Automatic Exercise Classification

Development and Accuracy Evaluation of a Deterministic Algorithm

Bachelor Thesis
Faculty of Science, University of Bern

submitted by

Aris Konstantinidis

from Athens, Greece

Supervision:

PD Dr. Kaspar Riesen
Institute of Computer Science (INF)
University of Bern, Switzerland

Abstract

The present study investigates the potential of computer vision for automatic tracking of physical activities. Research in this field has focused on systems that can recognize a fixed set of exercises, without the ability to track a subject's range of motion (ROM) during a particular exercise. We present a system that is able to learn a new exercise by only observing one initial repetition and track ROM in real time. Arguably, such a system could automate the quantification of training regimes, which can be used to assist a wide spectrum of trainees, ranging from bariatric patients to top athletes. By developing a mobile application prototype, we show, that a smartphone's front facing camera can be used as a basis for real-time classification of exercises. Moreover, we measure the system's accuracy by setting up a standardized testing framework. The classification method is evaluated by testing 7 common fitness exercises. After the system has learned to recognize these movements, two subjects execute a short workout that contains repetitions from each exercise. By optimizing the classification algorithm's 6 tuning parameters, the system achieves a exercise recognition accuracy of 90.7%. In addition, the system is able to track the subjects' ROM during exercise execution with an average precision of 83.9%. Although testing data is limited due to the time-costly nature of manual labeling, and the prototype only reaches its full potential under certain conditions, the results make a promising argument for a future application that is able to reliably quantify workouts.

Acknowledgements

I am grateful to be able to do research on a system that has the potential to improve people's lives, by assisting them in their physical activities. Yet, this would not be possible without the support and encouragement of the people around me. Specifically, I want to thank Professor Riesen for his trust in my work, my girlfriend Nadia for her calming effect on me, my mother Sofia and my father George for always believing in me, and my brother Pavlos, with whom during endless nights, we co-developed a full featured version of the mentioned system.

Contents

Chapter 1

Introduction

The quantification of physical activity has been the focus of numerous research articles, demonstrating a plethora of benefits for various patient groups, including those with diabetes [1] and bariatric patients [2]. By quantifying a trainee's workout, progress can be objectively measured, evaluated, and adjusted according to their needs. This quantification also enhances trainees' awareness of their overall movement quality.

One promising field for automatic movement quantification is computer vision. For instance, Yu Chen et al. [3] developed a system based on deep learning of images that could reliably detect the execution of several fitness exercises. Another approach in computer vision involves the calculation of body nodes. In this paradigm, key body landmarks are extracted from videos and processed for further analysis. Zhou et al. [4] utilized this technique to assess the similarity between previously recorded exercises and real-time execution.

Despite the advancements in this field, existing research systems have several shortcomings that reduce their usefulness in the fitness industry. Systems using deep learning of images are not scalable, as they only recognize a few exercises they have been trained on. The addition of new exercises or individualized variations would require extensive training on a vast amount of new labeled data, which may not be readily available. On the other hand, systems based on extracted body landmarks are often specialized for specific movement types [5], focus solely on pose extraction optimization, or do not track the Range of Motion (ROM) executed in real-time.

In this study, we will present a prototype solution capable of recognizing arbitrary exercises by recording only one initial repetition and tracking the current ROM in real-time. The research goal is to systematically test and analyze the system's accuracy in these tasks. By doing so, we aim to encourage the development of industry-oriented applications that can assist trainees in achieving their fitness goals. Although the prototype development is outside the study's scope, we will

explore the main topics influencing its accuracy.

Chapter 2 will delve into the fundamentals of HPE, exploring its applications and the role of ROM in assessing movement quality. Understanding these concepts is crucial for developing systems that accurately quantify physical activity. The chapter will also highlight the importance of measuring physical activity in different populations, such as patients with chronic conditions or those undergoing rehabilitation.

Chapter 3 will outline the conceptual framework of the proposed prototype. It will discuss the integration of various components that contribute to the system's accuracy. This chapter will emphasize the need for a holistic approach in designing and evaluating such systems, considering factors like user variability and environmental conditions.

In Chapter 4, we will detail the experimental setup used to validate the prototype. This section will cover the methodology for data collection, the metrics for evaluating accuracy, and the statistical analysis of the results. By testing the system, we aim to establish its reliability and practical applicability.

The final chapter will synthesize the study's findings, addressing the research questions posed at the outset. We will discuss the system's limitations, such as potential inaccuracies in different environments or with varying user demographics, and propose directions for future research. Enhancing the system's robustness and expanding its applicability will be key areas for further investigation.

Chapter 2

Basic Concepts

In the following chapter, the field of Human Pose Estimation (HPE) will be introduced, the importance of exercise quantification will be discussed and the concept of Range of Motion (ROM) will be defined.

2.1 Human Pose Estimation

Human Pose Estimation (HPE) is a computer vision task that involves locating and identifying key points on a person's body, thereby defining their pose in a given space. This area of research has gained significant attention due to its broad applications, ranging from human-computer interaction and gaming to sports analytics and healthcare. Accurate pose estimation is crucial in understanding human behavior, enabling machines to interpret and respond to human actions.

There are various approaches to tackle the challenge of human pose estimation, broadly categorized into two main groups - 2D pose estimation and 3D pose estimation. In 2D pose estimation, the focus is on body landmark detection, where Convolutional Neural Networks (CNNs) have shown remarkable success in localizing landmarks accurately. For example, Cao et al. [6] have shown a method of detecting the 2D pose of multiple persons with impressive accuracy. Heatmap regression is another technique in 2D pose estimation, with deep neural networks, particularly Hourglass Networks, regressing heatmaps to predict keypoints [7].

In 3D pose estimation, monocular methods estimate the 3D pose from a single 2D image, leveraging deep learning architectures to predict depth information and reconstruct the 3D pose. Multi-view methods, on the other hand, utilize multiple camera views, employing triangulation and structure from motion (SfM) to synthesize a three-dimensional representation of the human pose.

Human pose estimation algorithms generally follow a two-step process: training and inference. During training, a model learns the relationship between input im-

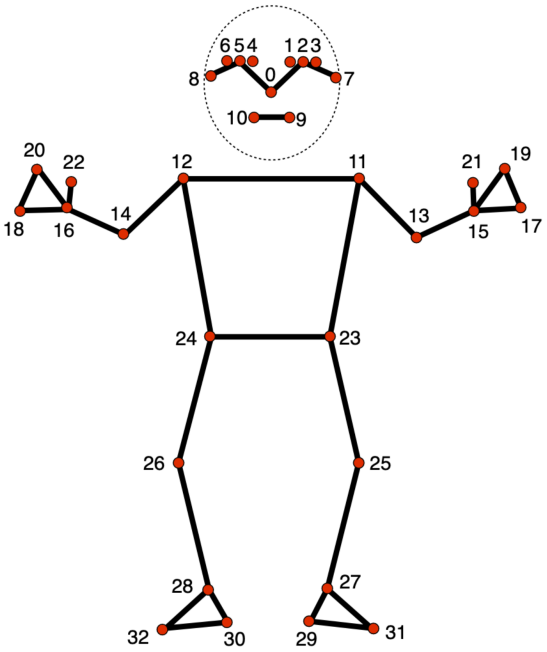


Figure 2.1: Media-Pipe's 33 body landmark locations.

ages and corresponding annotated pose data. The inference stage involves applying the trained model to new images to predict body landmark locations. Deep learning architectures, particularly CNNs and recurrent neural networks (RNNs), play a pivotal role in learning complex spatial dependencies and temporal sequences, enhancing the accuracy of pose estimation.

Mediapipe Pose Detection, a prominent library in computer vision, employs Convolutional Neural Networks (CNNs) and heatmap regression for accurate 2D keypoint detection. It excels in real-time pose estimation, aligning with established methodologies [6]. Furthermore, its capabilities extend to 3D pose estimation, making it a versatile choice for sports applications. The library's adherence to deep learning architectures ensures robust spatial and temporal sequence understanding, enhancing pose estimation accuracy. As shown in Fig. 2.1, Mediapipe uses a skeleton-based approach to model the body. This renders it especially useful for fitness applications, where body joints can be used to detect and describe movements. In this study, we will use this library for the extraction of body landmarks. Yet, for the classification of exercises and ROM, we will rely on a custom algorithmic solution.

2.2 Tracking of Physical Activity

The benefits associated with exercise tracking for improved fitness outcomes have been the focus of a number of research articles. For example, Burke et al. [8] delved into sports psychology, proposing that self-monitoring plays a crucial role in enhancing exercise adherence and performance. The act of logging and tracking exercises contributes to a heightened sense of accountability and motivation among individuals engaging in fitness activities.

Locke and Latham's goal-setting theory [9] further accentuates the positive impact of setting specific, challenging goals on performance. Keeping a record of exercises allows individuals to establish clear goals and monitor their progress, potentially bolstering adherence to workout routines and, consequently, influencing results. In a study by Jakicic et al. [10], the incorporation of technology for self-monitoring and feedback was associated with significantly increased weight loss among participants. While not specifically focused on gym exercises, these findings underscore the potential benefits of incorporating monitoring and feedback mechanisms in a fitness context.

In addition, the prevalence of fitness apps and wearable technology in the contemporary exercise landscape further suggests a perceived value in monitoring workouts. For the scope of this study, we will focus on the automatic recognition of performed exercises and tracking of a subject's ROM during execution. Based on this data, executed sets and repetitions can be measured. Furthermore, tracking the ROM enables us to monitor a subject's movement speed, which is also a crucial parameter in fitness.

2.3 Range of Motion (ROM)

Range of motion (ROM) is a fundamental parameter in fitness that refers to the extent to which a joint or series of joints can move in various directions. It is commonly measured in degrees and is crucial for evaluating flexibility and mobility. The importance of ROM in physical activity cannot be overstated. A healthy ROM is essential for performing everyday tasks with ease, as well as for excelling in various sports and exercise routines. The ability to move joints through their full ROM contributes to overall musculoskeletal health and reduces the risk of injury. Understanding and optimizing one's ROM can enhance the quality of life and athletic performance.

The concept of ROM has its roots in medical and anatomical literature. For instance, the importance of ROM in fitness is underscored by research in sports

science. In their study, Suchomel et al. [11] discuss how achieving optimal ROM in specific exercises can lead to improved strength and power development. This highlights the practical implications of ROM quantification in the context of strength training and sports performance.

Chapter 3

Novel Method

In this chapter, we will present the technological decisions, implementation process and overall structure of the system. In a later step, we will introduce the core classification method and its configuration parameters. During this process, the need for a standardised testing framework will be motivated.

3.1 Technologies

For the prototype, we chose to use a mobile platform. The choice was driven by the need for accessibility and convenience. Mobile platforms offer users the flexibility to utilize the app without significant infrastructural constraints, which is a pivotal aspect of fitness guidance. In this sense, the ubiquity of smartphones ensures that users can integrate their fitness routines into their daily lives, fostering engagement and progress.

In terms of the technology stack, Java emerged as a robust choice for the development of the application. Java's reputation for speed and efficiency in mobile application development is well-established. In view of the system's requirements, where various processes, such as model inference, deterministic classification, GUI updates and data persistence, need to operate concurrently, Java's inherent support for multithreading has proven invaluable.

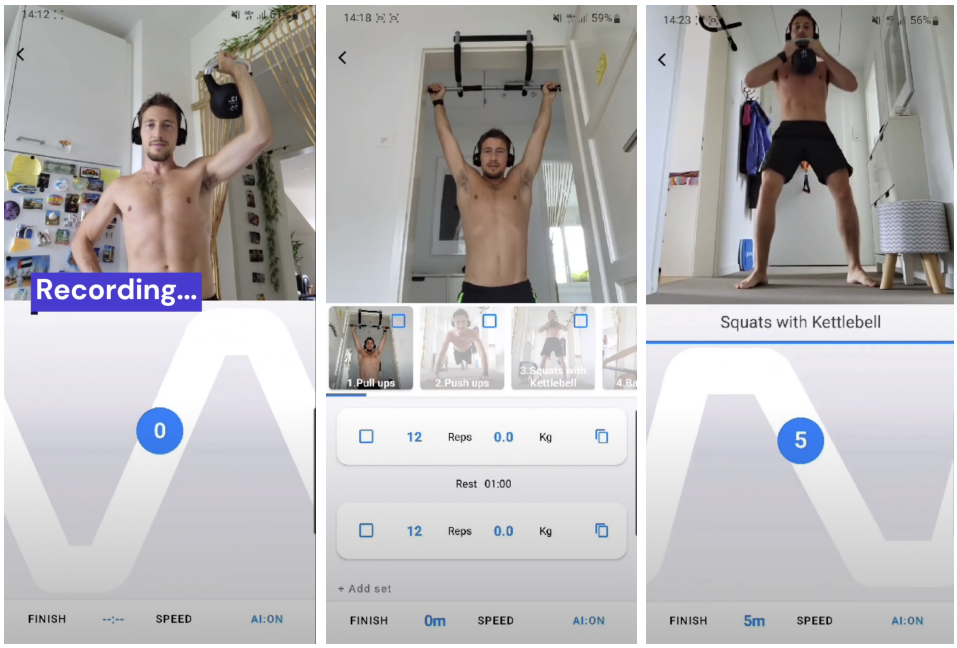
As mentioned previously, for the inference of body landmarks from image frames, Google's Media-Pipe pose detection solution was used. Besides from being an open source project, its thorough documentation, planned maintenance and constant development are some of the key benefits that motivated this decision.

3.2 Application Overview

The application's main use cases are the recording of new exercises, the recognition of previously recorded exercises and the tracking of ROM's. In the first scenario, the user is prompted to execute the new movement in a guided and systematic way. As shown in Fig. 3.1 (a), the subject synchronises their ROM with a GUI element that moves up and down with a configurable velocity. This approach enables the system to model an exercise as a set of consecutive poses, where each pose describes a specific phase of the movement.

One repetition of the new movement is sufficient for the system to create the internal model of the exercise that it can later recognize. We define this process as the system "learning an exercise". In the second case, shown in Fig. 3.1 (b), when a subject performs a previously recorded exercise, the system automatically recognizes its starting pose. Additionally, a visualisation of the system's belief (`fixationValue`) that a given exercise is actually executed is shown with the use of a loading bar. This feature enables the user to visually perceive whether a given exercise is being recognized.

In the third use case, shown in Fig. 3.1 (c), the system has locked the currently executed exercise, and synchronises a GUI element with the user's ROM. Based on that, the trainee's speed of execution can be measured and guided accordingly. Furthermore, repetitions are automatically counted, which forms the basis of quantifying a subject's complete training regime.



(a) Recording Phase (b) Recognition Phase (c) Tracking Phase

Figure 3.1: Visualisation of the system's three main use cases.

3.3 System Architecture

On a conceptual level, the application consists of five main modules - the AppManager, the ImageProvider, the LandmarkDetector, the ExerciseRecorder and the ExerciseClassifier[1]. The AppManager acts as a finite state machine that directs data flow and renders the GUI depending on the application's current use case. The ImageProvider offers a camera preview and provides the LandmarkDetector with a continuous stream of image frames.

The LandmarkDetector uses Google's Media-Pipe Pose-Detection ML solution internally, to infer data describing the current pose. After inference, the data flow can have two distinct pathways, depending on the current use case. For the recording use case, the data is passed to the ExerciseRecorder, where it is processed and persisted. The ExerciseRecorder is also responsible for rendering the graphical user interface that guides the user through the recording process.

For the recognition and tracking use cases, the data is directly passed to the ExerciseClassifier, which identifies the exercise that the current pose belongs to. It also returns the ROM in which the subject's pose is estimated to be in. The results are then passed to the AppManager where the GUI is updated.

3.4 Exercise Classification

The system's ability to recognize exercises is determined by three main factors - the model's inference quality, the recording method and the classification method. Being state of the art, we view the chosen pose detection solution as a system-context component. In this sense, improvements in this sector are outsourced to Google's Media-Pipe research.

On the other side, both recording and classification methods are mutable system processes. After significant development efforts, we make the assumption that the ExerciseRecorder stores the inferred poses in a reliable and accurate way. As a logical conclusion, the ExerciseClassifier is estimated to have the highest potential to determine the system's accuracy. Thus, for the scope of this study, we will only focus on the inner workings of the ExerciseClassifier.

3.4.1 Data Description

In order for the ExerciseClassifier to work, the current pose object and a set of recorded exercises are required. A pose object is a list of 33 normalized 3D coordi-

[1]Full code available at: `https://github.com/aris-konstantinidis/fugazi/blob/main/src/main/java/org/example/Classifier.java`

nates, each describing the position of a body landmark location. These pose objects are inferred by Mediapipe's pose detection model, based on the given image frames. Each list item also contains information about the presence and the visibility of each landmark. While presence implies the probability that a given landmark is within the image frame, visibility also takes into account the possibility of occlusions. Occlusions occur when from the viewer's perspective, one landmark is hidden by another. Figure 2.1 illustrates all available body landmark locations provided by the Media-Pipe framework.

The set of recorded exercises contains lists of pose objects, each belonging to a specific exercise. Here, each pose object also contains a ROM value, indicating its position within the movement. For example, the exercise "Squats", could contain a list of 5 pose objects, where the first pose object was captured at the start of the exercise's ROM and the last pose was captured at the end of the exercise's ROM. Consequently, the pose objects in the middle occupy 20%, 40%, 60% and 80% of the exercise's full ROM.

3.4.2 The ExerciseClassifier

In its simplest description, the ExerciseClassifier is a Java class. From a more complex point of view, it is a stateful and event-driver module, responsible for the accurate and reliable recognition of the currently executed pose. The most minimal way that such a system could work, would be to compare the current pose input with all the poses that have previously been recorded. Formally, given a pose input p and a cached array of poses P, one could define the optimal result to be the pose in the list of poses that is the most similar to the input pose. Given that the data, in its most atomic level is expressed in 3D coordinates, the Euclidean Metric appears to be a reasonable measure of fitness. Thus, in our case, the distance between two poses p_1 and p_2 would be calculated by the following formula:

$$d(p_1, p_2) = \sqrt{(p_{1_x} - p_{2_x})^2 + (p_{1_y} - p_{2_y})^2 + (p_{1_z} - p_{2_z})^2} \tag{3.1}$$

For computational efficiency, squaring will be ignored, as the resulting distances can still be compared and only the relation between them is important. Although in theory, this computation would always return the closest match, the are multiple problems that arise. The following section discusses these issues in detail, and motivates the need for a more sophisticated classification scheme.

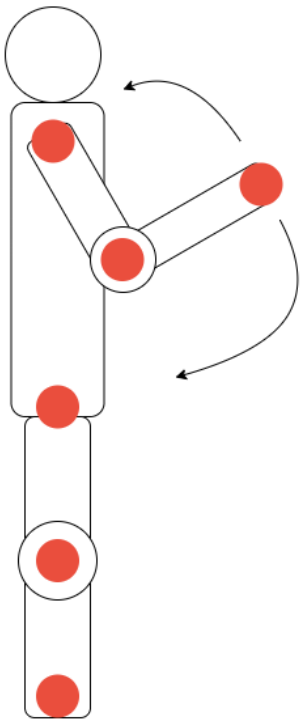


Figure 3.2: An ambiguous pose that can be found in many common resistance training exercises.

Considerations and Main Issues

First of all, the inferred input stream is noisy. For example, if an occlusion happens, the model cannot robustly infer where the hidden body landmarks are positioned. Combined with the inference model's probabilistic nature, this leads to high jitter of the input signal.

Furthermore, collisions between poses are not atypical for exercises. Many movements during physical activities contain poses that are almost identical. For example, the pose shown in Fig. 3.2 can be found in cable triceps extensions, biceps barbell curls, shoulder side raises, dumbbell lateral raises, and many other exercises. A system that solely relies on a distance metric would inevitably lead to an unstable classification experience.

In addition, variation in the execution of a specific exercise should not be underestimated. Muscular and neurological fatigue occurs at almost every high-intensity training, a fact that guarantees within-subject variation of execution between repetitions. Between-subject variation in movement execution is also prevalent, given the differences in fitness, psychological state, and expertise.

The previous problems lead to the realization that the classification algorithm should allow some degree of deviation from the optimal movement execution, yet it should still be able to differentiate between noise and an actual pose pattern. Inevitably, the statistical problem described as Type-1 and Type-2 trade-off emerges.

A liberal classification scheme would result in a disproportionate number of false positives, a highly conservative approach would lead to an excessive number of false negatives. In the next section, we will discuss ways in which these problems can be controlled.

Exploration of Sophistication Methods

A plausible way to enrich the classification system with some degree of deterministic intelligence, is by mimicking a human approach to classification. Instead of focusing on finding a better suited metric, previous experience can be used as a bias for the next predictions. Analogous to human behavior, the system could maintain a notion of classification certainty and be conditioned by past events. In digital systems, memory can be expressed through persistence of data. By storing a sequence of observed data points, a system can extract a pattern in time, and bias itself towards an expected future outcome. The following example introduces the main questions that emerged during development.

> The system has learned to recognize two exercises E_1 and E_2. Although the two movements differ in their overall displacement of the body's landmarks, both contain a pose that is almost identical. These poses are defined as p_{E_1} and p_{E_2}. At some time, the ExerciseClassifer has observed 10 consecutive poses that belong to exercise E_1. Then, based on the euclidean distance between the newly observed pose p_n and all the other recorded poses, the closest found distances are $d(p_n, p_{E_1})$ and $d(p_n, p_{E_2})$. It is computed that $d(p_n, p_{E_1}) > d(p_n, p_{E_2})$. Should the system immediately accept that the subject switched to exercise E_2? Should it recognize this computational result as an outlier and still predict E_1? After how many such outliers should the system change its belief that exercise E_1 is performed?

The previous example motivates the necessity of four constructs that will be programmed as configuration parameters into the classification algorithm, namely *memory*, *certainty*, *deviation* and *fixation*. Memory describes the persistence of past events. Certainty expresses the belief that the system has, that its current prediction is correct. Deviation defines an artificial threshold that the system will use to differentiate between variations that fall within a typical range, and variations

that fall outside of this boundary. This is needed in order for the system to be able to differentiate between noise and actual pose patterns. Fixation describes the act of ignoring a purely computational result, in favour of a heuristic belief, that another outcome should be the correct one.

3.5 Classification Method

In this section, the configuration parameters, the main algorithm and the need for a rigid testing framework will be introduced.

3.5.1 Configuration Parameters

Based on the previously mentioned constructs of memory, certainty, deviation and fixation, during development, 6 configuration parameters emerged. In the following, the motivation for the existence of each parameter and their importance for the system's performance will be presented.

1. exerciseHistorySize

The exercise history buffer is essentially a list that holds the last n exercise predictions. This parameter facilitates the integration of a historical bias into the exercise recognition process. For example, when the most recent k predictions consistently indicate the same exercise, it may suggest a higher likelihood that the current pose corresponds to the same exercise. If the buffer size is excessively large, it can exert an overwhelming influence on predictions based on past data. Conversely, if the buffer size is too small, predictions may primarily be influenced by the present input, which, given the noisy nature of our data, could potentially result in jitter and an overall unstable classification experience.

2. exerciseSwitchThreshold

The exercise switch threshold complements the *exerciseHistorySize* by specifying the number of occurrences of the same exercise within the history buffer, required to trigger exercise recognition. Given that the model infers 33 3D landmarks from a single image frame, the model will always generate some number of erroneous outputs. Therefore, it is crucial to select an appropriate switch threshold, that allows for some error, but can still help the system differentiate between noise and a pattern over time. A small value is prone to a series of outliers that can trigger a false positive classification, while a large value may act conservatively, making exercise recognition slow.

3. maxFixationValue

The `maxFixationValue` expresses the classifier's confidence in the execution of an exercise. For instance, during the initial prediction of an exercise like "Squats," the belief that this exercise is actually executed should be relatively weak, as the observed data could potentially only include noise. However, with consistent "Squats" predictions, the classifier should strengthen its belief , as the probability that noise consequently triggers the recognition of a specific exercise is estimated to be low. This parameter adds an additional layer of robustness against outliers, ensuring relative stability in predictions. Its second responsibility is to allow short deviations from the optimal movement execution, without directly aborting the exercise. For example, during exercise execution, a trainee might take in a muscle relaxing pose that deviates significantly from the target poses. In this case, the `fixationValue` will act as a counter that will prevent the exercise being exited in the middle of the set. Determining the ideal value for this parameter is challenging, as a high fixation value could lead to a erroneous fixation on a false positive result, while a small value may lead to the loss of the desired bias, and an unstable classification scheme.

4. sdLow

During comparisons between the currently observed pose and all recorded poses, the ExerciseClassifier will always identify a "best fitting" pose. Thus, the possibility that no exercise is performed is not included in the set of possible outcomes. As a consequence, it is crucial to establish an absolute deviation threshold, when it comes to recognizing that no exercise is being performed. This necessitates the determination of a reference value. Through analysis described in a later section, such a value was found as a starting point. It's worth noting that this value is influenced by the size of the dataset used for the analysis and could have been optimized empirically. A high value can lead to a multitude of false positives, while a small value may not allow sufficient variation in exercise execution.

5. sdHigh

The `sdLow` parameter comes into play when the ExerciseClassifier has not yet locked a specific exercise. In this scenario, the current pose is compared with all recorded exercise poses, with a relatively small allowed deviation as specified by `sdLow`, to avoid false positives. However, once the ExerciseClassifier has locked an exercise, the comparisons are intentionally limited to the currently fixated exercise poses, allowing for a greater deviation as specified by `sdHigh` to accommodate a greater

variation in exercise execution. This is used as a fixation method, to make the overall classification experience more stable. It's important to note that this parameter is currently also an estimate, based on `sdLow`.

6. landmarkPresenceThreshold

The model used for inferring landmarks assigns a presence value to each joint, indicating the probability that a joint is visible and not occluded in the image frame. The `landmarkPresenceThreshold` is used by the classifier to determine whether a comparison is rational. For example, if the knees have a low presence value, it would be problematic to compare them with a stored pose where the knees were clearly visible. A high presence threshold acts conservatively, allowing exercise recognition only when joints are clearly visible, which may not always hold true in real life scenarios. On the other hand, a low value may result in the use of joints from the current pose whose 3D coordinates are likely not very accurate.

3.5.2 Main Classification Algorithm

The ExerciseClassifer's main function is to predict which exercise is performed and at which ROM. Exercises are defined by a unique integer (`eid`) while ROM (`rom`) is expressed as a percentage value. These two values make up the ClassificationResult, that is computed after each classification iteration. Internally, the ExerciseClassifer uses two lists, namely the `exerciseHistory` and the `fixationValues`. The `exerciseHistory` is a limited size queue, where the newly predicted `eid` is stored, and the oldest entry is removed. The `fixationValues`'s length equals $N+1$, where N is determined by the number of exercises that have been recorded. The additional entry is reserved for the fictional *nullExercise*, which expresses that no exercise is recognized.

During initialization, while each exercise's `fixationValue` is set to 0, the nullExercise's value is set to the maximal allowed fixation value, which is one of the 6 configuration parameters. Each time that an exercise is believed to be executed, if another exercise's fixation value is greater than 0, it is decremented. If the list only contains 0's, the predicted exercise's fixation value is incremented. This technique has proven to smoothen switching between exercises and increases classification stability. The algorithm is made up of two main conditional branches. The first branch, called *interExerciseCase*, is chosen when based on its previous measurements, the system believes that the trainee is not executing an exercise yet. The second branch, called *intraExerciseCase* is chosen when based on its previous measurements, the system believes that the subject still performs a particular exercise.

Purposefully, the previous outcome biases the handling of the next data point.

In the *interExerciseCase*, the current input pose is compared with all recorded exercises' starting poses, by measuring their distance. As mentioned previously, this comparison will always return an exercise, which excludes the possibility that no exercise is performed. To solve this issue, the notion of deviation is needed. The system should be able to compute, whether a given pose significantly deviates from the target pose. The subsection "Exploring an Absolute Deviation Threshold" describes the research that was done to explore possible deviation boundaries.

Continuing with the *interExerciseCase*, the system outputs the best matching pose, given that the found pose lies within a given deviation threshold. If an exercise is found, the fixation value of this particular exercise is incremented, and the `eid` of the exercise is inserted into the exercise history. If no exercise is found, the fixation value of the last exercise that was believed to be most probable is decremented, and `null` is inserted into the exercise history. If the fixation value of an exercise exceeds a certain threshold, defined by the `maxFixationValue`, the ExerciseClassifier accepts it as the currently executed movement.

This leads to the *intraExerciseCase*, where the system is challenged to maintain its belief that a particular exercise is still executed. In the *intraExerciseCase*, the Classifier compares the currently observed pose only with the poses of the exercise that is believed to be executed. If the found pose lies within an acceptable threshold, the fixation value of this exercise is incremented, and its *eid* is inserted into the exercise history. In the opposite case, its fixation value is again decreased and `null` is inserted into the history buffer. If the ratio between the exercise's *eid* occurrences to the exercise history list size lies below some defined value - called the *exerciseSwitchThreshold*, and the fixation value of the exercise is 0, the belief that this exercise is still performed is aborted, and the *interExerciseCase* gets active. To sum up, Fig. 3.3 presents the classification algorithm in a visual way.

Exploring an Absolute Deviation Threshold

As mentioned previously, the system requires a threshold that separates typical and atypical movement deviations from a target movement. In order to find such a reference value, we took advantage of the fact, that no two exercise repetitions are exactly the same. When a trainee executes an exercise, each repetition can be viewed as a time-series. In Fig. 3.4, the change in the angle of the left elbow over time, for 3 repetitions of biceps dumbbell curls is illustrated. As shown, although all 3 curves show the same overall pattern over time, there are evident differences between them. To improve visualization and help understanding, the following example will use angles that were inferred from the actual 3D positions of body

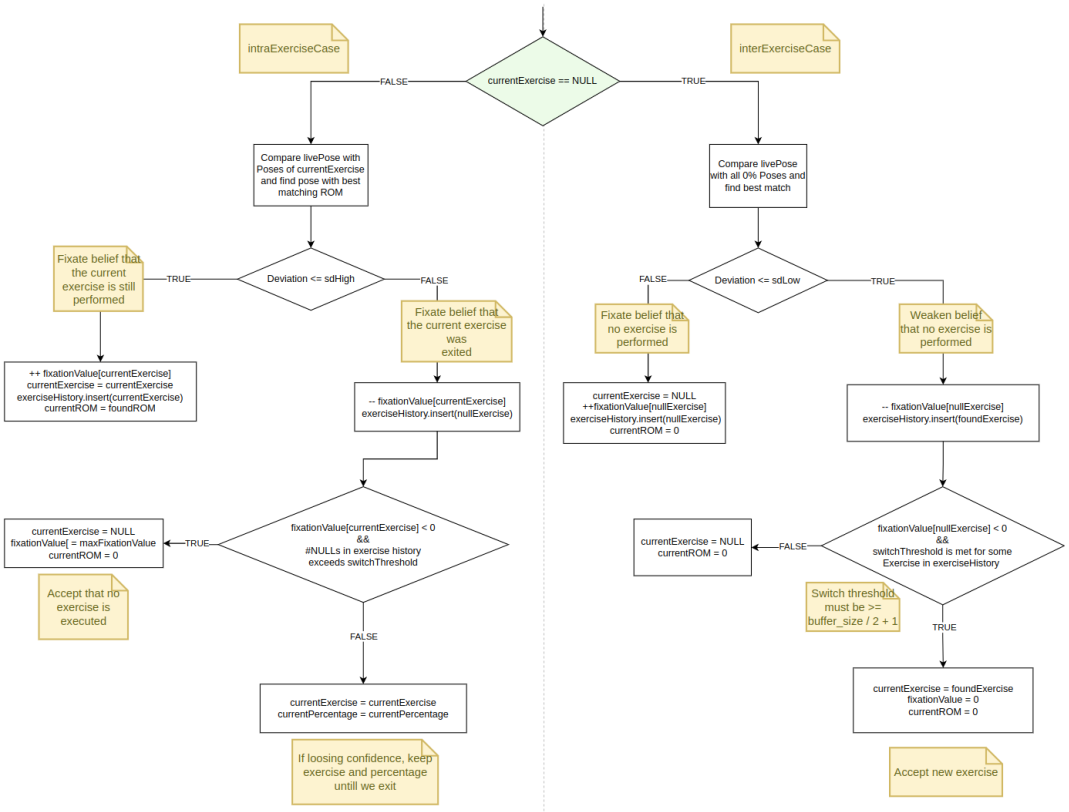


Figure 3.3: Flowchart representing the classification algorithm.

landmarks. It is noted, that the actual implementation works directly with the 3D coordinates, as the abstraction of angles is unnecessary and increases latency.

To derive an approximate measure of deviation, the following statistical procedure was used. We recorded several repetitions of the same exercise and aggregated all repetitions for each joint angle as a time-series. For a fixed point in time (ROM), based on all execution variants, we calculated the mean angle of the particular joint. Then, we calculated the standard deviation of the angle for this point in time. The resulting vector of standard deviations was then used to derive a distribution. As seen in Fig. 3.4, the distribution appears to follow the Gaussian distribution. In other words, most of the time, the standard deviation - or variance in execution of an exercise, lies within a particular boundary, and only rare are there significant outliers in variation.

For example, for bicep curls, it was calculated that for the most time, the variation of the elbows' angles between different repetitions typically lies between 14 and 19 degrees. Having computed an estimate for how much variation in movement execution can be expected, allows us to be able to better differentiate between typical and atypical variations, and ultimately be able to recognize whether a pose does not match any of the recorded poses.

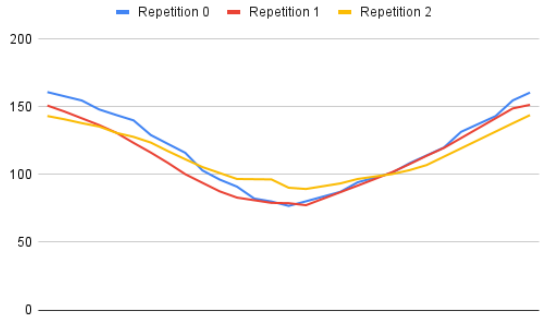


Figure 3.4: Change in the angle of the left elbow over time, for 3 repetitions of biceps curls.

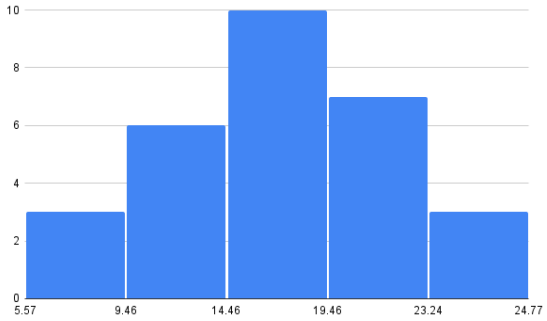


Figure 3.5: The distribution of standard deviations.

3.6 Parameter Tuning

During development, numerous combinations of the 6 presented configuration parameters were tested. Quickly, the interdependence between them became evident.

For example, we could increase both `sdLow` and `switchThreshold`, which would allow greater movement variance but require more occurrences of a specific exercise in the exercise history to result in this exercise's recognition. On the other side, we could decrease `sdLow`, narrowing the allowed movement deviation, and decrease the `switchThreshold`. Even if less exercise occurrences are required in the history buffer, less poses would fall within the defined standard deviation. As one can see, both configurations could theoretically result in similar classification strategies.

Although each configuration parameter seems to play a vital role in the classification scheme, the assumed dependencies between them do not allow each parameter to be optimized separately. Thus, in order to find the parameter combination that will output the highest classification accuracy, we will use a data-driven approach. In the following chapter, we will discuss the design of a standardized experiment with which classification accuracy will be analysed.

Chapter 4

Experimental Evaluation

In this chapter, the procedure followed to quantify the system's classification accuracy is presented.

4.1 Experimental Architecture

The experiment consists of two separate phases - the data collection phase and the accuracy testing phase. While the first phase is conducted using the existing system infrastructure, the second phase is delegated into a standalone testing project[1].

4.1.1 Data Collection Phase

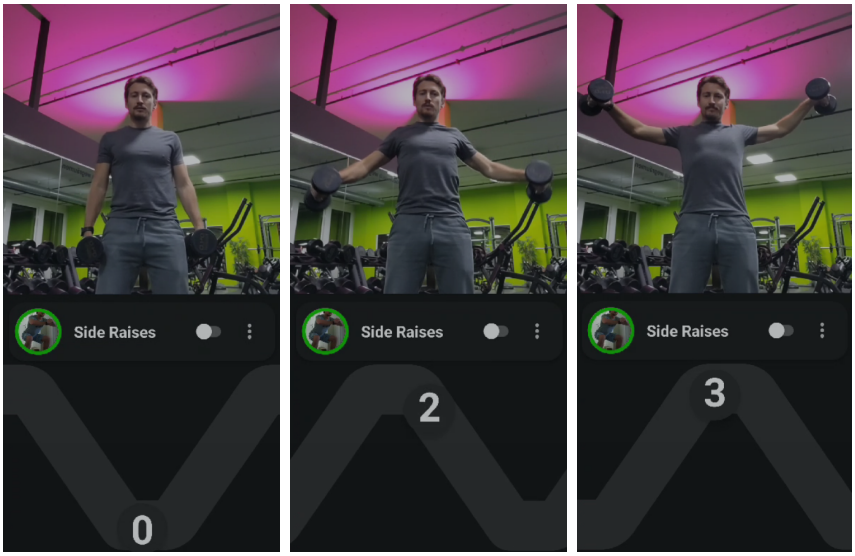
As a first step, we will use the application's recording use-case to generate a SQLite database file that contains the exercises that the ExerciseClassifier can recognize. This process is visualized in Fig. 4.2. The ExerciseClassifier can then load this data in a way that allows efficient comparison with a new pose input. We define this dataset as the *reference dataset*.

In a second step, we will record videos in the form of a list of consecutive image frames, pass these image frames to Mediapipe's pose detection model for inference and store the inferred pose data into a CSV file. Each entry will also contain the image frame's captured timestamp. In parallel, we will store each image frame and again use the captured timestamp as the file's name. After all poses have been stored in the CSV file, we will proceed with the labeling process. To do so, based on each inferred pose's timestamp, we will visually analyse the corresponding image frame and manually add an exercise and ROM value.

For example, the pose in Fig. 4.1 (a) would be assigned the executed exercise's `eid` for the exercise label and 0 for the `rom`, as the subject has not yet initialized the

[1]The repository can be found at: `https://github.com/aris-konstantinidis/fugazi`

movement. Analogous, the pose in Fig. 4.1 (b) would be assigned the same `eid` for the exercise label and 0.5 for the `rom`. The pose in Fig. 4.1 (c) would also be assigned the same `eid` for the labeled exercise and 1 for the `rom` value, as the subject's pose is at the maximum ROM for this exercise. Although the objectivity and validity of this labeling process can not be proven scientifically without significant additional effort, we tried to label each pose as accurate as possible. This data is defined as the *test dataset*.



(a) 0% ROM. (b) 50% ROM. (c) 100% ROM.

Figure 4.1: A subject performs Standing Dumbbell Shoulder Side Raises (SDSSR) at different ROM's.

Each entry can then be loaded and passed to the ExerciseClassifier for classification. The recorded videos will depict trainees executing a random permutation of the exercises that the system is able to recognize.

4.1.2 Accuracy Testing Phase

To evaluate the system's accuracy, we will create a separate testing project. As shown in Fig. 4.3, the testing tool instantiates an ExerciseClassifier and provides it with several parameters. Firstly, the SQLite file containing the reference data is loaded. This way, the ExerciseClassifier accesses the reference data, with which comparisons can be made. Secondly, the CSV file containing the manually labeled poses is passed through the ExerciseClassifier's `classify` method. This data acts as the testing data, based on which the classification accuracy will be evaluated. Thirdly, the ExerciseClassifier is given a set of parameter combinations. For each parameter configuration, a ClassificationResult is returned. ClassificationResults contain the predicted `eid` and `rom`. Finally, each prediction's accuracy score is

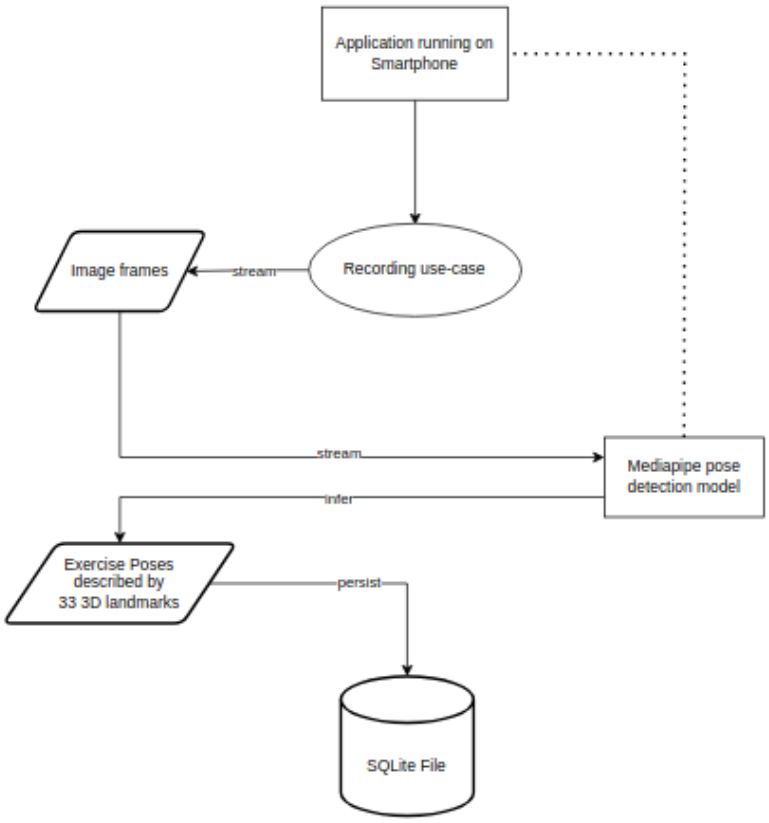


Figure 4.2: The experiment's reference data collection process.

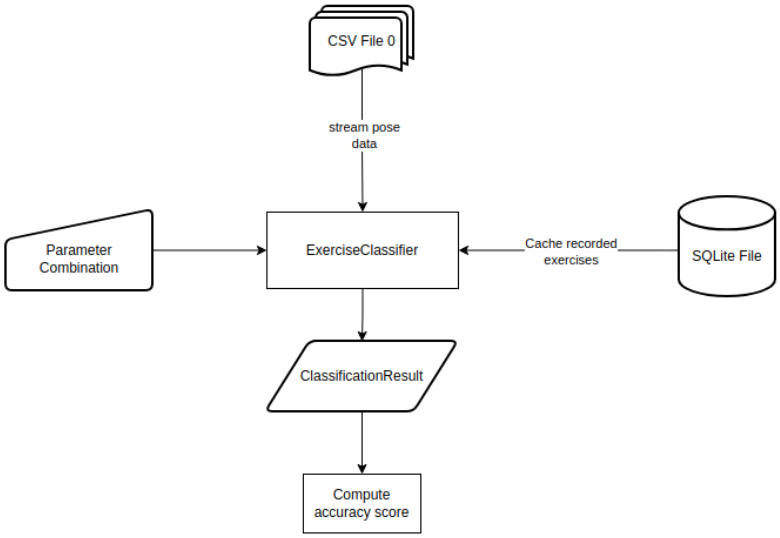


Figure 4.3: The experiment's testing process.

computed. The module[2] responsible for computing classification accuracy measures whether the predicted exercise matches the actual observed exercise, and whether the predicted ROM falls within a specific boundary around the actual `rom`.

Finding whether the correct exercise was predicted is an elementary task, as the predicted `eid` can be directly compared to the labeled `eid`. For the ROM, being continuous in nature, a margin of error of 0.15 around the correct percentage was allowed. Thus, a negative or positive difference between prediction and actual percentage that was lower or equal to 0.15 was deemed as a correct prediction.

4.2 Parameter Ranges and Computational Considerations

Each parameter combination creates a unique classification strategy that directly influences the ClassificationResult. Thus, it becomes evident that the choice of the parameter combinations that are tested is of vital importance. Given the novel nature of the classification algorithm, we will approach the search for the optimal parameter configuration in a computational manner by using a grid search[3]. Yet, for a grid search to be computationally feasible, each parameter has to have a clearly defined start, end and step value. The step value will be added to the start

[2]Full code available at: `https://github.com/aris-konstantinidis/fugazi/blob/main/src/main/java/org/example/MetricScorer.java`

[3]Full code available at: `https://github.com/aris-konstantinidis/fugazi/blob/main/src/main/java/org/example/GridSearch.java`

value after each iteration, until the end value is reached. The discretisation of the value space that a parameter can take is a demanding task, as too many values might reach computational limits, whereas too view values do not guarantee that a representative sample of the possible parameter value space is explored. In the following subsection, the selection of ranges for the 6 parameters will be discussed.

ExerciseHistorySize

On an average modern smartphone, the system analyzes approximately 15 image frames per second. When a user takes in the starting pose of an exercise, they should get immediate feedback, to avoid making them doubt the correctness of the pose, or lose trust in the system's classification method. As mentioned by Miller [12] in his research article "Response time in man-computer conversational transactions", response times of more than 4 seconds for actions that await communicative response lead to a disruption of the communication thread. Ideally, feedback should occur approximately within 500 milliseconds to 2 seconds. From a technical point of view, if the system only considers the last image frame input, as discussed in previous chapters, classification would be jittery. Thus, it is assumed that the ideal `exerciseHistorySize` lies between 5 and 20. The lowest step increment of 1 is chosen.

ExerciseSwitchThreshold

This is the only parameter that is not manually configured by the researchers, but is automatically computed based on the `exerciseHistorySize`. The exerciseSwitchThreshold's lower and upper bounds are restricted by the exerciseHistorySize. Specifically, given an exercise history buffer of length N, it is assumed that the optimal value lies within $[floor(N/2)+1, N)$. Setting the `exerciseSwitchThreshold` to a lower value has been empirically shown to destabilize classification, as the system becomes prone to outliers. The lowest step increment of 1 is chosen.

MaxFixationValue

The fixation value acts as an output stabilizer that allows for errors. Again, assuming that an average smartphone produces 5 image frames per second, a very high value like 50, would result in a very slow rejection of a previously performed exercise. On the other side, a very low number like 1, would mean that an exercise will be rejected if a trainee deviates from the movement threshold for more than half a second. Thus, it is assumed that the optimal value should lie between the values 3 and 20. The lowest step increment of 1 is chosen.

SDLow

The method for finding a plausible standard deviation reference value was discussed above. The results indicated that during different repetitions of the same exercises, body landmarks show an average standard deviation value of 0.01. In addition, empirically, we observed that values above 0.02 result in a significant increase of false positive exercise classifications. On the other hand, values below 0.01 allow little to no movement deviation. It is thus assumed, that the optimal value should lie between 0.01 and 0.02. Considering the computational restrictions of a grid search, we chose a step value of 0.0025.

SDHigh

To avoid limiting the search space by additional assumptions, the same values as `sdLow` are used.

LandmarkPresenceThreshold

By means of trial and error, it was found out, that the `landmarkPresenceThreshold` remains very high (above 0.92), unless a body landmark completely disappears from the image frame. Thus, we will assume a minimum value of 0.92 and a maximal value of 97. A step value of 0.01 is chosen.

Using the boundaries presented above and given that each iteration completes in approximately 14.8 nanoseconds, we evaluated the grid search's feasibility and estimated the running time needed to explore the defined value space. Alg. 1 illustrates the used method. All computations were executed on a Lenovo ThinkPad X1 Carbon Gen 10, which features a 12th Gen Intel Core i7 processor, a 32GB LPDDR5 RAM at 5200 MHz, and 1TB PCIe SSD of storage.

We calculated that the grid search will test 237600 unique parameter combinations with a running time of $N \cdot 237600$ milliseconds, where N is the number of image frames in the reference dataset. Thus, for a 5 minute video with 15 FPS, the experiment would need 392040000 computations and a running time of approximately 1 hour and 37 minutes to complete.

4.3 Results

In this chapter, the collected data will be presented and the experiment results will be analysed.

Algorithm 1 calcGridSearchStats

Require: imageFrames
Ensure: {paramCombs, iters, nanos}
  paramCombs ← 0
  iters ← 0
  nanos ← 0
  for $a \leftarrow 5$ to 20 by 1 **do**
    for $b \leftarrow \text{floor}(a/2) + 1$ to a by 1 **do**
      for $c \leftarrow 3$ to 20 by 1 **do**
        for $d \leftarrow 0.01$ to 0.02 by 0.0025 **do**
          for $e \leftarrow 0.01$ to 0.02 by 0.0025 **do**
            for $f \leftarrow 0.92$ to 0.97 by 0.01 **do**
              paramCombs ← paramCombs + 1
              for $g \leftarrow 0$ to length(imageFrames) − 1 **do**
                iters ← iters + 1
                nanos ← nanos + 14.8
              end for
            end for
          end for
        end for
      end for
    end for
  end for
  return {paramCombs, iters, nanos}

Exercise	Poses	Reps
Squats	251	6
Back Rows	274	8
Shoulder Presses	186	3
Shoulder Side Raises	250	7
Overhead Triceps Extensions	277	7
Right Torso Rotations	291	7
Right Shoulder External Rotations	331	8
No exercise	1312	-

Table 4.1: Absolute number of poses and repetitions recorded during test data collection.

4.3.1 Collected Data

Due to the fact that the data labeling process is extremely time costly, we only used a home fitness testing scene. To still collect a representative amount of data, a relatively big set of exercises was recorded. The exercises were chosen based on their commonality in the field of fitness. Furthermore, we aimed to include compound exercises as well as isolated ones. The list includes Squats, Back Rows, Side-Shoulder-Raises, Shoulder-Presses, Overhead-Triceps-Extensions, Right-Torso-Rotations and Right-Shoulder-External-Rotations.

It is noted, that this kind of setting, containing exercises that are not performed on machines, is more difficult to be tracked, as the variation in execution, and the subject's positioning in the camera frame can vary more, compared to movement execution on machines. To collect testing data two subjects performed a random permutation of the recorded exercises. The first recorded video has a duration of 5 minutes and 27 seconds and produced 1657 poses. The second one has a duration of 4 minutes 59 seconds and produced 1515 poses.

1312 of the overall 3172 collected image frames[4] depict the subjects not executing any exercise, and will act as noise. The overall executed repetitions for each exercise are shown in Tab. 4.1. The sequence of the exercises was chosen randomly by the subjects during recording. Between sets, the subjects were instructed to show a normal relaxation behavior. Manual labeling of the test data[5] took approximately 8 hours to complete.

[4]A subset of the collected image frames can be found at: `https://github.com/aris-konstantinidis/fugazi/tree/main/data/3/frames`

[5]The CSV file that contains the complete labeled test data can be found at: `https://github.com/aris-konstantinidis/fugazi/blob/main/data/2/test-data.csv`

Parameter	Value
exerciseHistorySize	11
exerciseSwitchThreshold	8
maxFixationValue	4
sdLow	0.0149
sdHigh	0.0174
landmarkPresenceThreshold	0.96

Table 4.2: Optimal parameter values found via grid search.

4.3.2 Optimal Parameter Combination

In sum, 237600 different parameter combinations where evaluated on 3172 different poses. Each computation took about 14.8 nanoseconds. The overall grid search's running time was 3 hours 5 minutes and 11 seconds. Tab. 4.2 shows the parameter configuration that yielded the highest exercise and ROM classification accuracy.

In order to be able to visualize the explored parameter value space and its effect on ROM accuracy, we held `sdLow`, `sdHigh` and `landmarkPresenceThreshold` constant and equal to their optimal found values and repeated the grid search with only 3 degrees of freedom. The explored parameter value space and the resulting ROM accuracy scores are shown in Fig. 4.4. As expected, the grid forms a triangular shape which is determined by the lower and upper bounds that `exerciseHistorySize` sets for `exerciseSwitchThreshold`. We observe that the highest ROM accuracy is achieved when `exerciseHistorySize` is greater than 10, `exerciseSwitchThreshold` has a value of about 60% of the `exerciseHistorySize` and `maxFixationValue` is below 5. From the found data, we assume that the optimal classification strategy should take into account the image frames of the last 1 second, allow a 4/10 noise to data ratio and avoid high exercise fixation.

4.3.3 Accuracy Evaluation

As mentioned previously, accuracy is evaluated for exercise classification and ROM classification separately. By using the parameter values from Tab. 4.2, the system achieved an overall exercise classification accuracy of 90.7%[6]. Using a 15% margin of error allowance, the predicted ROM's matched the actual ROM's in 83.9%[7] of

[6]CSV file with predictions can be found at: `https://github.com/aris-konstantinidis/fugazi/blob/main/data/2/predictions/0.9079445.csv`

[7]CSV files with predictions can be found at: `https://github.com/aris-konstantinidis/fugazi/blob/main/data/2/predictions/0.8398487.csv`

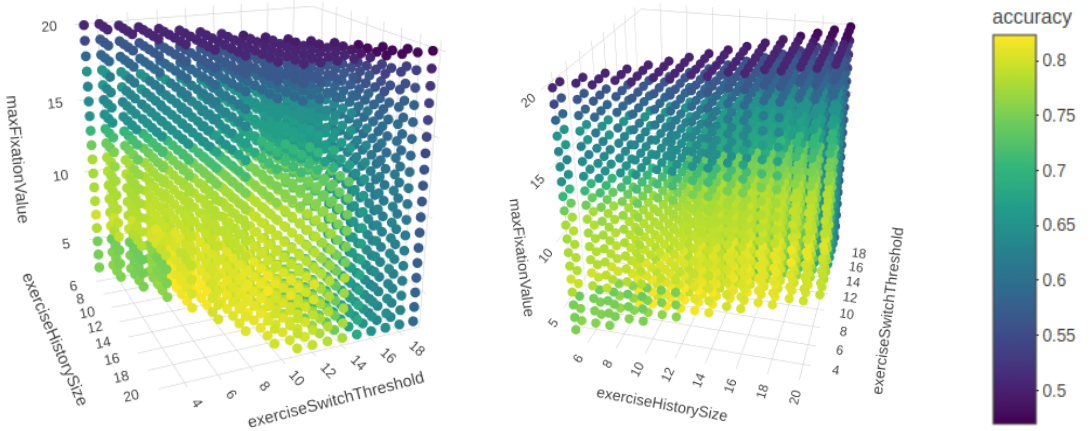


Figure 4.4: The parameter combination outcome space for variable exerciseHistory-Size, exerciseSwitchThreshold and maxFixationValue and fixed sdLow, sdHigh and landmarkPresenceThreshold.

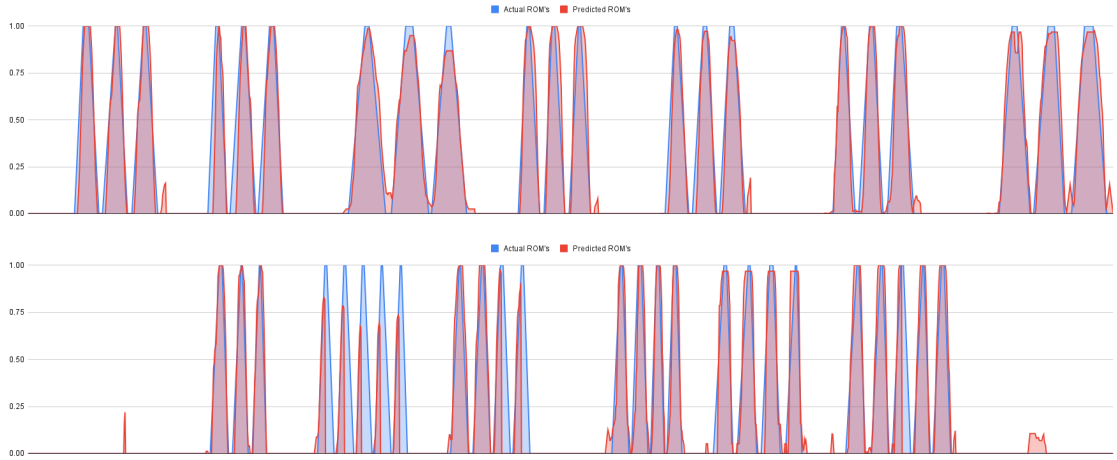


Figure 4.5: ROM time-series for the 2 recorded test videos. Labeled values are visualized in blue, predicted values are visualized in red.

the cases. Figure 4.3 illustrates the actual and the predicted values for the two test cases.

Chapter 5

Conclusions and Future Work

Based on the experimental results and further empirical testing, we could prove that the system can learn and recognize new exercises by only observing one repetition. In addition, we could show that tracking of ROM's in real time is possible. Both tasks are also performed with a relatively high precision. Yet, as shown in Fig. 4.5, accuracy can fluctuate significantly. This variation in accuracy can be attributed to many factors.

As discussed in previous chapters, the deviation magnitude between the learned movement and the executed movement plays a critical role. In fitness, most exercises can be performed using multiple variations. Even if two trainees execute the same exercise variation, their movement style will likely differ. Thus, classification accuracy is highest, when the same trainee records and later executes the same exercise variation.

In addition, movement speed seems to be negatively correlated with ROM classification quality. If a trainee executes a movement with high velocity, the inferred poses are more likely to be erroneous, which makes classification more challenging.

Furthermore, classification is not possible if the subject's angle towards the camera differs more than approximately ± 30 degrees between the learning phase and the recognition phase. Mediapipe's pose detection model, in its current state, delivers distorted poses depending on the body's overall angle towards the camera. Faulty poses are also generated when body landmarks are occluded.

Thus, to achieve the highest possible classification accuracy, the following conditions should be met

- The significant joints of an exercise should not be occluded during learning or recording phase.
- The angle of the subject's body towards the camera during learning phase should be approximately equal to the angle during recognition.

- The same exercise variation for learning and recording should be used.
- The subject's moving speed should not be lower than 1 second for a complete ROM.

In order to further optimize the classification method, the step values for `sdLow`, `sdHigh` and `landmarkPresenceThreshold` could be made smaller. In addition, the usage of more testing scenes would ensure that the parameters are not specialized for a specific setup and exercise execution variant. For example, if the test dataset contained poses that perfectly match the reference data, `sdLow` and `sdHigh` could be small. On the other hand, if reference and testing data showed great discrepancies, `sdLow` and `sdHigh` would need to be higher, in order for any of the observed poses to fall within the allowed threshold.

Appendix A

Code, Data and Results

Code and data used in this study is referenced in footnotes. The complete testing project can be found at `https://github.com/aris-konstantinidis/fugazi`.

Bibliography

[1] Shiyuan Yu, Zhifeng Chen, and Xiang Wu. The impact of wearable devices on physical activity for chronic disease patients: Findings from the 2019 health information national trends survey. *Int J Environ Res Public Health*, 20(1), January 2023.

[2] John M Jakicic, Kelliann K Davis, Renee J Rogers, Wendy C King, Marsha D Marcus, Diane Helsel, Amy D Rickman, Abdus S Wahed, and Steven H Belle. Effect of wearable technology combined with a lifestyle intervention on long-term weight loss: The IDEA randomized clinical trial.

[3] Kuan-Yu Chen, Jungpil Shin, Md Al Mehedi Hasan, Jiun-Jian Liaw, Okuyama Yuichi, and Yoichi Tomioka. Fitness movement types and completeness detection using a Transfer-Learning-Based deep neural network. *Sensors (Basel)*, 22(15), July 2022.

[4] Jiangkun Zhou, Wei Feng, Qujiang Lei, Xianyong Liu, Qiubo Zhong, Yuhe Wang, Jintao Jin, Guangchao Gui, and Weijun Wang. Skeleton-based human keypoints detection and action similarity assessment for fitness assistance. pages 304–310, 2021.

[5] Zhonghan Zhao, Shanzhen Lan, and Shujun Zhang. Human pose estimation based speed detection system for running on treadmill. pages 524–528, 2020.

[6] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. 2017.

[7] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. 2016.

[8] Lora E Burke, Valerie Swigart, Melanie Warziski Turk, Nicole Derro, and Linda J Ewing. Experiences of self-monitoring: successes and struggles during treatment for weight loss. *Qual Health Res*, 19(6):815–828, April 2009.

[9] Edwin Locke and Gary Latham. A theory of goal setting task performance. *The Academy of Management Review*, 16, 04 1991.

[10] John M. Jakicic, Kelliann K. Davis, Renee J. Rogers, Wendy C. King, Marsha D. Marcus, Diane Helsel, Amy D. Rickman, Abdus S. Wahed, and Steven H. Belle. Effect of Wearable Technology Combined With a Lifestyle Intervention on Long-term Weight Loss: The IDEA Randomized Clinical Trial. *JAMA*, 316(11):1161–1171, 09 2016.

[11] Timothy J Suchomel, Sophia Nimphius, and Michael H Stone. The importance of muscular strength in athletic performance. *Sports Med*, 46(10):1419–1449, October 2016.

[12] Robert B. Miller. Response time in man-computer conversational transactions. In *Proceedings of AFIPS '68 Fall Joint Computer Conference, Part I, San Francisco, CA, USA*, pages 267–277, 1968.