UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

DESIGNING RACING GAME CONTROLLER BY IMAGE-BASED HAND GESTURE RECOGNITION

Author: David López Adell

Director:	Dr. Meysam Madadi
Performed in:	Departament de Matemàtiques i Informàtica
Barcelona,	January 23, 2023

Abstract

This thesis is focused on exploring how hand gesture recognition can be used to replace controllers in racing games. The goal is to understand how to develop a system that is accurate, allowing for more responsive control when driving virtual vehicles. The first step of the research project is to analyze existing gesture recognition technologies, such as the Microsoft Kinect, and how they can be used in racing games.

By studying existing implementations, the thesis aims to gain key concepts that can help improve the user's experience. The research will also investigate various hardware requirements, such as camera placements and sensors, that would be necessary for the system to function effectively in a racing game environment. Once the research requirements are established, testing will be carried out to evaluate how effective gesture-based control systems are compared to traditional controllers. The results of these tests will be analyzed to evaluate how well the system performs compared to existing controller-based racing games.

The ultimate goal of this thesis is to create a more natural form of control that allows players to focus on the thrill of racing without worrying about button presses or joystick movements.

Resumen

Este trabajo de final de grado (TFG) se centra en explorar cómo el reconocimiento de gestos de las manos puede usarse para reemplazar los mandos en videojuegos de carreras. El objetivo es comprender cómo desarrollar un sistema preciso, permitiendo un control más sensible al conducir vehículos virtuales. El primer paso del proyecto de investigación es analizar las tecnologías de reconocimiento de gestos existentes, como Microsoft Kinect, y cómo se pueden usar en juegos de carreras.

Al estudiar las implementaciones existentes, la investigación pretende obtener conceptos clave que puedan ayudar a mejorar la experiencia del usuario. También se investigarán varios requisitos de hardware, como ubicaciones de cámaras y sensores, que serían necesarios para que el sistema funcione eficazmente en un entorno de juego de conducción. Una vez que se establezcan los requisitos de investigación, se realizarán pruebas para evaluar qué tan efectivos son los sistemas de control basados en gestos en comparación con los controladores tradicionales.

El objetivo final de esta tesis es crear una forma más natural de control que permita a los jugadores concentrarse en la emoción de la carrera sin preocuparse por los botones o los movimientos del joystick.

Contents

Chapter 1

Introduction

1.1 Context

Hand gesture recognition in video games has been a topic of increasing interest in the gaming industry over the past decade. [9] Hand gesture recognition technology is the ability to identify and track hand and finger motions by computers. It is a form of computer vision technology that allows machines to interpret human hand and finger motions. By analyzing and understanding the gestures, computers can simulate them in a gaming environment. It has been used in a wide variety of applications, including virtual reality (VR) [25] and augmented reality (AR) games, motion-controlled games, and gesture-controlled gaming consoles.

The development of hand gesture recognition technology in video games has taken many steps in recent years. Initially, this technology was limited to users providing pre-programmed finger commands. However, as the technology has grown, it has become possible to provide users with more precise and accurate recognition of their hand gestures. [5] For instance, recent advances in machine learning algorithms and the use of deep neural networks have enabled computers to identify and track hand and finger motions. [13]

Furthermore, the addition of cameras and trackers has made it possible for video game consoles and computers to understand the movements of players'hands, allowing for more interactive gaming experiences. [8] For example, some video games have implemented advanced hand gesture recognition technology that allows players to perform moves with their hands and fingers like they would in the real world. [2]

The most common type of hand gesture recognition used in videogames is known as skeletal tracking. [10] This method uses cameras and depth sensors to capture a 3D image of the player's body, including their hands and fingers. The software then analyzes the images to detect and interpret the hand gestures. [6]

Hand gesture recognition can be used to control a variety of game elements, including character movement, object manipulation, and menu navigation. It can also be used to detect gestures that are used to perform special actions, such as casting spells in role-playing games.

Hand gesture recognition technology has the potential to revolutionize the way people interact with their games. It can make gaming more immersive and engaging. [20] However, it is still in its early stages and there are many challenges that need to be addressed before it can be widely adopted. [27]

In summary, the use of hand gesture recognition in video games has a relatively short history, but has seen a rapid evolution with the advent of new technologies and improved computer vision algorithms. Early attempts at hand gesture recognition in video games were not very successful, but recent advancements in virtual reality and computer vision technology have led to a resurgence in interest and success in the use of hand gestures in video games.

1.2 Motivation

In the ever-evolving world of racing games, controller-based input is rapidly becoming obsolete. Innovations in hand gesture recognition are offering gamers a new way of controlling their favorite vehicles on the track. This thesis aims to explore how hand gesture recognition systems can be implemented in order to replace the traditional controller-based input systems in use today.

One of the most exciting developments in gaming is the emergence of hand gesture recognition as a means to control gaming experiences. This technology has the potential to revolutionize the way we play games, allowing for a more intuitive and immersive experience.

One of the main motivations for replacing controllers with hand gesture recognition in racing games is the potential for increased realism. Hand gestures are a natural and intuitive way for people to interact with their environment, and by incorporating these gestures into racing games, players can feel more like they are truly behind the wheel of a car. This is because gestures like turning the steering wheel or shifting gears are mapped directly to the player's physical movements, rather than to the press of a button or the movement of a joystick.

Hand gesture recognition has already been implemented in some gaming platforms, such as Microsoft Kinect, in these platforms, players use their hands to control on-screen characters or objects. Furthermore, the technology can also be used to add another layer of realism to racing games, by allowing players to control their vehicles through hand gestures rather than controllers.

1.3 Goals

This thesis will explore how hand gesture recognition can be used to replace controllers in racing games. It will discuss the potential benefits of this technology and how it can be implemented in modern gaming systems. It will also look at some possible drawbacks, such as hardware requirements, user fatigue, and potential compatibility issues with existing controllers. Finally, this thesis will provide suggestions on how hand gesture recognition can be used most effectively in racing games.

The first step of this research project is to identify and analyze existing gesture recognition technologies, such as the aforementioned Microsoft Kinect, and how they could be used in racing games. In order for these systems to provide an effective alternative to traditional controllers, they need to be accurate and responsive. Through this research, the thesis will aim to develop a system that is capable of detecting multiple input gestures concurrently with minimal latency, allowing for more responsive control when driving virtual vehicles.

The next step is to investigate existing racing games that have implemented gesture-based control systems and what techniques they have utilized in order to provide a successful gaming experience. By studying existing implementations, there may be key concepts and algorithms that can be gained and improved upon in order to further enhance the user's experience when controlling a vehicle via hand gestures. In addition, research should also be conducted into various hardware requirements for such a system, including camera placements, sensors, etc., in order for it to work effectively in a racing game environment.

Once the research requirements have been established, testing should be carried out in order to evaluate how effective gesture-based control systems are compared to traditional controllers. This includes testing accuracy against different combinations of input gestures as well as how user-friendly it is for all age groups. The results of these tests should then be analyzed in order to evaluate how well the system performs compared to existing controller-based racing games.

To sum up, this thesis aims to explore how hand gesture recognition systems can be effectively implemented into racing games in order to replace traditional controller-based inputs. Through researching existing technologies and testing various methods of integrating them into racing games, a successful system can be developed that provides an even more engaging and immersive experience than before. Ultimately, this thesis will strive towards creating a more natural form of control that allows players to focus on the thrill of racing without worrying about button presses or joystick movements.

Chapter 2

Related work

Hand gesture recognition is the process of identifying and interpreting a person's hand and finger movements and gestures to control a computer system or other technology. It is a form of non-verbal communication that involves the use of hand and finger motions to control and interact with a device. Hand gesture recognition technology has become increasingly popular in recent years, as it provides a convenient and intuitive way for users to interact with computers and other devices.

The applications of hand gesture recognition are numerous. [14] It has been used to control video games, control robotic arms, and even to interact with virtual reality environments. It is also increasingly being used for sign language interpretation, allowing users to communicate with computers without the need for verbal communication. Finally, hand gesture recognition is being explored as a way to interact with vehicles and other machines, allowing for more natural, intuitive interactions as researched by E. Ohn-Bar et al. [17]

The implementation of this technology is a multi-step process that involves a range of activities. The initial step in this process is capturing the gestures of the user through specialized hardware such as cameras or sensors. Once the gestures are captured, they must be processed to accurately identify the specific gestures being performed. This step is crucial as it lays the foundation for the subsequent steps. [23] Following the identification of the gestures, the next step is to map them to specific actions that the computer can perform. This might be opening a file, closing a window, or simulating a joystick movement.

2.1 Pipeline

The task of communicating with a computer via hand gestures is notably complex, as noted by Murthy et al. [14] Even though the multiple strategies follow a similar pipeline, what makes them different is the dataset and how we decide to control our in-game car.

In this project we propose a pipeline, depicted in Figure 2.1. This pipeline outlines the entire process, from acquiring images or videos of hand gestures to mapping the corresponding output to an in-game action. The primary stages that we will delve into are: Obtaining images/videos and pre-process them (Section 2.2), Process the data with an accurate convolutional neural network (CNN) (Section 2.3), Produce an in-game action according to the detected gesture (Section 2.4)

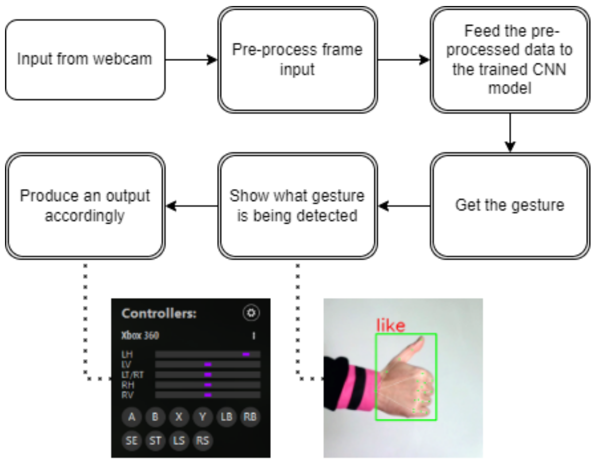
Input from webcam
Pre-process frame input
Feed the pre-processed data to the trained CNN model
Get the gesture
Show what gesture is being detected
Produce an output accordingly
Controllers:
Xbox 360
LH
LV
LT/RT
RH
RV
A
B
X
Y
LB
RB
SE
ST
LS
RS
like

Figure 2.1: Flowchart of the proposed algorithm. The normal rectangles represent inputs and the double rectangles the main steps of the pipeline. The images serve as visual examples of the outputs.

The steps of the previous diagram (Figure 2.1) can be summarized in the following way:

1. The initial phase concentrates on the pre-processing of the camera input. This step is not essential, however, it plays a vital role in enhancing the efficiency of the following stages, for instance, by decreasing the size of the image which can decrease the computational load on the system.

2. The next step of the hand gesture recognition pipeline involves utilizing the pre-processed frames as inputs to a Convolutional Neural Network (CNN). This step is essential as it is the CNN that will be responsible for recognizing the hand gestures in the images or videos.

3. The third stage of the pipeline is focused on extracting the accurate gesture from the output of the Convolutional Neural Network. After the frames have been processed through the CNN, the network will output a label for the gesture it has recognized. This stage is crucial as it is at this point that the system can determine what gesture is being made by the individual.

4. With the identification of the hand gesture, the system can now proceed to the final step of the hand gesture recognition pipeline, which is to produce an in-game action

according to the detected gesture. The goal of this step is to translate the recognized gesture into a specific action that can be executed within the game. One way to accomplish this can be to simulate traditional controller inputs. This approach is particularly useful as it allows for compatibility with a wide range of game, since most games are designed to respond to inputs from traditional controllers.

It is worth noting that the pipeline described is not specially tailored to explore the use of hand gesture recognition as a replacement for controllers in racing games. However, this application of hand gesture recognition has been chosen as it provides a clear and specific use case for the technology, and it allows for the examination of the potential benefits and limitations of using hand gestures as a means of input in racing games. Racing games are a perfect scenario to test hand gesture recognition as the inputs in this type of games usually require a lot of degree of freedom, especially for the turning of a car. [21]

2.2 Image acquisition & pre-processing

To begin, it's important to recognize that input for detecting hand gestures can come from various sources, not just a camera. This flexibility in input sources allows for greater versatility in the types of systems and applications that can utilize hand gesture recognition technology.

An early method for obtaining input for gesture detection was the use of gloves. This approach required the use of a specialized hardware device that would collect data from the joints in the hand. While this method had its advantages, such as faster data collection and the need for less input data due to the information being transferred directly from the fingers, it also had significant downsides. [19] One of the main drawbacks was the cost and weight of the gloves, making them impractical for natural interaction and usage. [3]

An alternative approach for obtaining input for gesture detection is through the use of computer vision. This method involves acquiring images or videos from a camera or similar device and analyzing the input to detect gestures. One of the major benefits of this approach is that it allows for more natural interfaces. [28] Unlike the glove-based approach, which required specialized hardware, this method can be implemented using commonly available devices such as smartphones, tablets, or webcams. Additionally, this method is not limited to hand gestures and can be used to detect other body movements as well as outlined by the article of Rui Ma et al. [12]. Furthermore, this approach is more cost-effective and can be integrated into a wide range of applications, such as gaming, virtual reality, and human-computer interaction.

Lastly, there is the depth-based approach which utilizes depth cameras. [23] These devices work differently from traditional cameras, as they do not capture regular images, but instead record the distance of the points in the scene with depth sensors (Figure 2.2). A well-known example of this type of device is the Microsoft Kinect. Initially developed for gaming, it has now found other uses beyond gaming, including hand gesture recognition. For example, researchers B. Ma and others have presented a system that uses the

Kinect to control a robot. [11]

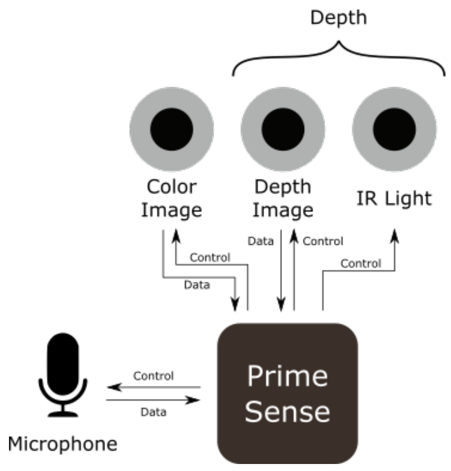

Figure 2.2: Microsoft Kinect architecture diagram.

Once input is received from the device, preprocessing is often carried out to improve the accuracy and performance of the system. [24] This step, while not strictly necessary, can include tasks such as resizing and cropping images, normalizing pixel values, and removing noise.

2.3 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are specifically good processing and analyzing images as inputs. This architecture makes them particularly well-suited for tasks such as image recognition, object detection, and image classification. [18]

CNNs are composed of multiple layers, each with a specific purpose. The convolutional layer, which gives the network its name, plays a key role by extracting features from the input image. It does this by applying a mathematical operation called convolution, in which a small matrix called a kernel or filter is moved over the input image, and the values of the kernel are multiplied with the corresponding values of the image. This process is repeated for different kernel positions and different kernels, resulting in multiple feature maps. By extracting features, the convolutional layer allows the CNN to learn local patterns such as edges, textures, and shapes, which are important for recognizing objects in the image.

The pooling layer follows the convolutional layers and is responsible for simplifying the output by down-sampling the feature maps produced by the convolutional layer. This results in a reduction of the complexity of the subsequent layers.

Lastly, the fully connected layer is utilized to learn non-linear decision boundaries by combining the features extracted in the previous layers, and thus producing the final output of the network. [1]

This stage of the pipeline is divided into two parts that must be combined in order to successfully determine the gesture being made. The first step involves using a hand detector, which filters out the image of the hand from the entire input image. This is a crucial step as it allows the system to focus on the hand and disregard the rest of the image, thus making it easier to identify and classify the gesture. The hand detector uses a combination of techniques such as image processing, machine learning and computer vision to detect the presence of a hand in the image, and to extract features such as the position and shape of the hand. [26]

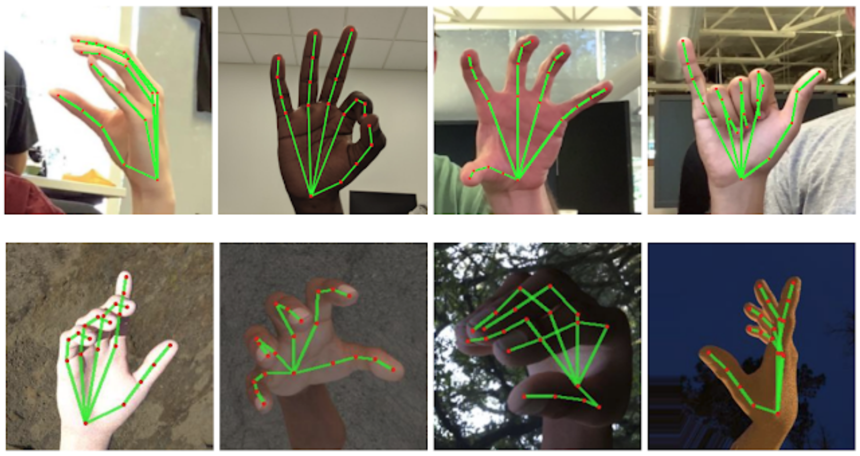
Figure 2.3: Example of detected hands. (Source: mediapipe)

Once the hand is detected, the system proceeds to the next step of identifying the gesture being made. This is accomplished by using a convolutional neural network (CNN) to analyze the hand image. [26]

Despite their ability to produce good results in the field, CNNs must be well thought out and properly trained in order to achieve efficient and accurate results. For example, research by H. Y. Chung et al. [4] demonstrated that through the training, they were able to achieve recognition rates of up to 95.61% for 6 different gestures. In contrast, one of the models only achieved a recognition rate of almost 85% on the test set. This illustrates how the training process and the design of the CNN architecture can significantly impact the performance of hand gesture recognition.

2.4 Gesture mapping

In Human-Computer Interaction (HCI) by hand gesture recognition, the selection of hand gestures is a crucial aspect in designing an effective gesture vocabulary. [22] The unique characteristics and the range of motion of hand gestures must be taken into account when making this selection since the goal of HCI is to create an interface that allows users to control computer tasks through a set of commands in the form of hand gestures. [3] This requires a thorough understanding of the potential gestures available and the ability to select those that are most intuitive and easy to understand. It's important to conduct some user testing and gather feedback in order to optimize the gesture vocabulary and improve the overall usability of the interface. [15] This step allows us to evaluate the gesture vocabulary and make sure it is easy to understand and perform, and that it will not confuse the user.

All in all, the selection of vocabulary (in this case hand gestures) is an essential aspect of HCI, as it provides the foundation for a natural and intuitive interface. [16] A well-designed gesture vocabulary can lead to greater efficiency and satisfaction for the user, and therefore, it is a critical step in the pipeline for the development of a successful HCI system.

2.5 Conclusions

In summary, recognizing hand gestures in the field of human-computer interaction (HCI) is a challenging task.

In this chapter, a general pipeline for an HCI interface utilizing hand gesture recognition was presented and the main approaches in the literature were reviewed. In the following chapter, we will propose a Proof of Concept (PoC) of this pipeline while making some simplifications. These simplifications may include the use of pre-trained CNN models and the use of a predefined dataset of hand gestures. Additionally, we will not be exploring beyond hand gestures in this PoC, thus excluding some topics such as hand pose estimation.

Chapter 3

Poject proposal

As previously discussed, we will create a proof of concept for hand gesture recognition as a means of controlling racing games. This section will delve into the design and method used to create a demonstration, as well as the strategy chosen to create an intuitive and natural human-computer interaction.

We will follow the pipeline flow while explaining the proposed strategy for the proof of concept. Additionally, we will outline the requirements and simplifications applied to the demo.

3.1 Requirements

This Proof of Concept will have specific requirements that will determine its functionality. The first requirement will focus on usability, ensuring that the system is intuitive and easy for users to understand. Additionally, the interface must be flexible, allowing for customization of gestures. The system must also be robust, able to accurately recognize gestures regardless of variations in hand size or skin tone.

In terms of results, the Proof of Concept must demonstrate its ability to meet these requirements and provide accurate and reliable gesture recognition. Additionally, the user experience must be evaluated to ensure that it is meeting the needs of its intended audience, this means that the system should be easy to adapt to as well as easy to learn. The final outcome of this proof of concept should be a functional system that is user-friendly, accurate and robust in recognizing gestures with flexibility in customization.

The interface should also include gestures that are easy to transition between, particularly in fast-paced situations like racing games. For instance, taking the “okay” and “like” gestures depicted in Figure 3.1, transitioning from an “ok” sign to a “like” sign quickly may not be easy. Additionally, the gestures should be intuitive and make sense for the actions they represent as it doesn't make sense, for example, to map the turning left action to a “like” gesture while using a “one” gesture to turn right, since using gestures that do not have opposite meanings for opposite actions will confuse the user.

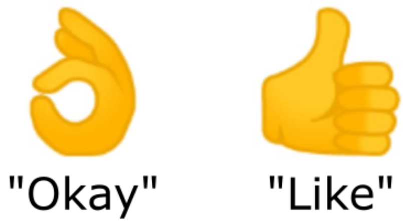

Figure 3.1: Visual example with emojis of the “like” and “okay” gestures.

3.2 Image Acquisition

In this study, the vision-based approach was chosen for hand gesture recognition as it offers a cost-effective and practical solution. Additionally, this approach has been widely researched and has been shown to be effective in a variety of applications, making it a suitable option.

Figure 3.2: Image of a Trust GXT 1160 webcam.

To further illustrate the accessibility of this method, a Trust GXT 1160 webcam (Figure 3.2) was selected as the camera input source. This was a deliberate choice as it is not a high-end, expensive camera. By using a less costly camera, it is highlighted that the vision-based approach to hand gesture recognition can be used by people with a range of resources and budget. It also serves as a demonstration that the system can function effectively with a relatively low-cost camera, making it more accessible to a wider audience.

Meanwhile, in the pre-processing step, we will be taking the raw input image and preparing it for the machine learning model by performing a series of operations on it. One of the key operations is to resize the image. This is important because the machine learning model expects the images to be of a certain size and resizing it to that size will help the model process the image more efficiently.

Another operation we may need to do is to convert the color space of the image. The color space of an image refers to the way the colors are represented in the image and different models may expect different color spaces. We might need to convert the image to a specific color space so it meets the model requirements. This will ensure that the model is able to process the image correctly and extract relevant features from it.

All these operations are important as they help the model to extract relevant features from the image, and ultimately, help to improve the model's performance.

3.3 Convolutional Neural Network

In order to simplify the process and not lose focus of the goals, some simplifications are being added as the whole process to hand gesture recognition is very complex and time-consuming.

Properly training a CNN for hand gesture recognition can be a time-consuming task. One of the most time-consuming aspects is collecting and labeling a diverse and extensive data set of hand gestures. The use of an already existing data set was deemed appropriate as it can save a significant amount of time.

It's important to keep in mind that a high quality and diverse data set is crucial for the performance of the model, as variations in hand shape, skin tone and other factors can affect the classification of gestures. Without a robust data set, the model may not perform well on new, unseen data.

Another crucial step in the process of hand gesture recognition is training a CNN. This step can be quite time-consuming, and requires a high-quality dataset as well as a well-designed CNN architecture. Designing the right architecture for a system can be a complex process that often requires experimentation and adjustments to achieve optimal accuracy. It is not always clear what the initial architecture should be, so it is necessary to try different combinations of layers and fine-tune parameters such as kernel size, stride, and activation functions through trial and error until the desired results are achieved.

The training process itself can also be quite time-consuming, as it depends on several factors such as the size of the training dataset, the complexity of the CNN architecture, and the computational resources available. This process can take several hours or even days, depending on the number of images in the dataset, the complexity of the CNN architecture, and the computational resources available.

Given the length of time required for this step, it was determined to be more practical to use a pre-trained model. This approach can save time and computational resources while still providing good results. Using a pre-trained model means to use a model that has been previously trained on a large and diverse dataset, which can be helpful in situations where it is not possible or practical to train a model from scratch.

For this study, the HaGRID dataset was selected. [7] This dataset is substantial, containing over 500,000 samples and 18 labeled classes as seen in Figure 3.3. Additionally, it provided a variety of pre-trained models with high accuracy that can be used.

By utilizing the HaGRID dataset and the provided pre-trained models, the time and resources required to achieve good results in hand gesture recognition can be minimized. This approach is expected to result in a robust system that meets the requirements for accurate hand gesture recognition as well as save time.

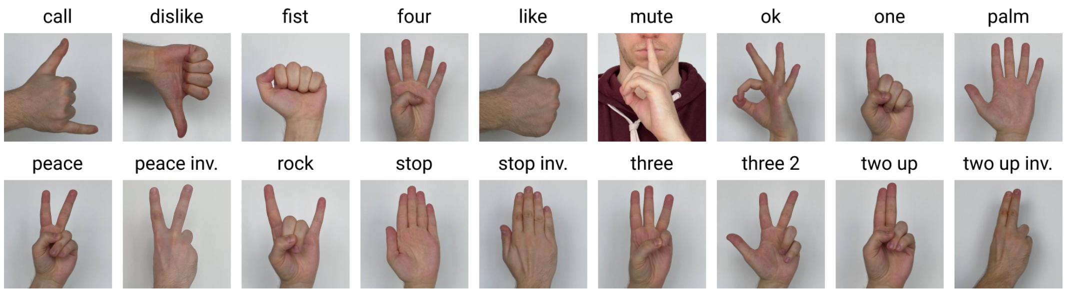

Figure 3.3: Image all the gestures available in the HaGRID dataset.

3.4 Gesture Mapping

For this part of the proposal we have to take into account the data set we choose, because it will define what gestures we have available to map. As seen in Figure 3.3 we can map up to 18 gestures, and we can play around with the lack of gesture also. In the first place, we need to think about what actions we have to perform in order to play racing games. The initial thought and the most obvious controls anyone would think of are: Turning (either left or right), Breaking and Accelerating.

In order to have an intuitive and natural interface, we first need to think of a combination of gestures that might be similar to real-life actions and are easy to learn and understand. However, we should also take into account the effort that is needed in order to play the game using the interface as having a set of gestures that are tiresome will be detrimental to the utility of the system and the performance of the player and thus the ability of the system to replace traditional controllers will be affected.

In the chosen data set we only have static gestures so initially we can't rely on real-life-like actions as in day to day driving no one uses static gestures. However, this is easily solvable. For example, thanks to the hand detector in the CNN stage (Section 2.3) we can track the positions of the hands in the image and thus calculate the angle formed by both hands and use it to determine the amount and direction of turning.

For this proposal, in order to explore multiple approaches on hand gesture mapping, we will propose 3 different mapping models that will help understand the different capabilities of the gestures.

1. This first version will try to use only static gestures. This approach will lead to an accessible interface as the movement of the hands is not necessary outside transitioning from one gesture to another. However, both hands will be needed for playing the game since in racing games you can, and most likely will, turn while accelerating or braking.
2. For the second model, we will shift to a less accessible but way more intuitive approach. In this approach we intend to use the solution proposed earlier in which we use the angles formed by both hands to calculate how much the car turns. With this we pretend to emulate a car turning wheel to turn. However we will still need 2 gestures for breaking and accelerating.
3. The third and last approach will provide a middle ground, where the interface will be not as intuitive as the second but in return we'll get more accessibility. The proposition is to take the angle generated by one hand to determine the amount of turning, this means that, by turning the hand like depicted in Figure 3.4, we can turn the car. With this we can keep an intuitive interface making it more accessible as we just change the orientation of the hand instead of moving around both of them.

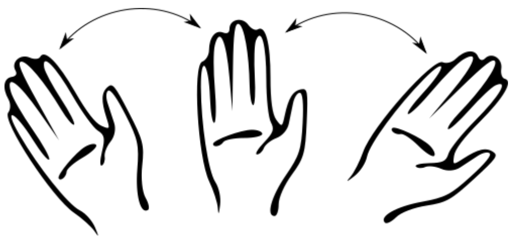
Figure 3.4: Visual example of how the hand can be turned to control the car turning.

These 3 approaches will help us understand the strengths and weaknesses of different hand gesture mapping models, and will also provide a basis for comparison to determine which approach is the most effective for this specific application of hand gestures in racing games. Additionally, by evaluating the different models, we will be able to identify which aspects of the interface are important for the users, such as ease of use, precision, and intuitiveness. Ultimately, this proposal aims to provide a comprehensive understanding of how hand gestures can be used in racing games to enhance the player’s experience and improve the overall usability of the interface.

Chapter 4

Coding project

To develop our proof of concept, we have chosen Python due to its many advantages. One key advantage is the large and active community of developers, which provides access to a wide range of libraries such as TensorFlow and PyTorch that are commonly used for CNN development.

Additionally, Python is a popular choice in the field of machine learning and deep learning, providing ample resources for learning and development. Furthermore, its simple syntax helps to focus on the development of the proof of concept without getting bogged down by the complexities of the language.

Overall, Python's community support, wide range of libraries, and simplicity make it a great choice for developing our proof of concept. We are confident that using Python will allow us to quickly and efficiently develop our interface proof of concept.

4.1 Capturing and Preprocessing Images

In order to begin capturing camera input we will use OpenCV, a popular Python library in the field of computer vision. It provides a wide range of functions for processing images, such as image transformation and video capture. Additionally, it is compatible with various deep learning frameworks, including TorchVision, enabling developers to utilize pre-trained models in their applications. The Python community offers ample support for OpenCV with many resources, tutorials and guides available. One such resource that is easy to understand and follow are the tutorials provided by the YouTube channel "Tech With Tim". [1]

Before inputting the images into the model, we will perform preprocessing steps to ensure that the images are suitable for the model. One such step is resizing the images to meet the requirements of the model. This can be easily accomplished using the OpenCV library. Additionally, we will use the TorchVision library to convert the image type (PIL

[1] The tutorials made by the channel "Tech With Tim" can be found in this youtube playlist.

object) to a tensor, which is a format that the model can work with. This step allows the model to effectively utilize the input images and make the whole process more efficient by reducing the computational cost.

4.2 CNN and data set

Once the image has been pre-processed and is in a format that the model can understand, it can be fed into the CNN model. In this case, we will use one of the pre-trained HaGRID models, which includes a hand detection task. This means that we don't need to use multiple models and can proceed directly to identifying the gestures.

However, before we can use the model, we need to load it and set a threshold value (ranging from 0 to 1) to determine the level of confidence required for the algorithm to recognize a gesture. For example, the detector returns two values indicating the level of confidence that a hand is doing one of the gestures. If the result from the detector is [0.0419, 0.0329] and we set the threshold to 0.5, the detector will indicate that there are no hands on the screen and therefore no gestures will be detected. But if either of the scores is greater than the confidence threshold, the detector will indicate that there is a hand and its gesture.

It's important to be mindful of the threshold value as setting it too low may result in false positive detections, and setting it too high may result in false negative detections.

4.3 Mapping hand gestures

For this project, we have chosen to map hand gestures to an emulated or virtual controller in order to achieve compatibility with a wide range of games and to gain precision by using the full range of motion of a controller inputs. To achieve this we are using the "vgamepad" python library which is a simple way to emulate a controller.

For the PoC, as outlined in section 3.4, we will explore three different approaches. The first approach is to use static gestures, which means that one gesture corresponds to one in-game action. While this approach increases accessibility by requiring less arm movement, it sacrifices intuitiveness. Furthermore, it does not take advantage of the full potential of mapping gestures to a controller. For example, the"one" gesture can be used to turn the car, but it does not allow for fine-tuned control over the joystick of the virtual controller.

With our second approach, we traded accessibility with intuitivity. The proposal for this approach is to use the hand position to emulate a turning wheel, which will provide greater precision when turning. However, braking and accelerating will still be mapped to static gestures.

The third approach aims to find a compromise between accessibility and intuitiveness by using one hand to turn while maintaining precision by rotating the hand. This approach allows for more control and precision than the first approach, while still being more accessible than the second approach.

Upon examining the available gestures in the data set (Figure 3.3), it was decided to primarily use the “palm” gesture for acceleration and the “stop” gesture for braking. This selection was not made arbitrarily, but rather based on a few key factors. Firstly, transitioning between these two gestures is quick and easy as it simply involves bringing the fingers together or separating them. Additionally, these gestures are intuitive and easy to understand, as the “stop” gesture is commonly used to indicate stopping in our everyday language. Furthermore, making the “stop” gesture requires tensing the hand, which can be related to the action of braking in a car, as applying pressure on the brake pedal usually requires tensing the foot.

Furthermore, for the first approach, the “like” and “dislike” gestures were selected for turning as they represent opposite concepts like “left” and “right”.

However, in order to evaluate the ease and comfort of the interface, as well as the effort required to use it, these gestures will not be fixed but will serve as a reference point.

4.4 Conclusions

The project followed the pipeline outlined in Figure 2.1 and we can now assess if some of the requirements were met. At first glance, some of the proposals of the interface look intuitive and easy to understand thanks to the gestures chosen. However, we have limited control over the robustness of the CNN as we are using a pre-trained model and neither did we create our own data set. Despite this, we can ensure that there will be no issues with hand size or skin tone due to the large data set used.

Chapter 5

Testing & Results

In this section, we will explain how we're testing the interfaces as well as exploring the results obtained. The test was designed to obtain data about intuitively, ease of use and the ease to learn the interfaces. In order to accomplish it, the test was done by 10 people all with different demographics in order to test the different requirements from various points of view since this kind of solution can be directed to a lot of different groups of people. Additionally, we'll be able to test the robustness of the pre-trained model.

5.1 Testing

We decided to test the 3 different approaches to replacing traditional controllers in racing games by playing Assetto Corsa[1]. The game's large modding community provides plenty of options for us to select a track that is appropriate for the test. The track selected was the "Karting Cardedeu" circuit[2] as it provides a good mix of hard braking zones and fast corners, as seen in the track layout from Figure 5.1. This allows for testing a variety of gesture combinations in-game. Additionally, the circuit is not too long, allowing users to focus on learning the interface rather than the circuit layout.

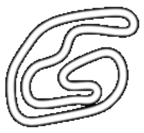

Figure 5.1: Track layout of the selected circuit.

The testers will be required to try the 3 interfaces, as seen in Figure 5.2 as well as a classic form of input, such as a traditional controller, a sim racing wheel if they have access to one or both. This will enable us to compare the results not only between the 3 interfaces but also to contrast the proposed interfaces with the traditional forms of input.

[1] Assetto Corsa is a racing simulation game with a huge modding community.
[2] This specific mod created by the user "carreras" of can be found here.

Additionally, after testing each interface, we will ask the testers if they would like to change the gestures used in case they felt there could be a more intuitive combination of gestures.

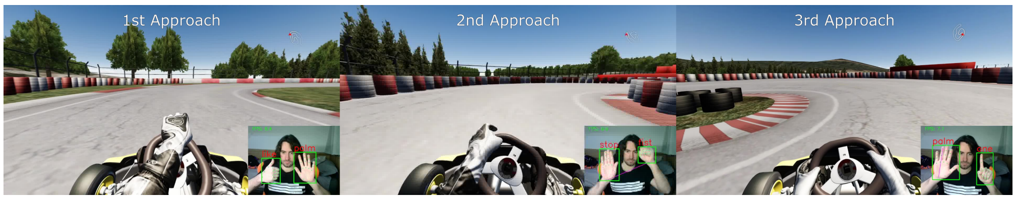

Figure 5.2: Examples of the 1st, 2nd, and 3rd approach.

The testers will be asked to run 10 complete laps around the circuit to evaluate the users'adaptability to the interface and the effort required and overall overall competitiveness of the interface.

For this test we selected 9 individuals with varying ages and levels of video game knowledge for the study. These participants represented a diverse demographic, including both genders and various age ranges. The program was tested on participants over the age of 18. Out of the 9 individuals, 6 were men and 3 were women. The group of men included 3 participants between the ages of 20 and 30, 2 between the ages of 30 and 40, and 1 between the ages of 40 and 50. The men between the ages of 20 and 30 had extensive experience with video games, particularly racing games. Conversely, the men over 30 had general experience with video games but not specifically with racing games. As for the women, one had no prior experience with video games, while the other two had played games occasionally.

5.2 Results

Before delving into the results, it's important to note that some limitations were encountered during the development of the Proof of Concept (PoC). The model struggled to recognize gestures when the hand was rotated. This should be considered when mapping the gestures for the second and third approach, as their main feature involves controlling steering by rotating the hands. Additionally, during some of the tests, some players were unable to complete the 10 laps, which was unexpected.

5.2.1 Baseline: Sim racing wheel and Controller

Table 5.1 shows that among 5 players who had access to a sim racing wheel, initial lap times range from 0:43.542 to 0:51.283. Figure 5.3 also demonstrates that improvements for the initially faster drivers are minimal, with those starting below 44 seconds improving by less than one second.

Tester	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7	Lap 8	Lap 9	Lap 10
1	0:43.542	0:44.139	0:43.232	0:44.082	0:43.121	**0:43.027**	0:43.979	0:43.699	0:44.075	0:43.199
2	0:48.950	0:47.197	0:47.502	0:48.823	0:46.895	0:47.791	0:48.783	**0:46.836**	0:46.897	0:46.867
3	0:51.283	0:49.325	0:48.463	0:47.124	0:47.093	0:46.157	**0:45.085**	0:45.864	0:45.517	0:46.637
4	0:43.709	0:43.624	0:43.708	**0:43.617**	0:43.628	0:43.706	0:43.715	0:44.028	0:43.730	0:44.072
5	0:47.759	0:45.602	0:46.497	0:44.921	0:45.439	0:44.029	0:43.600	0:44.290	0:44.110	**0:43.862**

Table 5.1: Lap times done with sim racing wheel.

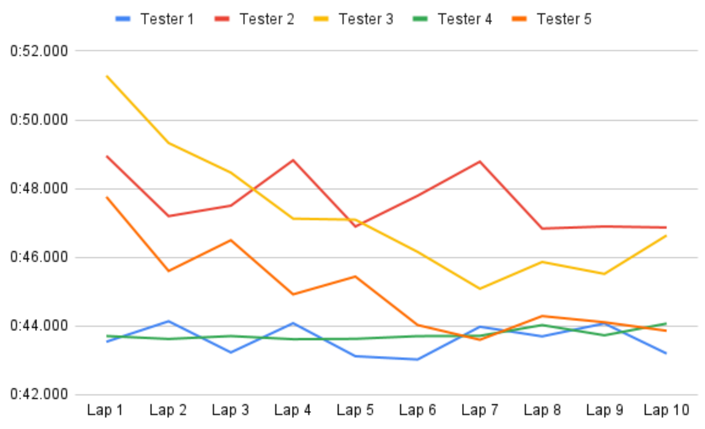

Figure 5.3: Line chart of the times with sim racing wheel.

The chart (Figure 5.3) also reveals that most drivers tend to improve, with some showing improvements of up to 4 seconds, which is noteworthy considering the difficulty of improving one's time as it gets better.

Tester	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7	Lap 8	Lap 9	Lap 10
1	0:44.955	0:44.657	0:45.352	0:43.989	0:45.187	**0:43.826**	0:43.955	0:45.297	0:43.975	0:45.476
2	0:51.364	0:50.723	0:49.475	0:49.462	0:50.813	0:49.875	0:49.765	0:49.752	**0:48.103**	0:50.532
3	0:55.704	0:51.010	0:49.508	**0:48.964**	0:55.417	0:50.245	0:49.937	0:54.314	0:53.105	0:49.176
4	0:44.663	0:44.633	0:44.826	0:44.910	0:44.764	0:43.810	**0:43.777**	0:44.874	0:44.893	0:44.368
5	1:05.243	1:03.284	1:00.347	0:59.101	1:02.320	0:57.778	1:04.603	1:00.164	0:58.513	**0:53.983**
6	1:46.289	1:31.394	1:44.931	1:42.585	1:40.245	1:39.367	1:35.939	1:36.164	**1:26.442**	1:27.693
7	1:10.888	1:05.516	1:04.927	1:08.451	1:04.937	**0:57.184**	1:01.381	0:58.443	0:59.646	1:00.105
8	0:46.151	0:44.634	0:44.982	0:43.468	0:45.344	**0:43.329**	0:43.903	0:44.260	0:43.974	0:43.476
9	1:03.935	1:02.350	0:57.630	1:00.379	0:59.567	0:57.556	0:56.889	**0:53.621**	0:55.868	1:02.729

Table 5.2: Lap times done with controller.

On the other hand, Table 5.2 shows lap times up to 1:46.289 for testers who have never held a controller before. Similarly, this group also demonstrates the same improvement pattern as before, where the initially faster testers have difficulty improving while the slower ones have an easier time. Additionally, a quick observation reveals that there are three drivers who are notably faster than the rest, providing an opportunity to compare the effectiveness of different interfaces.

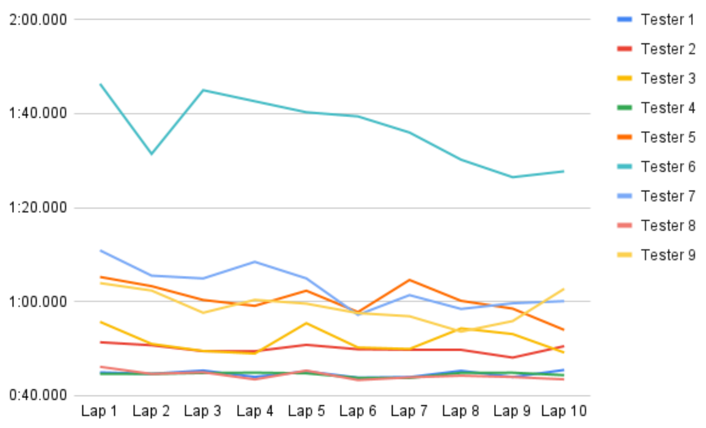

Figure 5.4: Line chart of the times with controller.

Furthermore, from the chart in Figure 5.4 we can see how even though some testers have a clear and consistent improvement overall, others don't. Thanks to this, we'll be able to see how our models compare in terms of learnability to the traditional controller.

5.2.2 1st Approach

During the test of the first approach, unexpected results were observed, as some testers were unable to complete the 10 laps.

Tester	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7	Lap 8	Lap 9	Lap 10
1	1:06.141	1:03.489	1:10.502	1:07.263	1:04.314	1:02.562	1:00.020	**0:55.755**	1:03.896	0:58.377
2	2:01.314	1:58.093	1:49.396	1:48.164	1:46.869	1:46.869	1:34.948	**1:31.177**	1:33.843	1:37.928
3	2:09.822	2:11.988	2:03.690	1:55.614	1:54.535	**1:46.209**	1:57.708	1:52.718	1:49.708	1:50.159
4	1:27.856	1:32.873	1:25.090	1:20.206	1:30.798	1:22.069	1:27.563	1:19.075	1:28.275	**1:18.050**
5	3:24.661	3:33.314	3:18.986	**3:09.137**	3:33.866	3:28.950	3:27.995	—	—	—
6	3:46.088	3:41.530	**3:26.599**	—	—	—	—	—	—	—
7	3:16.244	3:20.598	3:19.369	**3:04.888**	3:19.818	—	—	—	—	—
8	1:36.188	1:42.099	1:17.022	1:47.788	1:29.837	**1:15.396**	1:30.141	1:19.799	1:28.522	1:34.723
9	3:06.617	3:25.485	**2:32.487**	3:19.347	2:59.377	3:01.967	2:40.074	2:40.071	3:25.653	2:38.379

Table 5.3: lap times done with the 1st approach.

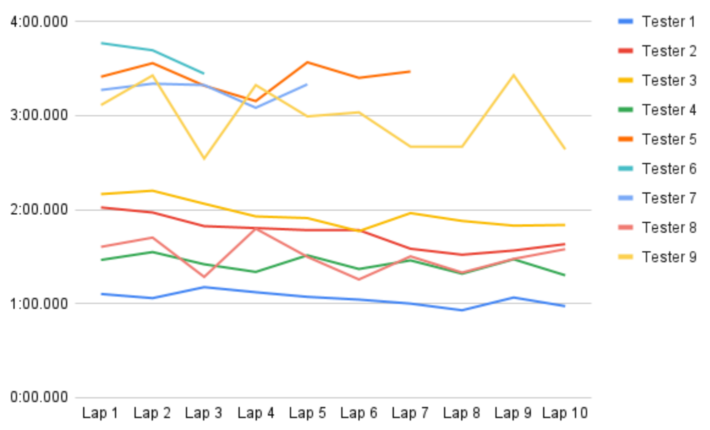

Figure 5.5: Line chart of the times with the 1st approach.

Figure 5.5 illustrates that players who performed best in the baseline continue to have the best results with this interface, however, they are slower and inconsistent.

All players reported that this interface was physically exhausting as it required them to keep their hands in front of the camera at all times. Additionally, the testers found the interface not intuitive due to the turning gestures rather than the approach itself. They found the breaking and accelerating gestures to be fine but felt that the turning gestures were not clear enough. Despite this, they were unable to find better gestures as they were missing gestures that clearly indicate left or right turns.

5.2.3 2nd Approach

Tester	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7	Lap 8	Lap 9	Lap 10
1	0:51.761	0:49.916	0:51.270	0:50.632	0:48.685	0:50.200	0:50.147	0:49.495	**0:48.204**	0:49.004
2	1:20.664	1:23.059	1:15.644	1:22.420	1:20.246	1:16.460	1:15.253	1:11.598	1:14.753	**1:10.965**
3	1:32.649	1:27.190	1:29.970	1:27.911	1:28.210	1:25.013	**1:22.841**	1:23.085	1:24.690	1:23.124
4	0:55.635	0:56.635	0:55.210	0:55.925	0:55.019	0:54.963	0:53.596	0:55.405	**0:53.532**	0:54.514
5	2:55.381	2:46.527	2:48.582	2:37.733	2:40.117	**2:37.396**	2:52.476	2:49.940	2:50.155	2:41.214
6	3:28.007	3:13.681	3:16.758	3:06.738	3:14.850	3:12.287	3:09.385	3:07.345	3:06.568	**3:05.358**
7	2:47.210	2:43.208	2:41.296	2:48.991	2:41.893	2:36.588	**2:32.056**	2:40.279	2:36.751	2:35.091
8	0:56.197	0:57.245	0:56.276	0:57.911	0:55.467	0:53.700	0:52.443	0:51.628	**0:50.949**	0:52.445
9	2:20.350	2:19.928	2:23.278	2:17.623	2:17.911	**2:06.395**	2:11.802	2:12.248	2:10.829	2:08.455

Table 5.4: Lap times done with the 2nd approach.

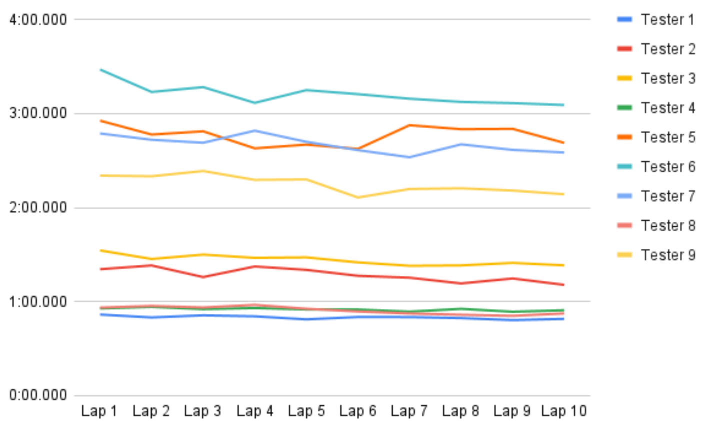

Figure 5.6: Line chart of the times with the 2nd approach.

Figure 5.6 allows for comparison between the first two approaches. A clear improvement in consistency is observed among testers, but the three fastest drivers still perform worse than with a traditional controller. However, it's worth noting that this interface is still better than the first approach, with some drivers achieving times under 50 seconds.

The testers felt more natural and confident using this interface, with testers 1 and 4 attributing most of their improvement to increased turning precision. However, the majority of players still found this option tiring due to the need to keep both arms in front of the camera.

5.2.4 3rd Approach

Tester	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7	Lap 8	Lap 9	Lap 10
1	0:50.197	0:48.989	0:48.823	0:49.392	0:46.969	0:47.829	0:46.551	**0:46.460**	0:46.840	0:47.320
2	1:01.588	0:58.403	0:56.912	0:57.217	0:55.759	0:56.105	0:58.461	0:55.233	0:54.239	**0:53.822**
3	1:02.076	1:01.059	0:58.400	0:59.350	**0:58.350**	0:58.906	0:58.535	0:59.163	0:58.782	0:58.369
4	0:51.567	0:50.346	0:49.934	0:50.102	0:48.873	0:47.947	**0:47.841**	0:48.381	0:49.091	0:48.716
5	1:19.023	1:15.974	1:21.646	**1:09.324**	1:12.013	1:10.415	1:10.052	1:14.426	1:09.605	1:11.370
6	2:53.743	2:49.535	2:36.690	2:45.745	2:43.668	2:36.528	2:37.195	2:26.954	**2:21.671**	2:27.440
7	1:26.740	1:31.480	1:32.388	1:29.383	1:18.103	1:19.335	1:22.335	**1:12.657**	1:14.460	1:16.987
8	0:50.696	0:50.061	0:49.353	0:50.398	0:48.390	0:47.088	0:48.109	0:47.626	0 :46.416	**0:45.740**
9	1:13.176	1:14.293	1:09.635	1:12.752	1:07.900	**1:04.924**	1:09.060	1:07.111	1:08.515	1:07.880

Table 5.5: Lap times done with the 3rd approach.

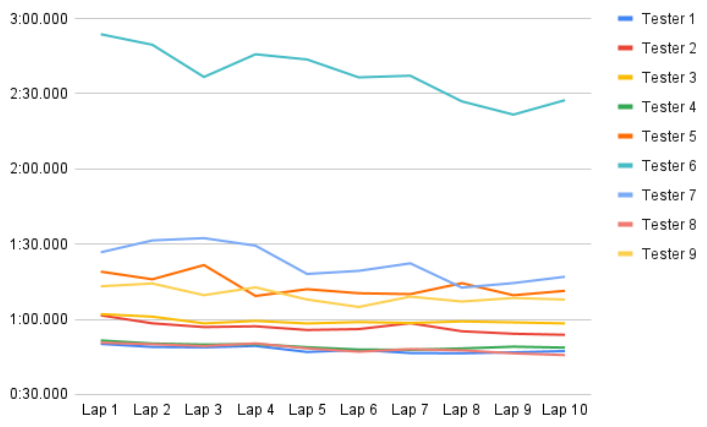

Figure 5.7: Line chart of the times with the 3rd approach.

Figure 5.7 shows a distribution similar to that of a traditional controller. Five out of the nine testers consistently achieved lap times under one minute, despite some starting with lap times over one minute.

This final approach was the preferred option among testers, who found that even if the rotation of the hand is not as intuitive as simulating you have a steering wheel, it reminds enough to the turning movement. Many players also suggested implementing a way to turn and accelerate using the same hand, in line with this approach's philosophy.

Additionally, while some testers reported slight strain on the wrist, they still found it more comfortable than the other approaches.

5.3 General outcomes

From these tests, we can now evaluate if we met our proposed requirements. First, we asked users to rate the comfort of the 3 interfaces on a scale of 1 to 10, with 1 being the worst and 10 being the best. The overall scores were 2.44, 4.88, and 7.44 for the first, second, and third approaches, respectively. As previously stated, most players felt tired or strained after playing for just 10 laps, but the third approach received the least complaints.

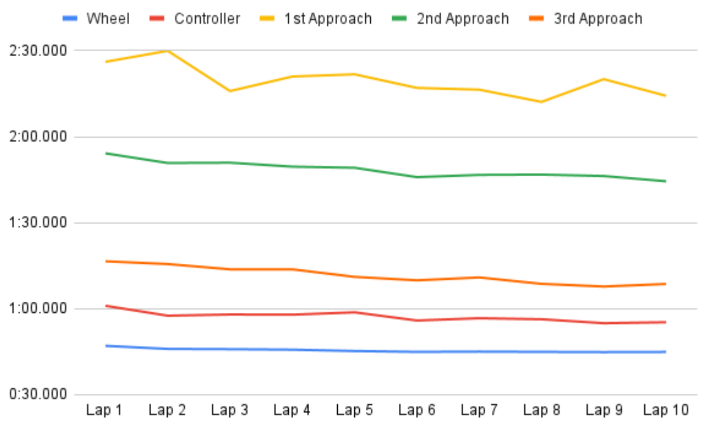

Figure 5.8: Line chart of the overall times of each lap.

Secondly, as seen in Figure 5.8, the first approach shows the most improvement, but it is also the most inconsistent. This deosn't mean that the users improved since they can’t keep running as fast as they once achieved. The third approach is more consistent but with less notable improvement. Additionally, the third approach shows an improvement of over 11% between the initial and best lap times, while the controller shows an improvement of almost 10%. This suggests that, even though the improvement of the third approach is higher, the learning rate of the third approach and the controller are similar, as it becomes harder to improve as one gets closer to the best time possible.

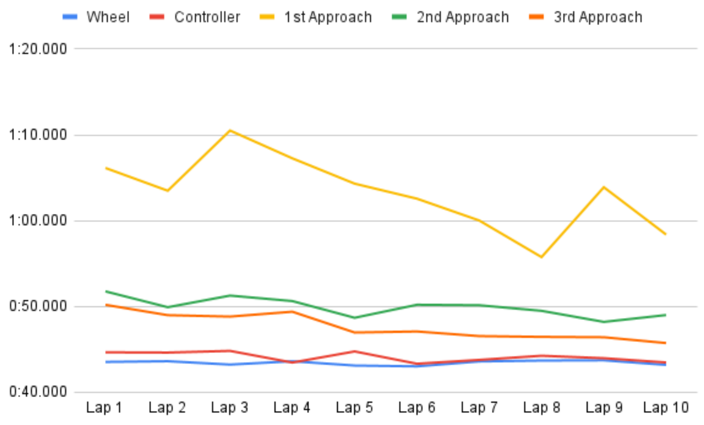

Figure 5.9: Line chart of the best times of each lap.

Looking at the best times for each lap (Figure 5.9), we can see that the first approach is not consistent, while the other two follow a smoother curve.

It’s also noticeable that the second and third approaches started closer than where they finished, while the opposite happened to the 3rd approach and the controller.

In terms of comfort and intuitivity, the controller was rated first by testers due to its minimal movements and lack of physical fatigue. Among the proposed approaches, the third approach was considered the best as it was more comfortable and had a noticeable improvement in performance.

From this results we can conclude that out of the 3 proposals, the third one produced the best results. However, even with its increased intuitivity, this PoC still couldn't overcome the comfort that the traditional controllers provide. Nevertheless, in terms of performance, even though the 3rd approach did not surpass traditional controllers, it has to be noted that our approach is less costly economically speaking. While the webcam used costed around 20€[3], an official Microsoft controller costs no less than 50€[4].

[3] Price of the used webcam in PcComponentes

[4] Price of an official Microsoft controller in PcComponentes

Chapter 6

Conclusions & Further investigation

6.1 Conclusions

The creation of an interface for playing racing games using vision-based hand gesture recognition is a broad field that requires the development of multiple techniques and strategies. However, by analyzing the key processes of hand gesture recognition and comparing them to previously established requirements, we have obtained a sufficient understanding of the subject.

All this data has been fundamental in order to propose a pipeline like the one seen in Figure 2.1 where a minimal flow for a game controller was described as the pipeline has been based on how generally hand gesture recognition is achieved and adapted to create a human-computer interface.However, during the pipeline's description, it was observed that most methods concentrate on hand gesture recognition and not human interaction.

With the pipeline in place, it became clear how a vision-based hand gesture recognition interface for playing games could be developed, which led to the creation of a proof of concept (PoC). This design was then tested by implementing the PoC, taking into account each step of the proposed pipeline such as acquiring input images, preprocessing the images, using a neural network, and mapping gestures.

The proof of concept (PoC) was designed to not only evaluate how well static gestures fare in racing games, but also to provide different models to differentiate the weaknesses and strengths of each. This allowed for a comprehensive evaluation of the hand gesture recognition interface.

The results of the PoC provided valuable insight into how various interfaces performed. The design of the PoC allowed for this evaluation to be done without having to modify the base code, as all the necessary variables to run the interfaces were the default set ones.

This streamlined the testing process and made it easier to compare the performance of different models. Overall, the PoC was a valuable tool in assessing the potential of hand gesture recognition interfaces in gaming and identifying areas for future development.

6.2 Future work

Despite the positive results that were obtained from the proof of concept (PoC), there are still areas for improvement. One of the most significant improvements that can be made is to create a convolutional neural network (CNN) from scratch, either using the proposed data set or creating a new one. This would allow for greater customization as the best gestures could be selected in advance and the CNN could be specifically designed for the dataset.

Another area for improvement in the PoC is in the third approach to the interface, where testers suggested the ability to control the car (braking, accelerating, and turning) with the same hand. This would reduce fatigue on the users as the posture taken to play could be more comfortable and the second hand would be freed. However, due to the limitations of the model that was selected, this could only be achieved by creating a new model from scratch, as mentioned earlier.

In addition, to further broaden the investigation, it would be beneficial to research other camera point of views as most of the proposals use the same point of view. This would provide a more comprehensive understanding of the potential of hand gesture recognition interfaces in gaming and allow for the identification of new areas for development.

Appendix A

Technical manual

A.1 Software versions

To develop this application Python 3.9.4 was used on Windows 10 for 64 bits. To download the code you can navigate to the release and download it. This is the version released with the thesis.

A.2 How to install

Once you dowloaded the code from the release unzip it in your prefered location. Once unzipped, open a terminal and navigate to where the root folder of the project. With the terminal, in the root directory of the project, install the requirements with this command line:

```
pip install -r requirements.txt
```

A.3 How to run

To run the project use the following command line:

```
python main.py -v {1, 2 or 3 to choose the approach}
```

or

```
py main.py -v {1, 2 or 3 to choose the approach}
```

A.4 Configuration

1. **Confidence threshold**: In order to change the threshold of confidence of the CNN head to the “constants.py” file located in the root and modify the “threshold” variable

2. **Gesture mappings**: To change the gesture mappings of the 1st and 2nd approach head to the “constants.py” file located in the root and modify the values of the “gestures” dictionary.

   To change the 3rd approach mappings, change the “mappings” dictionary keys in the “v3.py” file.

3. **Angle needed to turn**: To change the maximum angle to turn in the 2nd and 3rd approach head to either “v2.py” or “v3.py” depending on the one you want to change. Changing “v2.py” will change the 2nd approach angle while changing “v3.py” will affect to the 3rd approach angle. Modify the “turning_angle” variable. 30 means that generating an angle of 30 degrees you will get the maximum turning input.

A.5 Recommendations

These are a couple of recommendations for using this PoC. First, stay a bit away of the camera. If you are close is more probable that by moving the hand some part will be out of the frame, and the gesture will not be recognized.

Test that the gestures you selected before playing seriously. Might happen that because of a complex background some gestures are not detected properly. For example, “palm” and “stop” worked fine for all cases but the “fist” gesture had problems depending on the background.

Bibliography

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi, *Understanding of a convolutional neural network*, 2017 international conference on engineering and technology (ICET), IEEE, 2017, pp. 1–6.

[2] M. H. Amir, A. Quek, N. R. B. Sulaiman, and J. See, *Duke: enhancing virtual reality based fps game with full-body interactions*, Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology, 2016, pp. 1–6.

[3] L. Chen, F. Wang, H. Deng, and K. Ji, *A survey on hand gesture recognition*, 2013 International conference on computer sciences and applications, IEEE, 2013, pp. 313–316.

[4] H. Y. Chung, Y. L. Chung, and W. F. Tsai, *An efficient hand gesture recognition system based on deep cnn*, 2019 IEEE International Conference on Industrial Technology (ICIT), IEEE, 2019, pp. 853–858.

[5] J. Galván-Ruiz, C. M. Travieso-González, A. Tejera-Fettmilch, A. Pinan-Roescher, L. Esteban-Hernández, and L. Domínguez-Quintana, *Perspective and evolution of gesture recognition for sign language: A review*, Sensors **20** (2020), no. 12, 3571.

[6] J. Han, L. Shao, D. Xu, and J. Shotton, *Enhanced computer vision with microsoft kinect sensor: A review*, IEEE transactions on cybernetics **43** (2013), no. 5, 1318–1334.

[7] A. Kapitanov, A. Makhlyarchuk, and K. Kvanchiani, *Hagrid-hand gesture recognition image dataset*, arXiv preprint arXiv:2206.08219 (2022).

[8] N. V. Le, M. Qarmout, Y. Zhang, H. Zhou, and C. Yang, *Hand gesture recognition system for games*, 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), IEEE, 2021, pp. 1–6.

[9] J. Liu and M. Kavakli, *A survey of speech-hand gesture recognition for the development of multimodal interfaces in computer games*, 2010 IEEE International Conference on Multimedia and Expo, IEEE, 2010, pp. 1564–1569.

[10] M. A. Livingston, J. Sebastian, Z. Ai, and J. W. Decker, *Performance measurements for the microsoft kinect skeleton*, 2012 IEEE Virtual Reality Workshops (VRW), IEEE, 2012, pp. 119–120.

[11] B. Ma, W. Xu, and S. Wang, *A robot control system based on gesture recognition using kinect*, TELKOMNIKA Indonesian Journal of Electrical Engineering **11** (2013), no. 5, 2605–2611.

[12] R. Ma, Z. Zhang, and E. Chen, *Human motion gesture recognition based on computer vision*, Complexity **2021** (2021).

[13] A. Mujahid, M. J. Awan, A. Yasin, M. A. Mohammed, R. Damaševičius, R. Maskeliūnas, and K. H. Abdulkareem, *Real-time hand gesture recognition based on deep learning yolov3 model*, Applied Sciences **11** (2021), no. 9, 4164.

[14] G. R. S. Murthy and R. S. Jadon, *A review of vision based hand gestures recognition*, International Journal of Information Technology and Knowledge Management **2** (2009), no. 2, 405–410.

[15] M. Niranjanamurthy, A. Nagaraj, H. Gattu, and P. K. Shetty, *Research study on importance of usability testing/user experience (ux) testing*, International Journal of Computer Science and Mobile Computing, IJCSMC **3** (2014), no. 10, 78–85.

[16] K. Norman and J. Kirakowski, *The wiley handbook of human computer interaction set*, John Wiley & Sons, 2017.

[17] E. Ohn-Bar and M. M. Trivedi, *Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations*, IEEE transactions on intelligent transportation systems **15** (2014), no. 6, 2368–2377.

[18] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, arXiv preprint arXiv:1511.08458 (2015).

[19] M. Oudah, A. Al-Naji, and J. Chahl, *Hand gesture recognition based on computer vision: a review of techniques*, journal of Imaging **6** (2020), no. 8, 73.

[20] S. S. Rautaray and A. Agrawal, *Vision based hand gesture recognition for human computer interaction: a survey*, Artificial intelligence review **43** (2015), no. 1, 1–54.

[21] C. Saha, D. Goswami, S. Saha, A. Konar, A. Lekova, and A. K. Nagar, *A novel gesture driven fuzzy interface system for car racing game*, 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2015, pp. 1–8.

[22] J. J. Stephan and Sana'a K., *Gesture recognition for human-computer interaction (hci).*, Int. J. Adv. Comp. Techn. **2** (2010), no. 4, 30–35.

[23] J. Suarez and R. R. Murphy, *Hand gesture recognition with depth images: A review*, 2012 IEEE RO-MAN: the 21st IEEE international symposium on robot and human interactive communication, IEEE, 2012, pp. 411–417.

[24] X. Y. Wu, *A hand gesture recognition algorithm based on dc-cnn*, Multimedia Tools and Applications **79** (2020), no. 13, 9193–9205.

[25] D. Xu, *A neural network approach for hand gesture recognition in virtual reality driving training system of spg*, 18th International Conference on Pattern Recognition (ICPR'06), vol. 3, IEEE, 2006, pp. 519–522.

[26] P. Xu, *A real-time hand gesture recognition and human-computer interaction system*, arXiv preprint arXiv:1704.07296 (2017).

[27] M. Yasen and S. Jusoh, *A systematic review on hand gesture recognition techniques, challenges and applications*, PeerJ Computer Science **5** (2019), e218.

[28] Y. Zhu, Z. Yang, and B Yuan, *Vision based hand gesture recognition*, 2013 International Conference on Service Sciences (ICSS), IEEE, 2013, pp. 260–265.