claVision: visual automatic piano music transcriptio

CLAVISION: VISUAL AUTOMATIC PIANO MUSIC TRANSCRIPTION

MOHAMMAD AKBARI
Bachelor of Science, Shahid Bahonar University of Shiraz, 2010

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

CLAVISION: VISUAL AUTOMATIC PIANO MUSIC TRANSCRIPTION

MOHAMMAD AKBARI

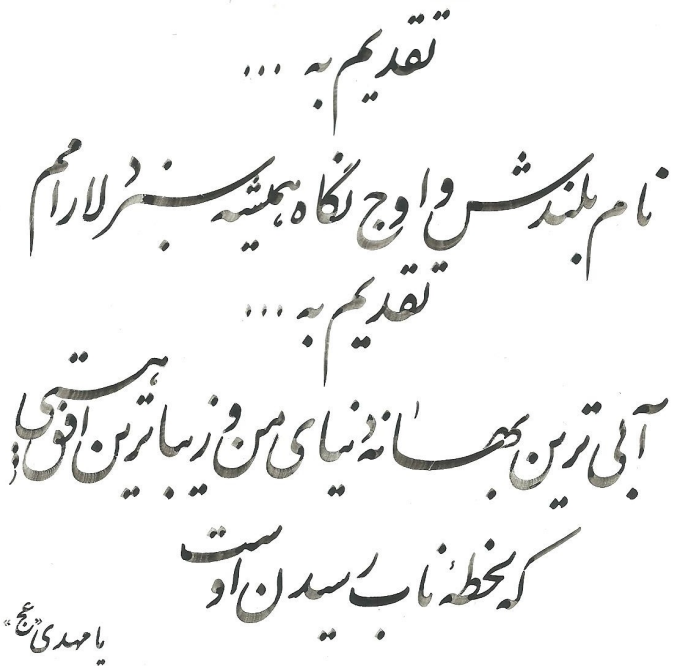
Date of Defense: December 11, 2014

Dr. Howard Cheng Supervisor	Associate Professor	Ph.D.
Dr. John Zhang Committee Member	Associate Professor	Ph.D.
Dr. Craig Coburn Committee Member	Associate Professor	Ph.D.
Dr. Hadi Kharaghani Chair, Thesis Examination Committee	Professor	Ph.D.

Dedication

To the greatness of his name and evergreen ken, which is always soothing to me...

To the bluest reason of my worldly life and the most beautiful horizons, which will be the pure moment of his arrival...



Abstract

One significant problem in the science of Musical Information Retrieval is Automatic Music Transcription, which is an automated conversion process from played music to a symbolic notation such as sheet music. Since the accuracy of previous audio-based transcription systems is not satisfactory, an innovative visual-based automatic music transcription system named claVision is proposed to perform piano music transcription. Instead of processing the music audio, the system performs the transcription only from the video performance captured by a camera mounted over the piano keyboard. claVision can be used as a transcription tool, but it also has other applications such as music education. The software has a very high accuracy (over 95%) and a very low latency (less than 6.6 ms) in real-time music transcription, even under different illumination conditions. This technology can also be used for other musical keyboard instruments. claVision is the winner of the 2014 Microsoft Imagine Cup Competition in the category of innovation in both Canadian national finals and world semifinals. As one of the top 11 teams in the world, claVision advanced to World Finals in Seattle to be demonstrated at the University of Washington, Microsoft headquarters, and the Museum of History & Industry.

Acknowledgments

I am using this opportunity to express my gratitude to everyone who supported me throughout this thesis. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work.

I would like to express the deepest appreciation and thanks to my supervisor Dr. Howard Cheng for all his kindness, compassion, support, efforts, and engagement through the process of this master thesis. He continually and persuasively conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. He has been one of the best and the most tremendous teachers and mentors that I have ever had in my life. Without his supervision and constant help, this thesis would not have been possible. From bottom of my heart, I honestly appreciate you.

I also want to thank my committee members, Dr. John Zhang and Dr. Craig Coburn, for their useful comments and remarks during my master studies. In addition, a thank you to Dr. Hadi Kharaghani for serving as my thesis examination committee chair. I would also like to thank Dr. Wendy Osborn and Dr. Kevin Grant for their instructions in my master courses as well as their great support for my successful Ph.D. application to SFU. Furthermore, I would like to express my appreciation to Dr. Deanna Oye for all her helpful remarks and advice on the musical aspect of this project, especially during the Imagine Cup competition.

Moreover, I like to thank the companies and organizations including Microsoft, Alberta Innovates Technology Future (AITF), MindFuel, Beakerhead, and Calgary Philharmonic Orchestra, which really helped me in my achievement in the Imagine Cup Competition as well as the business side of my idea. A special thanks to Susan Ibach, Bill Buxton, Kate

Gregory, Etienne Tremblay, and Mitch Barnett from Microsoft, Bill Halley, Jennifer Hill, and Adam Brown from AITF, Angie Chiang and James Landsburg from MindFuel, and Morgan Guo from University of Lethbridge.

I also express my gratitude to all of my good friends who supported and incented me to strive towards my goal. A warm and special appreciation to Maryam Bahrami, Mahsa Miri, Mohadeseh Majdi Yazdi, Farshad Barahimi, Hossein Naseri, and Robab Hashemi for all their help, time, and engagement. I also want to acknowledge my friends, Saeed Abbasi, Dr. James Parks, Seyed Hossein Hosseini, Farzad Aryan, Fariha Naz, Manoj Kumar, and Seyed Majid Shahabi.

The deepest thanks and appreciation go to my family. Words cannot express how grateful I am to them for all of the sacrifices that they have made on my behalf. Your prayer for me was what sustained me thus far.

Thank you very much,
-Mohammad

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The growth of digital music has considerably increased the availability of content in the last few years. Digital Music is a method of representing analog sound as numerical values in a computer. In order to access, organize, analyze this vast amount of data, Musical Information Retrieval (MIR) has become an important field of research. MIR is the science of retrieving information from music [10]. This information can be bibliographical (e.g., artist, genre, style, etc.) or musicological (e.g., melody, rhythm, tempo, instrumentation, dynamics, etc.). MIR is an interdisciplinary field, combining different aspects such as computer science, music, psychology, etc. It has many different applications, such as music recommender systems [48], music classification [59], automatic music transcription [28], and automatic music generation [25].

1.1 Problem Definition

1.1.1 Automatic Music Transcription

One important problem in MIR is Automatic Music Transcription (AMT), which is the process of automatically converting music to a symbolic notation such as a music score or a Musical Instrument Digital Interface (MIDI) file using a computer. However, there are other definitions by other researchers. For instance, E. Benetos et al. [5] defined it as a conversion process from an acoustic musical signal to a common form of musical notation. Since manually transcribing music to sheet music is a time consuming step of music composition, professional musicians and composers can use AMT to speed up this

process.

J. A. Moorer [34] and M. Piszczalski and B. A. Galler [45] were the researchers who first used the term "automatic music transcription" to describe an audio engineering problem in 1977. They believed that by analyzing digital recordings of music using a computer, the pitches of the melodies and chords could be detected and converted to the corresponding sheet music.

The AMT process can be divided into several tasks such multi-pitch detection, note duration detection, rhythmic information extraction, and music notation [5]. Pitch detection is considered as the main task of AMT process, which is used for estimating different pitches from audio signals. The first step of this task is to digitize the analog music in order to be processed in computer. Since the best pitch detection algorithms work in the frequency domain, the Fourier Transform is used to do the transformation of the digital music data from the time domain. Then, using pitch detection algorithms, each spectrum of the frequencies is analyzed to identify the peaks. Finally, after extracting the candidate pitches from the analyzed peaks, they are mapped to the closest musical pitches (notes).

1.1.2 Visual Automatic Music Transcription

As defined in Section 1.1.1, AMT is known as the process of extracting the musical sounds from a piece of music and transcribing them to musical notations. The main input in this procedure is audio or sound because the music is basically generated from sound. That is why all previous developed AMT methods are based on audio processing techniques.

In this research, a new concept for AMT called **Visual Automatic Music Transcription (VAMT)** is defined. VAMT is the process of visually extracting the musical notes played on the musical instrument and then transcribing them to the corresponding symbolic notation. The whole process in this technique is based on visual analysis of the instrument and the player's gestures, for instance, the piano and pianist's hands and fingers. Unlike the general AMT techniques, the audio of the music played using an instrument is not considered in

this method. Instead, the video of the instrument used for playing the music is captured during the performance to be analyzed using video processing techniques. Then, according to the visual features, the notes played on the instrument are detected and transcribed to the corresponding musical notations.

1.2 Motivation

Automated transcription of music, especially instrumental music, is an interesting problem for researchers. Almost all proposed methods are based on audio signal processing and none of them are satisfactory because of the audio drawbacks described in Section 2.4.1. Thus, new technologies such as VAMT are required to deal with this problem.

In this thesis, a new VAMT system named **claVision** is introduced to perform automatic transcription of piano music. In this software, a camera is located at top of the piano keyboard to capture a video of the performance. The software visually analyzes the music played on the piano based on the pressed keys and the pianist's hands. Finally, according to this visual information, the transcription of the played music is automatically produced without analyzing the audio of the music. The name claVision is a combination of two words, **clavier**, which means piano in many European languages, and **vision**, which comes from computer vision.

This idea was inspired by the famous composer and pianist, Ludwig van Beethoven [61, 65]. By the last decade of his life, he was almost totally deaf, forcing him to rely more heavily on the visual relationship between his fingers and piano keys. It is amazing that many of his most admired works were composed when he was deaf. claVision mimics what a deaf musician such as Beethoven does. Unlike other similar products that perform automatic music transcription by analyzing the audio signal, in claVision, the audio of the played music is ignored. As a result, the drawbacks of existing transcription techniques from audio are no longer present. Some of these drawbacks will be discussed in Section 2.4.1.

claVision was the winner of the 2014 Canada Microsoft Imagine Cup Innovation Competition that is a global contest for the best new innovative software. In addition, as one of the top 11 teams around the world, claVision won the 2014 Imagine Cup Innovation Competition: World Semifinals to be chosen as the Canada's representative in the World Finals in Seattle in July 2014 [19]. In the World Finals, claVision was demonstrated in different venues in Seattle, USA such as the University of Washington, Microsoft headquarters, and the Museum of History & Industry.

1.3 Applications

AMT has many important applications. The transcribed music score can be used as an alternate compact representation for the computer to store, reproduce, and play the music. This information can be used in the analysis and arrangement of the piece of music. Music transcription allows music to be easily shared between users. Moreover, since one of the challenging steps of composing music is writing the music score, a novice composer can have the task of music transcription automated as the music is being improvised or composed. Thus, manual music transcription can be replaced by automatic and fast music transcription especially when no musician is available to perform it.

In addition to the technical applications of AMT for musicians, claVision has a number of other useful applications:

- claVision can be used in piano education. For example,
  - claVision highlights pressed keys during a piano performance. It can be used to teach piano visually to hearing-impaired people who wish to learn piano;
  - if claVision has the musical information of a specific piece already, it can watch a student's performance and identify his/her mistakes in playing the piano (correct and incorrect pressed keys);
  - remote music lessons: a teacher can watch the live video of the student playing

the piano and see if the keys pressed are correct. When the quality of the network communication is poor, there may be significant noise in the video and the audio stream that the teacher has access to. The played keys are highlighted so they are easier for the teacher to see remotely. In addition, symbolic representations such as music scores are more resilient to noise;

- using claVision, concert pianists can visually show their performances (pressed keys on the piano and the music score) in a live concert;
- the performances of pianists in musical competitions can be evaluated by considering the outputs of claVision such as highlighted keys and music scores;
- claVision only requires a camera looking over a "piano-like" keyboard. It can be used on pianos, electronic keyboards, or even toy pianos. Thus, anyone can build their own keyboard and have it sound like a real piano using music synthesis software.

1.4 Summary of Contribution

The contribution of this thesis includes both the design of the claVision system as well as the resulting working implementation. All musical keyboards such as piano, harpsichord, electronic organs, etc. can be used with the software. Even if only a portion of the keyboard is captured by the camera, claVision still transcribes music correctly, as long as all the keys played are visible. It has a high accuracy in transcribing piano music from video performances, over a wide range of speed and complexity of the played music. Unlike other techniques and software products, claVision handles all steps in music transcription automatically with no need for human music experts to assist the transcription. Both live (real-time) and recorded video processing can be handled by claVision. The real-time transcription is performed with a very low latency. Despite the sensitivity of image and video processing techniques to the illumination issue, claVision can deal with many different lighting conditions using an illumination correction algorithm.

1.5 Overview

In Chapter 2, a background of primary music elements and musical instruments will be summarized. In addition, the previous approaches to AMT and its related problems developed by other researchers will be reviewed. The system architecture, description of algorithms, and implementation of claVision will be discussed in Chapter 3. Finally, in Chapter 4, the effectiveness of claVision in terms of accuracy and processing time will be evaluated. The conclusion of the whole thesis and some potential directions to extend claVision will be given in Chapter 5.

Chapter 2

Background

2.1 Primary Elements of Music

> "Music is the expression of the deepest part of our souls. It expresses what words and paintings cannot. And for true understanding, music requires careful attention and the engagement of intellect..."
>
> - Jeremy Yudkin [69]

There exist various literary and technical definitions of music. However, it can generally be defined as an artistic and organized collection of sounds designed to be pleasantly heard by other people. In other words, music can be perceived as the language of sounds. Like a language, it is a method of human communication, either spoken or written, consisting of the use of letters and words in a structured way. The written form of music notation is called a music score or sheet music [42].

In this section, we define some basic music elements such as **melody**, **pitch**, **dynamics**, **rhythm**, **tempo**, **harmony**, and **texture**.

2.1.1 Melody

A small unit of sound in music is called a **note** [69]. The notes in music are similar to the letters or syllables in a language. A mixture of notes forms a **melody**, which is the first basic element in music. A melody consists of different types of note-to-note movements called melodic motions.

Pitch is the term used to describe the highness or lowness of a note based on the frequency[1] of vibrations creating the sound. The faster the vibration, the higher the pitch. For instance, the highest note in music has a frequency of 4,186 vibrations per second, while the lowest one has only 27.5 vibrations per second.

There are 7 natural notes[2] which are shown with the letters A, B, C, D, E, F, and G in western countries. This pattern is repeated again and again for notes of different frequencies. An **interval** is the distance between two pitches or notes. Most of the intervals are made up of combinations of whole and half steps. The interval from one natural note to the next one is a whole step. According to the number of steps between the notes, 7 different intervals can be defined: **second**, **third**, **fourth**, **fifth**, **sixth**, **seventh**, and **octave**. For example, the interval from C to F is a fourth, which is the result of counting the 4 notes C, D, E, and F. If we count up to eight from a specific note, we will get to another note with the same name (e.g., C up to C). This interval is called an octave.

For notating pitches in a sheet music, notes are located on a **staff**, which consists of five parallel lines with spaces in between [36]. In order to establish the locations of the notes on a staff, **clef** signs are used. **Treble** or G **clef** and **bass** or F **clef** are two common clefs used in sheet music. Figure 2.1 demonstrates the arrangement of the notes on the lines and spaces of a staff in each clef.

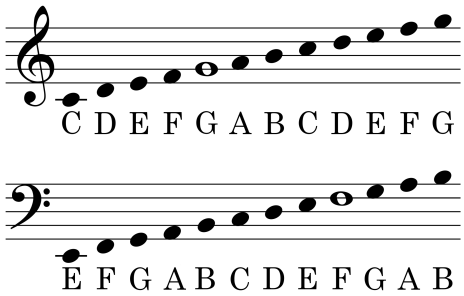


Figure 2.1: Treble (top) and bass (bottom) clefs with the arrangement of the notes.

[1] Some music texts refer to this as speed [69].

[2] The note is called natural if no alteration is applied on its pitch. For example, the lowest 7 natural notes have the frequencies of 27.5000, 30.8677, 32.7032, 36.7081, 41.2034, 43.6535, and 48.9994.

A staff constructed using a combination of a treble and a bass clef is called a **great** or **grand staff** (Figure 2.2). Grand staves are utilized for covering a very wide range of pitches, for example, in piano music.

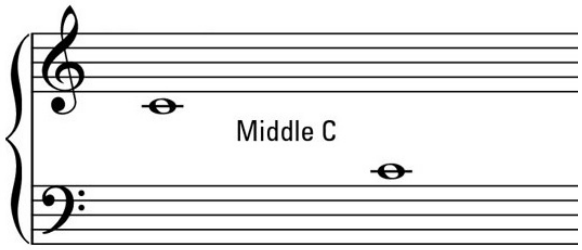


Figure 2.2: Grand staff with the Middle *C*.

In any piece of music, notes are played in specific temporal durations called **note durations** or **note values** [36]. The note with the longest duration is called the **whole note**. The length of a whole note is determined according to a time signature explained in Section 2.1.2. The other values are calculated by binary divisions. **Half**, **quarter**, **eighth**, and **sixteenth** are the other common notes used in music. Although the smaller values (e.g., 32nd note) can be calculated, they are seldom used. For every note value, there is a corresponding unit called **rest** used for representing a specific duration of silence or pause. Based on the durations described above, the notes are written as some specific shapes in sheet music, which are demonstrated in Figure 2.3.

A note is called **sharp** (♯) if it is a half step up from a particular natural note, and called **flat** (♭) if it is a half step down. Each natural note has two main modifiers, ♯ (except *B* and *E*) and ♭ (except *C* and *F*) [69]. Based on this definition, the sharp modifier of a specific note is the same as the flat modifier of the next note. There are 12 notes including both natural notes and their modifiers, and they are written as *A*, *A*♯ (*B*♭), *B*, *C*, *C*♯ (*D*♭), *D*, *D*♯ (*E*♭), *E*, *F*, *F*♯ (*G*♭), *G*, and *G*♯ (*A*♭). In sheet music, the sharps and flats are written before the corresponding note. However, if they are going to be common in a piece of music, they can be written in **key signatures** at the beginning of each line in the staff (Figure 2.4).

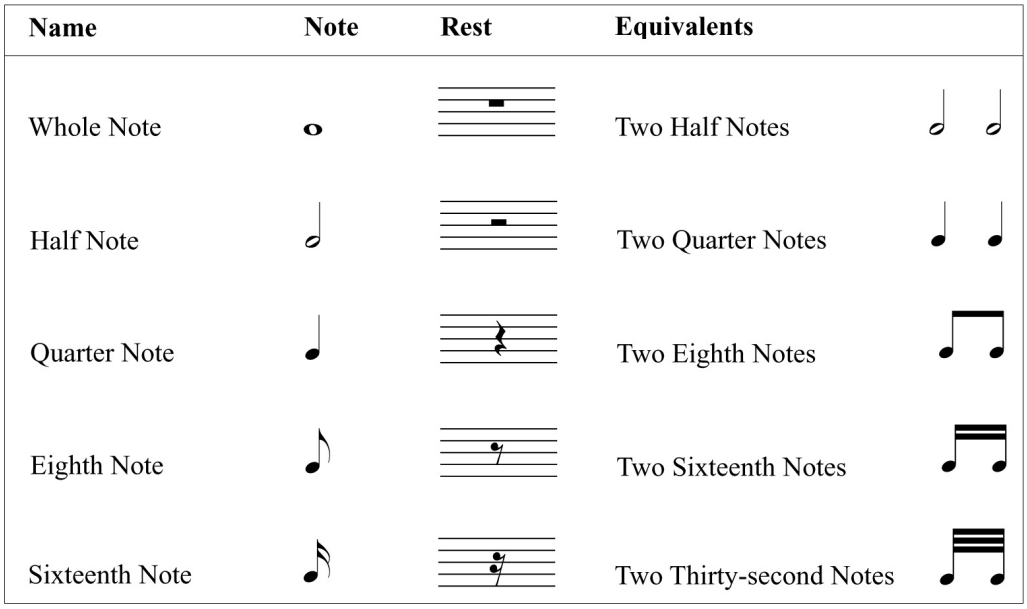


Figure 2.3: Note values, rests, and binary divisions.

One of the quickest ways to get the audience's attention in a musical performance is increasing or decreasing the volume in playing melodies. These sudden changes in volume are called **dynamics**, which result in the loudness or softness of music [36, 69].

2.1.2 Rhythm

Rhythm is the second fundamental component of music. It is regarded as the essence of melody as well as one of the most significant distinguishing features in music. The main element of rhythm is the concept of **beat**, which provides the regular pulse of music. A group of beats is called a **measure** marked off by small vertical lines in sheet music. To describe the number and length of the beats in each measure, a **meter** or a **time signature** is used. The meter is shown by a symbol including two numbers. As an example, Figure 2.4 shows the meter $\frac{3}{4}$, which in the upper number (3) shows there are 3 beats in a measure and the lower number (4) indicates that the value of each beat is a quarter note [36, 69].

A **tempo** is the speed of a given piece of music [42]. Tempo can be indicated in different ways. The most common way for musicians, which indicates the number of beats per

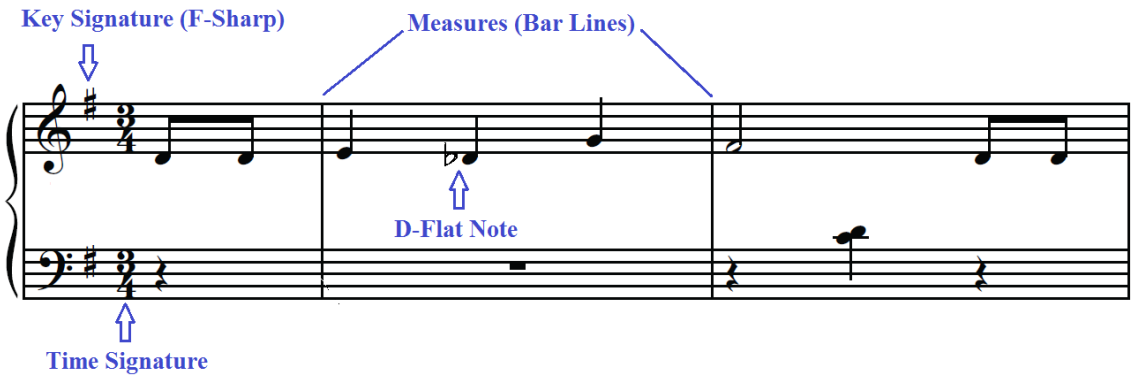


Figure 2.4: Key signature (the sharp modifier is applied to all *F* notes played in octave number 4), time signature (the meter $\frac{3}{4}$), measures (separated by bar lines), and D-Flat Note (the flat modifier is applied only to this specific *D* note).

minute (BPM), is to write it at the start of the piece. This mathematical tempo became popular after the invention of the metronome (a device that measures the beats in music) [42]. In western classical music, the tempo was described by one or more words (e.g., **Allegro** means fast). However, in cases when there is no indicated tempo, it is up to the performers to choose an appropriate speed for playing the piece using their interpretations.

Tempo rubato describes the expressive flexibility of rhythm or tempo by slightly increasing and then decreasing the speed of playing a piece of music [42]. Although tempo rubato literally means taking the duration from one measure or beat and giving it to another, in modern practice it can be applied to any kind of irregularity of rhythm or tempo. There are two types of rubato. The rubato is **melodic**, if the tempo of the melody is flexible, while the accompaniment remains in its regular pulse. However, when both melody and accompaniment are affected by this flexibility in time, the rubato is known as **structural**.

2.1.3 Harmony

The combination of a melody and its accompaniment is called **harmony**, which can create different moods and feelings in a piece of music. **Chords** play the most important role in accompanying the melody, which are shaped by playing three or more notes together [36].

The **texture** of music is another important aspect of harmony, which describes the way different musical sounds are combined. There are three main textures in music, **monophony**, **homophony**, and **polyphony**. Monophony is a texture that involves a single melodic line heard unaccompanied (e.g., a single person singing in the shower). If the melody is accompanied by chords, the music is known as homophony (e.g., instrumental music). Polyphonic texture describes having two or more different musical lines at the same time (e.g., symphonies) [69].

2.2 Musical Keyboard Instruments

There is a variety of instruments used for making music. If the musical instrument is played using a keyboard, it is called a keyboard instrument. The most common keyboard instruments are the piano, the harpsichord, the organ, and the synthesizer [69].

During the early 19th century, a variety of shapes and constructions was established with pianos. The modern piano was developed during 1830–1850, when two basic models, upright and grand pianos were mass-produced [13]. The frame and strings are horizontal in grand pianos, while they are vertical in upright pianos. In order to modify the sound of the played notes, piano pedals are used, which are operated by pianist's feet at the base of the piano. The **soft** and **sustain** pedals are two main pedals used for softening and sustaining the played notes. The modern pianos usually include one more pedal named **sostenuto** used for sustaining selected notes [13].

Harpsichord was considered as the king of musical keyboard instruments from the 15th to 18th century [13]. It is known as the most important predecessor of the piano [69]. From the outside, harpsichord is very similar to the piano, especially the keyboard, but their mechanisms for producing the sound are different. By pressing the keys on the harpsichord, the corresponding strings are plucked to produce the sound, while they are hammered in the piano [13].

The organ is a popular instrument in large halls and churches [13, 69]. In this instru-

ment, air moves through pipes to produce sound. Moreover, there is a keyboard used for controlling the route of the air. Organs range in size from small to very large. Large organs can have hundreds or thousands of pipes and two or three keyboards.

The synthesizer is an instrument that electronically imitates the sounds of other musical instruments [69]. It is commonly used in popular music, film soundtracks, and theater industries. Synthesizers are usually controlled using keyboards. However, different input devices can be used for this purpose.

Regardless of the mechanisms that different keyboard instruments use for producing musical sound, they all have similar physical features in the structure of their keyboards. All musical keyboards include some white and black keys indicating the natural notes and their modifiers. In almost every modern piano, there are 52 white keys and 36 black keys, which is 88 keys in total [13, 36, 69] (Figure 2.5). Modern pianos include seven octaves (numbered from 1 to 7) plus four extra keys (A_0, $A_0\sharp$, B_0, and C_8). However, the number of keys and octaves in older pianos as well as other musical keyboards may be different.

Generally, the *C* note with a frequency around 261.6 Hz is called the **Middle** *C*, which is the fourth *C* key (C_4) on a standard 88-key piano keyboard. However, some musicians refer to this as the middle of their instruments. As explained in Section 2.1, a grand staff is used for notating piano music. The Middle *C* in this staff is the note *C* placed on the first **ledger line**[3] below the treble staff as well as the first ledger line above the bass staff (Figure 2.2).

2.3 Musical Instrument Digital Interface (MIDI)

Musical Instrument Digital Interface (MIDI) is a protocol which allows different electronic instruments and other digital musical tools to communicate with one another [18]. This technology was developed by the MIDI Manufacturers Association (MMA) in 1983 [2]. MIDI carries a series of event messages such as notation, pitch, volume, etc. In order

[3]The ledger lines are the small lines drawn on the spaces above and below the staff in order to extend the staff for writing more notes [36].

to produce sound, a MIDI instrument is required to interpret these messages. A MIDI instrument can be hardware (e.g., electronic keyboard) or software (e.g., Ableton Live [20]). Compactness of representation (storing songs in a few hundred lines), ease of manipulation, and flexible choice of instruments for synthesizing the sound of the notes are some advantages of MIDI. MIDI information can also be saved as a standard MIDI file.

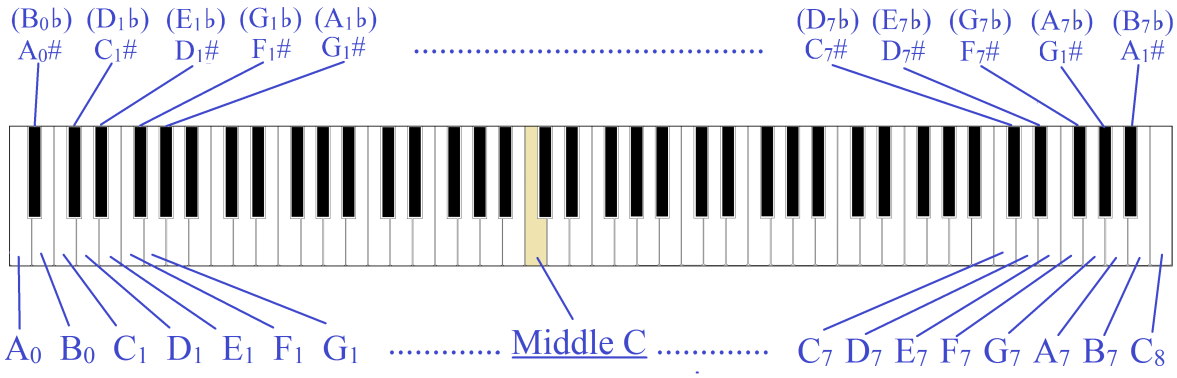


Figure 2.5: 88-key piano notation.

The basic messages related to playing the notes in a MIDI structure are **Note On** and **Note Off**. When a note is played in a MIDI instrument, the message Note On is created. In an electronic keyboard, it is created when a key is pressed. This message contains two pieces of information, note and **velocity**. The note describes the pitch of the played note with a value between 0 to 127. For example, the Middle *C* in MIDI is the note with the value of 60. The velocity indicates the volume of the played MIDI note, which has a value between 0 (low) to 127 (high). The message Note Off is created when playing a note ends (releasing a pressed key on a keyboard).

There are some other event messages used in MIDI, for examples, the messages corresponding to instrument name, tempo, time signature, and key signature. If the instrument name, the tempo, and the time signature are not specified in a MIDI file, they are assumed to be acoustic grand piano, 120 BPM, and $\frac{4}{4}$.

2.4 Automatic Music Transcription

In this section, different approaches to automatic music transcription developed by other researchers are discussed. First, a brief background of audio processing techniques is given. Then, some other algorithms proposed based on image and video processing for visually transcribing music will be presented.

2.4.1 Audio Signal Processing

Many people believe that the musical information of a played piece can be retrieved by considering the music audio because the music is basically generated from sound. Researchers have tried to develop audio processing techniques to analyze the pitches and frequencies in a piece of music in order to extract the played notes. In below, some of them will be mentioned.

In 1977, Piszczalski and Galler proposed a spectrum-based system for analyzing the recordings of single instruments with strong fundamental frequency [45]. In the same year, the first method with the ability to detect two simultaneous musical sounds was proposed by Moorer [34]. Various research groups did research in polyphonic pitch detection using different techniques. The first polyphonic pitch-tracking approach was developed by R. C. Maher [33] in 1990, which could process real musical recordings if the voices did not cross each other. Two voices (melodic lines) are said to cross when the higher one (consisting of higher pitches) intersects the lower one (consisting of lower pitches) and vice versa. As a result, lower pitches are present in the higher voice and higher pitches are present in the lower voice.

Most of the recent polyphonic music transcription research have focused on multi-pitch detection and note tracking algorithms. Multi-pitch detection is considered as the main problem in AMT, which is the estimation of simultaneous pitches in a time frame. Note tracking algorithms are used for analyzing time-frequency representation of input music in order to detect note events including the corresponding pitch values and onset/offset times

[5]. Multi-pitch detection systems can be classified into different types based on the way that they perform their estimation [68]. Feature-based multi-pitch detection [68, 11], statistical model-based multi-pitch detection [43], and spectrogram factorisation-based multi-pitch detection [54] are some of the different approaches considered for the problem of multi-pitch detection. Although there has been some recent progress in multi-pitch estimation techniques, it remains a research challenge [27].

There are also a few software products that attempt to perform automatic transcription of music by analyzing audio signals alone [1, 20, 21, 23, 37, 63]. In spite of the robustness of their algorithms in processing and manipulating audio signals and waves, they are not very accurate and efficient in the transcription tasks because of a number of difficulties as described below.

- At any instance in time, there may be multiple lines of music (melodies and chords) being played, for example, in polyphonic and homophonic music. The combination of various audio signals makes it very difficult to distinguish all the notes that are played at the same time. Software products such as Ableton Live [20], Melodyne [37], and Logic Pro X [21] cannot accurately transcribe music with multiple lines [57].

- Similarly, if two or more instruments are played together (e.g., in an orchestra), several lines of music are produced at the same time. It is very difficult to isolate the music played by one instrument from the others.

- The process is susceptible to audio noise. As a result, extracting the main audio signal is very difficult.

- It is difficult to detect the exact duration of a note from audio signals. In other words, onset (the beginning of the musical note) and offset (the end of the musical note) of the played notes cannot be accurately detected [3]. This process is much more

difficult for some instruments such as the violin because the pitches of the played notes gradually change over a long time period [8, 27].

- If the instrument is not perfectly tuned, it is impossible to extract the played notes correctly by identifying the correct pitches of the notes.

Because of these difficulties, these products are often not very accurate in their transcription. Therefore, many of these products (e.g., intelliScore [23], Transcribe [64], and TwelveKeys [63]) require human music experts to assist (or correct) the transcription.

There are also a few products such as ScoreCloud Studio [1] and Akoff Music Composer [30] that can perform perfect transcription. However, they require special electronic MIDI keyboard instruments to be connected to the computer running the software.

2.4.2 Digital Video Processing

Piano

Detecting the keys pressed on the piano from a video performance is a challenging problem in computer vision. In April 2014, an automated approach was proposed by P. Suteparuk for visually detecting and tracking the piano keys played by a pianist using image differences between video frames [56]. First of all, the piano keyboard was located, rotated, and cropped for further processing. Then, the hands were detected using skin colour detection. The hands were removed from the images, and the black and white keys were extracted. Eventually, the difference image between the background and current frame was computed to identify the changed pixels indicating the pressed keys. The background image containing only the piano keyboard was required to be manually taken by the user at first. Figure 2.6 shows some intermediate results in the procedure in [56].

As described in [56], a number of YouTube videos was used for testing the algorithm instead of using a video camera to capture the performance video. Some of the keys were not detected correctly because of noise and shadows of the hands produced due to the use of skin colour algorithm. The algorithm had a 63.3% precision rate, a 74% recall rate, and

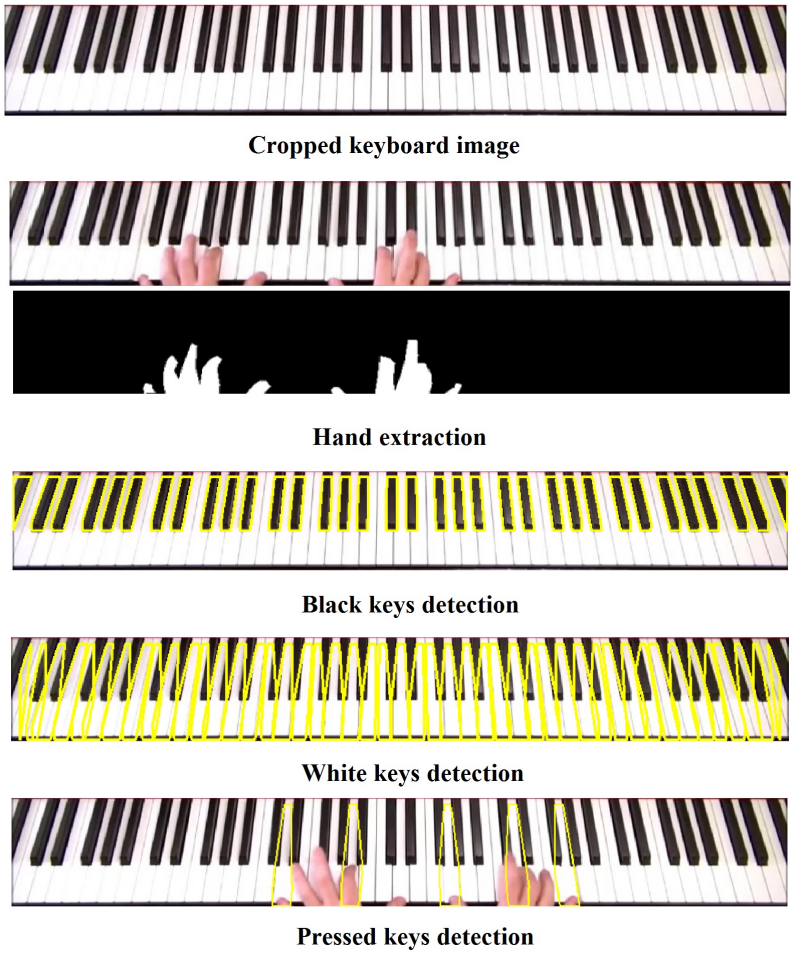


Figure 2.6: Different stages used in the proposed method in [56].

an F1 score of 0.68 in identifying the pressed keys [56]. However, for faster and more complicated pieces of music, the proposed algorithm was less accurate and slower. The algorithm was not tested under different illumination, scales, or image quality. Since the played music was not transcribed to the actual musical notes in [56], the described method cannot be regarded as an automatic music transcription system.

I am not aware of any products and research on visual automatic transcription of acoustic piano music or any other musical keyboards in order to transcribe the music played on the piano to the corresponding sheet music, especially in real time.

Other Musical Instruments

The first visual method for automatic transcription of violin music was proposed in 2007 [70]. The goal of this work was to use visual information of fingering and bowing on the violin in order to improve the accuracy of audio-only music transcription. This visual analysis contained four main steps: multiple finger tracking, string detection, fingering event detection, and physics-based automatic note inference. In order to track the four fingers of the violinist's left hand, the Condensation algorithm [24, 70] was applied. This algorithm is based on joint finger model dynamics and Gaussian skin colour model to track the fingers. Then, Hough line transform [17] was applied to detect the four strings as well as their starting and ending points at the nut and bridge of the violin. Based on the positions of the four fingertips on the four strings, the fingering events (i.e., press and release), the string number, and the distance form the pressing point to the nut were extracted. Finally, according to the physics of string vibration, the fingering events were translated to the played notes.

As described in [70], the multiple finger tracking algorithm had an accuracy of 92.4%. The string detection algorithm was successful in 94.2% of the cases for the correct detection of the starting and ending points of the strings. However, automatic note inference including the onset, offset, and pitch detection of an inferred note had a very low accuracy of 14.9% compared to human annotated results.

2.5 Fingering Information Extraction

Almost all musical instruments are played using hands and fingers. In order to make the hands and fingers movements comfortable for players, especially beginners, most music scores include appropriate fingering information indicating which hand and finger should be used for playing which note. As a result, fingering information can be considered as another useful piece of information required to be extracted using AMT. There exists some research related to the extraction of fingering information of music played on musical instruments

such as piano and guitar in recent years [15, 26, 38, 41, 52, 55]. Various techniques and methods have been proposed in order to deal with this detection process.

2.5.1 Digital Video Processing

In 2006, D. O. Gorodnichy and A. Yogeswaran [15] demonstrated a video recognition tool called C-MIDI to detect which hands and fingers were used to play the keys on a piano or a musical keyboard. Instead of detecting the played notes using transcription methods, the musical instrument was equipped with a MIDI interface to be able to transmit MIDI events associated to the played notes to the computer. A video camera was located above the piano keyboard to capture the pianist's hands and fingers during a performance. Then, a three-stage procedure including keyboard image rectification, hand tracking, and finger location estimation was applied to detect the hand and finger corresponding to each pressed key (Figure 2.7). After the initialization step, hands were detected as foreground objects on the piano keyboard and then continuously tracked based on hand templates. In order to localize the positions of the fingers, an edge detection approach called crevice detection was developed. Crevices were defined as the contact points of two convex shapes. In this approach, the edges of fingers were detected as crevices. As mentioned in the paper, tracking the hands did not work well if hands overlapped or the music piece was complex. Moreover, only half of the finger annotations obtained by their algorithm were correct.

Figure 2.7: The three stages used in [15]: 1) Keyboard detection, 2) Hand tracking, and 3) Finger detection.

Recognizing human gestures, especially hands and fingers, is a very challenging problem in computer vision community. It has many applications such as computer-human

interaction, robotic, and multimedia. Some researchers have considered the extraction of fingering information as a gesture recognition problem. There are various approaches used for recognizing human gestures. For classifying gestures, one method is based on Hidden Markov Models (HMM), which are commonly used for problems involving stochastic processes [67]. M. Sotirios and P. Georgios [55] used HMM to present a method for retrieving fingering information from the right hand of a pianist through the use of a digital camera. The camera was installed in front of the piano to record the finger movements (Figure 2.8). The video signal was then transferred to the computer for offline recorded video processing. The retrieval process included three steps: finger segmentation and tracking, feature vector extraction, and HMM classification. The algorithm did not recognize the finger movements used for playing the black keys on the piano. In addition, it was not tested on different pianists nor under different circumstances. Experimental results such as the accuracy and processing time of recognition were not given in the paper [55].

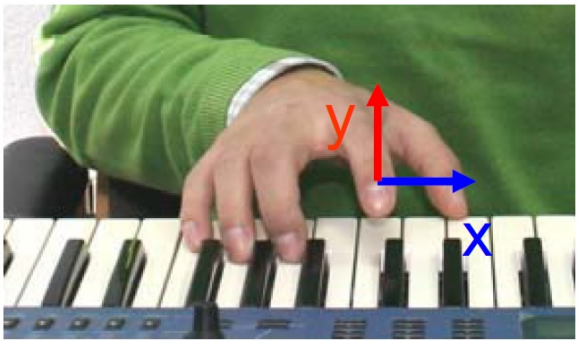


Figure 2.8: Camera view in [55].

Another technique related to recognizing gestures used for playing piano keyboard was proposed by F. Jean and A. B. Albu [26]. They used a meta-instrument keyboard made of wood with the keys painted in different colours. The keyboard was put on the ground, and the algorithm tracked feet motion on this keyboard in order to extract "key-press" events played by the feet. According to the experimental results in [26], the proposed system worked with a low latency (0.03 sec) in a real-time process and the average error in key

identification was only 9.4%.

Retrieval of fingering information is also a very important task for other musical instruments, especially the guitar [6, 35, 52]. Since audio AMT methods do not produce this positional information required for guitar music notation, different vision-based techniques have recently been developed. Motokawa and Saito [35] presented an educational system based on computer vision for assisting people to play the guitar. Their system required a visual marker attached to the guitar to be able to detect and track the guitar fret-board. The accuracy of this system was not given.

As an another example, J. Scarr and R. Green [52] introduced a marker-less algorithm for locating guitar fret-board as well as detecting individual locations of guitar's fretting and fingers using computer vision techniques. For locating the fret-board, the largest cluster of the lines (strings) detected by a line detection algorithm was found. This cluster was then matched with a bounding rectangle. The frets were extracted by applying the horizontal Sobel filter [14] on the rectangle (fret-board). Next, the four largest individual contours were detected as the four fingers used for playing the guitar by computing the number of skin-colour pixels in each contour. Finally, by analyzing the detected frets as well as the contours corresponding to the fingers, the location of each fingertip was determined to see which frets were active. The results shown in [52] indicate that the proposed system had a 69% accuracy in detecting the finger positions and a 52% accuracy in recognizing the chord progression. This approach used no tracking algorithms to reduce latency. In addition, the algorithm was very sensitive to different lighting conditions [52].

Recent developments have focused on multi-modal transcription systems utilizing both audio and video processing techniques [12, 41, 46, 47]. Frisson [12] implemented a multi-modal system for extracting information from guitar music. However, this system used markers for visual tracking.

In 2008, M. Paleari et al. [41] presented a complete multi-modal approach for guitar music transcription. Their method first used visual techniques to detect the position of the

guitar, the fret-board, and the hands. Then, the extracted visual information were combined with the audio data to produce an accurate output. According to the results described in [41], this technique had an 89% accuracy in detecting the notes.

2.5.2 Depth Sensors

Finger detection algorithms in most of the previous works have been implemented using digital cameras and digital video processing techniques. However, there are some works that utilize depth sensors such as the Microsoft KINECT. For example, A. Oka and M. Hashimoto [38] presented a marker-less (no special markers such as colour labels on the fingers) method for recognizing piano fingering through the use of depth images obtained from a KINECT depth sensor. They believed depth images were more suitable than video images in order to extract piano fingering because of the diversity of skin colours and illumination issues. The proposed method in [38] consisted of two modules: learning (registering fingering patterns) and recognition (matching the pattern with the input image). In the learning module, a dictionary data set comprising many depth images for different key pressing patterns was generated. This data set was then utilized in the recognition module to be compared with all input depth images using Nearest Neighbor search algorithm. The results obtained by testing the developed approach with sample piano music written for beginners showed that it had a 91.6% accuracy in recognition and required 120-millisecond processing time per note. However, the experiments with more complicated music were not done. As in the method in [15], the musical notes were acquired from a MIDI-keyboard in [38].

2.6 Virtual Keyboards

A projection keyboard [29, 66] is a form of computer input device whereby the image of a virtual keyboard is projected onto a surface: when a user touches the surface covered by an image of a key, the device records the corresponding keystroke. There are typically

two main steps. First, the virtual keyboard is generated using a laser projector, then finger motion is detected, and the coordinates of the affected key are determined using a sensor or a camera. Some of the developed projects have done the second step by using optical geometry (detecting the action of finger by the light reflected back) [29], and some used a camera and image processing techniques [31].

There are some products based on virtual pianos. For example, there is an application software named Synthesia Piano Hero [32] that is used for educational purposes. Using this package, the piano learners can watch and follow the notes received from a recorded MIDI file or an electronic keyboard connected to the computer. There are some other projects related to this topic that have been developed before, such as Piano Hero With Virtual Keyboard [31] and Virtual Piano Project [44]. Note that there are no publications for the mentioned projects. The only available references related to them are website links given in the bibliography.

Chapter 3

claVision: Visual Automatic Piano Music Transcription

The software claVision is a new way to perform automatic transcription of music played on a piano keyboard or any similar musical keyboard, using only a digital camera (e.g., web-cam) looking over the keyboard. The camera is located at the top of the piano keyboard to capture a video of the pianist's performance (Figure 3.1). Then, this video stream is sent to the software to be processed using computer vision techniques by analyzing the hands and the pressed keys on the piano. Finally, without any audio data, the played music is automatically and accurately transcribed to the corresponding music sheet. Real-time music transcription is the most important goal of claVision. So, the algorithms chosen for developing this system are mostly simple to decrease the latency in real-time processing.

In claVision, the audio is ignored. As a result, many of the drawbacks of existing transcription techniques from audio (Section 2.4.1) are no longer present as described below:

- claVision can easily distinguish multiple played notes because it sees all the keys that are pressed at the same time;
- claVision is not affected by the presence of other instruments because it only watches the piano keyboard;
- without audio, there is no audio noise;
- claVision extracts the exact duration of the played notes by considering the press and release times of the keys;

- even if the piano is not tuned, claVision transcribes correctly because it sees which keys are pressed. This is particularly useful for beginning students at home, when perfectly tuned equipment may not be available.

3.1 System Architecture

3.1.1 Hardware and Setup

The requirements for performing real-time video capturing and music transcription are a digital camera and a tripod, stand, or any kind of stable mounts holding the camera at the top of the piano. Any type of low-cost camera with an appropriate resolution and video frame rate can be used. However, to achieve a good performance, it is recommended to use a camera with spatial resolution and frame rate of at least 320×240 and 24 frames per second (FPS). The higher the video frame rate, the more accurate fast pieces of music can be transcribed. However, it may also increase the processing time because more frames are required to be processed.

The camera should be mounted over the piano keyboard so that the appropriate portion is visible. The camera does not need to cover the entire piano keyboard. Even if only a portion of the keyboard is captured by the camera, the software can still function properly, as long as all the keys played are visible. Although the software can adjust for variations in camera positions leading to rotated and angled views of the keyboard, it is recommended that the camera be located not far from the ideal location as demonstrated by a sample setup on a piano keyboard in Figure 3.1.

By connecting the camera device to the computer (e.g., through a USB cable), the video stream can be transmitted to the computer from the camera. claVision automatically detects all connected cameras to the computer.

Figure 3.1: Ideal location of the camera over the piano and electronic keyboards. The angled view of the camera causes the pressed keys to be seen more clearly.

3.1.2 Software

claVision has been implemented in Microsoft Visual Studio .Net 2013 with the C-Sharp (C♯) language. This software is compatible with Microsoft Windows (XP or higher) and can be installed on any Windows-based computer in a few simple steps using an executable setup file. Microsoft.NET Framework 2.0 or higher needs to be installed on the PC before running claVision.

Both live (real-time[4]) and recorded video can be processed by claVision. In other words, the input to this software can be either a live video stream captured by a camera or a recorded video file. Real-time music transcription in claVision means that the pianist can play a piece of music on the piano keyboard and can see the result of transcription at the same time.

claVision can produce three different types of outputs.

[4]A system is real-time if it guarantees response within specific deadlines (strict time constraints) [4].

1. Highlight of the pressed keys in the video window showing the piano performance in the user interface (Figure 3.2), which can also be recorded and saved as a video file.

2. Music score of the played music (Figure 3.2) that can be exported as a Portable Document Format (PDF) file.

3. MIDI sound synthesized from the transcribed music, which the user can save as a MIDI file and play it back after.

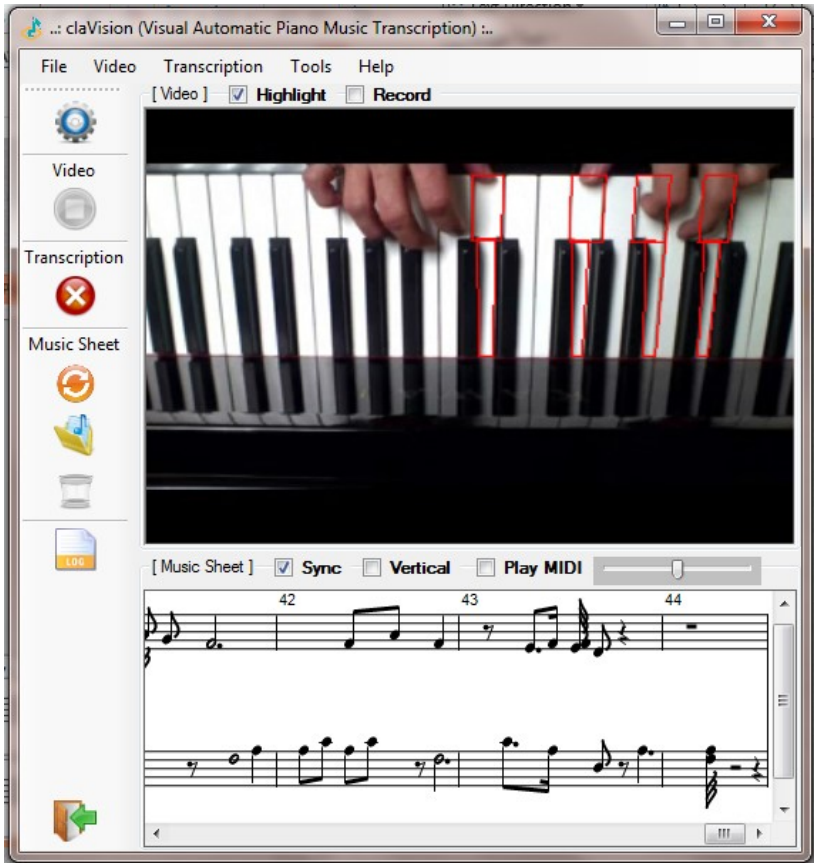


Figure 3.2: Screenshot of claVision showing the outputs.

Three different open-source libraries have been utilized in the implementation of the software.

1. Some basic and well-known image and video processing algorithms implemented in the AForge.NET Framework [62] were benefited from to implement the software.

The AForge.NET Framework is an open source C-sharp framework designed for developers and researchers in the fields of computer vision and artificial intelligence such as image processing, neural networks, genetic algorithms, fuzzy logic, machine learning, robotics, etc.

2. To display, save, and print the sheet music produced in the software, the open source library Midi Sheet Music [60] is used. This package is also used for playing and adjusting the MIDI files produced in claVision as well as converting them to the corresponding sheet music.

3. Toub.Sound.Midi [58] is a .NET package for reading, writing, modifying, and playing MIDI events and files. In claVision, this framework is utilized for the real-time process of constructing and playing the MIDI sound.

3.2 Description of Algorithms

In claVision, there are four main stages to perform music transcription: keyboard registration, illumination normalization, pressed keys detection, and note transcription. First, the location of the keyboard and the keys on it are computed and registered. Then, a correction process is applied to each working video frame to deal with issues caused by varying illumination. Following that, the played keys in each frame are recognized and mapped to the respective musical notes. At the final stage, the obtained musical information is transcribed to the MIDI structure and sheet music. In the following subsections, each of the four steps and the required sub-tasks in detail will be described. Most of the algorithms used for developing claVision are relatively simple. The design philosophy is to use the simplest methods possible in order to reduce the latency in real-time processing, while maintaining a high transcription accuracy. In this section, we give a brief outline of the algorithms used in claVision. However, some details will be omitted in order to protect the intellectual property during the patent application process.

3.2.1 Keyboard Registration

This stage is divided into three tasks. Keyboard detection is done at first, which includes locating, transforming, and cropping the piano keyboard. After that, the best background image, which includes the piano keyboard without any hand covering it, is extracted from the first few frames. Having the adjusted keyboard image, the black and white keys as well as their musical features are calculated and stored to be used in the next stages.

Task 1. Keyboard Detection

The keyboard is a quadrilateral shaped by four edges or lines. To compute the location of the keyboard, the positional features of these four lines are extracted. Hough line transform [17] is utilized to detect all existing lines as well as their features in the input image. Before applying Hough transform, an edge detector is applied to identify the boundaries or edges of the objects within the image. There are different edge detection methods in image processing such as Difference edge detector, Sobel edge detector, and Canny edge detector [14]. After evaluating various methods, Canny edge detection was considered as the most effective algorithm for claVision (Figure 3.3). By performing Hough transform on the image resulted from the edge detection algorithm, the polar coordinates (radius and theta values) of all detected lines are calculated. Then, the lines are converted into Cartesian form.

In order to find the four lines forming the keyboard quadrilateral, all 4-combinations of the set of extracted lines are evaluated based on their intersection points. To reduce the number of candidates, only those combinations that result in at least four intersections in the plane are considered for further processing. The four intersections in each combination are considered to be the four corners of each quadrilateral.

Since the rectangles in the video image may be obtained with minor distortions (e.g., rotated, perspective, etc.), they are required to be transformed to rectangular images using a quadrilateral transformation algorithm. Given the four corners of each quadrilateral, the

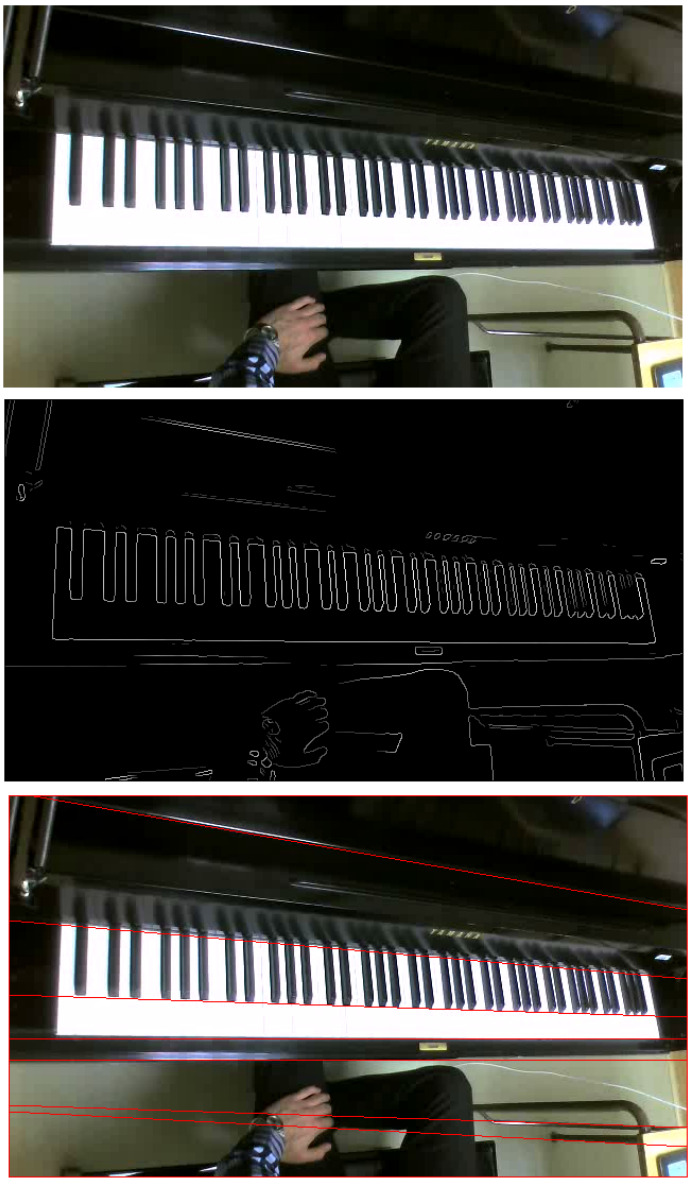


Figure 3.3: Original input image (top), Canny edge detection (middle), and line detection using Hough line transform (bottom).

homogeneous transformation method proposed by P. Heckbert [16] is utilized to do this transformation (Figure 3.4).

The next step is to find the transformed rectangle that contains the keyboard. According to the structure of piano keyboards, the black keys are placed at the upper two-thirds of the keyboard. The lower one-third only consists of the white keys. Thus, a rectangle can be considered as the keyboard if it has 1) the maximum brightness in the lower one-third and

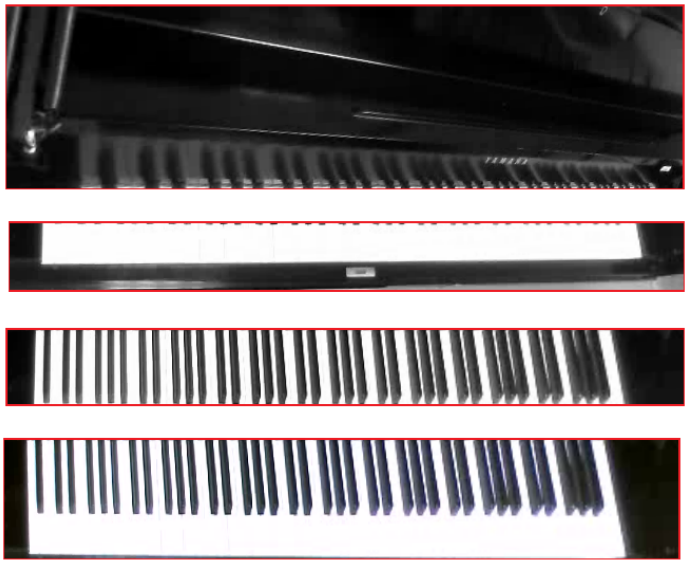


Figure 3.4: A few sample transformed rectangles obtained from the line detection image in Figure 3.3.

2) the maximum number of black keys located in the upper two-thirds part. For the first condition, the means of all intensities in the lower one-third of all candidate rectangles are compared with each other. For counting the number of the black keys in the second condition, blob extraction method that is based on connected components labelling algorithm [50] is used to extract and count individual objects (blobs) in the image. Since the blob detection algorithm is performed on binary images, the keyboard image needs to be binarized with an appropriate threshold. The Otsu clustering-based thresholding method [39] is used to automatically calculate this threshold point for differentiating the objects (black keys) from the background (white keys). Figure 3.5 demonstrates blob detection applied to the bottom image in Figure 3.4.

In some cases, the keyboard may not be successfully detected in the first video frame due to issues such as lighting and/or hands coverage. In these cases, the keyboard detection process needs to be repeated for the next frames until the desired keyboard is identified.

As soon as the keyboard detection process is done, the four points of the chosen quadrilateral are stored to be used for locating and transforming the keyboard image in the next

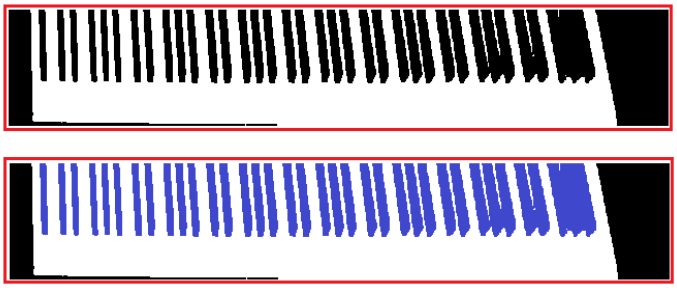


Figure 3.5: The binarized image computed by applying Otsu method to the bottom image in Figure 3.4 (top) and highlight of the blobs detected by blob detection (bottom).

video frames using the same quadrilateral transformation explained above. In addition, the image of the detected keyboard is considered as the initial background image, which will be required in the next stages.

Task 2. Background Update

After detecting the keyboard location, a procedure is used to continuously improve the initial background image by analyzing the next video frames based on the two conditions described in Task 1. In other words, if a keyboard image is found whose brightness in the lower one-third and its number of black keys in the upper two-thirds are more than the ones in the previous background image, it is replaced as the new background image.

There are two main reasons for updating the background image continuously. As described before, the best background image is the keyboard without any hands covering it. In some situations, when the camera starts capturing the video of the performance on the piano, the pianist's hands have already covered the keyboard. In this case, the initial background image determined in the first frames includes the hands as well. To handle this issue, the background image is replaced in the next frames in the hope that there is a frame the hands are not present. The other difficulties are noise and variations in lighting conditions during the performance. In order to accurately detect pressed keys, the background image should be consistent with the current lighting conditions. Therefore, the background image needs to be updated as the lighting conditions change.

Task 3. Keys Detection

Keys detection is applied to the obtained background image because it does not include any hand covering the keys. Keys detection is done in two steps: determining the location of both white and black keys on the keyboard, and then assigning the relevant musical notes to them. Considering black keys as objects in white background, they are extracted using the blob detection algorithm explained in Task 1. In order to locate the quadrilateral associated to each black key, the four points upper-left, upper-right, lower-right, and lower-left are determined. These four corners are extracted based on the points related to the left, top, right, and bottom edges of each black key. For instance, the intersection between the left and top edges gives the upper-left point.

Unlike the black keys, it is difficult to differentiate the white keys from each other, particularly the ones with no black key in between. This is because the lines separating the white keys on the keyboard are not often obvious enough in the captured image (e.g., the bottom image shown in Figure 3.4). However, according to the standard dimensions of the piano keys [9, 51], it is possible to estimate these dividing lines based on the positions of the black keys (Figure 3.6). The white keys on the keyboard are isolated using information from adjacent black keys except for those white keys which have no black key between them. There are four different cases considered.

1. The line separating the white keys G and A is placed in the middle of the black key $G\sharp$.

2. The line separating the white keys C and D is placed on the right one-third point of the black key $C\sharp$. This rule is the same for the white keys F and G with the black key $F\sharp$ between them.

3. The line separating the white keys D and E is placed on the left one-third point of the black key $D\sharp$. This rule is the same for the white keys A and B with the black key $A\sharp$ between them.

4. There is no black key between the white keys B and C. In order to locate the line separating them, the midpoint between the left ($A\sharp$) and right ($C\sharp$) black keys is used. The same calculation is applied for the white keys E and F based on the black keys $D\sharp$ and $F\sharp$.

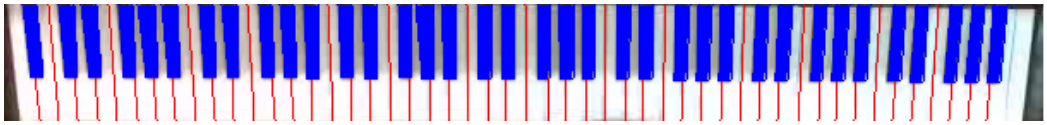


Figure 3.6: Estimated lines of the white keys based on the adjacent black keys.

The lines dividing the white keys may not always be vertical due to the imperfect image transformation performed in Task 1. In order to draw these lines, two different points are required. The first point is identified using the estimation process described above. Given the first point and the slope of the line, the second point can be calculated. The slope of the line is equal to the slope of the black key used for finding the first point. Finally, the white keys can be approximately located based on the estimated lines. Two quadrilaterals are computed for each white key: one placed at the upper two-thirds of the keyboard surrounded by the black keys and the other one placed at the lower one-third of the keyboard. So, each white key is modeled as the union of two quadrilaterals (Figure 3.7).

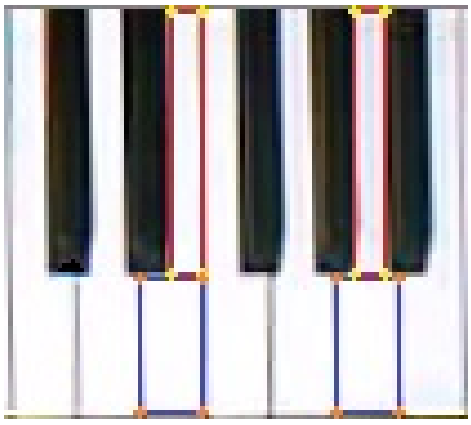


Figure 3.7: Two quadrilaterals corresponding to the white keys. Two sample white keys are highlighted.

The next step is to assign musical features such as the octaves and the notes to all located keys. In some cases, the captured video does not include the entire piano keyboard with all octaves because of the position of the camera. As a result, identifying the exact octave numbers is impossible even for humans. However, the octave numbers can be estimated by considering the middle visible octave as the octave including the Middle *C* key (it is usually known as octave 4). Then, the other octaves located on the left and right can be numbered accordingly. The middle octave can be extracted by dividing the number of visible octaves by two. In each octave in the piano keyboard, there are five black keys. So, the number of visible octaves is the number of black keys (computed using the blob detection algorithm) divided by five.

The black keys on the piano are divided into groups of two black keys ($C\sharp$ and $D\sharp$) and three black keys ($F\sharp$, $G\sharp$, and $A\sharp$). This pattern is repeated in the whole keyboard. Since there is no black key between these two groups, the space separating them is doubled in comparison to the space between the black keys inside each group. Based on this property, these two groups to determine their associated musical notes in each octave can be distinguished. The natural notes corresponding to the white keys are then determined based on the notes of the black keys and the estimated separating lines. For example, the two quadrilaterals which are separated by the line with the start point on the black key $G\sharp$ ($A\flat$) are assigned with the notes *G* and *A*.

Finally, the location (one quadrilateral for each black key and two quadrilaterals for each white keys), the octave number, and the musical note associated to each key on the keyboard are stored to be used in the next stages.

3.2.2 Illumination Normalization

Dealing with different illumination conditions is one of the most significant issues in image processing. This problem is also challenging for video processing, especially when the lighting conditions (brightness or darkness) may change in each video frame. Different

types of noise and shadows are other factors that also cause problems in image and video processing. In order to deal with these issues, various correction techniques are used by researchers. A general method for handling these problems in video processing is to reduce the effect of noise by decreasing the difference between the video frames. This method is similar to low pass (smoothing) filters used for averaging rapid changes in intensities. The proposed method in claVision is to consider a specific frame (e.g., first frame) as the overlay image. Then, based on the intensities in this overlay image, a low level manipulation of pixels is performed in other video frames. As a result, the minor differences (e.g., in illumination, noise, and/or shadows) between the overlay and other images are reduced. This method is implemented in the Aforge.Net Framework [62], in the MoveTowards filter.

In claVision, the MoveTowards filter is used to deal with the problems related to lighting, noise, and shadows (e.g., hands shadow) by considering the background image identified in Section 3.2.1 as the overlay image. The MoveTowards filter applies the following formula to each pixel in all video frames:

$$res := frm + \min(|bgr - frm|, step) \times Sign(bgr - frm) \tag{3.1}$$

where frm and bgr are the pixel values in the source image (video frames) and the overlay image (background image). The resulting pixel values are assigned to res. The parameter $step$ defines the maximum amount of change per pixel in the source image. The range of the step size is from 0 to 255. The resulting image will be the same as the overlay image if the step size is 255 and it will be equal to the source image if the step size is 0. Having an appropriate step size results in reducing the minor differences caused by different illumination, noise, and shadows between the background image and other video frames (Figure 3.8).

Figure 3.8: The result of MoveTowards algorithm with a step size of 50: Background image (top), current video frame (middle), and the normalized image (bottom).

3.2.3 Pressed Keys Detection

Given the background image and the normalized image in each video frame, pressed keys detection is done in four steps. First, the difference image between the background image and the normalized video image in each working frame is computed. Next, the location of the pianist's hands on the keyboard is determined in order to identify the candidate keys potentially pressed by the pianist. Based on the pixels identified in the difference image, the pressed keys are then detected and matched with their related musical features. The detected keys are finally highlighted according to their locations on the keyboard.

Task 1. Background Subtraction

Pressing the keys on the piano keyboard causes some changes at the pixel values at the locations of the pressed keys. In order to detect these intensity variations, the background subtraction technique is utilized. Every single pixel in the background image is subtracted from the corresponding one in each normalized video frame. All resulting values are between -255 and 255. The positive and negative values resulted from this subtraction are used separately for analyzing the pressed black and white keys.

1. **Positive difference image**: when a black key is pressed down on the keyboard, the adjacent white keys are more prominent. In other words, some of the dark pixels of the pressed black key are replaced by the brighter pixels of the adjacent white keys in the image. The background image includes the unpressed black key and the working video frame includes the pressed one. As a result, a difference image with positive values is produced. These positive values represent the changes caused by pressing the black keys (Figure 3.9).

2. **Negative difference image**: when a white key is pressed down on the keyboard, the adjacent black keys are more prominent. In other words, some of the bright pixels of the pressed white key are replaced by the darker pixels of the adjacent black keys in the image. The working video frame includes the pressed white key and the background image includes the unpressed one. As a result, a difference image with negative values is produced. These negative values represent the changes caused by pressing the white keys (Figure 3.10).

The advantage of analyzing positive and negative differences separately is that it distinguishes the dark-to-bright and bright-to-dark pixel variations. The detection of the pressed white and black keys is done separately to make sure the white keys are not incorrectly detected as their adjacent black keys and vice versa. The resulting difference images are both converted to the corresponding binary images. The foreground pixels can be either a few blobs or some isolated pixels. The binary positive and negative difference images will be referred as Bkeys and Wkeys images in the next sections.

Task 2. Hand Detection

The keys are usually pressed in the areas that the pianist's hands and fingers are present. Thus, the search domain for finding the pressed keys can be limited by detecting the location of the hands on the keyboard. As a result, pressed keys detection can speed up. Also, noisy detection results from other parts in which no hands exist can be removed.

Figure 3.9: Positive difference image (RGB and binary) resulted from subtracting the background image from the normalized video frame in Figure 3.8. Two black keys have been pressed.

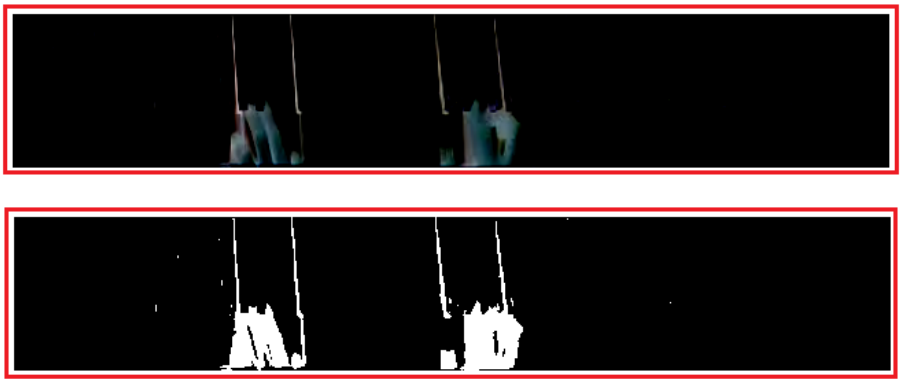


Figure 3.10: Negative difference image (RGB and binary) resulted from subtracting the background image from the normalized video frame in Figure 3.8. Four white keys have been pressed.

Since the pixel intensities associated to skin colour are lower than those associated to the white keys (even in the grayscale image), the hands and fingers on the piano keyboard can be determined using the Wkeys image resulted from the negative difference as described in Task 1. The blob detection algorithm is then applied on the Wkeys image in order to extract and locate the bounding boxes including the hands and fingers. These bounding boxes are then reversibly transformed to be highlighted in the original video image (Figure 3.11).

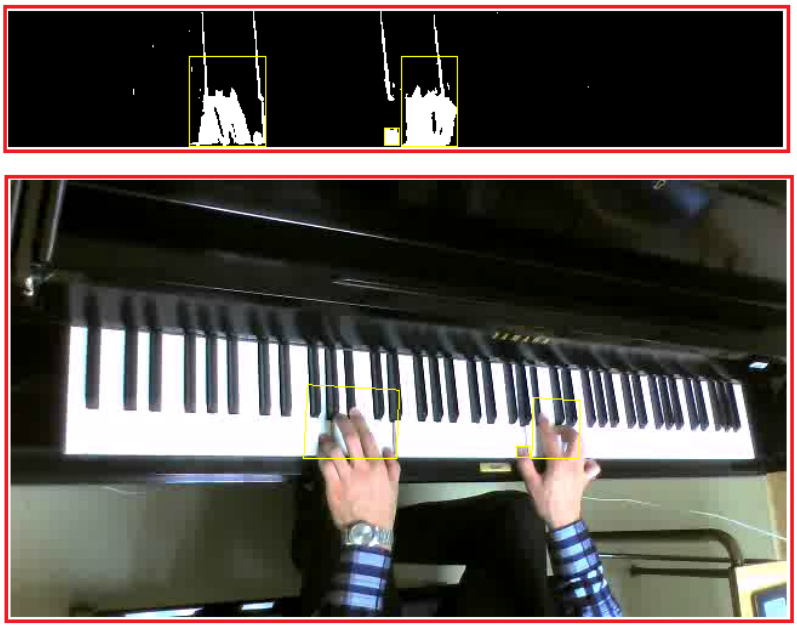


Figure 3.11: Hand detection: identified bounding boxes in the Wkeys image (top) and the original video frame (bottom).

Task 3. Note Detection

Having the list of all piano keys located and registered in Section 3.2.1, the two difference images (Wkeys and Bkeys) obtained in Task 1, and the location of the hands computed in Task 2, the notes played by the pianist in each video frame can be detected as follows. A key is considered as pressed if:

1. it is located in at least one of the bounding boxes including the hands,
2. its associated quadrilateral in the difference image includes at least one blob, and
3. the height of the considered blob is at least half of the key's height.

The white and black keys identified by the first condition are searched in the difference images Wkeys and Bkeys. The third condition is used to reduce the chance that the considered blob is noise.

For the located white keys with no black key in between (e.g., E and F), the situation is challenging. By pressing one of these keys, the blob(s) appear(s) in both keys. In this case,

the one with more blobs is chosen. If the number of blobs are the same, the key with more isolated pixels in its associated quadrilateral is chosen.

In addition to the blobs shaped by pressing the white keys in the Wkeys image, there are other blobs caused by pianist's hands covering the keyboard. To avoid the incorrect detection of these blobs as pressed keys, the lower quadrilateral of the white keys in the search is not considered because it is mostly covered by the hands.

All registered keys are searched using a linear search in order to detect the pressed keys using the conditions specified above. Although there are faster search methods (e.g., binary search), the linear search is fast enough for claVision. Next, their related features such as the location (quadrilateral), the octave number, and the note name are added to a list named PlayedList indicating the played notes in each video frame.

Task 4. Pressed Keys Highlight

Before highlighting the pressed keys in the original video image in each frame, the original positions of their associated quadrilaterals need to be calculated. Given the four corners of a quadrilateral and the generated mapping matrix [16], the transformation applied in Section 3.2.1 can be reversed. As a result, the four corners of the quadrilaterals in the original video images are identified. Finally, all quadrilaterals related to the pressed keys are highlighted by drawing coloured polygons (Figure 3.12).

3.2.4 Music Transcription

For transcribing a played note to symbolic notation (e.g., MIDI structure), the features such as the note name, the octave number, and the note duration are required. The list of the played notes (PlayedList) including the note names and their corresponding octave numbers were determined in Section 3.2.3. In order to calculate the note duration, the onset and offset times (attack and release times) of the played note need to be obtained. The following procedure is performed to identify the onset and offset of each played note.

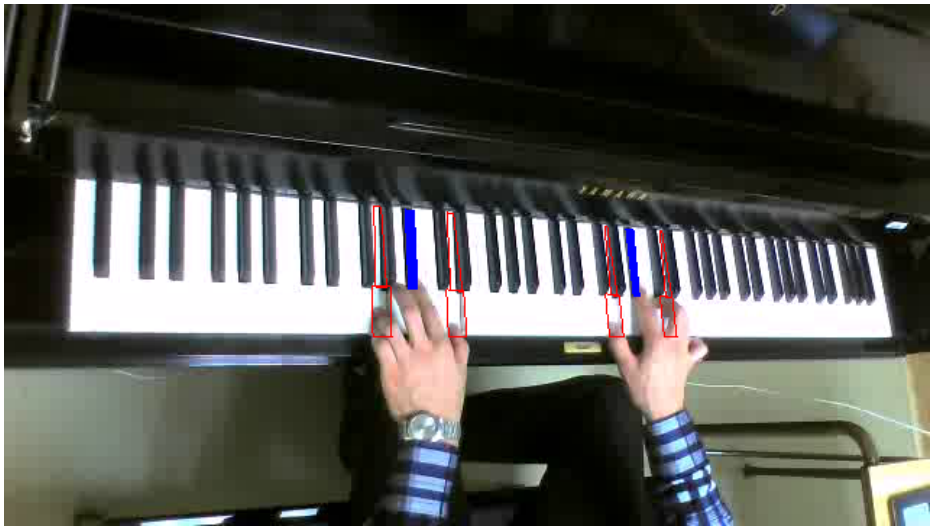


Figure 3.12: Pressed keys detection: highlight of the pressed white and black keys with the colours red and blue.

1. A temporary list named PreList indicating the onset/offset status of all previous played notes is defined.

2. Once a played note is detected and added to PlayedList in each video frame, one of the following cases may occur.

   (a) The note does not exist in PreList. Its onset is computed in milliseconds based on the system time and attached to the note features. Then, the note including its features is added to PreList. The real-time synthesized MIDI sound of the corresponding note is played at the same time.

   (b) The note exists in PreList. It means the note is still being played. In other words, the pianist has not released it since the last press. In this case, PreList remains unchanged.

3. In each video frame, all the notes in PreList are studied for their presence in PlayedList. If there is a note in PreList that does not exist in PlayedList, it means it has been released by the pianist. So, the current time is considered as the note offset attached to the note features. Playing the synthesized MIDI sound of the corresponding note is

then stopped. The same note may be replayed later by the pianist. Since there should be only one record related to each note in the list, the current released note is removed from PreList.

Given the note features such as name, octave number, and onset/offset, a new MIDI event including the required Note On and Note Off messages related to each played note is created. These events are then added to a collection used for storing all MIDI events produced in each video frame. A MIDI stream is used to store this collection. The other corresponding MIDI messages such as the instrument name, tempo, and time signature can be manually determined by the user. If not, they are assumed to be acoustic grand piano, 120 beats per minute, and $\frac{4}{4}$ by default as described in Section 2.3. By adding these messages, the MIDI stream is regarded as a complete MIDI structure representing the transcription of the played music.

Finally, the constructed MIDI file is converted to the corresponding sheet music using the Midi Sheet Music library [60] as explained in Section 3.1.2. The sheet music is then displayed with the appropriate features and symbols such as time and key signatures, measures including vertical bars, notes and rests with their durations, chords, etc. (Figure 3.13). The sheet music is made up of a grand staff including a treble clef for higher notes and a bass clef for lower notes. The notes played in the first octaves (from 1 to 3) are written in the staff with the treble clef and the ones in the last octaves (from 4 to 7) are written in the bass one.

3.3 Implementation and Design

3.3.1 Fast Image Processing

Memory management in .NET is done in two different ways, managed and unmanaged [49]. In managed memory, a central entity called garbage collector (GC) is used to track each of the references to that memory. GC is responsible for releasing the objects in memory when there are no references to them. As a result, there is a guarantee that all resources

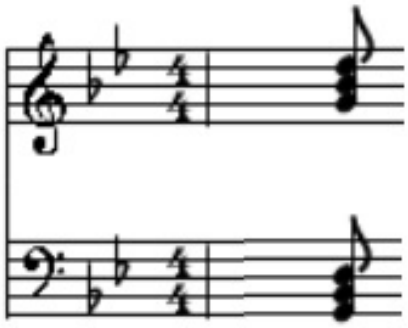


Figure 3.13: The produced sheet music corresponding to the pressed keys in Figure 3.12. It shows two played chords ($G_2, B_2\flat, D_3$) in the bass clef and ($G_4, B_4\flat, D_5$) in the treble clef.

are eventually disposed by an automatic procedure. For unmanaged memory, user code should release the objects in memory. If the user does not do this by mistake, the resources will never be released. So, managed memory is safer, but there is overhead involved.

In terms of the way that digital images are managed in memory in .NET, they can be either managed or unmanaged images [62]. In order to apply an image processing routine to managed images, the images need to be locked at first and then unlocked after the process is done. Although this lock/unlock operation has some benefits in memory management as described above, it causes overhead and low performance in image processing, especially when multiple routines are applied to a single image. In contrast, there is no need to do any lock/unlock to unmanaged images because they use unmanaged memory buffers. As a result, it is possible to apply faster image processing routines without any overhead. Since processing time is very important in claVision, all input images (video frames) are converted to unmanaged images (managed by user code) before applying the required image processing algorithms.

In addition, as no colour image processing is required in the implementation of claVision, all colour images are converted to grayscale images to decrease both processing time and memory usage.

3.3.2 Multi-threaded Programming

Simultaneous running tasks including a sequence of instructions in a computer program that can be executed and managed independently are called threads [53]. Multi-threaded programming is an execution model that allows multiple threads to exist and share the resources (e.g., memory) in a single program (process). It has many advantages and applications for software developers. Using multi-threading, a program remains responsive to input such that it does not freeze because of other running tasks in the background. Splitting a process into parallel sub-tasks can result in faster execution, especially on machines with multi-core CPUs.

Since threads can easily share data, code, and files with each other, the communication between them is simple. Threads need to be synchronized in order to communicate and share the resources with one another. If this synchronization is not performed correctly by the programmers, different problems may occur. As an example, a program may enter into deadlock, which happens when two threads are waiting for each other to finish and release a common resource, but none of them does [40].

In claVision, multi-threading is utilized by defining four different threads explained below.

1. The first running thread in claVision is the main thread. It is used for controlling the user interface of the software including all the required tools for video capturing and processing. This thread is also used for transmitting the video from the camera and showing it in the software. In addition, the illumination normalization and pressed keys detection processes are executed in this thread.

2. Keyboard registration (Section 3.2.1) is the most time consuming stage in the system, and is done in the first few frames. So, it may delay the process of transmitting video frames from the camera or the video file to the software and showing them in the corresponding user interface. To avoid this delay in the first frames, the whole process in this stage is performed in a separate thread.

3. As described in Section 3.2.1 (Task 2), a continuous procedure for updating the background image is performed for each video frame. As this procedure requires some time to finish, it is executed as an independent thread to make sure it does not slow down the real-time process of transcription performed in each working video frame.

4. In order to have a real-time transcription system, the music score containing the transcribed music needs be updated and displayed every time a key is pressed on the piano keyboard. This time consuming procedure consists of constructing the entire MIDI file based on the new played notes, converting it to the sheet music, and displaying the produced sheet music in the software. To increase the speed of producing the sheet music (Section 3.2.4) as well as decrease the processing time needed for analyzing the video frames, this procedure is also executed in a separate thread.

Chapter 4

Evaluation

In this chapter, the performance of claVision is evaluated based on accuracy and processing time. In addition, some limitations in claVision will be discussed at the end of this chapter.

The software was installed and tested on a laptop with an Intel Core i7-4510U CPU (2.00 GHz) and 16 GB DDR3 RAM. Although many images and videos of different piano and electronic keyboard performances have been used to evaluate the effectiveness of claVision, a few of the representative ones as samples are taken to demonstrate the results. They were captured using two different digital cameras, an SD 240p webcam (with resolution and frame rate of 320×240 pixels and 24 FPS) and an HD 720p webcam (with resolution and frame rate of 640×360 pixels and 30 FPS). Table 4.1 presents the list of the test videos with their features used in this evaluation.

Table 4.1: List of the sample videos including different slow and fast pieces of music (the column "Number of Keys" shows the number of visible white and black keys in the video).

Video ID	Number of Frames	Resolution	Frame Rate (FPS)	Number of Keys	Keyboard Type	Speed (Tempo)
V1	2596	320×240	24	35	Piano	Slow (40 BMP)
V2	1048	640×360	30	88	Piano	Slow (60 BMP)
V3	2090	640×360	30	60	Electronic	Medium (80 BMP)
V4	1491	640×360	30	60	Electronic	Medium (80 BMP)
V5	857	640×360	30	60	Electronic	Medium (120 BMP)
V6	4502	640×360	30	88	Piano	Fast (140 BMP)
V7	2281	640×360	30	60	Electronic	Fast (200 BMP)
V8	3607	640×360	30	66	Piano	Very Fast (240 BPM)

4.1 Accuracy and Processing Time

The four main stages of keyboard registration, illumination normalization, pressed keys detection, and music transcription are separately evaluated in this section. First, the accuracy and processing time of the tasks in keyboard registration including keyboard detection, background update, and keys detection are discussed. Following that, the effectiveness of the illumination normalization method used for dealing with the lighting issues will be analyzed. The recognition rate and the latency associated to detecting the hands and pressed keys in each sample video will then be given. Finally, the correctness of the music transcribed to the corresponding MIDI structure and sheet music will be evaluated.

As mentioned in Section 3.1, it is ideal to locate the camera with an angled view over the piano keyboard in order to see the pressed keys more clearly by exposing the side of the keys. The software can function properly under different illumination conditions as long as they do not change drastically during the performance. In Section 4.2, the limitations related to lighting will be discussed. In this section, the experiments and evaluations are performed under the normal situations mentioned above.

4.1.1 Keyboard Registration

The first step for detecting the keyboard is to apply Canny edge detection to the image (Section 3.2.1). This edge detector extracts edge pixels based on low and high threshold values (lowThreshold and highThreshold) [14, 62]. The next step is to detect the lines in the image using Hough line transform. In this algorithm, the lines are detected based on a minimum relative intensity (minRelInt), which is the ratio of lines' intensities to maximum intensity found in the image [17, 62]. According to the experimental results, the best default values for lowThreshold, highThreshold, and minRelInt in the algorithm are 20, 50, and 0.4, respectively. However, they can be changed to adapt to different circumstances.

The results show that the keyboard locations in all sample videos as well as many other images were correctly detected and transformed to the corresponding rectangles (Figure

4.1). The videos and images were captured in different conditions such as camera view (angled or rotated), lighting, quality, resolution, hands coverage, incomplete piano keyboard, etc. However, there are some situations in which the algorithm may not detect the keyboard correctly. These situations will be discussed in Section 4.2.

Figure 4.1: Eight sample images of different piano and electronic keyboards in different situations (top image in each sample), and the detected and transformed rectangles (bottom image in each sample).

Correct detection of the background image is strongly dependent on the success of keyboard detection. From the experiments on the sample videos, once the keyboard location is detected, the background image is appropriately detected in either the first frame after locating the keyboard or the next frames using the background update procedure (Section 3.2.1).

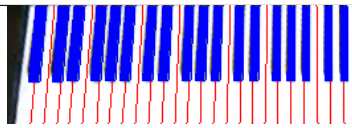
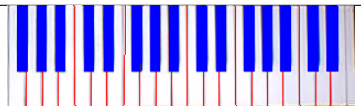
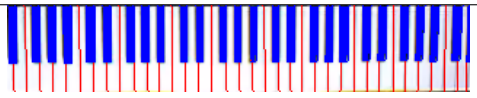
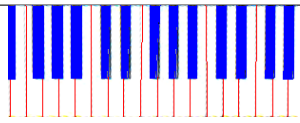
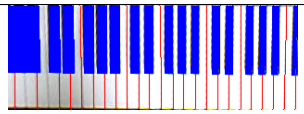
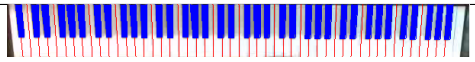
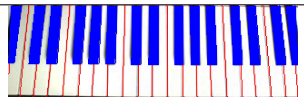
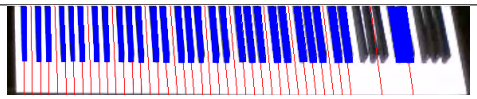
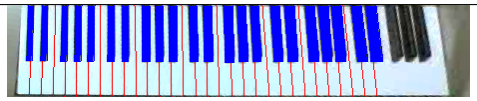
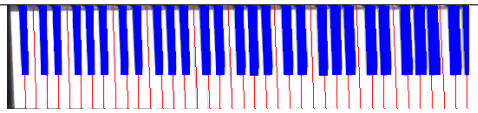
Table 4.2 demonstrates the accuracy of the white and black keys detection applied to the background images extracted from different videos. Image $I7$ is the background image of video $V1$ and image $I8$ is the one detected in the videos $V2$ and $V6$. The other images were not taken from the sample videos given in Table 4.1. As seen in Table 4.2, the detection rate

in most of the cases is 100%. However, some black keys in images $I5$ (leftmost octave), $I8$ (rightmost octave), and $I9$ (right-most octaves) were not located correctly because they were not detected as separate blobs. This is due to the perspective view of the camera taking the original image (before transformation), which resulted in connected black keys. Since the white keys are located based on the corresponding adjacent black keys in the algorithm, the ones locating around the undetected black keys were not successfully detected either. The average detection rate is 95.2%, which shows the robustness of keys detection in the software.

As stated in Section 3.3.2, the keyboard registration process causes no delay in the system, especially in the transmission of the video from the camera to the software, because all the steps in this stage are performed in a separate thread. Nevertheless, the processing time of each task in the keyboard registration process is evaluated (Table 4.3). Analyzing and updating the background is done in almost all video frames. The processing time of background update given in the table is the maximum and average over all video frames in the corresponding sample video. For real-time processing, an upper bound on the time taking for each step to finish is analyzed. Therefore, the maximum processing time in this evaluation is considered as well.

The keyboard detection process in videos $V3$, $V4$, $V5$, and $V7$ took about three times longer than the other sample videos. The frame colour of the electronic keyboard in these videos is grey (e.g., the ones in Figure 4.2). As a result, a lower value of MinRelInt is required to detect the lines separating the area including the keys from the keyboard frame. The lower the value of MinRelInt, the slower the line detection is performed. That results in more processing time required to find the keyboard in these videos. For the other videos, it was faster because the frame colour of the pianos used in them is black (e.g., the ones shown in Figure 4.1), which is easier to distinguish from the keyboard. Table 4.3 shows very low processing times for performing keys detection with a maximum and an average of 20 ms and 16.6 ms. The maximum processing time of the background update process is

Table 4.2: Results of keys detection. The column “Number of Keys” shows the number of visible white and black keys in the image and “Located Keys” shows the number of keys correctly located using keys detection.

Image	Background Image	Number of Keys	Located Keys	Detection Rate %
I1		44	44	100
I2		36	36	100
I3		60	60	100
I4		30	30	100
I5		43	37	86.0
I6		88	88	100
I7		35	35	100
I8		88	68	77.3
I9		60	53	88.3
I10		66	66	100
Average:				**95.2**

53 ms, while its average is only 6.1 ms. This maximum value is required for only one or two of video frames, which is caused by system issues such as caching, operating system, and the interactions between them.

Table 4.3: Processing time of keyboard registration.

Video	Keyboard Detection (ms)	Background Update (ms)		Keys Detection (ms)
		Maximum	Average	
V1	265	19	4	7
V2	335	23	4	15
V3	1030	47	7	20
V4	1080	47	7	20
V5	1060	47	6	18
V6	350	47	6	16
V7	1100	53	7	19
V8	315	53	8	18
Maximum:	**1100**	**53**	**8**	**20**
Average:	**691.9**	**42.0**	**6.1**	**16.6**

4.1.2 Illumination Normalization

Experimental results show that the most appropriate default value for the step size in the MoveTowards algorithm is 50 (Section 3.2.2). In situations where noise and shadows are more sever during a performance, this value could be increased. The higher this step size is chosen, the more undesired changes (e.g., noise and shadows) are removed in the video frames, but fewer pressed keys might be detected. This is because the pressed keys are also detected as changed pixels in the video frames. Therefore, choosing very high step sizes may cause these changes to be removed as well. According to the experiments, the illumination normalization method used in claVision system significantly corrects most of the issues related to lighting, noise, and shadows in video frames without sacrificing pressed keys detection. For example, in the Microsoft Imagine Cup Competition, claVision was demonstrated under a variety of illumination conditions in different locations such as a tent (with natural and artificial lighting), a museum (with shadows from people walking around), conference rooms (with camera flashes of photographers), etc. and it functioned very well.

The maximum and average running times required for performing illumination normalization on different video frames in each sample are shown in Table 4.4. Before starting

pressed keys detection, illumination normalization is performed on every single video frame in order to correct the issues such as lighting. Unlike keyboard registration, this stage is not performed in a separate thread. As shown by the results summarized in Table 4.4, illumination normalization with a maximum and an average processing time of 6 ms and 1.8 ms does not cause any significant latency for the next stages of processing such as real-time pressed keys detection and music transcription.

Table 4.4: Processing time of illumination normalization. The average time for video $V1$ is about a quarter of the others because of a lower resolution.

Video	Maximum (ms)	Average (ms)
V1	4.0	0.6
V2	4.0	2.0
V3	4.0	2.0
V4	4.0	2.0
V5	5.0	2.0
V6	4.0	2.0
V7	5.0	2.0
V8	6.0	2.0
Maximum:	**6.0**	**2.0**
Average:	**4.5**	**1.8**

4.1.3 Pressed Keys Detection

In order to binarize the difference images (Wkeys and Bkeys) computed in Section 3.2.3, an appropriate threshold value is required to distinguish the foreground pixels from the background as well as the noise. Performing illumination normalization on the video frames results in a significant noise removal. As a result, almost all of the remaining foreground pixels in the difference images need to be considered as the blobs or the isolated pixels associated to the pressed keys. That is why a threshold with the value of 1 is chosen to take all non-background pixels (greater than 0) into account. This value could be changed if necessary, based on the step size chosen for illumination normalization.

According to the experimental results, all bounding boxes corresponding to the hands

and fingers present on the piano keyboard are correctly identified, regardless of their skin colour, size, and number. Even if a very small portion of a finger is visible on the keyboard, it is detected properly. Figure 4.2 shows some sample images in which the detected boxes bounding the hands and fingers are highlighted. Since the hand detection algorithm in claVision is not sensitive to the skin colour and the shape of the hands and fingers, any other external objects (e.g., pen) used for pressing the piano keys are located as hands and their corresponding bounding boxes are extracted.

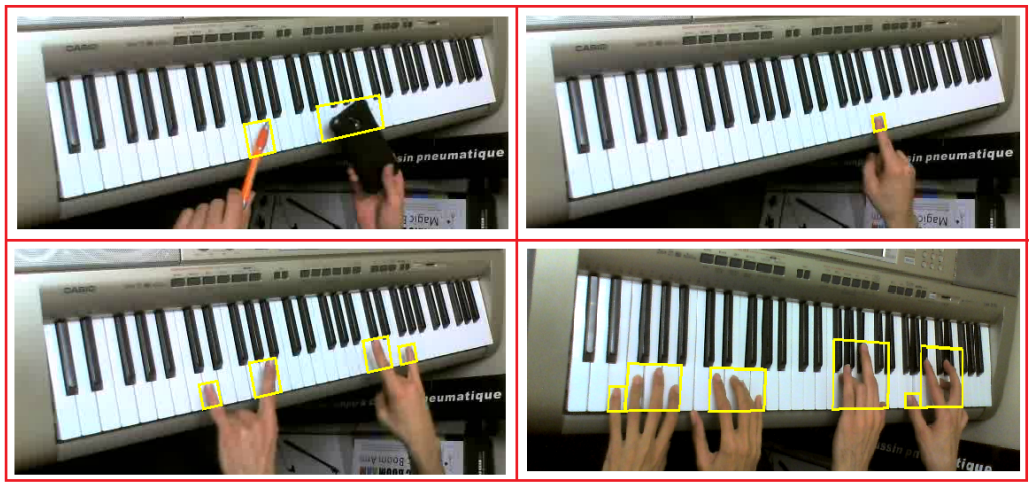


Figure 4.2: Sample images showing the results of hand detection (top-left: detection of a pen and a cellphone used for pressing the keys, top-right: detection of a small portion of a finger, bottom-left: detection of four separate fingers, and bottom-right: detection of four hands used for playing the keyboard).

The effectiveness of pressed keys detection in claVision is analyzed using the following statistics:

- True Positive (TP): the number of pressed keys detected as pressed.
- False Positive (FP): the number of unpressed keys detected as pressed.
- False Negative (FN): the number of pressed keys not detected as pressed.

These statistics can be used to compute other measures commonly used to evaluate classification algorithms. These statistical measures are recall, which is the ratio of de-

tected pressed keys to actual pressed keys, and precision, which is the ratio of pressed keys detected correctly to all detected pressed keys (both correct and incorrect). Recall and precision are calculated using the following formulas:

$$Recall := TP/ActualPositives \tag{4.1}$$

$$Precision := TP/(TP+FP) \tag{4.2}$$

where *ActualPositives* is the number of actual pressed keys during the performance.

As the test results demonstrate in Table 4.5, claVision has a very high accuracy in detecting the pressed keys even for very fast pieces of music. The average recall and precision rates are 97.5% and 97.4%, which indicate the effectiveness of claVision in this stage. However, there are some small errors, which are mostly because of one (or more) of the following problems:

- failure of keys detection in locating all keys on the keyboard;
- insufficient number of changed pixels (blobs and isolated pixels) in the difference images, which could be because of one of the following:
  - pressing white keys with no black key in between (e.g., *B* and *C*);
  - pressed keys that are directly below the camera (discussed more in Section 4.2);
  - pressing keys partially;

There are some other issues that influence the accuracy of pressed keys detection, which will be analyzed in Section 4.2.

The processing time corresponding to each task in the pressed keys detection process is summarized in Table 4.6. Since all three tasks are repeatedly applied to every single video frame in each sample, the maximum and average processing times required for performing

Table 4.5: Test results of pressed keys detection. All frames in the sample videos were considered for calculating these results.

Video	TP #	FP #	FN #	Recall %	Precision %
V1	303	12	11	96.5	96.2
V2	91	3	0	100	96.8
V3	322	2	5	98.5	99.4
V4	224	2	8	96.6	99.1
V5	137	10	3	97.9	93.2
V6	1078	36	35	96.9	96.8
V7	442	3	12	97.4	99.3
V8	1177	20	44	96.4	98.3
Average:				**97.5**	**97.4**

them on each video are considered. The average processing times for background subtraction, hand detection, and note detection are 4.7 ms, 0.7 ms, and 1.2 ms and the maximum ones are 17 ms, 10 ms, and 8 ms. In total, there is a low latency of 35 ms in maximum in the real-time processing of the pressed keys in each video frame. The total average processing time is 6.6 ms.

Table 4.6: Processing time of pressed keys detection.

Video	Background Subtraction (ms)		Hand Detection (ms)		Note Detection (ms)	
	Maximum	Average	Maximum	Average	Maximum	Average
V1	7.0	2.0	4.0	0.2	8.0	1.5
V2	11.0	4.5	7.0	0.6	8.0	1.0
V3	16.0	6.0	5.0	1.0	5.0	1.3
V4	12.0	6.0	8.0	1.0	4.0	1.3
V5	14.0	5.5	6.0	1.0	5.0	1.5
V6	12.0	4.0	9.0	0.5	4.0	1.0
V7	17.0	5.5	10.0	1.0	5.0	1.3
V8	14.0	4.5	7.0	0.6	4.0	1.0
Maximum:	**17.0**	**6.0**	**10.0**	**1.0**	**8.0**	**1.5**
Average:	**12.9**	**4.7**	**7.0**	**0.7**	**5.4**	**1.2**

4.1.4 Music Transcription

In order to evaluate the correctness of the transcribed music produced by claVision, the sheet music of the song performed in the sample video $V5$ is analyzed (Figure 4.3). This song is called "Twinkle Twinkle Little Star" used for piano learners. The time signature of this music is $\frac{4}{4}$. There are 12 measures numbered in the sheet music. No black keys are played in this music, so that no key signature is written in the sheet music. As mentioned in Table 4.5, 10 keys are incorrectly detected as pressed (false positive) in this piece of music. These incorrect notes are highlighted in the sheet music in measures 3 (C_3 note), 4 (F_3 note), 5 (C_3 note), 9 (two C_3 notes), and 12 (C_3 note) of the bass staff as well as measures 2 (F_5 note), 5 (F_5 note), and 11 (two F_5 notes) in the treble staff. In addition, there are 3 pressed keys that are not detected (false negative) in measures 2 (E_5), 5 (E_5), and 11 (E_5) of the treble staff. All errors are related to the adjacent white keys with no black key in between (e.g., E and F). This is because they are more difficult to be distinguished from each other in the difference images (Section 3.2.3).

Considering the synthesized MIDI sound produced by claVision, it is obvious that the transcribed music sounds the same as the played music on the piano keyboard. However, there are some notation mistakes associated to the note durations in the sheet music produced from the MIDI file. As explained in Section 3.2.4, all calculated note durations are stored in milliseconds which are understandable by computer. Based on the $\frac{4}{4}$ time signature of the played music in $V5$, there are 4 beats in each measure and the value of each beat is a quarter note (Section 2.1.2). In addition, as summarized in Table 4.1, its tempo is 120 BMP (120 beats per minute or 2 beats per second). As a result, the duration of each quarter note (one beat) in this piece of music is 500 ms. The duration of other notes can be accordingly extracted using binary divisions described in Section 2.1.1.

The software claVision produces the appropriate note shapes and measures in the sheet music based on the note durations in milliseconds. So, the more regular the notes are played based on the durations computed above, the more accurate the sheet music is pro-

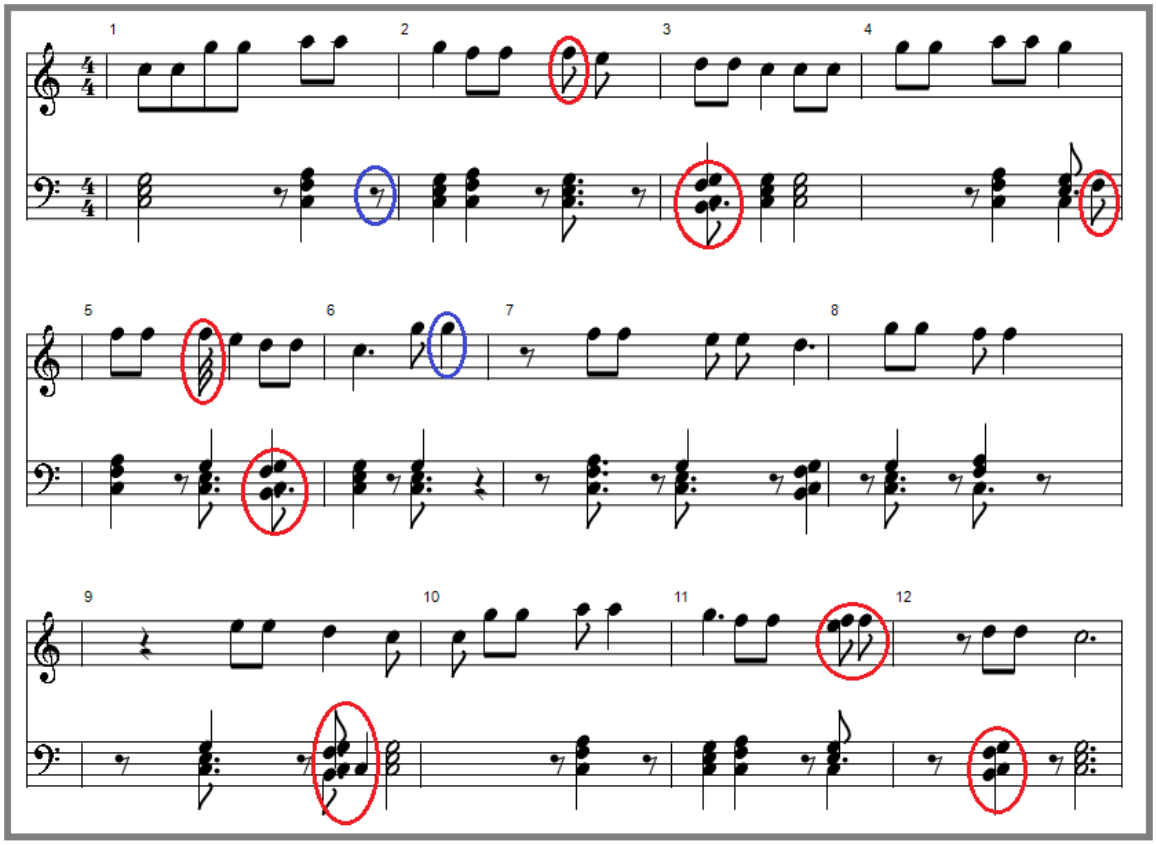


Figure 4.3: The produced sheet music of the sample video $V5$ (the song ”Twinkle Twinkle Little Star”).

duced. However, pianists usually apply tempo rubato to their performances, which results in flexibility in time and irregularity of rhythm and/or tempo (Section 2.1.2). In video $V5$, the pianist performed a variety of tempo rubato in the performance, which caused some notation errors in the duration and location of the notes and rests in the measures. For example, the number of beats in the first measure of the treble staff is only 3 beats, while it is 5 beats in the same measure in the bass staff. The extra beat in the bass staff is related to the eighth rest highlighted in the sheet music. One way to adjust the number of beats in this measure is to move this extra rest from the bass staff up to the treble staff. Another example is the quarter note G_5 in measure 6 of the treble staff, which should be an eighth note.

In order to accurately adjust the sheet music produced from the constructed MIDI file by claVision, a variety of software products developed for converting and editing MIDI files

and music scores can be used [1, 7, 22].

The time required for producing the sheet music in each sample video is about 5 ms. Moreover, the processing time of creating the MIDI structure in each video frame is very small (less than 0.1 ms) for all sample videos.

4.2 Limitations

In this section, we discuss a number of limitations in the claVision software. Some of these limitations are intrinsic to the approach and cannot be removed. However, there are some that can be removed using more sophisticated and often more time-consuming algorithms. Since the software is already very accurate in most practical situations, these algorithms are not incorporated into claVision in order to reduce latency.

4.2.1 Lighting Conditions

In Sections 3.2.2 and 4.1.2, the ability of illumination normalization to deal with the problems related to lighting conditions in the system was described. However, in situations where the lighting conditions change sharply during the performance, the software cannot function properly. This is because the illumination normalization algorithm cannot adjust major differences between the videos frames and the background image with an appropriate step size.

In this case, low step sizes result in too much noise in the difference images, and high step sizes result in the removal of the changed pixels associated to the pressed keys. Figure 4.4 shows the result of performing background subtraction on two video frames whose lighting conditions have changed during the performance. In both difference images, the high amount of noise is obvious. In Section 3.2.1, it was noted that the background image is updated as the lighting conditions change. Since this update process is based on the maximum brightness in the lower one-third of the keyboard, the background image is only replaced by a new one if the image gets brighter rather than darker. However, this brightness

should not be drastic such that the black keys become unclear. In these cases, claVision can actually handle the lighting conditions change and detect pressed keys accurately once the background image has been updated.

Figure 4.4: Result of background subtraction on two video frames with high brightness (left in the middle) and darkness (right in the middle). The background image is in the top and the results of background subtraction are shown in the bottom.

Sometimes, the illumination is stable during the performance, but the brightness or darkness of the images is very intense. As a result, the black and white keys cannot be differentiated from each other. This issue results in the failure of keyboard registration (Sections 3.2.1 and 4.1.1) in locating the keyboard and the keys on it.

4.2.2 Camera View Angle

The ideal location of the camera over the piano keyboard was demonstrated in Section 3.1.1. There are some situations that the keyboard and the keys cannot be located using the methods described in Section 3.2.1. For example, the software cannot deal with rotations of more than 45 degrees, or drastic perspective views of the camera (Figure 4.5).

In addition, if the camera looks straight down over the keys on the piano, the key press and release events cannot be reliably detected using the pressed keys detection algorithm. When a key is pressed down on the keyboard, the adjacent white or black keys are more prominent (Section 3.2.3). As a result, some intensity changes are detected in the difference images. The pressed keys detection algorithm analyzes these changed pixels to detect the

pressed keys. The more changed pixels are produced, the more accurate the pressed keys can be detected. When the keys located right below the camera are pressed, their adjacent white or black keys are not sufficiently prominent so fewer changed pixels appear in the difference images. As a result, the corresponding pressed keys cannot be detected properly.

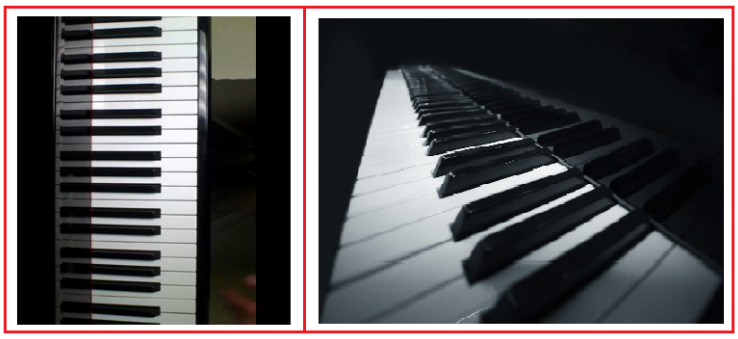


Figure 4.5: A 90-degree rotation of the keyboard image (left) and a perspective view of a grand piano keyboard (right) in which the keys in the low octaves are not clearly seen.

In order to evaluate the effectiveness of claVision in different camera view angles, a few videos of a piano performance were recorded and evaluated at various angles. In this video performance, all keys on the piano including white and black keys were pressed to see how different angles affect the recognition rate of pressed keys detection. Table 4.7 shows the list of the mentioned sample videos with their features. The table also summarizes the test results of pressed keys detection for these videos, which were computed based on the statistics defined in Section 4.1.3. All videos are in resolution and frame rate of 640×360 pixels and 30 FPS. The speed of pressing the keys in all videos is almost the same.

Since all white and black keys in the video $V9$ are obvious and differentiable to the camera, they were properly located using the keys detection process. The angle of the camera in this video is 0, which means the camera looks straight down over the piano keyboard. As a result, the pressed keys located right under the camera cannot be detected as described before. As seen in the table, the videos $V10$ and $V11$ have the best accuracy because the camera view used for recording them were closer to the ideal one explained in Section 3.1.1. It is obvious from the table, as the view angle of camera increases, fewer

piano keys are successfully located on the piano keyboard because of a drastic perspective view of the camera and the accuracy of pressed keys detection decreases.

Table 4.7: List of the sample videos recorded in different camera view angles. The column "Number of Keys" shows the number of visible white and black keys in the video and "Located Keys" shows the number of keys correctly located using keys detection.

Video ID	Angle (degree)	Number of Keys	Located Keys	TP #	FP #	FN #	Recall %	Precision %
V9	0	56	56	41	7	14	74.6	85.4
V10	30	59	56	42	3	21	66.7	93.3
V11	45	61	49	39	0	23	63.0	100
V12	60	61	39	24	14	35	40.7	64.1
V13	75	57	22	8	20	43	15.7	28.6

4.2.3 Hands Coverage

As stated in Section 4.1.1, covering the piano keyboard by the pianist's hands does not cause any problem in the keyboard detection as long as this coverage is not more than 60% of the piano keyboard (Figure 4.6). This percentage has been obtained by the experiments on many keyboard images and corresponds to the choice of the threshold used in the algorithm for detecting the keyboard. This is because the method used for detecting the keyboard in the software is based on the maximum brightness ratio of the lower one-third of the keyboard including the white keys and the maximum number of black keys in the upper two-thirds part. So, covering the major portion of the keyboard significantly decreases the brightness ratio in the area of white keys and the number of black keys, which may result in the failure of the keyboard detection.

The best background image is the one in which no hands cover the keyboard. If the pianist holds his/her hands on the keyboard during the whole video performance, the desired background image will never be extracted, which results in lower accuracy in the other stages. In addition, the basis of the keys detection algorithm is to locate the black keys on the background image and then estimate the white keys. If the background image includes the hands covering the black keys, keys detection cannot successfully locate all the black

keys on the keyboard. As a result, the adjacent white keys cannot be estimated either.

Hands coverage also affects pressed keys detection. It was explained in Section 3.2.3 that two different types of blobs may be generated in the Wkeys image (the difference image used for detecting the pressed white keys), the ones shaped by pressing the white keys and some extra blobs caused by the pianist's hands covering the keyboard. The algorithm considers only the upper quadrilateral of the white keys to avoid the incorrect detection of the hands as pressed keys because the hands are mostly present in the lower quadrilateral. However, in situations in which the hands move toward the upper part of the piano keyboard, they may be incorrectly detected as pressed keys. Moreover, the pressed keys under the hands cannot be seen by the camera or even the humans (Figure 4.6).

Figure 4.6: The pianist's hands covering about 60% of the piano keyboard.

4.2.4 Vibrations

Vibration of the camera or the piano keyboard is another limitation in the software. Once the quadrilateral of the keyboard is detected, its four points indicating the location of the keyboard are stored. Next, the keys on the keyboard are located inside the keyboard quadrilateral. The procedure of locating the keyboard and the keys is done only once in the first frame(s). The keyboard and the keys in the next video frames are located using the coordinates of the stored quadrilateral. If the camera or the piano vibrates or moves during the performance, the four points of the registered quadrilateral do not indicate the correct position of the keyboard anymore. As a result, the pressed keys in the next video

frames cannot be detected correctly based on their determined locations in the keyboard quadrilateral.

4.2.5 Other Issues

Sometimes, the pianist plays extremely fast pieces of music resulting in pressing the keys faster than the video frame rate. In this case, the camera cannot capture some of the changes caused by the keys pressed on the piano keyboard. As a result, they cannot be processed and detected in claVision.

There are some aspects in music that are related to the human or emotional or expressive side of music. Performing tempo rubato, dynamics, and **articulations**[5], etc. are some instances used for creating musical expression in a performance. Although there are some verbal and symbolic ways to indicate musical expression in music notation, they are imprecise and highly dependent on the player's interpretation [36, 69]. In Section 4.1.4 the effect of performing tempo rubato on music transcription performed by claVision was studied. The dynamics, which are created by increasing or decreasing the volume in playing the music, cannot be recognized and notated in the system because it does not consider the music audio. The articulations cannot be identified and transcribed in claVision because the soft and hard transitions in pressing the keys on the piano cannot visually be detected. In addition, none of the audio-based music transcription techniques is able to deal with this process precisely.

As explained in Section 2.2, piano pedals can be used to modify the sound of the played notes. Since the audio is ignored and the camera does not capture the pedals in the system these sound modifications cannot be detected.

[5] Articulation describes the transition or continuity on a single note, or between multiple notes. In other words, articulation indicates the manner of attacking or releasing the sound in a piece of music [36, 42, 69].

Chapter 5

Conclusion

In this thesis, a new vision-based system for automatic transcription of piano music named claVision was developed. The transcription process in this system is performed only by visually analyzing the piano keyboard using a digital camera (e.g., webcam) located at top of the piano keyboard. Since claVision does not consider the audio, none of the difficulties corresponding to audio-based music transcription techniques is present anymore. The idea of claVision was inspired by the famous composer Beethoven who was deaf in the last decade of his life. Due to this disability, he was forced to rely on the visual relationship between his fingers and the keys on the piano to be able to play and compose music. In addition to the applications of music transcription, claVision can be used for other purposes, including automatic improvised music transcription, different educational purposes such as teaching piano to hearing impaired people, and visual performances in live concerts and musical competitions. The software works with all musical keyboards, has a high accuracy of transcription and very low latency in the real-time process of music transcription, and can deal with different lighting conditions.

A variety of computer vision techniques and algorithms was used to implement claVision. The algorithms were described in four stages including keyboard registration, illumination normalization, pressed keys detection, and music transcription. The effectiveness of the system in different steps was evaluated based on accuracy and processing time. The locations of the keyboard in all sample images were successfully detected with a maximum and an average processing time of 1100 ms and 691.9 ms. The keys detection process had

a very high accuracy of 95.2% and a very low maximum processing time of 20 ms. The background update process was done in a maximum and an average time of 53 ms and 6.1 ms. The processing times for this stage did not affect the real-time processing because it was done in a separate thread. According to the experimental results, issues such as varying illumination conditions and noise were satisfactorily dealt with using the illumination normalization algorithm.

This algorithm has a maximum and an average processing time of 6 ms and 1.8 ms, and did not cause any significant latency for real-time processing in claVision. It was demonstrated that the software has a very high accuracy in pressed keys detection with recall and precision rates of 97.5% and 97.4%. There was a very low latency of 6.6 ms for the pressed keys detection stage. The synthesized MIDI file was accurately produced based on the given musical information and it sounded the same as the played music. However, there were some notation errors corresponding to the duration of the notes and rests and their locations in the measures, which resulted from the amount of tempo rubato (flexibility in time) performed by the pianist. Most of these errors can be corrected using different tools developed for manipulating MIDI files and music scores.

5.1 Improvements

A number of limitations in the software such as drastic lighting changes, inappropriate camera view angle, hands coverage of the keyboard, vibrations of the camera or the piano, etc. was analyzed in this thesis.

- There are many different approaches that can be used for dealing with lighting issues in image processing. However, because they prevented real-time processing in the system, using the more computationally intensive techniques was avoided.

- Another issue was the unsuccessful detection of the pressed keys that were directly below the camera. One proposed solution to deal with this is to use a second camera

with a different view over the piano keyboard. The second camera can also improve pressed keys detection when the first camera is in a drastic perspective view.

- One way to ignore the hands covering the keyboard and the keys is to use skin colour algorithm in order to identify the pixels related to the hands. However, it may not work with different skin colours.

- In order to deal with the vibrations of the camera or the keyboard, there are some methods used for adjusting and/or tracking the objects (e.g., keyboard and hands) in the video images while they vibrate or move during the video. This limitation can also be solved by re-performing the keyboard registration process in order to relocate the keyboard and the keys on it. However, more computation is needed in each video frame, and real-time transcription may not be performed anymore.

- The effect of piano pedals on modifying the sound of the played music can also be visually analyzed using a second camera located at the base of the piano. The feet and pedals movements can be processed in order to detect which pedal is being pressed.

5.2 Future Plans

There is a number of potential directions to extend claVision:

- in addition to recognizing the musical notes, extend claVision to also identify fingering information, in order to detect which hand and finger is used for which pressed key. This will be very useful for teaching and learning to play piano;

- incorporate other technologies such as depth sensors (e.g., Microsoft KINECT) to improve the accuracy of the transcription, especially, fingering information extraction;

- implement a music recognizer that can not only transcribe the music played, but also retrieve bibliographical information (e.g., artist, lyric, etc.) of the piece of music

played from online databases;

- implement a mobile version of claVision to make it more accessible, portable, and easily usable for the average users. Automatic transcription of piano music can be done using cellular phones and their built-in digital cameras;

- develop a service-based claVision, for example, using Microsoft Azure, which is a cloud-based platform. Users may use claVision online without installing it on their computers;

- combining the idea of VAMT with audio processing technologies to provide a robust multi-modal system for music transcription.

The idea developed for claVision can also be incorporated into other applications:

- the video processing technology of claVision can be used to implement a separate automatic piano tuner. Currently, most piano tuner applications require the user to first enter which note is about to be played, before the applications "listen" and analyze the played note. This idea can be used to automate this process as the software can "see" which key is played;

- Some research is being conducted to determine how video processing techniques can be applied to other musical instruments.

Bibliography

[1] DoReMIR Music Research AB. Scorecloud studio. http://www.scorecloud.com. [Online; accessed 27-September-2014].

[2] MIDI Manufacturers Association. Musical instrument digital interface. http://www.midi.org/. [Online; accessed 15-October-2014].

[3] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[4] M. Ben. *Principles of Concurrent and Distributed Programming, Second Edition*. Addison-Wesley, second edition, 2006.

[5] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41:407–434, 2013.

[6] A. Burns and M. M. Wanderleym. Visual methods for the retrieval of guitarist fingering. pages 196–199, 2006.

[7] MuseScore BVBA. MuseScore. http://www.musescore.org/. [Online; accessed 27-September-2014].

[8] N. Collins. A comparison of sound onset detection algorithms with emphasis on psycho-acoustically motivated detection functions. *Journal of the Audio Engineering Society*, pages 28–31, 2005.

[9] R. H. Dorf. *Electronic Musical Instruments*. New York, Radiofile, 1968.

[10] J. S. Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003.

[11] K. Dressler. Multiple fundamental frequency extraction for MIREX 2012. *Music Information Retrieval Evaluation eXchange*, 2012.

[12] C. Frisson, L. Reboursire, W. Chu, O. Lhdeoja, J. Mills Iii, C. Picard, A. Shen, and T. Todoroff. Multimodal guitar: Performance toolbox and study workbench. *QPSR of the numediart research program*, 2(3):67–84, 2009.

[13] J. Gillespie. *Five Centuries of Keyboard Music*. Wadsworth Publishing Company, Belmont, California, 1921.

[14] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[15] D. O. Gorodnichy and A. Yogeswaran. Detection and tracking of pianist hands and fingers. *The 3rd Canadian Conference on Computer and Robot Vision*, page 63, 2006.

[16] P. Heckbert. Projective mappings for image warping. Master's thesis, University of California, Berkeley, 1989.

[17] P. V. C. Hough. Method and means for recognizing complex patterns. http://www.google.de/patents/US3069654, 1962. US Patent 3,069,654.

[18] D. M. Huber. *The MIDI manual*. Carmel, Ind., USA : Sams, 1991.

[19] Microsoft Imagine Cup. 2014 world finalists: Innovation. `https://www.imaginecup.com/Custom/Index/2014Finalists_Innovation`. [Online; accessed 28-October-2014].

[20] Ableton Inc. Ableton Live 9. `https://www.ableton.com`. [Online; accessed 27-September-2014].

[21] Apple Inc. Logic pro x. `https://www.apple.com/ca/logic-pro`. [Online; accessed 27-September-2014].

[22] Avid Technology Inc. Sibelius. `http://www.sibelius.com`. [Online; accessed 27-September-2014].

[23] Innovative Music Systems Inc. intelliScore. `http://www.intelliscore.net`. [Online; accessed 27-September-2014].

[24] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. volume 1, pages 343–356, 1996.

[25] B. L. Jacob. Algorithmic composition as a model of creativity. *Organised Sound*, 1:157–165, 12 1996.

[26] F. Jean and A. B. Albu. The visual keyboard: Real-time feet tracking for the control of musical meta-instruments. *Signal Processing: Image Communication*, 23(7):505–515, 2008.

[27] A. Klapuri. Automatic music transcription as we know it today. *Journal of New Music Research*, 33(3):269–282, 2004.

[28] A. Klapuri and M. Davy. *Signal Processing Methods for Music Transcription*. Springer-Verlag, New York., 2006.

[29] H. E. Korth. Method and device for optical input of commands or data, 1998. US Patent (5767842).

[30] Akoff Sound Labs. Akoff music composer. http://www.akoff.com. [Online; accessed 27-September-2014].

[31] C. Li and R. Hu. Piano hero with virtual keyboard. Technical report, Electrical and Computer Engineering at Cornell University, 2013. [Online; accessed 01-September-2014].

[32] Synthesia LLC. Synthesia piano hero. http://www.synthesiagame.com. [Online; accessed 01-September-2014].

[33] R. C. Maher. Evaluation of a method for separating digitized duet signals. *JAES*, 12(38):956–979, 1990.

[34] J. A. Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, 4(1):32–38, 1977.

[35] Y. Motokawa and H. Saito. Support system for guitar playing using augmented reality display. *IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 243–244, 2006.

[36] R. Nelson and C. J. Christensen. *Foundation of Music: A Computer-Assisted Introduction*. Wadsworth Publishing Company, Belmont, California, 2000.

[37] P. Neubcker. Melodyne. http://www.celemony.com. [Online; accessed 27-September-2014].

[38] A. Oka and M. Hashimoto. Marker-less piano fingering recognition using sequential depth images. *Frontiers of Computer Vision, (FCV), 2013 19th Korea-Japan Joint Workshop on*, pages 1–4, 2013.

[39] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[40] D. Padua. *Encyclopedia of Parallel Computing*. Springer, 2011.

[41] M. Paleari, B. Huet, A Schutz, and D. Slock. A multimodal approach to music transcription. In *15th IEEE International Conference on Image Processing*, pages 93–96, 2008.

[42] C. V. Palisca. *The New Grove Dictionary of Music and Musicians*. Macmillan Publishers, London, 2001.

[43] P. Peeling and S. Godsill. Multiple pitch estimation using non-homogeneous Poisson processes. *IEEE Journal of Selected Topics in Signal Processing*, 6(5):1133–1143, 2011.

[44] A. Pietrosanto, Paolillo A., L. D'Acunto, L. Bolivar, M. Ribas, and R. Vidal. Virtual piano project. Technical report, Universit degli Studi di Salerno (Italy), 2013.

[45] M. Piszczalski and B. A. Galler. Automatic music transcription. *Computer Music Journal*, 4(1):24–31, 1977.

[46] G. Quested, R. Boyle, and K. Ng. Polyphonic note tracking using multimodal retrieval of musical events. In *Proceedings of the International Computer Music Conference (ICMC)*, 2008.

[47] L. Reboursière, C. Frisson, O. Lähdeoja, J. Mills Iii, C. Picard, and T. Todoroff. MultimodalGuitar: a toolbox for augmented guitar performances. In *Proceedings of the New Interfaces for Musical Expression++ (NIME++)*, 2010.

[48] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer., 2011.

[49] J. Richter. Garbage collection: Automatic memory management in the Microsoft .NET Framework. `http://msdn.microsoft.com/en-us/magazine/bb985010.aspx`, November 2000. [Online; accessed 01-September-2014].

[50] H. Samet and M. Tamminen. Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):579–586, 1988.

[51] J. G. Savard. The size of the piano keyboard. `http://www.quadibloc.com/other/cnv05.htm`. [Online; accessed 10-October-2014].

[52] J. Scarr and R. Green. Retrieval of guitarist fingering information using computer vision. *25th International Conference of Image and Vision Computing New Zealand (IVCNZ)*, pages 1–7, 2010.

[53] A. Silberschatz and P. B. Galvin. *Operating System Concepts*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[54] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, 2003.

[55] M. Sotirios and P. Georgios. Computer vision method for pianist's fingers information retrieval. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '08, pages 604–608. ACM, 2008.

[56] P. Suteparuk. Detection of piano keys pressed in video. Technical report, Department of Computer Science, Stanford University, April 2014.

[57] Sydeux. Ableton Live 9 Audio to MIDI vs. Melodyne. `https://www.youtube.com/watch?v=25MqclpXI7k`, March 2013. [Online; accessed 27-September-2014].

[58] S. Toub. Toub.sound.midi (.NET Framework for working with MIDI events). `https://code.google.com/p/ghgame/source/browse/trunk/Toub.Sound.Midi/?r=192`. [Online; accessed 01-September-2014].

[59] G. Tzanetakis and G. Essl. Automatic musical genre classification of audio signals. In *IEEE Transactions on Speech and Audio Processing*, volume 10, pages 293–302, 2001.

[60] M. Vaidyanathan. Midi sheet music. http://midisheetmusic.sourceforge.net, July 2013. [Online; accessed 01-September-2014].

[61] Ludwig van Beethoven website. Ludwig van Beethoven. http://www.lvbeethoven.com. [Online; accessed 2-October-2014].

[62] AForge.NET Website. Aforge.NET Framework. http://www.aforgenet.com/framework. [Online; accessed 01-September-2014].

[63] NCH Software Website. Twelvekeys music transcription software. http://www.nch.com.au/twelvekeys. [Online; accessed 27-September-2014].

[64] Seventh String Website. Transcribe! http://www.seventhstring.com. [Online; accessed 27-September-2014].

[65] Wikipedia. Ludwig van Beethoven — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Ludwig_van_Beethoven&oldid=625402621. [Online; accessed 2-October-2014].

[66] Wikipedia. Projection keyboard — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Projection_keyboard&oldid=611449534. [Online; accessed 27-September-2014].

[67] J. Yang and Y. Xu. Hidden Markov model for gesture recognition. Technical report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1994.

[68] C. Yeh. *Multiple fundamental frequency estimation of polyphonic recordings*. PhD thesis, Universite Paris VI - Pierre et Marie Curie, France, 2008.

[69] J. Yudkin. *Understanding Music*. Boston University, 2012.

[70] B. Zhang, J. Zhu, Y. Wang, and W. K. Leow. Visual analysis of fingering for pedagogical violin transcription. pages 521–524. ACM, 2007.