University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme / specialisation: Master's in engineering / Robot Technology and Signal Processing	Spring semester 2023 Open
Author(s): Ådne Hult Karlson and Stian Wiik Berg	
Faculty supervisor: Nejm Saadallah Supervisor(s): Nejm Saadallah and Ketil Oppedal	
Thesis title: Design a continuous feedback system utilizing velocity-based training and posture estimation improving the efficiency and performance of a squat	
Credits (ECTS): 60 ($2 \cdot 30$)	
Keywords: Velocity-based training, posture estimation, Mediapipe, squat, application	Pages: 91 + appendix: 3 Stavanger 15. June 2023

Abstract

Staying in shape is important for physical and mental health. Adding squats to a regular workout routine is beneficial, but performing them correctly and determining the right number of repetitions can be challenging. A feedback system utilizing body movement identification methods can assist in performing squats correctly and efficiently. The system we have developed incorporates velocity-based training principles and posture estimation using Mediapipe's pose landmarks. It provides feedback on squat depth, neck angle, and improvement techniques.

Our system functions as a visual feedback application. It was tested for velocity-based training, posture estimation, and feedback and interface functionality. A velocity loss of 10-20% indicates the need for a break during squats, where comparing current squat velocity to the initial squat in a set notifies the user of the velocity loss. The maximum velocity of the upward motion of the squat was calculated with a standard deviation which translates to a potential error of 0.008% in velocity comparison. The depth of the squat is determined by the femur's parallelism to the ground, with a deviation corresponding to a 10% error. A neck angle within a determined range indicates good posture.

The application improves squatting performance, advises on breaks, and offers user-friendly features like gesture control, however, the gesture control does not work as intended and requires further work. Future work can expand posture feedback to include additional requirements like stance, knee position, back angle, and bar placement. If adapted as a phone application, it can serve as a versatile and user-friendly exercise aid, providing valuable information on correct exercise performance.

The solution for this thesis could be found in the Github repository link: `https://github.com/MAS-2023-Nejm/body-movement-identification`

Acknowledgments

We would like to thank the Department of Electrical Engineering and Computer Science at the University of Stavanger to provide with a room we could work as well as equipment used for testing and research.

Our supervisors Nejm Saadallah and Ketil Oppedal gave valuable feedback on the creative process, research, and writing process, and proved valuable for this thesis. Thank you for always being available and going to the lengths required for us finishing this project.

At last, we would like to thank our fellow students Emil Wiik Larsen and Vebjørn Njåtun Krøyer for letting us record squats and use the data acquired in this project.

Abbreviations

1D	1 Dimension
1RM	1 Repetition Maximum
2D	2 Dimensions
3D	3 Dimensions
App	Application
BGR	Blue, Green, Red
CPU	Central Processing Unit
FPS	Frames Per Second
GPU	Graphics Processing Unit
IMU	Inertia Measurement Unit
PBT	Percentage-Based Training
RGB	Red, Green, Blue
ROI	Region Of Interest
SD	Standard Deviation
VBT	Velocity-Based Training
WM	Whole Movement
YOLO	You Only Look Once

Contents

1 Introduction **1**

1.1 Motivation 2

1.2 Existing solutions 3

1.3 Problem definition 5

1.4 Layout 6

2 Background **7**

2.1 The theory of velocity-based training 7

2.2 The theory behind the squat 9

2.3 Human pose estimation 11

2.4 Gesture recognition 13

2.5 Finite difference method 14

2.6 Performance evaluation method 15

2.7 The geometric calculations 16

2.7.1 Distance calculation 16

2.7.2 Angle calculation 17

3 Solution Approach **19**

3.1 Input 20

3.2 Human pose estimation 21

3.3 Collection of landmarks 22

3.3.1 Defining a coordinate system 22

3.4 Velocity-based training and posture 24

3.4.1 Velocity-based training 24

3.4.2 Posture estimation 28

3.5 The application 31

3.5.1 Gesture control 32

3.5.2 User interface 32

3.5.3 Scoring system 34

4 Testing and results **36**

4.1 Setup 36

4.2 Testing of velocity 37

4.2.1 Testing the equation for the average velocity 37

4.2.2 Testing the velocity-based training 42

4.3 Posture measurements 49
4.3.1 Testing the depth function 49
4.3.2 Estimating a good neck angle 51
4.3.3 Testing neck angle with simulated input 52
4.3.4 Testing errors of mono camera-based measurements . 54
4.4 Testing of the application 60
4.4.1 Testing the gesture control 60
4.4.2 User interface and scoring system 62

5 Discussion and future work **69**

5.1 The input 70
5.2 Human pose estimation 71
5.3 Velocity-based training 73
5.3.1 Finding the region of interest 73
5.3.2 Calculating the velocity 74
5.4 Posture measurements 75
5.4.1 The theory behind the squat 75
5.4.2 The depth estimation of the squat 76
5.4.3 The neck estimation 77
5.4.4 The missing posture estimations 78

5.5 The application 79

5.5.1 The continuous feedback system 79

5.5.2 The gesture control 80

5.6 Further future work 81

5.6.1 Instructions to user 81

5.6.2 Accessibility 82

5.6.3 Depth perception 82

5.6.4 Velocity zones 83

5.6.5 Different methods to identify the squat 84

5.6.6 Identify different movements 84

6 Conclusion **86**

Bibliography **91**

Appendix **91**

A Poster **92**

B Gesture table **94**

Chapter 1

Introduction

This thesis utilizes machine learning models to develop an approach for identifying body movements, which is subsequently applied to design a dedicated software for assisting individuals in performing squats. The software provides guidance on the squat's intensity and posture, thereby helping the user determine when to take a break and how to execute the squat correctly.

This chapter will cover the motivation for the thesis, existing solutions that use similar methods, the problem definition as well as the layout of the report.

1.1 Motivation

Staying in shape and exercising is important for a large group of people. In a study done by *Statistics Norway*, there was found that up to approximately 57% of people in Norway above the age of 16 years exercise several times a week. [1] One exercise that is often overlooked or misunderstood is the squat, which may seem basic or unnecessary. However, this exercise is actually incredibly powerful and beneficial when performed correctly, and is referred to as the most important exercise in the training arsenal [2], making the squat an essential addition to any regular workout routine. Performing the squat correctly, however, could prove challenging, as a bad performance could lead to poor progression, and in the worst-case scenario lead to injuries. As a result, it may be discouraging to attempt this particular exercise.

There is a plethora of training materials available online, but many of them provide conflicting advice without any academic foundation. This could both be frustrating for the people that want to perform a given exercise, but could also be overwhelming for people that want to start exercising.

Having an application easily available that could help with both how to perform the exercise and give an insight into the intensity required, could help with these goals. This would mean that it would be easier to begin training, and motivational to know that you are performing the exercise correctly. But how should the exercise be performed, and what is the right intensity?

In addition, developing an effective body movement identification approach has the potential to pave the way for a diverse range of applications. As smartphones and comparable devices become increasingly ubiquitous, a user-friendly and adaptable application leveraging this approach could emerge as a feasible possibility.

Some potential usages could be in virtual and augmented reality training programs for athletes and dancers, allowing them to see and correct their movements with greater accuracy. In the medical field, body movement models could potentially be used to design tailor-made rehabilitation programs for patients recovering from injuries or surgeries that affect their mo-

bility. Additionally, they could be used to monitor and track the progress of individuals with movement disorders like Parkinson's disease, allowing for earlier intervention and more effective treatment. [3]

1.2 Existing solutions

There exist multiple products or solutions that implement velocity-based training (VBT) or that examine the posture of a workout, some of these are *Alphatek's AlphaPWR*[4], *Viruve's* VBT device[5], Metric's *MetricVBT* app[6], and an example from Github regarding posture estimation[7].

Alphatek is a company that makes an equipment called *AlphaPWR*. This is a platform that the user stands on while they are exercising, and gives the user feedback on a screen after each repetition. This helps the user keep track of their tempo and intensity to optimize the training. The platform measures the users ground reaction force and gives feedback on how many repetitions would yield the best result based on velocity loss. This process is called *velocity based training*.[4]

The *AlphaPWR* could be found at multiple fitness centers, and therefore require the user to have a membership at one of these centers. This could be quite expensive over time, making it both harder for people to start exercising and inconvenient to use. The platform has very specific functionalities, including balance, jump, pull, and squat. This offers the end user a relatively small variation of exercises. In addition, the *AlphaPWR* gives no indication of how well the exercise is performed, and how to improve.

Virtuve has another example of a device that utilizes VBT. The device employs accelerometers or linear encoders and consists of a box with a reel attached to it. This device could be placed on the floor and attached to an exercise bar. When lifting the bar, the reel gets pulled out from the box, allowing for the measurement of velocity. This allows for real-time measurements, where *Virtuve* provides the user with the velocity and accompanied fatigue data.[5]

An early development of an application utilizing the principles from velocity-based training is the Metric's *MetricVBT app*. [6] With the use of a smart-

phone camera to capture videos, the Metric team was able to count the number of repetitions of the exercises deadlift, front squat, and bench press and calculate the velocity needed for their purpose. This velocity data was analyzed and compared to other exercise data gathered by Metric, and the user could then get feedback on how their exercise is compared to the standard set by Metric. This solution utilizes velocity-based training to compare exercises, making it a potentially valuable tool. It's designed as a smartphone application, which would make it user-friendly, affordable, and portable.

The solution proposed by Metric is however lacking in providing information on how to perform the exercise and providing accompanying feedback on this. There is also a lack of information regarding real-time indication of how many repetitions the user should do, and when to take a break.

An example of how to determine the correct posture for a squat can be found on a Github repository. [7] The solution from the example is designed to monitor the number of repetitions, as well as the angles of the knee-joint and hip-joint. It creates a virtual skeleton of the person performing the squat, enabling the user or fitness instructor to view real-time repetitions and angles. While the solution appears to effectively track real-time data, it does not provide any feedback to the user on how to improve. Instead, the user or instructor must interpret the data themselves.

1.3 Problem definition

Using body movement identification methods to define a continuous feedback system to help the user to exercise correctly and efficiently.

This includes;

- Monitor the users' velocity and provide feedback on when the user needs to take a break.
- Monitor the user's posture to help the user lift correctly.
- Deliver a score to the user on how well the exercise was carried out.
- Give feedback on the exercise, and provide useful information on how to improve.

The squat has been the main focus of this thesis.

1.4 Layout

This project is composed of five chapters; The background, the solution approach, testing and results, discussion and future work, and the conclusion.

The background will cover the necessary theoretical information for comprehending the thesis work and decisions. The theory of velocity-based training and squatting form the core of this chapter. The former explains how to identify the right time for the user to take a break, while the latter teaches how to execute a proper squat. Additionally, the chapter covers the theory of human pose estimation, which involves locating a person's skeletal structure using Mediapipe pose landmarks and gesture recognition. It also covers a method on how to find the velocity, and how to evaluate the performance. The mathematical formulas used in the estimation process are also explained in this chapter.

In the solution approach, the methods utilized in the thesis will be outlined along with an explanation of how the achieved results were obtained. The approach flow is described, starting from the input to the system and ending with the application.

In the chapter regarding testing and results, the outcomes of the implementation described in the solution approach will be presented. Multiple tests were conducted to test the accuracy and reliability of the implementation of velocity-based training and posture estimation. Furthermore, the application was tested to ensure that the feedback system and user interface functioned as intended.

The discussion and future work goes into depth on the limitations and assumptions made, as well as some potential reasons why some of the results worked or did not work as intended. The chapter also presents suggestions for future work, outlining ways to enhance and broaden the existing implementations.

Finally, a conclusion is drawn in which the work is summarized and an explanation is given regarding the effectiveness of the solution made.

Chapter 2

Background

2.1 The theory of velocity-based training

The method of velocity-based training is quite effective in determining the right exercise intensity for a user. It involves analyzing velocity data to determine when a user should take a break between sets. A linear position transducer is typically used in VBT to measure the user's velocity [8], which then helps decide when an exercise set is completed, and the user should take a break. This method has the potential to enhance both the exercise experience and gains of the user.

When using the *repetition until failure method* for training, there is a risk of errors due to fatigue and mechanical stress. [9][10] Instead, an alternative method is to determine the exercise intensity as a percentage and adjust the number of repetitions accordingly. This could be achieved through percentage-based training (PBT) or VBT.

The level of exercise intensity is widely recognized as the crucial factor that affects strength, and it is commonly associated with the relative load, also known as the percentage of one repetition maximum (1RM). [8] This approach is referred to as PBT in this section.

When it comes to prescribing training loads for performance improvement,

VBT could be a more accurate alternative to traditional PBT, both in terms of increasing specific and general performance. [11] Traditional PBT has several limitations that VBT overcomes. One limitation of PBT is the need for potentially risky tests which could lead to injury if not performed correctly. [8] Additionally, changes in 1RM could fluctuate on a daily basis due to factors such as nutrition, fatigue, and biological variability. [8][11] On the other hand, VBT is believed to better reflect changes in the individual load-velocity profile compared to PBT. [8]

An implementation of a VBT training program is recommended to be implemented in different ways, for instance; [11]

- Using velocity zones as a part of separate or combined training programs.
- Applying velocity losses of 10-20%.

It's important to emphasize that receiving feedback instantly is crucial, regardless of the method used for implementation.

It appears that utilizing velocity zones is the most effective way to use VBT. To achieve this, it is necessary to determine the load/velocity profile for an individual and use this data to establish an optimal velocity zone. To accomplish this, the individual's 1RM must be determined, and they must perform repetitions at various predetermined relative or absolute loads. These loads range from 30-85% of the 1RM, and the velocities measured during the repetitions serve as the basis for the individual's velocity zone. [8]

Applying the velocity losses of 10-20% could help induce neuromuscular adaptations and reduce neuromuscular fatigue. [11] Velocity loss would mean comparing the current velocity to the first movement of the set. If the loss reaches 10-20%, it's advised for the user to take a break before starting the next set. This approach is general and doesn't require any profiling.

The topic of velocity zones is often discussed in relation to athletes and their training optimization. [8] However, this thesis aims to create a user-friendly application for a broader audience to facilitate exercise, and the

applied velocity loss of 10-20% achieves this goal. It is worth noting that the effectiveness of VBT is still a subject of debate and ongoing verification. [12][13]

2.2 The theory behind the squat

It is crucial to perform exercises correctly to reap maximum benefits and avoid the risk of injury. Any incorrect movement could be harmful when repeated, so it's essential to pay attention to proper form. The squat is a powerful exercise that could reap many benefits, however, it is important to perform it correctly. The theory behind the squat will determine the basis for the user's posture score and subsequent feedback in the application.

According to Mark Rippetoe's book, Starting Strength, the squat is the most important exercise in the training arsenal. [2, p. 7] He highlights six crucial elements to consider for this project: depth, knee position, stance, eye gaze, back angle, hip drive, and bar placement.

When a squat is not performed deeply enough, Rippetoe refers to it as a *partial squat*. The partial squat could cause strain on the knees and quadriceps, without effectively targeting the glutes, adductors, or hamstrings. To avoid this issue, it is recommended to perform a deep squat, which will allow for the proper execution of the exercise. Examples of both the deep squat and partial squat could be seen in the accompanying figure, Figure 2.1.

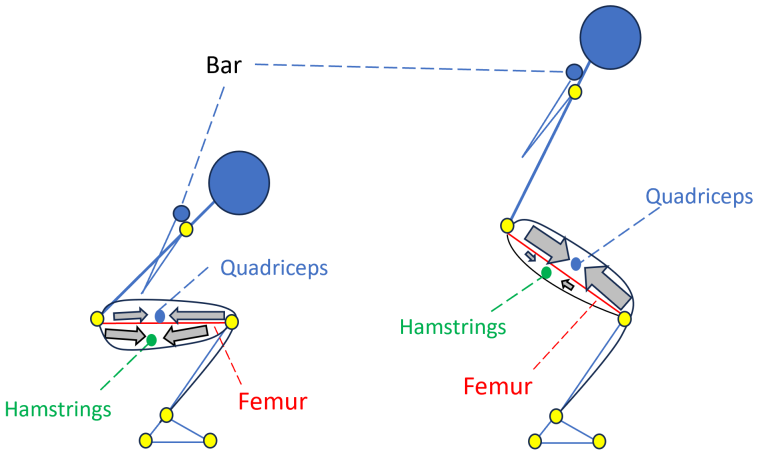


Figure 2.1: The full squat. Illustration of correct depth on the left, and a partial squat on the right.

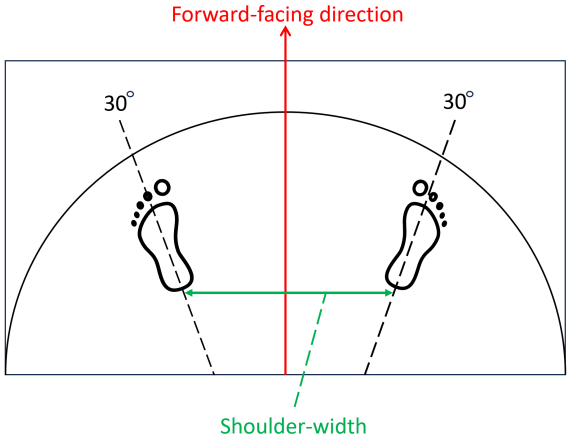


Figure 2.2: Correct stance.

The stance and the knee position are highly correlated. With the feet flat on the ground, the heels should be shoulder-width apart. The feet point outward at an angle of about 30° of the forward-facing direction of the user, with the femurs parallel to the feet. The knees are just a little forward of the toes. The back is at an angle, about 45°, that would place the bar over the middle foot. Furthermore, the eye gaze plays an important part in driving the hips and preventing unwanted stress from the bar. These guidelines

could be seen in the accompanying Figure 2.1 and Figure 2.2. Additionally, the eyes should be fixed at a position on the floor approximately 1.5 meters in front of the feet, keeping the neck at an angle corresponding to the angle of the back. [2, p. 20-23]

2.3 Human pose estimation

When utilizing velocity-based training and analyzing a squat, it's crucial to have a method for collecting data on a person's position. Human pose estimation is a viable solution as it employs machine learning models and computer vision to estimate the position of a human's joints. [14]

When it comes to human pose estimation, there are numerous options to consider. Thankfully, there are many pre-trained open-source models that could be utilized for this purpose, making it a wise choice to use one of them.

Google has developed the Mediapipe framework pipeline, which is designed to handle machine learning and computer vision tasks with precision in object detection. Thanks to its extensive training data, it is an ideal platform for pose estimation. [15]

The Mediapipe pose library is capable of identifying 33 body parts called landmarks, as shown in Figure 2.3. This makes it a great choice for image-based exercise applications. [16]

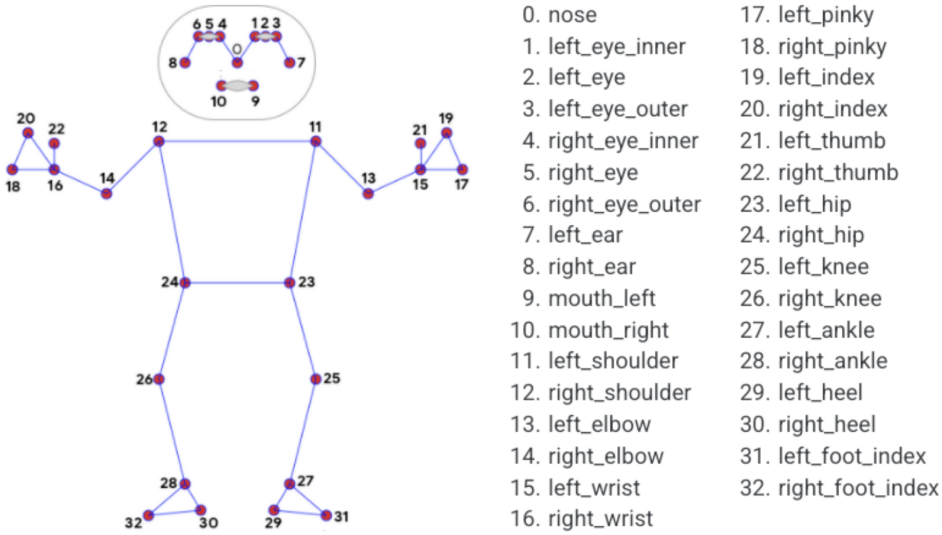


Figure 2.3: Mediapipe pose landmarks [17, Fig 4.].

Mediapipe uses a bundle of models for human pose estimation, one where it detects the presence of a human and another that finds the landmarks shown in Figure 2.3. This bundle utilizes a convoluted neural network with is supposedly similar to the MobileNetV2[18] which is optimized for real-time applications such as fitness models.[16] This makes it ideal to use for this project as the goal is to make a squatting application.

Aside from recognizing the body parts depicted in Figure 2.3, Mediapipe also has the capability to identify intricate hand parts estimation. A single hand is depicted as 21 3D data point landmarks, as illustrated in Figure 2.4. [19]

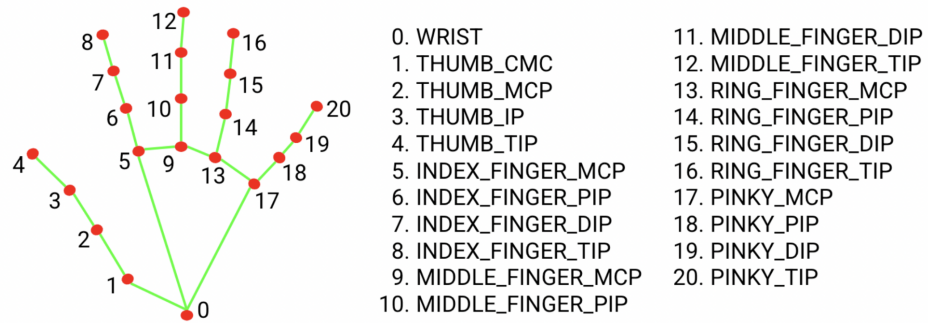


Figure 2.4: Mediapipe hand landmarks.[19]

2.4 Gesture recognition

When developing an application, it is important to prioritize a user-friendly interface. The proposed application in this project, with a camera-based input, requires the user to be at a distance from the device during use. Therefore, it is recommended to integrate a method for controlling the application without physical proximity.

Gesture recognition could be an effective solution for this purpose, as it has been proven to be increasingly prevalent across various industries as a means of controlling different products. One example of this is utilizing it in automobiles to control the infotainment system. BMW is one of the automobile manufacturers that use this technology to control multiple functions in their infotainment system, like, turning up the volume on the music, accepting/rejecting calls, pausing, and changing the song playing. when using gesture recognition to control the output of a function, it is referred to as gesture control.[20]

To implement this, the *hand pose landmarks* was utilized in combination with a pre-trained model for hand gesture recognition found on Techvidvan's website.[21] with this model and a combination of Mediapipe, TensorFlow[22], and OpenCv[23]. The wanted hand gestures were found.

Figure 2.5 illustrates all the possible gestures possible to detect using Tensorflow and the trained model found on Techvidvan's website.

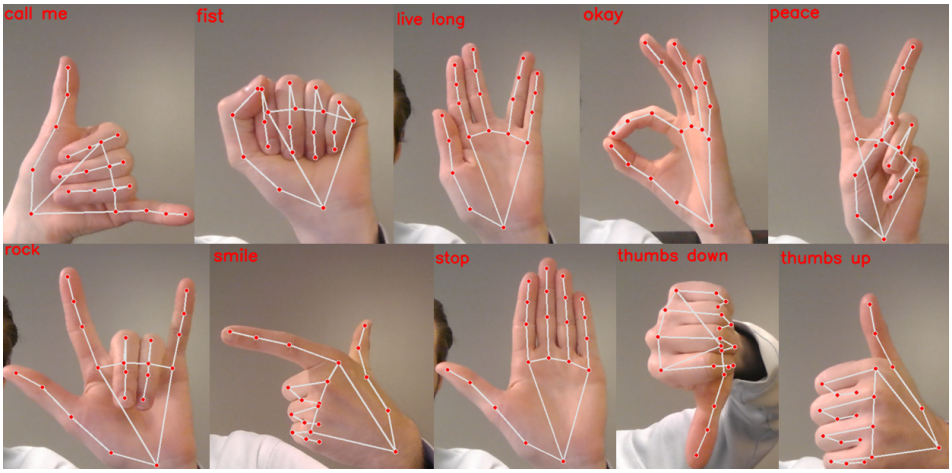


Figure 2.5: Various gestures recognized using TensorFlow.

2.5 Finite difference method

In order to successfully implement the theory of velocity-based training, access to velocity is crucial. Real-time velocity information could be obtained by calculating the derivative of the position in real time, using the backward difference formula as a helpful tool for approximating this derivative. The formula takes into account current and previous positions to determine the velocity. The formula for the backward difference is shown in Equation(2.1). [24]

$$f'(x) = \lim_{h \to 0} \frac{f(x) - f(x-h)}{h} \tag{2.1}$$

In Figure 2.6 the approximation is illustrated. Where the $f(x)$ represents the current position, $f(x-h)$ represents the previous position, and h represents the distance or time between the mentioned parameters. The blue line is an arbitrary equation only used for illustrative purposes.

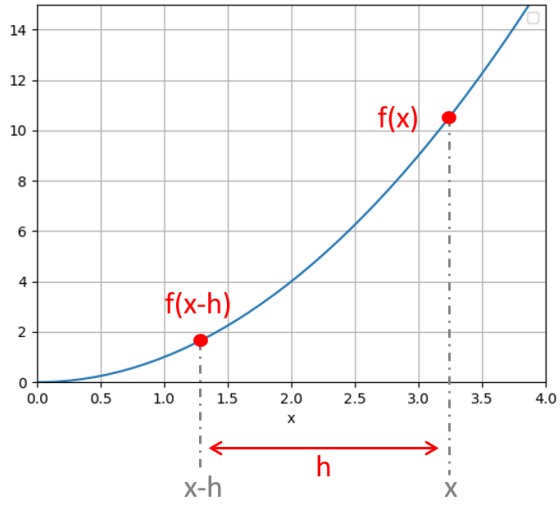


Figure 2.6: Illustration of the backward difference formula.

2.6 Performance evaluation method

When performing tests in this project, some methods are needed to evaluate the performance. A much-used method in this project was to look at the standard deviation (SD) as shown in Equation (2.2), for the data gathered.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{2.2}$$

where:

- σ represents the standard deviation.
- N is the total number of data points.
- x_i represents each individual data point in the dataset.
- μ denotes the mean of the dataset.

Calculating the SD could be helpful in determining the spread of a data set around the mean. It could also indicate whether the mean accurately represents the data. [25] Additionally, the SD provides the same unit as the original data used in the calculation.

2.7 The geometric calculations

In order to find and rate the elements mentioned in Chapter 2.2 *The theory behind the squat*, the pose landmarks could be used. This method is based on the assumption that body movements could be analyzed by examining the joints and the links between them. By measuring the distances and angles of these links, one could estimate the posture. This requires knowledge of mathematical concepts such as vector theory, specifically in calculating distances and angles using the x, y, and z axes for both three-dimensional(3D) and two-dimensional(2D) calculations. The following section details the mathematical theory behind these calculations.

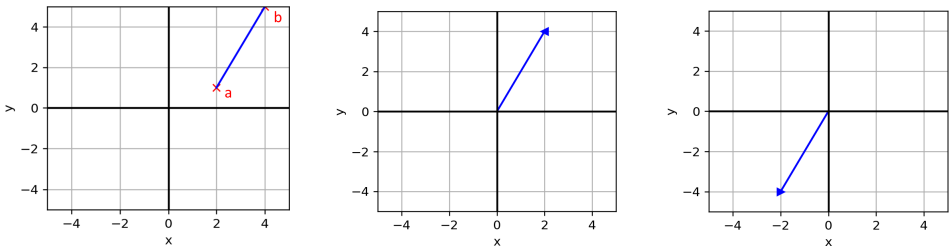
2.7.1 Distance calculation

To determine the distance between two coordinate points in a room, a straightforward method is to consider the relevant axes, such as the x-axes, and subtract the smaller x value from the larger one.

$$Distance = x_1 \ - \ x_2 \tag{2.3}$$

The current approach only functions when measuring distance in a straight horizontal line. However, for the code to be universally applicable and adaptable to different dimensions and directions, a solution must be found that doesn't depend on the orientation of the coordinates.

This is where vector calculation comes in handy. Vector calculation ensures that the input order of coordinates and the vector's dimension, whether it's 2D or 3D, does not matter. Figure 2.7 illustrates how two 2D coordinate points could be transformed into a vector by subtracting one from the other.



(a) a line between point a and b.

(b) b minus a.

(c) a minus b.

Figure 2.7: Three graphs illustrating vector calculations.

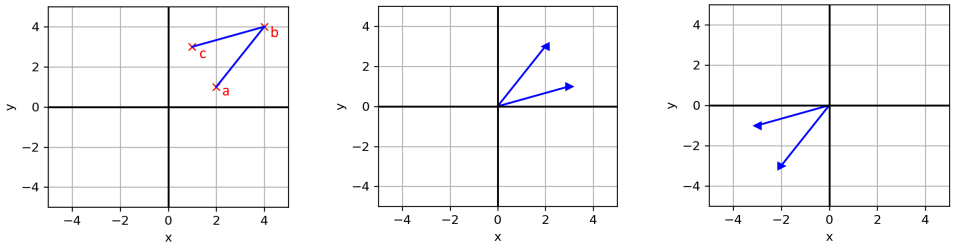
As shown in Figure 2.7, the length of the vector remains constant regardless of the coordinates being subtracted. By converting the points to vector form, the distance could be calculated by determining the vector norm. One possible approach to achieve this task is to utilize the *numpy.linalg.norm()* function, which is included in the Numpy library [26]. In the example shown in Figure 2.7, the length could be calculated as 4.47 using Equation(2.7).

$$a = [2, 1] \tag{2.4}$$
$$b = [4, 5] \tag{2.5}$$
$$V = a - b = [-2, -4] \tag{2.6}$$
$$\mid V \mid = \sqrt{(-2)^2 + (-4)^2} = 4.47 \tag{2.7}$$

2.7.2 Angle calculation

It is important to have a universal estimate for calculations of various dimensions, as this would simplify the development of the application. Fortunately, this could be achieved by implementing vector solutions. For instance, in order to determine the angle between a person's thigh and leg, one would typically require the hip, knee, and ankle coordinate points. However, by simplifying these points to two vectors, as demonstrated in Chapter 2.7.1 *Distance calculation*, it becomes possible to calculate the angle between the vectors using the norm of each vector and the dot product between them.

Figure 2.8 illustrates how three points could be changed into two vectors in two different ways. Looking at Figure 2.8a, the outer points point a and point c and the connecting point in the middle point b are illustrated. In Figure 2.8b, the vectors are made by subtracting point a from point b, and subtracting point c from point b. Thereby creating two vectors pointing upward in the right top section of the plot. In Figure 2.8c, the opposite is done, subtracting point b from point a and subtracting point b from point c, thereby getting two vectors pointing in the opposite direction.



(a) Three positional points; a, b, and c.

(b) subtracting a and c from b.

(c) subtracting b from a and c.

Figure 2.8: Illustration of how points could be transformed to vector form.

Both plots, illustrated in Figure 2.8b and Figure 2.8c, share a common feature; the vectors they display form an identical angle. To determine this angle, the utilization of the dot product and norm of the vectors are used, as demonstrated in Equation(2.8). In Equation(2.8), θ represents the angle, V_1 and V_2 represent the vectors, and $|V_1|$ and $|V_1|$ represent the norms of the vectors.

$$\theta = \arccos\left(\frac{V_1 \cdot V_2}{|V_1||V_2|}\right) \tag{2.8}$$

Chapter 3

Solution Approach

The solution proposed in this thesis can be summarized by referring to the flowchart depicted in Figure 3.1. To collect data on body movement, an input is needed. This input is used to perform pose estimation and obtain the pose landmarks necessary for estimating the squat. The landmarks are then saved as position data for future reference. Position data serves as the basis for the calculations and estimations required to accomplish velocity-based training and posture estimation. The interface of the application provides the user with continuous feedback, including details on when to take breaks, tips on how to enhance their performance, and a scoring system.

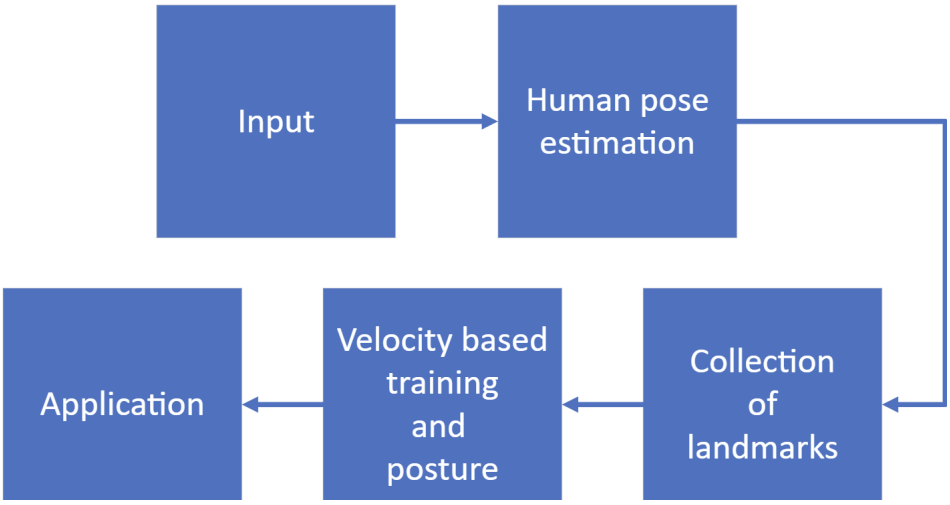


Figure 3.1: Flowchart of the proposed solution.

3.1 Input

The main goal of the input is to gather as much information about the movement as possible, in order to be able to calculate the velocity of the user and also to give a valid pose estimation. A camera solution was chosen as it's a practical and economical way to gather the required data for the system. The camera's adaptability allows it to acquire all the essential information for both the velocity-based training and posture estimation aspects of the task, meaning that only one input is required. This fact makes the application easy to use, and a user-friendly and convenient choice. As the proposed solution is made on a computer, the input corresponds to a mono-based-camera solution.

Another reason the camera was chosen as the input is to keep the application adaptable for smartphones, as this was seen as a natural future development when designing the application. By using a camera solution the adaptability of the application is preserved.

3.2 Human pose estimation

To process the input, the images are read using OpenCV[23] and then undergo a series of modifications. Firstly, the images are re-scaled to a smaller format in order to reduce unnecessary data from the image matrix. This step could be done as Mediapipe does not require a high resolution. Next, the images are reformatted from BGR (blue, green, red) to RGB (red, green, blue) as Mediapipe requires the RGB input.

The image undergoes processing through Mediapipe, which identifies and records all landmarks into a dictionary. These landmarks are then utilized to create the virtual skeleton, as shown in Figure 2.3,. The skeleton is not crucial or necessary for the computations performed in this project, however, it provides an excellent visual representation of the body's connections and could provide valuable feedback for the user to see if the landmarks are properly found. This process is then repeated for every new image. Mediapipe does not retain the landmarks once a new image arrives, so it becomes necessary to gather them for comparison calculations, such as velocity calculations.

3.3 Collection of landmarks

When utilizing Mediapipe, a landmark refers to the points of interest that construct the human skeleton, as depicted in Figure 2.3 from Chapter 2.3 *Human pose estimation*. To perform velocity calculations and pose estimations, these landmarks are gathered and saved.

A common issue that arises during the estimation of landmarks with Mediapipe is jitter. When performing pose estimation on a live video stream, the algorithm analyzes one image per frame, resulting in approximately 30 analyses per second. However, slight variations in the coordinates of seemingly similar images can cause jitter, which can make it more difficult to analyze the resulting data. While not an immediate concern, this issue can impact the accuracy of the output. The solution was to use a simple median filter. This filter could effectively smooth out the data, aligning changes with its data points for a more seamless result.

3.3.1 Defining a coordinate system

Defining the coordinate system is important as it sets the standard for how to understand and calculate the data. Mediapipe offers two main methods of finding the landmarks: *pose landmarks* and *pose world landmarks*. [17]

The *pose landmarks* provide normalized x- and y-coordinates in relation to the image's width and height, ranging from 0 to 1. For instance, the top left corner of an image would have a value of $[x = 0, y = 0]$ as illustrated by the right image in Figure 3.2, while the bottom right corner would have a value of $[x = 1, y = 1]$. Additionally, the z-coordinate represents the depth of the landmark, with the midpoint of the hip as the reference point. As the z-value decreases, the landmark gets closer to the camera.

The *pose world landmark* yields estimated real-world 3D coordinates in meters. The origin is set at the center of the hips and would change from frame to frame. This corresponds to the left image in Figure 3.2

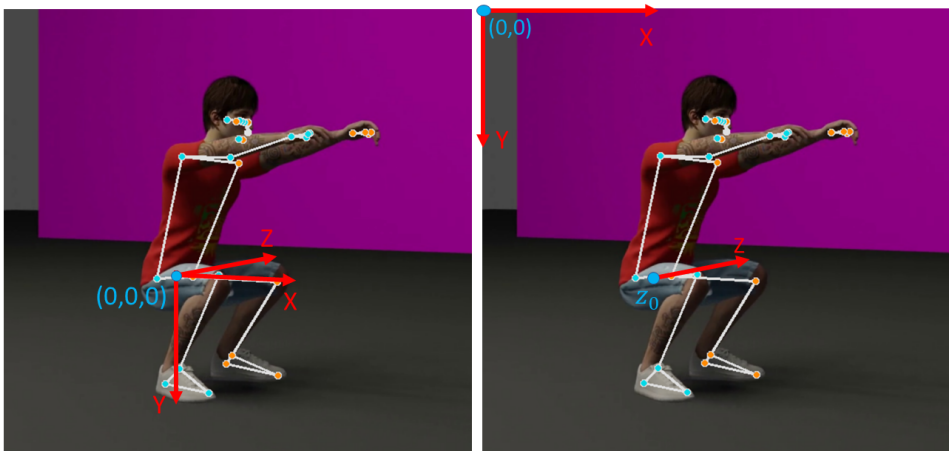


Figure 3.2: An illustration of the definition of axes.

The use of real-world 3D coordinates works great when angles and distances are calculated in the same frame. However, because of the potential change in the origin for each frame, this would not work as well when comparing data from different frames. For this reason, the *pose landmarks* is primarily used for the velocity calculations and the *pose world landmarks* is primarily used for the pose estimations. This is true for the rest of the thesis unless specifically mentioned otherwise.

3.4 Velocity-based training and posture

To ensure correct and efficient exercise, the squat's velocity and posture will be examined. Velocity will be monitored by analyzing the user's changes in speed for each squat, indicating when rest is needed due to fatigue. This information is based on Chapter 2.1 *The theory of velocity-based training*. Correct posture is crucial and involves observing the angles of various body parts in relation to each other. Proper lifting technique not only prevents injury but also maximizes benefits from the squat. This is based on Chapter 2.2 *The theory behind the squat*.

Using the correct data

Prior to conducting various calculations, it is crucial to have a clear understanding of the incoming data. Chapter 3.3.1 *Defining a coordinate system* explains the two distinct types of coordinates - one for the world and another for the image frame. Additionally, it is essential to determine the orientation of the camera in order for the app to adjust accordingly. To achieve this, a function was developed to identify which side of the person should be focused on. This was accomplished by analyzing the z-data from each side of the hip and selecting the smallest z-data, which indicates the closest proximity to the camera. In essence, when analyzing the user's position data, the point closest to the camera is always used in calculations.

3.4.1 Velocity-based training

The real-time position data is obtained by the collection of landmarks. It is used to analyze and determine the squatting direction, which is then compared to the previously determined direction. This helps to establish the region of interest (ROI). The peak velocity is then calculated within this region to determine the velocity of each squat. The flowchart in Figure 3.3. demonstrates this process. In this section, it is explained how the ROI is determined and how the velocity is calculated, as well as identifying the expected errors from using this method.

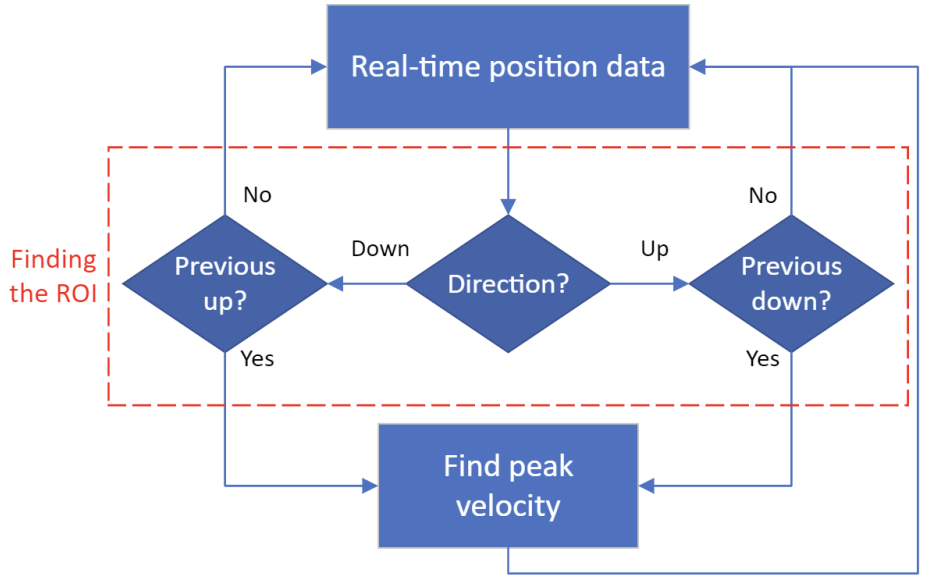


Figure 3.3: Flowchart illustrating how the ROI and the corresponding velocity is found.

Region of interest

According to the theory of *velocity-based training*, the ROI would correspond to the upwards thrust movement of the squat. By identifying when the user is thrusting upwards in the squat, the maximum velocity of that movement could be found. This could then be compared to the next thrust, and the next, and so on.

The distance from the shoulder down to the ankle of the person is referred to as the height of the person in this section, whereas the distance measured while the person is standing is referred to as the maximum height. This is illustrated in Figure 3.4.

By looking at the difference between the current height of the person and the height in the past, and looking at the sign, the direction of the squat could be determined. By trial and error, it was found that the difference between the current height and the height 10 frames in the past, gave acceptable results. An example of this is illustrated with the left and middle image in

Figure 3.4, with H_1 and H_2 respectively, representing the current heights of that frame. Here the current height H_2 would be compared with the last height H_1 and the direction would be determined to be an upward direction. Note that the difference in the images is not 10 frames in between, but exaggerated for illustration purposes.

In addition, to ensure that the detected difference was a squat and not noise or small insignificant movements in the shoulder, the maximum height of the person was used. By comparing the absolute value of the detected difference to a percentage of the maximum height, the detected ROI should work independently of the distance from the camera and the person doing the squats. By trial and error, 2% was found to give acceptable results. So if the detected difference is larger than 2% of the person's maximum height, the difference is considered large enough to be a part of the squat movement. The maximum height is illustrated in the right image of Figure 3.4.

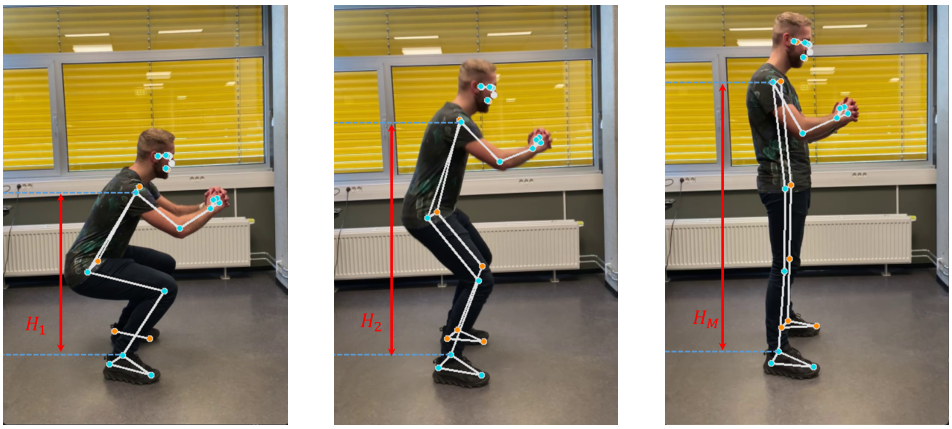


Figure 3.4: Three positions illustrating the upward direction, with the bottom position at the left and the maximum position at the right. Current height is illustrated as H_1 and H_2, and maximum height as H_M.

Finding the velocity

By looking at the position of the shoulder closest to the camera, the velocity could be calculated by first finding the distance between the current position and the last position in the xy-plane. With this distance the derivative could be found, resulting in the velocity. The utilization of the theory in Chapter 2.5 *Finite difference method* is used in order to find this derivative. By assuming linearity between the position points of interest used to calculate the velocity, the equation could be simplified to Equation(3.1).

$$v_{avg} = \frac{\Delta x}{\Delta t} \tag{3.1}$$

The Δx would correspond to the distance between the current position and a previous position. The time Δt corresponds to the time between the positions used in Δx. The frames per second (FPS) is assumed to be known and could be used to calculate this time by dividing the number of frames between the positions used in Δx and the FPS. Equation(3.1), Δx, and the FPS are combined below in Equation(3.2):

$$v_{avg} = \frac{\Delta x}{(\frac{b}{\text{FPS}})} \tag{3.2}$$

In the updated average velocity Equation(3.2), b represents the number of frames between the positions used in Δx. If for instance the current position and the last position are used as Δx, with one frame between, the average velocity equation becomes $v_{avg} = \frac{\Delta x}{(\frac{1}{\text{FPS}})}$

Expected errors

When calculating the average velocity using a series of frames, there is a possibility of introducing errors. The value of b in Equation(3.2), which represents the frame between, must be an integer. If there is a significant change in position right after a frame is captured, the velocity calculated using the wrong b value can be inaccurate. In the worst-case scenario, this error would correspond to one frame. For a video stream with 30 FPS,

which is one of the standard FPS rates, this error would be $\frac{1}{30}s \approx 33ms$. While this error is likely to be negligible, it is still important to test whether it could affect the data.

3.4.2 Posture estimation

Defining a good posture

In order to implement a posture estimation, a good posture must first be defined. From Chapter 2.2 *The theory behind the squat*, some key points regarding the posture of the movement were defined. This included depth, knee position, stance, eye gaze, back angle, hip drive, and bar placement. Although the deepness of the squat has more to do with the efficiency of the exercise it could also be interpreted as a form of posture and is therefore included here. The proposed mono-camera-based solution for analyzing squat form is assumed to require the side profile view of the user, making the estimation of the stance difficult. The stance aspect of the posture feedback is therefore ignored in this solution.

The main focus of the posture part for this thesis was decided to be the depth of the squat as well as the neck position/angle. This was done to create a proof of concept, where the remaining posture estimations could be implemented at a later time. In this section, the approach to finding and monitoring the neck angle and squat depth is presented.

Neck angle monitoring

In Chapter 2.2 *The theory behind the squat*, it was established that maintaining a good neck posture entails orienting the head in a way that the eye gaze of the user corresponds to the user looking approximately 1.5 meters ahead of their standing position. To ensure this is achieved, it is important to calculate and monitor the angle of the neck to provide the user with feedback on their progress.

From Chapter 2.3 *Human pose estimation*, the identifiable body parts are

defined. In order to determine the neck angle, three points were used; hip, shoulder, and ear. By creating a baseline using the hip and shoulder points, the offset of the ear point forms an angle that can be calculated using the principles outlined in Chapter 2.7 *The geometric calculations*. The practical application of this can be seen in Figure 3.5 below where the angle was found to be 17°.

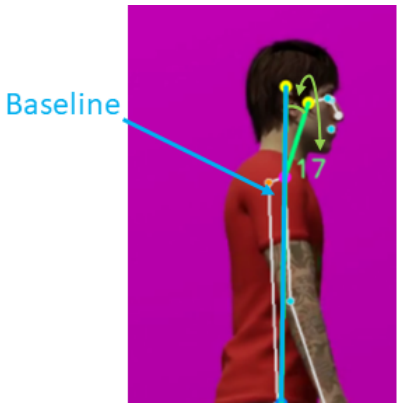


Figure 3.5: Graphic illustration of the neck angle.

To determine the acceptable neck angle, testing is required. This test will identify a suitable area for the neck angle, with a lower and upper limit defining this area. The neck angle will be continuously recorded during squatting and used as the basis for the scoring system.

Depth of the squat

To maximize the advantages of performing squats, it is essential to determine the appropriate depth for the exercise. In Chapter 2.2 *The theory behind the squat*, a suitable squat depth is classified as when the femur is parallel to the feet. Therefore, it would be beneficial to devise a method to monitor and ensure this alignment.

The solution chosen was to use the hip, knee, and ankle points to measure the distance from the knee and hip points down to the ankle height and compare the distances. In the illustration displayed in Figure 3.6, the blue

and the green lines represent the distances of interest these, called hip and knee height respectively.

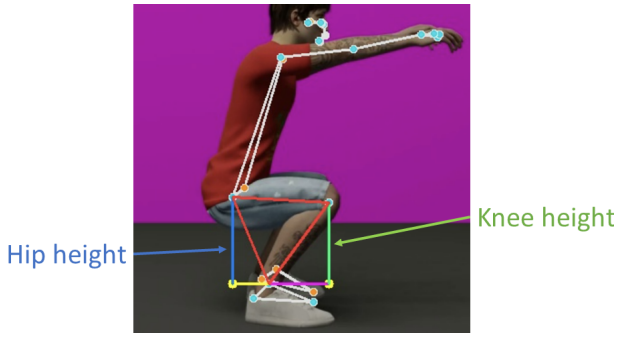


Figure 3.6: Graphic illustration a suitable squat depth.

The lines representing the hip and knee height illustrated in Figure 3.6 were calculated by subtracting the position of the ankle landmark from the knee and hip landmarks. This was done using only the y-coordinates ($hip_{height} = |y_{hip} - y_{ankle}|$) effectively getting the distance of that straight blue and green lines from Figure 3.6. These distances were then recorded to calculate a depth score after each squat. A good squat is achieved when the hip and knee heights are the same length, corresponding to the subtractions equaling zero. However, a deeper squat is considered better, so negative difference values are also taken into account. By calculating the distances this way an assumption was made that the y-coordinate alone would be sufficient to accurately calculate the distances.

3.5 The application

The application plays a crucial role in the product's functionality. It is where the user receives feedback on their workout. The application has been designed to provide continuous feedback to the user on various aspects such as the squat velocity, velocity change, depth of the squat, neck position, as well as depth and neck scores. Upon opening the app, the camera feed is displayed along with the mentioned outputs, as shown in the layout illustrated in Figure 3.7.

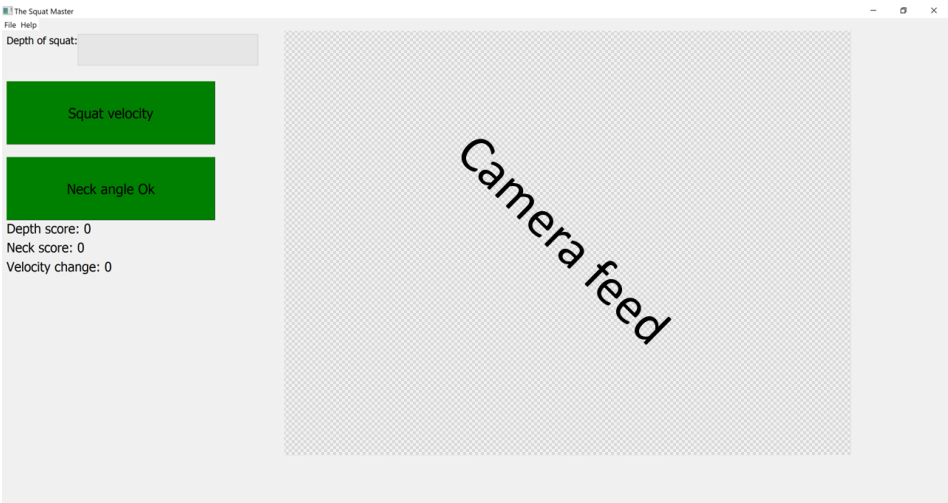


Figure 3.7: Graphic illustration on the layout of the application.

Upon starting the application the layout shown in Figure 3.7 is presented to the user. Real-time feedback on squat depth, as well as prompts to take a break or adjust neck angle, are displayed on the left side through the *Squat velocity* and *Neck Angle Ok* boxes. The camera feed is displayed on the right side, allowing the user to view themselves along with a virtual skeleton for guidance. The user can initiate and conclude squatting sessions via gesture control or by clicking a button labeled *Start Squat* located in the file tab.

3.5.1 Gesture control

To make the application more accessible, a method for initializing squat estimation was necessary. The idea was for the user to position the device with the application in a place where the whole body of the user could be clearly seen. The user would then stand in front of the device, ready to perform a squat. It would be convenient if they could start and stop the squat estimation without pressing any buttons on the device.

By implementing the pre-trained model explained in Chapter 2.4 *Gesture recognition*, a wide range of gestures was possible to use for the purpose of controlling the application. In order to identify practical gestures for the application, it was necessary to conduct a comparative test of the various potential gestures displayed in Figure 2.5 from Chapter 2.4 *Gesture recognition*.

3.5.2 User interface

The user interface is designed to be straightforward. Upon opening the application, the user is greeted with the layout displayed in Figure 3.7. The *camera feed* section displays a feed of the user with a virtual skeleton overlaid on top. To ensure accurate tracking, the user should position themselves at a 90-degree angle to the camera, ensuring the skeleton captures their entire body. The user can initiate the squat by either performing the *okay* gesture or selecting the *start squat* option under the file tab.

The user could then begin squatting. The first squat determines the initial velocity of the upward movement. In Figure 3.7, there are two green squares that indicate the user's speed and neck position. The square labeled *Squat velocity* turns orange when the upward thrust's velocity drops by 10% from the initial velocity. This indicates to the user that they need to either speed up or take a break. If the velocity drops by 20%, the square turns red, indicating that the user should take a break immediately. This update happens after every squat. If the user positions their head too far back or too far forward, the *Neck Angle OK* square will turn red and the message will change to *Control your neck!*. This check is performed continuously to ensure proper posture.

The scores will be displayed below the two squares. The neck score will update each time a change of direction is made in the squatting movement, whether moving downward or upward. The depth score will be calculated after each squat completion, and both scores will be shown as percentages.

The *velocity change* feature in the app displays the percentage change in velocity between the current squat and the initial squat. To stop the squat session, users can either use the *peace* gesture or press the *Start squat* button again in the file tab. The application will then show the total depth score, neck score, and a plot of the velocity change history as could be seen in Figure 3.8.

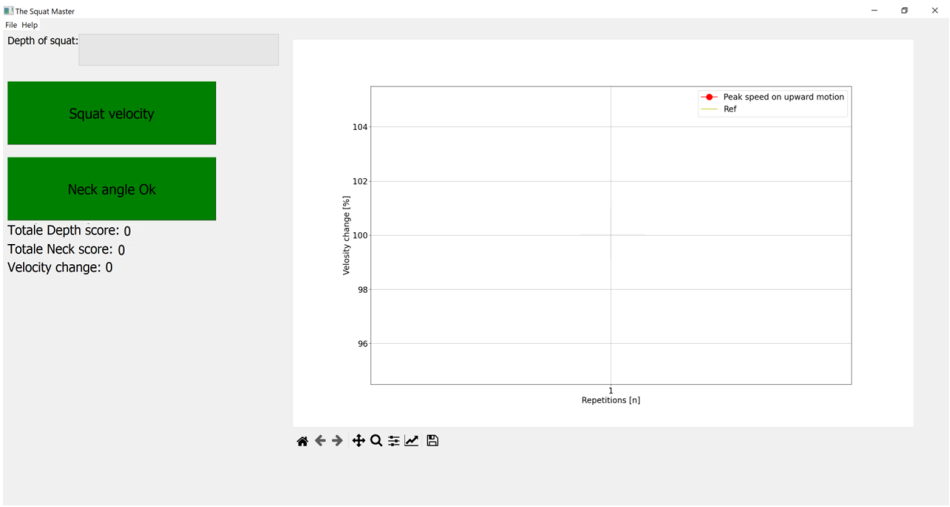


Figure 3.8: Graphic illustration of the parallel squat.

To close the app, either Cross it out in the top right corner or press exit in the file tab.

3.5.3 Scoring system

The scoring of the squat is divided into two categories: depth score and neck score. Velocity does not receive a score as it varies based on the user and their number of squats. In the previous chapter, the hip and knee heights were calculated to determine depth. The score is obtained by subtracting the knee height from the hip height, resulting in a delta value that determines the score, as shown in Equation(3.3).

$$score = 1 - (hip\_height \ - \ knee\_height) \tag{3.3}$$

The scores for depth are stored and utilized in computing the final score for the set. The Total Depth score is obtained by determining the average of all the depth scores and presenting it. It should be emphasized that a perfect squat would earn a score of 100%, but a deeper squat results in a higher score exceeding 100%. The graphical display in Figure 3.8 illustrates the Total Depth score below the boxes.

Similar to the depth, the neck angle is also calculated continuously and saved. An area is needed to define good neck posture, where a lower limit and an upper limit would determine how the score is calculated. This is shown in Equation(3.4).

$$score = \begin{cases} 1 & ;[\text{lower limit} < angle < \text{upper limit}] \\ 1 - (\frac{overshoot}{\text{scaling factor}}) & ;[angle > \text{upper limit}] \\ 1 - (\frac{undershoot}{\text{scaling factor}}) & ;[angle < \text{lower limit}] \end{cases} \tag{3.4}$$

The lower and upper limit from Equation(3.4) will determine when the continuous feedback, which is displayed in the application with the box labeled *Neck Angle OK*, shows as good or bad. When the neck angle calculated in real-time is above the upper limit or below the lower limit, the box will change color to red and display the text *Control your neck!*.

The score is calculated based on the data collected, which includes all saved neck angles since the last change in direction. Each angle is then evaluated

using a scoring Equation(3.4), with a score of 1 being awarded if the angle falls within the upper and lower limits. If the angle exceeds these limits, however, the score is calculated using the equation above. The terms *overshoot* and *undershoot* refer to the amount by which the angle exceeds or falls below the limits, while the *scaling factor* determines the severity of the impact of an incorrect neck angle on the score. Testing is required to determine the limits as well as the scaling factor.

Chapter 4

Testing and results

4.1 Setup

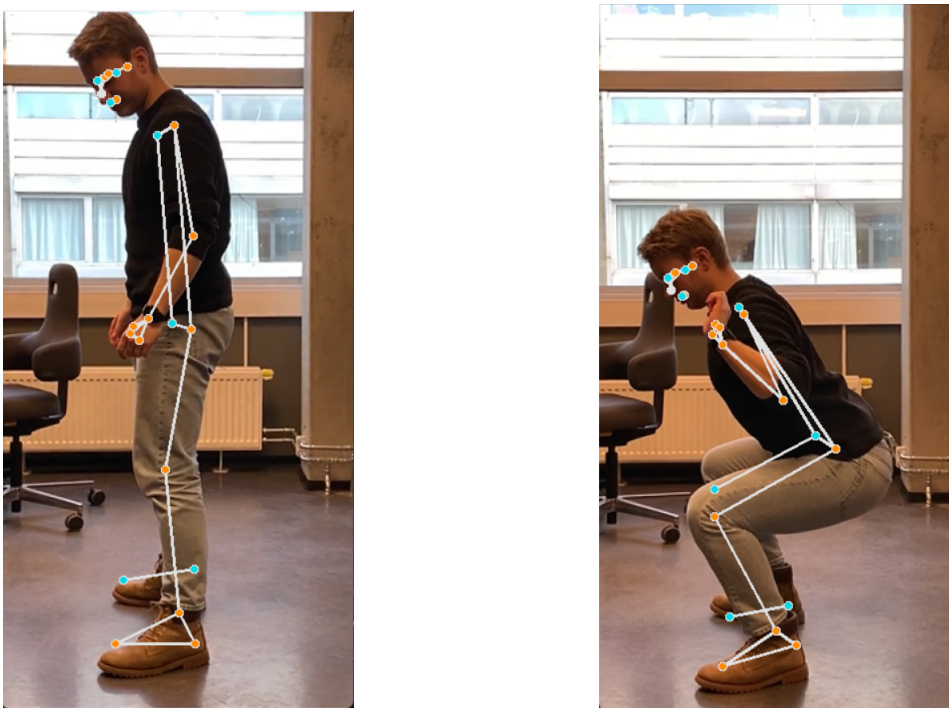
Various tests have been conducted using videos of real and simulated individuals. To guarantee consistent results, all tests were performed multiple times, explained in each test. The real videos were captured using a smartphone. To expedite the testing code, some video clips were divided into separate image folders. However, the original video was assessed to confirm the accuracy of the outcomes. The simulated video has been made and rendered using Blender[27]. An example of a squat was collected from Adobe's Mixamo [28]. The character *Bryze* and the simulation *air squat* were chosen from Mixamo, and the file was imported into Blender. The squat was then edited to fit the wanted tests.

4.2 Testing of velocity

The equation used to calculate average velocity required testing to ensure its accuracy. Furthermore, it was necessary to develop and test the logic for velocity-based training. This would enable users to receive continuous feedback on their velocity and know when to take a break.

4.2.1 Testing the equation for the average velocity

A real video was captured with a phone of a person doing one squat. In the figure below, Figure 4.1, the starting position and the bottom position illustrates the full movement. The full movement began at the starting position, down to the bottom position and up again.



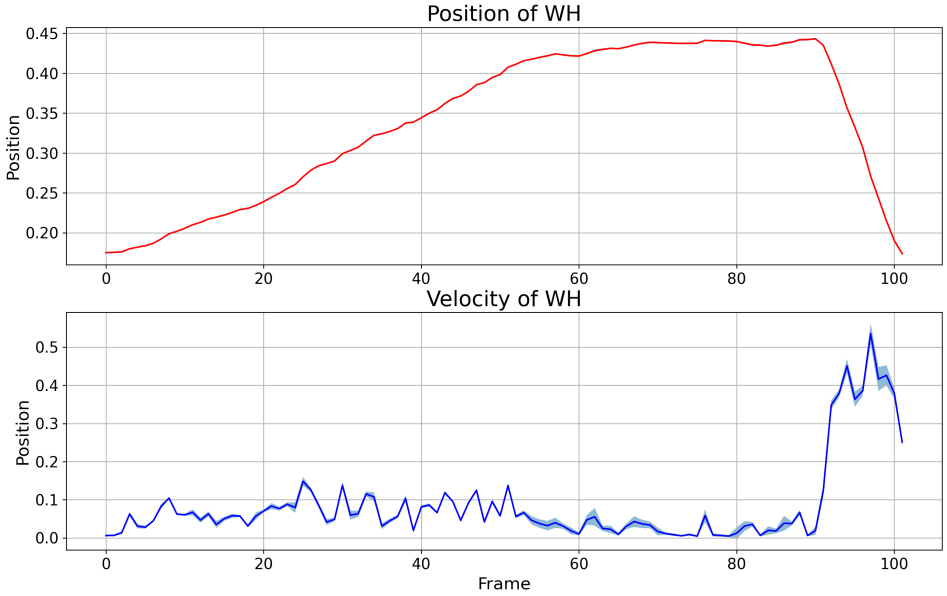
(a) Starting position. **(b)** Bottom position.

Figure 4.1: The captured video illustrated with the starting position and the bottom position.

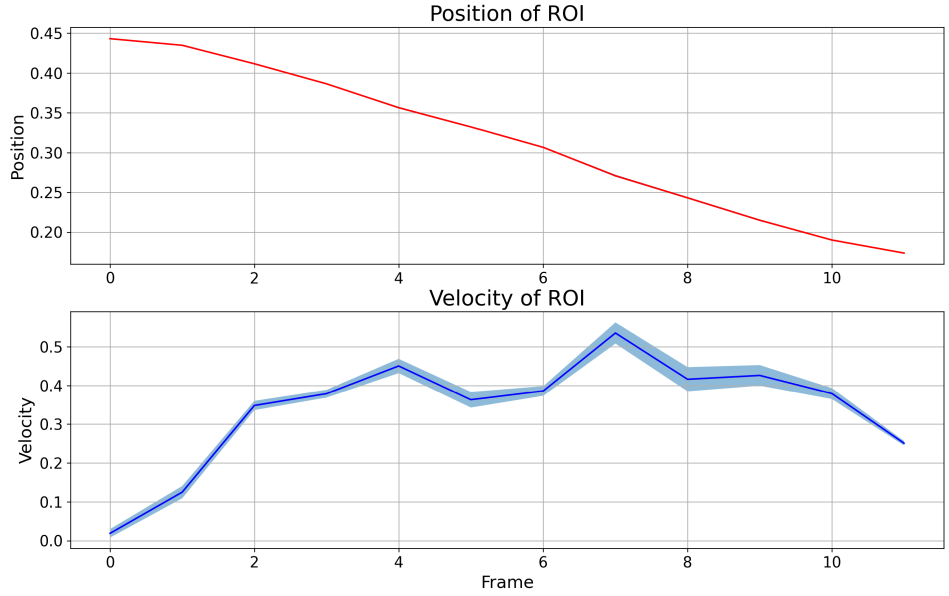
The equations from Chapter 3.4.1 *Velocity-based training* were implemented with one frame between each calculation, as shown below in Equation(4.1).

$$v_{avg} = \frac{d}{(\frac{b}{\text{FPS}})} = \frac{d}{(\frac{1}{\text{FPS}})} \tag{4.1}$$

Figure 4.2 demonstrates the implementation results. To obtain the result, the squat from Figure 4.1 was repeated 20 times, and the outcomes were compared. In Figure 4.2a, the position of the whole movement (WM) and the corresponding velocity in the WM is illustrated, while in Figure 4.2b, the position in the ROI and the corresponding velocity in the ROI are shown. The average position and velocity from the 20 runs are represented by red and blue lines, respectively, while the SD is displayed in each plot as the area around the line.



(a) Position of the WM with the corresponding velocity.



(b) Position of the ROI with the corresponding velocity.

Figure 4.2: Position of the WM and the ROI, with the corresponding velocities.

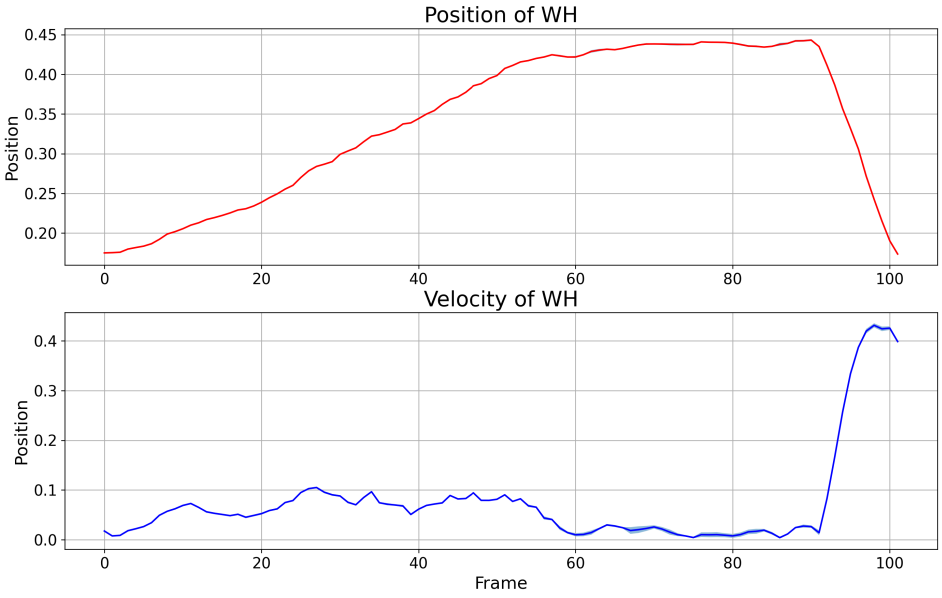
As could be anticipated, the derivative appears to be highly responsive to even the slightest movements, ss depicted in Figure 4.2. The velocity plot's standard deviation averaged 0.00791, and the highest velocity attained within ROI was found at 0.53441 ± 0.0272.

By using the same equation as earlier, Equation(4.2), but finding the distance d over a larger time and compensating with a larger b, the noise was improved. This worked as a running smoothing filter.

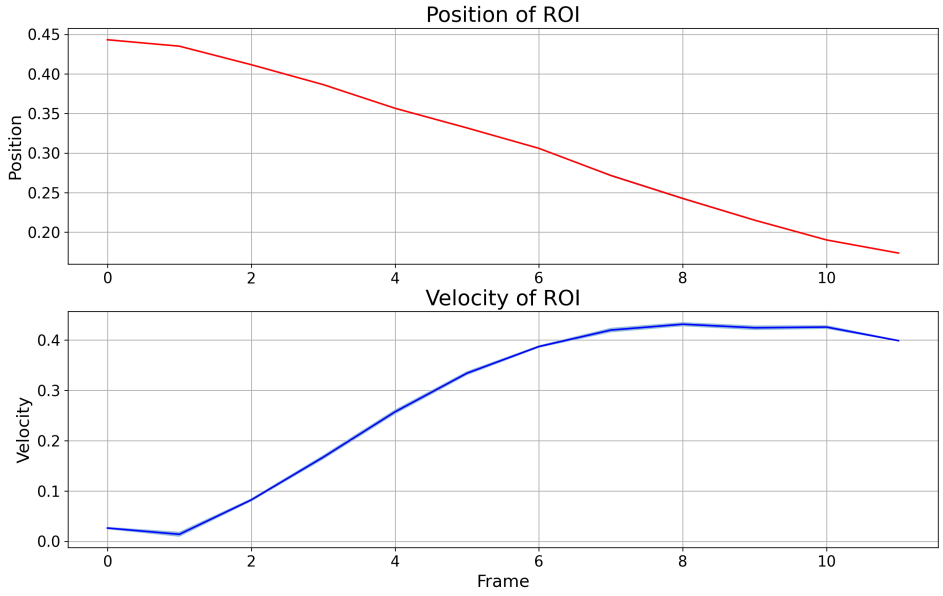
Multiple values were tested for the Δx and the corresponding b. The test that yielded the most promising results calculated the difference Δx with the current position value and the position value of ten frames in the past. This would then correspond to $b = 10$ shown in the updated equation, Equation(4.2).

$$v_{avg} = \frac{\Delta x}{(\frac{b}{\text{FPS}})} = \frac{\Delta x}{(\frac{10}{\text{FPS}})} \tag{4.2}$$

The result of implementing this updated equation is shown in Figure 4.3:



(a) Position of the WM with the corresponding velocity using the updated equation.



(b) Position of the ROI with the corresponding velocity using the updated equation.

Figure 4.3: Position of the WM and the ROI, with the corresponding velocities using the updated equation.

As depicted in Figure 4.3, the outcomes were considerably more polished. While there was still some noise in the WM, it had become noticeably smoother in the ROI. The velocity plot's average SD was found at 0.00228, and the maximum velocity achieved in the ROI was found to be approximately 0.4319 ± 0.0036, which enhanced the overall accuracy and reliability of the outcomes compared to the outcome without the smoothing filter. The SD of approximately 0.0036 for the peak velocity corresponded to a potential error of 0.008% in the comparison of velocities and was considered to be negligible in the scope of this thesis. The expected error introduced by using FPS, discussed at the end of Chapter 3.4.1 *Velocity-based training*, was also considered negligible with the use of this updated equation.

4.2.2 Testing the velocity-based training

Some tests were needed to ensure that the program was able to find out when the person should take a break, based on Chapter 2.1 *The theory of velocity-based training*. Three videos were tested, two simulations from Blender, and one real video.

Red circles were used to visualize the maximum average velocities, and green circles were used to visualize a squat that had a significant velocity drop with respect to the first squat. In other words, if the maximum average velocity was visualized with a green circle, the criteria for the velocity-based training were met for the figures in this section.

The simulations from Blender

First, a simulation from Blender was tested. One squat was done and repeated eight times in a row. During the sequence of squats, each upward movement was gradually scaled up in relation to the first squat, with random increments until the last squat. This scaling resulted in a slower velocity of the upward movement for each squat, as the movement occurred over a longer period of time. The downward movement of each squat was unchanged and identical to each other. Figure 4.4 shows the starting position and the bottom of the first squat.

(a) Starting position. **(b)** Bottom position.

Figure 4.4: Starting position and the bottom position of the first squat of the simulated training set.

The position and the velocity were found as before. A check if the current squat had a velocity loss of 15% or greater in regards to the first squat was implemented, and the results are shown in Figure 4.5. The simulation was tested 20 times, and the figure shows the average positions and the corresponding average velocities from the tests.

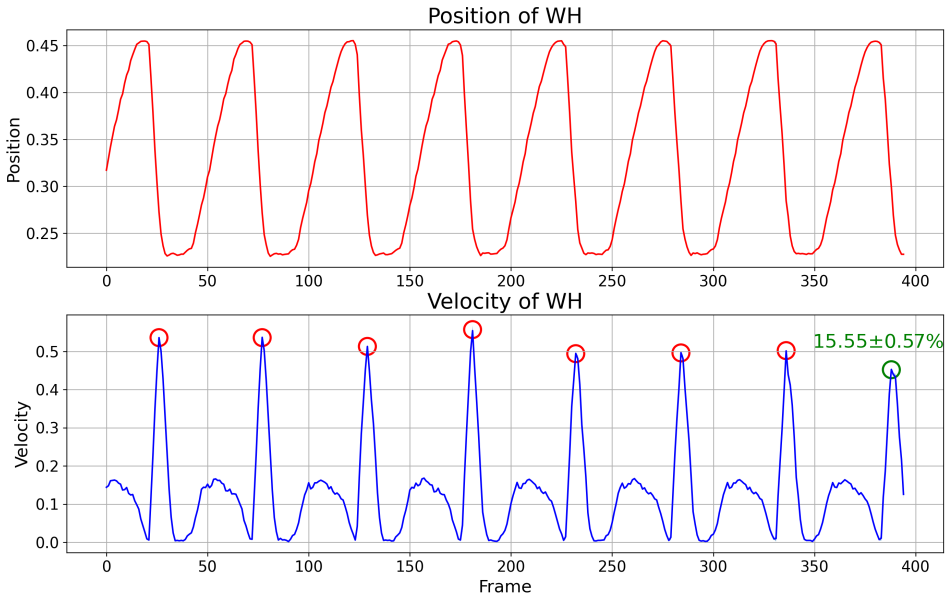


Figure 4.5: A simulated training set, where the green circle percent a significant velocity drop.

From Figure 4.5 it could be seen that all the maximum velocities were found and illustrated with red circles. However, the last squat met the requirement of the 15% velocity loss and was represented with a green circle. According to the figure, the velocity loss was equivalent to $(15.55 \pm 0.57)\%$.

To better visualize the peak velocities and their corresponding SDs, a bar plot was created. Each peak velocity represented by a red circle, including the last one, represented by a green circle in Figure 4.5, was displayed as a blue bar. The SD was also included as an error cap at the top of each bar. Figure 4.6 displays the bar plot.

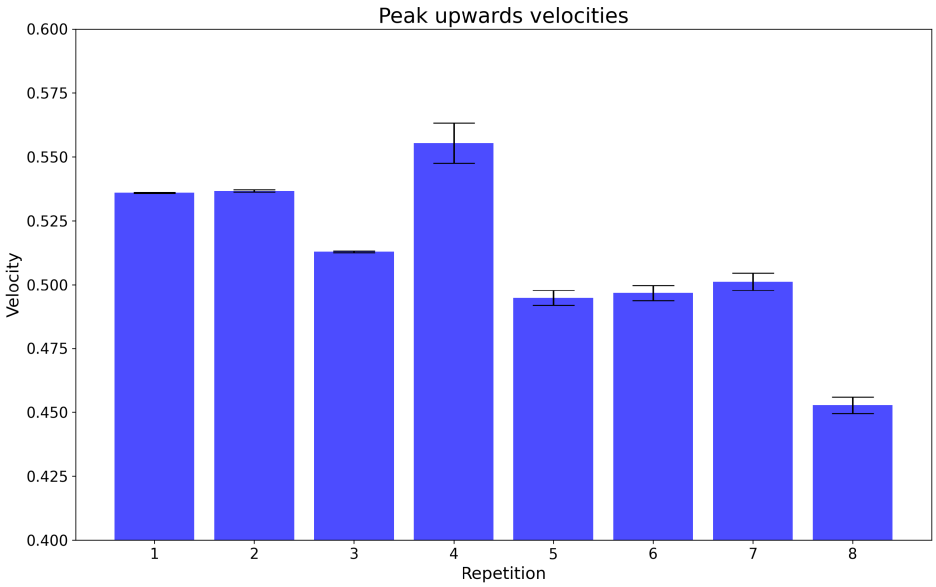


Figure 4.6: Bar plot showing the peak velocities with the corresponding SDs illustrated as error caps.

In Figure 4.6, it was noticeable that the SDs detected were relatively minor. The largest SD recorded was found around 0.0079 for the fourth repetition, which was considered an acceptable margin of error.

A new simulation was created and tested using Blender. It was based on the same simulation as the one shown in Figure 4.5, but this time the downward movements were scaled as well in order to check if varying velocity in this movement affected the results.

It is important to mention that the scaling process in Blender may not result in peak velocities matching Figure 4.5. However, the relationship between peak velocities should remain consistent. If the downward movement had no impact on the results, the percentage for the last velocity was expected to be similar to the one shown in Figure 4.5. The results of the new simulation are illustrated in Figure 4.7.

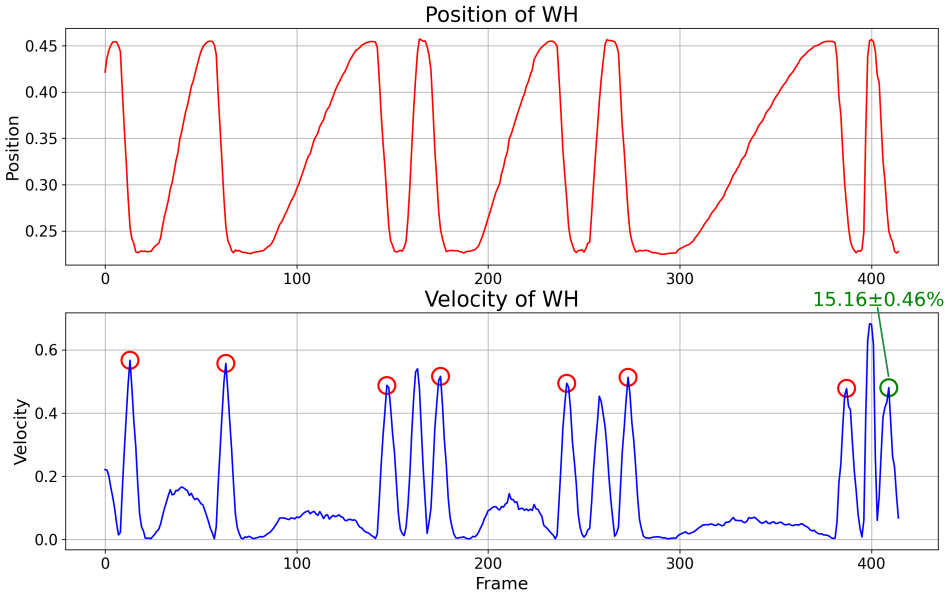


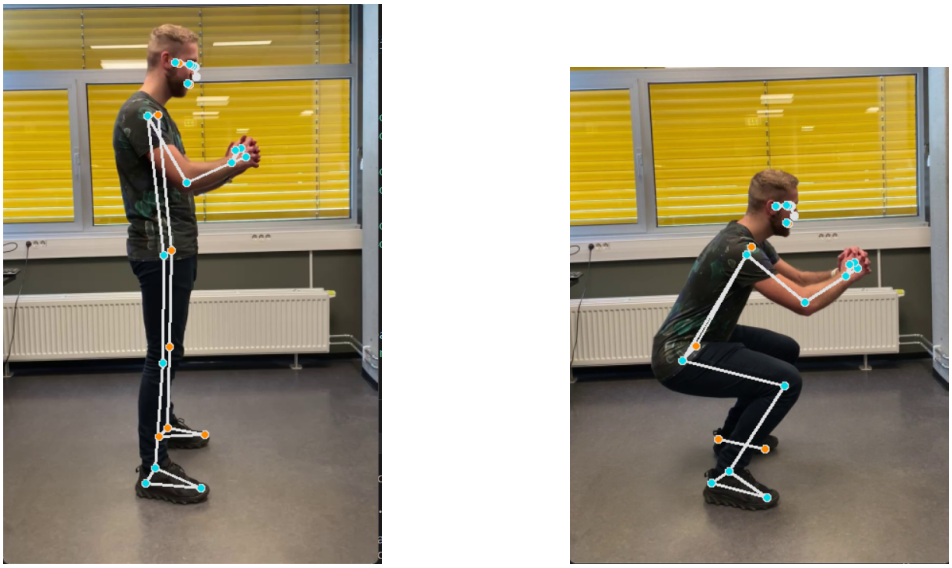
Figure 4.7: A simulated training set, with variation in the downward velocity.

From the test illustrated in Figure 4.7 it could once again be seen that all the maximum velocities were found and illustrated with red circles. The last squat met the requirement of the 15% velocity loss and was represented with a green circle.

The downward movement varied randomly and was quite exaggerated. This was not representative of a real training set, however, the results showed that the percentage was in the same region as shown for the first simulation and could imply that the variation in the downward velocity did not affect the results.

The real video

A new video was taken of a person, where he took two fast squats followed by one significantly slower squat relative to the first two. The video was tested 20 times. Figure 4.8 depicts the person in the starting position and at the bottom of the first squat. The figure below, Figure 4.9 shows the position and the velocity.



(a) Starting position. **(b)** Bottom position.

Figure 4.8: Starting position and the bottom position of the first squat.

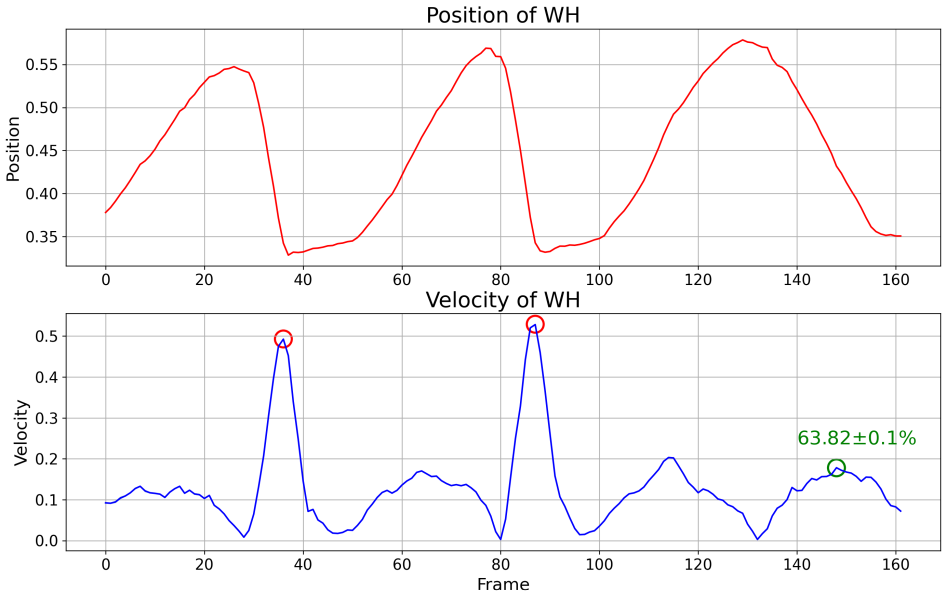


Figure 4.9: A Training set with three squats, where the green circle percent a significant velocity drop.

From Figure 4.9 it could be seen that the maximum velocity for each thrusting movement was found. The last squat was found to be $63.82 \pm 0.1\%$ relative to the maximum velocity of the first squat. The maximum SD found for one squat was approximately 0.0008 and was assumed to be negligible.

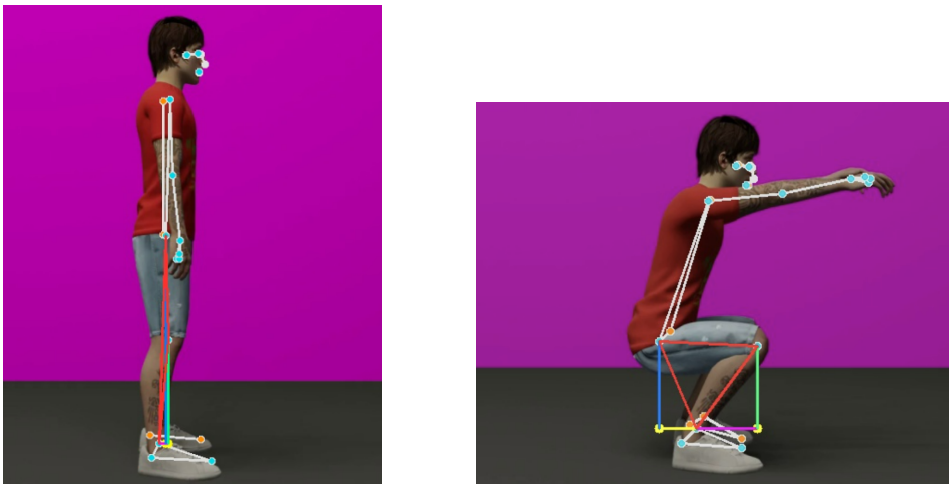
By comparing the position with the velocity, it could also be seen that the maximum velocity of the downward movement in the last squat was larger than the maximum velocity of the upward thrust. The ROI was still found as expected and the right maximum velocity was found, as indicated with the green circle and percent in the velocity plot in Figure 4.9.

4.3 Posture measurements

In order to ensure that the posture measurement system was functioning properly and providing accurate feedback to the user, various tests had to be conducted. This included testing the depth function using a simulation from Blender, and assessing the angle estimation using both a real video and a simulated video that would help determine the scoring equation parameters for the neck angle. Additionally, two simulated tests was conducted to determine the accuracy of mono-camera-based measurements.

4.3.1 Testing the depth function

In order to ensure accurate calculations and reliable landmarks, a simulated video was utilized to test the effectiveness of the squatting technique. This approach provided greater assurance that the test was executed as intended. Figure 4.10 depicts the simulation done with the starting position and at the bottom position of a squat. Ideally, at the bottom position, the difference between the blue and green lines should be as close to zero as possible.



(a) Starting position. **(b)** Bottom position.

Figure 4.10: Starting position and the bottom position of the simulated squat.

The purpose of the test was to measure the blue and green distances depicted in Figure 4.10b and analyze the difference between them while the individual performed a squat. This process was repeated 20 times to ensure accuracy. The results of the test are demonstrated in Figure 4.11. The upper plot displays the advancement of the squat movement in terms of distance from the hip and distance from the knee, while the lower plot indicates the difference between the two distances with a dotted line indicating the desired end position. The plot depicts the movement from the starting position, where the difference in height amounts to 0.4, to the bottom position.

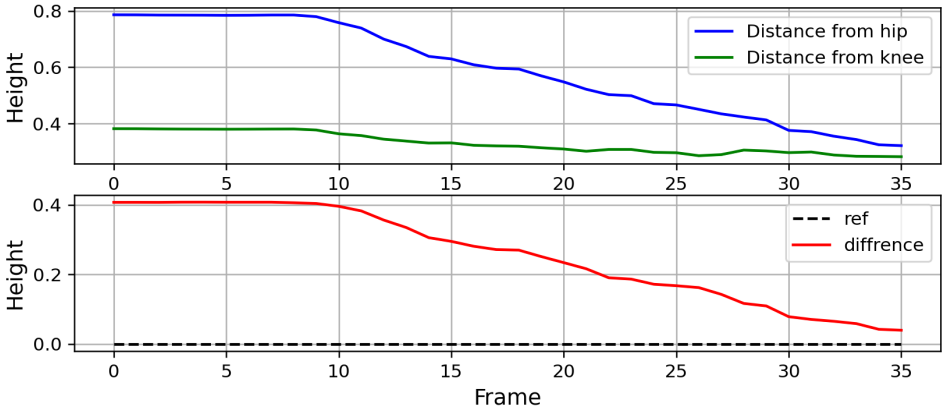


Figure 4.11: Parallel distances.

Upon analyzing the plots illustrated in Figure 4.11, the results were generally as anticipated. As an individual descends, the difference between hip and knee height decreases towards the desired height. The squat's bottom value was measured at 0.04, which corresponds to a 10% offset from the reference value of 0 considering the character's initial standing height difference of 0.4. This outcome was deemed satisfactory. After analyzing the SD between the 20 runs, it was found to be approximately ± 0.0007 and determined to be negligible.

4.3.2 Estimating a good neck angle

A test was conducted to determine the appropriate neck position, where a person held their head still in the direction outlined in the theory Chapter 2.2 *The theory behind the squat*, looking approximately 1.5 meters in front of the individual's feet. This is depicted in Figure 4.12. A video was recorded of the individual, where the camera initially had a close distance to the individual, before gradually moving back to a distance of approximately 6.5 meters away from the individual. The anticipated outcome of this test was that the angle would remain consistent when up close, but become progressively less precise as the camera moved further away. Figure 4.12 displays the starting position of the video.

Figure 4.12: Starting position for the video used for estimating the neck angle.

The test results were represented in a plot shown in Figure 4.13. It can be observed that the angle fluctuates between around 29° to 34°. The average value for the entire test was determined to be 31.6°.

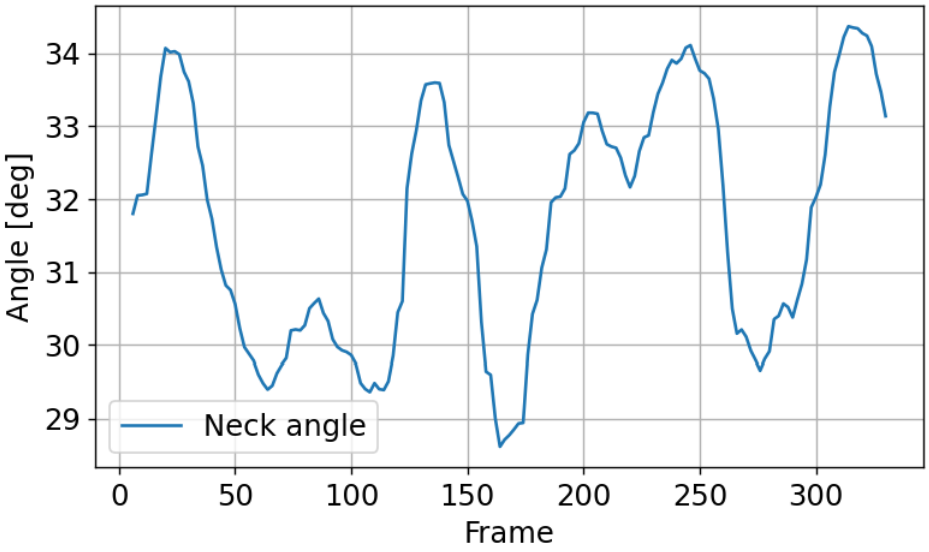


Figure 4.13: The behavior of the neck angle.

The result is not as expected. There seems to be more interference than anticipated, and it's unclear if the measurement improves as the camera is closer to the person. It's hard to determine the actual neck angle from the graph in Figure 4.13. Nevertheless, assuming the correct angle is in the data found, by analyzing the plot it is possible to see an oscillation of about 5°. This oscillation was treated as an error.

4.3.3 Testing neck angle with simulated input

To check how the neck monitoring develops in a controlled environment while doing a squat, a test was contrived where the simulated video from the depth test would be used. The reason for using the simulated video was that the character in the video had a fixed neck position during the entire squat. This made the input more predictable.

Figure 4.14 presents the result after running the same squat 8 times in a row. The top plot presents the position of the squat, where the smallest value represents the character from the simulation in a standing position and the large value represents the character from the simulation in a bottom squat

position. The plot below shows the neck angle through all the repetitions.

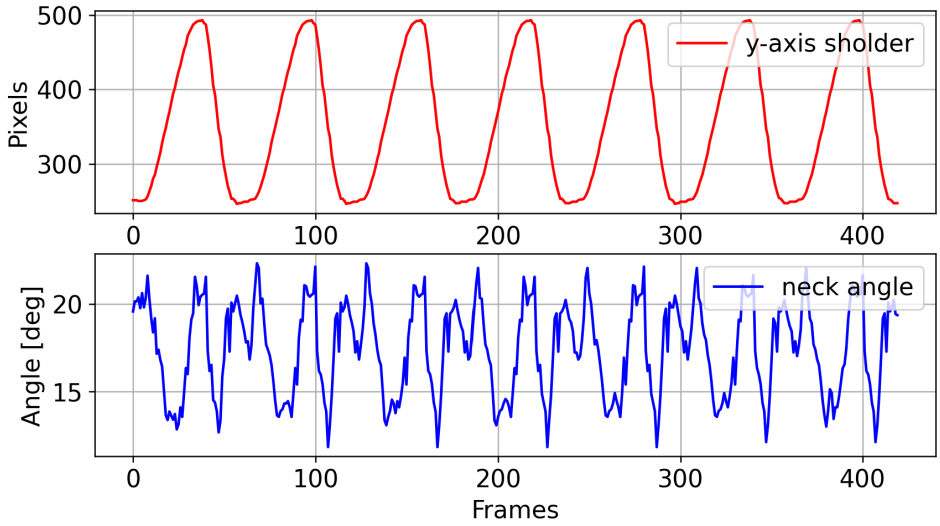


Figure 4.14: Position of the shoulder and the associated neck angle of the character from the simulation, over the span of 8 squats.

In the earlier test illustrated in Figure 4.13, it was predicted that there would be a variation of approximately 5°. However, upon analyzing the data, it was discovered that the angle varies between approximately 12° and 22°. This was a greater range than anticipated. Further examination of the video and the Mediapipe skeleton showed that when the simulated character squatted, their arms moved upwards, which caused the shoulder point to shift from its original position. As a result, the line between the hip and shoulder slightly shifted in relation to the ear point, resulting in a change in the angle of the neck.

Combining the discoveries from the two tests, the parameters of the scoring equation for the neck angle were determined. The scoring equation referred to was derived in Chapter 3.5.3 *Scoring system* in Equation(3.4). By examening the plot in Figure 4.14 and Figure 4.13, an area defined as a good neck posture was chosen to be between 15° and 34° in neck angle, meaning the lower limit amounted to 15 and the upper limit amounted to 34. The scaling factor was chosen as 10, ensuring that the wrong neck angle would be properly reflected in the score. The scoring equation, with the parameters determined, is shown below in Equation(4.3).

$$score = \begin{cases} 1 & ;[\text{lower limit} < angle < \text{upper limit}] \\ 1 - (\frac{overshoot}{\text{scaling factor}}) & ;[angle > \text{upper limit}] \\ 1 - (\frac{undershoot}{\text{scaling factor}}) & ;[angle < \text{lower limit}] \end{cases}$$

$$\Rightarrow score = \begin{cases} 1 & ;[15 < angle < 34] \\ 1 - (\frac{overshoot}{10}) & ;[angle > 34] \\ 1 - (\frac{undershoot}{10}) & ;[angle < 15] \end{cases} \tag{4.3}$$

4.3.4 Testing errors of mono camera-based measurements

During the application's development, it was assumed that the camera should be placed with a side view of an individual during an exercise. This assumption was made to ensure that the results would be accurate and not misrepresented due to camera misalignment. However, it was necessary to test what happened if the camera came out of the assumed alignment.

To test the camera positioning and potential errors from misalignment, a simulation was created where a character stood in a squatted position while the camera was moved in different directions. In Figure 4.15 the character is depicted from a bird's eye view to give an enhanced comprehension of the testing process. One of the tests conducted involved rotating the camera anti-clockwise 360° around the character starting from the right side at 0° and ending back at the same position. This test is referred to as the **360° test**.

Another test involved viewing the character from the right side at 0°, while the camera moved up in a negative direction on the y-axis and rotated in a negative direction of rotation around the x-axis. After the movement up in a negative direction on the y-axis and back down again, the camera rolled towards the back and towards the front of the character equating to a negative direction of rotation and then a positive direction of rotation around the z-axis. The rotations around the z-axis happened while viewing the right side of the character at 0°. This test is referred to as the **Roll test**.

The tests consistently maintained the same distance from the center of the

character throughout and were simulated 10 times to obtain the accompanying standard deviation and ensure the reliability of the results.

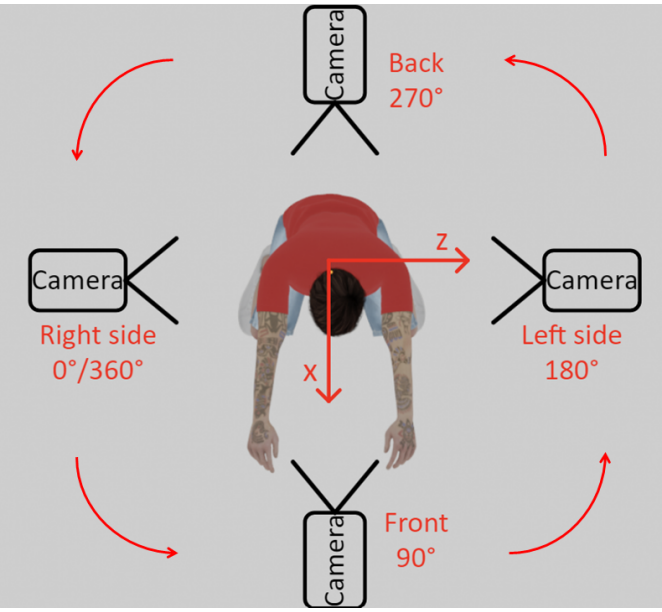


Figure 4.15: An illustration of how the camera was positioned in the test.

To analyze these tests, the depth height was used as this function was a crucial part of the application. From the earlier test in Chapter 4.3.1 *Testing the depth function*, the depth of the simulated character was found to be 0.04. In the test conducted in this section, this value is considered to be the true depth value. In other words, if the measured depth deviates from this value, it is considered to be wrong due to the camera alignment in relation to the character. It is assumed that any deviations and corresponding errors in the measurement of the depth due to the camera alignment correspond to the same error due to the same camera alignment in all the other measurements and calculations done in this project.

In these tests, the assumption that the use of the y-coordinate alone was sufficient in calculating the functions needed was also tested. This was done by comparing the calculations done with only the y-coordinate and calculations with both the y and the z-coordinates, referred to as one-dimension (1D) and two-dimensions (2D) calculations for this section.

360° test

The test was conducted as described and is shown in Figure 4.16. The target line, illustrated with a black dotted line in Figure 4.16, represents the true depth value at 0.04. The upper plot displays the depth height found from the right side data points and the lower plot displays the depth height found from the left side data points. The red and blue lines present the depth found while doing 1D and 2D calculations respectively. The shaded area around the mentioned lines represents the SD.

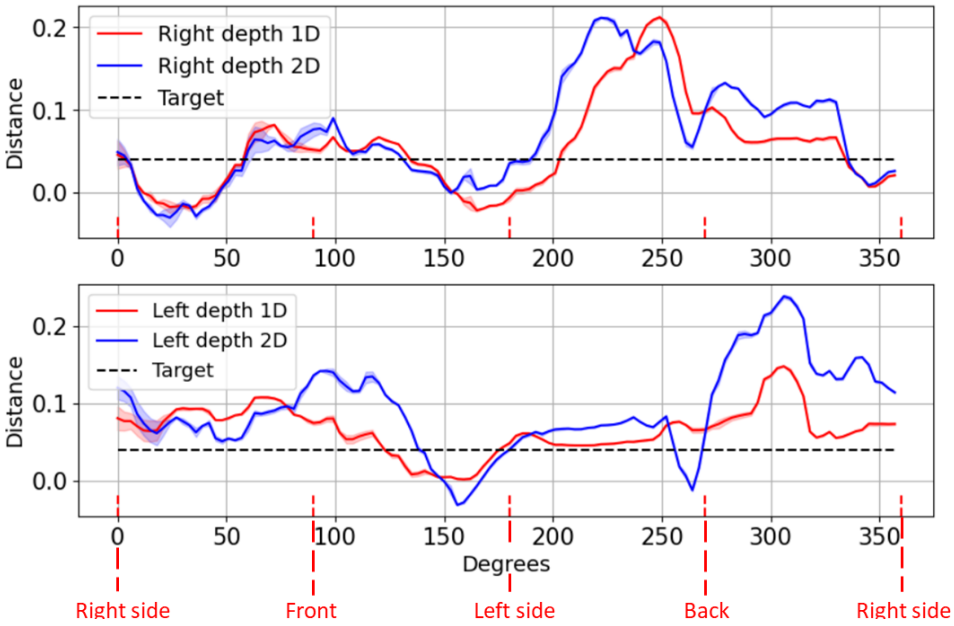


Figure 4.16: Illustration of the calculated difference between hip and knee height plotted against the degrees in camera position.

From the upper plot in Figure 4.16, it was interesting to see how the result seems to improve when the camera was aimed toward the front of the character at 90°. It was also clear to see that a camera position around the back of the character (270°) would not be ideal.

Drawing any concrete conclusion just by looking at the plot could be challenging. Therefore, the standard deviation of the four areas shown in Figure 4.15 was calculated. The data was divided into four parts: 45° to 135°, 135°

to 225°, 225° to 315°, and 315° to 45° with equates to the area around 0°, 90°, 180°, and 270° respectively as depicted in Figure 4.15. The SD for these areas was then calculated against the true depth set at 0.04. The results are presented in Table 4.1 below.

	1D		2D	
Side:	**Right depth**	**Left depth**	**Right depth**	**Left depth**
0°	0.016	0.017	0.022	0.035
90°	0.010	0.018	0.011	0.031
180°	0.018	0.009	0.024	0.014
270°	0.039	0.020	0.045	0.044

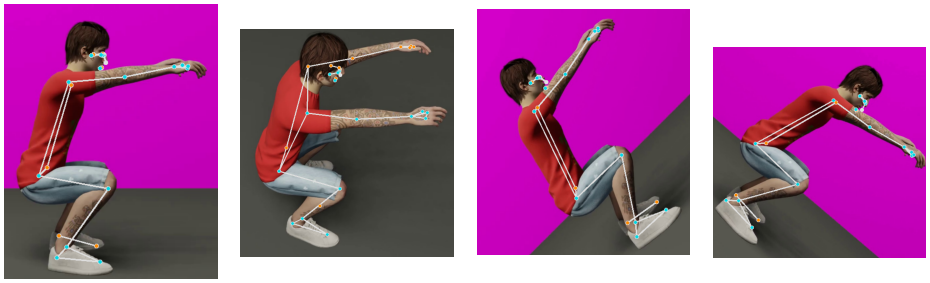
Table 4.1: The standard deviation from the target divided into 4 areas around the regions of interest.

After examining Table 4.1, it is evident that the 1D calculations exhibited the lowest SD for the left depth at 180°, which was logical as the camera was positioned at the character's left side. However, it was surprising to note that the right depth at 90° also had a low SD, as this corresponds to the camera being in front of the character and was assumed to produce errors. On the other hand, it is worth noting that the same can not be said for the left depth at 90°.

Regarding the comparison between 2D calculations and 1D calculations, it cannot be concluded that the former was better as the standard deviation consistently appears worse for the 2D calculations. This means the 1D calculations gave a better result.

Roll test

The test was conducted as described, and the starting position, top view, right roll, and left roll are illustrated in 4.17, along with the corresponding frames the images are taken from as described in the captions.



(a) Starting position, Frame 0. **(b)** Top view, Frame 60. **(c)** Right roll, Frame 150. **(d)** Left roll, Frame 215.

Figure 4.17: Outtakes from the test video.

As described earlier, it was assumed that the camera had to be correctly aligned, viewing the side profile of a person. Thus, it was expected that the depth height would be unstable around the true depth at 0.04. The graph shown in Figure 4.18 displays the calculated depth height for both the 1D and 2D represented by the red and blue lines respectively. The target line, illustrated with a black dotted line in Figure 4.18, represents the true depth value at 0.04.

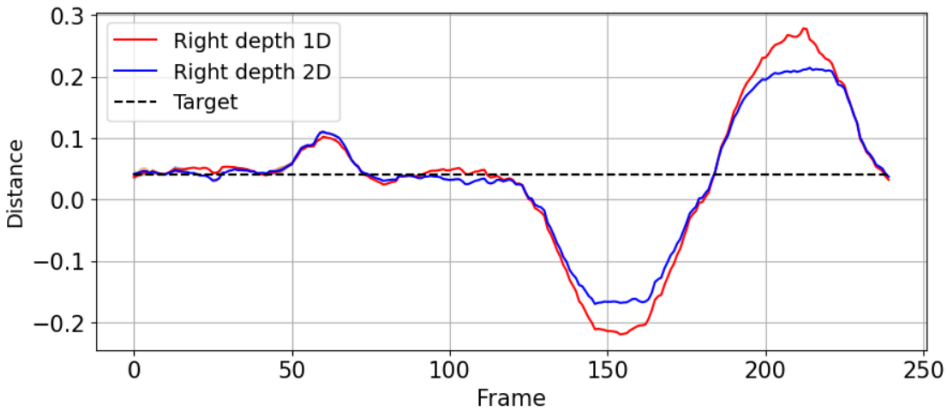


Figure 4.18: Plotting the depth distance while the camera changes height and orientation.

After analyzing the plot and the accompanying video frames presented in Figure 4.17, it appeared that smaller movements along the negative direction in the y-direction along with the corresponding rotation around the

x-axis did not have as much impact on the results as expected. However, as the camera approached the top point represented in Figure 4.17b, there was a slightly higher deviation from the target line. The rolling movement captured in Figure 4.17c and Figure 4.17d seemed to induce a large deviation as could be seen in the corresponding sections of Figure 4.18. In fact, the largest deviation is at 0.3 and is not considered an acceptable error.

Based on the previous 360° test, there was no indication that the 2D calculation provided any advantages over the 1D calculation. Nevertheless, Figure 4.18 displays that the 2D calculation was more proficient at handling the roll action disturbance represented in Figure 4.17c and 4.17d. However, it did not seem to improve the overall estimation of the depth as the areas where the 2D calculations had an improvement over the 1D, were after the estimations had already deviated to far from the target line.

After conducting two tests, it was concluded that although the first test showed some minor improvements when the camera had the side view of the user in regard to the other views, the deviations were still quite small. On the other hand, regarding the 1D versus 2D calculations aspect of the test, it had not been demonstrated that using the z-coordinate as well as the y-coordinate provided any significant advantages over using only y-coordinates, meaning the assumption that only the y-coordinate is sufficient is valid.

4.4 Testing of the application

The application served as the medium for user interaction and feedback on squat performance. It was important to conduct testing to ensure its proper functionality. The gesture control and user interface, along with the scoring system, were thoroughly tested and assessed for quality assurance.

4.4.1 Testing the gesture control

As stated in Chapter 3.5.1 *Gesture control*, a pre-trained library was found that could recognize various gestures. In order to determine the most practical gesture for initiating and concluding a squat, a test was conducted to evaluate the reliability of the various gesture estimations. Chapter 2.4 *Gesture recognition* showcases Figure 2.5, which displays all possible gestures that the pre-trained library can estimate. Eight of these were chosen and required testing to determine their reliability, the chosen gestures were *call me*, *fist*, *live long*, *okay*, *peace*, *stop*, *thumb down*, and *thumb up*.

To conduct the test, a video was recorded for each gesture where the gesture was made and moved around in the frame in order to see how often the gesture was detected, and if the model identified a different gesture than it was supposed to. Then, each video was run through the gesture recognition model, saving all gestures found for each test.

The bar plot in Figure 4.19 illustrates the results of the different gesture tests. The blue bars show how many times the correct gesture was identified, while the red bars represent how often any gesture was identified. In other words, the red bar displays how often a gesture is found for each test, regardless of the gesture. The y-axis displays a score that ranges from $0\% = 0$ to $100\% = 1$, indicating how often the item in question was detected.

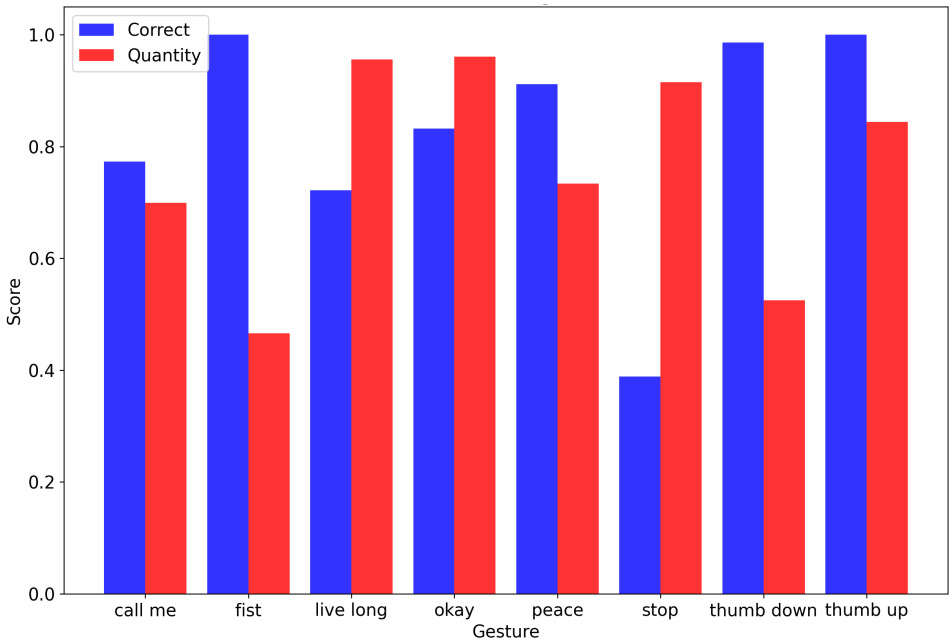


Figure 4.19: Gesture recognition scoring.

Upon examining Figure 4.19, it was discovered that the gestures of *fist*, *thumb up*, and *thumb down* had high scores. However, upon further examination of the blue bars in comparison to the red, it became clear that while *fist* and *thumb down* were often detected correctly, they were also the most difficult to detect out of all the images sent in. More than 50% of the images sent in had no recognizable gestures.

It was also important to consider unintentional gestures. For example, *stop* and *live long* is basically just an open hand, which is a natural gesture for a person to make. This means it could easily be detected by mistake. The same could be said for *thumb up*, especially if the person would be holding a weight. For these reasons, the gestures of *okey* and *peace* were chosen. From the bar plot, these two gestures had acceptable scores and were some of the least natural gestures to make. This eliminated the possibility of unintentional commands. The mentioned results as well as the rest of the selected eight gestures in their respective test are attached in Appendix B *Gesture table*.

After selecting these gestures, it was important to ensure that they didn't have any correlation with each other. It would be unfortunate if the user tried to start the squat and immediately stopped because the model detected another gesture meant to stop the exercise. In Figure 4.20 below, the results of the recognition test for the *okay* and *peace* gestures are displayed. The bar plots show how many times the gestures were detected, as well as any other gestures that were detected during the recognition test.

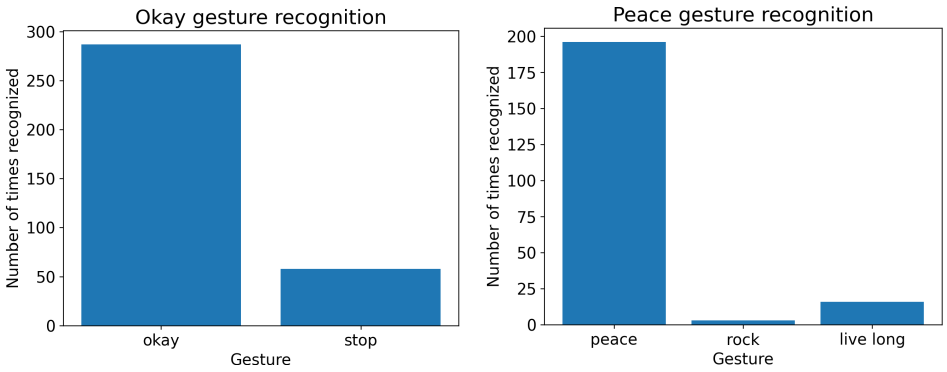


Figure 4.20: Recognition test to ensure reliability.

Based on the analysis of the bar plots in Figure 4.20, it can be concluded that the *okay* and *peace* gestures were not correlated. Therefore, they were suitable to use without the risk of unintentional commands or confusion with each other.

4.4.2 User interface and scoring system

A test where a person started the application and did the squat from start to end was done. The computer screen recorded the whole exercise session, and representative screenshots of the session will be shown in this section. The interesting points to examine were the user interface and scoring system.

The user started the application and got into position, making sure that the whole body was in the frame, and when the user was ready, the squatting set began. An immediate problem was discovered as the person doing the squat

had a hard time seeing the feedback on the screen due to the necessity of the camera position, and therefore the screen, relative to the person. This was solved by having another person tell the user about the feedback, ensuring that the feedback was given continuously and that the user could focus on squatting.

In Figure 4.21, the start of the exercise is shown. Here it could be seen that the camera feed was displayed as intended and that the skeleton of the person was found. In the top left corner of the figure, it could be seen that the squat monitoring gets started in the application manually by using the *file* tab and pressing the *start squat* button.

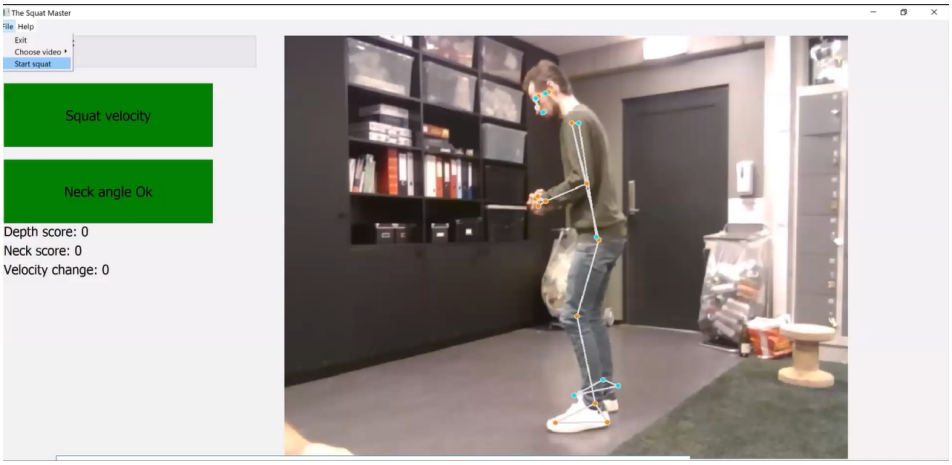


Figure 4.21: The start of the exercise.

After squatting for some repetitions, the velocity of the upward movement dropped over 10% compared to the initial velocity, and the square labeled *Squat velocity* turned orange as shown in Figure 4.22.

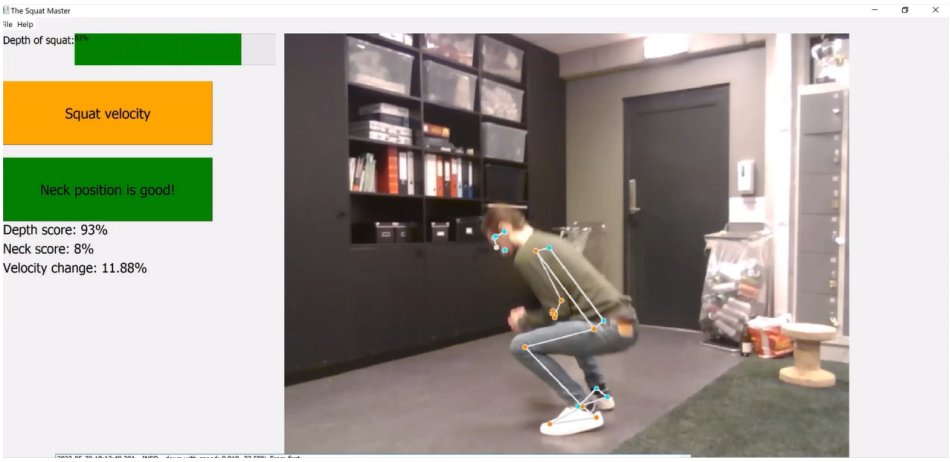


Figure 4.22: A screenshot showing the change of the square labeled *Squat velocity*.

From Figure 4.22 it could also be observed that below the two boxes with the labels *Squat velocity* and *Neck position is good!* the depth score, neck score, and velocity change were displayed as percentages. As this particular screenshot was taken while the user was performing the downward motion of the squat, this corresponded to the highest depth percent of the last squat, the neck score from the last squat's upward movement, and the velocity change from the last squat compared to the initial squat. The velocity change is displayed as 11.88% and the box labeled *Squat velocity* was changed as designed.

The screenshot below, shown in Figure 4.23, shows the upward motion subsequently from the downward movement shown in Figure 4.22. In other words, Figure 4.23 shows the upward motion to the same squat as shown in Figure 4.22.

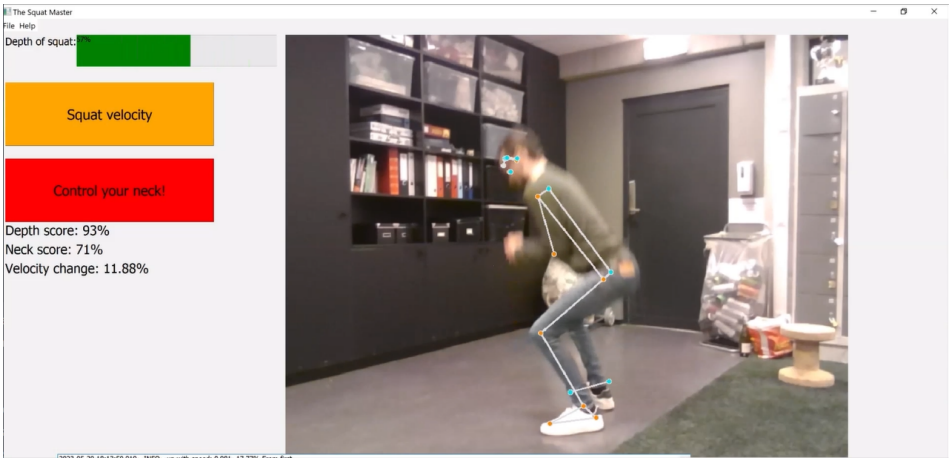


Figure 4.23: A screenshot showing the upward motion.

Figure 4.23 shows the box that was labeled as *Neck position is good!* changed the label to *Control your neck!* and changed the color to red.

The user got the feedback on the orange box labeled *Squat velocity* and chose to continue the session. The user managed to increase the velocity, resulting in the box changing color to green again. This, however, only lasted for a few more repetitions before the velocity drop was greater than 20% and the box turned red. This could be seen in the screenshot shown in Figure 4.24, where the user was standing still after the session, while another person ended the squat session in the application by pressing the *Start squat* button in the file tab.

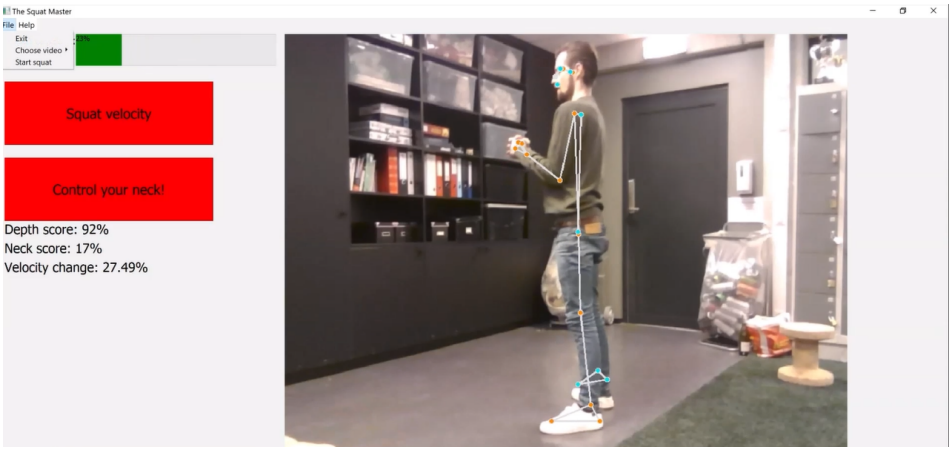


Figure 4.24: A screenshot showing the end of the repetitions, and the other person ending the session.

From the screenshot shown in Figure 4.24 the velocity change displayed was 27.49%, and the box labeled *Squat velocity* was changed as designed.

When the button *Start squat* was pressed the squat session ended the camera feed was replaced with a plot illustrating the peak velocity of each squat. In addition, the total depth score and total neck score were displayed, as could be seen in Figure 4.25.

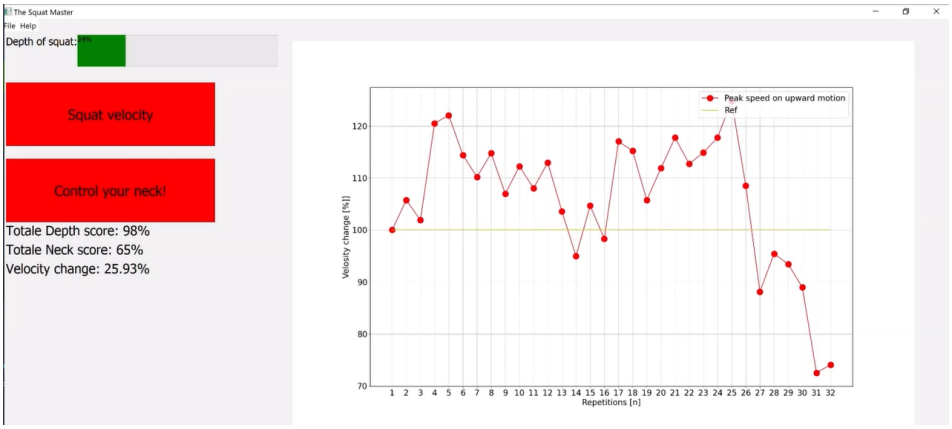


Figure 4.25: A screenshot showing the end of the session.

From Figure 4.25 it could be observed that 32 squats were performed, with 32 peak velocities shown as red circles in the graph section of the figure. A yellow vertical line from the first peak velocity illustrates the reference for all the other peak velocities. The first peak velocity represents 100% velocity, however, as shown in the plot, the peak velocities could be larger than the determined 100%.

Upon closer examination of the graph section, it was evident that the 27th squat experienced a peak velocity loss of over 10%. This corresponded to the velocity loss of 11.88% and the orange box labeled *Squat velocity* shown in Figure 4.23 and Figure 4.24. The next squat had a smaller velocity loss than the 27th squat, meaning that the user increased the velocity after the feedback from the orange box was received. At the 31st squat, the velocity loss was greater than 20% meaning the requirement was met for the user to take a break immediately. However, as the feedback was provided at the end of the squat, the user performs one more squat before ending the session. This corresponds to the screenshot shown in Figure 4.24.

In Figure 4.25 the implementation of the total depth score and the total neck score is displayed, in this particular test the scores amounted to 98% and 65% respectively. From observing the full video the screenshots were taken from, an impression of how well the user performed in regard to the depth and the neck position was formed. The scores seemed to represent the impressions gotten.

A new squat session was performed by another user. This was done the same way as the one already presented in this section, with similar continuous feedback results. The main difference was the length of the session, as well as the user generally having a deeper squat and worse neck position off each repetition when compared to the first test. The end of the squat session of the new squat performed is shown in Figure 4.26.

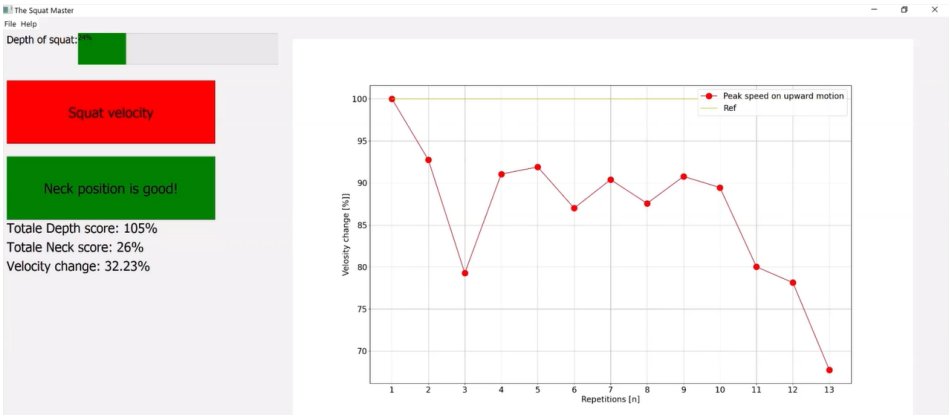


Figure 4.26: A screenshot showing the end of the new session.

From Figure 4.26 it could be seen that 13 squats were performed. The 3rd squat fulfilled the 20% velocity loss, however, due to the user not observing or getting this information the session continued. The 11th squat was exactly 20% velocity loss and due to how the logic was implemented the box labeled *Squat velocity* was displayed as orange and not red. The next squat, however, had a larger velocity loss and the box was displayed as red. Similar to the previous test, the information regarding velocity loss was provided after the squat was completed, leading to an extra squat being performed.

The total depth score and total neck score are displayed as well in Figure 4.26 as 105% and 26% respectively. This shows that both the depth and the total depth score could be displayed as more than 100%. These results were also consistent with the impression gathered from the full video.

Chapter 5

Discussion and future work

The goal of this thesis was to develop a continuous feedback system, using body movement identification techniques, to assist users in exercising correctly and efficiently. The system involved monitoring the user's posture and velocity, providing feedback on when to take breaks and guidance on proper lifting techniques. It also included a score to evaluate the exercise and helpful feedback on improving squats.

A laptop application has been developed to provide users with accurate feedback. The implementation of VBT works as intended, indicating when a break is needed with a low error rate of 0.008% for one peak velocity and a standard deviation of ± 0.57 when examining velocity drop. The posture estimation feature has also been implemented, accurately measuring the depth and neck angle, with an error in depth measurement equivalent to 10% and an error in neck measurement of 5°, with a good neck angle defined within a specific range.

After conducting research on existing solutions utilizing similar approaches, it is evident that the solution proposed in this project holds value. Unlike other solutions, this project combines the velocity-based training aspect with posture estimation, making it a unique and innovative combination. Additionally, the effort put into creating a user-friendly, easily accessible, and cost-effective application for general users further enhances the usefulness of the developed application.

This chapter thoroughly examines key aspects of the application. It discusses the assumptions that were made, and potential problems with the developed solution, and compares it to other existing solutions and alternative approaches that could have been used. Additionally, it provides suggestions for future development.

5.1 The input

The purpose of the input is to collect ample data regarding the user's movement in order to accurately calculate their velocity and provide a reliable pose estimation. While a camera solution was implemented, several alternative options were also explored that would achieve the same objective.

A similar design of the *AlphaPWR* mentioned in Chapter 1.2 *Existing solutions*, is an excellent choice of input if only the data of the user's velocity is considered. Although it can be argued that a more affordable and user-friendly design can be developed using this concept, it would still require additional equipment and be less user-friendly compared to the solution developed in this thesis. Alternatively, a smartwatch or something similar with a built-in accelerometer could be used to calculate the velocity of the user. This combined with a screen for the feedback system could provide the user with a good application for the velocity-based training part.

For posture estimation, one approach that was considered was to use an inertia measurement unit (IMU). This would have the potential for precise measurement ensuring a good posture estimation, however, this approach would require more equipment and calibrations for the end user compared to the developed solution in this thesis.

The difference between the mentioned inputs above and the selected one is mainly in terms of user-friendliness and convenience. Moreover, the chosen input can serve both the velocity and posture aspects of the application. With the camera input, users can easily launch the application and receive feedback from the camera stream, complete with the virtual skeleton and other helpful information, which will guide them on what actions to take. Furthermore, since smartphones are prevalent, an application for smartphones could be made as a future development, creating an app that users

can conveniently access on their devices. This approach ensures that users can readily operate the application and access it whenever they need it. This input method gives users the benefit of receiving real-time updates and feedback on their phone screens, making it user-friendly and accessible.

5.2 Human pose estimation

During the research of human pose estimation, various versions were evaluated such as *You Only Look Once* (YOLO) and Mediapipe.

While Mediapipe can only detect the pose of one person, YOLO has the advantage of detecting multiple poses. However, since this application was designed to focus on only one person, made this feature was unnecessary. YOLO has the capability to use both the graphics process unit (GPU) and central process unit (CPU) for pose estimation, unlike Mediapipe which only uses the CPU. [29] Despite this, Mediapipe was chosen for its higher accuracy.

Ultimately, Mediapipe was selected for this project due to its higher number of landmarks and perceived accuracy and stability. [29] Additionally, Mediapipe's website claims that their service can be utilized on multiple devices including computers, tablets, and smartphones, making it an attractive option. [15] The application was aimed toward the possibility of being used on smartphones in for future development, making this a desirable feature.

Collecting the landmarks

When collecting the landmarks found by Mediapipe, a problem arrived. As Mediapipe has so many landmarks it could take a long time to log all of them. As previously stated in this thesis, there was both used the *pose landmarks* and the *pose world landmarks*, with 33 landmarks for each category. The application, therefore, needed to catalog 66 different landmarks into various lists. Combined with the processing time Mediapipe needed to estimate the landmarks, meant this would result in latency for the application. This meant that the video presented to the user would lag.

The solution to address this issue was by implementing threading and down-

sampling techniques, which could reduce the workload and allow other sections of the code to run concurrently. The code was split up into 4 different threads; interface, gesture, pose estimation and collection of landmarks, and calculations for the squat estimation to try and make it faster and non-interruptive. This helped some, as the application now worked well but, the video feed still came in slow and lagging. The video format sent into the applications was 30 FPS and it was concluded that Mediapipe did not need that many frames, so instead of processing all the images only every other image was processed. By doing this the application then had more time to process and collect the data. This meant that the video presented got less lagging. However, when the program ran for a longer period, the dictionary containing the data started to become quite big, which again resulted in lagging. To solve for this a solution was put in where old data would get deleted when the dictionary became larger than 100 data points per body part landmark.

Although this approach helped reduce lagging, it introduced a new issue. The application began to experience problems with velocity calculations, resulting in different velocity outcomes for each run of the same input video. To address this, each landmark collected was assigned an index size, ensuring that the time between frames was consistent for every calculation. This helped minimize errors. Based on an external test conducted outside the application, no errors were found, leading to a fair assumption that the results from the application should yield the same outcome. However, despite conducting emulative analyses to identify the flaw, no issues were found with the correct FPS or frame distance. However, a small SD still persisted, and the reason for this remains unknown.

5.3 Velocity-based training

The velocity-based training seemed to be implemented correctly. As seen from the tests done in Chapter 4.2 *Testing of velocity*, every maximum average velocity of interest was found, and a significant enough velocity drop was found. This would mean that, in the application, the user could be notified of when to take a break. Although there was some slight uncertainty and variation in the data, the test results showed that the outcome was highly accurate and dependable.

While the implementation of velocity-based training could be seen as successful, there were certain assumptions made when determining the region of interest, which is discussed in detail in the next section. Furthermore, the process of determining the velocity is also elaborated upon.

5.3.1 Finding the region of interest

To identify the ROI, the difference between the present and past height of the person was analyzed, the sign was obtained, and the direction of the squat was found. It is worth noting that this approach relies on certain assumptions.

After conducting several tests, it was concluded that incorporating the current height and the height from 10 frames earlier would produce satisfactory outcomes. Though the ROI was consistently located despite this assumption, implementing an adaptable approach to determine the required number of frames between the current height and an earlier height would enhance the dependability of the final product.

To confirm that the identified variation was indeed a squat and not just a minor movement in the shoulder, a buffer of 2% of the user's maximum height was deemed appropriate. This measure helps eliminate any potential noise or irrelevant data independent of the distance between the camera and the user. The buffer was determined through trial and error and should be tested more extensively to ensure that it functions as intended.

5.3.2 Calculating the velocity

In the thesis, a solution was proposed that involved calculating average velocity utilizing the theory from Chapter 2.5 *Finite difference method*, using the derivative, height difference, and FPS. Although the average velocity was accurately calculated, there was susceptibility to noise, requiring the implementation of a smoothing filter. An assumption made here was about the linearity between the position point of interest simplifying the equation for the average velocity(Equation(3.1)).

To explore more options, it is worth considering other approaches, such as a Kalman filter, to compare the velocity results with the method used in the project. Using this information as an argument, the most effective method can be decided upon. Alternatively, the velocity results obtained in the project could be verified by using a smartwatch or a similar device that has a built-in accelerometer.

5.4 Posture measurements

During the posture testing, accurate feedback was provided to the user through the depth calculations, which allowed them to receive a score. Although the neck calculations were slightly unstable, they still proved useful in indicating the correct neck placement.

The assumption of the camera needing the side view of the person side view proved wrong according to the test conducted in Chapter 4.3.4 *Testing errors of mono camera-based measurements*. According to the test, a front-facing view might also work, however, this would need to be tested for the other calculations done in the application, for instance, the neck angle.

The goal of monitoring the user's posture to lift correctly, as described in Chapter 1.3 *Problem definition*, can be considered achieved. However, only the depth of the squat and the neck angle provide feedback. The assumptions, problems, and theory behind the posture measurements are discussed and elaborated upon.

5.4.1 The theory behind the squat

In this project, the book written by Mark Rippertoe, *Starting Strength*[2], laid the foundation for how to rate the posture of the squat. In researching the subject, there was found many different sources of information about the squat.

These different sources often talked about the same six subjects to think about when performing a squat; depth, knee position, stance, eye gaze, back angle, hip drive, and bar placement. However, the different sources often had misleading and/or contradictory information in relation to each other about the subject. In addition, the authors of this thesis do not have any sort of background in the theory surrounding exercise. A decision was therefore made use the book by Mark Rippertoe as the only source of information.

With only one source of information, however, the posture part of this application would work as a proof of concept, and not act as the only

source of information needed to perform the squat as originally intended. However, with more research on the subject and a corresponding change in the parameters set in the application, the posture part of the application could prove to be a useful source of information and potentially remove the need of getting information about the posture from other sources for the user.

5.4.2 The depth estimation of the squat

When looking for a solution for the depth estimation, multiple solutions were considered, like checking the knee angle and looking at the height change of the user, but since a good squat was defined as parallelity of the height and the ground/foot, the solution used seemed like the most effective. When the depth estimation was done, some assumptions were made. One of these assumptions was that the line created by Mediapipe, between the hip point and the knee point, could represent the femur. This was done in order to more easily apply the theory of the squat.

The hip point found by Mediapipe tends to be defined a little higher on the body than expected, as represented by an extreme case in Figure 5.1 taken from Chapter 4.2 *Testing of velocity*, where the velocity equation was tested.

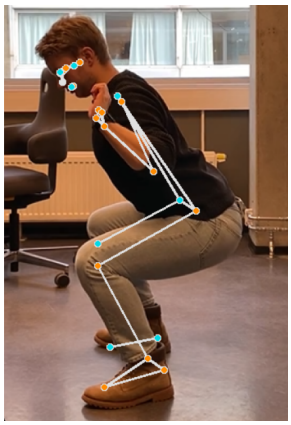


Figure 5.1: Figure taken from Chapter 4.2 *Testing of velocity*, illustrating the potential problem with the hip point.

If the hip point was taken as high as the example from Figure 5.1 the assumption made about the line corresponding to the femur would be wrong.

Chapter 2.2 *The theory behind the squat* stated that when the femur was parallel to the ground, the squat was deemed deep enough. This would mean that if the hip point was located too high, the user would technically be notified later than necessary when the user has a femur lower than parallel to the ground. From the depth test in Chapter 4.3.1 *Testing the depth function* it was found an error of 10% at the lowest depth position, this could be seen as a significant error which could be rectified by adding a bias as to define the lower position on the simulated test as 100% parallel to the ground. However, as it was found through research in Chapter 2.2 *The theory behind the squat*, a good depth is when the femur is parallel to the ground, but the lower down could be seen as better. Meaning an error of 10% was deemed acceptable.

With this said, the application makes sure that the user is notified when performing a partial squat, and could therefore be seen as a success. However, finding a better estimation of the femur is desirable as this would better the overall quality of the application.

5.4.3 The neck estimation

Some issues arose after determining the neck angle. As per the test conducted in Chapter 4.3.3 *Testing neck angle with simulated input*, it was discovered that the shoulder movement had an adverse impact on the outcome. Since the calculation requires the hip, shoulder, and ear to determine the neck angle, moving the arms shifted the shoulder, resulting in an inaccurate estimation. It is possible that the simulated character from the test moved their arms more than an average person would, but this was still considered a potential problem. One possible remedy could be to identify other points that could be used for estimation. The ear to eye line could be used to track the user's gaze, but it was observed that these points moved a bit. Therefore, it was decided not to focus on this, although it could work with filtering of the data points. Currently, the user must keep their arms still to fully benefit from neck posture monitoring, making it a less than ideal solution.

In Chapter 4.3.2 *Estimating a good neck angle*, an evaluation was carried out to determine the acceptable range of neck angles. This was done in combination with the test mentioned above, and provided a rough estimate of the required border limits. However, in retrospect, the test should have been conducted on a larger sample size to gain a more accurate understanding of the ideal border limits. This would have resulted in more data to analyze and a better comprehension of what constitutes a good neck angle for the user.

5.4.4 The missing posture estimations

Chapter 2.2 *The theory behind the squat* states that the feet should be flat on the ground with the heels shoulder-width apart. The feet should point about 30°, with the femur parallel to the feet. Due to the limitations of the mono camera-based measurements, and the fact that the camera needs the side profile of the user, the application does not have any way to estimate this stance. In addition, the posture estimations for the knee position, back angle and bar placement were not implemented. This means that the monitor, score, and feedback of the stance estimation, knee position, back angle, and bar placement of the user's posture are not complete.

It was considered to use the heel and toe landmarks to monitor that the foot stayed in the same place throughout the squat. On closer inspection of the landmark's behavior during the pose estimation session, it was found that there was not reliable as that moved around a lot. Making the method unreliable. Regarding knee placement, the toe landmark might be stable enough to check the difference between the knee landmark and the toe landmark to get an estimation that the knee is in the correct area.

For the back angle and the bar placement, the Mediapipe library might be too limited to check this directly using only Mediapipe. For the bar placement, a combination of Opencv and Mediapipe might work. Where Opencv could be used to find the circle of the bar and use the position of the bar to see where it is placed in association with the Mediapipe's generated skeleton. The back angle might monitored by using image processing and edge detection to find the curvature of the back, this can however introduce other problems where the human body comes in many shapes which could make concrete rules for the curvature hard to set.

With the bar, however, the shoulder could be obscured by the weight of the bar. This is unfortunate as the shoulder is used in calculating the height, the neck angle estimation as well as the velocity calculation. A solution for this is to identify when a bar is used and alter the calculations accordingly using the midpoint of the bar's side view, instead of the shoulder. Testing this to ensure reliability is necessary.

5.5 The application

The user interface and scoring system both work as intended, however with some problems. The user interface gives all the information required for the user to take a break when needed, and also on what in the movement needs improvements. However, by having the feedback on a screen that is positioned in an unfortunate way for the user, the feedback could be challenging to observe. In addition, the feedback regarding when the user needs to take a break comes after the squat that meets the requirement. As seen in the tests preformed in Chapter 4.4.2 *User interface and scoring system*, this could lead to the user doing more squats than recommended. In fact, a velocity loss of 27.49% was found at the end of one of the sessions. Furthermore, the gesture control proved to be less than ideal for the user.

5.5.1 The continuous feedback system

The user interacts with the program and receives feedback and scores through the application. While it functions properly, there are some limitations and design choices that could be improved.

It was assumed that the user remains in a fixed position relative to the camera while performing each set of exercises. While this simplified the necessary calculations to provide feedback, it also limited the program's adaptability as the user would be constrained. In addition, as the screen had the camera in close proximity this created a challenge for the user to view the feedback while squatting due to the necessity of the camera angle relative to the user. The user had to turn their head to see the screen, making it less than ideal. An alternative way of providing the feedback

system could be to use auditory cues.

The idea would be to mimic a personal trainer. The user could wear a set of headphones connected to the device where the application would be running, and when the different conditions are met, the user would get an auditory cue. For instance, when the user had completed a deep enough squat, the auditory cue could be a voice that says*up*. When the condition for the velocity-based training had been met, the auditory cue could be *break*.

The use of auditory cues combined with a visual summary after each set could prove useful as a continuous feedback system. This would eliminate the need for the user to always look at the screen, but still provide the necessary feedback in real-time.

The feedback displays were selected at the discretion of the authors. In order to make sure that the feedback was easily comprehensible for the average user, several tests should be conducted with a diverse group of participants who could provide their opinions. Based on their feedback, the design should be improved to ensure that the feedback ended up intuitive enough to be understood by any general user.

5.5.2 The gesture control

To enhance user experience, gesture control was implemented to facilitate starting and ending sessions. As a result, users should be able to easily launch the application from any device it's operating on, without having to physically navigate to the device. During the gesture testing, initially, promising outcomes were observed, leading to the conclusion that the gesture was functioning as intended. However, it was later discovered that this was due to a flaw in the testing conditions rather than an accurate representation of the gesture control's performance. Specifically, the images used to evaluate the recognition accuracy of the gestures were taken from close proximity to the hand. When tested from further away, where the user would typically be positioned in a squatting session, the gesture recognition did not function effectively.

As the recognition worked from close proximity, it was possible that the

problem lies in that the gesture recognition gets too much information when the whole body of the user was visible. It may be worth testing whether cropping the input image to only show the hands and sending this information to the gesture recognition system would solve the problem. Another option could be to consider a more user-friendly method for starting and stopping the session. For example, if a smartwatch is used in future development, the user could simply press a button to initiate or terminate the session.

5.6 Further future work

In addition to the already mentioned suggestion for future work, while discussing the results, there are a lot of improvements that could be made. The ones mentioned in this section are implementation for providing the user with relevant instructions, the accessibility, the depth perception, as well as the possibility of using different methods to identify a movement and identify different movements.

5.6.1 Instructions to user

Based on Chapter 1.3 *Problem definition*, one of the points made was that the application should *give feedback on the exercise, and provide useful information on how to improve.*

The application provides feedback in the form of depth score and neck score, which can help users improve their squatting technique. However, there is currently no user guide available, nor any information on what constitutes a good squat. To address this, the application could include an instructional popup upon startup, which users can dismiss once read. Additionally, a help drop-down bar could offer more detailed guidelines on the app's functionality and how to optimize performance.

5.6.2 Accessibility

The accessibility of the applications is argued to be of huge importance, as the accessibility could be a factor for both people that would like to start exercising, but also for people that want to continue their exercise routine.

As the application stands, the program needs to use a computer (laptop) to work. This is less than ideal, as the computer is not always easily available. This is especially true if the user would like to combine the application with the use of a gym.

As mentioned in this thesis, the application was made with the use of smartphones as a potential future development, making it more accessible to the general public. This would mean that the application needs to be rewritten to fit the format of smartphones. As most people have their smartphone with them at all times, the hope is that it becomes much easier to begin and maintain an exercise routine.

5.6.3 Depth perception

Finding depth through mono-camera-based estimation can be challenging. Although there are techniques to extract this information, employing additional cameras appears to be the simplest solution.

To enhance the accuracy of depth estimation, the application could use two cameras. This would be an improvement on the current application, as the stance estimation would be possible to calculate. With the inclusion of stance estimation in the monitoring process, the user's posture can be evaluated, scored, and provided with feedback. However, using two separate cameras would make the application less user-friendly and more challenging to operate.

By reformatting the application for smartphones, users can utilize the stereo-camera system on their device to capture depth information from the side profile. This enhancement maintains the user-friendly aspect of the project's solution while also providing additional depth information.

A feasible method of achieving stance estimation through the stereo-camera system is by evaluating the depth discrepancy between the toe and heel points. The said difference can then be utilized to compute an approximation of the foot angle. Additionally, the knee and hip points can be evaluated in the same manner, and by comparing the two angles, it is possible to determine if the user is performing an acceptable stance. This method assumes that if the user's stance is correct on the side facing the device, then their stance on the opposite side facing away from the camera is also correct.

Both methods mentioned would make sure that the monitor, score, and feedback of the stance estimation of the user's posture is completed.

If the accurate depth is determined, it could unlock several potential opportunities for future development. One such possibility is utilizing the user's depth information to generate adaptable calculations based on depth. This would remove the need to assume that the user is always in the same position relative to the camera.

Another possibility is to use the depth information to compute a velocity calculation that is more comparable. This would enable the user to compare exercise sets more easily, such as comparing the velocity of their current exercise with their previous workout.

5.6.4 Velocity zones

In Chapter 2.1 *The theory of velocity-based training*, two approaches were discussed; applying velocity losses of 10-20% and incorporating velocity zones into separate or combined training programs. While the velocity loss method was effective for a wider range of users, velocity zones could be provided as an alternative for individuals looking to tailor their workouts to their specific needs.

One way to achieve this is by including an extra feature in the application, which would guide the user through the necessary tests to determine their individual 1RM. After that, a technique should be developed to determine the required repetitions at different predetermined relative or absolute loads. This would establish the basis for the individual's velocity zone.

5.6.5 Different methods to identify the squat

In this thesis, a method to determine whether a squat is executed correctly or not by using Mediapipe and posture estimation is identified. While this method generally works well, it has limited flexibility. As an idea to extend the proven concept, methods using a predictive feedback system and creating a new neural network are proposed.

The proposed predictive feedback system involves creating a model of a specific exercise, such as the squat, and analyzing performance data to determine how closely it aligns with the predicted outcome. To ensure accurate predictions, the model must be flexible enough to adapt to the initial data gathered during the exercise. This data may include details on the initial speed and posture, which will help determine the trajectory of the movement.

Some suggestions for creating a predictive feedback system are using Python's Dynamic Mode Decomposition (pyDMD) or its sparse identification of nonlinear dynamical systems. [30][31] These methods are known for identifying and analyzing nonlinear dynamical systems, which can help create a mathematical model and system states. By utilizing this information, a predictive feedback system can be developed. Additionally, having the system states and mathematical expressions could help find the velocity. When combined with a well-designed model and appropriate assumptions, this approach could be a promising option for future work.

Another approach is to have the user perform several squats and assess their form using a scoring system, similar to the methodology employed in this project. Once sufficient data has been collected on the user's ideal squat form, it can be used as the benchmark for subsequent squats, with each squat being compared to the model.

5.6.6 Identify different movements

Only the squat movement was identified in this project. However, by expanding the range of movements detected, the application could become more helpful and provide users with a variety of exercises to choose from,

along with relevant information and assistance.

To achieve this, the same methodology used for squats in this project could be utilized, using Mediapipe and the geometry of the lines formed between the pose landmarks to obtain a posture estimate. Alternatively, we can apply comparable techniques as discussed in Chapter 5.6.5 *Different methods to identify the squat*, to identify the various exercises.

For exercises where the velocity detected corresponds directly to the power lifted or pushed, velocity-based training should be effective. Nonetheless, further research is needed to determine how the velocity can correspond to power for specific exercises.

Chapter 6

Conclusion

This thesis presents the development of an application designed for laptops. It incorporates theories on velocity-based training, human pose estimation, and the squat exercise to offer users helpful guidance and feedback. The application focuses on three key aspects: velocity-based training, posture estimation, and a continuous feedback system.

The velocity-based training laid the foundation for providing feedback to the user on the intensity level of the training and when to take a break. This has been implemented successfully, with an SD corresponding to an error of 0.008% for one peak velocity and the largest SD found when examining the velocity drop amounted to ± 0.57. There are, however, some assumptions made that could be a potential problem. The velocity is found by calculating the maximum velocity of the ROI, which corresponds to the upwards motion of the squat. This ROI is found by using the current height and the height found 10 frames in the past to determine the direction of the squat. This proved successful in all the tests conducted, however, implementing an adaptable and more robust approach is desirable. In addition, to avoid any possibility of interference from noise or slight movement in the shoulder that may impact the recognition of the ROI, a buffer of 2% of the user's maximum height was incorporated. The buffer size was established through trial and error and requires further testing for optimal performance.

The posture estimation was implemented as a way to give the user reliable and continuous feedback on how to perform and improve the movement of the squat. The depth estimation works as intended and gives feedback to the user when performing a deep enough squat with an error of 10%. There is, however, an assumption about the femur approximation that has

the consequence of telling the user that a good squat is achieved later than necessary. The neck angle was found with an error of 5° and an acceptable area with an upper and lower limit was determined. The feedback works as intended, however, the determined limits should be tested further with a larger sample of people. This would ensure that the limits, and therefore the feedback, are more versatile and reliable.

Only the depth estimation and neck angle estimation were implemented. This means that the posture part of the application is incomplete as the application does not give any feedback on how to perform and improve the stance, knee position, back angle, and bar placement.

The overall goal of this project was to make an application that would use body movement identification methods to define a continuous feedback system to help the user exercise correctly and efficiently. The user velocity was successfully monitored, and feedback was provided on when to take a break. In addition, the user's posture was also monitored, and feedback was provided to help the user lift correctly. However, the posture feedback has some key elements missing from the system. A score was delivered to the user on how well the exercise was carried out and feedback was provided, however, some key elements from the posture estimation are missing here as well. The overall goal is achieved, however, with some areas in need of improvement.

During the assessment, several recommendations were proposed for future development. One of the suggestions is to incorporate posture feedback that was previously missing. Implementing accurate parameters using multiple sources of information would greatly contribute to achieving this goal. In addition, improving the feedback system with auditory cues can enhance the application's quality and usability. Furthermore, the application's accessibility could be enhanced by creating a mobile version, enabling users to perform the exercises anywhere they go.

An effective way to enhance the velocity-based training part of the application is to include velocity zones as a feature in the application. This option would allow users to customize their workout according to their individual requirements. Different methods and identifying different movements could enhance the application significantly, transitioning from a limited squat-specific design to an all-encompassing exercise app.

Bibliography

[1] Statistisk sentralbyrå (SSB). Sports and outdoor activities, survey on living conditions. https://www.ssb.no/en/kultur-og-fritid/idrett-og-friluftsliv/statistikk/idrett-og-friluftsliv-levekarsundersokelsen. accessed: 01.02.2023.

[2] M. Rippetoe. *Starting strength*. The Asgaard Company, 3 edition, 2012. p. 7-70.

[3] Ireland. D, Wang. Z, Lamont. R, and Liddle. J. Classification of movement of people with parkinsons disease using wearable inertial movement units and machine learning. https://pubmed.ncbi.nlm.nih.gov/27440290/. accessed: 17.03.2023.

[4] Alphatek. Alphapwr. https://www.alphatek.no/alphapwr. accessed: 30.01.2023.

[5] Virtuve. Unlock your athlete's potential. https://vitruve.fit/. accessed: 14.07.2023.

[6] Metric Jacob Tober. Reliability and validity of metricvbt beta. https://www.metric.coach/articles/reliability-and-validity-of-metricvbt-beta. accessed: 26.02.2023.

[7] Pradnya Rajendra Patil. Squat angle detection using opencv and mediapipe. https://github.com/Pradnya1208/Squats-angle-detection-using-OpenCV-and-mediapipe_v1. accessed: 08.06.2023.

[8] Mladen Jovanović and Dr Eamonn P. Flanagan. Researched applications of velocity based strength training. https://scholar.

google.com/scholar_lookup?title=Researched+applications+of+velocity+based+strength+training&author=Jovanovi%C4%87,+M.&author=Flanagan,+E.&publication_year=2014&journal=J.+Aust.+Strength+Cond.&volume=22&pages=58%E2%80%9369. accessed: 02.02.2023.

[9] M. Izquierdo, J J González-Badillo, K. Häkkinen, J. Ibáñez, W J. Kraemer, A. Altadill, J. Eslava, and E M Gorostiaga. Effect of loading on unintentional lifting velocity declines during single sets of repetitions to failure during upper and lower extremity muscle actions. http://www.thieme-connect.de/products/ejournals/abstract/10.1055/s-2005-872825. accessed: 06.03.2023.

[10] L. SÁNCHEZ-MEDINA and J J GONZÁLEZ-BADILLO. Velocity loss as an indicator of neuromuscular fatigue during resistance training. https://journals.lww.com/acsm-msse/Fulltext/2011/09000/Velocity_Loss_as_an_Indicator_of_Neuromuscular.16.aspx. accessed: 06.03.2023.

[11] M. Włodarczyk, P. Adamus, J. Zieliński, and A. Kantanista. Effects of velocity-based training on strength and power in elite athletes—a systematic review. https://doi.org/10.3390/ijerph18105257. accessed: 30.01.2023.

[12] Daniel A. Hackett, Timothy B. Davies, Rhonda Orr, Kenny Kuang, and Mark Halaki. Effect of movement velocity during resistance training on muscle-specific hypertrophy: A systematic review. https://www.tandfonline.com/doi/full/10.1080/17461391.2018.1434563. accessed: 02.02.2023.

[13] Aristide Guerrieri, Carlo Varalda, and Maria Francesca. The role of velocity based training in the strength periodization for modern athletes. https://www.mdpi.com/2411-5142/3/4/55. accessed: 02.02.2023.

[14] Amrutha K, Prabu P, and Joy Paulose. Human body pose estimation and applications. In *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–6, 2021.

[15] Google. Mediapipe. https://mediapipe.dev/. accessed: 24.04.2023.

[16] Mediapipe. Pose landmark detection guide. https://developers.google.com/mediapipe/solutions/vision/pose_landmarker. accessed: 03.06.2023.

[17] Google. Mediapipe pose. https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md. accessed: 24.04.2023.

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[19] Google. Hand landmarks detection guide. https://developers.google.com/mediapipe/solutions/vision/hand_landmarker. accessed: 24.04.2023.

[20] Steven Paul. Bmw gesture control: The quick how-to guide. https://www.bmwblog.com/2022/10/07/bmw-gesture-control-how-to-guide/. accessed: 14.06.2023.

[21] Techvidvan. Real-time hand gesture recognition using tensorflow & opencv. https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/. accessed: 29.05.2023.

[22] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[23] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[24] Peter Moczo, Jozef Kristek, and Ladislav Halada. The finite-difference method for seismologists. *An Introduction*, 161, 2004.

[25] Lee Dong Kyu, In Junyong, and Lee Sangseok. Standard deviation and standard error of the mean. *kja*, 68(3):220–223, 2015.

[26] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[27] Blender. Blender 3.5. `https://www.blender.org/`. accessed: 08.06.2023.

[28] Mixamo. Air squat on bryce. `https://www.mixamo.com/#/?page=1&query=squat&type=Motion%2CMotionPack`. accessed: 08.03.2023.

[29] JG Vishnu and SJ Divya. A comparative study of human pose estimation.

[30] Nicola Demo, Marco Tezzele, and Gianluigi Rozza. Pydmd: Python dynamic mode decomposition. *Journal of Open Source Software*, 3(22):530, 2018.

[31] Brian M. de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, 2020.

Appendix A

Poster

The poster from the poster presentation is included in the page below.

Body movement identification

Defining a continuous feedback system to be used for squatting correctly and efficiently.

Authors: Ådne Hult Karlson and Stian Wiik Berg

University of Stavanger

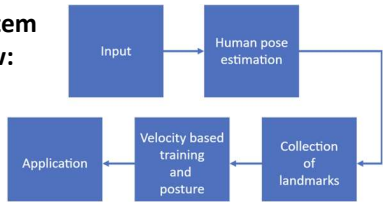
Introduction/motivation

- Different training guides and videos online that give contradictory information. Frustrating for the people that want to perform a squat, but also overwhelming for people that want to start exercising.
- Using Velocity-based training and posture estimation to create a versatile and user-friendly application using a camera-based solution.

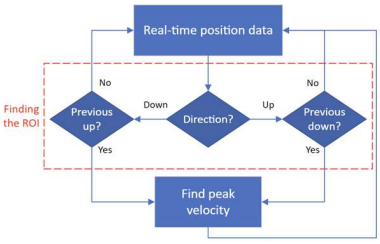
Alphatek's *AlphaPWR* is an existing product utilizing velocity-based training

Method

System flow:



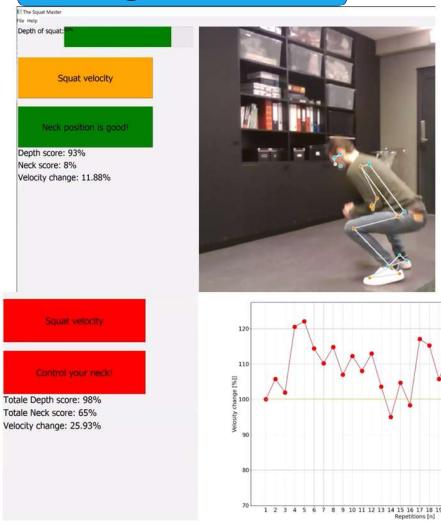
Velocity-based training:



Posture estimation:

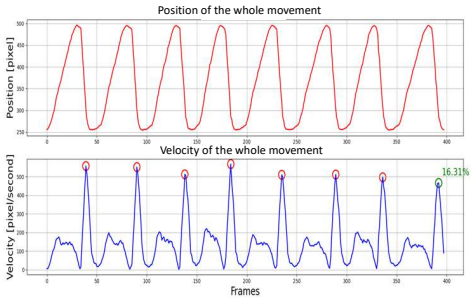
- Depth of squat
- Neck angle/position

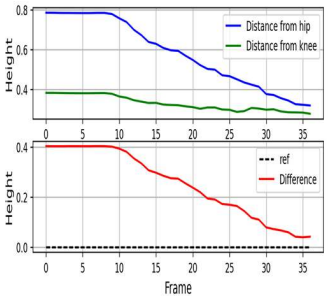
Testing and results



- Feedback on the depth using a progress bar.
- Feedback on the neck
- Squat velocity is monitored and updated for each squat. Yellow means close to finished. Red means take a break.
- Depth score, neck score, and velocity change for each squat in %

- Total depth and neck score at the end of the set.
- Plot showing all the peak velocities in the set.





Conclusion

- The application improves the squat, telling the user when to take a break and some posture feedback.
- The posture estimation is missing some key posture feedback for instance the stance, knee position, and an estimation of the center of gravity.

Appendix B

Gesture table

	Okay	**Peace**	**Thumb up**	**Thumb down**
Okay	0.83	0	0	0
Peace	0	0.91	0	0
Thumb up	0	0	1.0	0
Thumb down	0	0	0	0.986
Call me	0	0	0.22	0
Stop	0	0	0	0
Live long	0.003	0	0	0
fist	0	0	0	0

	Call me	**Stop**	**rock**	**Live long**	**Fist**	**Smile**
Okay	0	0.17	0	0	0	0
Peace	0	0	0.014	0.074	0	0
Thumb up	0	0	0	0	0	0
Thumb down	0	0.005	0.009	0	0	0
Call me	0.77	0	0	0	0	0.008
Stop	0	0.39	0	0.61	0	0
Live long	0	0.186	0	0.722	0	0.089
fist	0	0	0	0	1.0	0

Table B.1: Here is a breakdown of the results for each gesture test. The first column displays the names of the tests, while the first row lists the names of the gestures found.