SAPIENZA
UNIVERSITÀ DI ROMA

Keyrtual: A Lightweight Mixed Reality Musical Keyboard For Smartphone

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Computer Science

Andrea Princic
ID number 1837592

Advisor
Prof. Luigi Cinque

Academic Year 2023/2024

Thesis defended on October 22, 2024
in front of a Board of Examiners composed by:

Paola Velardi (chairman)
Luigi Cinque
Daniele Friolo
Fabio Galasso
Marco Raoul Marini
Eugenio Nerio Nemmi
Salvatore Pontarelli

Keyrtual: A Lightweight Mixed Reality Musical Keyboard For Smartphone
Master thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: princic.1837592@studenti.uniroma1.it

Contents

Abstract

Hand gesture recognition has become a critical area of study, particularly with the increasing prevalence of virtual and augmented reality technologies. This thesis presents the development of Keyrtual, a lightweight mixed reality musical keyboard application for smartphones. The application allows users to play a virtual musical keyboard by recognizing hand gestures over a physical keyboard drawn on paper. Leveraging the capabilities of modern smartphone cameras and augmented reality frameworks and tools like MediaPipe, this system eliminates the need for external hardware, such as tracking gloves or depth sensors, making it accessible and portable. The project explores the integration of OpenCV and MediaPipe for image processing and hand detection in a mixed reality application developed in Unity, and the possibility to involve depth-sensing technologies like LiDAR, Time-of-Flight and stereo vision sensors available in modern smartphones. By reproducing musical notes based on finger positioning in the physical world, the application offers an immersive, educational tool that combines music and technology. The study also discusses the system architecture, challenges in real-time hand tracking, and the potential future enhancements in the realm of mobile AR applications.

Ringraziamenti

Ringrazio la Fondazione TIM per aver supportato finanziariamente il progetto tramite l'iniziativa Call for IDEAs – Inclusione Sociale, e la Fondazione Anna Maria Catalano per averci guidato durante tutte le fasi del concorso e per aver gestito la parte burocratica e sociale del progetto.
Ringrazio il professor Luigi Cinque e Marco Raoul Marini per aver creduto per primi nel progetto, e per averci aperto le porte del VisionLab.
Ringrazio tutti i ragazzi e le ragazze del VisionLab, che ci hanno fatto trovare un ambiente accogliente e amichevole oltre ogni aspettativa.
Ringrazio Valerio Venanzi, il mio partner in questa avventura che è cominciata come l'ennesimo progetto di coppia da fare insieme per l'ennesimo esame.

Chapter 1

Introduction

1.1 Aims and scope

Hand gesture recognition is becoming an increasingly important field, thanks in part to the advent and spread of virtual reality (VR) and augmented reality (AR) devices. In these scenarios, the ability to use only their hands allows the user to feel completely immersed in the experience, without necessarily having to use or wear additional devices such as controllers or tracking gloves. Nevertheless, augmented reality and virtual reality devices often make use of special hardware dedicated specifically to hand detection and gesture recognition.

The purpose of this work is to create a mixed reality application for smartphones that relies only on the hardware available on the smartphone itself, and allows the user to use their hands to play on a semi-virtual piano, drawn on paper, reproducing the sound of the notes played.

The main target group of the application is primary and secondary school students. For them, the application would serve as a first approach to learning the piano, allowing them to experience first-hand how it works and what it means to play, but without the initial investment needed to buy the physical instrument, nor all that relates to its transport, storage and maintenance.

1.2 Importance of music and its education

Music has always been an important component in the culture of human civilizations. It is one of the oldest and most widespread art forms, traces of which can be found as far back as the Paleolithic [43], and has enabled entire peoples to preserve and pass on traditions, share ideas and unite people.

Music education has a rich history that can be traced back to ancient Greece, where melody was combined with poetry and dance, creating an almost inseparable whole. Here, music was considered a gift from the muses, gods of the arts, and was thought to have the power to influence people's thoughts and actions in various ways, and different instruments were associated with different effects on the emotions and character of individuals [77].

According to Plato, musical education starts from early childhood, when mothers should sing to and dance with their children. They, in fact, unable to speak

and understand complex thoughts, can nevertheless appreciate melody and singing, through which they can be educated and instructed. The period of formal education, again according to Plato, should start at the age of 6 and last until 20, focusing on dance, music and gymnastics [77].

In Europe, music is part of the school curriculum in almost all countries. According to a research carried out by *meNet* from 2006 to 2009, it emerged that in most European countries, music is treated similarly in similar age groups: first, it is offered as an optional subject in pre-schools, and becomes a compulsory subject during primary school in all countries. It remains compulsory in almost all countries at least up to the age of 14, which corresponds more or less to middle school.

The Czech Republic, Hungary, the Netherlands and Slovakia have the longest compulsory period, which goes from the age of 5 to the age of 18.

In Italy, music education in schools has a history of more than a hundred years [11]. Starting out as an optional subject called "Canto corale" ("Choral Singing") in 1888, music was regarded as a subject of low importance, more useful for recreation than education, for 35 years.

It was in 1923 that the subject "Canto" ("Singing") was elevated to a compulsory subject for elementary schools, in the group of "Insegnamenti artistici" ("Art Teaching") along with "Disegno spontaneo" ("Spontaneous Drawing"), located first in the order of subjects.

In the 1959–60 school year, the experiment to make the teaching of music compulsory in all three years of middle school was launched. The experiment failed and we will have to wait until 1977 to see the subject "Educazione musicale" ("Music Education") become compulsory in all three years of middle school.

Finally, it is with a reform in 2010 that a high school curriculum focusing on the teaching and practice of music, called "Liceo musicale" ("Music High School"), is established.

1.3 VR and AR for music education

In recent years, especially during and after the COVID–19 pandemic, there has been an increase in the use of technology in schools for management and educational purposes. Among other technologies that have entered the world of education, virtual reality and augmented reality represent two innovative technological tools that can transform the way teaching and learning are conducted in schools. Both technologies offer new opportunities to enrich the educational experience, making learning more interactive, immersive and engaging, and giving students the opportunity to experience authentic situations and apply their knowledge in practice.

Music education and learning an instrument are challenges for both the student and the teacher, also due to budget cuts and little emphasis on arts subjects in schools. Using VR or AR in music education can be an alternative approach to improve both the learning and teaching experience, and to bring attention back to this too often undervalued subject.

Augmented reality and mixed reality, in particular, have the added advantage over virtual reality of bringing the physical and digital worlds into direct contact through objects that exist in both forms and enable more realistic experiences with

more immediate feedback.

The potential advantages of augmented reality teaching methods over traditional methods include: learning and practicing rhythmic skills, being able to play together while being apart, overcoming stage fear, and training STEAM (Science, Technology, Engineering, Art and Math) and acoustic skills [71]

Chapter 2

State of the art

In this chapter, the current state of the art in computer vision is presented, with a focus on depth vision and hand detection, both on mobile and non-mobile devices, especially applied to the fields of augmented reality, mixed reality and virtual reality.

2.1 OpenCV

OpenCV, which stands for Open Source Computer Vision Library, is an open source library dedicated to image processing and computer vision. Originally developed by Intel in 1999 [15], OpenCV has grown considerably over the years, becoming an essential tool in various fields, from robotics to augmented reality, from surveillance to aerial imaging [15].

2.1.1 Origin and history

The story of OpenCV began in the late 1990s, when Intel started the project as part of an initiative to promote the use of CPUs in artificial vision applications. In 2000, OpenCV was released to the public in its alpha version. It then evolved through 5 beta versions between 2000 and 2005, finally reaching its first official release version 1.0 in 2006 [15]. The library is now at version 4.10.0, and on GitHub it has 63 releases and almost 77000 stars.

2.1.2 Library overview

OpenCV is written in `C` and `C++` [15], but offers bindings for other programming languages such as `Python`, `Java`, `C#` and `MATLAB`, making it accessible to a wide range of developers. The library contains over 2500 optimised algorithms covering different areas of image processing and computer vision, including:

- Object detection
- Facial recognition
- Gesture recognition
- Image segmentation

- Motion tracking
- 3D reconstruction
- Video processing

2.1.3 Main applications

Image and Video Processing One of the most common uses of OpenCV is image and video processing. The library provides tools for manipulating, transforming and analysing images in various formats. These tools include noise reduction filters, geometric transformations, edge detection, and many other image enhancement techniques.

Object and Face Recognition OpenCV is widely used for object and face recognition [42, 14, 69]. Algorithms such as Haar Cascade, LBP (Local Binary Patterns) and OpenCV's DNN (Deep Neural Network) technology make it possible to detect and recognise faces and other objects in images with high accuracy. These algorithms can be used in security applications, such as surveillance systems, and in social media apps for adding face filters.

Augmented Reality In the field of augmented reality, OpenCV can be used to render digital information on top of the real world. OpenCV's tracking and sensing algorithms enable the detection and tracking of physical objects, such as AR markers [47, 68, 64], which serve as reference points for overlaying virtual content. This capability is crucial for creating immersive and interactive experiences.

Robotics In robotics, OpenCV is used to equip robots with "vision" [23, 35, 78]. Through the library and its functions listed above, robots can perceive and understand their surroundings, identify obstacles, detect visual signals and follow predefined paths. These capabilities are essential for the development of autonomous robots and drones.

Medical Applications OpenCV also has applications in the medical sector, where it is involved in the analysis of medical images such as X-rays [20], MRI [60, 5, 25] and ultrasound images [3, 38, 37, 85]. The automatic analysis of medical images helps doctors diagnose diseases more accurately and quickly.

2.2 Depth vision

Depth sensors are devices used to measure the distance between objects and the sensor itself. This technology has applications in many fields, including robotics, augmented reality (AR), virtual reality (VR), and computer vision in general. Depth sensors can be classified according to their principle of operation, with the main types including infrared sensors, Time-of-Flight (ToF) sensors, and stereo vision sensors.

2.2.1 Infrared sensors

Infrared sensors project a pattern of structured infrared light onto the scene and capture the reflected image on the infrared receiver. By analysing the distortion of the pattern caused by objects in the scene, the sensor is able to compute their distance. A classic example of this type of sensor is the Microsoft Kinect.

Infrared technology is particularly effective in indoor environments with controlled lighting conditions, but may be less accurate outdoors or in the presence of strong light sources, and is also subject to measurement errors that may depend on the material of the objects exposed to the sensor such as smooth and shiny surfaces [39].

2.2.2 Time-of-Flight sensors

Time-of-flight sensors measure the time it takes for emitted light to reach an object and return. This principle is similar to radar, but uses light instead of radio waves. An example of a ToF sensor is Microsoft's Kinect v2.

ToF sensors are known for their high accuracy and ability to operate effectively in outdoor environments, unlike infrared sensors. However, even these can be affected by the reflectivity of objects and environmental conditions such as fog or rain.

2.2.3 Stereo vision sensors

Stereo vision sensors consist of two or more cameras that are used simultaneously to capture images of the same scene from slightly different angles, similarly to how human vision works. The depth in the scene is computed by analysing the differences between the images captured by the different cameras.

Stereo vision sensors are very versatile and can operate in a wide range of environmental conditions, but require more hardware and more processing power to analyse the captured images.

2.2.4 Leap Motion Controller

The Leap Motion Controller is a device for hand recognition that uses infrared sensors to accurately track the position and movement of hands in three dimensions. The Leap Motion sensor emits infrared light that is reflected by the user's hands and captured by two infrared cameras. The Leap Motion software processes these images to create a three-dimensional map of the hands and fingers, enabling natural and intuitive interaction with VR and AR applications.

The Leap Motion is particularly appreciated for its precision and responsiveness, making it ideal for applications requiring fine control of hand movements. In the field of augmented reality and virtual reality, Leap Motion is widely used due to its high compatibility with leading VR and AR viewers such as the Oculus Rift, HTC Vive, Pico and Varjo.

This versatility makes it a popular option for developing applications in a variety of fields beyond the gaming and entertainment industry, including the medical field, in particular physical and cognitive rehabilitation [65, 13], social skills development

for people with autism [22, 28], education and training in various disciplines [53, 21, 63, 55], and sign language recognition [19, 50].

The Leap Motion Controller is the device used in the very first prototype of this project [10].

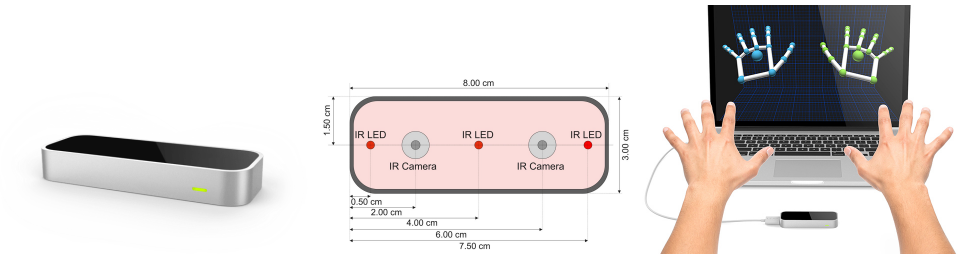


Figure 2.1. Leap Motion Controller, sensors [82] and usage

2.2.5 Kinect

The Microsoft Kinect is probably one of the best known and most widely used depth sensors. It exists in two versions, which use different technologies to measure depth. The first version of the Kinect used structured infrared light technology to create a depth map with a resolution of 640×480 pixels and a frequency of 30 FPS [57], while the second version, the Kinect v2, uses a ToF sensor with a resolution of 512×424 pixels [57] and a frequency of 30 FPS [34].

The Kinect was originally designed for the Xbox 360 gaming console, but quickly became a popular tool in research and development of interactive applications due to its ability to track human body movements in real time and the dual presence of an infrared depth sensor and an RGB camera.

The Kinect has found applications in many areas beyond the video game industry, including education [41, 12, 44], robotics [6, 4], and surveillance [30, 33].

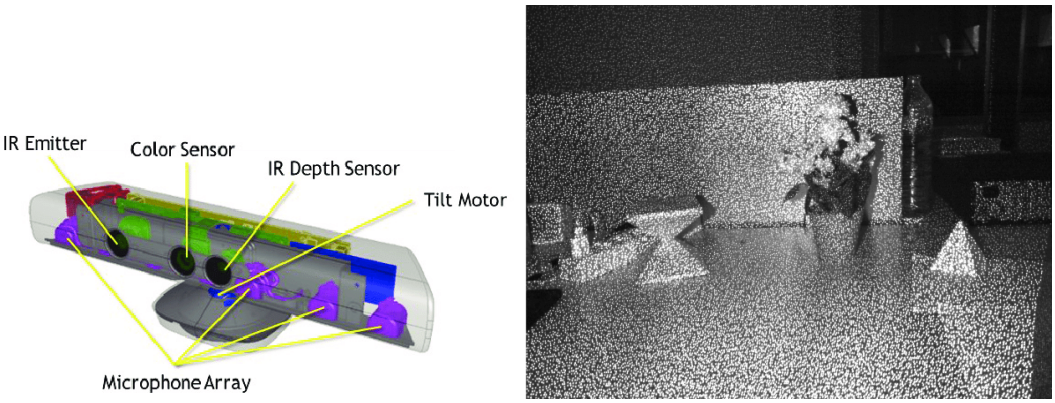


Figure 2.2. Microsoft Kinect components [27] and structured light pattern [81]

2.2.6 RealSense

Intel's RealSense sensor line uses a combination of technologies, including stereo vision and ToF, to provide high-resolution depth data. RealSense sensors are designed to be integrated into a wide range of devices, from drones to robots to AR and VR devices, thanks in part to the robust SDK developed by Intel, which facilitates integration and custom application development.

RealSense sensors capture both depth images, with a resolution of up to 1280×720 pixels and a frame rate of 90 FPS [73], and RGB images, with a resolution of up to 1920×1080 pixels and a frame rate of 30 FPS [73].

The ability of RealSense sensors to capture detailed depth maps makes them ideal for 3D mapping applications even in outdoor environments [16]. Among other fields of application, RealSense sensors are widely used in face detection and its applications such as facial expression recognition [61], 3D face reconstruction [87], and face detection [18].

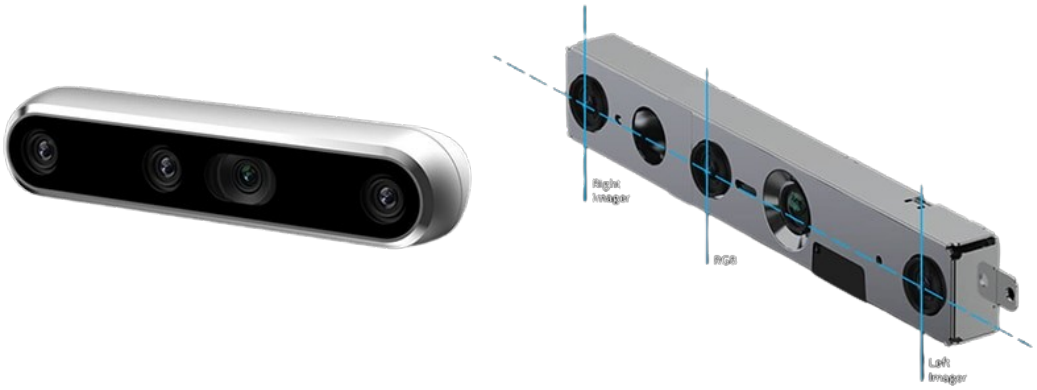


Figure 2.3. RealSense components

2.3 Depth vision on smartphone

With the advent of smartphones equipped with powerful sensors and advanced cameras, depth sensing has become accessible and usable in numerous applications. It is mainly used in smartphones as an aid to the camera, to capture additional information about the scene being framed and to improve sharpness and focus in photographs, but it can also be used outside of simple photography apps, such as in augmented reality apps.

2.3.1 Depth sensors on smartphone

Modern smartphones are often equipped with depth sensors that exploit different technologies, such as LiDAR (Light Detection and Ranging) and ToF (Time-of-Flight). These sensors calculate the distance to objects by emitting a light or laser pulse and measuring the time it takes for it to return to the sensor after hitting an object.

LiDAR It uses laser pulses to create three-dimensional maps of the environment. It is particularly useful in low light conditions and provides highly accurate depth data. It can also measure the direction and intensity of the reflected signal, providing three-dimensional information on the shape and structure of objects. In smartphones, LiDAR greatly enhances AR capabilities, enabling more realistic and interactive applications. This type of sensor can be found on Pro and Pro Max models of iPhone since the iPhone 12.

ToF It works similarly to LiDAR, but uses infrared light instead of a laser pulse to measure distance. This type of sensor is often cheaper and less accurate than LiDAR and does not provide information about the shape of objects, but is still effective for many applications, including indoor mapping and object detection.

2.3.2 ARFoundation for AR applications

ARFoundation is a Unity framework that enables developers to create cross-platform AR applications. By integrating the functionality of Apple's ARKit and Google's ARCore, ARFoundation makes it possible to develop a single application that can run on both iOS and Android devices.

Cross-platform One of the main features of ARFoundation is its ability to abstract the API specifications of ARKit and ARCore, allowing developers to write code once to run applications on both platforms. This reduces development time and simplifies code maintenance.

Detection and tracking ARFoundation supports tracking the position and orientation of the device, detecting horizontal and vertical surfaces, anchoring virtual objects in real space and detecting 3D images and objects, all essential functionalities for creating immersive and interactive AR experiences.

2.3.3 Applications

The use of sensors to detect depth images with a smartphone is applied in various fields, from object detection and tracking, to health monitoring and assisting visually impaired people.

In [58], the smartphone is used as a real-time sign language translator. Depth images collected by the smartphone's depth sensor are pre-processed and then classified by a MobileNet+LSTM deep-learning model directly on the mobile device.

Another tool of assistance for visually impaired people can be found in [59], where a new solution for spatial orientation and navigation is proposed. In this work the authors make use of techniques for distance detection and object recognition from RGB images, using the MiDaS 2 Lite model to extrapolate a depth map and the MobiNet3 neural network for object classification.

Similar to the previous work, an object recognition system is proposed in [88], but this time based on the depth sensor of the smartphone and not on RGB images.

A remarkable result was obtained in [49], where the TrueDepth scanner of an iPhone 12 Pro was used to obtain a depth scan of a stoma model in order to detect

its contours, achieving a remarkable accuracy (<2mm). In fact, it turns out that the TrueDepth scanner has a higher accuracy and density of infrared dot projection than the LiDAR sensor.

However, it should be noted that the TrueDepth scanner is the one used for FaceID and is located on the front face of the smartphone, which makes it unusable together with the camera or any other sensor located on the back face of the smartphone.

2.4 Hand detection

With the advent of the era of augmented reality and virtual reality, the demand for virtual or semi-virtual interfaces that do not require physical hardware to be used is growing more and more, and one of the most common methods of using these interfaces is through the use of the hands, the most natural way we have of interacting with our surroundings.

In order to develop these types of interfaces so that they are reliable and comfortable to use, there is a need for precise methods for detecting hands in a two- or three-dimensional space, which can understand their positioning and interpret their gestures.

2.4.1 MediaPipe

MediaPipe is an open-source framework developed by Google for multimodal processing of data streams. Its flexibility and power make it an ideal tool for developing virtual reality and augmented reality applications. MediaPipe allows the creation of modular pipelines that can be used in various applications such as facial recognition, object tracking, hand detection and many others.

Origin and history

MediaPipe started as an internal solution to manage and standardise the various computer vision algorithms used in their products. In 2019 [46], Google decided to make MediaPipe an open-source project, allowing the developer community to contribute and exploit the potential of the platform. Since its release, MediaPipe has seen rapid adoption due to its ability to simplify development.

Components and architecture

MediaPipe's architecture is based on a modular design that allows developers to combine various components to create customised pipelines. The main components of MediaPipe include:

- *Calculator:* it is the fundamental unit of calculation that perform specific operations on data. Each calculator receives an input, processes the data and produces an output.
- *Graph:* it is a network of calculators that defines the flow of data. The graph determines how the data is transformed and transferred between the various calculators.

- *Packet:* it is the data transport unit in the graph. Packets may contain any type of data, such as images, videos or processing results. A particular type of packet is the side packet, a packet without a timestamp that can be used to transport data that remain constant, unlike a stream that represents a flow of data that changes over time.

Pipeline A MediaPipe pipeline is essentially a graph of calculators. This modular design allows the construction of highly customisable pipelines. For example, a hand detection pipeline may include calculators for image pre-processing, hand detection, tracking of key points and post-processing of results. MediaPipe provides a number of predefined calculators, but developers can also create their own calculators [26] for specific application needs. The main components of MediaPipe are:

- *Input Streams:* these are sources of data such as video feeds, audio inputs, or sensor data.
- *Processing Nodes:* these nodes perform specific tasks like image transformation, feature extraction, and machine learning model inference.
- *Output Streams:* the processed data is outputted for use in applications, such as displaying detected hand gestures or controlling a robotic arm.

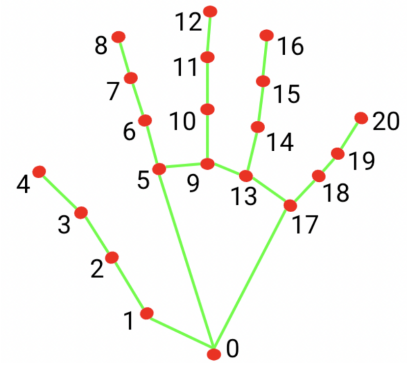
Hand recognition One of the most advanced and popular features of MediaPipe is hand recognition [86]. This feature is designed to be accurate and fast, working in real time even on mobile devices [76, 67]. MediaPipe's hand recognition pipeline consists of several steps:

- *Hand Detection:* it uses a machine learning model to identify the presence of hands in an image. This model is optimised to identify hands in various positions and orientations.
- *Landmarks Localization:* once the hand is detected, another machine learning model tracks 21 specific landmarks of the hand, which include the positions of the finger and palm joints in 2.5D. The landmarks used by MediaPipe are shown in Figure 2.4
- *Post-Processing:* landmark data can be further processed for specific applications, such as gesture recognition or interaction with virtual user interfaces.

MediaPipe's hand detection model has been trained on a large dataset to ensure high accuracy and robustness. The pipeline is optimised to run in real time, making it suitable for interactive applications such as augmented reality games and gesture-based controls.

2.4.2 ManoMotion

ManoMotion is an advanced hand gesture recognition technology that fits into the field of augmented reality and virtual reality. Founded in 2015, ManoMotion's main goal is to provide intuitive and natural interaction between humans and digital devices through real-time detection of hand movements.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 2.4. MediaPipe Hand Landmarks

Features ManoMotion offers several advanced features that make its hand gesture detection technology particularly useful:

- *3D movement recognition:* the technology is able to track hand movements in three dimensions, detecting not only the position, but also the depth and orientation of the hands.
- *Complex gesture recognition:* ManoMotion can recognise a wide range of complex gestures, including finger movements, grasps and releases, rotations and more.
- *AR and VR integration:* ManoMotion is designed to easily integrate with leading AR and VR development platforms, such as Unity and AR Foundation, facilitating adoption by developers.
- *Real-time analysis:* ManoMotion's algorithms process data in real time, ensuring a smooth interaction with no perceptible delays.
- *Compatibility with mobile devices:* one of the main innovations of ManoMotion is the ability to operate on mobile devices, using standard cameras built into phones and tablets.

Smartphone The way ManoMotion works on a smartphone is particularly impressive: it relies on a remote computing system in which the smartphone continuously sends the camera's video stream to the ManoMotion server, which then performs hand and gesture recognition and sends them back to the smartphone in real time.

This method involves shifting the computational load from the smartphone to the ManoMotion servers, leaving the smartphone with the sole task of transmitting the video stream to the server. ManoMotion certainly requires a stable and fairly fast connection (note that the frames sent by the smartphone are not in high resolution anyway), but it results in a huge reduction in computational load on the part of the smartphone.

This makes ManoMotion particularly suitable for the development of educational applications, thus intended to be in a classroom, where a school Wi-Fi could be

available for use. An example is the work done in [84], in which ManoMotion is used to create an educational application for chemistry. This application uses real-time touchless interaction for kinesthetic and machine learning with a dual purpose: teacher and student evaluator.

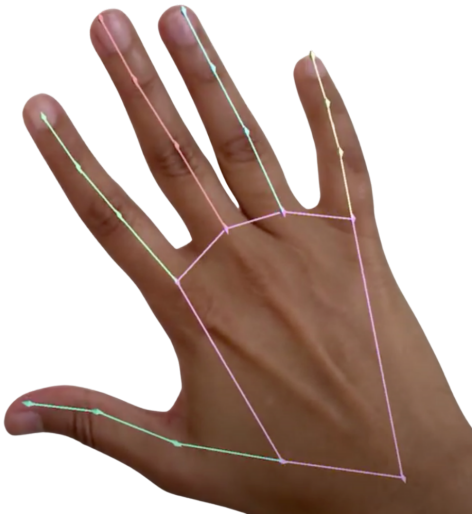


Figure 2.5. ManoMotion hand skeleton

2.5 Related work

There are many software and technologies that bring music into virtual reality or augmented reality applications, from systems for playing virtual or semi-virtual instruments, to virtual tutors evaluating the musical performance of a beginner musician.

2.5.1 Virtual and semi-virtual instruments

One of the most important features of VR and AR is to make the user interface with something that does not exist, but at the same time is visible in a virtual or semi-virtual environment, with even the possibility of giving feedback visual or tactile feedback in response to an action.

In [32] Cyber Composer is proposed, an interactive cyber instrument to enable both musicians and music laypersons to dynamically control the tonality and the melody of the music that they compose through hand motion and gestures. This system is empowered by the embedding of music theories to ensure the music generated makes musical sense in the perspective of musicians.

Two sensors are used to track hand movements: a pair of CyberGlove and a Polhemus FASTRAK 3D Position Tracker. The CyberGlove provides high-performance hand measurements and real-time motion capture thanks to bend and flexion sensors on fingers, palm, and wrist. A Polhemus sensor is attached to each glove, to provide 3D positioning and orientation information.

The software consists of four components: music interface, CyberGlove interface, background music generation module and melody generation module, all linked together by the main program, which provides a user-friendly graphical user interface.

The system implements seven musical expressions, mapping each one to a different kind of motion of gesture with one or two hands. The musical expressions are rhythm, pitch, pitch-shifting, dynamics, volume, dual instrument mode, and cadence. The gestures used to control the generated melody include, among others, flexion of the right wrist, height of the right hand relative to the ground and lifting of the same hand, position of the left hand relative to the right hand, and bending the fingers of the left hand.

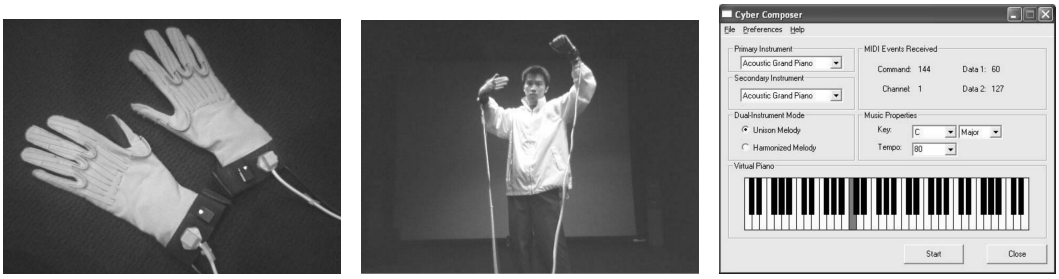


Figure 2.6. CyberGloves used in Cyber Composer [32] and its graphical user interface

In [72] two semi-virtual instruments based on physical interfaces are implemented: a flute and a drum.

A semi-virtual flute is created using a real plastic tube with three buttons and a small fan inside, which is activated by the user's breath. The rotation of the fan is transmitted to the computer, which processes it together with the data on the buttons pressed, producing the corresponding sound. A drum, on the other hand, is created using only a stick with a red object attached to the tip. The red tip is tracked by the webcam, which calculates its movement by keeping track of its position and speed.

Both the flute and the drum can be visualized live on the computer screen, accompanied by visual effects which inform the user of the type of sound produced.

Figure 2.7. Semi-virtual flute, semi-virtual drum and their visualization in [72]

In [74], a system is developed to play a virtual instrument using hand gestures to modify its parameters such as pitch name, octave, accidentals, instrument pitch bend, note value, and frequency equalizer.

Pitch, octave and accidentals are controlled with the right hand. The number and position of the fingers bent towards the palm are used to distinguish pitch, while the position of the palm in relation to the shoulder is used to control octave

and accidentals. The left hand is used to control the type of instrument (trumpet or piano) and the pitch bend using three movement actions. Based on the angle of both hands, the note value and frequency can be controlled.

This system works in both real-time and delayed mode, achieving an accuracy of 90.8% and 98.85% respectively.

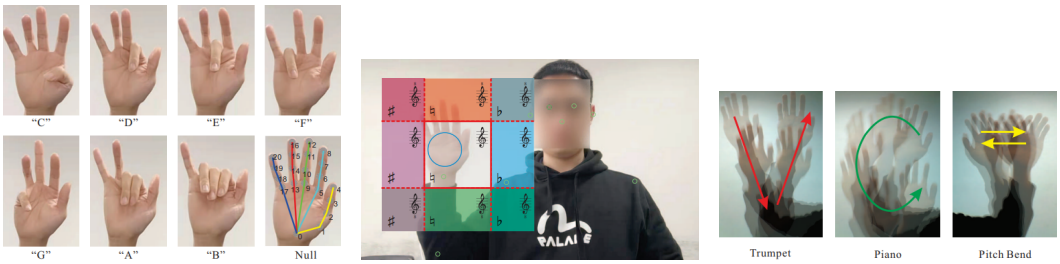


Figure 2.8. Instrument controls from [74]

In [62], an EMG based MIDI controller is implemented and tested. The controller needs both a complex physical architecture and an intricate software to run. The purpose of the controller is to identify four hand gestures, each associated with a different note.

The hardware consists of four components: three EMG (Olimex Shield-EKG-EMG) sensors to be connected to the arm that is to act as the controller, an Arduino UNO board to read the data from the three EMG sensors, a Raspberry PI 4B board to receive the data from the Arduino and interpret it with the gesture recognition software, and a computer that receives the final signal containing the MIDI note to be played.

The software communication between these devices is architected in the following way, and is schematised in Figure 2.9. First, the Arduino UNO board reads data from the three EMG sensors. Then the Raspberry PI board requests this data from the Arduino, and when it receives it, it passes it to the classifier that produces the hand gesture. The result of the classifier is sento to the Arduino, which converts it into a MIDI note and sends it to the computer so that the note can be played.

Three types of classifiers are used to test the system: Random Forest, Support Vector Machines and a Neural Network with three hidden layers. All three classifiers are trained on a dataset consisting of 60 measurements per each of the four gestures.

The results are unfortunately lower than hoped for, with a maximum of 78% obtained by SVM and a minimum of 71% obtained by the Neural Network. According to the authors, the cause for such a low result could be the Arduino, which is deemed compromising for applications that require high sampling rates.

2.5.2 Virtual trainers

When it comes to learning a musical instrument using virtual reality tools, it is not only important to have a tool that allows one to play but also a tutoring system that gives feedback on the quality of the novice musician's performance.

The work done in [56] is not exactly a virtual reality application, but still represents an example of how depth sensors can be used for teaching and learning how to play a musical instrument. This paper proposes a method for recognising piano

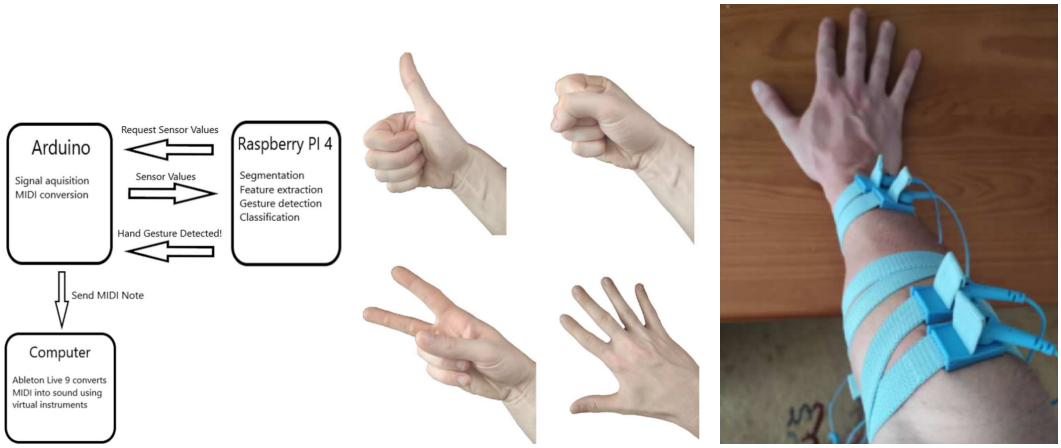


Figure 2.9. Architecture, gestures and sensor position from [62]

fingering by analysing motion of multiple fingers through the use of depth images acquired with a depth sensor. The depth sensor used in the paper is a Microsoft Kinect, but in theory any other depth sensor may be used.

The proposed method consists of two modules: a learning module and a recognition module. The learning module is used to generate a dataset of depth images for various key pressing patterns. The recognition modules takes a depth image as input and tries the best matching key pressing pattern in the dataset, with the goal to identify which one is being performed in the input depth image. The recognition process uses the Nearest-Neighbor search technique. Both the learning module and the recognition module make use of a background subtraction technique, to separate the depth information about the keyboard from those about the hand.

The method was tested on three songs, with a dataset of 780 data elements in 130 classes acquired by the learning module, obtaining a maximum recognition rate of 100%, and a minimum of 91.6%. The recognition time was also measured and found to be 120 milliseconds per image, making this method applicable to slow-tempo songs for piano beginners.

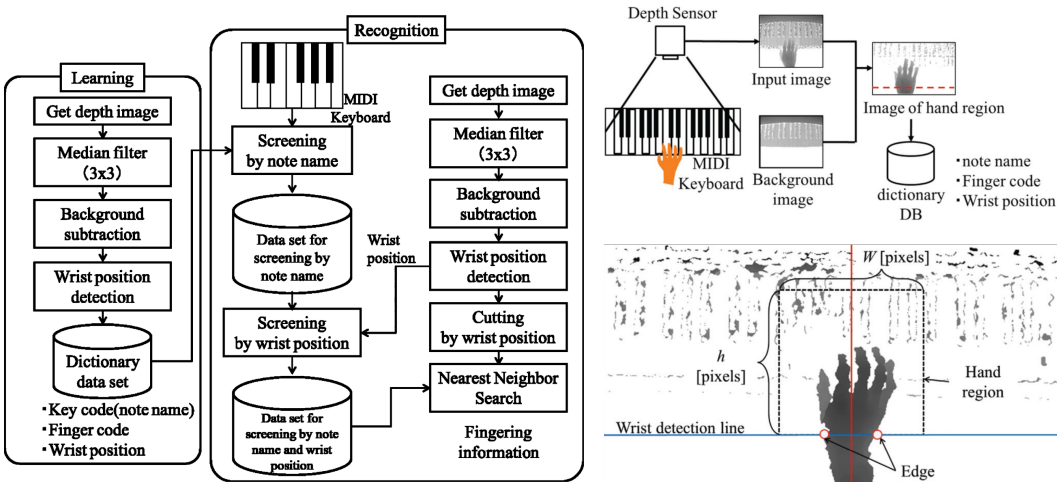


Figure 2.10. Architecture scheme, learning flow and recognition phase from [56]

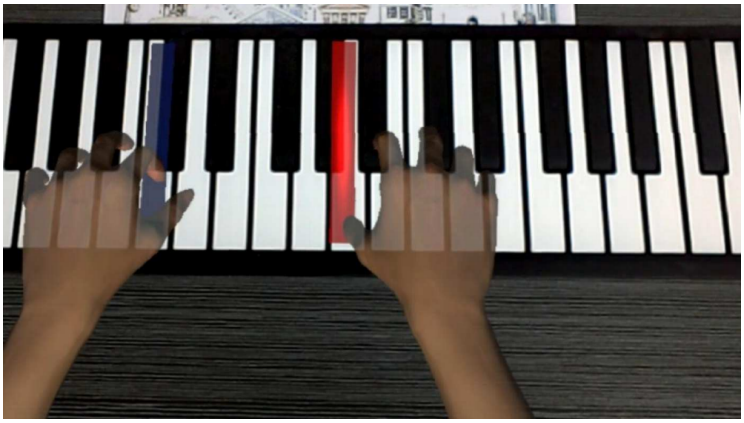


Figure 2.11. Fingering generation in an augmented reality environment from [29]

In [29], an AR-based individual tutorial system is proposed. Its goal is to automatically generate hand action animations and displaying animations with real pianos using head-mounted displays. Starting from a piece of music given as input, the system generates a coherent and natural-looking animation of hands playing the given sequence of notes on a virtual piano.

To go from a MIDI file to an animation of hands playing a piano, two steps are taken. The first step is to convert the raw MIDI file o a fingering-tagged MIDI file, in which every note is tagged with the finger that plays it. This is done with a Hidden Markov Model (HMM) and the Viterbi algorithm. The HMM can be applied not only to single notes but also to chords that require a multi-finger movement. The second step consists in creating the actual hand animation from the finger-tagged MIDI file. The three patterns of fingering that can be recognised are Basic Finger Motion, Alternative Finger Motion and Multi-finger Hand Motion. To produce human-like animations for all these patterns, the authors have defined step-by-step animations for each and every case above.

Piano Beginner [45] is a finger training system with auditory and visual feedback (AVFB), which supports both single-finger and multi-finger movements of two hands. By wearing a pair of 5DT Data Glove Ultra and an optional VR headset, the user can experience an immersive simulation of piano training.

The peculiarity of this project is in the setup phase: rather than using a generic algorithm to detect finger pressure, the authors develop a calibration process in which the user performs some actions like closing the hand and pressing each finger singularly, to let the program learn what is the correct threshold to detect pressure for each finger.

The results of the experimentation show that users' accuracy increases by more than 20% after 20 training sessions.

GuitarGuru [51] is a multi-featured application that caters for beginner guitarists with four methods to learn and improve their performance on the guitar, combining computer vision and deep learning techniques.

The first feature allows a user to upload a video, or a link to a video on YouTube, in which a musician plays the guitar. GuitarGuru analyses the video and extracts all the chords from it in the form of a chord sheet. With this function the user can

Figure 2.12. Glove-based finger training system from [45] and its Unity3D environment

also play the guitar in front of the webcam and get the chord sheets in real-time.

The second feature consists of a training session: the user selects a chord sheet from those previously generated with the first feature, then starts the webcam and the application analyses their performance in real time, evaluating its correctness and calculating the accuracy rate.

The third feature is similar to the previous but is a more specific and focused test on individual chords. The user stands in front of the webcam and plays one chord at a time, while the application measures the accuracy of each individual chord that is played.

The last feature is about music theory, offering a section where the beginner can learn notions about all the music-related theories that will help the user in learning the basics of music, including brief information about chords, scales, pitch, etc.

The dataset to train the deep learning model was built using python with OpenCV for preprocessing and MediaPipe for extracting hand gestures, resulting in 25000 images: 5000 per each chord A, C, F, G, and 5000 representing a no-chord, in which none of the previous chords was detected. The deep learning model was built with Keras, with one input neuron and one output neuron, 32 filters in each layer and the size of a layer being (5, 5) with Relu as the activation function.

The model gives an accuracy of 97.09% for the 4 chords A, C, F, G and a loss of 2.91%. In the future, more chords can be added for identification, which would help expand the range of music genres to which GuitarGuru can be applied.

Another trainer for pianists can be found in VRmonic [48], an immersive VR-based piano trainer that uses the Unity Game Engine and a VR headset with a hand detector to compare the user's hand movements to the ones of a virtual expert oracle, on a set of 48 scales for each of the 12 notes. The goal of this project is not only to teach how to play the piano, but also how to play it with the correct hand form, preventing long-term injuries in musicians of all levels.

The oracle's hand poses are recorded using a capture environment in which some expert pianists perform the scales on an 88-key piano. The pianist's hands are captured using a Microsoft Azure RGB-D camera mounted 88.25 cm above the piano. Five trials are performed for each scale and the resulting joint positions are merged together into the final oracle positions by averaging the positions in each trial.

When running a training session, the user needs any VR headset that supports

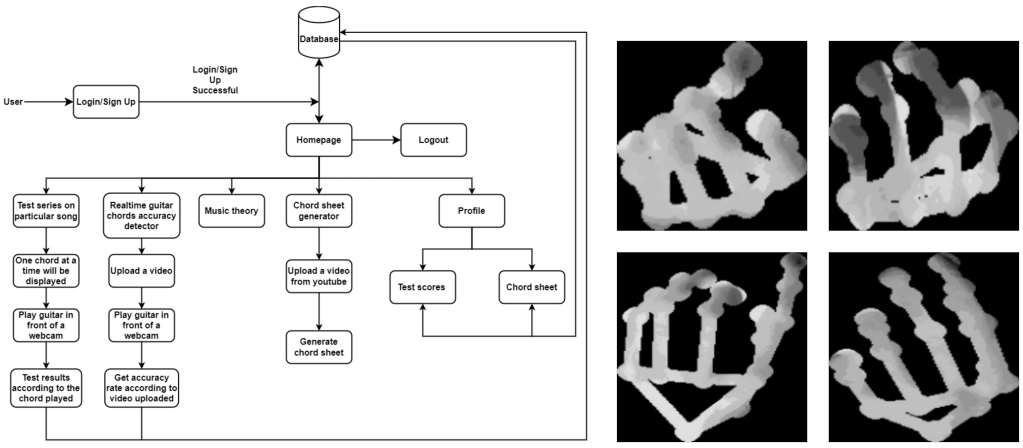


Figure 2.13. Architecture scheme and 4 chords (A, C, F, G) recognised from [51]

Unity and is capable of doing hand detection. After having selected the note, scale, and beats per minute, the user can start to record a playback or run a playback by the oracle. Pieces recorded by the user can be later played back together with the ones played by the expert oracle, creating an overlay of two pairs of hands, one belonging to the user and one to the oracle. The user can set the tolerance level, which represents the number of collisions between corresponding joints on the two pairs of hands, to control the level of feedback on alignment with the expert. VRmonic currently does not support real-time feedback.

VRmonic is a promising project with great potential. Future development possibilities include adding support for real-time feedback mode, the incorporation of qualitative feedback via large language models (LLM) to provide personalised voice-based feedback based on the magnitude of the error, and the addition of further control over the pianist's posture as well as the position of his hands.

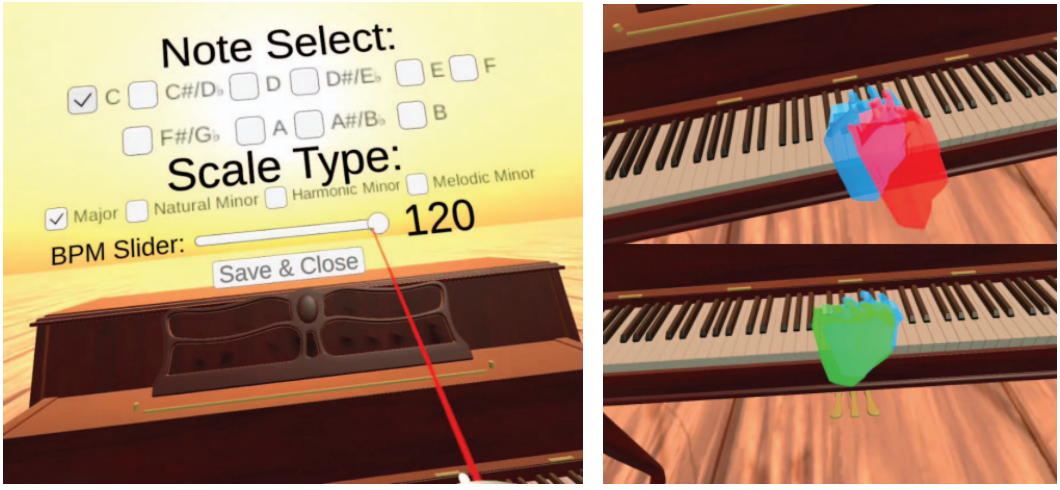


Figure 2.14. VRmonic interface and overlapping hands example: inaccurate vs accurate

Chapter 3

System architecture

In this chapter, the project is presented in its entirety, starting from the initial idea with the first prototype, going through the various iterations, the choices made and the hypotheses discarded, up to the current, but not final, version of the Keyrtual application.

3.1 History of the project

This project originally began in January 2023 as work for a paper [10] presented at the 20th International Conference on Computer Analysis of Images and Patterns (CAIP 2023). While the paper's version and the current version of the project are similar in purpose, they are completely different in implementation. The main difference between the two is that the first one was not intended for mobile devices, but was essentially a prototype designed to assess the general feasibility of the project, although it was still meant to be suitable for low-end (non-mobile) devices.

In this first prototype, the procedure to use the program is longer and more cumbersome, and is summarized in Figure 3.1.

3.1.1 Constraints

General constraints The paper sheet where the keyboard is drawn should have a sufficiently strong contrast to the table, and the same applies to the keys drawn to the paper.

The camera and Leap Motion should also be placed at an adeguate distance to be able to frame the paper and hands in their entirety. They should be placed perpendicular to the table, so that they have as good a view as possible from above, and oriented so that the user appears at the top.

Hardware prerequisites First, the use of an external webcam and a Leap Motion Controller is required. These two devices should be placed next to each other perpendicular to the table on which the user intends to use the drawn keyboard, so as to have an elevated view of the keyboard and the user's hands.

As for the computer, there is no special prerequisite since the program is designed to be used even on low-end devices.

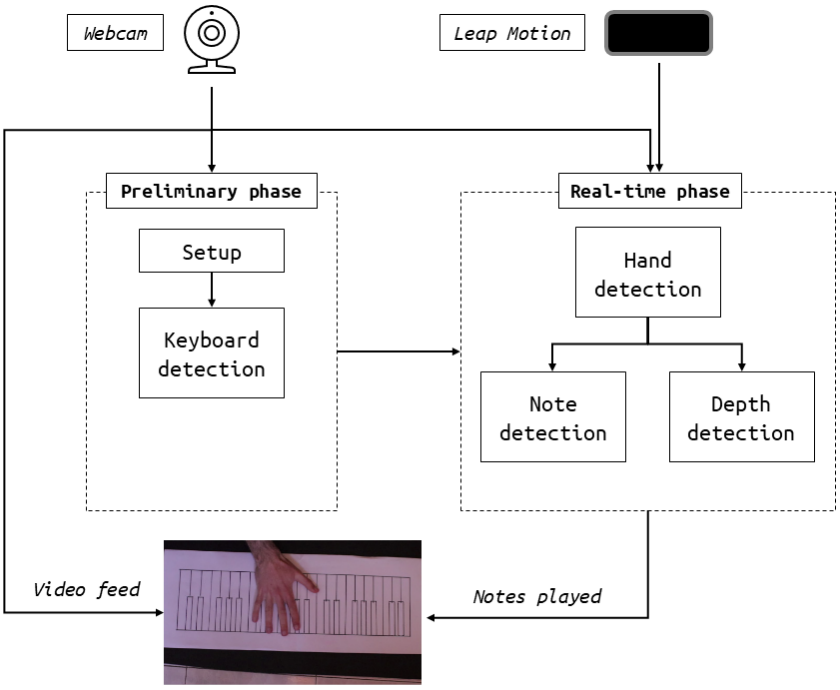


Figure 3.1. Architecture of the first prototype

3.1.2 Preliminary phase

The preliminary phase is divided into two parts: the background subtraction part and the keyboard detection part. The whole process is illustrated in Figure 3.2

Background subtraction Once the components are assembled, we move on to the preliminary stage of keyboard detection. For this purpose, a background subtraction operation is first performed, which consists of taking two pictures: the first will be a picture of the table without the keyboard, while the second will include the keyboard. By calculating the differences between the two images, the background is removed, leaving only the keyboard.

Keyboard detection Once the area of the image on which the keyboard is located has been identified, we proceed with key detection. This is done by applying different filters to the image to increase the contrast between white sheet and black keys, and using classic algorithms such as Canny edge detector, probabilistic Hough Transform procedure and K-means clustering to extrapolate the rows that make up the keyboard keys. These rows are separated according to their length and position to separately identify white keys and black keys.

3.1.3 Real-time phase

After the setup is finished, we move on to the real-time phase: the user can now play on the keyboard, and the program will play back the notes played. In this phase,

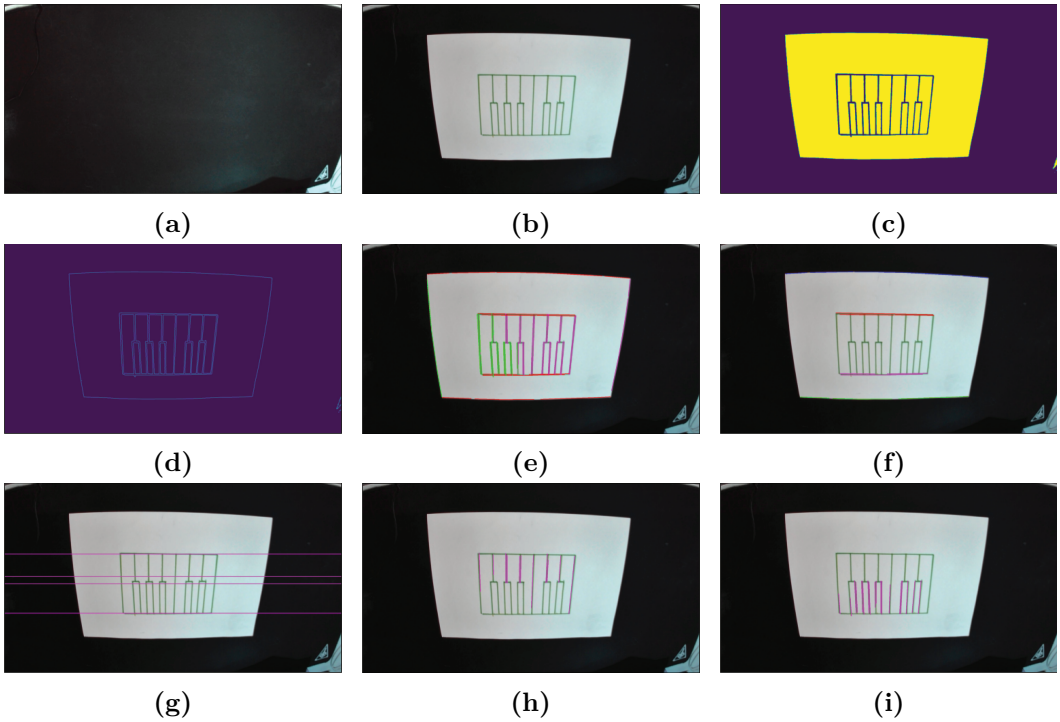


Figure 3.2. Preliminary phase pipeline. (a) First frame; (b) Second frame; (c) Background highlight; (d) Canny edge detection; (e) Hough transformation; (f) Horizontal lines detection; (g) Borders highlight; (h) White tiles highlight; (i) Black tiles highlight.

MediaPipe and the Leap Motion Controller are used in conjunction to detect which key is being pressed and if it is actually pressed, respectively.

This is necessary because MediaPipe, despite its accuracy in detecting the 2D position of the hands on the image, is not at all accurate in detecting the third coordinate, that is, the distance of the fingers from the camera.

Sensor fusion For this, a sensor fusion technique is used between MediaPipe and the Leap Motion Controller: MediaPipe is constantly used to detect the presence and position of hands on the screen, but it is the Leap Motion Controller that triggers the process of playing a sound: when the controller detects that a finger has fallen below a certain threshold, it uses the coordinates provided by MediaPipe to determine the button on which the lowered finger is located, and the corresponding note is played using the MIDI protocol.

3.2 Evaluation of Porting Strategies

Following the prototype publication, various intermediate solutions were hypothesised, tested and discarded before arriving at the current version of the Keyrtual mobile application. This section will explore the various steps that led us to the final version, the unsuccessful attempts and the reasons why certain choices were made.

3.2.1 Depth vision

Nowadays, the presence of two or more photo sensors has become the standard for all smartphones, even low-end ones. Among the sensors that can be found on smartphones there are for example depth sensors, macro sensors, wide-angle sensors and tele sensors.

One of the first ideas, when we started thinking about porting the project to smartphones, was to use these different sensors to derive information about the position of the hands in the three-dimensional space of the scene, in two possible ways:

- *Depth sensor:* directly use the smartphone's depth sensor to obtain depth information on the playing fingers, to be combined with the information obtained from the RGB sensor and MediaPipe, just like in the first prototype.
- *Stereo vision:* use two different camera sensors at the same time to capture two images of the same scene, from two slightly different points of view, and then use classic stereo vision algorithms to derive the depth of the hands in the scene.

At first glance, both of these methods might seem excellent for solving the problem of depth ambiguity in single-perspective images, but unfortunately both have technical difficulties that make them practically unusable in our use case.

Depth sensor

The depth sensor in smartphones is used, among other things, to improve focus, sharpness and to apply visual effects on the final image, such as background blur and focus on an object in the foreground. The idea of using this sensor to derive depth information on the hand is unfortunately unrealistic given the low resolution and sensitivity of this type of sensor.

In our case, we need an accuracy in detecting depth in the order of a centimetre. In fact, to detect the touch of a finger, we need to detect a small change in depth, of about 1 cm, in the area of a single finger, about 1 cm^2 in size, at a distance from the sensor of about 40 cm.

This accuracy could perhaps be achieved by the sensors built in a handful of high-end smartphones, but since one of the goals of this project is to reach as many devices as possible, this method was abandoned very early on.

Stereo vision

This method presents even greater technical difficulties than the previous one. Firstly, as mentioned above, although smartphones are equipped with multiple camera sensors, they all perform a different function and differ deeply in terms of size, accuracy and the type of output produced. To use the stereo vision technique, one would need a smartphone with two identical cameras.

Moreover, the presence and positioning of these sensors on the back of the smartphone is by no means regular, and can vary greatly from model to model. This would be a major obstacle in using stereo vision algorithms, which depend on the position and proximity of the sensors used.

Iphone 14 Pro Samsung Galaxy S24 Xiaomi 13T

Figure 3.3. Different models of smartphone with different camera sensor number and position.

Dedicated software

We also considered the possibility of using dedicated software to calculate a depth map from an RGB image, such as MiDaS [66]. Unfortunately, these models are not designed for use on mobile devices, and require computing power that is not available on smartphones.

Although there are versions designed specifically for mobile devices, the resulting depth maps are not precise enough for our purpose.

3.2.2 MediaPipe for 3D coordinates

MediaPipe's hand detection feature not only detects hand landmarks as 2D coordinates in the image, but also produces extra coordinates representing the position of the landmarks in 2.5D space. In the 2.5D space, in addition to the existing x and y axes, a new z axis is added to represent the landmark depth with the depth at the wrist being the origin, and the smaller the value the closer the landmark is to the camera.

Among the methods hypothesised for porting the project to smartphones was to use this new coordinate to assess whether a finger was pressing on a button, using the z axis value. This method immediately revealed a number of problems, including the fact that the z coordinate, being derived from a single RGB image, is not precise and is also very flickering.

In addition, the z coordinate is not expressed in absolute reference but in reference to the landmark located in the wrist, which makes its value variable not only according to the position of the individual landmark but according to the position of the entire hand in relation to the wrist.

3.2.3 Point of view

A very important difference between the new and the old prototype is the change in the positioning of the camera. In the first prototype we chose to use a 90° angle, which gave us a very good view of the keyboard, precise information on the position of the hands on the plane in relation to the keys and, thanks to the Leap

Motion sensor, precise information on the distance and position of the hands in three-dimensional space. This results in complete information on the x_w and y_w axes (x and y world axes) given by the RGB sensor, and excellent information on the z_w axis given by the Leap Motion sensor.

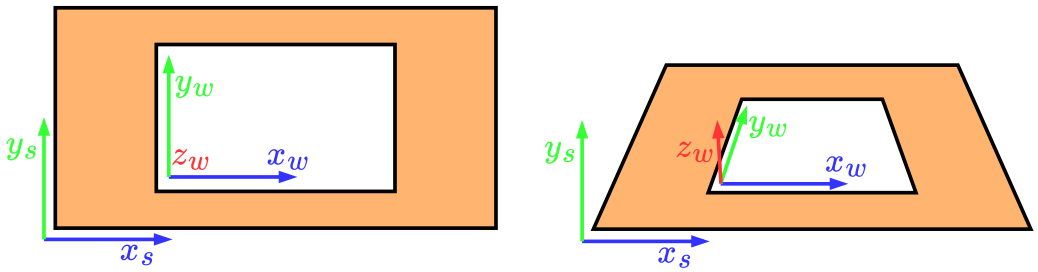
In the new version, in which we only have the RGB sensor, using the same positioning of the camera, the z_w axis information is totally lost. This can be partially recovered thanks to MediaPipe, which however does not provide sufficiently precise depth coordinates but rather inconsistent and flickering, so we could not rely solely on them.

Instead, we preferred to exploit a 45° angle of the camera in order to make the best use of the partial information this viewpoint provides us with regarding the z_w axis. Changing the angle from 90° to 45°, in fact, provides us with partial information on the z_w axis that is completely absent in the top view. The two methods are illustrated in Figure 3.4.

The first method, that of the old prototype explained in subsection 3.1.3 Real-time phase, gave complete information on the x_w and y_w axes, but no information on the z_w axis. This shortcoming, shown in Figure 3.4a, was compensated for by the use of the Leap Motion sensor, which provided accurate depth data and allowed the system to function correctly.

However, this method is no longer usable now that we only have an RGB camera at our disposal, and we were therefore forced to change to a different angle. This new viewpoint has the advantage of gaining partial information about the z_w axis, at the cost of a partial loss of information about the y_w axis, both of which are visible in Figure 3.4b.

The result is that the y_w and z_w axes are partially merged and "compressed", becoming part of the same y_s axis (y screen axis, given by MediaPipe), while the x_w axis maintains a virtually exact correspondence with the x_s axis.



(a) Camera angle at 90° **(b)** Camera angle at 45°

Figure 3.4. Different points of view of the camera on the paper sheet. The lower the angle, the more precision about axis z_w but less precision about y_w.

The partial loss of information on the y_w axis, however, is not a problem at all: it is in fact only used for note recognition, and this only occurs when a touch is recognised, i.e. when the finger presses directly on the paper. The loss of information on the y_w axis, therefore, does not affect note recognition at all, nor the functioning of the application in general.

Thanks to the presence of the frame area, during keyboard detection, we can

force the correct positioning of the smartphone, guaranteeing the success of real-time phase.

3.3 Unity

Unity is a professional, robust and versatile game development platform that has gained widespread popularity [31] for its ability to create both 2D and 3D applications. The Unity editor is cross-platform, being supported on Windows, macOS and Linux, and its engine supports builds for more than 19 different platforms, including mobile and desktop devices, consoles and VR devices.

Developed by Unity Technologies, it has evolved into a powerful tool for implementing high performance applications [17], used by developers worldwide for a variety of projects, ranging from games to simulations, augmented reality [8, 7], and virtual reality [9] applications.

Augmented reality applications made with Unity can easily be used in a classroom [36, 40, 54, 79] as an innovative and interactive teaching method for children, with examples of its usage in the teaching of subjects such as Astronomy [75], Volcanology [1], Engineering [83], Medicine [70] and more [2, 52]. This makes Unity the perfect tool for our project.

Our application requires the use of the OpenCV and MediaPipe libraries, and is written entirely in `C#` within the Unity Engine. This led us to the adoption of the corresponding plugins within the game engine. The exact versions of Unity and plugins used within the project are:

- Unity Editor: 2022.3.18f1
- OpenCV for Unity: 2.5.8
- MediaPipe for Unity: 0.14.1

3.3.1 OpenCV

OpenCV for Unity is a very popular plugin for using OpenCV within the Unity development environment. It can be purchased from the Unity Asset Store and easily imported into a project. From there, its use differs little from what it would be in `C++` or `Python`: it supports practically all the functions and constants present in the original module, while maintaining an API identical to its `C++` counterpart.

The support for WebCamTexture makes OpenCV for Unity easy to use in applications that need to perform real-time image processing, which makes it an excellent choice for our project.

3.3.2 MediaPipe

MediaPipe for Unity is a native plugin which ports the `C++` MediaPipe API to `C#`, one by one, so that it can easily be used within the Unity development environment. It can be installed from the documentation page and be directly imported into the project. Thanks to the precise documentation, the various example scenes and the clear and comprehensive tutorial, using MediaPipe in Unity is quick and easy.

This library supports all major desktop operating systems (Linux, macOS and Windows) as well as the mobile systems Android and iOS, for which GPU mode can also be enabled, which is why we chose to use this library in our project.

3.4 Method

As of now, the project has made progress towards user-friendliness and accuracy, and all that is needed is a smartphone with the application installed and a paper sheet. The pipeline is still divided into a preliminary phase and a real-time phase, but now all the work is done by a single device instead of three different devices.

Launching the application immediately gives visual feedback from the camera on the smartphone screen: the user directly sees what they are framing with the camera, making the preliminary phase much faster and easier, also thanks to a visual guide in transparency on the image, which is used to aim at the keyboard.

Once the keyboard is centered in the visual guide, the user can press a button on the application interface to start the process of detecting the keyboard. The detection process takes less than a second, and once it is finished, the screen will display in transparency the keys detected by the application, so that the user can assess whether the process was successful and, if they are not satisfied, repeat it as many times as necessary.

Both the visual guide and the keys in transparency can be hidden either before or after the preliminary step. When the user is satisfied with the result of the preliminary stage, they can immediately start playing on the semi-virtual keyboard, without any other interaction with the application.

The entire process is shown in Figure 3.5 and the implementation details are explained in the following paragraphs.

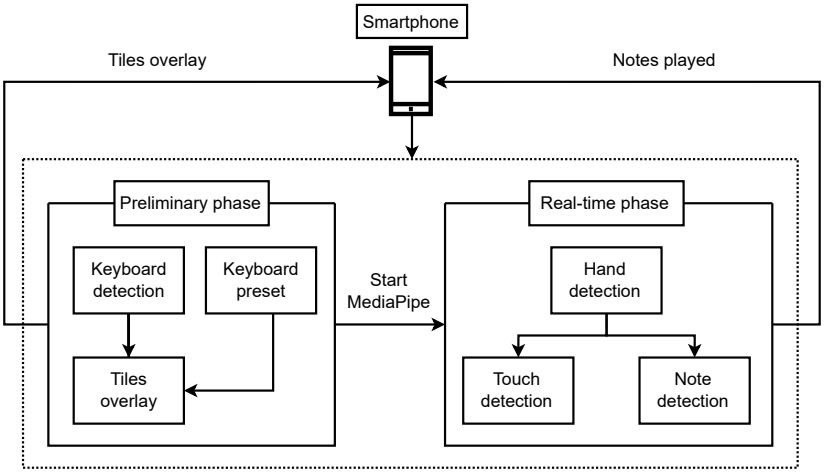


Figure 3.5. Architecture of the current version

3.4.1 Constraints

General constraints The sheet should be a common white A4 sheet and the keyboard should be drawn in black or blue for better contrast. There are no constraints

on the contrast between the paper sheet and table due to the new background subtraction method.

The phone should be placed at a sufficient distance to frame the paper and hands in their entirety (experimentally we have found that about 40 cm is fine) in a way such that the user, relative to the camera, appears to be on the top side of the image and the paper on the bottom side of the image.

Hardware prerequisites No special prerequisites are required to run the application. Thanks to the possibility of increasing and decreasing the FPS with which to run MediaPipe, which is the most computationally demanding component, it is possible to sacrifice some response speed to ensure that the application works on low-end phones.

3.4.2 Preliminary phase

The preliminary phase still consists of background subtraction and keyboard detection, but the process to follow is much faster, intuitive and automated.

All the single steps of this phase are shown in Figure 3.6.

Image acquisition As soon as the application is opened, the user can immediately go to the preliminary step of keyboard detection. Due to the high mobility of the smartphone and the immediate visual feedback from the screen, no real background subtraction process is required during this phase: a perimeter is drawn on the smartphone screen around the area within which the application expects the user to frame the drawn keyboard.

When the user frames the keyboard in the perimeter and presses the button to start the detection process, the application takes an image with a resolution of 1280×720, a reasonable resolution for all modern smartphones, and cuts off from the image all the area outside the perimeter, keeping only the area where the keyboard is located. This makes the background subtraction process much more effective and accurate, and also much faster since no particular algorithm is used.

Since smartphone webcams, used via Unity, have an image correction and automatic focus function, we had to take countermeasures to prevent the webcam from automatically moving the focus outside our perimeter, making the detection phase inaccurate. To do this, we forced the webcam's focus on the centre of the perimeter via Unity's API.

The perimeter and the result for this phase are shown in Figure 3.6a and Figure 3.6b.

Preprocessing Before performing the actual keyboard detection, we go through a second but very important preprocessing step: filters are applied to the cropped keyboard image to highlight the keys drawn in black against the white background of the paper sheet.

The image is converted to grayscale and Canny edge detector algorithm is applied to highlight the black lines. Two iterations of closing morphological operator are applied to the image, to enhance the visibility of the edges found by Canny.

We are not quite interested in cleaning those small white pixel spots that appear scattered over the image because they will be removed during the next step.

The single steps of this phase can be seen in Figure 3.6c and Figure 3.6d.

Tiles detection At this point, we proceed to the actual keyboard detection. After extensive experimentation, we concluded that the old method using probabilistic Hough Transform to detect lines was too inaccurate and inconsistent, and moreover wouldn't allow us to detect non-straight lines, which is a fundamental prerequisite in this case since we would like to detect hand-drawn keyboards. Therefore, we opted for a much simpler and more robust method: contour detection [80].

The contour detection procedure is applied to edges obtained from the previous step. The contours thus obtained, however, are filtered in several ways, eliminating:

- contours with width greater than height (non-vertical)
- contours with area smaller than 300 pixels
- contours with area greater than 5% of the image

The contours left represent the keys drawn on the paper and are shown in Figure 3.6e.

Tiles overlay With the previously obtained contours, the overlay with the keys drawn in transparency is created and shown to the user. To make the overlay look like a real keyboard, the keys are colored white or black according to their length: longer keys are white and shorter keys are black.

The result can be seen in Figure 3.6f

Notes detection The contours are sorted using the horizontal center as the sorting criteria. An image is then created with the same size as the original, initially all black, which is colored only inside the areas of the contours found. The areas are colored with a gray scale starting at 1 and going up by 1 for each key.

The color for each key represents an index, starting from 1 up to the number of keys found, which is used to determine the note number to be played according to the MIDI protocol. The details of this method are explained in section 3.4.3 Note detection.

The final result is shown in Figure 3.6g.

Keyboard presets

In this mode, instead of using a photo taken by the user to search for the keyboard and show the keys thus detected, the shape and position of the keys that the application expects to use is shown directly.

The user can download and print out on an A4 sheet the model of the keyboard that the application shows on the screen and must position it at the guides shown on the interface. In this way, the user's physical keyboard will perfectly match the one presented by the application, ensuring optimal performance and realism.

The PDF files for printing the keyboard presets will be available for download on the project's website. The different keyboard models made available by default by the application are shown in Figure 3.7.

3.4.3 Real-time phase

When the user is satisfied with the result of the detection phase, they can press a button to start the real-time phase. First, the graph is launched: the GPU manager is started, models for palm detection, hand landmarks and handedness are loaded, and side packets are created. Three data streams are started from the graph: *hand landmarks*, *world hand landmarks* and *handedness*, and an event listener is attached to each to receive the results of the processing of each frame. The graph is schematised in Figure 3.8.

The three data streams on which the outputs of the graph are passed work asynchronously with respect to the main loop and also with respect to each other. To ensure that the next step of the algorithm only starts when all three streams have produced a result for the same frame, it was sufficient to implement a simple synchronisation system for the three outputs to ensure that each frame is only passed to the next step of the algorithm when all three components are available.

Once all three components are available, our algorithm proceeds in its two phases:

- *Touch detection:* check if a finger touched the keyboard
- *Note detection:* if a finger touched the keyboard, find which note was pressed and play the corresponding sound

It is important to note that, in its current state, not all three streams are used by our algorithm. In particular, only the ones for *hand landmarks* and *handedness* are useful for our purposes. The other, *world hand landmarks*, is present mainly for reasons of compatibility with older versions of the algorithm and the possibility that it will be useful in the future.

MediaPipe on low-end devices While the webcam and the video feed on the smartphone interface work at 30 FPS, the frequency at which frames are sent to MediaPipe for hand detection is not fixed. In order to meet all requirements and to be able to support as many smartphones as possible, even low-end ones, we have implemented a way for the user to set the frequency at which frames are sent to MediaPipe at will, i.e. the FPS at which the hand detection process runs.

This has been done by including buttons in the interface to directly increase and decrease the FPS with which to run MediaPipe. Changing the value from the interface results in an immediate change in the operation of MediaPipe.

Through experimentation, we have found that a good value for MediaPipe's FPS is 15, which is high enough to ensure that the app runs smoothly with low response times, but is not too high to be unusable on mid-range smartphones.

Going below 15 FPS is possible, but results in an increase in the app's response time and a possible loss of some notes. Going above 15 FPS is also possible, and it

is recommended in the case of high-end smartphones, but being aware that increasing the frequency with which frames are sent to MediaPipe also increases battery consumption and could lead to the smartphone overheating.

Touch detection

At this early stage of the project, the touch detection procedure is not yet very refined. Since MediaPipe alone does not provide sufficiently precise depth coordinates (or camera distance) but rather inconsistent and flickering ones, and for the reasons explained in subsection 3.2.2 MediaPipe for 3D coordinates, we could not rely solely on them for the touch detection phase. We therefore exploit the advantages explained in subsection 3.2.3 Point of view.

To detect a touch, we rely on the position of the finger on the y_s axis: when a finger reaches down towards the paper to play a note, this movement is detected by MediaPipe thanks to the 45° view. The y_s position of the finger is given as input to an automaton that takes care of maintaining and updating the state of the finger at each frame. The automaton is illustrated in Figure 3.9.

When a new frame is ready, the x_s and y_s positions of the finger are extracted from MediaPipe and converted into pixels and the y_s position is given to the automaton. The automaton calculates the speed of the finger by comparing the new y_s with the last detected y_s, after which it advances the finger state accordingly. When the finger enters the *Touching* state, the x_s and y_s coordinates are passed to the note detector which is responsible for verifying that the finger has touched a key and, if so, playing the corresponding note. The automaton is made so that the *Touching* state remains active even if the finger moves horizontally.

In Figure 3.9, the `t` value indicates a threshold which serves to mitigate the instability of MediaPipe: the hand landmarks detected by MediaPipe are subject to minute, continuous variations which, if not taken into account, would cause a series of false positives and would repeatedly take the automaton in and out of the *Touching* state. Experimentally, we came to the conclusion that 8 is a suitable value for the automaton's activation threshold.

The pseudocode for this phase is shown in Algorithm 1.

Note detection

Once we have detected the fingers touching the keyboard with MediaPipe, we need to work out which notes have been touched. To do this, we use the image generated in the keyboard detection phase in section 3.4.2 Notes detection.

This image is the same size as the video feed and is totally black except where the keys are located. The keys are coloured in a grey scale where the first key is coloured with the value 1, the second with the value 2 and so on. This allows us to associate each key with a unique value that we can use to determine the note to be played.

Using the x_s and y_s coordinates provided by MediaPipe, we can superimpose them on the image generated in the keyboard detection phase to determine whether a finger has touched a key. If the pixel at position (x, y) is black, the finger is outside the keyboard and nothing happens. If, on the other hand, the pixel is not black, the

Algorithm 1 Touch detection

```
1: when new frame arrives
2: if no hand has been detected then
3:     return
4: end if
5: toPlay ← {}
6: for each hand h detected do
7:     for each finger f of h do
8:         x ← horizontal position of f
9:         y ← vertical position of f
10:        isPlaying ← automata(f).update(y)
11:        if isPlaying then
12:            toPlay ← toPlay ⋃ {(x, y)}
13:        end if
14:    end for
15: end for
16: do note detection with toPlay
```

finger is on one of the detected keys and therefore the MIDI signal corresponding to the detected note at that position must be played.

Playing sounds is not as straightforward a process as it might seem, and it does require a little forethought. This is because when passing from one frame to another, some fingers that were playing in the previous frame may still be playing the same key. In this case the note does not have to be played again, but only continued.

To do this, we keep a list of the notes that are being played and, at each frame, create two different sets of notes: the set of notes that were being played in the previous frame but are no longer being played in the current frame, and the set of notes that were not being played in the previous frame but are being played in the current frame.

This allows to detect notes that were being played in the previous frame and are still being played in the current frame, so that they are not played again but remain playing.

The pseudocode for this phase is shown in Algorithm 2.

Algorithm 2 Note detection

1: **Input**: the set of coordinates *toPlay* of the fingers touching the keyboard
2: $playing \leftarrow \{n \mid n$ is already being played from the previous frame$\}$
3: $detected \leftarrow \{n \mid (x, y) \in toPlay$ and
n is a non-black pixel at (x, y) in the note image$\}$
4: $stop \leftarrow playing \setminus detected$
5: $start \leftarrow detected \setminus playing$
6: **for** each n in *stop* **do**
7: stop playing note n
8: **end for**
9: **for** each n in *start* **do**
10: start playing note n
11: **end for**

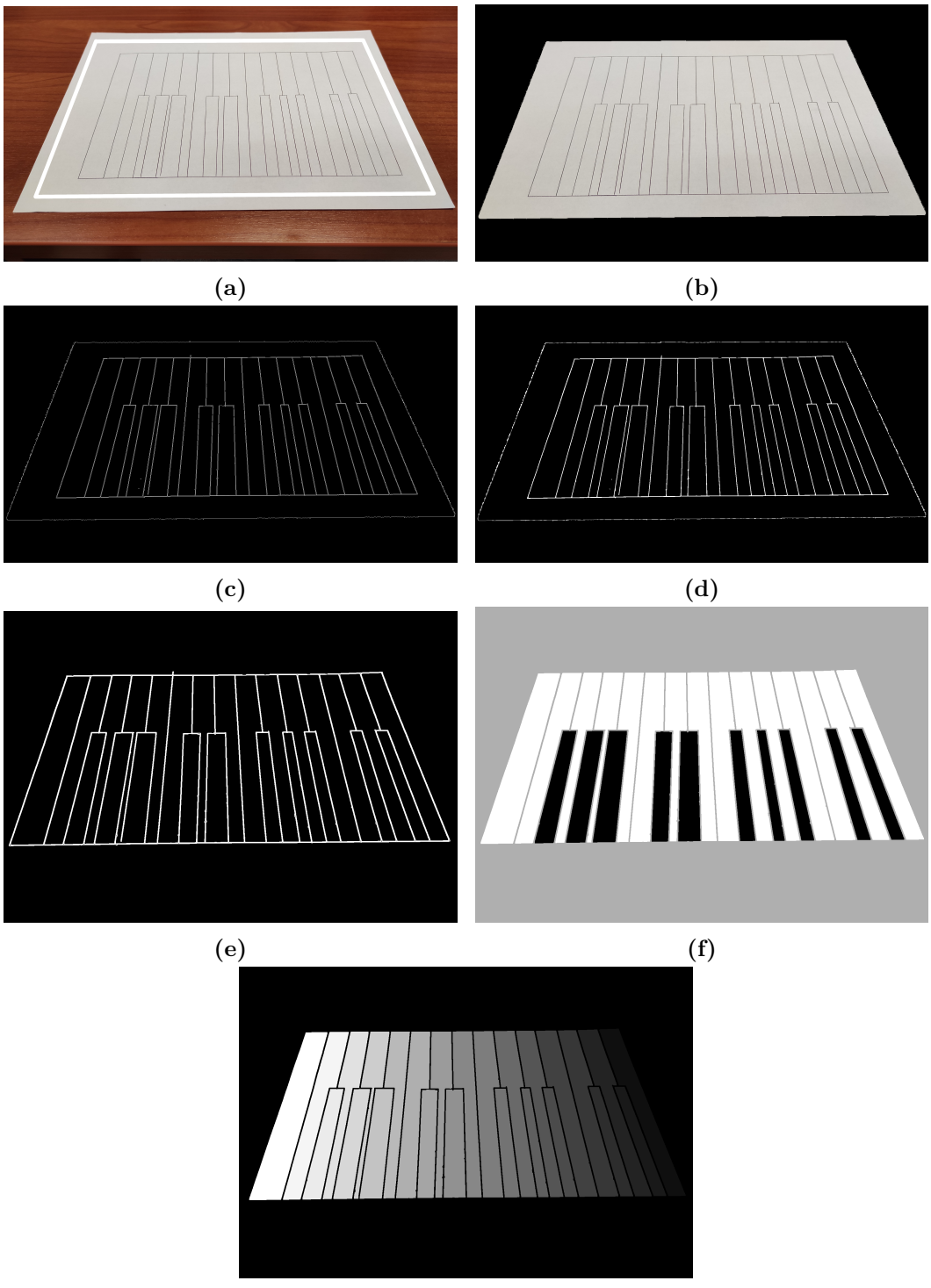
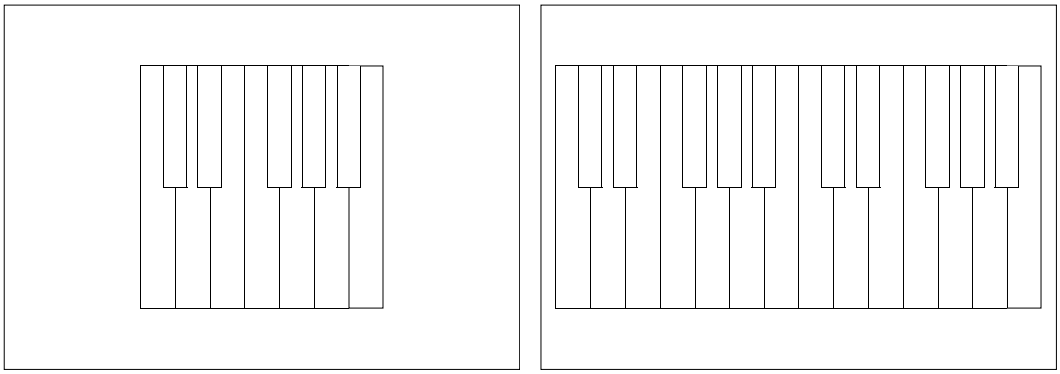


Figure 3.6. Keyboard detection pipeline. (a) Perimeter; (b) Keyboard after background subtraction; (c) Canny edge; (d) Canny edge after morphological closure; (e) Contour detection (after filter); (f) Tiles overlay (not in transparency for visibility); (g) Notes indexes (brightness enhanced for visibility).



(a) (b)

Figure 3.7. Keyboard presets different keyboard sizes. (a) One octave keyboard; (b) Two octaves keyboard.

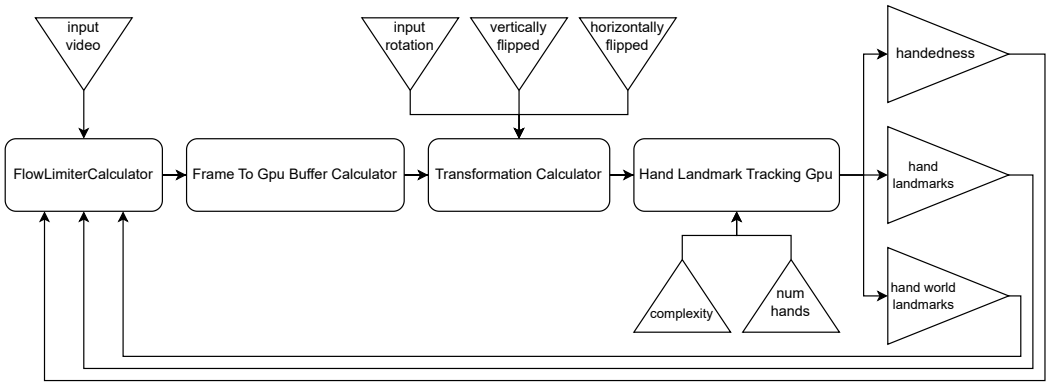


Figure 3.8. MediaPipe graph used in the application.

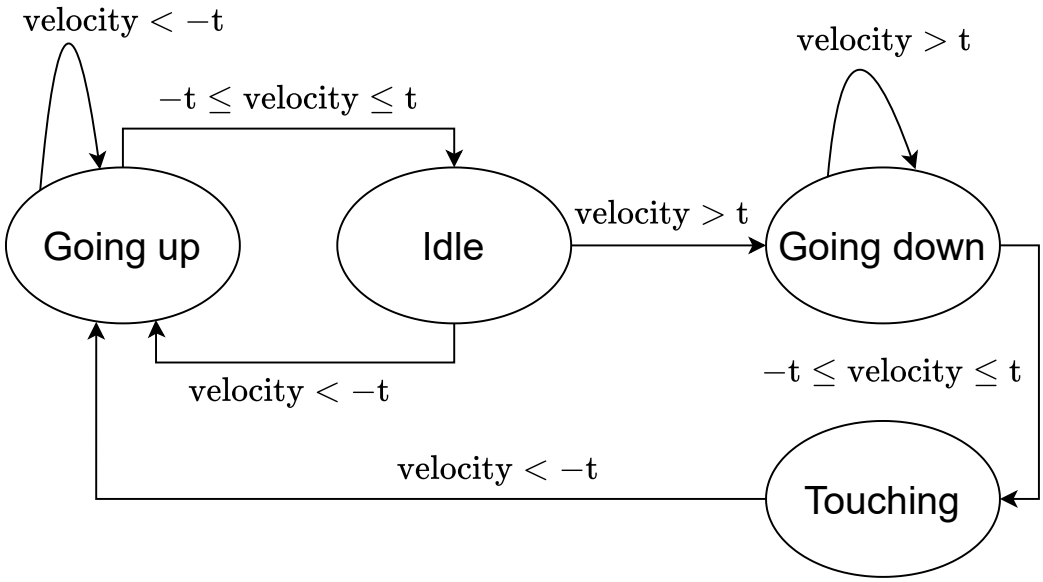


Figure 3.9. The automata used to detect finger touch.

Figure 3.10. Application during detection phase, with perimeter visible.

Figure 3.11. Application after keyboard detection, with keyboard overlay visible.

Chapter 4

Results

In this chapter, we present the findings from the experiments conducted to evaluate the performance and usability of the mixed reality musical keyboard application, Keyrtual. The results are structured into two primary sections: quantitative results, which measure the technical performance of the keyboard detection and real-time interaction processes, and qualitative results, which assess user feedback based on perceived ease of use and experience. Through these experiments, we aim to establish the effectiveness of the application and highlight areas for further development.

4.1 Quantitative results

The experiments carried out on the application concern the two fundamental phases of the pipeline: the keyboard detection phase and the real-time phase.

4.1.1 Keyboard detection

Experiments to evaluate the keyboard detection algorithm were carried out on 3 different types of keyboard: a hand-drawn keyboard, a keyboard drawn with a ruler as the one shown in Figure 3.10, and a keyboard printed using the preset provided by the application shown in Figure 3.7a.

The tests were carried out in a room with artificial light coming from the ceiling and natural light coming from an open window. For each of these keyboards, the test was carried out by changing the rotation and position of the keyboard with respect to the natural light 4 times, performing the keyboard detection 3 times for each position, for a total of 12 detections per keyboard type and 36 detections in total.

The evaluation of the experiments consists of comparing the result obtained from the keyboard detection, such as the one shown in Figure 3.6f, with the optimal result we would expect to obtain from a perfect detector. Of course, we are not able to obtain the optimal result with an algorithm, otherwise we would use that same algorithm to do the detection. So to create the ground truth we used a photo editing software to manually colour in black and white exactly the spaces occupied by the keys.

Once we had created the ground truth for all 36 photos, we applied the keyboard

detection algorithm to each of them and compared it to the ground truth by doing a simple subtraction of colours to highlight individual differing pixels. To calculate the pixel accuracy of our detection algorithm we used the following formula

$$Accuracy = \frac{TP}{n}$$

where TP (true positives) is the number of pixels correctly classified by the algorithm and n is the total number of pixels within the area bounded by the perimeter, as can be seen in Figure 3.10. We used the number of pixels inside the perimeter and not the total number of pixels in the image because doing so would have produced higher but untrue percentages, as the algorithm only searches for the keyboard in the area of the image bounded by the perimeter and not in the entire image.

As can be seen in Table 4.1, the results for the detection algorithm are excellent for each type of keyboard.

	Pixel accuracy
Hand-drawn	95.00%
Drawn with ruler	96.74%
Printed	97.69%

Table 4.1. Pixel accuracy of keyboard detection algorithm

Unfortunately, we did not conduct the same experiment for the keyboard detection algorithm in the old prototype, so we cannot compare the results. However, based on our experience with both algorithms, we can state with absolute certainty that the new algorithm is much better performing than the old one, which was only able to detect straight lines, needed the additional step of background subtraction, and didn't even produce a visible result like the new one does.

4.1.2 Real-time phase

To evaluate the real-time phase we used a method similar to the one used in the old prototype: we played a series of notes on the keyboard and counted the times in which the application reacted correctly or not.

In order to evaluate the performance of the application's real-time phase according to its accuracy, precision, and recall, we needed to calculate the amount of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). The meaning of these terms in our application is schematized in Table 4.2.

	User plays	User does not play
Application plays	TP	FP
Application does not play	FN	TN

Table 4.2. Meaning of TP, TN, FP and FN in our application

To achieve the maximum precision in calculating these values, we modified the real-time algorithm to be able to save each frame associated with a label indicating whether in that frame the algorithm detected a touch or not. After the recording,

we manually checked each frame with the corresponding label and counted if it was a TP, TN, FP or FN.

In this way we carried out 3 experiments. In each experiment the user had to press a finger on all the keys of the keyboard in order from left to right. The results of the experiments are shown in Table 4.3, together with the results from the old prototype.

	Accuracy	Precision	Recall
Old prototype	90.27%	56.87%	99.40%
New version	87.40%	80.23%	81.18%

Table 4.3. Comparison of real-time phase between old prototype and current version

The comparison between the old and the new version highlights how the real-time phase of the application is now much more precise than before, and on average more stable and reliable. The recall of the new version is much lower than before, but this is because in the old version there were very few false negatives thanks to the presence of the Leap Motion Controller.

4.2 Qualitative results

We also retrieved results about the perceived usability of our application with the same method used for the old prototype. We asked 15 participants to play with the application for 5 minutes, from the keyboard detection phase to the playing phase, and to fill out the UMUX questionnaire (Usability Metric for User Experience [24]) at the end. Among the participants involved in the experiment, there were people with different musical background knowledge: 6 had no musical knowledge at all, 6 had little musical knowledge mainly from middle school and high school, and 3 were professional piano players.

The results of the questionnaire are shown in Table 4.4, together with the results from the old prototype.

	Minimum	Average	Maximum
Old prototype	70.83%	79.86%	87.50%
New version	84.11%	92.28%	97.91%

Table 4.4. Comparison of usability between old prototype and current version

These results highlight how the porting from computer to smartphone, the elimination of all additional hardware, and the simplification of the keyboard detection procedure have greatly improved the usability and portability of the entire system.

Chapter 5

Conclusions

In this work, progress was made on the creation of a mixed reality application dedicated to teaching music in schools, especially in the first approach to the subject. The work starts with an existing prototype, which had the shortcoming of requiring a computer and specific hardware to run, and brings it to a mobile device, a smartphone, accessible to all and available at affordable prices. The software can also run without problems on cheap and not too recent devices, while still achieving performance similar to that of top-of-the-range devices.

The biggest improvements were found in the general usability of the application, thanks to the porting to smartphones and the elimination of all external hardware, and in the keyboard detection algorithm, which achieves almost perfect results on any type of keyboard without the need for any special lighting measures and now consists of a single step.

Some progress was made also in the playing phase, which is now more precise and reliable on average, but there is still much room for improvement.

5.1 Future development

In the next stages of development we plan to improve the touch recognition algorithm to achieve better precision and accuracy. This will be done by completely changing the keyboard detection phase from the current single-shot phase to a slightly longer procedure, but one that we hope will lead to better results.

The idea is to use plane detection techniques to locate the table with the paper on it, do the keyboard detection from above, and then move the phone to a very low angle, almost parallel to the table, tracking the position of the paper (and thus the keyboard) in 3D space while the phone moves to the new position.

This would allow us to have a perfect view of the distance of the fingers from the sheet, greatly increasing the accuracy of touch recognition, while maintaining full information on the position of the keys thanks to plane detection and tracking of the sheet during the transition from 90° to the flat position.

Plane detection and keyboard tracking can be made very accurate by the fact that the size of the sheet is known, being an ordinary A4 sheet. View from the flat position is shown in Figure 5.1.

As an alternative to the above strategy, it would be possible to develop an ad-hoc

neural network, suitable for running on mobile devices, with the aim of detecting, from the RGB image alone, which fingers are pressed on the surface of the sheet. This neural network, if developed, will be trained with a dataset of images of hands in different positions and with different fingers pressed, and will be able to return, for each frame, which fingers are pressed and on which keys.

We also plan to assess the feasibility of porting it to iOS, to expand the pool of users that our application can reach.

Figure 5.1. Flat view of the keyboard, still in development.

Bibliography

[1] Akhmalludin, H. and Ayu, M. A. Mobile based augmented reality to improve learning of volcanology for high school students. In *2019 5th International Conference on Computing Engineering and Design (ICCED)*, pp. 1–6 (2019). doi:10.1109/ICCED46541.2019.9161130.

[2] Ali, D. F., Johari, N., Omar, M., and Sunar, M. S. Armlaapps: Augmented reality application in microeconomics. In *2020 6th International Conference on Interactive Digital Media (ICIDM)*, pp. 1–5 (2020). doi:10.1109/ICIDM51048.2020.9339660.

[3] Almajalid, R., Shan, J., Du, Y., and Zhang, M. Development of a deep-learning-based method for breast ultrasound image segmentation. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1103–1108 (2018). doi:10.1109/ICMLA.2018.00179.

[4] Almetwally, I. and Mallem, M. Real-time tele-operation and tele-walking of humanoid robot nao using kinect depth camera. In *2013 10th IEEE INTERNATIONAL CONFERENCE ON NETWORKING, SENSING AND CONTROL (ICNSC)*, pp. 463–466 (2013). doi:10.1109/ICNSC.2013.6548783.

[5] Alshammari, M. and Mezher, M. A modified convolutional neural networks for mri-based images for detection and stage classification of alzheimer disease. In *2021 National Computing Colleges Conference (NCCC)*, pp. 1–7 (2021). doi:10.1109/NCCC49330.2021.9428810.

[6] Avalos, J., Cortez, S., Vasquez, K., Murray, V., and Ramos, O. E. Telepresence using the kinect sensor and the nao robot. In *2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS)*, pp. 303–306 (2016). doi:10.1109/LASCAS.2016.7451070.

[7] Avola, D., Cinque, L., Emam, E., Fontana, F., Foresti, G. L., Marini, M. R., and Pannone, D. Hand gesture recognition exploiting handcrafted features and LSTM. In *Image Analysis and Processing - ICIAP 2023 - 22nd International Conference, ICIAP 2023, Udine, Italy, September 11-15, 2023, Proceedings, Part I* (edited by G. L. Foresti, A. Fusiello, and E. R. Hancock), vol. 14233 of *Lecture Notes in Computer Science*, pp. 500–511. Springer (2023). Available from: https://doi.org/10.1007/978-3-031-43148-7_42, doi:10.1007/978-3-031-43148-7_42.

[8] Avola, D., Cinque, L., Fagioli, A., Foresti, G. L., Marini, M. R., Mecca, A., and Pannone, D. Medicinal boxes recognition on a deep transfer learning augmented reality mobile application. In *Image Analysis and Processing - ICIAP 2022 - 21st International Conference, Lecce, Italy, May 23-27, 2022, Proceedings, Part I* (edited by S. Sclaroff, C. Distante, M. Leo, G. M. Farinella, and F. Tombari), vol. 13231 of *Lecture Notes in Computer Science*, pp. 489–499. Springer (2022). Available from: https://doi.org/10.1007/978-3-031-06427-2_41, doi:10.1007/978-3-031-06427-2_41.

[9] Avola, D., Cinque, L., Foresti, G. L., and Marini, M. R. A novel low cybersickness dynamic rotation gain enhancer based on spatial position and orientation in virtual environments. *Virtual Real.*, **27** (2023), 3191. Available from: https://doi.org/10.1007/s10055-023-00865-1, doi:10.1007/S10055-023-00865-1.

[10] Avola, D., Cinque, L., Marini, M. R., Princic, A., and Venanzi, V. Keyrtual: A lightweight virtual musical keyboard based on rgb-d and sensors fusion. In *Computer Analysis of Images and Patterns* (edited by N. Tsapatsoulis, A. Lanitis, M. Pattichis, C. Pattichis, C. Kyrkou, E. Kyriacou, Z. Theodosiou, and A. Panayides), pp. 182–191. Springer Nature Switzerland, Cham (2023). ISBN 978-3-031-44240-7.

[11] Badolato, N. and Scalfaro, A. L'educazione musicale nella scuola italiana dall'unità a oggi. *Musica Docta*, **3** (2013), 87–99. Available from: https://musicadocta.unibo.it/article/view/4022, doi:10.6092/issn.2039-9715/4022.

[12] Blair, T. B. and Davis, C. E. Innovate engineering outreach: A special application of the xbox 360 kinect sensor. In *2013 IEEE Frontiers in Education Conference (FIE)*, pp. 1279–1283 (2013). doi:10.1109/FIE.2013.6685036.

[13] Bouatrous, A., Meziane, A., Zenati, N., and Hamitouche, C. An interactive virtual reality system based on leap motion controller for hand motor rehabilitation. In *2024 8th International Conference on Image and Signal Processing and their Applications (ISPA)*, pp. 1–5 (2024). doi:10.1109/ISPA59904.2024.10536781.

[14] Boyko, N., Basystiuk, O., and Shakhovska, N. Performance evaluation and comparison of software for face recognition, based on dlib and opencv library. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pp. 478–482 (2018). doi:10.1109/DSMP.2018.8478556.

[15] Bradski, G. and Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media (2008). ISBN 9780596554040. Available from: https://books.google.it/books?id=seAgiOfu2EIC.

[16] Brahmanage, G. and Leung, H. Outdoor rgb-d mapping using intelrealsense. In *2019 IEEE SENSORS*, pp. 1–4 (2019). doi:10.1109/SENSORS43011.2019.8956916.

[17] Canossa, A. Interview with nicholas francis and thomas hagen from unity technologies. *Game Analytics*, (2013), 137. doi:10.1007/978-1-4471-4769-5_8.

[18] Chen, W.-J., Chiu, C.-T., and Lin, T.-C. Landmark-based adversarial network for rgb-d pose invariant face recognition. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–5 (2023). doi:10.1109/AICAS57966.2023.10168669.

[19] Chuan, C.-H., Regina, E., and Guardino, C. American sign language recognition using leap motion sensor. In *2014 13th International Conference on Machine Learning and Applications*, pp. 541–544 (2014). doi:10.1109/ICMLA.2014.110.

[20] Dcunha, M., Naik, N., Francis, I., and Mulla, S. Pneumonia detection in x-rays using opencv and deep learning. In *2021 International Conference on Communication information and Computing Technology (ICCICT)*, pp. 1–6 (2021). doi:10.1109/ICCICT50803.2021.9510180.

[21] Deb, S. and Nama, T. Interactive boolean logic learning using leap motion. In *2018 IEEE Tenth International Conference on Technology for Education (T4E)*, pp. 220–222 (2018). doi:10.1109/T4E.2018.00060.

[22] El-Gohary, M., Aborizka, M., El-Sheikh, M., and El-Nagar, Z. Shopping training to autistic children and adolescents for enhancing daily life shopping skills using virtual reality and leap motion. In *2022 1st IEEE International Conference on Cognitive Aspects of Virtual Reality (CVR)*, pp. 000055–000060 (2022). doi:10.1109/CVR55417.2022.9967646.

[23] Farias, G., Torres, E., Fabregas, E., Vargas, H., Dormido-Canto, S., and Dormido, S. Navigation control of the khepera iv model with opencv in v-rep simulator. In *2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, pp. 1–6 (2018). doi:10.1109/ICA-ACCA.2018.8609740.

[24] Finstad, K. The usability metric for user experience. *Interacting with Computers*, **22** (2010), 323. doi:10.1016/j.intcom.2010.04.004.

[25] G, D. D., Doss, S., S, S., and N, S. C. Precision mri brain tumor identification: Leveraging advanced techniques for accurate classification. In *2024 International Conference on Computing and Data Science (ICCDS)*, pp. 1–5 (2024). doi:10.1109/ICCDS60734.2024.10560447.

[26] Google. Mediapipe calculators. Available from: https://github.com/google-ai-edge/mediapipe/blob/master/docs/framework_concepts/calculators.md.

[27] Gritsenko, P., Gritsenko, I., Seidakhmet, A., and Abduraimov, A. Generation of rgb-d data for slam using robotic framework v-rep. vol. 1880, p. 060005 (2017). doi:10.1063/1.5000659.

[28] Gulati, R. and Handa, K. Towards development of a virtual reality game for children with autism spectrum disorder. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pp. 452–456 (2021). doi:10.1109/ICICV50876.2021.9388374.

[29] Guo, R., Cui, J., Zhao, W., and Li, S. Ai and ar based interface for piano training. In *2020 International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 328–330 (2020). doi:10.1109/ICVRV51359.2020.00087.

[30] Hu, J., Hu, R., Wang, Z., Gong, Y., and Duan, M. Kinect depth map based enhancement for low light surveillance image. In *2013 IEEE International Conference on Image Processing*, pp. 1090–1094 (2013). doi:10.1109/ICIP.2013.6738225.

[31] Hussain, A., Shakeel, H., Hussain, F., Uddin, N., and Ghouri, T. Unity game development engine: A technical survey. *University of Sindh Journal of Information and Communication Technology*, **4** (2020).

[32] Ip, H., Law, K., and Kwong, B. Cyber composer: Hand gesture-driven intelligent music composition and generation. In *11th International Multimedia Modelling Conference*, pp. 46–52 (2005). doi:10.1109/MMMC.2005.32.

[33] Jha, S. and Trivedi, P. An automated video surveillance system using viewpoint feature histogram and cuda-enabled gpus. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1812–1816 (2013). doi:10.1109/ICACCI.2013.6637456.

[34] Jiao, J., Yuan, L., Tang, W., Deng, Z., and Wu, Q. A post-rectification approach of depth images of kinect v2 for 3d reconstruction of indoor scenes. *ISPRS International Journal of Geo-Information*, **6** (2017), 349. doi:10.3390/ijgi6110349.

[35] Jommuangbut, J. and Sritrakulchai, K. Development of the human following robot control system using hd webcam. In *2018 International Electrical Engineering Congress (iEECON)*, pp. 1–4 (2018). doi:10.1109/IEECON.2018.8712220.

[36] Kaviyaraj, R. and Uma, M. Augmented reality application in classroom: An immersive taxonomy. In *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 1221–1226 (2022). doi:10.1109/ICSSIT53264.2022.9716325.

[37] Khan, N. H., Tegnander, E., Dreier, J. M., Eik-Nes, S., Torp, H., and Kiss, G. Automatic detection and measurement of fetal femur length using a portable ultrasound device. In *2015 IEEE International Ultrasonics Symposium (IUS)*, pp. 1–4 (2015). doi:10.1109/ULTSYM.2015.0486.

[38] Khan, N. H., Tegnander, E., Dreier, J. M., Eik-Nes, S., Torp, H., and Kiss, G. Automatic measurement of the fetal abdominal section on a portable ultrasound machine for use in low and middle income countries.

In *2016 IEEE International Ultrasonics Symposium (IUS)*, pp. 1–4 (2016). doi:10.1109/ULTSYM.2016.7728557.

[39] Khoshelham, K. and Elberink, S. O. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, **12** (2012), 1437. Available from: https://www.mdpi.com/1424-8220/12/2/1437, doi:10.3390/s120201437.

[40] Koca, B. A., Çubukçu, B., and Yüzgeç, U. Augmented reality application for preschool children with unity 3d platform. In *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1–4 (2019). doi:10.1109/ISMSIT.2019.8932729.

[41] Kumar, P., Jaiswal, A., Deepak, B., and Reddy, G. *Hand Gesture-Based Stable PowerPoint Presentation Using Kinect*, pp. 81–94 (2018). ISBN 978-981-10-3372-8. doi:10.1007/978-981-10-3373-5_7.

[42] Kumari Sirivarshitha, A., Sravani, K., Priya, K. S., and Bhavani, V. An approach for face detection and face recognition using opencv and face recognition libraries in python. In *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, pp. 1274–1278 (2023). doi:10.1109/ICACCS57279.2023.10113066.

[43] Kunej, D. and Turk, I. New perspectives on the beginnings of music: Archeological and musicological analysis of a middle paleolithic bone "flute". In *The Origins of Music*. The MIT Press (1999). ISBN 9780262285698. Available from: https://doi.org/10.7551/mitpress/5190.003.0020, arXiv:https://direct.mit.edu/book/chapter-pdf/2298273/9780262285698_cao.pdf, doi:10.7551/mitpress/5190.003.0020.

[44] Lee, J., Gu, H., Kim, H., Kim, J., Kim, H., and Kim, H. Interactive manipulation of 3d objects using kinect for visualization tools in education. In *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, pp. 1220–1222 (2013). doi:10.1109/ICCAS.2013.6704175.

[45] Li, Z., Fan, D., Wang, H., Lu, Z., and Liu, C. Piano beginner: A glove-based finger training vr application. In *2022 International Symposium on Control Engineering and Robotics (ISCER)*, pp. 262–265 (2022). doi:10.1109/ISCER55570.2022.00052.

[46] Lugaresi, C., et al. Mediapipe: A framework for perceiving and processing reality. In *Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR)*, vol. 2019 (2019).

[47] Luppino, G., Bosisio, L., Conese, C., Fabris, D. M., and Tarabini, M. Metrology of a monocular vision system for markers localization and tracking. In *2022 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*, pp. 6–10 (2022). doi:10.1109/MetroInd4.0IoT54413.2022.9831684.

[48] Matzek, E., Yankee, T., Kohler, O., Lipke-Perry, T., Banerjee, N. K., and Banerjee, S. Vrmonic: A vr piano playing form trainer. In *2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, pp. 330–334 (2024). doi:10.1109/AIxVR59861.2024.00056.

[49] Monroy, F. L., Hussein, R., and Mamishev, A. Accuracy of smartphone depth cameras in stoma shape extraction for wafer fitting. In *2022 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pp. 40–45 (2022). doi:10.23919/SPA53010.2022.9927778.

[50] Naglot, D. and Kulkarni, M. Real time sign language recognition using the leap motion controller. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3, pp. 1–5 (2016). doi:10.1109/INVENTIVE.2016.7830097.

[51] Nagpurkar, V., Pattankar, N., Nayak, T., D'Souza, A., and Henriques, N. Guitarguru: A realtime guitar chords detection system. In *2023 International Conference on Communication System, Computing and IT Applications (CSCITA)*, pp. 107–112 (2023). doi:10.1109/CSCITA55725.2023.10104798.

[52] Nainggolan, E. R., Asymar, H. H., Nalendra, A. R. A., Anton, Sulaeman, F., Sidik, Radiyah, U., and Susafa'ati. The implementation of augmented reality as learning media in introducing animals for early childhood education. In *2018 6th International Conference on Cyber and IT Service Management (CITSM)*, pp. 1–6 (2018). doi:10.1109/CITSM.2018.8674350.

[53] Nainggolan, F. L., Siregar, B., and Fahmi, F. Anatomy learning system on human skeleton using leap motion controller. In *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, pp. 465–470 (2016). doi:10.1109/ICCOINS.2016.7783260.

[54] Nguyen, V. T. and Dang, T. Setting up virtual reality and augmented reality learning environment in unity. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pp. 315–320 (2017). doi:10.1109/ISMAR-Adjunct.2017.97.

[55] Nicola, S., Stoicu-Tivadar, L., Virag, I., and Crişan-Vida, M. Leap motion supporting medical education. In *2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC)*, pp. 153–156 (2016). doi:10.1109/ISETC.2016.7781080.

[56] Oka, A. and Hashimoto, M. Marker-less piano fingering recognition using sequential depth images. In *The 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, pp. 1–4 (2013). doi:10.1109/FCV.2013.6485449.

[57] Pagliari, D. and Pinto, L. Calibration of kinect for xbox one and comparison between the two generations of microsoft sensors. *Sensors*, **15** (2015), 27569. Available from: https://www.mdpi.com/1424-8220/15/11/27569, doi:10.3390/s151127569.

[58] Park, H., Lee, J.-S., and Ko, J. Achieving real-time sign language translation using a smartphone's true depth images. In *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pp. 622–625 (2020). doi:10.1109/COMSNETS48256.2020.9027420.

[59] Pastukh, V., Beshley, M., Chopyk, P., Beshley, H., Ivanochko, I., and Gregus, M. Smartphone visual assistance system with ai-based depth estimation and embedded/cloud voice controls. In *2023 IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT)*, pp. 87–91 (2023). doi:10.1109/AICT61584.2023.10452675.

[60] Patil, C., Mathura, M. G., Madhumitha, S., David, S. S., Fernandes, M., Venugopal, A., and Unnikrishnan, B. Using image processing on mri scans. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pp. 1–5 (2015). doi:10.1109/SPICES.2015.7091517.

[61] Patil, J. V. and Bailke, P. Real time facial expression recognition using realsense camera and ann. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 2, pp. 1–6 (2016). doi:10.1109/INVENTIVE.2016.7824820.

[62] Pedrosa, S. and Costa, A. Emg based midi controller. In *2023 7th International Young Engineers Forum (YEF-ECE)*, pp. 106–111 (2023). doi:10.1109/YEF-ECE58420.2023.10209314.

[63] Perdana, I. Teaching elementary school students new method of music performance with leap motion. In *2014 International Conference on Virtual Systems & Multimedia (VSMM)*, pp. 273–277 (2014). doi:10.1109/VSMM.2014.7136655.

[64] Rahchamani, M., Soboute, M. I., Samadzadehaghdam, N., and Abadi, B. M. Developing and evaluating a low-cost tracking method based on a single camera and a large marker. In *2018 25th National and 3rd International Iranian Conference on Biomedical Engineering (ICBME)*, pp. 1–5 (2018). doi:10.1109/ICBME.2018.8703592.

[65] Rakib, N. F., Mahmood, N. H., Ramli, N., Zakaria, N. A., and Razak, M. A. A. Preliminary results of hand rehabilitation for post stroke patient using leap motion-based virtual reality. In *2020 IEEE Student Conference on Research and Development (SCOReD)*, pp. 259–262 (2020). doi:10.1109/SCOReD50371.2020.9250985.

[66] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **44** (2022), 1623. doi:10.1109/TPAMI.2020.3019967.

[67] Recalde, M., Losito, A., Morillo, M. J., Pancho, C., Reyes, J. B., and Aranas, J. Creating an accessible future: Developing a sign language

to speech translation mobile application with mediapipe hands technology. In *2023 10th International Conference on ICT for Smart Society (ICISS)*, pp. 1–6 (2023). doi:10.1109/ICISS59129.2023.10291709.

[68] Saaidon, N., Sediono, W., and Sophian, A. Altitude tracking using colour marker based navigation system for image guided surgery. In *2016 International Conference on Computer and Communication Engineering (ICCCE)*, pp. 465–469 (2016). doi:10.1109/ICCCE.2016.103.

[69] Salihbašić, A. and Orehovački, T. Development of android application for gender, age and face recognition using opencv. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1635–1640 (2019). doi:10.23919/MIPRO.2019.8756700.

[70] Sathyamoorthy, M., Dhanaraj, R. K., Vanitha, C. N., and Krishnasamy, L. Augmented reality based medical education. In *2023 Intelligent Computing and Control for Engineering and Business Systems (ICCEBS)*, pp. 1–6 (2023). doi:10.1109/ICCEBS58601.2023.10449124.

[71] Serafin, S., Adjorlu, A., Nilsson, N., Thomsen, L., and Nordahl, R. Considerations on the use of virtual and augmented reality technologies in music education. In *2017 IEEE Virtual Reality Workshop on K-12 Embodied Learning through Virtual & Augmented Reality (KELVAR)*, pp. 1–4 (2017). doi:10.1109/KELVAR.2017.7961562.

[72] Serafin, S., Gelineck, S., Böttcher, N., and Martinussen, L. Virtual reality instruments capable of changing physical dimensions in real-time. In *Enactive 2005* (2005). Enactive 2005 ; Conference date: 17-11-2005 Through 18-11-2005.

[73] Servi, M., Profili, A., Furferi, R., and Volpe, Y. Comparative evaluation of intel realsense d415, d435i, d455, and microsoft azure kinect dk sensors for 3d vision applications. *IEEE Access*, **12** (2024), 111311. doi:10.1109/ACCESS.2024.3441238.

[74] Shang, K. and Wang, Z. A music performance method based on visual gesture recognition. In *2022 China Automation Congress (CAC)*, pp. 2624–2631 (2022). doi:10.1109/CAC57257.2022.10055445.

[75] Singh, V., Garg, V., and Soni, L. Exploring the stars: How augmented reality transforms the teaching of astronomic concepts in school. In *2024 IEEE International Conference for Women in Innovation, Technology & Entrepreneurship (ICWITE)*, pp. 71–77 (2024). doi:10.1109/ICWITE59797.2024.10503216.

[76] Sreenath, S., Daniels, D. I., Ganesh, A. S. D., Kuruganti, Y. S., and Chittawadigi, R. G. Monocular tracking of human hand on a smart phone camera using mediapipe and its application in robotics. In *2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC)*, pp. 1–6 (2021). doi:10.1109/R10-HTC53172.2021.9641542.

[77] Stamou, L. Plato and aristotle on music and music education: Lessons from ancient greece. *International Journal of Music Education*, **os-39** (2002), 3. Available from: https://doi.org/10.1177/025576140203900102, arXiv:https://doi.org/10.1177/025576140203900102, doi:10.1177/025576140203900102.

[78] Sultanov, R., Lavrenov, R., Sulaiman, S., Bai, Y., Svinin, M., and Magid, E. Object detection methods for a robot soccer. In *2023 7th International Conference on Information, Control, and Communication Technologies (ICCT)*, pp. 1–5 (2023). doi:10.1109/ICCT58878.2023.10347064.

[79] Sunil, S. and Kumaran Nair, S. S. An educational augmented reality app to facilitate learning experience. In *2017 International Conference on Computer and Applications (ICCA)*, pp. 279–282 (2017). doi:10.1109/COMAPP.2017.8079771.

[80] Suzuki, S. and be, K. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, **30** (1985), 32. Available from: https://www.sciencedirect.com/science/article/pii/0734189X85900167, doi:https://doi.org/10.1016/0734-189X(85)90016-7.

[81] Villena Martinez, V., Guilló, A., Azorin-Lopez, J., Saval-Calvo, M., Mora-Pascual, J., Rodríguez, J., and Garcia-Garcia, A. A quantitative comparison of calibration methods for rgb-d sensors using different technologies. *Sensors*, **17** (2017), 243. doi:10.3390/s17020243.

[82] Weichert, F., Bachmann, D., Rudak, B., and Fisseler, D. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, **13** (2013), 6380. Available from: https://www.mdpi.com/1424-8220/13/5/6380, doi:10.3390/s130506380.

[83] Yusof, Y. W. M., Rahim, A. I. A., Kamaluddin, N., and Kassim, M. Augmented reality apps on electronic devices for education using image-based technique. In *2023 IEEE 14th Control and System Graduate Research Colloquium (ICSGRC)*, pp. 192–197 (2023). doi:10.1109/ICSGRC57744.2023.10215481.

[84] Zahid Iqbal, M. and Campbell, A. G. Agilest approach: Using machine learning agents to facilitate kinesthetic learning in stem education through real-time touchless hand interaction. *Telematics and Informatics Reports*, **9** (2023), 100034. Available from: https://www.sciencedirect.com/science/article/pii/S2772503022000329, doi:https://doi.org/10.1016/j.teler.2022.100034.

[85] Zeng, Y., Wang, H., Sha, M., Lin, G., Long, Y., and Liu, Y. Object detection algorithm of vein vessels in b-mode ultrasound images. In *2022 7th International Conference on Control and Robotics Engineering (ICCRE)*, pp. 180–183 (2022). doi:10.1109/ICCRE55123.2022.9770248.

[86] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., and Grundmann, M. Mediapipe hands: On-device real-time hand tracking (2020). Available from: https://arxiv.org/abs/2006.10214, arXiv:2006.10214.

[87] Zhang, J., Lu, Z., Li, W., and Liao, Q. A robust and fast 3d face reconstruction method using realsense camera. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 2691–2695 (2017). doi:10.1109/WiSPNET.2017.8300251.

[88] Zhong, H., Kanhere, S. S., and Chou, C. T. Quickfind: Fast and contact-free object detection using a depth sensor. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 1–6 (2016). doi:10.1109/PERCOMW.2016.7457114.