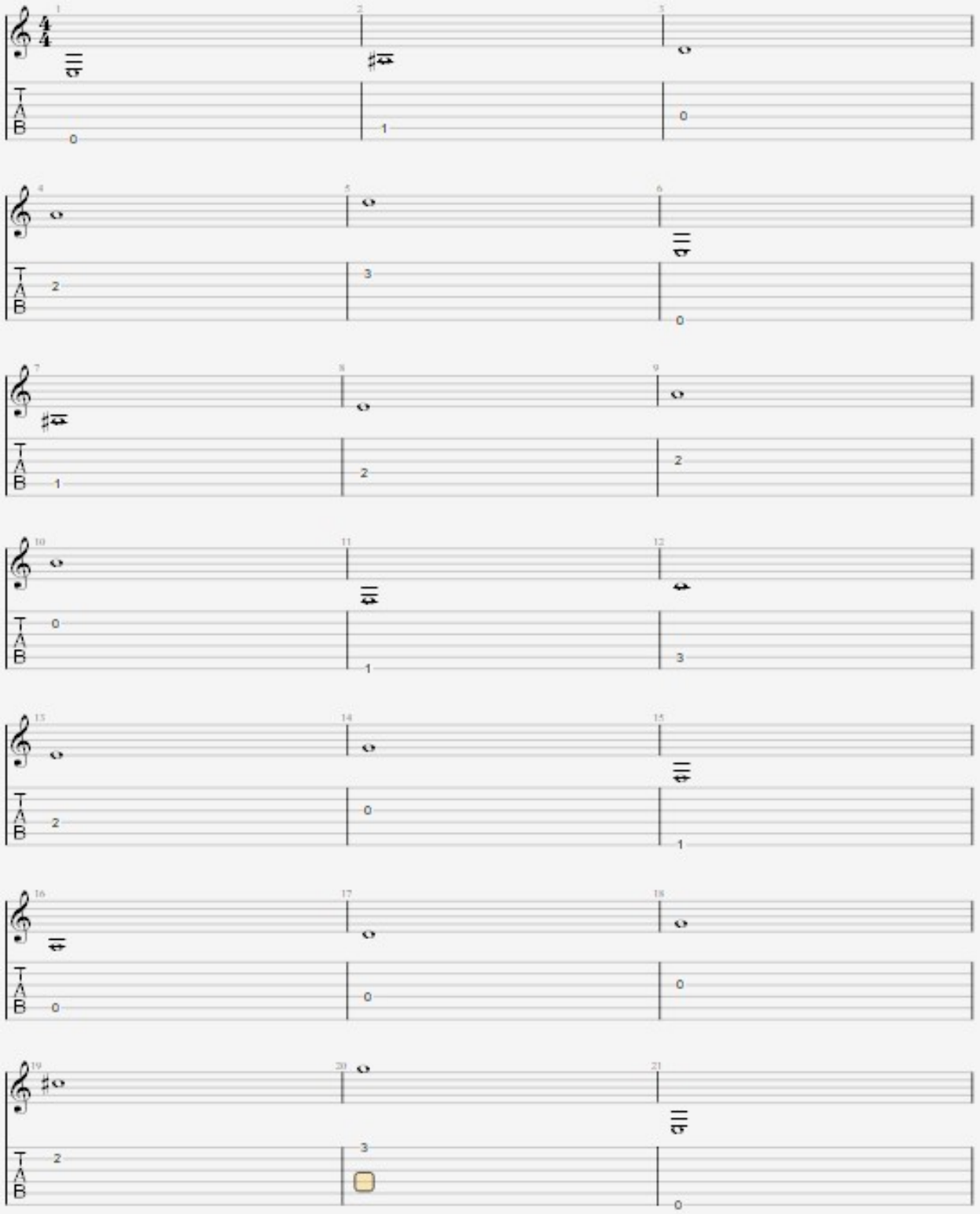


Figure A.1 Tabs used for this piece made of open strings (zeros) and hovering to test the detection of open strings.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
T
A
B

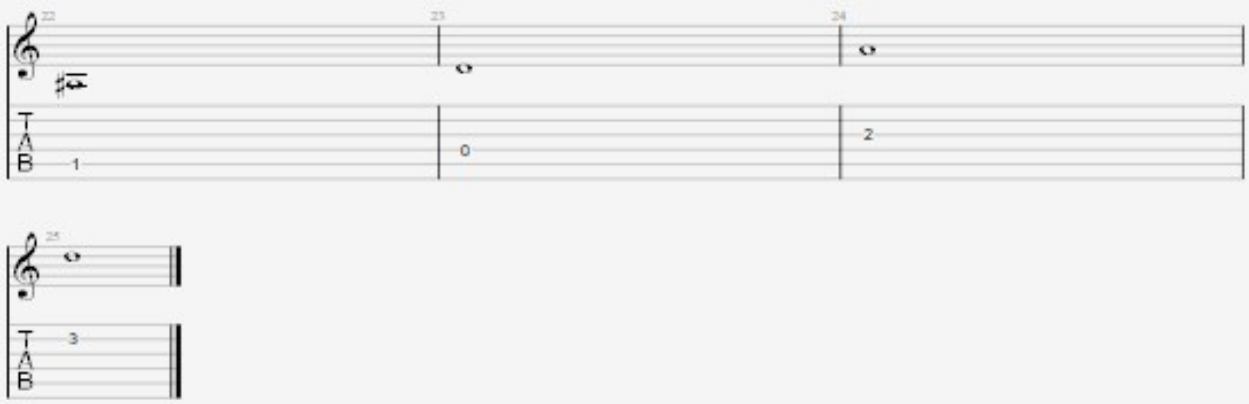


Figure A.2 Tabs used for this piece that is made of the single notes of the C major chord progressions used in [4].

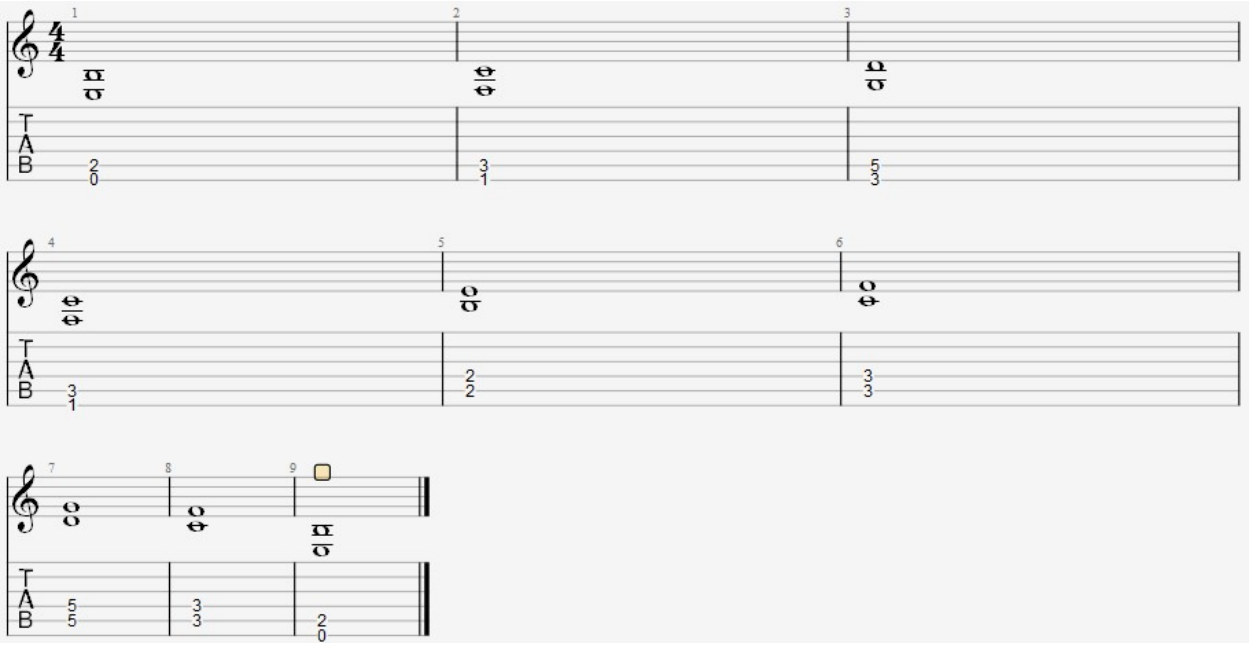


Figure A.3 The tabs used for this piece that is composed of 5^{th} and 4^{th} intervals.

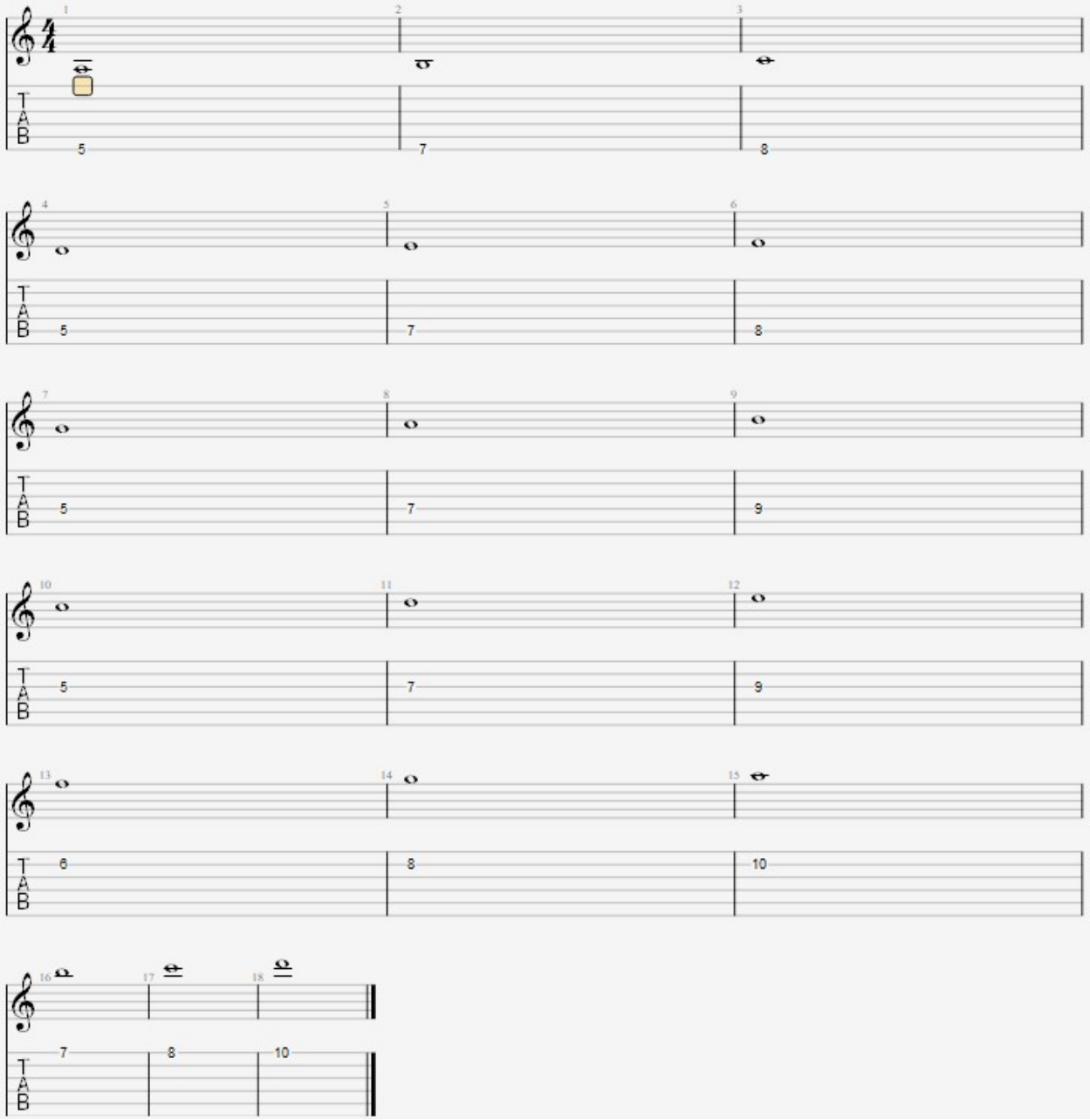


Figure A.4 The tabs used for this piece that represents the A minor scale played on a E standard tuned guitar with 3 notes per string.

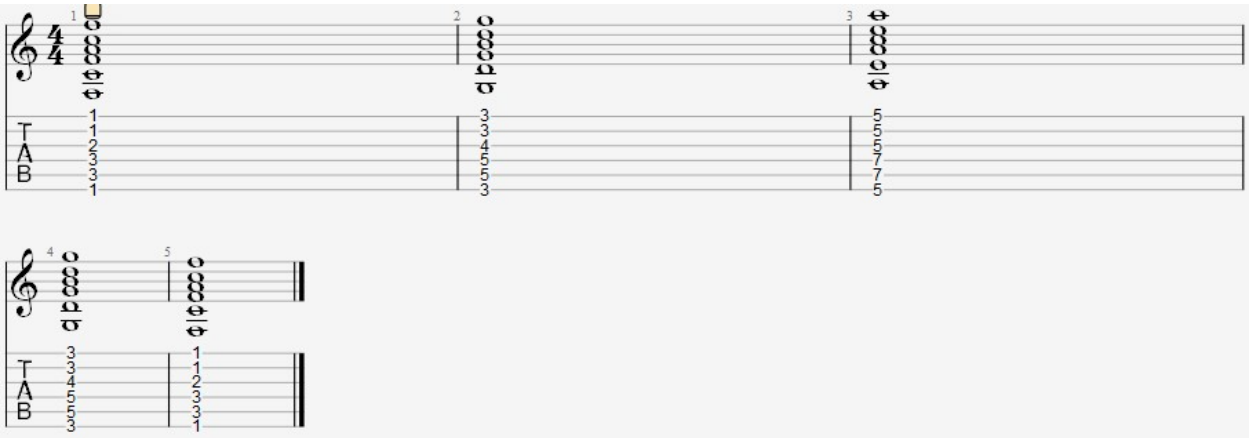


Figure A.5 Tabs used for this piece that is made of barre chords.

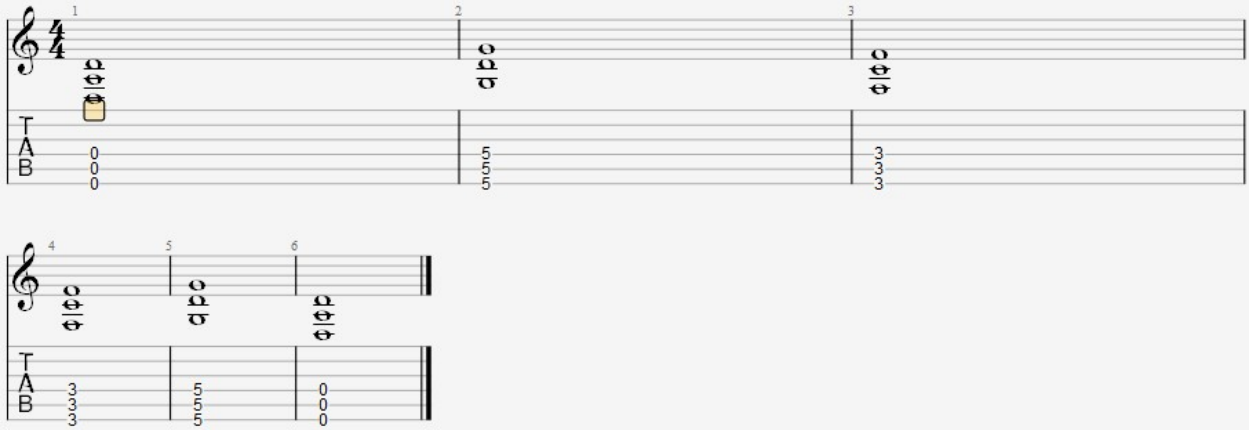


Figure A.6 Tabs used for this piece that is made of 3 strings barre chords. This piece would usually be played on a drop tuned guitar.

The "C major chord progression" (present in [41] and also [4] as "C major chords") would help us determine whether this approach will work with chords. The "C major scale" and "Ode an die Freude" also in [41] and used in [4] would test the ability of the approach to detect single notes. Other 6 were added that serve different purposes. Unlike most methods mentioned in the literature review section, we attempt to detect open strings, and to do so we need a musical piece to help us evaluate that use case. The "Hovering" piece shown in Figure A.1 helps us determine whether the model is able to work with open strings while also introducing different pressings (non-open strings) in between.

The "C major chord progression-singles", is the same as "C major chord progression" in [41] and used in [4], however, in this one the notes of every chord are played separately. This

is used to show that even if a finger is pressing, no transcription should happen unless the corresponding string was picked.

The "5^{th} 4^{th}" piece displayed in Figure A.3, consists of perfect 5^{th} and perfect 4^{th} intervals. It is used for assessing our methods accuracy regarding intervals.

The "A minor 3 per string" piece in Figure A.4 adds another video with single notes in it, however, this time 3 notes are played per string before moving on to the next one. Thus, if we have multiple fingers on one string only the highest pressed fret should be taken.

One of the added pieces is what we call "Barre chords", shown in Figure A.5, adds barre chords to our testing set and so our model can be tested with barre chords.

The last piece, "Drop power chords" (we also call it "Drop D" since it is played on a drop D tuning) as in Figure A.6, adds some power chords to our videos. These chords are usually played in a drop tuning, similar to barre chords but the finger does not press across all strings.