GameTrack: A Hand Tracker for People with Movement Conditions to Play Video Games

John H. R. Steward

MComp Computer Science with Placement
The University of Bath
2024-2025

GameTrack: A Hand Tracker for People with Movement Conditions to Play Video Games

Submitted by John H R Steward

Copyright

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Masters of Computing in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Abstract

Video games are a massive global industry that everyone should be able to enjoy. Companies attempt to make accessibility options for their consoles but fall short in some aspects like cost, and usability for certain people. We have set out to create a hand tracker that can be used by people with movement conditions such as Parkinson's Disease to play video games more effectively, since it is difficult for them to control precise actions such as pressing buttons on a controller or keyboard. Using MediaPipe Hands[24], we have created a tracker that exclusively registers large, jerky motions as inputs, which we call GameTrack, and have translated those inputs into our game of choice: Sonic the Hedgehog 2. In practice, our participants were mostly able to improve their play when using GameTrack over a 5 minute period, most starting to get used to the new style of play, and making more accurate inputs towards the end of the study. This is a promising result that, while needing further testing, shows that any people can learn to use GameTrack effectively, and can enjoy their favourite games in a new, more immersive way.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, Professor Peter Hall, whose guidance and support were invaluable to the completion of this project. I would also like to thank the rest of my supervisor group for their insight and fruitful conversations within our meetings. I would also like to thank Lani Widdeson for modelling for screenshots, and supporting me throughout this project, and I would like to thank Louis Sentinella for taking screenshots, when I had no hands free. I would finally like to thank all of my participants, as their contribution made the completion of this project possible.

Figures and diagrams in this paper were made using MatPlotLib[1] and draw.io[2].

Chapter 1

Introduction

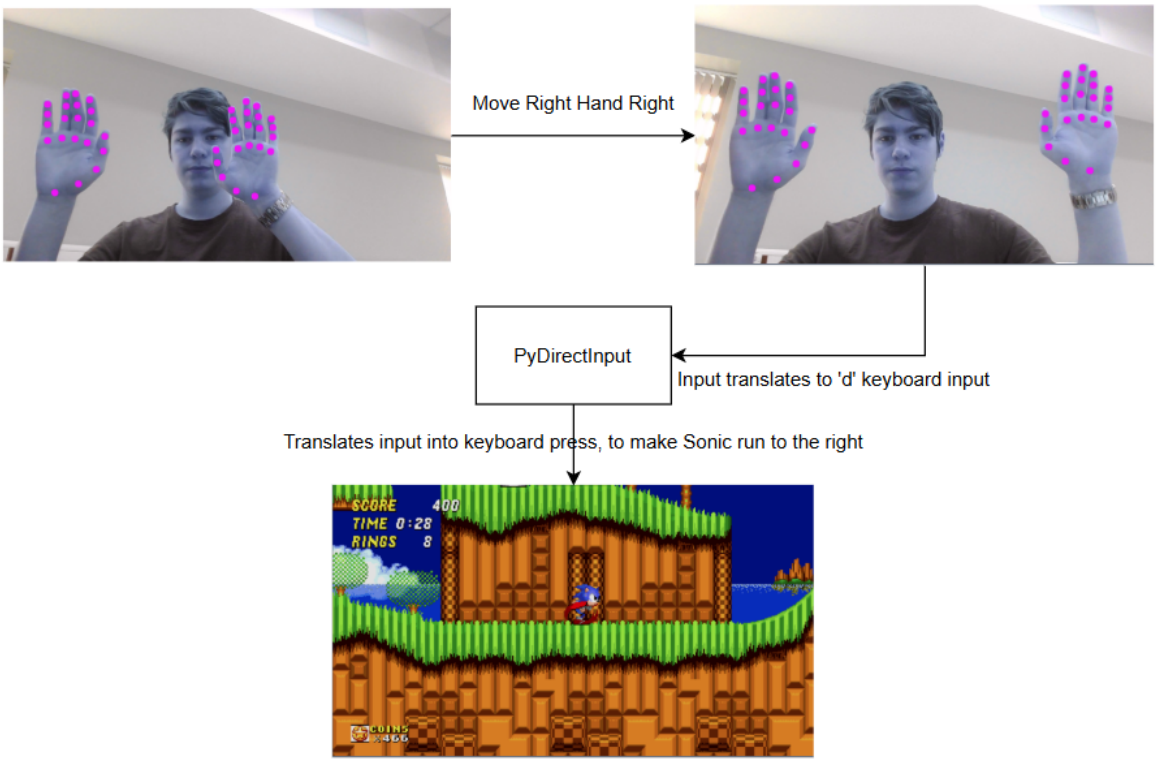


Figure 1.1: Diagram to show the pipeline that GameTrack follows to translate inputs into the game

In this paper, we present a method for controlling video games, which we call GameTrack, that is more accessible and affordable to people that are unable to use a traditional controller. In this case, people with Parkinson's Disease, preventing them from being able to control the small, precise movements that are required to use a controller or keyboard.

Body Tracking using computer vision techniques can have many different applications in many different domains, such as assisting with use of computers and other devices, machine translation, like translating American Sign Language (ASL)[15], or even controlling video games[20].

These tracking systems tend to assume that the user is able-bodied and can move in a

smooth motion, as the majority of people can. For this reason, it can be very difficult to create a system that can cater to people who are less able-bodied, who are unable to perform a smooth movement with their arms[25].

In the case of hand tracking, it can be difficult to keep track of a hand that can only move in big, jerky motions, and when working with something like gesture recognition, the hand needs to be still and able to form the correct gestures, which many people with movement conditions will be unable to do. We are looking to create a system that can allow people with movement conditions to be able to play video games. For this reason, we need to utilise a system that can track these jerky movements in real-time, otherwise this would introduce latency into a reaction-based game, in the case of this project, Sonic the Hedgehog 2.

Using MediaPipe Hands [24], we have made GameTrack, a hand tracker that aims to cater to the natural movement patterns of this user pool (see section 4.1), using big, jerky motions to transfer inputs into the game.

In our literature review, we will explore different hand tracking methods, gradually narrowing down to our final choice, and showing why we have chosen that over any other method.

We will present our motivations for this project, and an overview of our solutions to the problem outlined above. We will then explore the state-of-the-art for tracking and object detection, narrowing down to our chosen method for implementation. We will then provide a detailed description of our implementation, our hypotheses for the experiments, and the results of those experiments, followed by a detailed analysis of those results and a reflection on the participants' thoughts. We will conclude with a summary of the results and what they mean for the wider realm of hand tracking and assistive computing, along with overviewing possible future work and improvements to the product.

1.1 Motivation

The video game industry is one of the largest in the world, generating an estimated revenue of around $455 billion globally, and $97.7 billion in the US alone in 2024 [3]. Such a large industry should be able to be enjoyed by everyone, regardless of ability. Most video games are played either with a controller (see Figure 1.2), or with a mouse and keyboard. Both of these control methods, particularly with the keyboard, require small, precise movements to input the intended command. This can take a lot of getting used to for fully-abled people, and can be impossible for people with certain medical conditions to use. Members of our user pool often move their arms in a large, jerky motion and, due to hand tremors, it can be very difficult for them to press the intended buttons on a controller or keys on a keyboard reliably.

Figure 1.2: A PS5 Controller

Many video games offer accessibility options, but they are mainly visual options or ways of making the game easier to complete, rather than a more accessible control scheme. Companies like Playstation and Xbox have attempted to offer a more accessible control scheme by making accessible controllers for disabled users. An example of an accessible controller is in figure 2.2, with the Playstation Access Controller. This somewhat addresses the issue of needing small, precise inputs by having bigger buttons to press, however you still need to use a joystick, which can be very difficult for people with conditions affecting their hands and wrists.

As well as the issue of the small movements persisting with this controller and other accessible controllers like it, it is also extremely expensive to buy, with the controller in figure 2.2 costing $89.99 on the Sony Store[4], which is even more expensive than the regular PS5 controller, at $74.99 (see section 2.3).

There are many different reasons that someone may not have the dexterity or ability to play games with a controller or keyboard, such as Parkinson's Disease, Repetitive Strain Injury (RSI), or even having a broken hand or other hand injury. This project is more focussed on the hand tremors and jerky movements that come with Parkinson's Disease, but we will discuss possibilities to adapt this to other injuries. RSI often affects peoples' wrists and fingers, so having a product that can allow them to move their upper arm rather than their hand would be ideal for that kind of injury (see section 7.2 for a possible solution that could be made).

1.2 Summary of Product

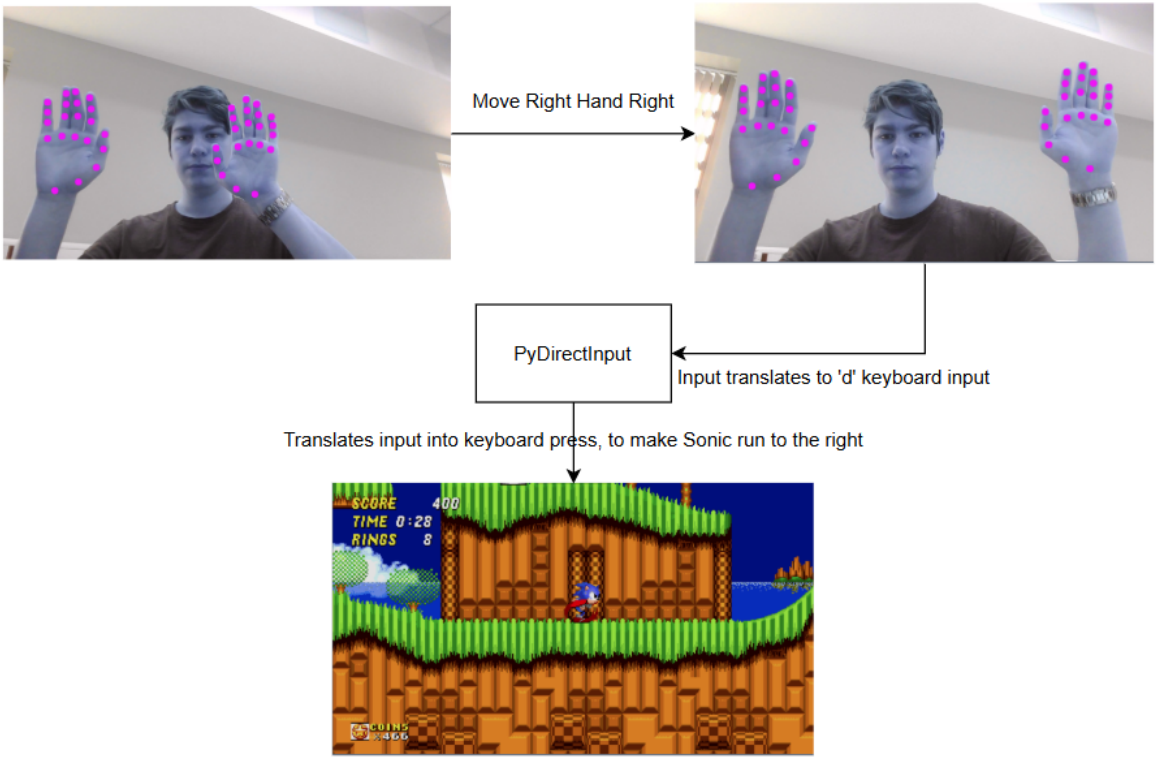


Figure 1.3: Reminder of the basic tracking pipeline

We have developed a control scheme specific to those mentioned above, who cannot control

small, precise movements with their hands. Using MediaPipe Hands [24], we have made a hand tracker called GameTrack that exclusively registers large, jerky movements as inputs, ignoring anything that is too slow or small. Each hand applies to different inputs, with each hand being able to input 4 different commands by moving up, down, left, or right. Using PyDirectInput[16], we map these movements to different keys or commands to act as inputs to the game that the user is playing. We used PyDirectInput over PyAutoGUI because it uses DirectInput scan codes and the more modern SendInput function, which works in more applications, as opposed to the functions that PyAutoGUI uses, which do not work in certain games, like the one we used to test GameTrack: Sonic the Hedgehog 2. This is a simple 2D platformer, with 3 main actions: run, jump, and "spin dash", which is an action which greatly increases Sonic's speed from a standstill. The simplicity of this game will provide a good baseline for how this system can be used to play games. See chapter 3 for a more in-depth description of the system.

Figure 1.4: Camera stream with overlay for the 21 landmarks on each hand

1.3 Project Aims

Our main aim in this project is to be able to create a hand tracker that can be used by people with movement conditions to be able to play video games. We will measure this goal in two ways:

- Comparing how accurately participants are able to use GameTrack while playing a game, and if their accuracy improves level to level
- Comparing in-game performance when using GameTrack vs using a controller

We will address shortcomings of other products within the domain of tracking and of accessibility in video games when it comes to this particular project, and we will propose a method that will attempt to improve upon these for use by less able-bodied users. In chapters 5 and 6, we show that our participants were able to improve their accuracy of input for GameTrack while playing the game, showing that they were able to learn an effective method of input in a short amount of time, and that, given more time with GameTrack, they will be able to use it much more effectively. We conclude that users that were inexperienced with a controller were able to learn GameTrack more effectively than those who were experienced,

making it more likely that our intended user pool are able to learn how to use it and be able to play games (see chapter 7).

Chapter 2

Literature Review

2.1 Overview of Problem

The main aim of this project is to allow people to play video games when they otherwise would not be able to. It is aimed towards those with movement conditions such as Parkinson's Disease and who do not have the fine motor skills to use a controller, or a mouse and keyboard. Current existing accessibility options for controllers still have issues regarding this problem, and are extremely expensive, potentially making it infeasible for people with movement conditions to play games with their children or grandchildren, or to try out any games themselves. We are looking to create a tracker to control PC games that is affordable, with the only expense being a budget webcam to record the video input.

2.2 Structure of Literature Review

This literature review will cover different types of trackers and object detectors, as well as looking at existing accessible methods for game control. We will cover the state of the art, pose estimation and how that relates to controlling games. We will then go over gesture recognition for hand tracking and sign-language. We will finally cover real-time 2D-Image-based methods and MediaPipe, which is what we ultimately chose to use for GameTrack.

Our ideal tracker would ignore any small movements, such as those that come with tremors, and would require quick motions to register, removing the need for smooth motion. In the 18 people studied in [25], 17 had more tremors in their right hand than their left, and 1 in their left more than their right. Since there is a possibility of people having more tremors in different hands, as well as registering different inputs with different hands, we will allow the user to customise each hand's inputs based on their needs and preferences.

For many people in our user pool, it will likely be their first time playing a game, perhaps wanting to play with their child or grandchild, so GameTrack will need to be simple enough for the user to be able to get used to it quickly and to be able to see their inputs on the screen, for easier adjustment to their hand positions.

2.3 Accessible Controllers

There are two main accessible controllers on the market right now: Playstation's Access Controller (see figure 2.2), and Xbox's Adaptive Controller (XAC, see figure 2.1). Both of

these controllers feature much larger buttons than on the usual controllers, allowing disabled users to more easily and reliably use each button. The XAC has two very large buttons that act as the A and B buttons, with all of the other buttons needing external switches that connect to the XAC. This means that the XAC is mainly used as a hub to create a control scheme that works for each specific user. This is a good idea that allows lots of customisation, however this also requires the user to buy many more switches to connect to the XAC, which already costs $99.99 on its own[5]. This is already close to double the price of regular Xbox controllers on the Xbox store, before the purchase of external switches.

Playstation's Access Controller, while not offering quite as much customisation as the XAC, still utilises the large buttons for more reliable inputs, and is cheaper, at $89.99 on the Sony store, while still being a full control scheme that does not require the purchase of external switches. This controller, however, does still use a small joystick, which can be very difficult for people with movement conditions to utilise.

Figure 2.1: Xbox Adaptive Controller

Figure 2.2: Playstation Access Controller

2.4 Trackers

In this section, we will go over different methods of tracking and object detection, and their compatibility with this project, narrowing down to a tracker that works well for the outlined problem. Most trackers assume that the user can move in a smooth motion, which is an assumption we cannot make for this system. We need a system that can pick up, in real time, a hand that is moving in a jerky motion with enough accuracy that we can register the correct input, and the detection of the hand placement is not affected.

2.4.1 Attention-Based Methods

The state of the art for image processing is attention-based methods using transformers[23]. Transformers use self-attention mechanisms to train an encoder-decoder architecture solely using attention, allowing for significantly more parallelisation than CNNs. For this architecture, inputs are split up between Query, Key, and Value vectors which are trained alongside the rest of the model. Since the different vectors are all accessed by different parts of the model, this attention encoding can be fully parallelised for every input (See figure 2.3 for model architecture).

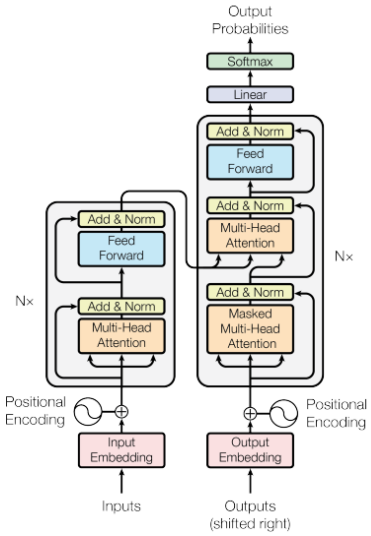


Figure 2.3: Transformer Model Architecture[23]

Methods like that shown in [12] are effective in using the attention architecture for visual tracking, however the experiments are done using an Nvidia GeForce RTX 2080 Ti, which is quite expensive for the average user that may have a lower performance graphics card, possibly resulting in more latency. The results and speed of the experiments shown in [23] were achieved when training using 8 Nvidia P100 GPUs which, while they were launched in 2016, are currently extremely expensive to purchase. Given the amount of parallelisation in this architecture, significant computing power is required and, with users looking to use this product on a lower budget, it is infeasible to use attention-based methods. We need to look to more traditional methods that require less computational power, and still push to real-time functionality.

2.4.2 Pose Estimation

The most popular form of using your body to control games was the Xbox Kinect, which uses depth imaging to track a user's pose [13]. This is done by splitting the body into different body parts, clustering the pixels for each to estimate the joint positions, then connecting those joints to fit the skeleton (See figure 2.4). A more in-depth look at the algorithm for clustering and joint predictions is found in [20]. Since this is tracking the whole body, there are more possible external factors to interfere with the tracker, such as clothing and differing body proportions. Also, since all of the points are being connected for the pose estimation, any small inaccuracies can affect the final result dramatically.

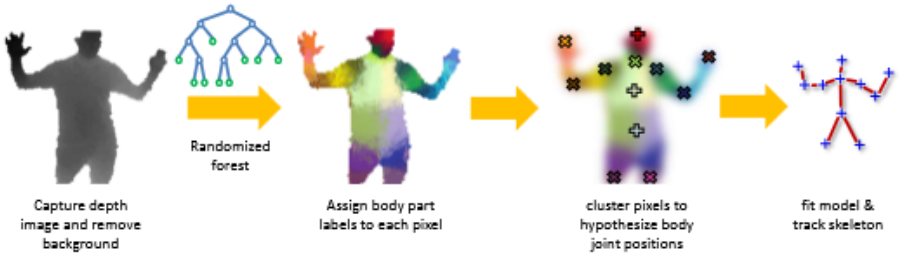


Figure 2.4: Kinect Skeletal Tracking Pipeline[13]

An issue that we encounter with using the Kinect is that they are quite inaccessible in recent times, given that Microsoft have discontinued their production, so the only available ones are pre-owned. If we were to purchase a 3rd party depth camera for the same purpose, they are extremely expensive [6], and infeasible for the average consumer to buy for use with video games. For this reason, we would prefer to use a standard RGB camera, as they are much cheaper to attain and many people will already own a webcam.

When we look at pose estimation using RGB cameras, we can look towards Tensorpose[21] for real-time pose estimation. Similarly to the Kinect, this achieves markless motion capture, allowing the user to use the software without the need to put on a certain set of clothes or markers for the system to track. This aims to greatly reduce the computational intensity of previous pose estimation methods that use traditional CNNs for real-time applications. Tensorpose replaces conventional convolution, applying a tensor decomposition [14]. It also incorporates Sparse Optical Flow and Kalman Filters to smooth movement for motion capture, considering the relationship between each frame to capture the movement of the user.

Figure 2.5: Results captured from the Tensorpose algorithm[21]

Considering the relationship between frames is important for creating a usable tracker for this project, as we need to be able to register big jerky motions as input, so we need to see the distance and direction the hand is moving in order to check whether the user has input correctly.

When controlling a video game, especially one that has not been entirely designed around the use of pose estimation, we want to have as few unnecessary features as possible in order to reduce inaccuracies, to reduce computational intensity ensuring real-time application, and to reduce confusion for the player, when we can achieve the same function just using hands rather than the entire body.

2.4.3 Gesture Recognition

Initial ideas for this project involved using gestures to map to different inputs within the game. There has been research into using gesture recognition to translate American Sign Language (ASL) to help deaf people communicate more effectively. Many gesture recognition models involve using depth cameras in order to build a full 3D model of the hand, like that proposed in [19] (see figure 2.6). This, while accurate and real-time, requires the purchase of a depth camera over an RGB camera, which is much more expensive. (see section 2.4.2)

Figure 2.6: Hand models produced in [19]

For this project, we also do not require a full 3D model of the hand. Earlier methods of translating ASL used RGB cameras while also being accurate. The real-time static hand gesture recognition system for ASL [15] uses a single RGB camera mounted on top of a monitor to capture the user's gesture, preprocesses the image by filtering noise and performs edge detection to keep the outline of the hand. It then crops the image to just the hand, extracts features and matches the gesture with the corresponding ASL letter. Matching the features was simply done by calculating the Euclidean distance between the feature vectors of the input frame and the training images.

Looking back at the specific user pool of this project, gesture recognition would be difficult to work with, considering they have difficulty controlling small movements and may have trouble forming the required gestures for each possible input. To remedy this issue, we look to a different method of input, where the user will move their hand in a big, jerky motion in order to register an input, by keeping track of the hand's position before and after input.

2.4.4 Real-Time 2D-Image-Based Methods

Many traditional CNNs are unable to work in real-time due to the computational cost of looking through the entire image every frame in order to detect the desired object. Selective search [22] acts as the basis for a Region Proposal Network (RPN)[18] which greatly optimises the detection algorithm by narrowing down the area that the detector searches, however the highest frame-rate that the Faster R-CNN can reach is 17 FPS[18]. This is not ideal when working in the realm of controlling video games, where the lowest frame-rate a game will run is 30 FPS, so the tracker would only work roughly every 2 frames.

In 2016, a detection method called "You Only Look Once" (YOLO) was introduced that pushed for more real-time object detection[17]. This method frames object detection as a regression problem rather than classification, and uses a single neural network to predict both bounding boxes and class probabilities from full images in a single evaluation. This method achieves an average of 45 FPS, but the increase in speed, in this case, has a detrimental effect on the accuracy, especially when working with smaller objects. When dealing with hand tracking, it would come across issues in localisation often given that the user may be standing at a significant distance from the camera, making the hand much smaller in the frame, and the big jerky motions would also contribute to inaccuracies. Given the intricacies of the hand, and the fact that many people have different hand shapes and finger lengths, YOLO would encounter issues for this use case.

MediaPipe Hands is a very accessible framework for hand tracking that works in real time, without sacrificing accuracy, since it has been optimised for use with hands exclusively.

2.4.5 MediaPipe Hands

MediaPipe Hands [24] allows us to track the position of the hand using a mechanism similar to BlazeFace [11], which can detect faces in less than a millisecond, extracting features from the image, taking a 128 × 128 pixel image and running it through a 2D convolution layer and a series of "BlazeBlocks" to extract the necessary features. MediaPipe Hands uses a similar architecture, but detects the palm of a hand. It uses the palm because images can then be labelled with consistently square bounding boxes, without having to account for the shape and length of fingers to make the bounding box. After it has detected a palm, the cropped image is used as input to another model to detect 21 landmarks on the hand (See figure 2.7).

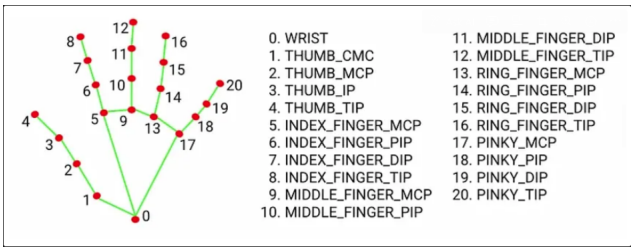


Figure 2.7: Diagram of the hand landmarks for MediaPipe Hands[24]

This software can run in real time, and is much more efficient than other object detection and tracking devices because the palm detector part of the tracker only needs to run when the hand has not yet been detected, that being when the code is first run, and when the hand has gone out of view. This is achieved by using the hand landmarks from the current frame to determine the location of the hand in the next. This is a similar concept to using selective

search for RPNs in Faster R-CNN, but improves upon it by, instead of just proposing a smaller region to detect an object, it uses the current frame to find the palm, and does not need to run the detector at all. This means that the palm detector does not need to run frequently, and it only needs to run the landmark detection part of the software when the hand is in view.

This does however present a problem with GameTrack when it comes to quick, jerky movements, as when the hand moves too quickly, it can no longer use the current position of the hand landmarks to find it on the next frame. This means that when there is a big movement from the hand, the tracker will lose the hand's position and need to run the palm detector again. However, as with Blazeface, it still runs extremely quickly and can detect the palm again in real-time, allowing us to use GameTrack to play a video game without significant latency.

The palm detector alone would be sufficient to build a basic hand tracker that can register inputs with large sweeping motions, but the landmark model can allow us to create more intricate controls than just sweeping the hand in a direction. We used the landmark model to detect whether a user is clenching their fist, which will force GameTrack to ignore any inputs, allowing them to bring their hand back into the frame if they are close to the edge, without unintentionally registering an input.

The MediaPipe Hands model has functionality to determine whether the user is showing their left or right hand, which allows us to map different controls to different hands. This functionality was achieved by manually labelling a subset of the real-world image data of the handedness [24]. This then allows the model to learn which hand is being shown, by learning whether the back or the front of the palm is being shown, and which direction the landmarks are going (i.e which direction the thumb is going).

This software can run in real time, on my Lenovo Yoga laptop, with an Intel Core i7-10750H CPU (2.6GHz), and an Nvidia Geforce GTX 1650 which was bought in 2020. This means that it will easily be able to run on the vast majority of recent devices, even when working with a budget, making the product easily accessible to people without having to buy an entirely new computer to run it. The tests for this product will be conducted with a webcam that was bought on Amazon for £20[7], which will be the only cost throughout this project, and the only potential cost to those who already own a somewhat recent computer. GameTrack has also been briefly tested with my laptop's internal webcam which, while lower quality than the Amazon webcam, can still track the hands so, if you do not want the extra cost of the webcam, the product can still be used with an inbuilt webcam (all experiments in this paper have been conducted using the Amazon webcam).

2.5 Summary of Literature

As outlined in the problem overview, we are looking to create a simple, affordable solution to allow users with movement conditions to effectively control video games without a keyboard or controller.

Given the high amounts of required parallelisation for state-of-the-art methods using transformers[23], [12], it will be infeasible to use these methods for this project as there may be high costs in needing to purchase a higher-performance graphics card in order to run in real-time. Methods that use pose estimation, like the Kinect [13], [20], while effective at controlling games that it has been specifically designed for, the fact that it has to create an

estimation for the entire body can introduce unnecessary complications and inaccuracies, so we looked to hand tracking for a simpler solution with less unnecessary computation.

Our initial idea was to use gesture recognition to register inputs from the user, however the most recent, effective methods use a depth camera [19] which, like transformers, introduce a higher cost than what we are looking for, so we look towards using RGB cameras, and gesture recognition can be difficult for people with movement conditions who may not be able to make the required gestures close enough to the accepted gestures to register inputs.

We looked towards more traditional image processing for real-time use, leading us to use MediaPipe [24], which allowed us to create a more usable method of input for the user pool, which exclusively registers inputs from big, jerky motions. The landmark model of MediaPipe Hands, as well as the functionality to distinguish between the left and right hand, allowed us to implement more functionality than just using a traditional CNN. For example, we were able to implement different controls for each hand, and to implement functionality to allow the user to ignore inputs when they clench their fist, allowing them to reset their hand position without accidentally registering an input.

Chapter 3

Product

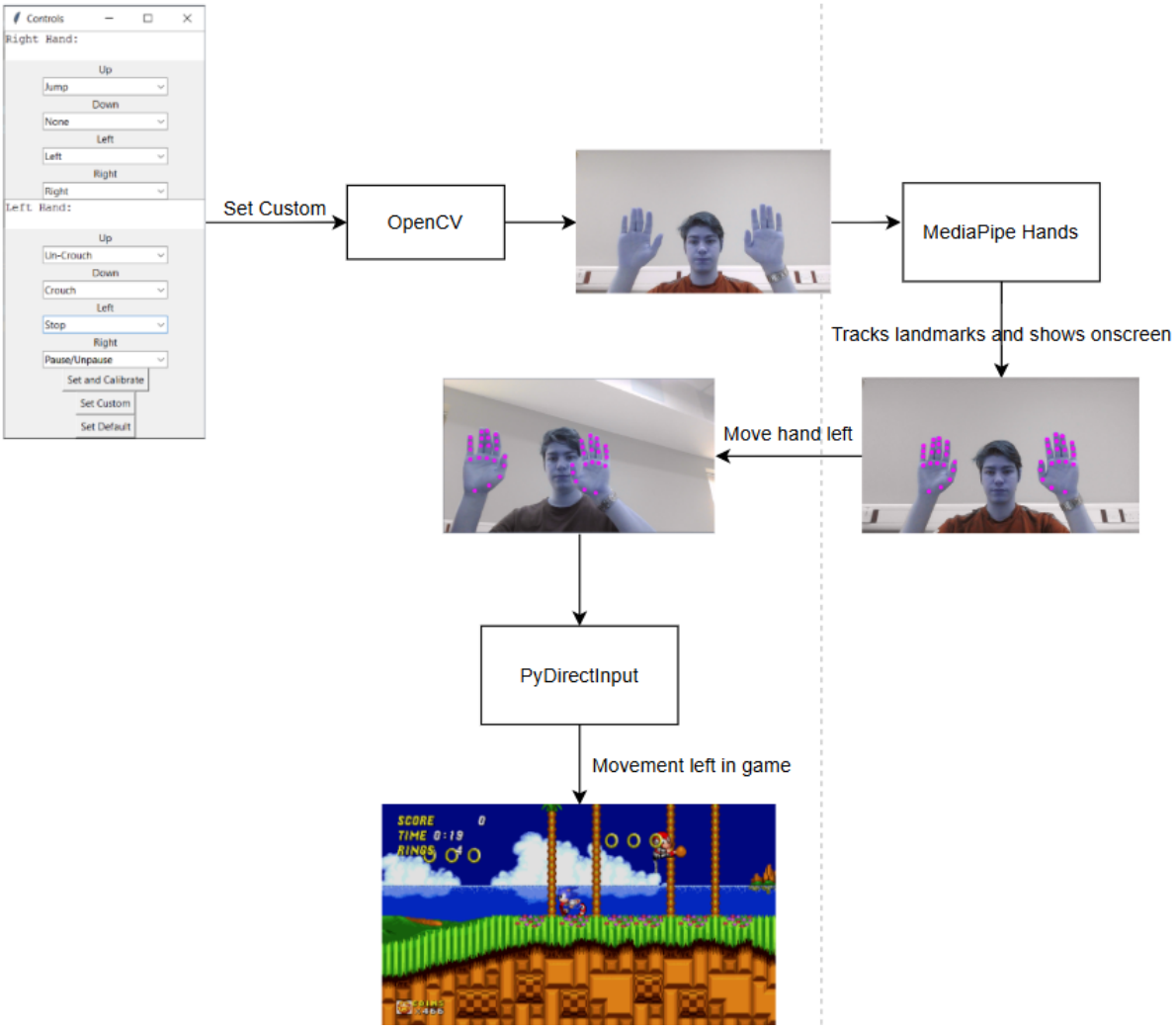
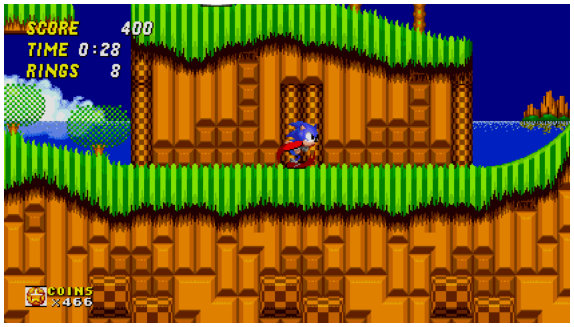


Figure 3.1: Detailed diagram of the full tracking pipeline from setting the control scheme to registering an input

GameTrack utilises OpenCV [8] for the camera stream and overlay (See Figure 1.4), and uses MediaPipe Hands [24] for the tracking system. Since MediaPipe uses the hand that has been detected in the previous frame to detect the landmarks in the current frame, it can be

(a) Right input on the camera stream (Moving the right hand to the right)



(b) Right input in-game

Figure 3.2: Right input on the camera stream and in-game, causing Sonic to run to the right

difficult for it to keep track of the hand when moving with a big jerky motion, until the hand slows back down. However, since the palm detector model can run in real-time regardless, we can use this to our advantage. We do this by constantly keeping track of the x and y coordinates of the hand in the previous frame that it was detected. This means that, when the hand is lost, it will have moved a larger distance from the previous frame, causing the system to register an input when the hand is recaptured. While the hand is being tracked, we check the individual x and y distances from the previous frame to see if a large enough movement has occurred to register an input. Our reasoning for doing this over checking a certain number of frames before, is that checking a certain number of frames back inherently introduces its own latency, potentially making the inputs slower than they need to be. This would be an issue when playing a game that relies on reaction inputs from the player, like Sonic the Hedgehog 2 does.

The overlay for the tracking shows every landmark on the hands that are currently being tracked. This allows the user to see if their hand is being tracked properly, and allows them to see why inputs do not register, and why they are registering when they do, helping them learn how to use the controls effectively.

MediaPipe also has functionality to detect whether the hand that is on-screen is the left or right hand. This means that we can assign different controls to different hands, and with 4 directional inputs on each hand (up, down, left, and right), we have 8 possible inputs for our system, which is more than enough to play the game. Each input has been hard-coded to correspond to each keyboard command for Sonic the Hedgehog 2, so if we wanted to use it to play a different game, the control scheme would need to be adjusted in the code. Each of the 8 commands corresponds to one specific keyboard input (or release of a key), meaning that commands that require multiple buttons on a controller to execute, such as the spin dash which requires the player to crouch, press the jump button, then un-crouch, will require 3 inputs from GameTrack to execute. To translate the tracking inputs, we used PyDirectInput over PyAutoGUI because it uses DirectInput scan codes and the more modern SendInput function, which works in more applications, as opposed to the functions that PyAutoGUI uses, which do not work in certain games, like Sonic the Hedgehog 2.

An example of one of the inputs can be seen in figure 3.2, showing the user move their right hand quickly to the right, registering an input that causes Sonic to run to the right. All other inputs, from the default control scheme, can be seen in the same format in appendix A.

Given the nature of GameTrack, it is possible that, if a user wants to input multiple movements in the same direction, that their hand would go off-camera, or they would accidentally register the opposite input when attempting to move their hand back into frame. To remedy this potential issue, we included a feature that, if the user clenches their fist, it will force the system to ignore any input from the user, allowing them to move their hand back into frame to prepare for their next input. This was done by checking the distance between the tip of the index finger, and the bottom of the palm (landmarks 8 and 0 on figure 2.7 respectively). If the distance between these two landmarks is small enough, then it registers as a clenched fist. To denote this on the overlay, it shows "No Input" in the top left corner of the screen.

While there is a default minimum distance for the hand to move to register an input, or distance between the index finger and the palm to indicate a clenched fist, the user is given the option of calibrating all of these at the start. This feature involves inputting each direction 3 times on each hand, then averaging the distance that it moves in those 3 iterations to give a new minimum distance for each input. The user is then asked to hold their hands their preferred distance away from the camera, allowing the system to check the distance between the index finger and the palm, to then use as a basis to check for a clenched fist. When calibrating, it checks the distance between them, then registers as no input when they are halfway or closer to each other. This will allow the user to be more comfortable when using GameTrack and allow them to cater it to their strengths. This customisation also extends to the control scheme itself, where the user is able to customise which hand movement performs which command (See Figure 3.3).

We used Tkinter[9] to make the GUI for customising the controls, making a simple drop-down menu for each of the possible commands and the user can choose which tracking input performs which command (see figure 3.4). "None" is included as a command within this drop-down menu, because there are 7 possible commands for Sonic the Hedgehog 2, as shown in the drop-down, but there are 8 possible inputs from GameTrack, so one of the inputs must correspond to nothing.

This further customisation allows the user to choose which control scheme would be the most comfortable for them, and be the easiest for them to remember in order to perform as well in-game as they can.

The feature for checking whether the hand being tracked is the left or the right is not perfect, and can sometimes detect the wrong one when the hand is moving vertically. Since it checks each hand individually, this can present issues with checking distance and can cause inputs to register when they should not. For this reason, we also introduced a maximum distance that the hand can travel to register an input. This causes the system to ignore any inputs when the hand has travelled too far, eliminating incorrect inputs from the other hand being detected incorrectly, while still keeping the intended input.

Figure 3.3: Calibration pipeline, from customising inputs to the calibration procedure

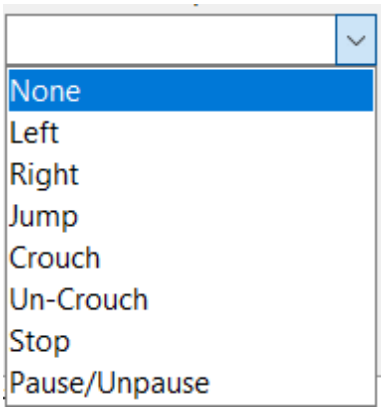


Figure 3.4: Drop-down menu for each input, showing the options for commands

Chapter 4

Experiment Design

4.1 User Pool

The main aim for this project is to cater to those with movement conditions such as Parkinson's Disease. Parkinson's occurs in 1-2% of people over 65, and the main symptoms include hand tremors and jerky movements in their arms [25]. This heavily restricts them from using a mouse and keyboard, as they require smooth, precise movements in order to use effectively, which can come with practice for able-bodied people, but people with Parkinson's may be physically incapable. Therefore, we need a tracker that can cater to the fact that the user cannot move their arms in a smooth motion, and that they cannot precisely control small movements.

4.2 Hypothesis

Participants from our user pool are likely going to be inexperienced with any kinds of video games, be that games that use a controller, or a keyboard. For this reason, we wanted most of our participants to be inexperienced with video games, but we also wanted some people who were experienced to be able to compare how each subset would react to using this new control scheme.

Our main hypotheses are:

- Inexperienced participants are likely to improve with GameTrack more quickly than the experienced participants.
- Inexperienced participants are likely to perform similarly in-game with GameTrack and the controller

This is because they have had little to no previous exposure with using a regular control scheme (keyboard or controller) for playing video games. Therefore, we believe they are more likely to take GameTrack at face value, while using it to learn the game itself and will be able to stay calm while using it, and learn more effectively how GameTrack is supposed to be used. In contrast, since our experienced participants have had previous exposure to regular control schemes, they are more likely to find GameTrack unnatural to use, which will cause them to become frustrated, and take longer to effectively learn how to use it accurately.

From these tests, we want to see how each participant goes about learning GameTrack, and their thoughts about using it, which will be able to fuel ideas for future work.

4.3 Tests

In order to test the ease of use of GameTrack, we needed to have participants who had never used it before. We were unfortunately unable to be supplied with participants that actually had Parkinson's Disease, due to complications with the University of Bath Department of Health. For this reason, we instead asked able-bodied participants to engage with the control scheme in the same way to how an intended participant would, that being, using large, jerky movements to register inputs, since that is the only way the system is designed to register inputs.

For the tests, the participants played the start of Sonic the Hedgehog 2 using both a regular PS5 controller (see figure 1.2), and with GameTrack. They played for 5 minutes with each control scheme, both from the start of the game. In order to reduce the possibility that the user would memorise the level design and improve their play based on that for the second control scheme, we alternated between participants starting with the controller and with GameTrack.

For the controller, we only recorded metrics about in-game performance: the number of times the user takes damage, the number of times they die, and the time it takes for them to complete each level (as shown on the in-game timer). For GameTrack, we also recorded these same metrics in order to compare the users' performances with each control scheme, however we also recorded performance metrics about the usage of GameTrack itself, generally: The number of attempted inputs by the user, and the number of those attempted inputs that registered as intended.

After using both control schemes, the participants were asked about their thoughts from using GameTrack: how easy it was to use, what they struggled with, or liked about using it. This was done to help gain some insight from a first-time user's perspective on what aspects worked and what needs improving, and the user's thought process when learning how to use GameTrack.

The purpose of these experiments is to both test the accuracy of GameTrack, and to test how easy it is to use it for the game itself, along with how quickly users were able to learn how to use it effectively. We had a mix of participants that were experienced and inexperienced with using a controller to compare how people of different skill levels would react to using the new control scheme, and whether it was easier for someone completely new to learn a new control scheme, or for someone who was already experienced with one control scheme to learn a second.

Chapter 5

Experiment Results

5.1 Test Results

When participants were using GameTrack to play the game, we kept track of the number of inputs that the user attempted, and how many were registered in-game as their intended input. This was done in order to test how effectively users were able to learn it having had no experience with it prior to the test. we kept track of these metrics individually for each level in order to track how well they were able to learn GameTrack over time and whether their overall accuracy of inputs increased level-to-level. We split our test participants into two groups, based on whether or not they were experienced with using a controller. The results for each test group are shown in figures 5.1 and 5.2.

Tables 5.1 and 5.2 show the accuracy differences between level 1 and 2 for each participant, along with the average difference for each test group, where a positive difference means that their accuracy improved from level 1 to level 2, and negative means their accuracy declined. Accuracy percentage difference is defined as:

$$Accuracy(lvl_2) - Accuracy(lvl_1)$$

where

$$Accuracy(lvl_x) = \frac{registered\ inputs(lvl_x)}{attempted\ inputs(lvl_x)}$$

When the outlier from the inexperienced test group (participant 4, the only participant from this group who's accuracy declined between levels 1 and 2) is removed from the average calculation, the average accuracy difference increases to 11.16%. Implications of these numbers and trends will be discussed in the next section.

	Part. 1	Part. 2	Part. 3	Part. 4	Part. 5	Part. 6	Average
Accuracy Diff. (%)	16	3.8	16.8	-10.4	15.8	3.4	7.57

Table 5.1: Accuracy difference in accuracy between level 1 and 2 for inexperienced participants

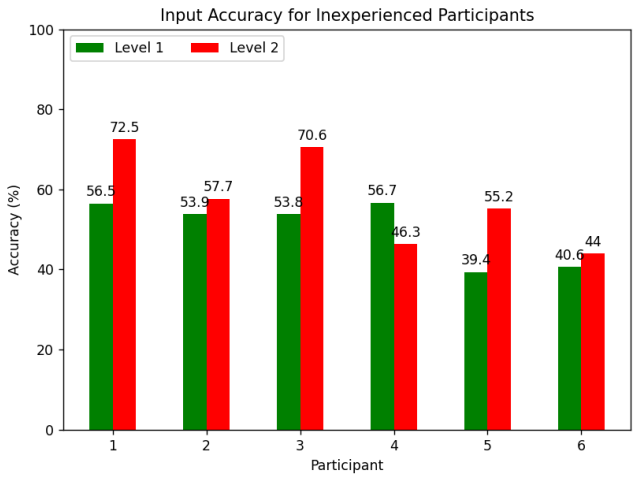


Figure 5.1: Percentage of attempted inputs that were registered as intended, per level for inexperienced participants

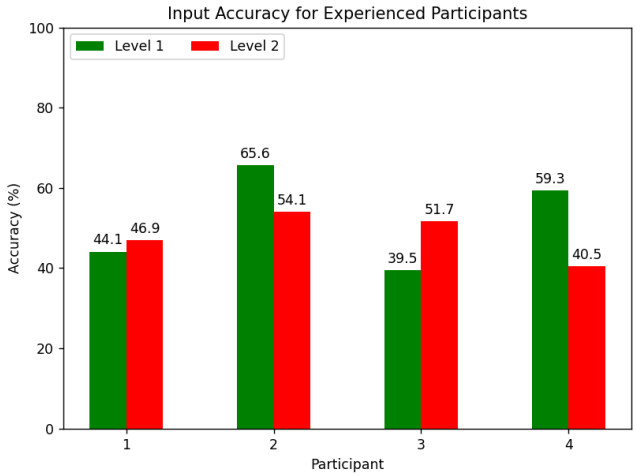


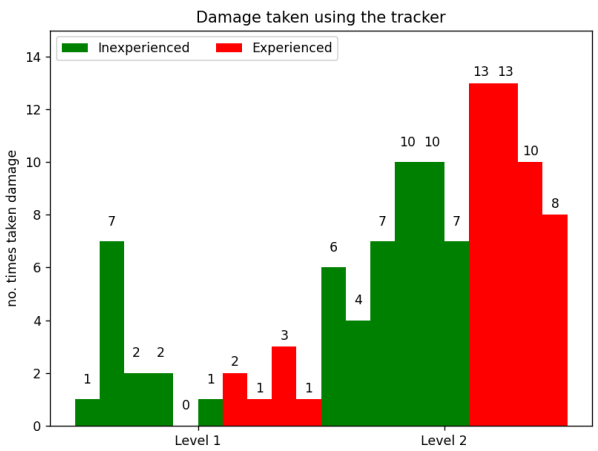
Figure 5.2: Percentage of attempted inputs that were registered as intended, per level for experienced participants

	Part. 1	Part. 2	Part. 3	Part. 4	Average
Accuracy diff. (%)	2.8	-11.5	12.2	-18.8	-3.83

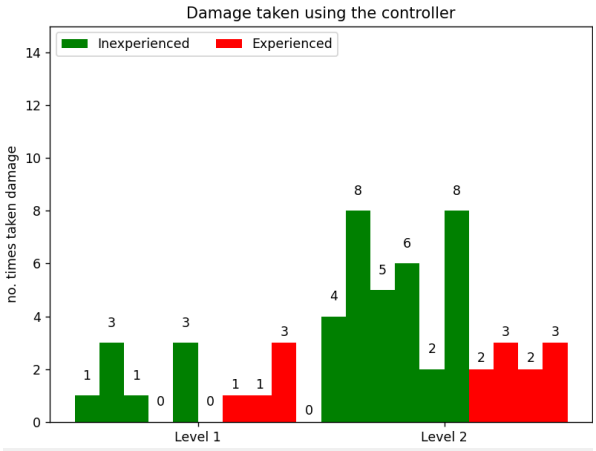
Table 5.2: Accuracy difference in accuracy between level 1 and 2 for experienced participants

5.2 In-Game Performance

As we see from figures 5.3, 5.4, and 5.5, the in-game performance was generally worse when participants were using GameTrack on all metrics (See section 4.3 for a list of the metrics), with more of a disparity in performance from experienced participants. When the time measurement shows 0 on figure 5.5a, it means that the participant did not complete the level. The disparity is worse with experienced participants because, as we saw with the tracking results in figure 5.2 and table 5.2, it was generally more difficult for the experienced participants to learn to use GameTrack within the 5 minute time period.

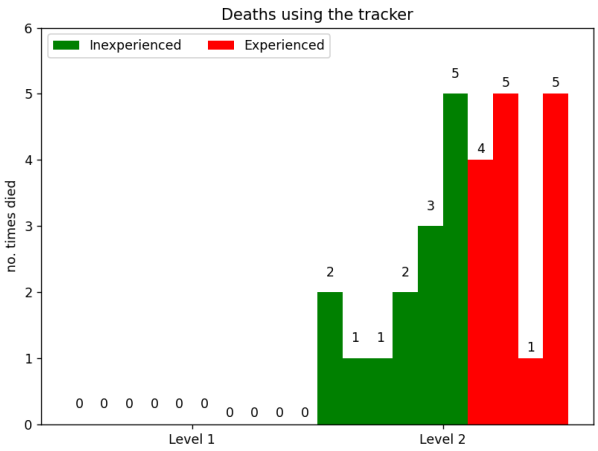


(a) Damage taken using GameTrack

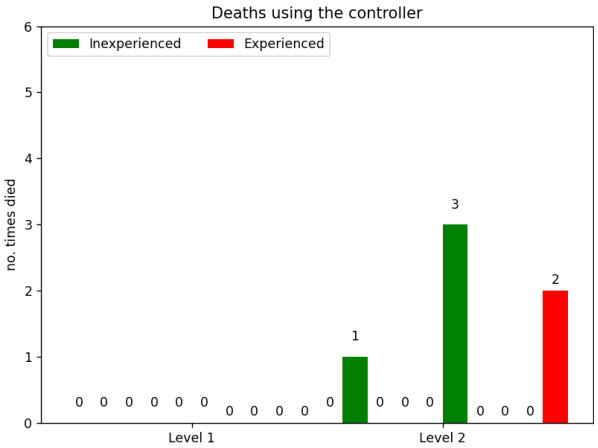


(b) Damage taken using the controller

Figure 5.3: Comparison of the number of times taken damage using GameTrack vs the controller between inexperienced (green), and experienced (red) participants



(a) Deaths using GameTrack



(b) Deaths using the controller

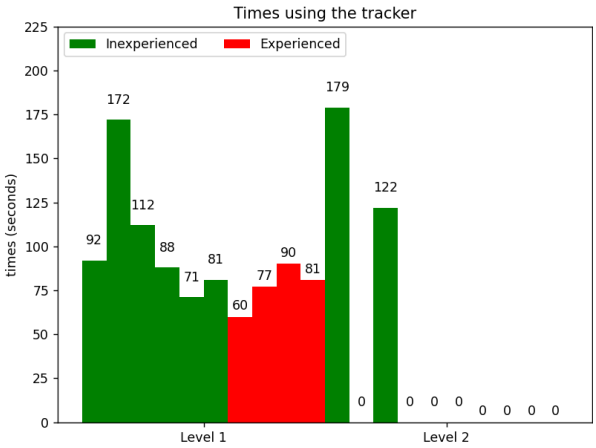
Figure 5.4: Comparison of the number of times participants died using GameTrack vs the controller between inexperienced (green), and experienced (red) participants

5.3 Participants' Thoughts

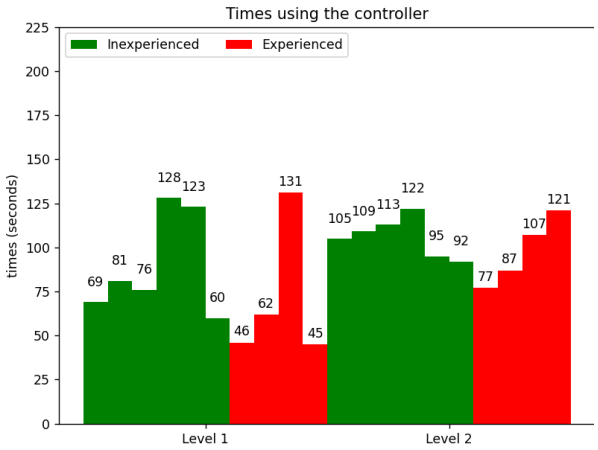
After using both GameTrack and the controller, the participants were each asked their thoughts about using GameTrack.

Most people, towards the end of the study, enjoyed using the control scheme, finding it immersive and they enjoyed physically controlling Sonic with their hands, once they began to get used to the new control scheme. They particularly enjoyed the action of physically charging the spin dash with multiple upward inputs, and enjoyed the feedback of letting go and seeing Sonic set off quickly. One participant compared GameTrack to playing VR games such as Beat Saber [10], in which the user uses their hands to control a lightsaber to hit blocks in rhythm.

Some people found it difficult to keep their hands parallel to the camera when moving. One of the problems that was brought up most by participants was forgetting which input



(a) Times using GameTrack



(b) Times using the controller

Figure 5.5: Comparison of the time taken to complete each level (seconds) using GameTrack vs the controller between inexperienced (green), and experienced (red) participants

performed which command, causing them to have to stop and think, or input multiple commands to attempt to get the correct one. They found that the initial control scheme that they had decided on was not the best one to use when actually playing the game, and they needed the hands-on experience of playing the game with GameTrack in order to figure out what controls would work best for them in the future, improving their play as they spend more time using and learning it (See section 7.2 for an expansion on how we can fix this issue).

When talking about the responsiveness of the controls, stopping movement was brought up as a particularly unresponsive command due to how stopping works in the game's logic, which is out of control for GameTrack. On the other hand, people generally found movement and jumping to be responsive.

Having the camera feed available to see at all times, along with overlays providing feedback on what GameTrack can see was extremely helpful to the participants in showing them why a particular input would fail, allowing them to adjust their movement in order to improve their accuracy and play the game more effectively.

Chapter 6

Analysis of Results

6.1 Tracking Results

From the bar chart in figure 5.1, we can see that most inexperienced participants were able to get used to using GameTrack fairly quickly, improving their accuracy of inputs between level 1 and level 2. This trend from our inexperienced participants shows that, while there may be a higher skill floor for most people using GameTrack compared to pressing buttons on a controller, meaning the baseline skill required for our specific participants was higher with GameTrack, improvements were still made in quite a short time-frame, over the 5 minute period that they were able to use it. The fact that the participants in our study were all fully able-bodied with no issues with fine motor skills, contributed to their ease of using the controller over GameTrack when starting out, however, when people with movement conditions try to use the controller, it is likely that they will find it much harder than our participants did.

Since our participants are fully able-bodied, some of them found it difficult and unnatural to use jerky motions when using GameTrack, but most of them were able to get used to it quite quickly and improve their accuracy in the second level if even just by a small margin, despite all but two people not having time to complete the level. In general, the inexperienced players were able to stay calm whilst using GameTrack, since they were not used to any control scheme, and were learning the game fresh, with no prior experience in using either control scheme. This meant that they would not subconsciously fight against it, expecting it to do something that it wasn't, they were learning how to play the game, using the control scheme that was presented to them.

Contrasting that with our experienced participants, their average accuracy difference was slightly negative, with two participants' accuracy declining, and two participants' accuracy increasing. It seems much more likely for people that are experienced with controllers to either panic or get frustrated early on when learning such a different control scheme, causing them to become less careful with their inputs, and therefore not registering them correctly. This led them to often forget what controls performed which command and start erratically moving their hands in the hope that it would register a correct input, and getting frustrated when it did not, inflating the number of attempted inputs that did not register (See section 7.2 for an expansion on how we can fix this issue in the future).

We also found that the background of the camera shot and the lighting did have some impact on the accuracy of GameTrack. For example, if the background was very clear and mainly one colour, it was much easier for participants to start with to register inputs, and if

the background was more cluttered, it became more difficult for the participant to register certain inputs, mainly up and down movements. Also, if the light was shining towards the camera lens through a window, it would struggle slightly more to detect the hand. However, even in this situation, participants were still able to figure out a good way to more reliably register inputs, and still improve their accuracy throughout the study.

6.2 Performance in-game

On average, performance in-game was worse with GameTrack for both inexperienced and experienced participants (see section 4.3 for a list of in-game metrics). The time it took to complete the first level was generally more with GameTrack than the controller, while people were learning how it worked, and most did not complete the second level. Out of the two people that completed the second level with GameTrack, it still took longer, with more damage taken and more deaths (partially due to taking a different route, neither died on the boss fight, just in the level). However, as users improve with it, this will improve their times on the level and they will be able to progress further into the game. It shows that people who learnt how to use GameTrack exceedingly quickly within the 5 minutes were able to get the furthest in the game, and shows the potential for beginners to make it further as they learn to improve.

The performance shows that GameTrack has a higher skill floor to be able to use for able-bodied people. This means that, for our able-bodied participants, it was easier for them to pick up and play using the controller for the first time than when using GameTrack for the first time. Some participants found the new control scheme unnatural to use at first, so to be able to fully judge their ability to use it, more testing will be required and they will need more time with GameTrack to make it more natural for them to use. When testing with participants of our intended user pool, it is hopeful that they will initially find GameTrack easier to use than the controller, however we will need to perform tests using them to verify.

The system is not built with our current participants in mind, leading them to perform worse with an unnatural control scheme for them. The system is designed to be more natural for people with movement conditions so it is more likely for GameTrack to be more natural to them than the controller, leading them to perform better when given GameTrack for the first time than when given the controller for the first time.

6.3 Analysis of Participant Comments

The thing people most enjoyed about using GameTrack was that it made the game more immersive for them, allowing them to physically control Sonic, over just pressing buttons on a controller. The participants that got used to the control scheme more quickly than the rest, were able to settle more into the game and play it without worrying about the way they were moving their hands, since they were doing it naturally more accurately. This allowed them to have more fun with the game itself and beat the second level within the 5 minutes. This shows that GameTrack can potentially have a wider impact to general users, not just members of our user pool. If we add more customisations to the control scheme itself (see section 7.2), we can offer an even wider user pool for GameTrack, allowing them a fun, new way to play games that they may have already beaten many times.

Inexperienced participants generally enjoyed their time with GameTrack more than experienced participants, but this was not 100% the case for all participants, with some experienced participants enjoying being able to play a game with a new control scheme, and some inexperienced getting frustrated. Since our intended user pool are likely to be inexperienced with video games, this trend shows that they may be more likely to enjoy using GameTrack, finding it immersive and will enjoy physically controlling games over the less physically engaging way of using a controller. With one participant comparing the use of it to playing Beat Saber (See section 5.3), this shows that they enjoyed using it, and found it immersive enough to use, that it reminded them of an official VR game, with a more expensive control scheme and device, showing that we have been able to create a fun and engaging way of playing, on a much lower budget.

One of the biggest problems was participants forgetting which input corresponded to which command, causing them to either half-heartedly attempt an input, or attempt multiple inputs to try and register the one they want. This would often inflate their number of attempted inputs that did not register. Most said that the reason was that the control scheme they were using was unintuitive for them, and they found themself trying a different hand command, thinking it would be the correct one. They needed the hands-on experience of playing the game to figure out what control scheme would be the most intuitive for them, allowing them to think less about which hand does what, and let them just play the game. They would need more time with GameTrack to fully learn which control scheme works for them and to allow them to learn it well enough so that they do not need to think about their inputs and to fully focus on the game. Their improvement using GameTrack shows that, as they began to remember the control scheme a bit more, as well as learning the correct movement for it to actually register, they would be more accurate and confident in their inputs.

Something that a couple of participants found unnatural was having to keep their hands parallel to the camera as they moved to ensure that they would not move their hands back far enough for the software to think they did not want to register an input. Since we use the distance from the tip of the index finger to the bottom of the palm to check if the user is clenching a fist, if the hand is too far back, the distance is small enough for it to think their hand is clenched. The distance needs to be set at the start to check against since, if it checked every frame and updated the distance, the user would not be able to clench their fist quick enough for the software to register that their fist is clenched. This could be updated by using the depth of the hand to scale the distance, although the measure may be inaccurate when we are not using a depth camera. It could also be updated by including software to detect gestures, similar to [15], that way the user can change their distance from the camera as they please, making it less rigid to use if they prefer.

Having the camera feed with overlays showing the tracking was extremely helpful as a learning tool for the participants in helping them understand the kinds of movements that it would respond to, and helping them see why certain movements would not register, as well as being able to see whether their hands are in the frame. One participant said that they became too engrossed in the game and forgot to check the feed, which caused them to move their hands off-camera without realising. This may be because of where the feed was framed in relation to the game, so it would be better to have the camera feed very close to the user's eyeline while they are playing, in order to be able to see where they are from their peripheral vision.

A final problem that was brought up by participants was that stopping was unresponsive.

When the user wants to stop, they would input the correct command, and it would be the same as if they were to let go of the directional button on a controller. This means that Sonic would gradually come to a stop as his momentum decreased. This may be improved by the stop command causing Sonic to move in the other direction for a very short time then stopping, as that would halt his momentum quicker and potentially feel more responsive for the user. Improvements to this would need to be hard-coded as their own inputs, for example having one hand input corresponding to multiple in-game inputs, causing Sonic to stop quicker. It would be very difficult to generalise the control scheme to the game's movement logic, so, for this iteration of GameTrack, we have kept all hand inputs as one in-game command.

6.4 Link to Hypotheses

When we look back at our hypotheses in section 4.2, we can see that our first hypothesis: inexperienced participants are likely to improve with GameTrack more quickly than experienced participants, was proven true, since the average accuracy difference from level 1 to level 2 was improved for inexperienced participants (see table 5.1), and declined for experienced (see table 5.2).

Since in-game performance was generally worse for all participants using GameTrack over the controller, our second hypothesis: inexperienced participants are likely to perform similarly in-game with GameTrack and the controller, was proven false for our group of participants. This could partially be due to the fact that they are all able-bodied and a controller is more naturally designed for them. To confirm this hypothesis, we will require more testing with less able-bodied participants, where GameTrack is more likely to be natural for them than for our current participants.

Chapter 7

Conclusions and Discussion

7.1 Conclusion

This project set out to create a hand tracker that would allow our user pool, people with movement conditions like Parkinson's Disease, to effectively play video games. This user pool will likely be inexperienced with any control scheme when it comes to playing video games and will likely want to be playing games with their loved ones, rather than trying to find a new control scheme for games they have already played.

When we look at the results for inexperienced participants, we can see that the majority of them improved with GameTrack and enjoyed using it more towards the end of the study. Inexperienced participants were shown to be more likely to improve with the control scheme than experienced participants, and tended to enjoy using it more. In addition, people who were able to get used to the control scheme and use it effectively were able to play the game much better than those who would fight against the controls or panic when using them.

The participants in our study were not members of the user pool, but were fully able-bodied. Despite GameTrack not being specifically designed with them in mind, many of them were able to enjoy using it. This means that GameTrack could potentially have an impact outside of the user pool that it was designed for, and can be enjoyed by many more people, be that people playing games for the first time and looking for a more hands-on experience with controlling, or people who are very experienced and want to try out a new way of controlling their favourite games.

We designed GameTrack in the way that we did, because, as shown in [25], people with Parkinson's Disease are unable to control small, precise movements, and are generally unable to move their arms in a smooth motion. For this reason, we designed GameTrack to exclusively register large, jerky movements as input, making it more likely for our user pool to be able to use. In order to be able to perfect the design, we will need to incorporate participatory design, collaborating with members of the intended user pool. This will mean that they will be able to have an active part in testing and design, making it the best product it can be for the purpose it is intended for. In its current state, we are unsure whether GameTrack is fully usable for the intended user pool, since no testing has been able to be conducted with them as participants. Traditional trackers that are used for games tend to do full body pose estimation[20], which is unnecessary for GameTrack, and can lead to more inaccuracies since it is trying to track the entire body over just the hand.

We set out to create a simple-to-use tracker that costs as little money as possible, to make it as accessible as it can possibly be. The only expenses needed for this is to buy a webcam,

for which any webcam would work.

GameTrack was easy to learn for some, and difficult to get used to for others, leading to a clear disparity in accuracy of inputs (see figures 5.1, 5.2).

Despite this, many participants greatly enjoyed using it, and the aim of playing any game is to have fun, so we believe it was a success in that regard. Also, almost every participant was able to improve their accuracy of input between levels 1 and 2 (see figures 5.1, 5.2), showing that it is very possible to learn and effectively use this control scheme, in a relatively short period of time.

7.2 Future Work

There is still work to be done with GameTrack, and more testing will be required with participants of the intended user pool, to elevate this project further than a proof of concept. When participants were asked about their thoughts after using GameTrack, some said that they found it difficult to control their precise left and right movement using just their hands, and found it difficult and less responsive to stop. This is because, to quickly stop in-game, it requires two quick inputs: to move in the opposite direction, then to let go of that direction. One way that this slight lack of control can be mitigated, is to use face tracking in conjunction with the hand tracker to control left and right movement, while reserving the hand inputs for one-time inputs like jumping or crouching.

Another thing that can be done is to add more customisation for the player, by offering more control styles so they can use the one that feels most comfortable to them. If they are unable to get used to the control scheme presented in this paper, we could offer another one that they may find easier to use. For example, have the user place their hand on a certain part of the screen to correspond to a specific input. So, for example, instead of needing to input a large, jerky motion upwards to jump, they just need to have their hand in the top right corner of the camera feed to jump. If the area required to input a specific command is large enough, then the user will not need to worry about not being able to control small movements, as their hand will still be in the area. It would also be possible to overlay the camera feed, as we have done for the landmarks of the hand, with the areas and their inputs to reduce the risk of the user forgetting which inputs perform which command. This type of control scheme would be good for people with conditions such as RSI, since not moving their wrists and fingers too much is ideal.

There were three initial ideas for the control scheme, being: The control scheme presented in this project, holding your hand in a specific position on the screen, and using gesture recognition to correspond to different inputs. The reason the first scheme was chosen was because we believed it would be the most natural control scheme for the user pool. However, additional customisation is always preferable so that users are more likely to be able to make a control scheme that is natural for themselves, so implementing both of these other control schemes would go a long way to making GameTrack usable by anyone.

Given that some participants found the default control scheme unnatural to use, another test could be to let participants play the game and figure out the best control scheme for them, and use the majority of peoples' layout as the new default going forward. Similar to this issue was the fact that some people, when panicked, would forget the controls and start moving their hands erratically, inflating their number of attempted inputs that did not register. One way we could fix this is by improving the overlay on the camera stream to

include a Heads-Up Display (HUD) that shows all of the controls and which actions perform them, for example, showing a '+' symbol on each side of the screen representing each hand, and the controls on each side of the +, representing up, down, left, or right.

In summary, the first piece of future work that will be required is further testing, allowing participants to have more time with GameTrack to see if they continue to improve their accuracy. Specifically, we want to be able to test with participants that are part of our intended user pool to see if the trend of improvement continues with them, and to see if it is more difficult for them to use the controller or to use GameTrack. Other than this, more customisation is preferable, possibly offering additional control schemes, allowing a wider range of people to use it.

7.3 Final Discussion

After more thorough testing with our participants, and testing with participants that have the movement conditions that this system is designed for, we will be able to gain more insight into how well it works for the user pool, and we will be able to see more easily what specifically needs improving or changing within the system.

Despite not being a part of the intended user pool, our participants were able to improve their accuracy of input using GameTrack. This shows that it is possible for many demographics to use this effectively and, with added customisation to the control scheme, like the option of different control styles, this demographic could increase even more, allowing even more people to enjoy a new way of controlling their favourite games.

In conclusion, we set out to create a hand tracker that can be used by people with movement conditions to play video games. It is hard to say whether that has been fully achieved without more thorough testing, but trends as shown in chapters 5 and 6 show that it is very possible that many people will be able to learn how to use GameTrack in a short period of time.

We are extremely proud of the work that has gone into this project and we believe we have been successful in our aims to create a usable hand tracker to play Sonic the Hedgehog 2 that many people will be able to enjoy.

(Word count: 10222)

Bibliography

[1] [Online]. Available: https://matplotlib.org/.

[2] [Online]. Available: https://app.diagrams.net/.

[3] [Online]. Available: https://www.statista.com/topics/868/video-games/.

[4] [Online]. Available: https://direct.playstation.com/en-us/buy-accessories/access-controller?smcid=pdc:us-en:web-pdc-accessories-access-controller:buttonblock-buy-now.

[5] [Online]. Available: https://www.microsoft.com/en-us/d/xbox-adaptive-controller/8nsdbhz1n3d8.

[6] [Online]. Available: https://www.amazon.co.uk/s?k=depth+camera&crid=32OXACRHXF3Y2&sprefix=depth+camera%2Caps%2C84&ref=nb_sb_noss_1.

[7] [Online]. Available: https://www.amazon.co.uk/Streaming-Microphone-Youtube-Studying-Conference/dp/B09BQQV6K3?crid=1X2HAN9GX7JQA&dib=eyJ2IjoiMSJ9.IhhORSzASA1RQ6f4LrP4JccsBLt_u6mSWDda4Y5YPWnLqE6_oegiFpGAZdHwJFg10cXJAHn5ym9olnLi98Qbvfo5wOhROb2uZ3diA3rnJnDLwNULHOoJOTUtyvXOmzqGKovpGaJZuDtLZOA5Ejui5rLmGWW_OOdGVR8fIS-XX9Sz3CRwI647s.xT7ic-xxKZ60fQLzBpweDBNZ-l1IShh5-5LKEBh2wX0&dib_tag=se&keywords=webcam&qid=1728308976&sprefix=webcam%2Caps%2C186&sr=8-3.

[8] [Online]. Available: https://opencv.org/.

[9] [Online]. Available: https://docs.python.org/3/library/tkinter.html.

[10] [Online]. Available: https://www.beatsaber.com/.

[11] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "Blazeface: Sub-millisecond neural face detection on mobile gpus," *arXiv preprint arXiv:1907.05047*, 2019.

[12] S. Gao, C. Zhou, C. Ma, X. Wang, and J. Yuan, "Aiatrack: Attention in attention for transformer visual tracking," in *European conference on computer vision*, Springer, 2022, pp. 146–164.

[13] P. Kohli and J. Shotton, "Key developments in human pose estimation for kinect," *Consumer depth cameras for computer vision: Research topics and applications*, pp. 63–70, 2013.

[14] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[15] J. R. Pansare, S. H. Gawande, and M. Ingle, "Real-time static hand gesture recognition for american sign language (asl) in complex background," *Journal of Signal and Information Processing*, vol. 3, no. 3, pp. 364–367, 2012.

[16] *Pydirectinput 1.0.4*. [Online]. Available: https://pypi.org/project/PyDirectInput/.

[17] J. Redmon, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[19] T. Sharp, C. Keskin, D. Robertson, *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 2015, pp. 3633–3642.

[20] J. Shotton, A. Fitzgibbon, M. Cook, *et al.*, "Real-time human pose recognition in parts from single depth images," in *CVPR 2011*, Ieee, 2011, pp. 1297–1304.

[21] L. J. S. Silva, D. L. S. da Silva, A. B. Raposo, L. Velho, and H. C. V. Lopes, "Tensorpose: Real-time pose estimation for interactive applications," *Computers & Graphics*, vol. 85, pp. 1–14, 2019.

[22] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, pp. 154–171, 2013.

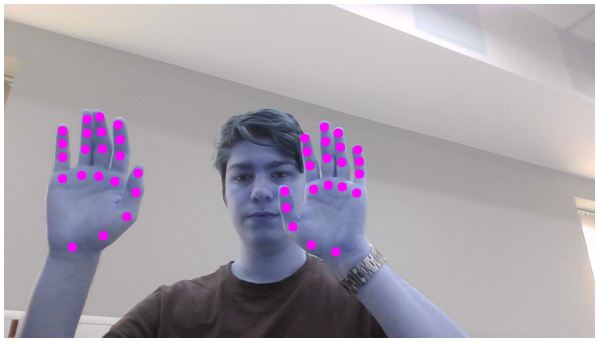
[23] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] F. Zhang, V. Bazarevsky, A. Vakunov, *et al.*, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[25] Y. Zhou, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "The measurement and analysis of parkinsonian hand tremor," in *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, IEEE, 2016, pp. 414–417.

Appendix A

Input Diagrams



(a) Left input on the camera stream (Moving the right hand to the left)

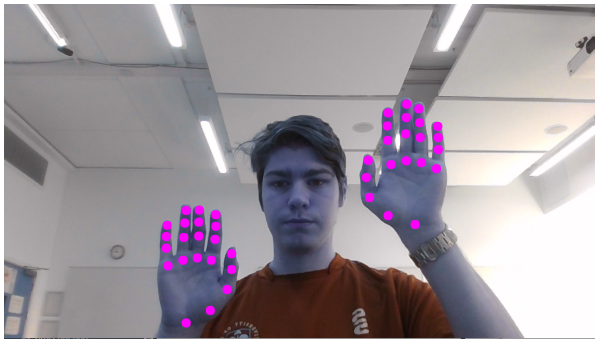
(b) Left input in-game

Figure A.1: Left input on the camera stream and in-game, causing Sonic to run to the left

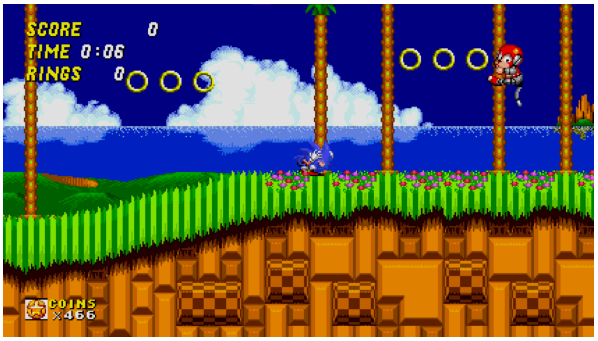
(a) Jump input on the camera stream (Moving the right hand upwards)

(b) Jump input in-game

Figure A.2: Jump input on the camera stream and in-game, causing Sonic to jump into the air

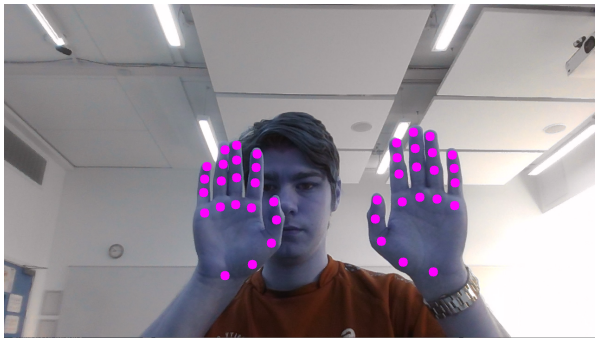


(a) Crouch input on the camera stream (Moving the left hand downwards)

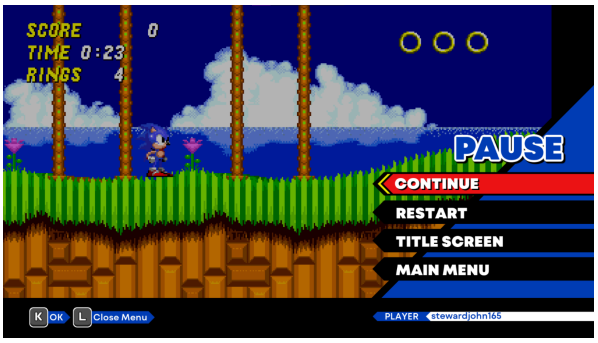


(b) Crouch input in-game

Figure A.3: Crouch input on the camera stream and in-game, causing Sonic to crouch and look down

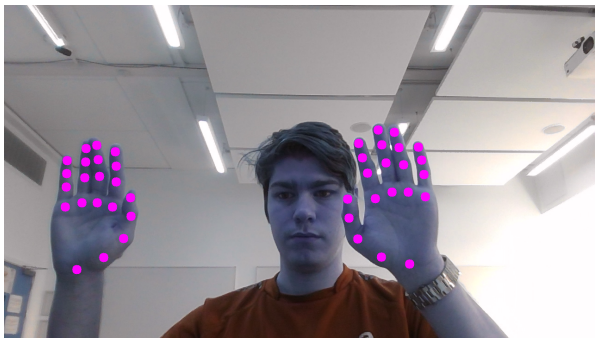


(a) Pause input on the camera stream (Moving the left hand to the right)



(b) Pause input in-game

Figure A.4: Pause input on the camera stream and in-game, pausing the game

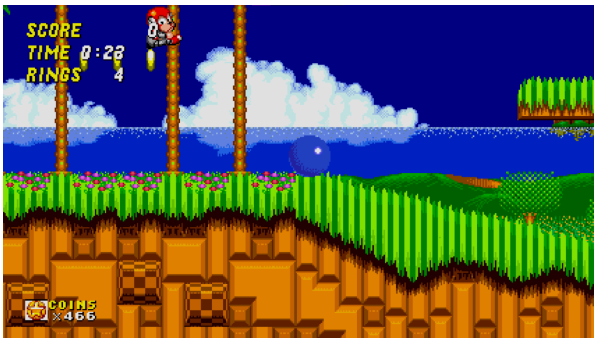


(a) Stop input on the camera stream (Moving the left hand to the left)

(b) Stop input in-game

Figure A.5: Stop input on the camera stream and in-game, causing Sonic to stop moving

(a) Un-crouch input on the camera stream (Moving the left hand upwards)



(b) Un-crouch input in-game, completing a spin dash

Figure A.6: Un-crouch input on the camera stream and in-game, completing a spin dash