Guitarist Hand Information Retrieval Using Computer Vision

A Thesis

submitted to the designated
by the Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Panagiotis Kouzouglidis

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION
IN DATA SCIENCE AND ENGINEERING

University of Ioannina
School of Engineering
Ioannina 2023

Examining Committee:

- **Christophoros Nikou**, Professor, Department of Computer Science and Engineering, University of Ioannina (Advisor)
- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Konstantinos Blekas**, Professor, Department of Computer Science and Engineering, University of Ioannina

Dedication

Dedicated to my parents and to my brother and sister for supporting me financially and spiritually throughout all these years of my studies.

Acknowledgements

I would like to thank my supervisor, professor Christophoros Nikou, for always being helpful, supporting and passing on to me advanced knowledge through his master course of computer vision. Also, I would like to thank Giorgos Sfikas, whose feedback and advice helped me a lot during the beginning steps of this master thesis.

Table of Contents

List of Figures

List of Tables

Abstract

Panagiotis Kouzouglidis, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2023.
Guitarist Hand Information Retrieval Using Computer Vision.
Advisor: Christophoros Nikou, Professor.

Music Transcription is the process of extracting information from a song or piece and converting it into some form of music notation. Traditionally, it is a time consuming task that is performed manually and it requires significant music knowledge and experience from the musician. The Automatic Music Transcription (AMT) problem has been mainly approached through audio signal processing. However, extracting pitch from an audio recording can be a difficult task due to the dependency of frequencies to factors such as the type of the instrument, tuning, sound enhancement using effects (e.g. distortion) and others.

In this work we study the problem of Guitarist Hand Information Retrieval (GHIR) which corresponds to a vision-based approach of the AMT problem for the guitar instrument. GHIR is a complicated and difficult task since it consists of many sub problems. First, we detect the guitar fretboard which is the area of the guitar where the fingers are placed and moving. Next, we detect the positions of the guitar's strings and frets. Then, we localize the fingertips of the guitarist's left hand that presses the notes. Finally, we combine the aforementioned information to predict for each string which note is pressed.

We try to solve the GHIR problem without using any markers, that could be useful for higher accuracy of fretboard detection or fingertip localization. Also, unlike other works we don't use frame references to interpolate frets that are missing due to the occlusion from the guitarist hand. We process, analyze and extract information from

each frame independently. In addition, we don't use any prior knowledge of the guitar's fret distances.

We introduce the problem of pick position prediction, which gives us the information of the candidate string the guitarist hits.

We evaluate our model on a dataset that we created and annotated. The dataset consists of video recordings, where we play a set of chosen chords and scales on guitar. We present our results and we compare the performance of our model to similar experiments of other works. We analyze the error cases of our model and we propose future improvements and extensions.

Εκτεταμενη Περιληψη

Παναγιώτης Κουζουγλίδης, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2023.
Ανάκτηση Πληροφορίας των Χεριών του Κιθαρίστα με Χρήση Υπολογιστικής Όρασης.
Επιβλέπων: Χριστόφορος Νίκου, Καθηγητής.

Η Μετεγγραφή Μουσικής (Music Transcription) είναι η διαδικασία εξαγωγής πληροφορίας ενός μουσικού κομματιού και η αποτύπωσή του σε κάποια μορφή μουσικής σημειογραφίας. Παραδοσιακά αυτή είναι μια αρκετά χρονοβόρα διαδικασία καθώς γίνεται χειροκίνητα και απαιτείται η μουσική γνώση και αντίληψη του ακροατή ώστε να μπορεί με το αυτί να αναγνωρίσει τις νότες του μουσικού κομματιού. Κατά το παρελθόν έχει μελετηθεί το πρόβλημα της Αυτόματης Μετεγγραφής Μουσικής (Automatic Music Transcription, AMT) κυρίως από την σκοπιά της ανάλυσης του ηχητικού σήματος. Το AMT πρόβλημα με την χρήση του ηχητικού σήματος περιέχει αρκετές δυσκολίες λόγω των πολλών διαφοροποιήσεων που μπορεί να έχει ο χώρος των συχνοτήτων, οι οποίες προκύπτουν είτε από την φύση του οργάνου, είτε από το εκάστοτε κούρδισμα, είτε από την χρήση ηχητικών εφέ και παραμορφώσεων (distortion) -που μπορεί να χρησιμοποιηθούν κατά την εκτέλεση του κομματιού- κ.α.

Στην παρούσα εργασία μελετάμε το πρόβλημα της Εξαγωγής Πληροφορίας των Χεριών ενός Κιθαρίστα (Guitarist Hand Information Retrieval, GHIR) με την χρήση Υπολογιστικής Όρασης (Computer Vision). Η εξαγωγή αυτής της πληροφορίας αποτελεί ουσιαστικά πρόβλημα μετεγγραφής μουσικής, καθώς τελικός στόχος είναι να μετατρέψουμε το μουσικό κομμάτι -που παίζεται στην κιθάρα- σε μουσική ταμπλατούρα. Το GHIR είναι αρκετά πολύπλοκο και δύσκολο πρόβλημα καθώς αποτελείται από διάφορα επιμέρους υπο-προβλήματα. Αρχικά εντοπίζουμε τον λαιμό της

κιθάρας, δηλαδή την περιοχή στην οποία βρίσκονται και κινούνται τα χέρια του κιθαρίστα. Στην συνέχεια εντοπίζουμε τις θέσεις των τάστων αλλά και των χορδών της κιθάρας. Έπειτα, βρίσκουμε τις θέσεις από τις άκρες των δαχτύλων του αριστερού χεριού του κιθαρίστα. Τέλος, συνδυάζουμε όλα τα παραπάνω ώστε να προβλέψουμε για κάθε χορδή της κιθάρας ποια νότα ''πατάει'' ο κιθαρίστας.

Επιχειρούμε να λύσουμε το GHIR πρόβλημα χωρίς χρήση σημαδιών (markers) που θα μπορούσαν να μας βοηθήσουν είτε στον εντοπισμό του λαιμού της κιθάρας, είτε στον ακριβέστερο εντοπισμό των δαχτύλων του κιθαρίστα. Επίσης δεν κάνουμε χρήση καρέ αναφοράς (frame reference), όπως γίνεται σε άλλες δουλειές και τα οποία βοηθούν στην αντιμετώπιση της επικάλυψης (occlusion) των τάστων από το χέρι του κιθαρίστα. Αντιμετωπίζουμε κάθε καρέ μεμονωμένα και ανεξάρτητα και προσπαθούμε από αυτό και μόνο να εξάγουμε ό,τι πληροφορία μπορούμε, ώστε να φτάσουμε στο τελικό αποτέλεσμα. Τέλος δεν κάνουμε χρήση καμιάς εκ των προτέρων γνώσης σχετικά με τις αποστάσεις που έχουν τα τάστα της κιθάρας.

Εισάγουμε το πρόβλημα της εύρεσης της θέσης της πένας και η οποία θα μας δώσει την πληροφορία για το ποια χορδή είναι υποψήφια να παίζει ο κιθαρίστας σε οποιοδήποτε δοσμένο καρέ.

Αξιολογούμε το μοντέλο μας πάνω σε σύνολο δεδομένων που δημιουργήσαμε και επισημειώσαμε (annotated). Το σύνολο δεδομένων αποτελείται από βίντεο παιξίματος κάποιων συγχορδιών (chords) και κλιμάκων (scales) σε κιθάρα. Παρουσιάζουμε τα αποτελέσματα του μοντέλου μας και τα συγκρίνουμε με παρόμοια πειράματα από άλλες δουλειές. Τέλος, σχολιάζουμε τις αδυναμίες του καθώς και προτείνουμε ιδέες για επεκτάσεις και βελτιώσεις.

Chapter 1

Introduction

1.1 Thesis Objectives

In this thesis we address the problem of Guitarist Hand Information Retrieval (GHIR) which corresponds to a vision-based approach of the Automatic Music Transcription (AMT) problem. We propose a solution for each of the individual sub-problems, GHIR consists of. Specifically, the problems that we try to solve are the following:

1. Guitar fretboard area detection and normalization.
2. Guitarist's hands segmentation and fingertip localization.
3. Guitar fret localization.
4. Guitar string detection.
5. Fret prediction for each guitar strings (which notes are pressed from the guitarist's left hand).
6. Pick position prediction (which is the candidate string that the guitarist hits).

There are some previous works that address the same problem using different approaches. The thing that these works have in common is either the usage of markers,

either the usage of frame references. Markers are useful for higher detection accuracy of the guitar fretboard. Also, they can be used on fingertips to make their localization easier. On the other hand, frame references are frames that in most of the cases are processed manually and they are used to retrieve missing information (i.e. missing guitar frets due to the occlusion from the guitarist hand). Finally, they use known formulas to approximate guitar fret positions.

We attempt to solve the GHIR problem without using any of the aforementioned practices. We try to extract all the information that we can from each frame independently and without any prior knowledge. We take into account every little detail that we can extract from the processed frames. The motivation for this approach is to come up with a fully automated solution that has high accuracy and doesn't depend on previous frames or manual interference. If we manage to come up with this kind of solution then with simple extensions it could outperform all the existing methods.

1.2 Thesis Structure

The rest of this thesis is organized as follows: In Chapter 2 we provide the essential background knowledge that we needed for this work. We cover topics related to guitar, image processing and machine learning. In Chapter 3 we provide details for our approach and related work. In Chapter 4 we present the experimental results of our method and we compare them with similar experiments of other works. In Chapter 5 we summarize our findings and we provide a list of future improvements and extensions of this work.

Chapter 2

Background Knowledge

2.1 Guitar Essentials

In this section we mention the most important guitar concepts that are needed for a better understanding of this thesis.

2.1.1 Guitar Anatomy

The guitar is a popular stringed instrument that has been used in a wide variety of musical genres, including metal, rock, pop, country, blues, jazz, classical, and folk. The guitar consists of a long, fretted neck with six strings that can be plucked or strummed to create different notes and chords. There are many different types of guitars, including acoustic guitars, electric guitars, and classical guitars. In current thesis we use an electric guitar for our experiments.

The main parts of a guitar are:

1. **Headstock**: The headstock is located at the top of the guitar neck and holds the tuning pegs, which are used to tune the strings.

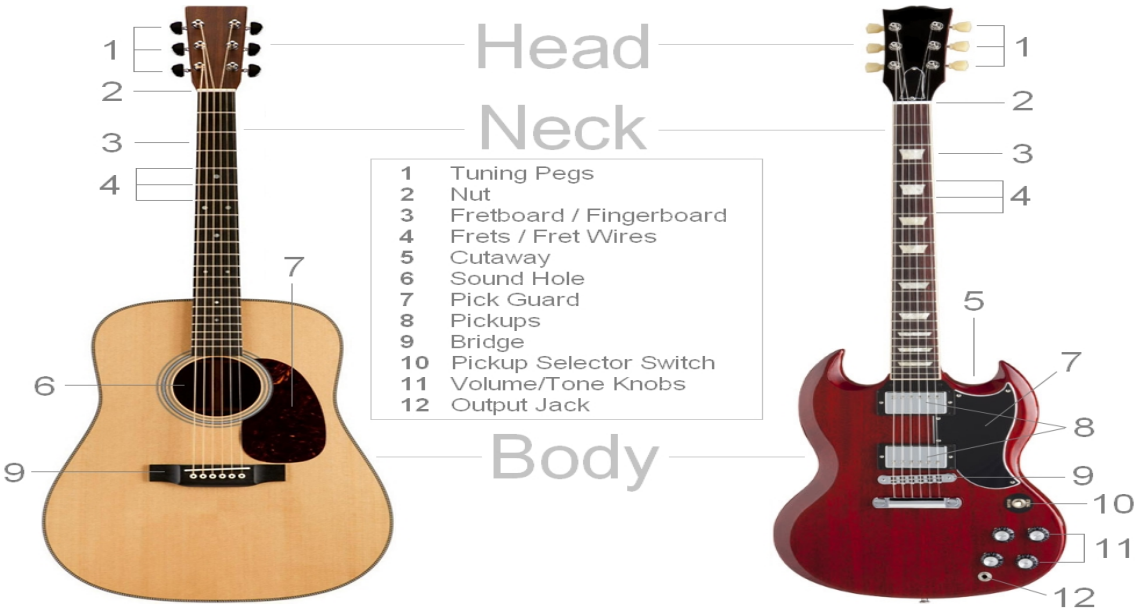

Figure 2.1: Guitar Parts

2. **Neck**: The neck is the long, thin part of the guitar that extends from the body. It holds the fingerboard, which is a smooth surface with raised metal frets that the musician uses to press down on the strings to change the pitch of the notes.

3. **Body**: The body is the largest part of the guitar and is typically made of wood.

4. **Strings**: The strings are thin wire ropes that are stretched across the neck and body of the guitar. They are plucked or strummed to produce the sound of the instrument.

5. **Bridge**: The bridge is a small piece of hardware located on the body of the guitar. It holds the strings in place and transfers their vibration to the body of the guitar, which helps to produce the sound.

6. **Sound Hole**: The sound hole is a small opening in the body of the guitar that allows sound to escape from the resonant chamber. Note that the electric guitars don't have a sound hole.

A detailed image for the guitar parts is provided in Figure 2.1

2.1.2 Tablature system

The guitar tablature system, also known as guitar tabs or simply tabs, is a musical notation system used to represent music played on the guitar. Unlike traditional sheet

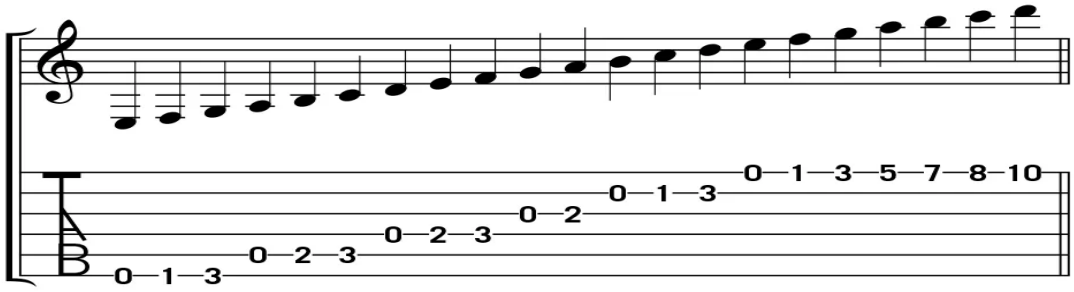

Figure 2.2: Guitar Tablature Example

music, which uses standard notation to represent the pitch and duration of each note, guitar tabs use a system of numbers and symbols to represent the location of each note on the guitar fretboard. Tabs are often used by guitarists to learn and play new music, particularly for popular songs and guitar solos.

In guitar tabs, each line represents a guitar string, with the top line representing the thinnest string (1st string) and the bot line representing the thickest string (6th string). For example, a ”0” on the bottom line means to play the open (unfretted) string, while a ”5” on the top line means to play the fifth fret on the 1st string. There are many techniques on guitar that have a specific representation in guitar tabs but we won’t go into further details, since they are not used in current work. An example guitar tablature is shown in Figure 2.2

2.1.3 Notes and Chords

In music, a guitar note is a sound that is produced by plucking, strumming, or otherwise vibrating a string on a guitar. When a guitar string is plucked, it vibrates at a specific frequency, which determines the pitch of the resulting note. The pitch of a note is determined by the frequency of the sound wave, with higher frequencies corresponding to higher pitches and lower frequencies corresponding to lower pitches. The 1st string of the guitar provides the highest pitches and the 6th string provides the lowest pitches. The guitarist presses different frets across the guitar to produce different notes. Each fret produces a different sound. The list of the guitar notes of the first 12 frets for the case of Standard Tuning, is shown in Figure 2.3.

Guitar notes can be played individually or in combination with other notes. A guitar chord is a combination of two or more notes played together on a guitar. The notes are played by strumming the strings of the guitar with the fingers or a pick. The

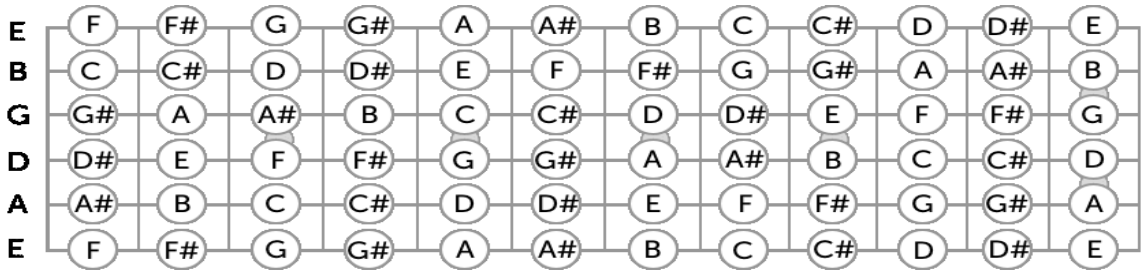

Figure 2.3: Fretboard Notes for Standard Tuning

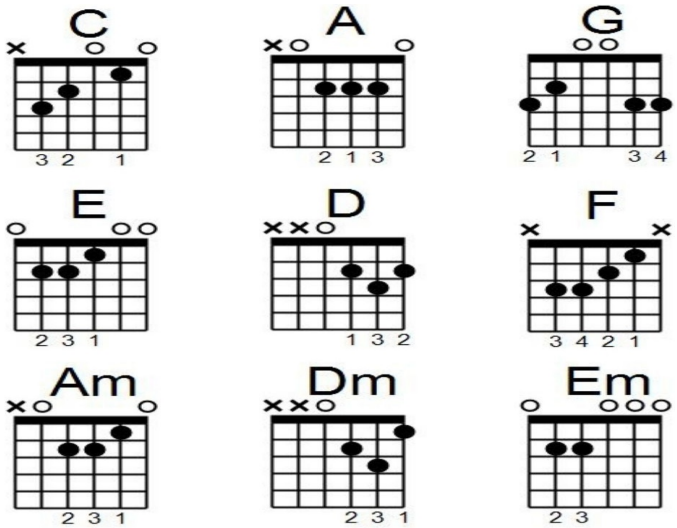

Figure 2.4: Chord Chart

most commonly used chords are shown in Figure 2.4. Also, we can play a sequence of chords in a specific order to create a chord progression.

2.2 Image Processing and Computer Vision

In this section we mention the background knowledge of Image Processing and Machine Learning that we needed for this work. We don't into details. Our purpose is to provide only an idea about the techniques of image processing and machine learning, that we use in this work.

2.2.1 Edge Detection Using Sobel Filters

Sobel edge detection is an image processing technique used to detect edges in digital images. It is named after Irwin Sobel, who developed the technique in the 1960s.

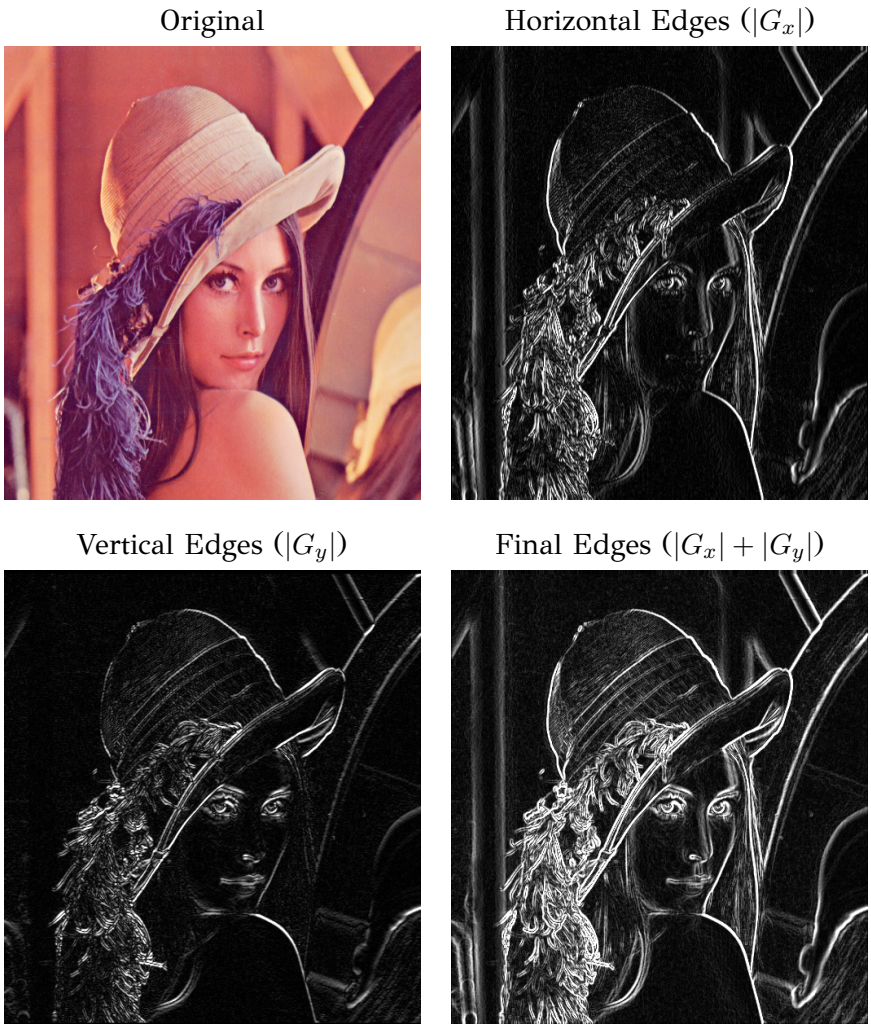

Figure 2.5: Edge Detection Using Sobel Filters

Sobel edge detection uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal edges, and one for vertical. For a given source image **A** the horizontal and vertical derivative approximations (G_x and G_y respectively) are calculated as follows:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \text{ and } G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A.$$

The gradient approximations can be combined to give the gradient magnitude, using $G = |G_x| + |G_y|$. The approximated derivatives of a selected image using the Sobel filters are shown in Figure 2.5.

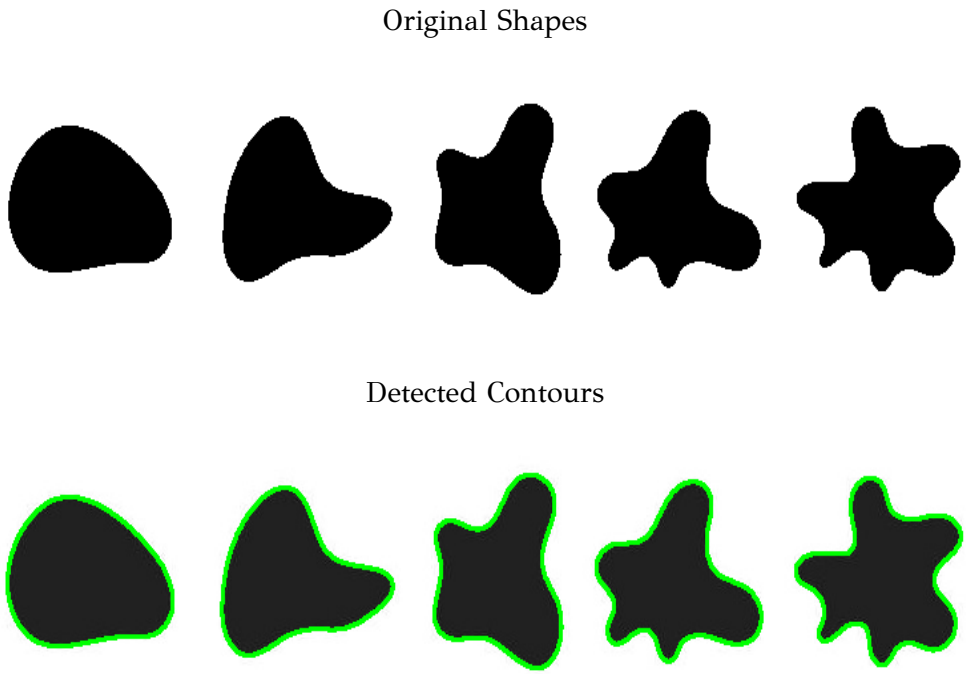

Figure 2.6: Contour Detection for Random Shapes

2.2.2 Contours

In image processing, a contour is a curve that connects points of equal intensity or color in an image. It can be used to extract important features in an image, such as boundaries or edges, and to segment the image into regions with different colors or intensities. The contour of an object in an image can be found using various algorithms, such as edge detection, thresholding, or region growing. Once the contour is found, it can be represented as a sequence of points that approximate the curve. An image illustration of contours for some random shapes is shown in Figure 2.6.

2.2.3 Filtering

Image filtering is a fundamental technique in image processing that is used to modify an image by applying a set of mathematical operations to each pixel or neighborhood of pixels in the image. The purpose of image filtering is to enhance or extract certain features of an image, such as edges, textures, or colors, or to remove noise or unwanted details. There are many types of image filters that can be used to achieve different effects, such as:

Figure 2.7: Image Smoothing

1. **Smoothing filters**: These filters are used to reduce noise or blur an image by averaging the intensity of nearby pixels. Examples of smoothing filters include the Gaussian filter, the median filter, and the bilateral filter.

2. **Edge detection filters**: These filters are used to highlight edges in an image by detecting regions where the intensity changes abruptly. Examples of edge detection filters include the Sobel filter, the Canny filter, and the Laplacian of Gaussian filter.

3. **Sharpening filters**: These filters are used to enhance the details or edges in an image by increasing the contrast of adjacent pixels. Examples of sharpening filters include the unsharp mask filter and the high-pass filter.

4. **Color filters**: These filters are used to modify the colors of an image by adjusting the hue, saturation, or brightness of the pixels. Examples of color filters include the color balance filter, the histogram equalization filter, and the color transfer filter.

An a example of smoothing filter applied to an image is shown in Figure 2.7.

2.2.4 Thresholding

Image thresholding is a technique in image processing that is used to convert a grayscale or color image into a binary image, where each pixel is either black or

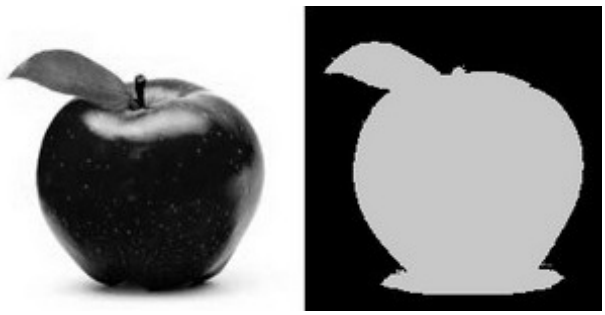

Figure 2.8: Image Thresholding

white (or 0 or 1). The goal of image thresholding is to separate objects from the background by finding a threshold value that separates the pixel values of the objects and the background. An example of image threshold is shown in Figure 2.8.

2.2.5 Morphological Operations

Morphological operations are a set of mathematical operations that are used in image processing and computer vision to modify the shape or size of objects in an image. The most common morphological operations are:

- **Dilation**: is used to expand the boundaries of objects in an image.
- **Erosion**: is used to shrink the boundaries of objects in an image.
- **Opening**: is the result of an erosion followed by dilation. It is useful in removing noise.
- **Closing**: is a dilation followed by an erosion. It is useful in used to fill small gaps or holes in an image

Both dilation and erosion are performed by applying a structuring element to the image, which is a small binary or grayscale image that defines the shape and size of the operation. In Figure 2.9 we see the result of a digital character after performing erosion and dilation accordingly. In Figure 2.10 we see the result of the morphological operations opening and closing.

2.2.6 Perspective Transformation

Perspective transformation, also known as projective transformation, is a mathematical process used in computer vision to convert an image of a scene taken from one

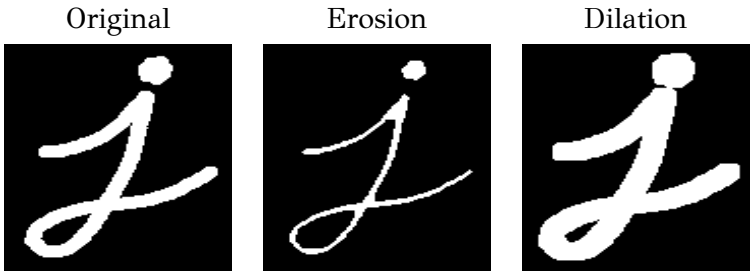

Figure 2.9: Morphological Erosion and Dilation

Figure 2.10: Morphological Opening and Closing

viewpoint into an image that appears as though it was taken from a different viewpoint. The goal of perspective transformation is to map a set of points in an image taken from one viewpoint onto a corresponding set of points in an image taken from a different viewpoint, such that the spatial relationships between the points are preserved. In Figure 2.11 we see the result of a perspective transformation that was applied to the handwritten paper.

2.3 Machine Learning

In this subsection we describe the concept of Linear Regression and K-means algorithm. We don't go into details since it's not necessary for the understanding of this work.

2.3.1 Linear Regression

Linear regression is a statistical technique used to model the relationship between two variables, where one variable is considered as the dependent variable and the other as the independent variable. The objective of linear regression is to find a linear

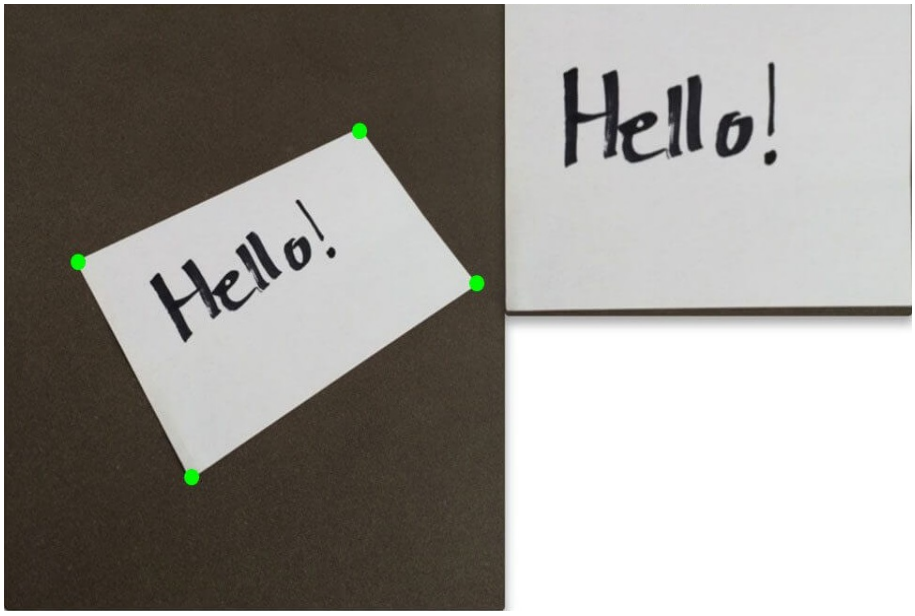

Figure 2.11: Perspective Transformation Example

relationship between the two variables, which can be used to predict the value of the dependent variable based on the value of the independent variable. Specifically, in linear regression, a line is fitted to the data points by minimizing the sum of the squared differences between the predicted values and the actual values of the dependent variable. The line is defined by an equation of the form $y = mx + b$, where y is the dependent variable, x is the independent variable, m is the slope of the line, and b is the y-intercept. The slope of the line represents the change in the dependent variable for a unit change in the independent variable. A graphical illustration of linear regression is shown in Figure 2.12.

2.3.2 K-means Algorithm

K-means is a clustering algorithm used in unsupervised machine learning, which groups a set of data points into k clusters based on their similarity. The algorithm works by iteratively assigning data points to the nearest cluster centroid, and then re-calculating the centroids based on the mean of the points in the cluster. Specifically the steps of the K-means algorithm are described below:

1. Specify the k which is the number of clusters.

2. Initialze randomly the k centroids.

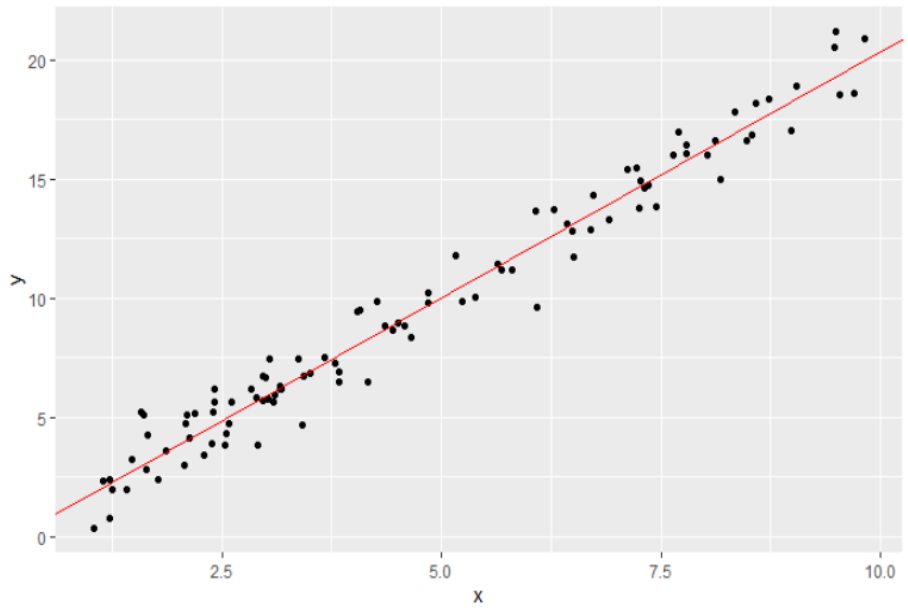

Figure 2.12: Linear Regression

3. Until centroid convergence repeat:

   (a) Assign each point to it's closest centroid.

   (b) Compute the new centroid of each cluster by computing the mean value of the points.

In Figure 2.13 is shown an example of the resulting clusters after the usage of the K-means algorithm.

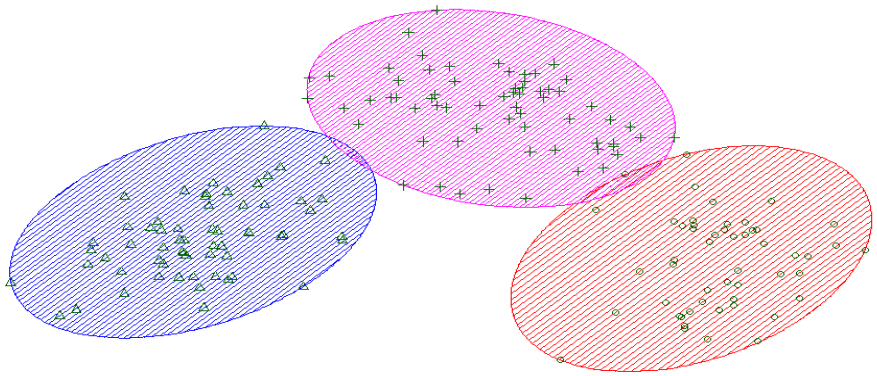

Figure 2.13: K-Means Algorithm (for k=3)

Chapter 3

Guitarist Hand Information Retrieval Using Computer Vision

3.1 Problem Definition

In GHIR problem, given a frame of a guitarist that plays the guitar, the objective is to return for each of the six strings, which note is pressed by the guitarist. GHIR corresponds to an Automatic Guitar Music Transcription problem.

3.2 Related Work

Automatic Guitar Music Transcription (AGMT) is an interesting problem and has been studied from different aspects over the last years using audio processing methods [1], [2], [3], [4], [5]. We are interested in works that approach the AGMT problem using vision-based methods. These works [6], [7], [8], [9], [10] provide a full solution for the GHIR problem while some other works focus on a specific sub-problem such as the guitar neck detection and tracking [11], [12] or guitarist fingertip tracking [13].

Their main approach for guitar neck detection is to initialize manually the first frame by selecting key points on the guitar fretboard and then by using optical flow (or other techniques) they detect the new positions of the key points in next frames. These points provide them proper information about the guitar fret positions. Hough transformation is used either to detect the guitar strings [7], the guitar fretboard area [6] or even fingertips [8]. The missing frets are interpolated either by using fret positions from previous frames, or by using known formulas that approximate the fret distances. The extraction of the skin pixels in most of the aforementioned works is done according to the Kovac model [14]. Fingertips are commonly localized by finding some local maxima on finger contours [7].

3.3 Our Approach

In this section we describe our approach to solve the GHIR problem. We provide a solution for each of the sub-problems the GHIR problem consists of. Also, we introduce the pick position detection problem, which corresponds to finding the string that the guitarist hits. The solution that we provide is for the case of a right-handed guitarist, which means that he presses the notes/frets with his left hand and he hits the strings using his right hand. However, the solution is general enough to work for the case of a left-handed guitarist with some minor modifications.

3.3.1 Fretboard Detection and Normalization

The first and the most crucial sub-problem that we must solve in GHIR is the detection of the fretboard area. It's the part of the image that has all the useful information that we need, in order to predict the notes that the guitarist plays. If for any reason this stage fails, then the rest of our method fails too. Without the correct information about the guitar area it's impossible to make correct predictions for the positions of the strings, frets, fingertips and guitar pick. In Figure 3.1 we see an example of a frame that is used as starting frame of our method.

As we can see in the starting frame, the camera is not totally focused on the guitar fretboard. There are other elements too, such as some part of the guitarist's face, the guitarist's t-shirt, the closet behind the guitarist, the light switch and some door's parts. All the aforementioned elements are noise in our data and they must

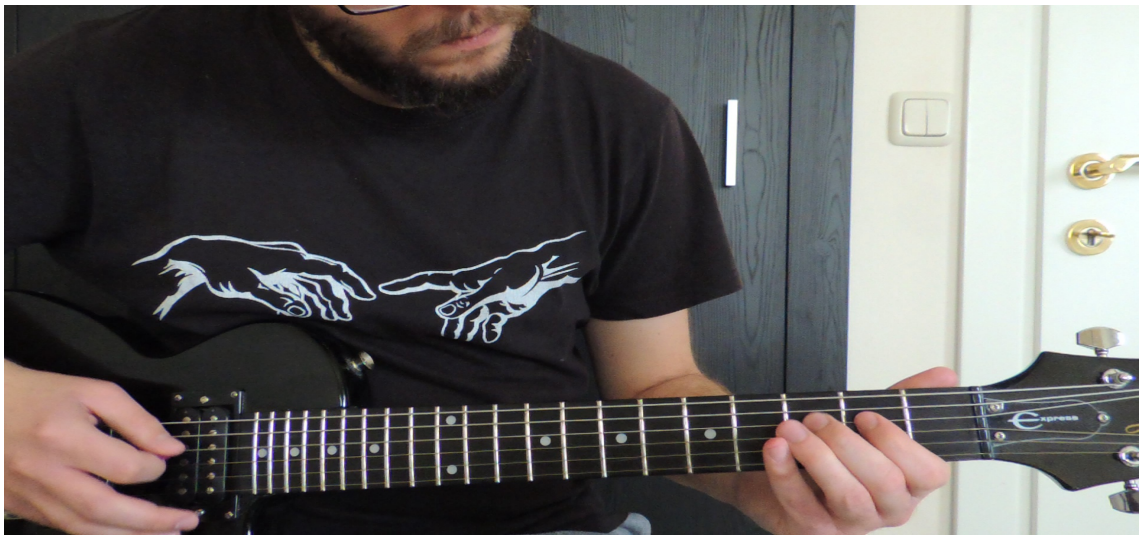

Figure 3.1: [GHIR 1/16] - Original Frame

be filtered out during our method. The guitar's bridge and headstock also make our task harder. They create edges that are noise for our detection and proper handling is required so they are filtered out too.

Our first goal is to detect the frets of the guitar. We start by computing the edges of the frame. The edges constitute the structure of the frame and hold all the precious information that we need to achieve our goal. We use the Sobel operator 2.2.1 to detect the horizontal edges of the frame. Canny edge detection could also be used, but in our case Sobel provided us without much effort the information that we needed. The horizontal edges are shown in Figure 3.2

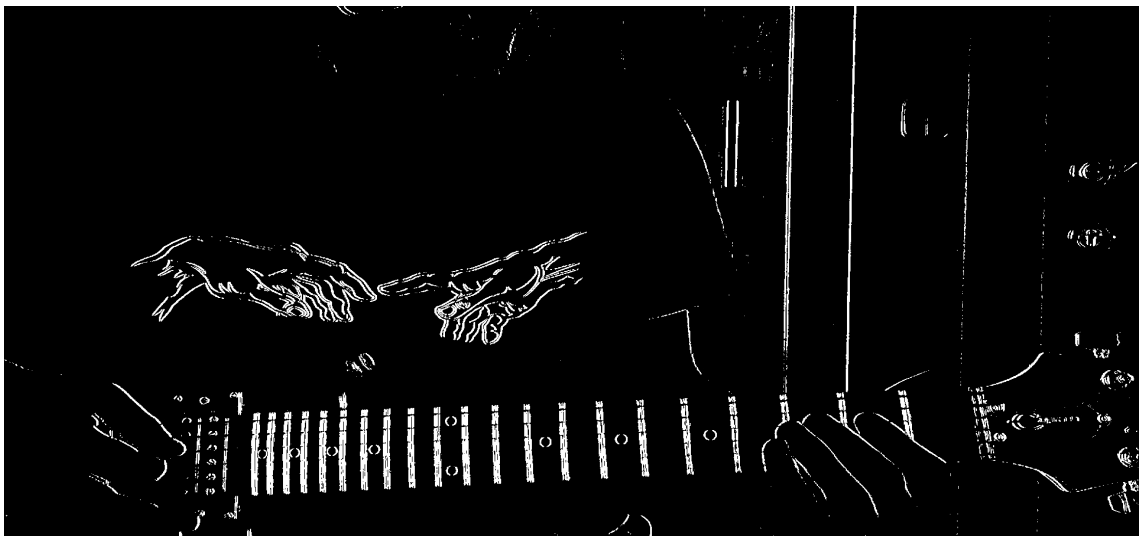

Figure 3.2: [GHIR 2/16] - Horizontal Edge Detection Using Sobel Filter

By looking the horizontal edges in Figure 3.2 we confirm our concern about the noise that is created from elements that are not part of the guitar fretboard. Also, we can see that some structure is shaped for the guitar frets. However, we need to make additional image processing to make the guitar frets more clear. For this purpose, we apply to the horizontal-edges image the morphological operations of closing and opening. The closing operations aim to fill the holes of the frets, while the opening operations aim to separate adjacent frets that have a fret marker between them. We must note that we use two different kernels during the morphological closing. The first kernel focuses in vertical closing and the second one focuses on horizontal closing. An example of the two kernels for *kernel_size = 3* is:

$$closing\_kernel_{vertical} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, closing\_kernel_{horizontal} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

As a final step of our processing we apply a smoothing filter. After the smoothing, all the pixels that have values greater than 0, are set to 255. The resulting thresholded image is shown in Figure 3.3.

Figure 3.3: [GHIR 3/16] - Horizontal Edges After Performing Morphological Operations and Smoothing

After applying the morphological operations and smoothing filter, we can see that the frets are now fully shaped. In the specific example, we notice that there is a fret, that is deformed after it is merged with other edges. Nevertheless, that doesn't affect our method. For the next step, we detect the contours of the resulted image. We keep only the contours that have area between some boundaries. We try to filter out

contours that are not candidate frets, either because they have small area, or because they have large area. The resulting contours are shown in Figure 3.4.

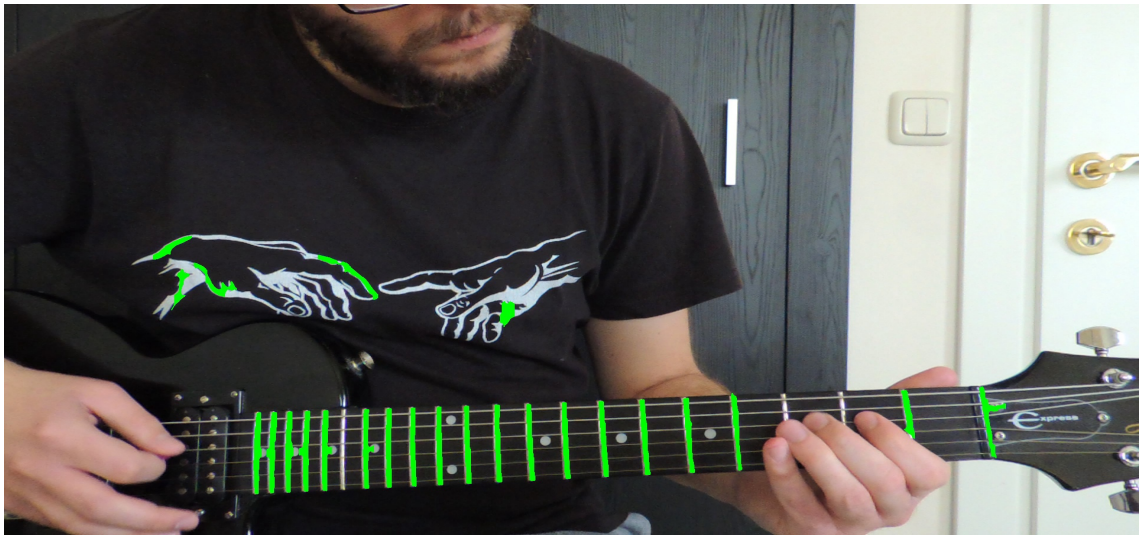

Figure 3.4: [GHIR 4/16] - Detected Contours of Horizontal Edges

We are now one step closer to detect the guitar fretboard area. The frets are shaped as rectangles and their edge points of the shorter sides, define the two lines that we need to detect. To localize these points, we find for each contour (1) the point that has the minimum y-coordinate and (2) the point that has the maximum y-coordinate. The detected points along with their corresponding contours are shown in Figure 3.5. For the rest of this work, we will refer to red-colored points of Figure 3.5 as the *top_points* of the candidate frets, while the blue-colored points will be referred as the *bot_points* of the candidate frets.

Now that we have the position of the *top_points* and the *bot_points* of the candidate frets, the next step is to detect the two lines that intersect the *top_points* and the *bot_points* of the real frets. The process goes as follows:

1. We sort the list of points based on their x-coordinate. Points with smaller x-coordinate are first on the list.

2. We create candidate lines by using pairs of the sorted points. Each point creates candidate lines only with points that have higher x-coordinate. For lower computational cost, we define a minimum index offset between the two points. Two points with greater distance probably define a better candidate line than two points that have smaller distance. We group the two candidate lines that

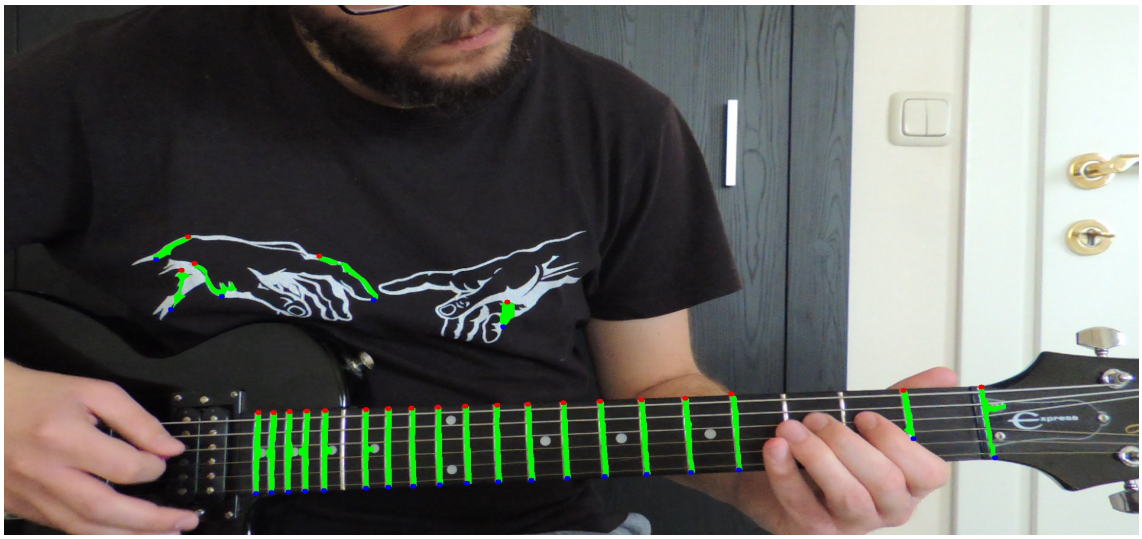

Figure 3.5: [GHIR 5/16] - Detected Contours with top and bot points

are created from a pair of frets. One line is created from the *top_points* of the two frets and one line is created from the *bot_points* of the two frets.

3. For each candidate line we compute it's score. The score of a candidate line is defined as the number of the points that intersects. We are not looking for a perfect intersection. We consider that a point (x_p, y_p) belongs to a line with equation $y = mx+b$ if $|(mx_p+b)-y_p| \leq error\_tolerance$, where $error\_tolerance$ is a small number (e.g. 3). For a robust implementation, we take into consideration the edge points of a fret, only if both the *top_point* and *bot_point* belong to the corresponding lines.

4. We take the points of the lines with the maximum score and we perform Linear Regression analysis. The resulted lines constitute the two lines that define the freatboard area.

The two detected lines, along with the best *top_points* and *bot_points* are shown in Figure 3.6. The two lines that define the fretboard area, are now detected. We set the equation of the line that intersects the *top_points* as $y = m_1x + b_1$ and the equation of the line that intersects the *bot_points* as $y = m_2x + b_2$.

For visualization purposes and for easier processing we apply a perspective transformation to our frame using the following four points:

- $top\_left = (0, b_1)$

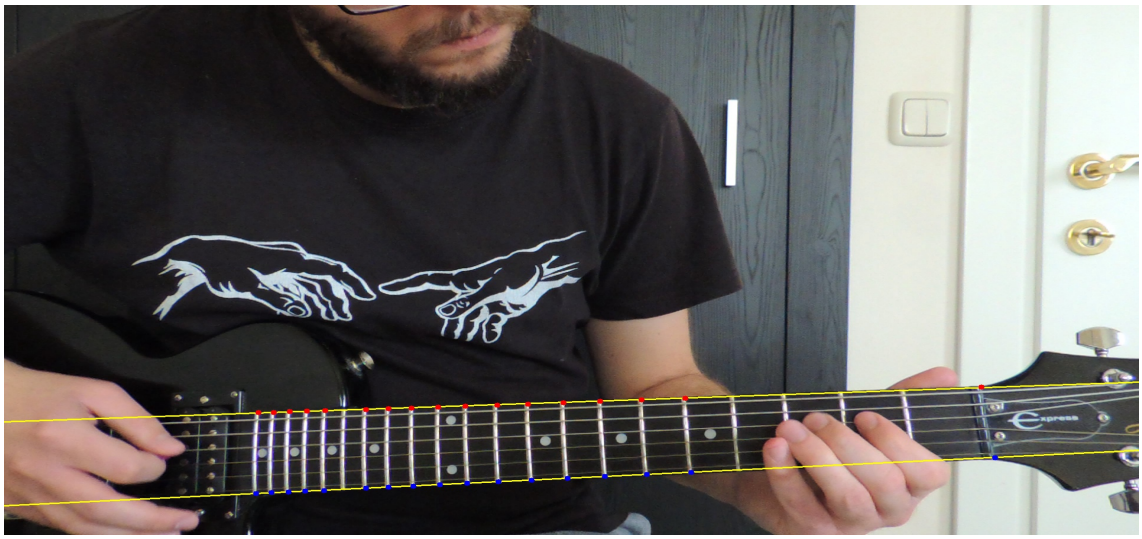

Figure 3.6: [GHIR 6/16] - Guitar Fretboard Detected Area

- $top\_right = (frame\_width, m_1 frame\_width + b_1)$
- $bot\_left = (0, b_2)$
- $bot\_right = (frame\_width, m_2 frame\_width + b_2)$

The resulted frame has width equals to the euclidean distance between the points (top_left, top_right) and it's height is equals to the euclidean distance between the points (top_left, bot_left). For the rest of our work, we will refer to this frame as the *normalized fretboard*. An example of the *normalized fretboard* is shown in Figure 3.7.

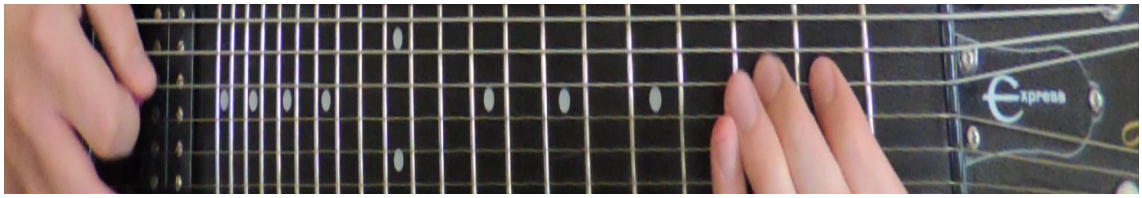

Figure 3.7: [GHIR 7/16] - Normalized Fretboard

3.3.2 Finger Detection and Localization

In this subsection we describe the method, that we use to localize the fingertips of the guitarist's left hand. To be able to extract this information, we must first find the pixels that correspond to skin pixels. In other words we must segment the normalized

freatboard into two clusters (1) the pixels that constitute the guitar part and (2) the pixels that constitute the guitarist's hands.

3.3.2.1 Skin Segmentation

Before proceeding to any kind of image segmentation of the normalized fretboard, we remove as many edges as possible from the image. This step has two benefits (1) it removes pixels which are not skin pixels, (2) in most of the cases it separates the fingers. Overlapping fingers can lead to poor fingertip detection performance. We use the edges of the normalized fretboard, that result after applying the Sobel operator in both vertical and horizontal directions. We use a very low threshold for the edges and we also apply some morphological operations and smoothing filtering, in order to make the resulting edges thicker. An example output of this process is shown in Figure 3.8.

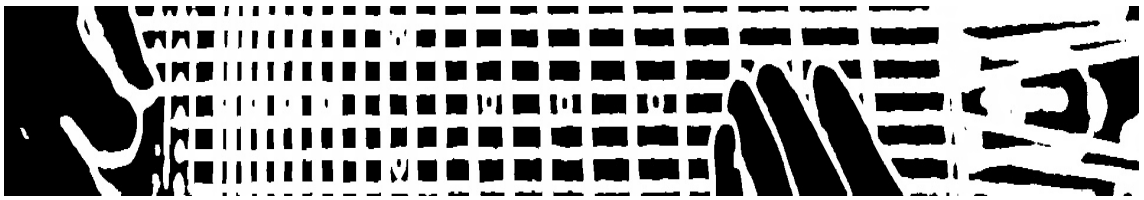

Figure 3.8: [GHIR 8/16] - Normalized Fretboard's Edges for Removal

We are interested only for the original pixel values that have a zero-value in the thresholded image of Figure 3.8. We proceed the image segmentation by using the K-means algorithm (with $k = 2$) to the pixels of interest. Hopefully, all the non-skin pixels are filtered out and the rest of the pixels are categorized into the pixels that represent the color of the guitar and the pixels that represent the guitarist's hands. The center of each cluster is a vector of size=3 and holds the information of the average R, G, B values of the pixels that belong to that cluster. As the cluster that represents the skin pixels, we select the one that has the higher variance between the center's values. The segmentation result is shown in Figure 3.9.

We notice in Figure 3.9 that some fret markers are also categorized as skin. We easily filter them out by removing the contours that have a small area.

The next important task is to distinguish between the two hands and separate the detected contours into the *right_hand_contours* and *left_hand_contours*. We group the contours based on their minimum *x-coordinate*. In the Figure 3.10 we color the

Figure 3.9: [GHIR 9/16] - Skin Segmentation

segmented right hand using blue color and the segmented left hand using magenta color.

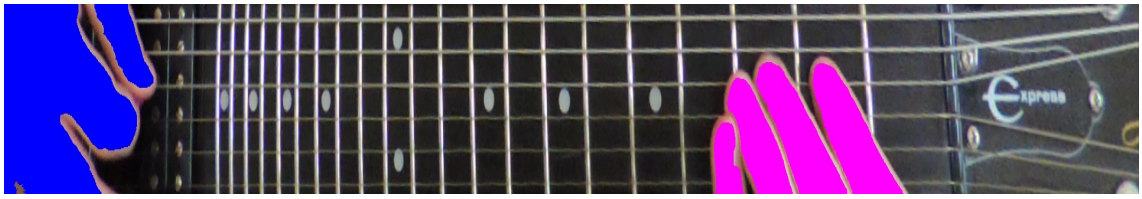

Figure 3.10: [GHIR 10/16] - Left and Right Hand Segmentation

The contour grouping can lead to *left_hand_contours* that in number can be greater than 4. Four is the maximum number of contours, that the guitarist's left hand can consist of (one contour per finger). In the case where we have more detected *left_hand_contours* than expected, we keep the four contours with the maximum area.

3.3.2.2 Left-Hand's Fingertips Localization

At the current stage we have the contours that represent the fingers of the guitarist's left hand. Our next goal is to detect the fingertips. To take advantage of the finger's curvature, we attempt to localize the fingertips by using the second derivatives of the *y-coordinates*. Specifically for each finger contour we do the following:

1. Create a function using the *y-coordinates* of the contour points. We start traversing the contour path from the bottom left point and we move to points with smaller *y-coordinate*. We take the *y-coordinate* for each point until we reach to the bottom right point.

2. We compute the second derivative of the *y-coordinates*. We consider as candidate fingertips, the points where the sign of the second derivative is negative, while the sign of the first derivative is positive.

3. We group adjacent candidate fingertips.

After the end of the above process, we end up with groups of candidate fingertips. We select the four groups with the higher number of candidate fingertips. As the final fingertips we take the mean values of these groups of points. The extracted fingertips are shown in Figure 3.11.

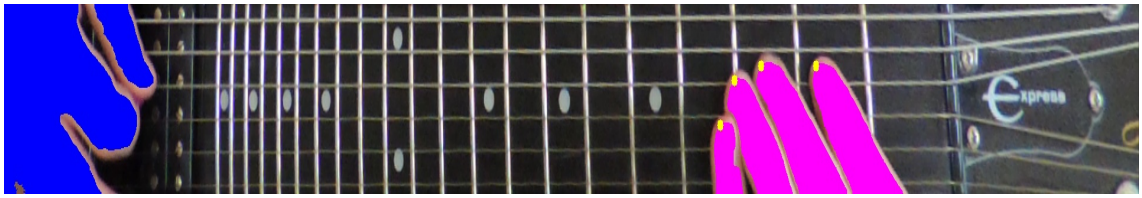

Figure 3.11: [GHIR 11/15] - Localized Fingertips

3.3.2.3 Guitar Pick Localization

In this subsection we introduce the problem of the pick position prediction. When a guitarist holds the guitar pick, the structure of his right hand and specifically his thumb, creates a curve. The hand picking techniques may differ from one guitarist to another, but in most of the cases our assumption is true. Hence, our objective is to traverse the contour path of the right hand and find the first local maxima of the *x-coordinates*.

Similarly to the case of the left hand, the right hand can also consist of multiple contours. The edge removal that was applied in a previous step of our method, seems to also separate the right hand of the guitarist into multiple contours. As a first step we select the contour that has as a starting *y-coordinate* the 0 and is located rightmost compared to the rest contours that also start from the top of the normalized fretboard. Finally, we traverse the contour path starting from the contour point *(max_x_coordinate, 0)* (i.e the rightmost contour point where it's y-coordinate is 0). and we move downwards until the first local maxima of the x-coordinate. This point is our prediction for the guitar pick position. The detected guitar pick position for our example is shown in Figure 3.12.

3.3.3 String Detection

We approach the string detection problem as simple as possible. We detect in the normalized fretboard, the vertical edges using the Sobel operator. Then, we apply

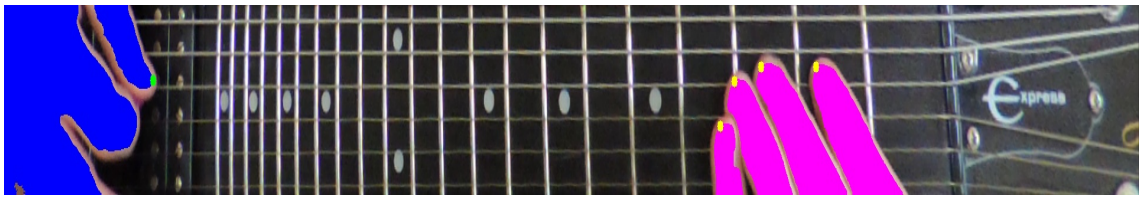

Figure 3.12: [GHIR 12/16] - Detected Guitar Pick Position

some morphological operations to fill the horizontal gaps of the strings. An example of that edge detection is shown in Figure 3.13.

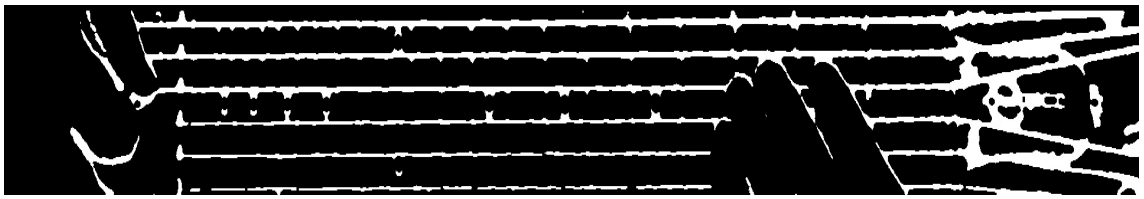

Figure 3.13: [GHIR 13/16] - Normalized Fretboard's Vertical Edges

We proceed by computing for each row, the percent of white pixels that it has. If that percent is above a defined threshold then we consider that row as a candidate string. After, finding all the candidate strings, we merge the consecutive adjacent candidate strings. We end up with a list of y-coordinates that represent the positions of the guitar strings. If the number of detected strings is greater than 6, we merge the closest strings until their number becomes 6. If the number of the detected strings is less than 6, then we fill the missing strings based on the existing string positions and distances. The detected strings in our example are shown in Figure 3.14.

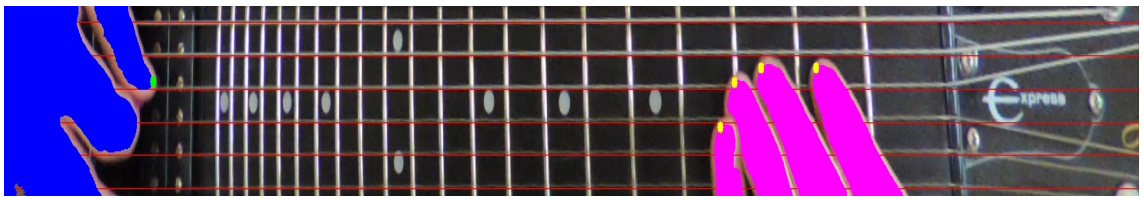

Figure 3.14: [GHIR 14/16] - Detected Guitar Strings

3.3.4 Fret Localization

The final step is to localize the guitar frets. First, we apply the perspective transformation that we already used in Subsection 3.3.1 to the original horizontal edge image that is shown in Figure 3.4.

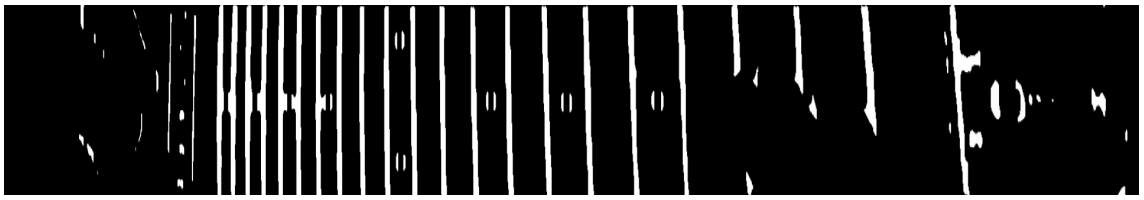

Figure 3.15: [GHIR 15/16] - Normalized Fretboard's Horizontal Edges

Then using the edge image of Figure 3.15 we do following steps to get the fret positions:

1. Find the contours and keep only the ones that have area greater than a threshold.

2. Filter out the contours that have smaller x-coordinate than the maximum x-coordinate of the right hand contours.

3. Filter out the contours that have higher x-coordinate than the maximum x-coordinate of the left hand contours.

4. Compute the *top_points* and *bot_points* of the filtered contours as in Subsection 3.3.1.

5. Filter out the contours where $top\_point.y > tol$ or $top\_point.y < width - tol$, where the tol is a small number (e.g 3) and $width$ is the width of the normalized fretboard image. In other words we keep only the contours that have as a starting y-coordinate the 0 and as an ending y-coordinate the width of the image.

6. Start from the leftmost fret position and by moving to the right, interpolate missing fret positions until we create a point that has higher x-coordinate than the maximum x-coordinate of the left hand contours. We take into account the edge image of Figure 3.15 to fix the positions of the created frets.

An example output of the process that we just described, is shown in Figure 3.16.

3.3.5 Tab Prediction

In previous subsections we detected and gathered the information that we need to be able to make a prediction. To predict the string that the guitarist hits, is quite

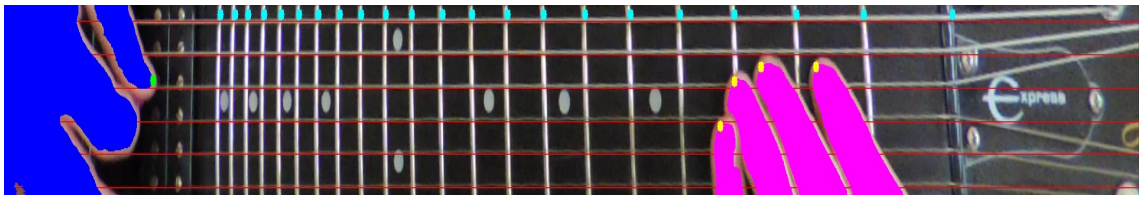

Figure 3.16: [GHIR 16/16] - Detected Frets

straightforward. We set as the *PRED_STRING*, the string that has the minimum distance from the pick position that we found in Subsection 3.3.2.3. The distance is calculated using only the y-coordinates.

To predict the notes that the guitarist presses with his fingers we do the following for each fingertip:

1. We compare the x-coordinate of the fingertip with the frets' x-coordinates and we find at which bin it belongs. The index of the bin is the predicted fret for this fingertip.

2. We compare the y-coordinate of the fingertip with the strings y-coordinates and we find at which bin it belongs. The index of the bin is the predicted string for this fingertip.

The guitar that we use for our example has six (6) strings. Hence, our method returns a list that has a size of six and each element contains fret predictions. We provide a more detailed explanation about the representation of our method's output in Subsection 4.1.

Chapter 4

Experiments

4.1 Dataset

For our experiments we needed video recordings from a guitarist that played some notes and chords on guitar. Unfortunately, there were no available datasets from previous works, so we had to create our own. For the purposes of this work, we created four (4) video recordings using a *NIKON COOLPIX P510* digital camera. In two of them we played chords and progressions that were previously tested in other works, to be able to make some kind of comparison with our approach. For the last video recording, we chose a guitar riff from a favorite metal song. Specifically, our dataset contains the following video recordings with a resolution of *1080p (1920x1080)*:

- **C Major Progression** that consists of the chords C, Am, Dm and G7 (see Figure 4.1 for tabs).
- **C Major Scale** (see Figure 4.2 for tabs[1]).

[1] https://www.guitarcommand.com/

- **F Major Scale** (see Figure 4.3 for tabs[2]).
- **Kissing the Shadows** [3] guitar riff from the metal band *Children of Bodom* (see Figure 4.4 for tabs[4]).

Since the dataset was created from scratch, we also needed to annotate it. In our case, in order to create a proper annotation we must assign for each frame of interest, a pair (*GT_NUM_STRING*, *GT_FRETS*), where *GT_NUM_STRING* is the number of the string that guitarist hits, while *GT_FRETS* is a list of numbers with length six (6) and corresponds to the frets that guitarist presses with his fingers. Note that as frames of interest we refer to the frames where the guitarist actually plays the guitar (either by strumming or individual string picking). All the frames, where the guitarist moves his left hand -to place the fingers to the correct position- are not taken into account.

For our annotation, we assigned the number 0 to the thickest string and the number 5 to the thinnest string. Also, the fret numbering starts from the right part of guitar (bridge) to the left part (headstock). Note that the aforementioned number assignment is the reversed compared to how we described the string and fret numbering in previous chapter. An example of an annotation pair is (0, [18, 19, -1, -1, -1, 20]), which is interpreted as: (1) The guitarist hits the thickest string (2) The guitarist presses with his fingers the third (3rd) fret of the sixth (6th) string, the second (2nd) fret of the fifth (5th) string and the first (1st) fret of the first (1st) string. The positions where the number is -1 are ignored, since the corresponding strings are not used to form the current chord or note that is played from the guitarist.

For the case of *C Major Progression* we set the *GT_NUM_STRING* to -1. Our approach for the estimation of guitar pick position works only for single-string picking, while in the case of *C Major Progression*, we have string strumming.

In the rest of the recordings we had single-string picking, so we needed to assign a string number to each of the annotation pairs. We processed all the frames manually and we detected the exact frames where the guitarist hits -with his right hand- one of the six (6) strings. For each of these frames we assigned the single fret number, which corresponds to the fret that the guitarist presses with his left hand. For our experiments we include only the frames where the guitarist hits a string.

[2] https://www.guitarcommand.com/

[3] https://www.youtube.com/watch?v=AW_wgKGoEmI

[4] https://www.ultimate-guitar.com/

Table 4.1: Dataset size per recording

Recording	Number of Frames
C Chord Progression	2052
C Major Scale	308
F Major Scale	225
Kissing the Shadows Riff	106

After the annotation of the video recordings, we ended up with a specific number of frames which are shown in the Table 4.1.

The dataset seems to have unbalanced sizes. We have to note that the annotation of a frame interval, where the guitarist strums the strings is much faster and easier. On the other hand, the single-string picking annotation is a time consuming task, since each frame needs to be processed individually. We decided to follow this approach for the single-string evaluation, since it's the most low-level testing we could do. An alternative approach would be to set a frame interval around the picking frame but that wouldn't give us exactly the information that we need to check the performance of our string predictor.

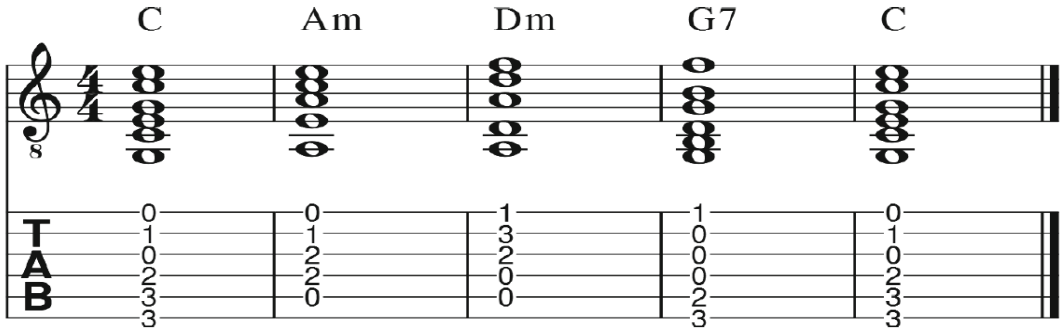

Figure 4.1: C Major Progression Tabs

4.2 Evaluation metrics

As we described in previous chapter, our predictor returns a pair (*PRED_STRING*, *PRED_FRETS*). To evaluate our prediction we use the corresponding groundtruth pair (*GT_NUM_STRING*, *GT_FRETS*).

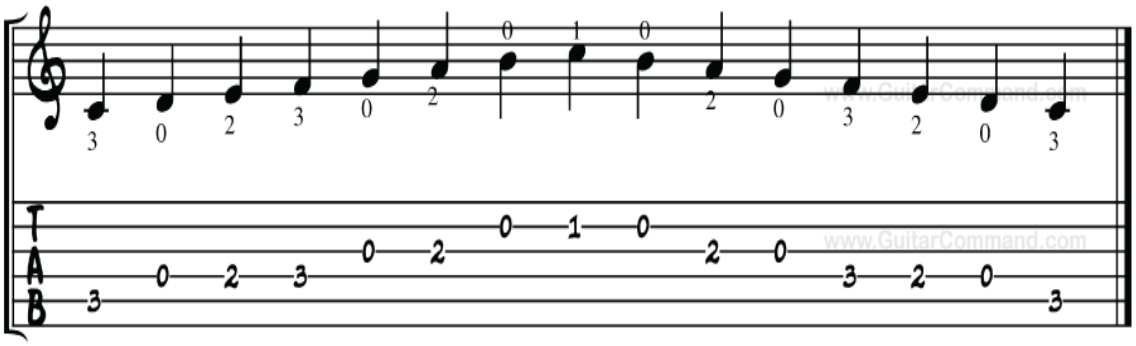

Figure 4.2: C Major Scale Tabs

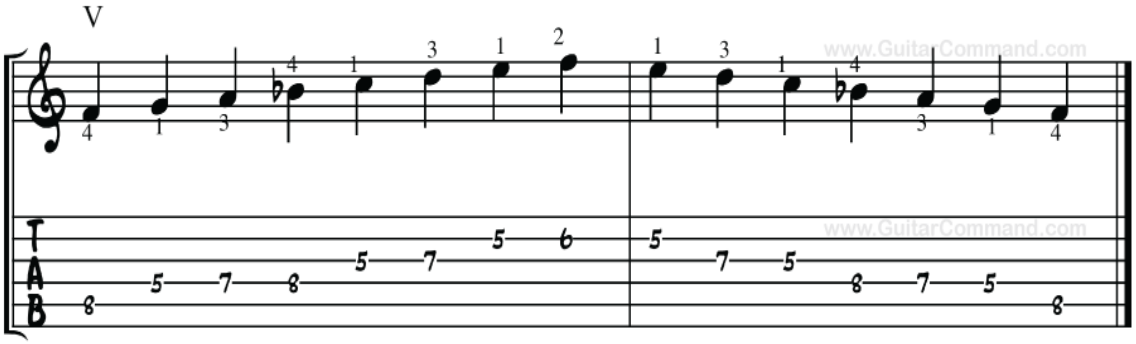

Figure 4.3: F Major Scale Tabs

4.2.1 Per Fret

We evaluate the fret-prediction as follows: for each groundtruth fret in the *GT_FRETS*, we check if it is included in the list of predicted frets (*PRED_FRETS*) for the corresponding string. Each string is evaluated separately. For a better understanding we include an example 4.2.1.

Example 4.2.1. Assume that for a given frame the list for the groundtruth frets is *GT_FRETS*=[18, 18, 19, -1, 20, -1] and the predicted tabs are *PRED_FRETS*=[[18], [18], [19, 20], [], [], []]. The result of the comparison is [True, True, True, None, False, None], which means that we predicted correctly the pressed frets for the strings *4*, *5* and *6*, but we made an incorrect prediction for the fret of the second (*2nd*) string. We ignore the string indexes with *None* values, because they are not part of the groundtruth.

We are unable to predict whether the guitarist actually presses a note, so cases such as *C Major Scale* 4.2, where the fret groundtruth value for a string is *0*, are not

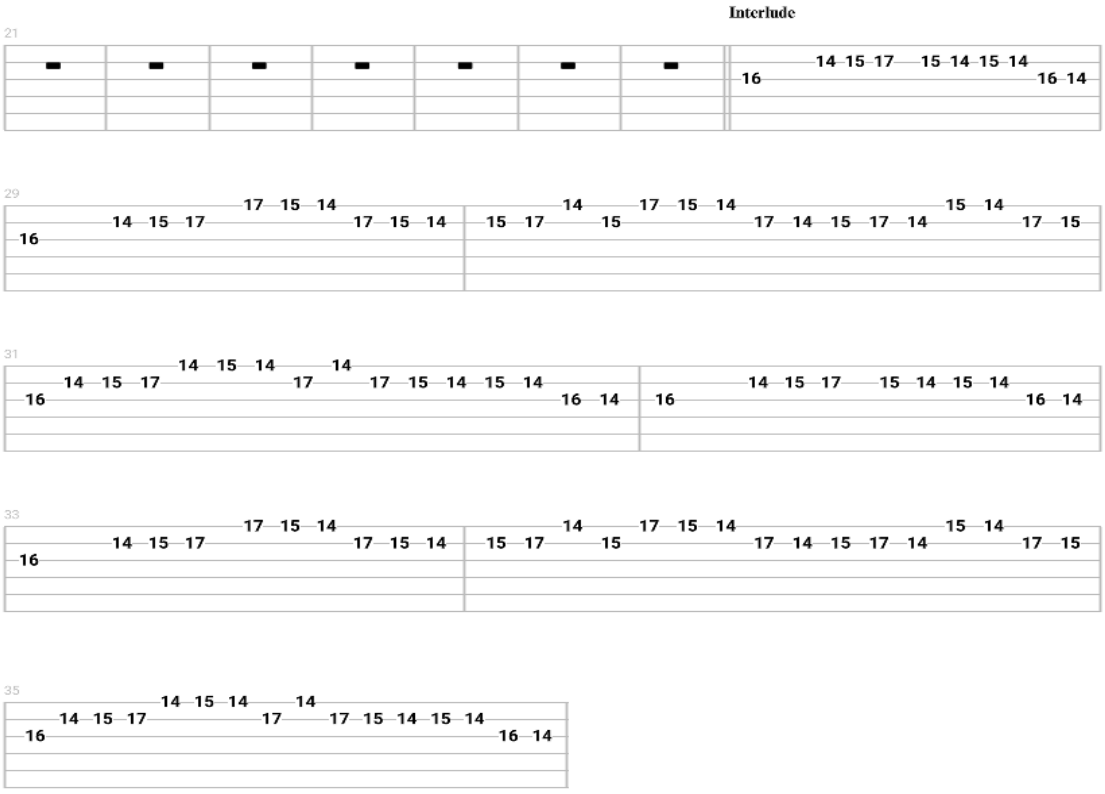

Figure 4.4: Children of Bodom - Kissing the Shadows Riff Tabs

considered to our evaluation.

4.2.2 Per String

The task of string-prediction evaluation is more straightforward. For a given frame, the string prediction is correct if *GT_NUM_STRING* and *PRED_STRING* are equals. Otherwise, the prediction is incorrect.

4.3 Experiments Configuration

To perform our experiments we must configure some parameters that are mainly related to morphological operations or filtering unnecessary information from our data. More specifically the list of parameters that we use, along with their values are described below:

- **SOBEL_X_THRESHOLD_PERCENT** (Default: 0.12): it is used to threshold

the image of horizontal edges (G_x) that results after applying the sobel operator to the original frame. We consider as edges the pixels that have value >= SOBEL_X_THRESHOLD_PERCENT*max(G_x). Note that G_x has the absolute values of horizontal derivative approximation.

- **GX_V_CLOSE_KERNEL_SIZE** (Default: 5): the size of the kernel that is used for vertical morphological closing to G_x.
- **GX_V_CLOSE_ITERATIONS** (Default: 4): the number of vertical morphological closing iterations that are applied to G_x.
- **GX_H_CLOSE_KERNEL_SIZE** (Default: 4): the size of the kernel that is used for horizontal morphological closing to G_x.
- **GX_H_CLOSE_ITERATIONS** (Default: 1): the number of horizontal morphological closing iterations that are applied to G_x.
- **GX_V_OPEN_KERNEL_SIZE** (Default: 5): the size of the kernel that is used for vertical morphological opening to G_x.
- **GX_V_OPEN_ITERATIONS** (Default: 1): the number of vertical morphological opening iterations that are applied to G_x.
- **GX_SMOOTH_KERNEL_SIZE** (Default: 6): the size of the smoothing kernel that is applied to G_x after the morphological operations. After the smoothing we consider as edges all the pixels that have value > 0.
- **GX_MIN_CONTOUR_AREA** (Default: 600): the minimum area of the G_x contours, that we use for the rest of our computations.
- **GX_MAX_CONTOUR_AREA** (Default: 1500): the maximum area of the G_x contours, that we use for the rest of our computations.
- **MAX_POINT_Y_ERROR** (Default: 3): the maximum error a contour point can have in order to be included in a candidate fretboard line.
- **MIN_POINT_INDEX_OFFSET** (Default 10): the minimum index distance between two sorted contour points that are used to create a candidate fretboard line.

- **SOBEL_Y_THRESHOLD_PERCENT** (Default 0.1): it is used to threshold the image of vertical edges (G_y) that results after applying the sobel operator to the original frame. We consider as edges the pixels that have value >= THRESHOLD, where THRESHOLD = SOBEL_Y_THRESHOLD_PERCENT*max(G_y). Note that G_y has the absolute values of vertical derivative approximation.

- **PGY_V_CLOSE_KERNEL_SIZE** (Default: 2): the size of the kernel that is used for vertical morphological closing to normalized G_y (pG_y).

- **PGY_V_CLOSE_ITERATIONS** (Default: 2): the number of vertical morphological closing iterations that are applied to pG_y.

- **PGY_H_CLOSE_KERNEL_SIZE** (Default: 5): the size of the kernel that is used for horizontal morphological closing to pG_y.

- **PGY_H_CLOSE_ITERATIONS** (Default: 2): the number of horizontal morphological closing iterations that are applied to pG_y.

- **PGY_SMOOTH_KERNEL_SIZE** (Default: 6): the size of the smoothing kernel that is applied to pG_y after the morphological operations. After the smoothing we consider as edges all the pixels that have value > 0.

- **MIN_FRET_AREA** (Default: 200): the minimum area of the contours of the normalized G_x (pG_x), to be considered as candidate frets.

- **STRING_MIN_PERCENT_FILLED** (Default: 0.3) the minimum percent of white pixels that are required from a vertical line of pG_y, to be considered as a candidate string.

- **PICK_POS_NUM_NEIGHBOURS** (Default: 15): the number of neighbors that are used to detect the x local maxima of the pick position.

As we can easily see, we have quite a number of configuration parameters. Although, it took no time and effort to find some good configuration, such that the experiments to be executed flawlessly. Probably, these configuration parameters could be initialized automatically using the following process: during the beginning of the video recording the guitarist presses specific notes that don't lead to finger overlapping. Then our method tries to find the best set of parameters that results to a correct prediction of the notes that the guitarist presses.

4.4 Numerical Results

Tables 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, contain the numerical results of our experiments. We also provide separate evaluations for the chords C, Am, Dm and G7 that form the C Major Progression.

Table 4.2: Fret-Prediction Evaluation for C Chord (528 frames)

Fret Prediction	String					
	1	2	3	4	5	6
Errors	—	63	—	264	306	469
Accuracy	—	88%	—	50%	42%	11%
Overall Accuracy: 48%						

Table 4.3: Fret-Prediction Evaluation for Am Chord (496 frames)

Fret Prediction	String					
	1	2	3	4	5	6
Errors	—	24	14	19	—	—
Accuracy	—	95%	97%	96%	—	—
Overall Accuracy: 96%						

Table 4.4: Fret-Prediction Evaluation for Dm Chord (526 frames)

Fret Prediction	String					
	1	2	3	4	5	6
Errors	21	47	21	—	—	—
Accuracy	96%	91%	96%	—	—	—
Overall Accuracy: 94%						

Table 4.5: Fret-Prediction Evaluation for G7 Chord (502 frames)

Fret Prediction	String					
	1	2	3	4	5	6
Errors	35	—	—	—	35	356
Accuracy	93%	—	—	—	93%	29%
Overall Accuracy: 72%						

Table 4.6: Fret-Prediction Evaluation for C Major Progression

Fret Prediction	String					
	1	2	3	4	5	6
Errors	56	134	35	283	341	825
Frames Compared	1028	1550	1022	1024	1030	1030
Accuracy	93%	91%	97%	72%	67%	20%
Overall Accuracy: 73%						

Table 4.7: Fret-Prediction Evaluation for C Major Scale

Fret Prediction	String					
	1	2	3	4	5	6
Errors	—	0	0	1	1	—
Frames Compared	—	22	44	88	22	—
Accuracy	—	100%	100%	99%	91%	—
Overall Accuracy: 98%						

Table 4.8: String-Prediction Evaluation for C Major Scale

String Prediction	String					
	1	2	3	4	5	6
Errors	—	0	1	1	5	—
Frames Compared	—	22	44	88	22	—
Accuracy	—	100%	98%	99%	77%	—
Overall Accuracy: 96%						

Table 4.9: Fret-Prediction Evaluation for F Major Scale

Fret Prediction	String					
	1	2	3	4	5	6
Errors	—	0	6	15	4	—
Frames Compared	—	45	60	90	30	—
Accuracy	—	100%	90%	83%	87%	—
Overall Accuracy: 89%						

Table 4.10: String-Prediction Evaluation for F Major Scale

String Prediction	String					
	1	2	3	4	5	6
Errors	—	4	15	5	0	—
Frames Compared	—	45	60	90	30	—
Accuracy	—	91%	75%	94%	100%	—
Overall Accuracy: 89%						

Table 4.11: Fret-Prediction Evaluation for Kissing the Shadows Riff

Fret Prediction	String					
	1	2	3	4	5	6
Errors	15	14	5	—	—	—
Frames Compared	30	60	16	—	—	—
Accuracy	80%	77%	69%	—	—	—
Overall Accuracy: 76%						

Table 4.12: String-Prediction Evaluation for Kissing the Shadows Riff

String Prediction	String					
	1	2	3	4	5	6
Errors	4	2	0	—	—	—
Frames Compared	30	60	16	—	—	—
Accuracy	87%	97%	100%	—	—	—
Overall Accuracy: 94%						

Overall our method works well. We don't provide qualitative results of our detections but we have to note that the high accuracy of the fret and string detection is the key of the high performance of our method. Also, the fingertip localization is remarkable and contributes to the high accuracy. We didn't measure the aforementioned accuracies, but looking the output videos of our method that contains frames like 3.16, we can easily notice the robustness of our detection.

4.5 Comparison with similar works

We are not able to make a proper comparison with other works, since the datasets are not the same. Although, we created a dataset that tests similar cases and the numerical comparison can give us an idea about the performance of our method. We present the comparison in Table 4.13.

Our method outperforms in both of the cases the Burns' method [8] and performs better than Duke's method [7] for the case of C Major Scale. We are sure that with a couple of improvements our method will totally outperform the rest of the methods. The advantage of our method is that we don't make approximations of the guitar fret positions. We use as much information as we can from the edges to find them. We approximate the fret positions only in the case where the hand covers the whole

Table 4.13: Comparison with previous methods.

Case	Scarr's method [8]	Duke's method [7]	Our method
C Major Progression	52%	**86%**	73%
C Major Scale	70%	76%	**98%**

fret. Another advantage of our method is that we separate the fingers before the localization of fingertips, which solves the finger overlapping problem.

4.6 Limitations and Error Cases

The performance of our method is very promising and at some cases achieves the maximum accuracy. However, our method has it's limitations and inevitably we have some error cases that currently are not handled properly. First of all, the main limitation of our method is the finger information that we lose, when we apply the perspective transformation.

For the case of the guitarist's right hand, this limitation may arise when he plays the 5th and the 6th string. Generally, when the guitarist plays the last three strings, his right hand is formed from two different contours. The thumb in most of the cases is separated from the rest of the hand. During the contour filtering process (based on their area), there is a chance that the thumb of the right hand is filtered out too. Without the contour path that corresponds to thumb, our prediction is not reliable, since we are searching the first local-maxima in the wrong contour path.

For the case of the guitarist's left hand, we can't handle the case where the guitarist presses the 6th string with his fingers. His fingertip is out of the normalized fretboard frame and our approach using derivatives to detect the fingertips won't work.

Our method has also it's limitations regarding the fingertip localization. After using the edge removal procedure, the fingers are separated and ideally each finger contributes with a single fingertip. Yet, that's not the case. In the current implementation, the groups of candidate fingertips -that are bigger in size- are taken into account for the computation of the final fingertips. This approach may result to multiple fingertips detection, per finger. Additionally, we have to note that the edge removal procedure in some cases creates structures in finger contours, that lead to a false fingertip detection.

Some error cases illustrations are shown in Figures 4.5, 4.6, 4.7, 4.8 and 4.9.

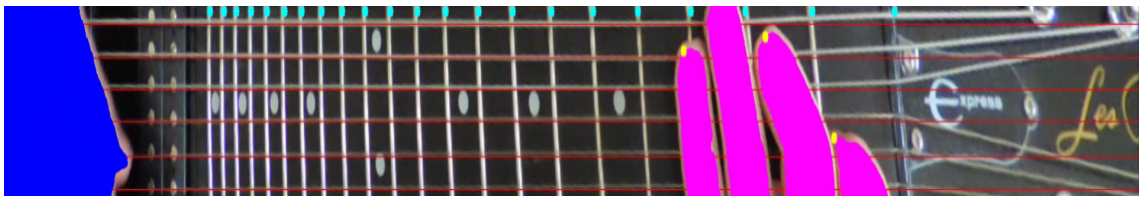

Figure 4.5: Limitation: Low accuracy to fingertips that press the 6th string.

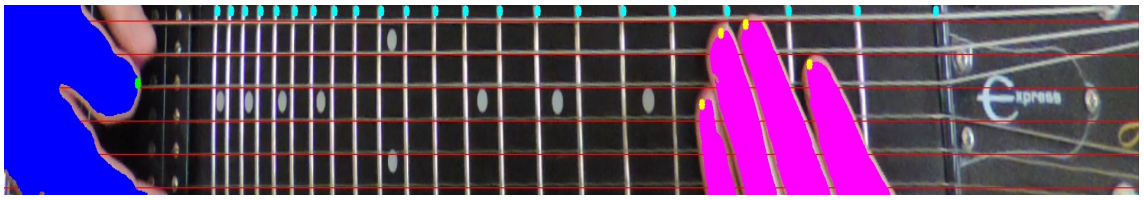

Figure 4.6: Error case: Right hand contour filtered out. Guitar pick detection fails.

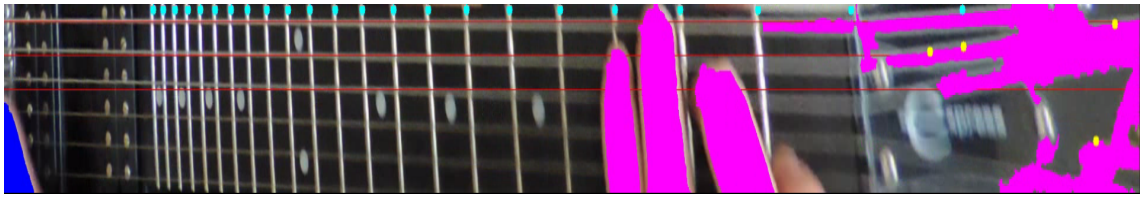

Figure 4.7: Error cases: (1) Poor fretboard area detection (2) False skin segmentation

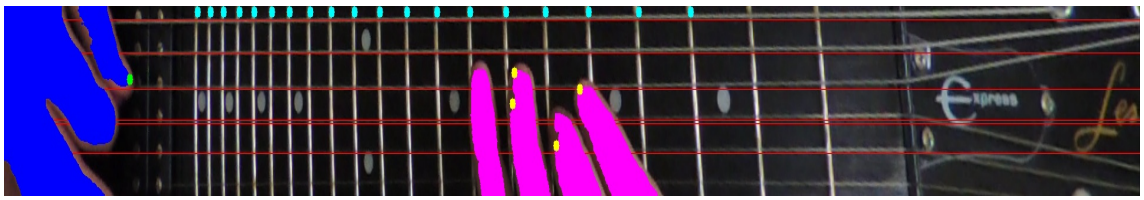

Figure 4.8: Error cases: (1) Multiple fingertip localization per finger (2) Duplicate string detection

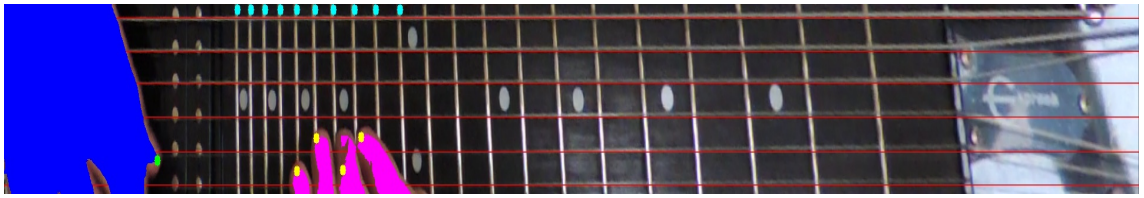

Figure 4.9: Error case: Shadow results to finger segmentation failure.

Chapter 5

Conclusions and Future Work

In this work we studied the problem of Guitarist Hand Information Retrieval which corresponds to a vision-based approach of the Automatic Music Transcription problem for the guitar instrument. We solved this problem by processing and extracting information from each frame independently and without the usage of any prior knowledge about the characteristics of the guitar. Also, we introduced the problem of pick position prediction, which gives us the information of the candidate string the guitarist hits. We evaluated our model on a dataset that we created and annotated. The experimental results show that our approach achieves high accuracy in both of fret and string prediction. Our method performs better in most of the cases compared to similar experiments of other works.

Although our approach performs quite well in our experiments, there are some issues that could be easily resolved:

- Duplicate detection of the same string.
- False skin detection of guitar areas, that results to false positive fingertips.
- Multiple fingertip localization that result from the same finger.

Finally, a list of possible extensions could be:

- Automatic setting of model's configuration by optimizing an objective function.
- Modification of the fingertip localization algorithm, so it can detect fingertips of the 6-th string with higher accuracy.

- Modification of the edge removal algorithm (during the phase of skin detection), in order to minimize the finger deformation, that results to false positive fingertips.
- Modification of the string detection algorithm, so it detects the strings as seams and not as horizontal lines.
- Use information from previous frames to improve the accuracy and also decrease computational cost.
- Execution speed optimization of the current implementation.

Bibliography

[1] A. M. Barbancho, A. Klapuri, L. J. Tardón, and I. Barbancho, "Automatic transcription of guitar chords and fingering from audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 915–921, 2011.

[2] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score-and instrument-related parameters." in *DAFx*, 2014, pp. 219–226.

[3] K. Yazawa, K. Itoyama, and H. G. Okuno, "Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3122–3126.

[4] G. Burlet and I. Fujinaga, "Robotaba guitar tablature transcription framework." in *ISMIR*, 2013, pp. 517–522.

[5] P. D. O'Grady and S. T. Rickard, "Automatic hexaphonic guitar transcription using non-negative constraints," 2009.

[6] 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*. IEEE, 2010, pp. 1–7.

[7] B. Duke and A. Salgian, "Guitar tablature generation using computer vision," in *Advances in Visual Computing: 14th International Symposium on Visual Computing, ISVC 2019, Lake Tahoe, NV, USA, October 7–9, 2019, Proceedings, Part II 14*. Springer, 2019, pp. 247–257.

[8] A.-M. Burns, "Computer vision methods for guitarist left-hand fingering recognition," 2006.

[9] Z. Wang and J. Ohya, "Tracking the guitarist's fingers as well as recognizing pressed chords from a video sequence," *Electronic Imaging*, vol. 2016, no. 15, pp. 1–6, 2016.

[10] G. Quested, R. Boyle, and K. Ng, "Polyphonic note tracking using multimodal retrieval of musical events," in *ICMC*, 2008.

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

[12] ——, "An accurate and robust algorithm for tracking guitar neck in 3d based on modified ransac homography," *Electronic Imaging*, vol. 2018, no. 18, pp. 460–1, 2018.

[13] C. Kerdvibulvech, H. Saito *et al.*, "Guitarist fingertip tracking by integrating a bayesian classifier into particle filters," *Advances in Human-Computer Interaction*, vol. 2008, 2008.

[14] J. Kovac, P. Peer, and F. Solina, *Human skin color clustering for face detection*. IEEE, 2003, vol. 2.

Author's Publications

P. Kouzouglidis, G. Sfikas and C. Nikou. Automatic video colorization using 3D conditional generative adversarial networks. LNCS Vol. 11844, pp.209-218. International Symposium on Visual Computing (ISVC'19), 7-9 October 2019, Lake Tahoe, Nevada, USA.

Short Biography

Panagiotis Kouzouglidis was born in Drama in 1994. He obtained a Diploma in Computer Science from the department of Computer Science and Engineering, University of Ioannina, Greece in 2019. He is currently a M.Sc. student in Data and Computer Systems Engineering at the same department, specialized in Data Science and Engineering. His research interests include Computer Vision and Machine Learning.