POLITECNICO DI TORINO

Master's Degree in Cinema and Media Engineering

Master's Degree Thesis

Development of a gym exergame using a low-cost RGB camera for digital health in telerehabilitation

Supervisors

Prof. Gabriella OLMO

Prof. Gianluca AMPRIMO

Dott.sa Claudia FERRARIS

Candidate

Giorgio BIANCHI

July 2024

Summary

The focus of this thesis is the development of an exergame prototype designed for the rehabilitation and telerehabilitation of the upper limbs. Development focused on creating a game set in a gym environment, using Google's MediaPipe library to detect the user's movements and spatial positioning.

The exergame was built using the Unity game engine, based on an existing open source solution that integrates MediaPipe's pose detection via a Python script with a Unity sample scene. From this, the solution was further developed by incorporating a server-side component, consisting of a NodeJS server and a MongoDB database, to store all data collected during the exercises. Real-time data is transmitted via WebSocket from both the Python code and Unity game engine, while CRUD operations, such as user's login and registration, recording successful repetitions and errors, are managed through REST APIs. Additionally, a web user interface (UI) was implemented to enable users to read and download all significant data collected, with filters available for user's name and surname, date and time, session, or specific exercises.

The architecture of this solution is designed to centralize the server, allowing multiple clients to connect simultaneously. This is especially important for telerehabilitation, enabling the possibility to retrieve data from users performing exercises remotely.

The exergame features a range of exercises focused on lifting the upper limbs, both laterally and frontally, in various modes: single lifts, alternating lifts, or simultaneous lifts. Users receive positive reinforcement through audio and visual cues during exercises. Additionally, the game can be used in an assisted mode, where game prompts can be guided by an operator via keyboard input, further allowing customization. To determine if a repetition is performed correctly, the game assesses the angle between the user's arm and the corresponding side of their torso. When the correct angle is achieved, the repetition is marked as correct. The game evaluates also the angle between the arm and the user's shoulder line; if the user fails to maintain the correct plane of movement, the game signals an error.

The objective of this study is to demonstrate that a markerless solution, utilizing an integrated webcam or simple RGB camera, can achieve a reliable and sufficient level of precision for the execution of rehabilitation exercises. During the development phase, data were analyzed to verify the accuracy and frequency of data recording.

Furthermore, the study highlighted the accuracy of Google's MediaPipe pose detection. The system performed well under optimal lighting conditions, where the subject was evenly exposed and the background had high contrast. Conversely, performance declined in suboptimal lighting conditions, such as when one side of the subject was overexposed or the background contrast was poor.

Acknowledgements

I would like to express my genuine thanks to Professor Gianluca Amprimo and Dr. Claudia Ferraris, for their guidance and support throughout the development of this work. Their expertise and insight have been instrumental in shaping my research and work. I am also deeply thankful to Professor Gabriella Olmo, for her insightful feedback and constructive criticism, which significantly enhanced the quality of this work. I would like to express my appreciation to Politecnico di Torino and Consiglio Nazionale delle Ricerche - Istituto di Elettronica e Ingegneria delle Informazione e Telecomunicazione for providing access to their laboratory during the course of my thesis research, which has been essential in conducting experiments and gathering data for this study.

Table of Contents

List of Tables

List of Figures

Acronyms

API	Application programming interface
CNN	Convolutional neural network
DAO	Data access object
DBMS	Database management system
DOF	Degrees of freedom
FPS	Frames per second
GPU	Graphics processing unit
HTTP	Hypertext transfer protocol
JSON	JavaScript Object Notation
ML	Machine learning

NPM	Node package manager
REST	Representational state transfer
RGB	Red blu green
RGB-D	Red blu green - depth
TCP	Transmission control protocol
UI	User interface

Chapter 1

State of the art

1.1 Introduction

As highlighted by recent studies, in the past few years the analysis and rehabilitation of patients in healthcare has often relied on technological solutions, including video analysis, pose detection, and development of exergames. These solutions and studies often proved that with the right tools and methodologies, the rehabilitation efficiency can be improved and significant data can be collected, to better study and understand the effects of rehabilitation.

In particular, one of the studies collected (Garcia-Agundez et al., 2019) [1] analysed the main solutions proposed in the medical field and the technologies primarily used, by collecting and analyzing sixty-four publications. This resulted in a comprehensive analysis of the main technologies, showing that the use of reliable and safe technologies for patients, such as Microsoft Kinect and Wii Balance Board, leads to solutions that are not only as effective as traditional motor rehabilitation but in some cases even more so. Additionally, the possibility of continuing rehabilitation through exergames at home is emphasized, using systems that allow for both autonomous rehabilitation and continuous remote monitoring by healthcare personnel.

Other studies analysed the performances of single camera systems, with markerless pose detection solution, comparing them with conventional three-dimensional solutions, such as RGB-D camera or marker based solutions [2][3]. Chung et al. [4] analysed and compared open source skeleton-based human pose estimation systems and algorithms; the findings of this study validated our choice to use Google's MediaPipe library, as explained in the sections 1.3.4 and 2.2.

The present work is based on these findings and constitutes an attempt to apply and expand this field of research, studying the possibility to use simple RGB cameras, such as integrated webcams of PCs, to reliably capture the pose of the users in a consistent and precise manner. Using markerless pose detection solutions with these devices, would not require the mandatory use of RGB-D cameras such as Microsoft Kinect, inherently less suited for telerehabilitation, given the complexity of use and higher costs for generic users.

1.2 Research methodology

To understand the current technological landscape, a literature search was conducted using PubMed database. Articles published from 2019 onwards were retrieved, using the keywords "exergame", "markerless", "pose detection", "pose estimation" or "rehabilitation" and combinations of these terms.

The search produced this results:

1. 528 results were found for the combination "rehabilitation exergame";
2. 45 results were found for the combination "markerless pose detection"
3. 155 results were found for the combination "markerless pose estimation"
4. no results were found for the combination "exergame mediapipe"

The figure 1.1 visualizes the results distribution, showing an increase on publications about these topics in the last five years (for 2024, only the first six months have been considered).

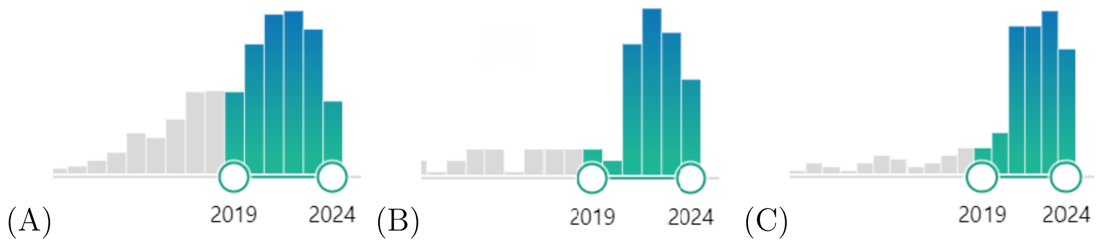


Figure 1.1: Search results distribution for "rehabilitation exergame" (A), "markerless pose detection" (B) and "markerless pose estimation" (C).

To gather the results reported below, a subset of twenty publications were analysed:

1. one published in 2019;
2. one published in 2020;
3. two published in 2021;

4. eight published in 2022;
5. five published in 2023;
6. three published in 2024.

The research excluded all the results not regarding of exergames, virtual reality or video analysis, as they were not relevant in this context.

1.3 Results

1.3.1 Traditional rehabilitation versus telerehabilitation

A study conducted in 2022 (Hashemi, Yazdan et al., 2022)[5] indicates that exergames for upper limb motor rehabilitation, both supervised and unsupervised, using Microsoft Kinect, could improve certain aspects of sensory and motor functions in patients with upper limb impairments. Therefore, these technological solutions could be used in telerehabilitation, especially considering the limitations induced by the COVID-19 pandemic.

In another publication (Chuang, Chieh-Sen et al., 2022), twenty-three studies with 949 participants were analysed. The results indicate that exergames and virtual technology-assisted rehabilitation lead to significant improvements in balance and gait outcomes compared to usual treatments and other active control interventions [6][7].

These considerations highlight the potential of exergames and virtual technology-assisted rehabilitation as effective tools for improving motor functions and overall physical outcomes, particularly in the context of telerehabilitation.

1.3.2 Game as a rehabilitative tool

In the same publication mentioned earlier (Chuang, Chieh-Sen et al., 2022)[6], it was also analysed that there were no significant changes in the depressive levels of patients who underwent rehabilitation through exergames, neither positive nor negative. The evaluation of acceptability results also shows that all the exergames examined were adequately tolerated, as indicated by low dropout rates.

Another study conducted in 2023 (Barth, Marcus et al., 2023)[8] demonstrates the beneficial effects of exergame-based training under review and confirms the assumption of a similar impact in a home setting. The software used, which relies on Microsoft Kinect technology, meets patient approval, and the group dynamics appear to provide additional support for the desired goal of improving mobility. These dynamics should be considered an essential aspect of video games in therapy.

Chen et al. in 2023 [9] studied the effect of VR exergames on adults over sixty-five years old, for a total of twelve randomized controlled trials and 482 older adults, and proved that adults that completed exergames interventions improved their cognitive outcome as well as improve their balance.

Ultimately, other recent studies analysed the available solutions via randomized controlled trials that use pose detection combined with exergames or VR experiences (Manser et al. 2024, Pelaez-Velez et al. 2023) [10][11], studying the beneficial support of exergames in rehabilitation. These studies highlight that the use of games and VR experiences proved beneficial especially in the treatment of stroke [11], in addition to traditional physiotherapy.

Overall, these findings underscore the potential of exergames as a viable therapeutic tool, offering significant cognitive and physical benefits while maintaining high levels of patient acceptability and engagement.

1.3.3 Single camera markerless systems

In 2021 and 2022 different studies (Stenum et al. 2021, Scott et al. 2022) [12] [3] analysed healthcare applications that were using a single camera markerless motion capture solution. The study conducted by Scott et al. [3] analysed fifty different solutions and concluded that single camera markerless systems performed well when single plane measurements were involved, compared to three-dimensional marker-based solutions, but they were less effective at out-of-plane tracking and capturing fine movements, such as finger tracking.

Another study (Wade L. et al. 2022) [2] analysed the main limitations and applications of markeless motion capture methods, especially in bio-dynamics applications. The study exhibits that markerless solutions can be limited in capturing data with high accuracy and high frequency, in particular the possible limitations are:

1. self occlusion: a well know problem of two-dimensional motion capture [13][14], self occlusion of the captured joints can result in a data loss for one or more frames and inconsistent data collection;

2. most of the algorithms have been trained to extract only two point for each segment, but three points are required to analyse all six degrees of freedom (DoF) of the body or a part of it;

3. the pose detection algorithms are developed based on the assumption that the camera is perfectly aligned with the frontal or sagittal plane movements.

Despite these limitations, markerless motion capture systems continue to show promise in specific applications, offering valuable insights and practical benefits for healthcare research.

1.3.4 Google's MediaPipe usage

Among the solutions analysed, some stand out for their use of the Google MediaPipe library as a system capable of near real time pose detection and estimation.

The first solution considered (Güney, Gökhan et al., 2022)[15] utilizes MediaPipe's hand landmark detection for estimating hand tremors in patients with Parkinson's disease. In the experiment, hand movement was measured both with an accelerometer and through video estimation using MediaPipe to identify the accuracy of tremor perception via MediaPipe. Although the research results were positive, it does not serve as a comparison for the project in this thesis because it is not an exergame or a rehabilitation project, but solely a video analysis.

The second and most recent case considered (Jansen, Talisa S et al., 2024)[16] involved the use of MediaPipe, also in terms of video analysis, for the automated recognition of eyelid closure in neuro impaired patients and healthy individuals, to compare the results of the two groups analysed. In this case, MediaPipe's face landmark detection functionality was used.

MediaPipe's pose detection was used in detecting post-stroke compensatory movements (Lin et al., 2023)[17]. The study shows that, despite some limitations, the solution performed well in near real time analysis of the patients movements.

The present work will be based on Google's MediaPipe solution, given the numerous findings on the use of Google's MediaPipe in real time or near real time applications [17][18][19], proven to be effective for rehabilitation purposes and in healthcare context.

Furthermore, from a technical point of view, MediaPipe Pose:

1. outputs 33 landmarks, or keypoints, a more than sufficient number of joints for the purpose of this prototype, as well as for future implementations or exergames based on this solution [20];

2. can estimate the pose on three different models, lite, full or heavy; this allow easy settings modification to evaluate which model performs best [20];

3. it was specifically trained for fitness-related applications [21];

4. it's lightweight and multi device, even on a mobile device, using the CPU, it will work in near real time at 30 FPS [21];

5. its sampling frequency of 30 FPS matches the capture FPS of most of the webcams or simple RGB camera in use [21][20];

6. the main requirement to use the solution, that is keeping the head always visible so that the pose estimation can be correct [21], while it's prone to user error, it does not represent a blocking issue for the purpose of this work.

1.4 Conclusions

Exergames and non-immersive virtual reality technologies show promising potential for the rehabilitation of neuro impaired patients, by improving upper limb sensorimotor functions, balance, and gait. The rising number of solutions explored in the past five years shows growing interest in these technologies, especially markerless and easy to use solutions.

Rehabilitation through software/games is well accepted by patients, and group dynamics offer additional benefits, suggesting that these technologies could represent a significant advancement in rehabilitation strategies[8]. The use of simple RGB cameras combined with markerless pose detection solutions proved to be effective and functional, although with some limitations, if used together with traditional physiotherapy [3]. The ease of use of these solution could be the key factor to allow testing of telerehabilitation solutions, both based on exergames or video analysis. In the past five years, a lot of different open source solutions based on two-dimensional cameras were developed with different limitations, applications and performances [12]. Analysing different solution, the present work will be developed using Google's MediaPipe solution, given its fitness aimed training and proved efficiency in real time scenarios and healthcare applications [19][17][18].

Other studies show that the overall effectiveness still remains inconclusive, and further high-quality studies with a larger number of cases are needed [22]; highlighting the limitations that relatively new technologies and solutions face.

This work and solution aim to fit into this overview of possible solutions for the rehabilitation and telerehabilitation of impaired patients, providing patients with a simple, marker-less, intuitive, and engaging tool to accompany them on their rehabilitation journey, and providing healthcare personnel with an analytical tool to verify the health and recovery status of patients through consultation of data from various therapy sessions.

Chapter 2

Goals and methodology

2.1 Goals

The goal of this thesis, developed through the collaboration between Politecnico di Torino and Consiglio Nazionale delle Ricerche (CNR) - Istituto di Elettronica e Ingegneria dell'Informazione e delle Telecomunicazioni (IEIIT), is to create a prototype of an exergame that allows users to follow a rehabilitation program for the upper limbs.

Exergames, short for "exercise games," are video games that combine physical activity with interactive gaming, using technology to track and respond to the player's body movements to promote exercise and fitness in an engaging and entertaining manner [23].

Specifically, the research focused on developing an exergame based on the Google's MediaPipe library, using a computer webcam, and avoiding the use of specialized RGB-D cameras like Microsoft Kinect [24]. In a second phase, the aim of this project is to be used in rehabilitation and telerehabilitation for patients affected by Parkinson's disease or patients affected by cerebral stroke with minimum or no assistance.

The initial goal is to develop a prototype that can analyse the accuracy of MediaPipe's pose detection and its performance in a rehabilitation context, recording all data and users' positions during the rehabilitation session. The aim is to develop a solution that can record a number of information that can be relevant to analyse the movement and possibly the reaction time and amplitude of movement of the users.

2.2 Technology stack

Developing an exergame that accurately tracks human movement within a three-dimensional environment necessitates the implementation of a pose detection solution. With numerous options available, selecting the appropriate one is crucial. This section aims to provide a non-exhaustive overview of the main pose detection open source solutions, with a more specific overview on Google's MediaPipe library models.

2.2.1 Human pose estimation

Human pose estimation in a three-dimensional environment is employed to predict joint movements and positions in space, which are then translated into a three-dimensional model. There are three types of modeling used, kinematic, planar or volumetric.

- **Kinematic**: used by most methods, it represents human body as an entity formed by joints and limbs. It's also called "skeleton-based model", and it is mainly used to capture the relations between body parts, but it doesn't give information on external shape or texture.
- **Planar**: used in two-dimensional estimation to represent shape and texture of a human body. Each body part is represented by multiple rectangles that approximate the human body shape analysed.
- **Volumetric**: used for three-dimensional pose estimation, it represents the human body with a three-dimensional mesh. While it's the most precise model representation, it's not suitable for real-time pose estimations.

Most of the solutions use the kinematic model, which contains both body kinematic structure and body shape information. This is also the model most suitable for the solution, aimed to calculate the angles between arms and the user's torso.

In pose estimation, two primary approaches are employed to achieve accurate results.

- **Bottom-up** approaches involve the initial estimation of individual body joints, followed by the grouping of these joints to form a complete pose. This method focuses on identifying and localizing each joint independently before assembling them into a coherent pose representation.
- **Top-down** approaches begin with the detection of persons within an image or video frame. Once individuals are identified, the system proceeds to estimate

the positions of their respective body joints. This method prioritizes detecting people as a whole entity before refining the localization of specific body parts.

The choice between bottom-up and top-down approaches depends on factors such as computational efficiency, accuracy requirements, and the specific characteristics of the target application.

OpenPifPaf

In 2021, Kreiss et al. developed OpenPifPaf [25], a bottom-up approach open-source library designed for multiperson human pose estimation. This library focuses on detecting, associating, and tracking semantic keypoints in video data, even in complex scenes. The bottom-up methodology of OpenPifPaf is notable for its efficiency, stable field representation, accuracy and performance, often surpassing top-down methods. The model architecture features a shared base network, either ResNet or ShuffleNetV2, without max-pooling. Central to the framework are the Composite Intensity Field, which represents joint intensity, and the Composite Associations Field, which forms associations to track poses; the Composite Fields detect and construct a spatio-temporal pose, creating a single, connected graph with nodes representing semantic keypoints (such as body joints) across multiple frames. For temporal associations, the Temporal Composite Association Field (TCAF) is used; this approach enhances the model's ability to track poses over time, contributing to its robust performance in various scenarios.

MoveNet

Released in 2021 by Google, MoveNet is a real-time, bottom-up pose detection model developed by Google to identify seventeen keypoints on a single person using heatmaps [26][27]. Similar to MediaPipe, MoveNet employs the MobileNetV2 architecture [28] as its feature extractor, which includes four prediction components: the person center heatmap, keypoint regression field, person keypoint heatmap, and two-dimensional per-keypoint offset field. These elements work together to accurately predict human keypoints. MoveNet was trained on two datasets: the COCO dataset [29] and Google's internal dataset called Active. The model is available in two versions: Lightning and Thunder. Lightning provides faster inference times but with slightly reduced accuracy compared to Thunder.

OpenPose

OpenPose (Cao et al. 2019)[30] is one of the most well know pose detection solutions developed in recent years. It's aimed specifically to the pose detection and pose estimation of multiple bodies in a single image with high accuracy, using non-parametric representation, which is called Part Affinity Fields (PAFs), to learn to associate body parts with individuals in the image. In comparison to other solutions, OpenPose also rely on combined detection for body and foot placement, reducing the inference time. Based on COCO [29] and MPII [31] datasets, the applications for OpenPose can be many and not specific to a single area.

DeepCut

DeepCut, developed by Pishchulin et al. in 2016 [32] simultaneously manages detection and pose estimation, determining the number of people in a scene, identifying occluded body parts, and distinguishing body parts of individuals in close proximity. It was developed primarily for applications with multiple people present in the frame at the same time. Unlike other solutions analysed in this section, DeepCut uses a joint formulation to both detect and estimate the body poses. It employs a partitioning and labeling strategy for a set of body-part hypotheses generated by convolutional neural network (CNN)-based part detectors. This strategy, framed as an integer linear program, implicitly performs non-maximum suppression on the part candidates and groups them into body part configurations that respect geometric and appearance constraints.

AlphaPose

Born in 2016 and perfected in 2018, AlphaPose was theorized and implemented by Fang et al. [33]. The solutions was developed to tackle the pose detection and estimation of multiple bodies in a frame, using a regional multi-person pose estimation (RMPE) framework to facilitate pose estimation in the presence of inaccurate human bounding boxes. The solution is composed by three components, a Symmetric Spatial Transformer Network (SSTN), a Parametric Pose Non-Maximum-Suppression (NMS), and a Pose-Guided Proposals Generator (PGPG).

Google's MediaPipe

Google's MediaPipe was launched in 2019 as a framework for pose, head and hand detection and estimation, to be used on multiple devices, lightweight and trained specifically for fitness applications [34]. The pose detection task of MediaPipe is based on BlazePose model (Bazarevsky V., Grishchenko I. et al., 2020)[21] and outputs thirty-three keypoints (called landmarks), providing a full representation of a body and its movements in real-time.

Conclusions

Based on the analysis of detailed solutions, as outlined also in section 1.3.4, this study will focus on Google's MediaPipe solution.

1. Other frameworks were trained for multi person detection [33][30][32][25], which is out of scope in the present context.

2. Performances in real time are key in this context and MediaPipe proved to be reliable in different applications [17][15][16]; its lightweight and multi platform approach ensure satisfactory performances even on systems without a dedicated graphics processing unit (GPU).

3. Easy to use and to work on with a detailed Python guide [35][36]; the extensive documentation allow for an easy integration with Unity.

4. Even if the solution performed sub par in still image pose detection, compared to other solutions, its real-time video detection and estimation capabilities are among the highest in the frameworks evaluated [4].

2.2.2 Google's MediaPipe specifications

The MediaPipe Pose Landmarker task detects landmarks of human bodies in an image or video. The task identifies body locations, analyse posture, and categorize movements and uses machine learning (ML) models that work with single images or video. It outputs body pose landmarks in image coordinates and in three-dimensional world coordinates.

A series of models are used to predict the pose landmark. The first model, a pose detection model, detects the presence of a human bodies in the image or video, while a second model bundle, pose landmarker model, creates the kinematic model with landmarks (or keypoints).

This bundle, composed by a lite, full and heavy model, uses a convolutional neural network similar to the MobileNetV2 architecture (Sandler M., Howard A. et al., 2019) [28] and it's optimized for real-time applications. The pose landmarker model is a variant of the BlazePose model (Bazarevsky V., Grishchenko I. et al., 2020)[21] using GMHU (Hongyi Xu, Bazavan E. G. et al., 2020)[37] an end to end pipeline to estimate full three-dimensional body pose.

The BlazePose model outputs 33 landmarks (or keypoints) describing the approximate location of body parts:

1. nose;

2. right eye (3 keypoints): inner, center, outer;

3. left eye (3 keypoints): inner, center, outer;
4. ears (2 keypoints): right, left;
5. mouth (2 keypoints): right corner, left corner;
6. shoulder (2 keypoints): right, left;
7. elbow (2 keypoints): right, left;
8. wrist (2 keypoints): right, left;
9. pinky knuckle (2 keypoints): right, left;
10. index knuckle (2 keypoints): right, left;
11. thumb knuckle (2 keypoints): right, left;
12. hip (2 keypoints): right, left;
13. knee (2 keypoints): right, left;
14. ankle (2 keypoints): right, left;
15. heels (2 keypoints): right, left;
16. foot index (2 keypoints): right, left.

The output is a 33×5 array, representing the world landmark position with X, Y, Z coordinates and the visibility and presence of the landmark.

From the model documentation [20]:

1. X, Y coordinates are local to the region of interest and range from [0.0, 255.0].
2. Z coordinate is measured in "image pixels" like the X and Y coordinates and represents the distance relative to the plane of the subject's hips, which is the origin of the Z axis. Negative values are between the hips and the camera; positive values are behind the hips. Z coordinate scale is similar with X, Y scales but has different nature as obtained not via human annotation, by fitting synthetic data (GMHU model) to the two-dimensional annotation. Z is not metric but up to scale.
3. Visibility is in the range of [0.0, 1.0] and after user-applied sigmoid denotes the probability that a keypoint is located within the frame and not occluded by another bigger body part or another object.
4. Presence is in the range of [0.0, 1.0] and after user applied sigmoid denotes the probability that a keypoint is located within the frame.

Starting from this the pose landmarker model outputs the normalized coordinates and world coordinates for each landmark.

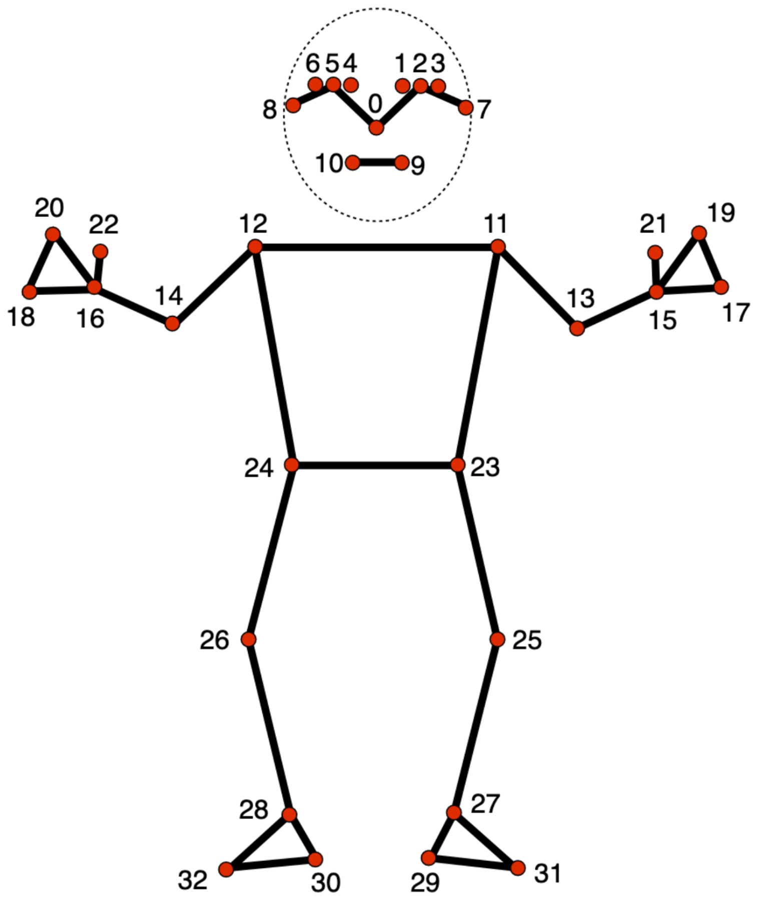


Figure 2.1: Pose landmarker model

2.2.3 Game Engine

A game engine is a software framework designed to facilitate the development and creation of video games. It provides a suite of tools and features, such as rendering graphics, simulating physics, handling input, managing audio, and scripting, which developers use to build and manage game environments. Game engines streamline the game development process by offering reusable components and systems, allowing developers to focus more on game design and functionality rather than underlying technical details. Examples of popular game engines include Unity [38], Unreal Engine [39], and Godot [40]. We chose to use the Unity game engine for this project, because it's widely used for academic purposes and the open source solution this prototype is based on was already developed with Unity game engine.

Unity

Unity is a widely-used game engine developed by Unity Technologies [38], known for its versatility and user-friendly interface. It supports the development of two-dimensional and three-dimensional games and interactive experiences across multiple platforms, including consoles, PCs, mobile devices, and virtual/augmented reality. The engine uses C# for scripting, providing robust capabilities to control game behavior and logic.

The package manager in Unity allows easy integration and management of various packages and plugins, ensuring that projects can be extended and customized efficiently. In the context of this work, WebSocketSharp [41] and Newtonsoft JSON.Net [42] packages were used.

Beyond game development, Unity is widely used in research and academic contexts. Its flexibility and powerful visualization capabilities make it a valuable tool for creating simulations and interactive educational content. Unity also enables rapid prototyping and iteration, allowing developers to test and refine their ideas swiftly.

Overall, Unity is a powerful and versatile game engine, supporting a wide variety of game genres and interactive experiences; for these reasons it was chosen as the game engine to develop the exergame work of this thesis.

2.2.4 Server and database

The implementation of a dedicated server and database for data persistence addresses the need to collect data from multiple devices and different clients, as well as the possibility to read and download the data not necessarily on the same device where the game is installed.

Furthermore the server and database are developed keeping in mind the scalability and development of the solution, opting for a No-SQL database for its scalability and support of both structured and un-structured data and the efficiency in storing big volumes of data. The server as well, with a MVC-inspired design pattern [43] allows to easily add models, services and DAO classes to save, access and manipulate different types of data and information.

Node.js

Node.js is a runtime environment that allows to run JavaScript on the server side [44]; it's known for its efficiency and scalability. Node.js uses an event-driven, non-blocking I/O model, making it ideal for building real-time applications (Dalbard and Isacson, 2021) [45].

A key feature of Node.js is the Node Package Manager (NPM) [46], which provides access to a vast library of reusable packages and modules, accelerating the development process. Its asynchronous nature and single-threaded event loop enable handling multiple connections simultaneously, ensuring high performance without the overhead of multiple threads.

Node.js is flexible and versatile, suitable for creating RESTful APIs, microservices, and serverless architectures. Its rapid prototyping capability allows for quick testing and iteration.

MongoDB

MongoDB [47] is a high-performance NoSQL database known for its ability to handle real-time data efficiently [48][49]. Utilizing a document-oriented model, it stores data in JSON-like documents, allowing for dynamic schemas and flexible data management. The architecture of MongoDB supports horizontal scaling through sharding, distributing data across multiple servers to ensure high availability and quick response times. This makes MongoDB particularly suitable for applications requiring real-time processing. With robust querying and indexing capabilities, MongoDB efficiently retrieves data from large datasets. Its aggregation framework supports real-time analytics and data transformations within the database. Key features like replication and automatic failover ensure data durability and reliability. Its schema-less design allows for flexible data modeling and rapid development, beneficial for agile projects and research requiring quick adaptation.

2.2.5 Web UI development

This section outlines the technology stack used to develop the web interface. The web UI must be compatible with multiple browsers and responsive to desktop screens of various sizes.

jQuery

jQuery [50] is a fast, small, and feature-rich JavaScript library. It simplifies HTML document traversal, event handling, and animation, making it easier to manage and manipulate web pages. jQuery's cross-browser compatibility ensures that code works consistently across different browsers. jQuery's simplicity and ease of use facilitates rapid development and reduces the complexity of JavaScript programming.

Bootstrap

Bootstrap [51] is a widely-used front-end framework for developing responsive and mobile-first websites. Bootstrap's responsive grid system allows for flexible layouts that adapt to various screen sizes, ensuring a consistent user experience across devices. Its pre-styled components and extensive documentation make it easy for developers to create aesthetically pleasing and functional interfaces quickly. Bootstrap's widespread adoption and strong community support make it a reliable choice for web development. In the context of this work, the last stable version 5 of Bootstrap was used.

2.2.6 Communication protocols

The detailed solution is based on different nodes that require specific communication protocols. This section provides an overview of the communication protocols used in the client-server architecture detailed in section 3.1.1.

TCP connection

TCP [52] is a reliable and connection-oriented protocol, ideal for ensuring the correct and ordered delivery of data between the client and server, which are fundamental characteristics for the type of application being developed.

1. **Reliability**: TCP ensures that data packets sent from the client to the server arrive in an orderly and lossless manner.
2. **Stable Connection**: As a connection-oriented protocol, TCP establishes a dedicated connection between the client and server before starting data transmission, guaranteeing continuous and uninterrupted communication.
3. **Congestion Control**: TCP includes mechanisms for managing network congestion, reducing the risk of network overload and improving data transmission efficiency.

WebSocket

The WebSocket communication protocol [53] establish a full duplex communication between the client and the server within a single TCP connection, facilitating real-time data transfer. Data is transmitted keeping the connection open, until one of the two involved, client or server, closes the connection. For this reason the WebSocket protocol was used to send all the real-time data captured by Google's MediaPipe pose detection to the server, as well as all the FPS informations gathered to test the solution performances. The following table details all the WebSocket communications of the developed solution.

Sender	Receiver	Goal
MediaPipe Python Script	Node.js Server	Performance recording (FPS)
MediaPipe Python Script	Node.js Server	Save landmark position
Unity Game	Node.js Server	Performance recording (FPS)

Table 2.1: Instances of WebSocket communication in the solution

HTTP

HTTP (Hypertext Transfer Protocol)[54] was used, especially with REST architecture, in the communications between the Unity game engine and the server, as well as in the communications between the web interface and the server. The following scheme in figure 2.2 sums up all the main APIs exposed by the server.

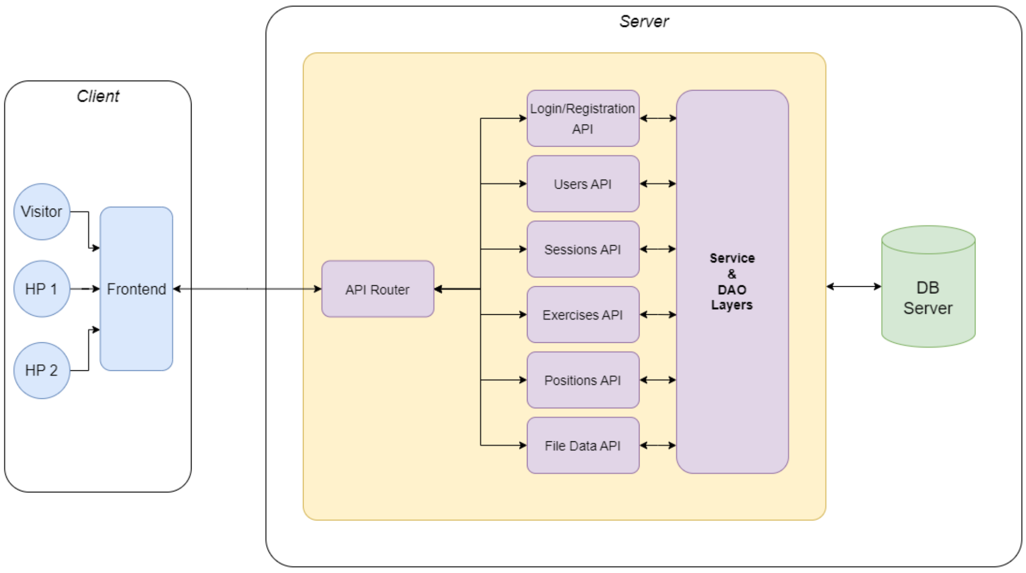


Figure 2.2: Main APIs provided by the server

Pipe

A pipe is used for passing information directly from the running Python process to the Unity process. In this context it's used to share the pose detection data in real-time. By writing data to a pipe, the Python script can immediately send information to Unity, which reads from the pipe.

The information provided from Python are:

1. a timestamp in microseconds;
2. a session unique id (UUID [55]) generated when the Python script starts;
3. for each landmark of the MediaPipe's pose detection: the x, y and z global position values and their visibility value.

These information are written by the Python code in a string in a shared format and read by Unity in a running thread (refer to section 3.2). The collected data is used to render the humanoid figure, following the movement of the user in real time. This allows for efficient, in-memory data transfer without the need for intermediate storage. In this context, it ensures that the two processes running on the same computer can quickly and reliably exchange information, enabling them to work together effectively.

2.3 Research methodology

The research on the rehabilitation exergame focused on the target audience for the prototype, the technologies chosen for use, and the project requirements outlined in the initial stages of thesis development.

2.3.1 Target

The final target audience for the prototype includes patients with Parkinson's disease or those who have experienced a cerebral stroke, both of whom often face compromised mobility. The primary objective of the game is to facilitate rehabilitation by enabling patients to perform therapeutic exercises tailored to their specific needs. For patients with cerebral stroke, the game is designed to assess and compare the recovery of the affected side with the healthy side, allowing for a detailed evaluation of rehabilitation progress.

Additionally, the exergame is designed to be inclusive for sitting users who have compromised or impaired lower limb mobility. This feature ensures that patients who cannot stand or walk can still participate fully in the rehabilitation exercises. The game will offer exercises that can be performed from a seated position, targeting the upper body.

By incorporating these features, the exergame aims to provide a comprehensive, accessible, and effective tool for physical therapy, supporting the recovery and improvement of motor functions in patients with significant mobility challenges.

2.3.2 Development requirements

The main requirements of the solution are:

1. rehabilitation exercises for the upper limbs, through lateral and frontal lifts. The difficulty should be selectable and should be based on three angles, 60°, 75° and 90°;
2. easy to use and engaging UI;
3. user authentication is required, both for the game and the web dashboard;
4. exercises types: single lifts, alternate lifts and simultaneous lifts;
5. two exercise modes: with a timer deadline or with a specified number of repetitions;
6. it should be possible for another user to indicate which arm the user should lift and when;
7. the solution should register all the position data from Google's MediaPipe library;
8. data should be searched and downloaded from a dedicated UI dashboard;
9. it should be possible to save automatically the data gathered in the current session.

2.4 Development methodology

The development methodology used for the prototype involved employing a Kanban board [56] to gather the macro-requirements and subsequently break them down into individual, actionable requirements. This approach ensured that all aspects of the project were systematically addressed and tracked, facilitating clear visibility and management of tasks throughout the development process.

To ensure the project remained aligned with its goals, a progress meeting was held every week. These meetings provided an opportunity to reassess priorities and make adjustments to the development process as necessary, reflecting an Agile-like software development structure [57].

For version control, GitHub [58] was utilized to maintain the source code and developer documentation. Project builds and releases followed semantic versioning (SemVer) [59], adhering to the `MAJOR.MINOR.PATCH` convention. Each version was tagged in the remote repository, ensuring a clear history of releases and facilitating easy reference to specific states of the project.

Developer documentation was written in English using markdown [60]. This choice was made to ensure that the documentation was accessible, easy to read, and could be immediately viewed on GitHub. Markdown's simplicity and readability made it an ideal format for maintaining comprehensive and clear documentation throughout the project's life cycle.

By integrating these methodologies and tools, the development process was both structured and adaptable.

Chapter 3

Prototype realization

3.1 Introduction

The development of this prototype serves as a starting point to introduce exergames using MediaPipe framework into a larger spectrum of solutions, aimed to the rehabilitation of neuro impaired patients, like Parkinson's disease affected patients or patients that suffered cerebral strokes.

While the prototype's user interface (UI) is written in Italian, to cater to the target, all the code and documentation are in English. This ensures that the technical aspects of the project are accessible to future developers who may contribute to or build upon this work.

The development of this prototype, as part of this thesis, aims to evaluate the performance of an end-to-end solution. This involves assessing the robustness of MediaPipe's pose detection and estimation capabilities, as well as the performance and usability of the exergame developed in Unity. An important aspect of this evaluation is the quality of the data collected during development and internal testing phases. This data include information such as the positions of each joint of the user during exercises, which is essential for analyzing movement accuracy and progress.

The development of the solution was organized so that it will be scalable and maintainable, with a versioning repository and developer's documentation, as well as a code organized and reviewed with ESLint [61], to ensure that the solution can be easily developed by others in the future.

3.1.1 Architecture

Given the requirements outlined in the previous chapter, particularly considering the future use of this prototype, it was decided to implement a solution with a client-server architecture, where the exergame, Python script and web UI dashboard are the clients, and the server is developed using Node.js , with MongoDB as Database Management System (DBMS). That means that the communication will be based on TCP communication protocol [52]. The following figure 3.1 details the high level architecture of the solution, as well as the communication protocols used between the nodes.

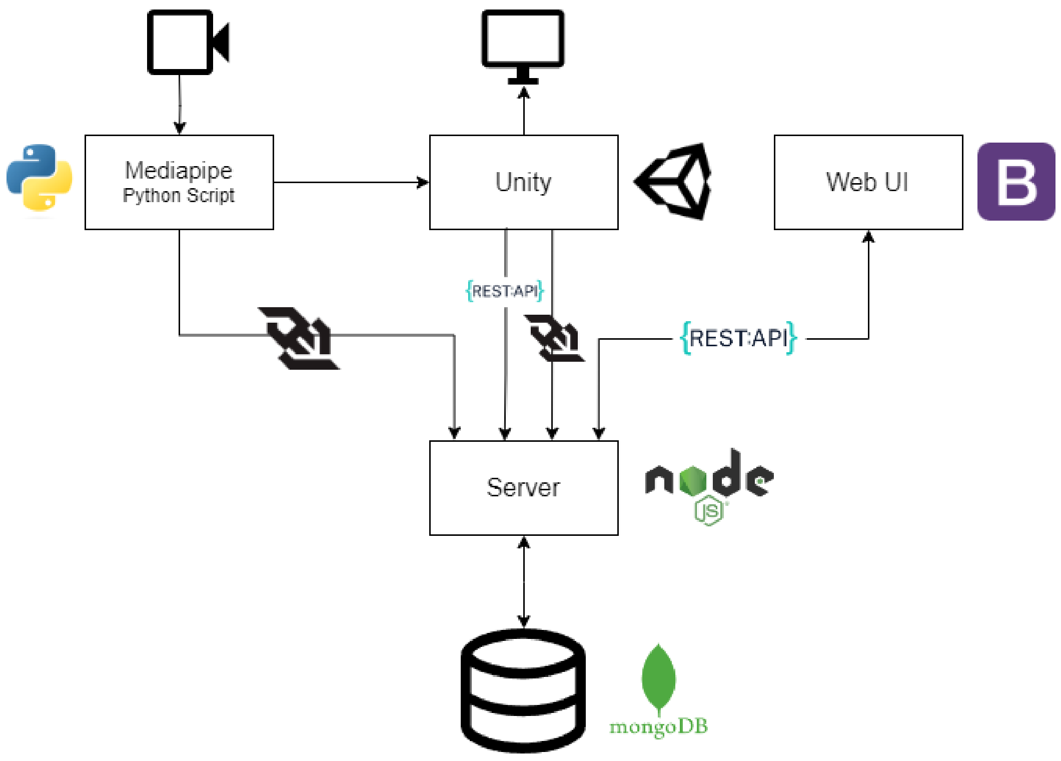


Figure 3.1: High-level architecture of the solution

3.2 MediaPipe pose detection development

The MediaPipe pose detection and estimation bundle was developed starting from the open source code developed by GitHub user ganeshsar [62] and expanded to address the needs of the solution. In particular, the existing solution is composed by a multi-thread process and the code is divided in three Python scripts:

1. `global_vars.py` : contains all the global variables used in the Python bundle;
2. `main.py` : launches the Python process, starting the BodyThread of the `body.py` script;

3. `body.py`: core script of the bundle, it starts the CaptureThread, BodyThread and WebSocketThread.

global_vars.py

Includes different parameters and variables used in the script bundle.

- `KILL_THREADS`: this variable is used to shut down the running thread if a keyboard input is given to stop the code execution.
- `DEBUG`: used to set the code in debug mode; if `True` shows the camera estimation output and print on console performances information.
- `RECORD_FPS`: used to determine if the performances information (Capture FPS, Theoretical FPS and Real FPS) should be sent to the server via WebSocket.
- `WEBCAM_INDEX`: used to determine which camera to use, if more than one camera is installed.
- `USE_CUSTOM_CAM_SETTINGS`: if `True` the parameters `FPS`, `WIDTH` and `HEIGHT` are used to capture the video.
- `FPS`: used only if `USE_CUSTOM_CAM_SETTINGS` is `True`. It sets the desired FPS (frames per second) for capturing the video.
- `WIDTH`: used only if `USE_CUSTOM_CAM_SETTINGS` is `True`. It sets the desired width in pixels for capturing the video.
- `HEIGHT`: used only if `USE_CUSTOM_CAM_SETTINGS` is `True`. It sets the desired height in pixels for capturing the video.
- `MODEL_COMPLEXITY`: used to determine which MediaPipe output model to use for the estimation, if lite (0), full (1), heavy (2) (see section 2.2.2 [36]).

These settings can be changed but can impact the performances, especially the parameters `DEBUG` and `MODEL_COMPLEXITY` have an impact on performances both perceived by the final user and measured in FPS. These findings are detailed in section 4.2.

main.py

The purpose of this script is to start the thread BodyThread of the `body.py` script and to kill all the threads when the session is closed. It's used to launch the Python solution execution.

body.py

The `body.py` script is the core script of the Python solution and it's main functionalities are:

1. establish a WebSocket connection with the Node.js server. If the connection fails, the script retries to connect. The server needs to always be reachable to use the solution;
2. start a thread (WebSocketThread) to send data to the server via WebSocket connection;
3. start a CaptureThread that captures video with the specified camera;
4. start the BodyThread to use the captured video as input for MediaPipe Pose Detection. The estimated positions are written to the pipe for use by the Unity game engine and sent to the Node.js server via WebSocketThread [36] [35];
5. create a unique id (UUID [55]) representing the session. This is used to identify the landmarks associated with the session.

In the method `run()` of the thread, the pose is captured with

```
    mp_pose.Pose(min_detection_confidence=0.80, min_tracking_confidence=0.5, model_complexity = global_vars.MODEL_COMPLEXITY,static_image_mode = False ,enable_segmentation = True) as pose:
```

It this case, the pose detection uses a `min_detection_confidence` of 0.80 and a `min_tracking_confidence` of 0.5. The code is structured so that the landmarks' positions are written in the pipe only if the specified pipe is open. The pipe is opened by the Unity code and in Python the thread checks if the pipe is open every second.

```
if self.pipe==None and time.time()-self.timeSinceCheckedConnection >=1:
    try:
        self.pipe = open(r'\\.\pipe\UnityMediaPipeBody', 'r+b', 0)
    except FileNotFoundError:
        print("Waiting for Unity project to run...")
        self.pipe = None
    self.timeSinceCheckedConnection = time.time()
```

The Python code writes a string with all the landmark positions as such:

```
for i in range(0,33):
    self.data += "FREE|{}|{}|{}|{}|{}\n".format(i, body_world_landmarks_world[i][0], body_world_landmarks_world[i][1], body_world_landmarks_world[i][2], str(session_id))
    landmarks.append({"x":'%.3f'%(body_world_landmarks.landmark[i].x), "y": '%.3f'%(body_world_landmarks.landmark[i].y), "z": '%.3f'%(body_world_landmarks.landmark[i].z), "visibility": '%.3f'%(body_world_landmarks.landmark[i].visibility), "landmark": i})
```

The `self.data` is written in the pipe, while the `landmarks` JSON array is used to save the landmarks positions in the server.

- First entry is the landmark `index`, corresponding to the joints presented in figure 2.1.
- Second, third and fourth entry are the x,y,z world positions estimated as described in section 2.2.2.
- Last entry is the `sessionId` generated for the present session.

The JSON body sent to the server is composed as follows.

```
{
    "sessionId" : string,
    "timestamp": float,
    "frameId" : integer,
    "landmarks" : [{
        "x" : float,
        "y" : float,
        "z" : float,
        "visibility" : float
        },
        . . .
    ]
}
```

A version 4 UUID [55] is generated at the start of the process and used to signal Unity and the server which session the landmarks are linked to. To improve code efficiency, the data is collected and sent in packages of thirty units, approximately once every second, depending on the performances. The `frameId` parameter is used to ensure that no package get lost in sending or receiving or saving data on the database, it's increased by one each time a frame is captured. The UNIX timestamp

[63] is generated in microseconds and it's used to evaluate the performance of the solution, as shown in the results chapter of this thesis.

3.3 Game development

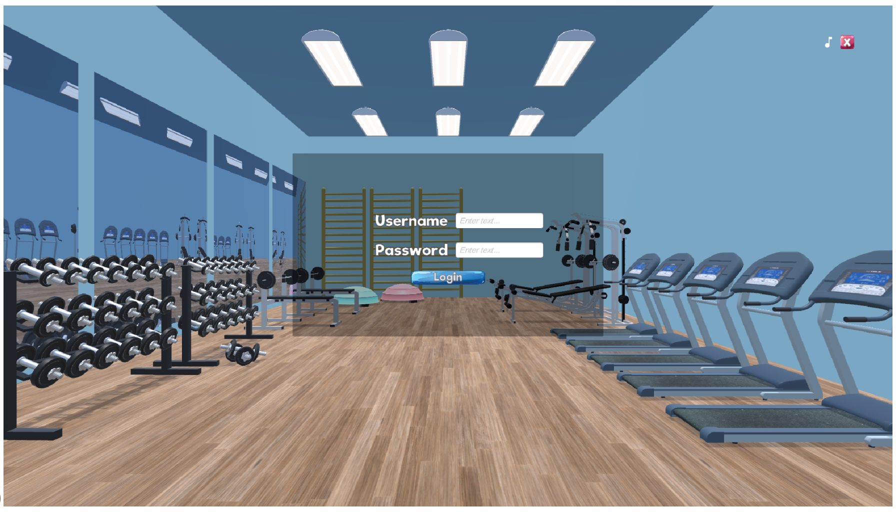
The exergame was developed starting from the requirements detailed in the section 2.3.2, and here summarized:

- create a set of exercises to help user with upper limbs rehabilitation;
- it should be possible to play the game with no help from another person, starting the exercises with hand gestures;
- user authentication is required;
- it should be set in a gym environment;
- the game should communicate successful repetitions as well as errors made by the users;
- it should be possible to play based on maximum time per exercise or maximum repetitions per exercise.

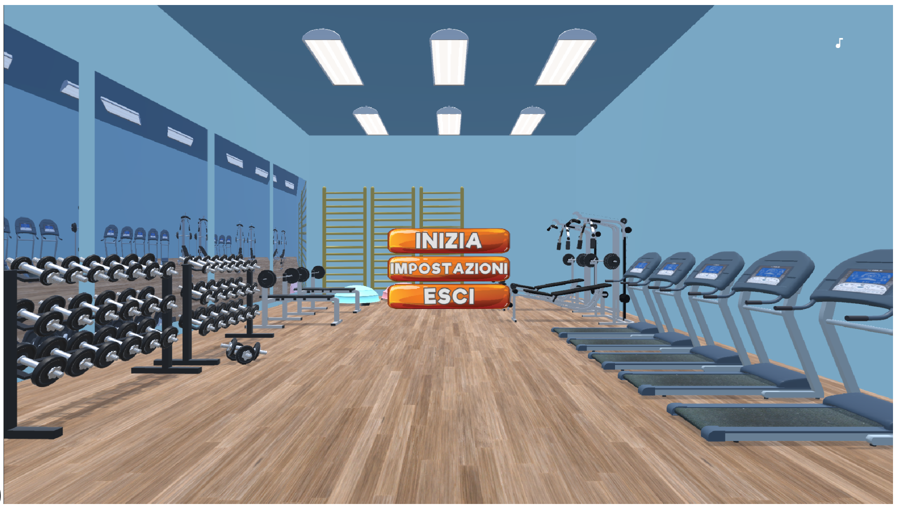
Given these requirements, the development started from an open source project by Github user ganeshsar [62] that tested the Google's MediaPipe pose detection inside of Unity using Python bindings. This was the starting point of the development, around which the environment, game mechanics and code improvements took place. The game is structured in Unity scenes:

1. **Login scene**: allows the user to log into the game with username and password;
2. **Main menu scene**: allows the user to start the game with default options, change options or exit the game;
3. **Settings scene**: allows the user to tailor the exercise session to their needs, as detailed in section 3.3.2;
4. **Exercise session scene**: core scene were the user plays and exercise; player movements are rendered in real-time via the pose estimation of MediaPipe.

To optimize the game, the environment is instantiated in the first scene and kept alive using the `DontDestroyOnLoad` method of Unity Engine. The same was done for the `AudioManager`, the component that plays the game soundtrack. Each scene also has a canvas on which all the UI components are rendered. Background



(A)



(B)

Figure 3.2: Login scene (A) and Main menu scene (B)

assets were provided to use for all the buttons in the game and the default font used was the Riffic Font (a sample is shown in figure 3.3). The figure 3.4 shows the different two-dimensional assets used as background for the buttons and text containers in the game.

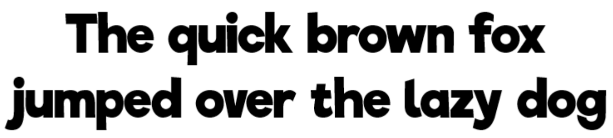

The quick brown fox
jumped over the lazy dog

Figure 3.3: Default font sample used for the game UI

Figure 3.4: 2D assets used as backgrounds for the game UI

3.3.1 Environment

To create the environment, the research group focused on a lighthearted theme, with bright colors and an easy setting, as represented by the two-dimensional assets and the font used for the UI. The environment was modeled using CC0 Public Domain assets [64] found online on CGTrader [65] and the scene was composed in the Unity game engine editor. Materials for all the assets were created directly in Unity, while the texture for the floor (formed by color, normal, roughness, smoothness, displacement and ambient occlusion maps) was downloaded from ambientCG [66], a library that allows to download CC0 public domain textures.

We decided to create a more open and airy environment with the use of mirrors on the left side, to create the illusion of a bigger room. The mirrors were created by placing secondary cameras on four panels and using the camera output as the panel's texture. The figure 3.5 shows the finished environment used in the prototype.

3.3.2 Settings

The settings panel allows to tailor the exercise session on the user's needs. The possible settings are detailed in the table 3.1.

After the user sets the desired criteria, or leave the default options, they can start the exercise session by clicking the "Inizia sessione" button. The figure 3.6 shows the game settings with default values.

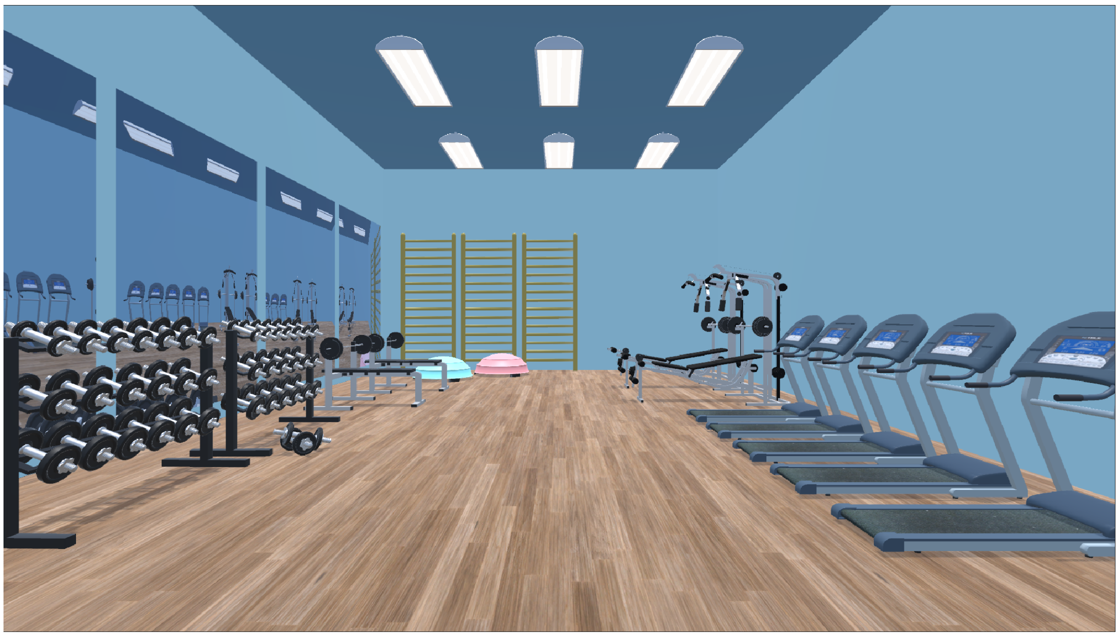
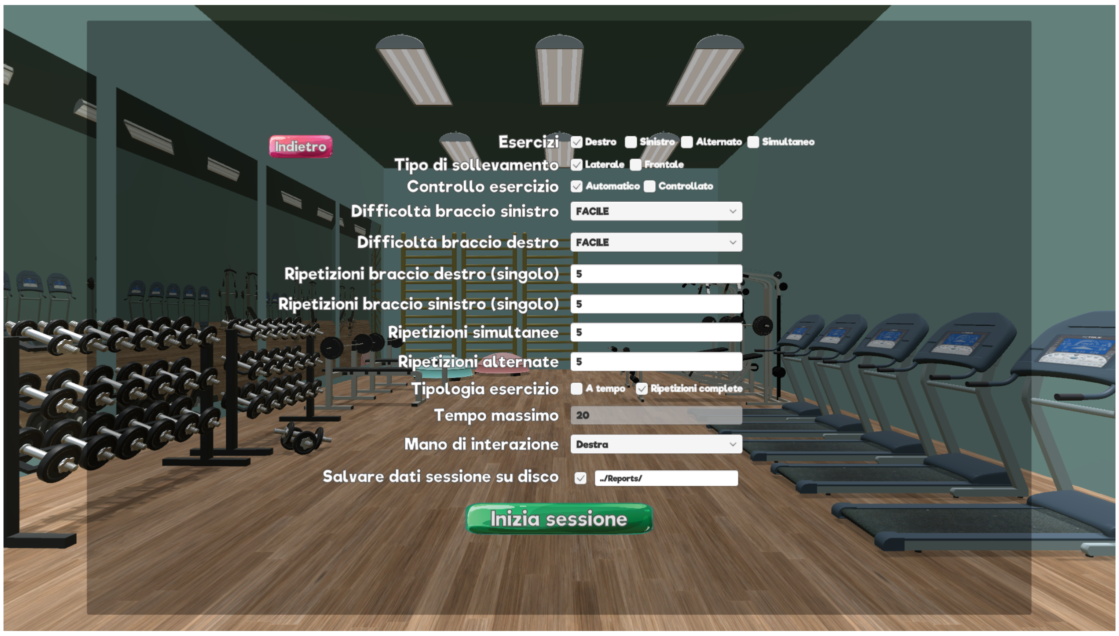


Figure 3.5: 3D environment used as background and setting for the game

Name	Purpose	Default
Exercises	Multiple selection of which exercise to perform, meaning which arm to lift: single right arm, single left arm, alternate arms, both arms simultaneously. More than one option can be selected.	Single Right Arm
Lifting mode	Select if frontal or lateral movement. The options are mutually exclusive.	Lateral
Exercise control	Select if the exercise are to be automatic or controlled by a second user. The options are mutually exclusive.	Automatic
Arm lift difficulty	Select the difficulty of the movement for the arm to be lifted (sixty, seventy-five or ninety degrees). It can be selected for either arm, as some users, particularly those affected by cerebral stroke, may have more difficulty lifting one arm than the other.	Easy (60°)
Number of repetitions	Can be written directly by the user in numeric values. One input for each exercise to be performed.	Five (5)

Exercise type	Choose whether the exercise ends when a specified time is reached or when the specified number of repetitions is completed. These options are mutually exclusive, so selecting one will disable the fields corresponding to the other.	Number of repetitions
Time limit	Set the time limit in seconds during which the user should do as much repetitions as possible.	20
Interaction hand	Select which hand to use to select the start of an exercise in automatic exercise control mode.	Right arm
Automatic saving data on disk	Allows the user to automatically save the data recorded on disk at the end of the training session. It also allows to specify a path. If a path is not provided, the file are downloaded and saved in the "Reports" folder in the root of the installed game.	True (saved in "Reports/" folder)

Table 3.1: Game settings and default values



Esercizi	✓ Destro ☐ Sinistro ☐ Alternato ☐ Simultaneo
Tipo di sollevamento	✓ Laterale ☐ Frontale
Controllo esercizio	✓ Automatico ☐ Controllato
Difficoltà braccio sinistro	FACILE
Difficoltà braccio destro	FACILE
Ripetizioni braccio destro (singolo)	5
Ripetizioni braccio sinistro (singolo)	5
Ripetizioni simultanee	5
Ripetizioni alternate	5
Tipologia esercizio	☐ A tempo ✓ Ripetizioni complete
Tempo massimo	20
Mano di interazione	Destra
Salvare dati sessione su disco	✓ ../Reports/

Figure 3.6: Game settings UI with default values

3.3.3 Game flow

Once the user logs in and edits the settings to their needs, the exercise session begins. This section provides a list of actions or events that describe the game flow during the rehabilitation session.

1. The users click on "Inizia sessione" (start session) from the settings page.
2. The superimposed UI changes, showing on the top of the screen the timer, back button and successful repetitions on the left, instructions in the center and the score, stop button and next exercise button on the right.
3. The humanoid figure is rendered into the environment, tracking the same movements of the user in real time.
4. A green spinning cube is rendered in front of the user, and in the instruction panel the user is invited to touch the cube with their interaction hand of choice to start the first exercise.
5. Upon the user touching the green cube, the cube disappears and two dumbbells are rendered and anchored to the hands of the user, depending on the exercise type; if the exercise is alternate or simultaneous lifts, one dumbbell for each arm is rendered, otherwise only on the arm involved in the exercise. The camera zooms in and pans on the user, showing the top-half part of the body better. A countdown starts so that the user can regain the starting position.
6. When the countdown stops, the exercise starts effectively and the user is instructed on which arm to lift and how, also bright colored arrows are rendered to help the user to understand which arm to lift.
7. The user completes the repetitions, or does as much repetitions as they can, based on the exercise mode. If the user does a valid repetition, a sound confirming the positive action plays, the score and the repetitions number are increased. If the user does a wrong repetition, a dull sound plays and the score is reduced.
8. At the end of the exercise a success finishing sound plays and the camera returns at the default location. If there are other exercises to be done, the cube reappears allowing the user to start the next exercise. If the completed exercise was the last one, the total score is shown to the user, with a button to exit the session.

All these actions can be done also by sitting users, the solution is accurate even if the user is sat and the figure 3.7 shows a gameplay screen of a sitting user.

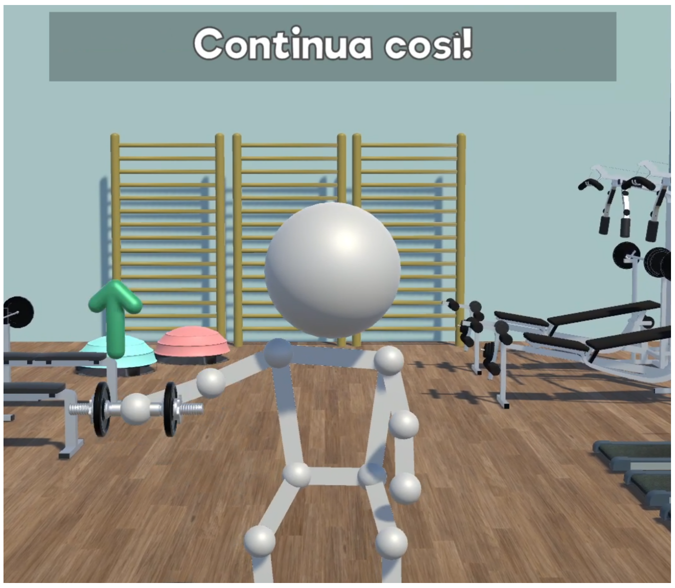


Figure 3.7: Gameplay screenshot with sat user

3.3.4 Score system and positive reinforcements

A score system was implemented to add a positive reinforcement for the user. For each correct repetition, the user is awarded with a hundred points, while for a wrong repetition the score decreases by twenty-five points. At the beginning of each exercise the score counter resets, but the score of the exercise is summed to the total session score. At the end of the session, the game shows the total score to the user, with a visual representation of the score with three stars. Stars are awarded if the total score of the session reach some thresholds:

- if the total score is greater than 30% of the maximum total score one star is awarded;
- if the total score is greater than 60% of the maximum total score two stars are awarded;
- if the total score is greater than 90% of the maximum total score three stars are awarded.

The total score possible (T) is calculated with the following formula.

$$T = \sum_{e=1}^{E} \left(\sum_{r=1}^{R_e} P \right) \tag{3.1}$$

Where:

- T is the total points.
- E is the number of exercises
- R_e is the number of repetitions for the e-th exercise
- P represents points per repetition (one hundred)

In timed exercise mode, the number of repetitions for each exercise is estimated by dividing the exercise duration in seconds by two, assuming that each complete lift cycle takes approximately two seconds. So in the case of timed exercise R_e changes as such

$$R_e = MaxTime/2$$

The figure 3.8 shows how the score appears to the user at the end of the session.

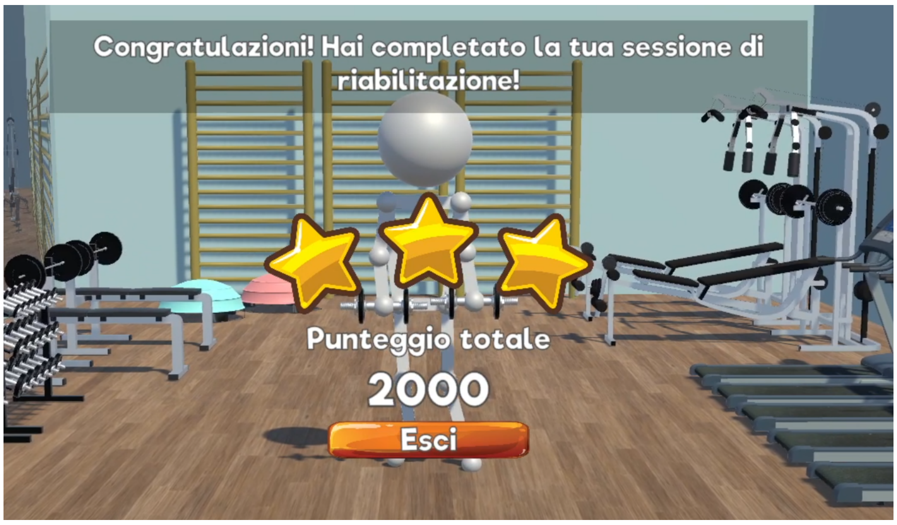

Congratulazioni! Hai completato la tua sessione di riabilitazione!
Punteggio totale
2000
Esci

Figure 3.8: Final session score UI

3.3.5 Visual and audio cues

Visual and audio cues help users to understand if they are doing the exercise correctly or not, as well as keeping track of how many repetitions are completed. The cues are divided in visual and audio cues.

Visual cues

Visual cues assist users with exercise execution and display score, time, and the number of successful repetitions. A three-dimensional arrow appears for each arm:

a blue arrow for the right arm and a green arrow for the left arm. Depending on the exercise mode, the arrows will either appear automatically or in response to controller input. For example, in alternate exercise mode, after the user completes a repetition with the right arm, the blue arrow disappears and the green arrow appears. Alternatively, the arrows will appear only when the user signals to lift the arm via keyboard inputs, using the keyboard arrows. These arrows are anchored to the respective arms, maintaining a fixed rotation along the longitudinal axis, used as animation.

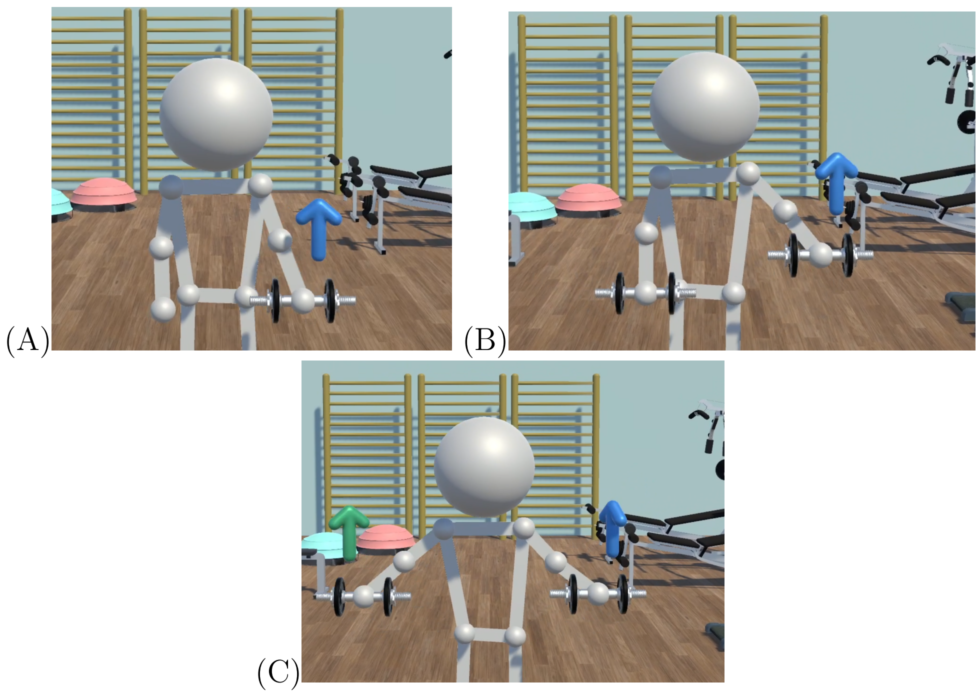


Figure 3.9: Dumbbell and arrows positioning for single arm exercise (A), alternate arms exercise (B) and simultaneous exercise (C).

Another visual cue is a rotating green cube positioned in front of the user. This cube must be touched with the interaction hand to start the exercise in automatic mode and disappears at the beginning of the exercise. Initially, the cube was placed to the side of the user, either right or left, depending on the interaction arm selected in the settings. However, this positioning can be challenging for users with reduced shoulder mobility. Therefore, the cube was re-positioned in front of the user, making it reachable by bending the elbow without requiring a shoulder movement.

A yellow small sphere is anchored to the hand on the interaction hand, serving as a selector, so that the user knows which arm to use to start the exercise, as shown in figure 3.10

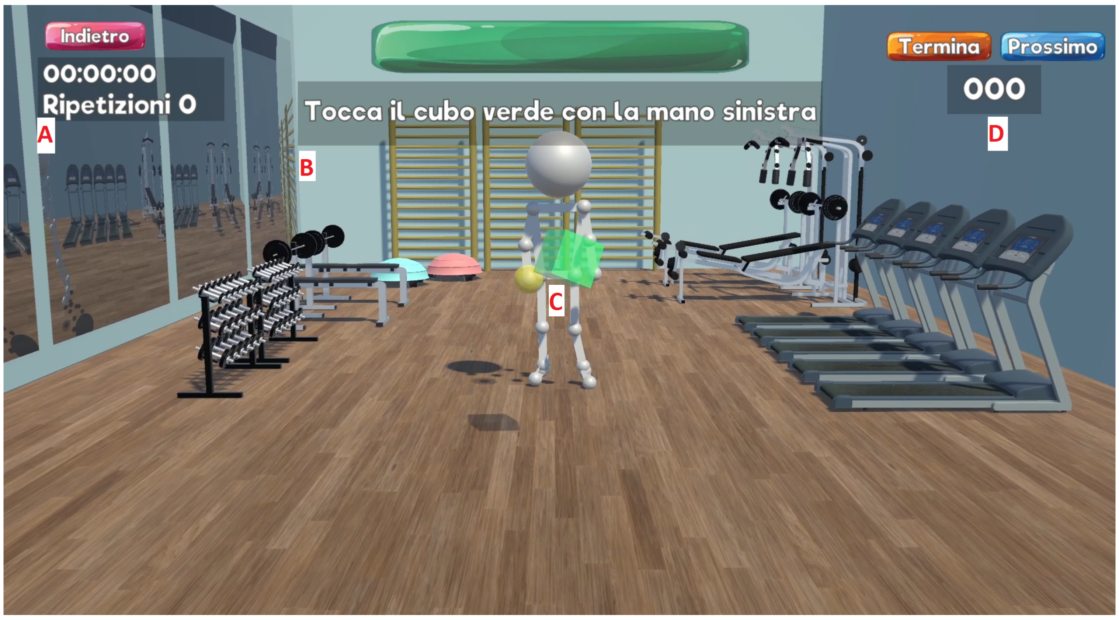


Figure 3.10: Starting UI for the exercise session. Repetitions and timer text (A), instruction text (B), start-exercise cube and selector sphere (C), partial score text (D)

Audio cues

Each time the user does a correct repetition, reaching the required angle, a success sound is triggered. Otherwise, if the user does not correctly lift their arm, for example with a wrong angle, a dull error sound is triggered. A success sound plays at the end of each exercise. All these sounds were gathered from a public domain CC0 sounds library, called freesound.org [67].

3.4 Server and database development

3.4.1 Server

The server utilizes Node.js as its runtime environment and is written in TypeScript, employing an MVC-like structure to ensure scalability and maintainability [43]. Initially, the server code was written in plain JavaScript but was soon adapted to TypeScript due to the latter's versatility and robust typing system [68].

The primary function of the server is to collect, manipulate, and store data received from clients, the Python script, the game, or the web dashboard, using WebSocket and REST API protocols. The server hosts the WebSocket server on one port and the HTTP server on another, with both ports configurable in the `environment.ts` file. This allows future developers to modify the ports without altering the source code. The main functionalities of the server are detailed in table 3.2.

Functionality	**Description**
WebSocket server	Open a WebSocket server on port 8080 (can be customized); the WebSocket connection is opened at the start of the solution and remains listening until the solution is closed.
HTTP server	Open a HTTP server on port 3080 (can be customized); as for the WebSocket connection, the HTTP server remains listening until the code execution is stopped. It exposes a list of APIs as detailed in figure 2.2.
Handle user's authorization and creation	The HTTP server exposes APIs to register new users and allow the user authorization via a login operation.
Data persistency	Using mongoose package the server saves the collected data in MongoDB, with pre-defined schemas and a defined DB structure detailed in section 3.4.1 and 3.4.2.
Expose collected data in organized JSON structures or files	Data collected in the database is exposed via REST APIs in structured JSON format for use by the web dashboard UI. It is also available in organized zip files for download by dashboard users.

Table 3.2: Node.js Server main functionalities

express.js

express.js package is used as web framework to develop the server solution. Used widely for these purposes, its ease of use, the HTTP utility methods and the flexible routing framework allow to build reliable and robust APIs [69]. All the Rest calls made to the server are processed by a router and, given the URL and HTTP method used by the client, redirected to one of the server services. Here is an example for the start exercise operation, used in the Unity code to signal the start of a specific exercise, identified by `exerciseIndex`. The `body` also contains information about the patient, identified by `loggedUserId`, the exercise plane of movement (if frontal or lateral lifts) and the current `sessionId`.

```
    this._router.put('/api/exercises/start', async function(req, res)
    {
            try {
                let exerciseDto = new Exercise();
                exerciseDto.patientId = req.body.loggedUserId;
                exerciseDto.sessionId = req.body.sessionId;
                exerciseDto.exerciseIndex = req.body.exerciseIndex;
                exerciseDto.exercisePlane = req.body.exercisePlane;
                res.status(200).json(await exerciseService.
   startExercise(exerciseDto));
            } catch (error) {
                console.log(error);
                res.sendStatus(500);
            }
        })
```

The server router parse the body in an `Exercise` object and uses it to call the asynchronous `startExercise` method in the `exerciseService` class. Once the operation returns a result, if no errors where detected, the router returns the call with an HTTP status 200 [70] and the JSON object of the created exercise. In this case the object will be used by Unity to retrieve the `exerciseId` generated by MongoDB.

mongoose

The mongoose package [71] is used to connect to the MongoDB database, define document schemas and perform database operations. Code snippet of the database connection:

```
    this.mongoose.connect(uri, { useNewUrlParser: true });

    const db = this.mongoose.connection;
    db.on('error', () => console.log("Connection Error"));
    db.once('open', () => {
        console.log(`[INIT] DB connection started on port ${db.port}
   ${db.name}`);
    })
```

Example of the exercise schema created to save the exercises in the corresponding MongoDB collection:

```
    const exerciseSchema = new mongoose.Schema({
        id: Number,
        exerciseName: String,
        sessionId: String,
        startTimestamp: Number,
        endTimestamp: Number,
        successfulReps: Number,
        wrongReps: Number,
        patientId: String,
        commandTimestamps: [Number],
    });
```

Mongoose schema will allow to save, get, update or delete documents in the corresponding collections of the database, using utility methods provided. Following the current use case, the create exercise operation is:

```
    Exercise = require('../models/dao/exercises');

    public async createExercise(sessionId: string, loggedUserId:
   string, exerciseIndex: number, exercisePlane: number, affectedSide
   : AffectedSides) {
       let exerciseDB = new this.Exercise();

       exerciseDB.startTimestamp = Date.now() * 1000;
       exerciseDB.patientId = loggedUserId;
       exerciseDB.exerciseName = ExerciseTypes[exerciseIndex];
       exercisePlane == ExercisePlanes.Lateral? exerciseDB.
   exerciseName = exerciseDB.exerciseName + ' laterale' : exerciseDB.
   exerciseName = exerciseDB.exerciseName + ' frontale';
       exerciseDB.sessionId = sessionId;

       return await exerciseDB.save();
    }
```

Where the async method `createExercise()` returns the created document as JSON while `Exercise` is the schema detailed above.

3.4.2 Database

As already mentioned in section 2.2 the DBMS of choice is MongoDB for the excellent performances in real-time data processing [48][49]. The solution's database is composed by five collections.

1. **Exercises**: stores the exercise data, linking the user and session, saving `userId` and `sessionId`.
2. **LandmarkPositions**: stores all MediaPipe pose positions data, linking them to the session by saving the `sessionId`. This collection is filtered during the

report download process by evaluating the timestamps and comparing them to the timestamps at the beginning and end of the exercise.

3. **Performances**: stores all the performance data from both the Python code and the Unity game engine, linking it to the session by saving the `sessionId`.

4. **Sessions**: stores all the session information, a session starts when the MediaPipe Python solution starts the video capture and ends when the solution is closed and the video capture stops.

5. **Users**: stores all the information about users.

The structure, using the Mongoose package, provides an easily scalable solution. When new schemas are created on the server, the corresponding collections are automatically generated in the MongoDB database. Additionally, as a NoSQL database, MongoDB allows future modifications to existing schemas without requiring any database operations; new documents can be saved without affecting the existing ones.

MongoDB auto-generate a 12-byte `ObjectId`, when a document is saved in a collection [72]. The hexadecimal string values of these ids are then used by the application to retrieve, modify or delete specific data.

3.5 UI Dashboard development

The server also provides all the files for a minimal web page. The web dashboard is developed using HTML5 [73] and Javascript, using jQuery [50] and Bootstrap [51].

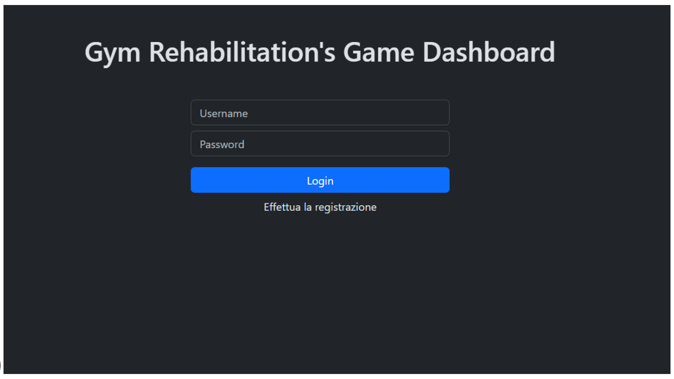
Login and registration pages

The first page rendered is the login, from which the user can log into the system or register a new user. To register a new user, it's mandatory to add the information detailed in table 3.3. The login and registration forms are showed in figure 3.11.

Field	**Values**
Username	Username of the new user. If a user with the same username already exists, the web page responds with an error message upon form submission.
Password	Password of choice for the user
Name	User's name
Surname	User's surname

Role	A selection of available roles, with four possibilities: patient, guest, developer or healthcare professional.
Age	User's age
Pathology	User's diagnosed pathology (if any)
Affected side	A selection of affected side, can be right, left or both
Affected for	Specifies for how many years the patient has been affected by the diagnosed pathology

Table 3.3: User registration input fields

(A)

Gym Rehabilitation's Game Dashboard
Username
Password
Login
Effettua la registrazione

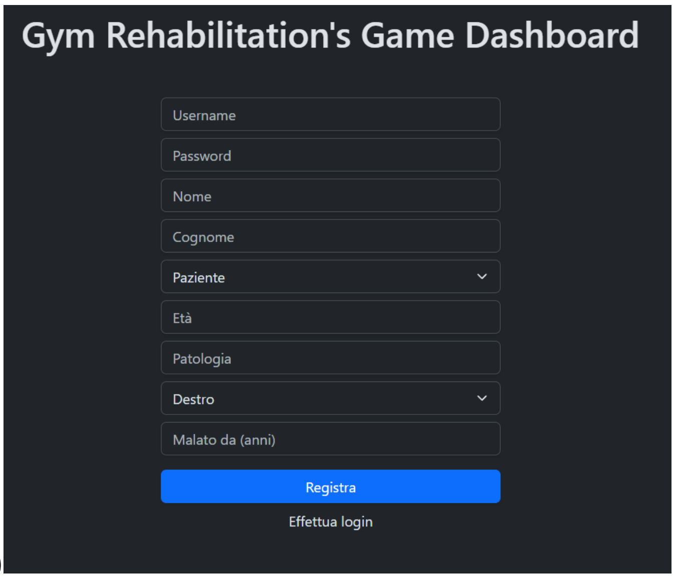
(B)


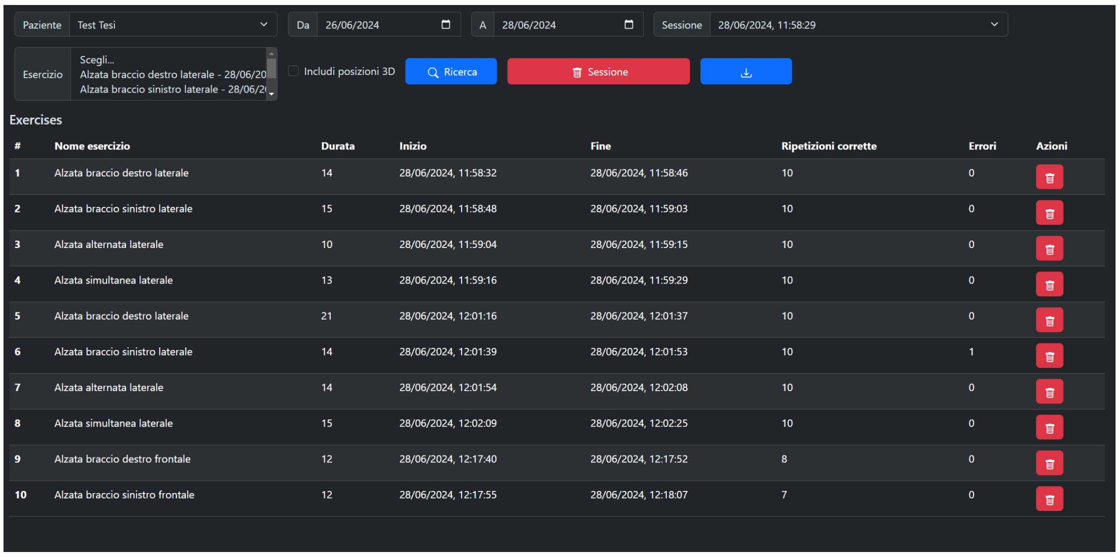
Figure 3.11: Login (A) and registration (B) forms for the web dashboard page

Dashboard page

The main web page is composed by a left vertical menu, which allows to visualize the homepage or the list of users registered. The homepage and default page is composed by a form that can be completed to filter the results of the research and subsequently the data to be downloaded. The table 3.4 details each applicable filter and the figure 3.12 shows the page with applied filter and results table.

Field	**Values**	**Mandatory**
Patient (Paziente)	All the patients present in the database, filtered by the server.	Yes
From (Da)	Select a date from which filter the results.	No
To (A)	Select a date to which filter the results.	No
Session (Sessione)	Select a specific session. Sessions are filtered by the selected user, so the options change based on the `userId`.	No
Exercise	List of exercises of the specific selected session. Multiple selection is possible.	No
Include 3D Positions (Includi posizioni 3D)	Choose to include 3D positions in the output of the search.	No

Table 3.4: Web search filters



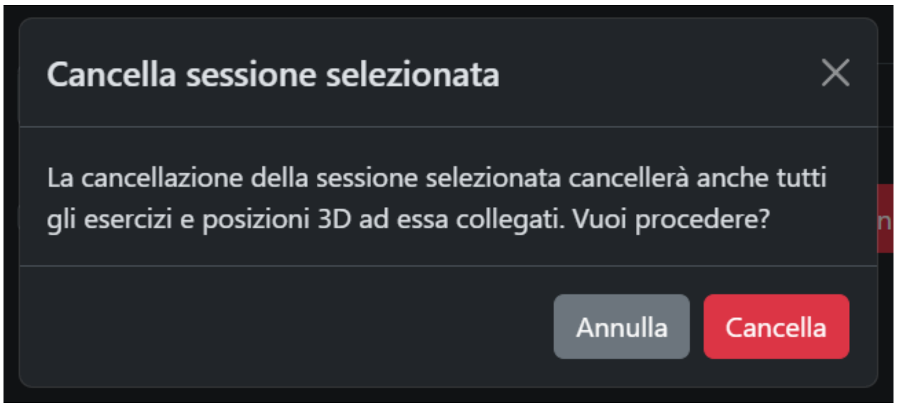
Paziente	Test Tesi	Da	26/06/2024	A	28/06/2024	Sessione	28/06/2024, 11:58:29
Esercizio	Scegli... / Alzata braccio destro laterale - 28/06/20 / Alzata braccio sinistro laterale - 28/06/2						

Exercises

#	Nome esercizio	Durata	Inizio	Fine	Ripetizioni corrette	Errori	Azioni
1	Alzata braccio destro laterale	14	28/06/2024, 11:58:32	28/06/2024, 11:58:46	10	0	🗑
2	Alzata braccio sinistro laterale	15	28/06/2024, 11:58:48	28/06/2024, 11:59:03	10	0	🗑
3	Alzata alternata laterale	10	28/06/2024, 11:59:04	28/06/2024, 11:59:15	10	0	🗑
4	Alzata simultanea laterale	13	28/06/2024, 11:59:16	28/06/2024, 11:59:29	10	0	🗑
5	Alzata braccio destro laterale	21	28/06/2024, 12:01:16	28/06/2024, 12:01:37	10	0	🗑
6	Alzata braccio sinistro laterale	14	28/06/2024, 12:01:39	28/06/2024, 12:01:53	10	1	🗑
7	Alzata alternata laterale	14	28/06/2024, 12:01:54	28/06/2024, 12:02:08	10	0	🗑
8	Alzata simultanea laterale	15	28/06/2024, 12:02:09	28/06/2024, 12:02:25	10	0	🗑
9	Alzata braccio destro frontale	12	28/06/2024, 12:17:40	28/06/2024, 12:17:52	8	0	🗑
10	Alzata braccio sinistro frontale	12	28/06/2024, 12:17:55	28/06/2024, 12:18:07	7	0	🗑

Figure 3.12: Default web page with results from exercises research

If the user selects a session, the session can be deleted. Deleting the session will also delete all associated exercises, performances and positions recorded. For this reason, being a non-idempotent operation, the UI asks the user to confirm the operation via a modal, shown in figure 3.13.


Cancella sessione selezionata

La cancellazione della sessione selezionata cancellerà anche tutti gli esercizi e posizioni 3D ad essa collegati. Vuoi procedere?

Figure 3.13: Confirm modal to delete a session

If the user clicks on search ("Ricerca") the homepage will render a table with all the exercises that match the filters, provided by the server, as shown in figure 3.12. If the user clicks on the download icon, a .zip file is downloaded with all the exercises data of the selected session, divided in structured JSON files. These files can then be used to analyse patient's data, containing all the information about the patient's positioning during the exercise, as explained in section 4.1.1.

3.6 Executable bundle development

A batch bundle was created to launch the application from a single batch executable file. It consists of three batch file and one executable file.

1. `launch.bat` : the main batch file that starts the solution. It starts the server batch, the Python batch and the Unity executable. It also intercept the game solution process closing, to kill also the other open processes.
2. `launchNodeJs.bat` : launches the Node.js server, ensuring that all the npm packages are installed with the `npm install` command.
3. `launchPython.bat` : launches the Python script bundle, creating a pyenv [74] environment to use the correct Python version to adapt to computers with multiple Python versions installed.
4. `GymRehabilitation.exe` : it executes the Unity game itself; once its killed, all the other processes are killed as well.

5. `launchDashboard.bat` : it launches only the server and opens the web dashboard in the default browser.

Chapter 4

Results

4.1 Types of data collected

Collected data helped the team to understand better the solution potential, its limitations and its improvement areas. Data was gathered during development in different areas: one room with poor and non-uniform lighting and, during a weekly meeting done at Politecnico di Torino, in a laboratory with ideal lighting conditions and setting; the space was free of obstacles with direct neon lights above the user, so that all the parts of the body were uniformly illuminated. All data were saved and collected in the database, associated with the user session by the `sessionId`.

The exported data, as described in section 4.1.1, consist of a list of JSON files containing all relevant information about the exercise, detailed in table 4.1. This JSON output is analysed in MATLAB [75] using a script that calculates the trajectories of the right and left arms and the average FPS rate. The average FPS rate is determined using the timestamps for the start and end of the exercise, along with the number of frames collected.

Data Type	**Purpose**
Capture FPS	The effective capture FPS of the camera during the pose estimation. Saved to gather what is the difference between camera FPS and pose estimation FPS. Saved once every second only if `DEBUG` variable is `True` (see section 3.2).
Theoretical FPS	Calculated immediately after the pose estimation by MediaPipe, it indicates what could be the ideal performance, indicated by maximum FPS of the solution. Saved once every second only if `DEBUG` variable is `True` (see section 3.2).

Real FPS	The effective FPS of the Python code after writing to the pipe and sending the data to the server. Saved once every second only if `DEBUG` variable is `True` (see section 3.2).
Landmark positions	As shown in section 3.2, all position data with timestamps were saved to be analysed afterwards.
Exercise start and stop	Each time a user starts or stops an exercise, the start or stop action timestamp is saved in database, linked to the exercise.
Command action	Each time a user receives a command to lift an arm, either in automatic or controlled mode, the command timestamp is save in database, to analyse the reaction time of the patient, comparing it to the timestamp in which the arm reaches the required angle.
Unity FPS	FPS of the rendered game, to observe an eventual drop in performances.
User's repetitions	Both successful and wrong repetitions completed by a user are saved in the database.

Table 4.1: Collected data overview

In the initial iterations of server and game development, MATLAB analysis helped identify and address performance issues. For example, while the capture FPS was expected to run at an average of 30 FPS, the actual FPS dropped to 22 FPS or lower, indicating a loss of eight or more FPS.

Improvements to the Python script, including the implementation of a multi-thread structure (refer to 3.2), enhanced the average FPS to 25. Further code optimizations in the Node.js server and bug fixes improved performance to an average of 29 FPS. These enhancements significantly reduced data loss, allowing for more accurate analysis and output.

4.1.1 Data structure

In this section it will presented the structure of data extracted and successfully analysed with MATLAB.

```
  {
    "exercise": {
      "id": "667e8958083fa5e072f5d955",
    "exerciseName": "Alzata braccio sinistro laterale",
    "startTimestamp": 1719568728364000,
    "endTimestamp": 1719568743728000,
    "commandTimestamps": [
      1719568730793000, 1719568731996000, ...
    ],
    "patientId": "667e8921083fa5e072f5d4bb",
    "sessionId": "e8bbc5b6-5027-4db1-b1ba-ab9dcd8949a2",
    "successfulReps": 10,
    "wrongReps": 0
  },
  "landmarks": [
        {
          "_id": "667e8959083fa5e072f5d961",
          "landmarks": [
            {
              "x": "0.011",
              "y": "-0.636",
              "z": "-0.361",
              "visibility": "1.000",
              "landmark": 0
            },
            {
              "x": "0.018",
              "y": "-0.674",
              "z": "-0.344",
              "visibility": "1.000",
              "landmark": 1
            },
            ...
          ],
          "timestamp": 1719568728395221,
          "sessionId": "e8bbc5b6-5027-4db1-b1ba-ab9dcd8949a2",
          "frameId": 579,
          "__v": 0
        }, ...
    ]
}
```

The extracted data is combined with all the information about the exercise, as detailed in section 3.4, and all the landmark positions recorded during the exercise. The server uses the `startTimestamp` and `endTimestamp` to retrieve only the landmarks positions needed.

4.2 Data analysis

The analysis is conducted with MATLAB, using a script that traces the movements of right and left arm, evaluating the angle between the arm and the corresponding side of the user. This analysis conducted during the development highlighted some limitations that can be mitigated following some rules and constraints:

1. the user should not wear big clothes that conceal body joints or limbs definition;
2. the lighting should be as diffused as possible, not leaving a side of the body obscured compared to the other;
3. the head of the user should always be visible in the frame. This is mandatory, otherwise the pose detection will fail, as explained in section 2.2.2;
4. the user should not wear bracelets or clocks of sort on their wrists;
5. the background should be as uniform as possible, with a good contrast with the user's figure.

If these rules are not followed, the pose detection can suffer lag or wrong estimation, especially if two joints overlap, as shown in figure 4.1, where the user was wearing a dark bracelet. This wrong estimations are also saved in the database, it's clearly visible a difference also in the data analysed with MATLAB, where there is a more jagged curve of movement (not representative of real movements) in sub-optimal conditions, and a much smoother curve in optimal or good conditions, as shown in figure 4.7.

To evaluate the performances of the finished prototype (version `0.10.3`) data was collected for each of the exercise type (single for both arms, alternate arms and simultaneous arms). Exercise type collected are described by the table 4.2. The data was collected in both the most suitable setting with optimal lighting and in the setting with sub-optimal condition.

Plane	**Mode**	**Exercise goal**
Frontal lifts	Automatic	Repetition number
Lateral lifts	Automatic	Repetition number
Frontal lifts	Controlled	Repetition number

Lateral lifts	Controlled	Repetition number
Frontal lifts	Automatic	Max repetitions in fixed time
Lateral lifts	Automatic	Max repetitions in fixed time
Frontal lifts	Controlled	Max repetitions in fixed time
Lateral lifts	Controlled	Max repetitions in fixed time

Table 4.2: Collected exercise data in different conditions

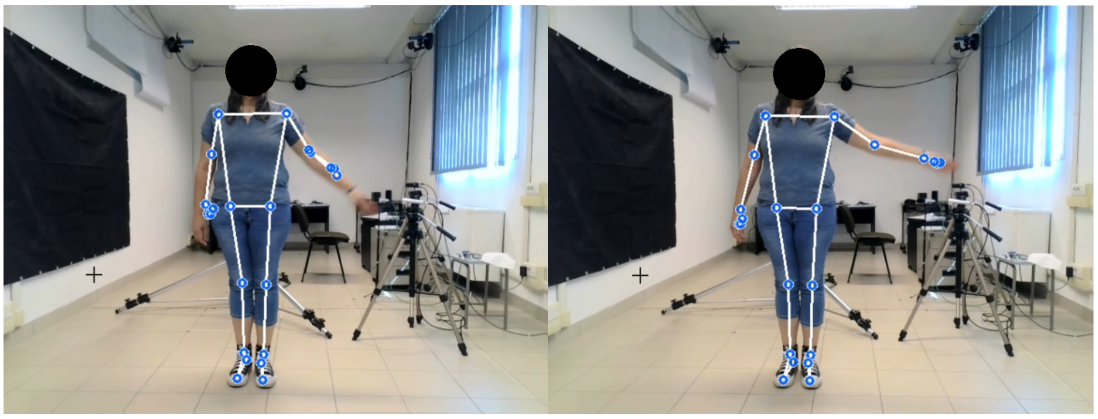
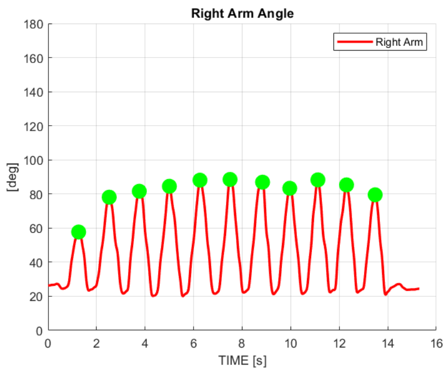


Figure 4.1: Examples of wrong and good pose estimation with good lighting, but complex background (wrong positions for wrist and hand joints)

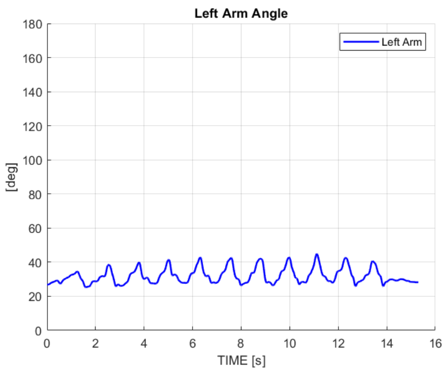
This produced a conspicuous number of results that were analysed:

- a total of ten sessions from the last version were gathered, from three individuals, two males and one female;
- different sessions were gathered from old and non-optimized versions of the solution during development, for the purpose of this work three of this sessions will be analysed;
- fifty-six different exercises in total were analysed in MATLAB;

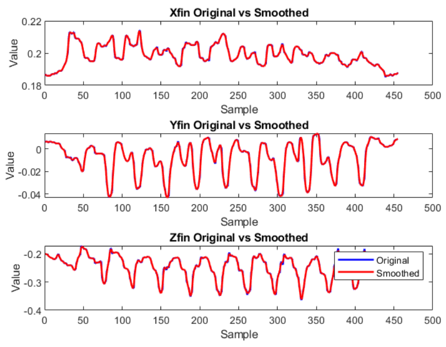
Data analysis in this section will be divided in areas, highlighting the differences between settings, old and new versions of the solution, or between a good set with ideal lighting and a set with sub optimal lighting.



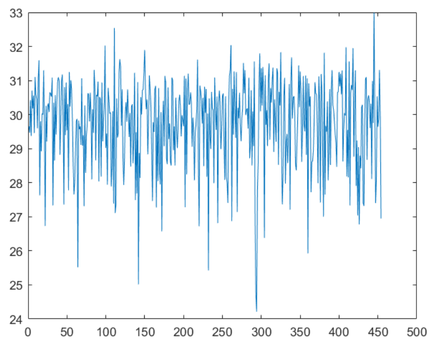
(A)



(B)



(C)



(D)

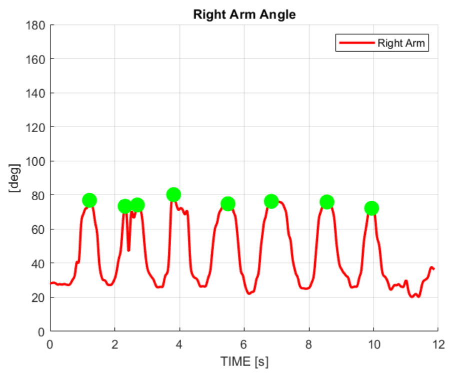
Figure 4.2: All the visual representations produced by the MATLAB analysis for a single arm exercise, with right arm angle (A), left arm angle (B), original vs smoothed arm joints positions (C) and frames per second (D)

Model complexity: performance analysis

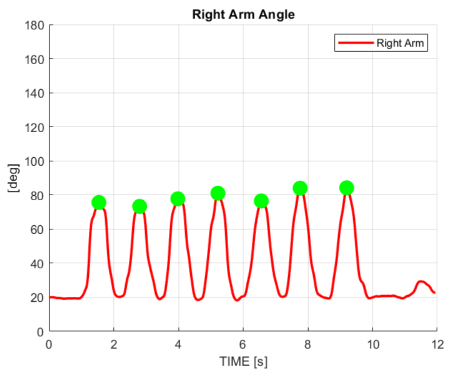
As detailed in Section 3.2, the `MODEL_COMPLEXITY` variable can be adjusted to select between the lite, full, or heavy detection models provided by MediaPipe [20]. Modifying this variable directly affects the user experience: the heavy model, while more accurate, can cause significant rendering lag, whereas the lite model offers better performance but with reduced accuracy. The figures 4.3 and 4.4 show comparisons between the different trajectories analysed with MATLAB and the FPS performances for all the three different model complexities.

To obtain these results, the same exercise was repeated with varying model complexity settings. The exercise involved a single arm lift, performed at a steady pace, with a maximum duration of ten seconds. The user executed the lifts without any errors. After completing the arm lifts, an additional two-second interval was included to allow the user to return their arm to the initial position before stopping the exercise. Consequently, each exercise session lasted a total of twelve seconds.

(A)



(B)



(C)

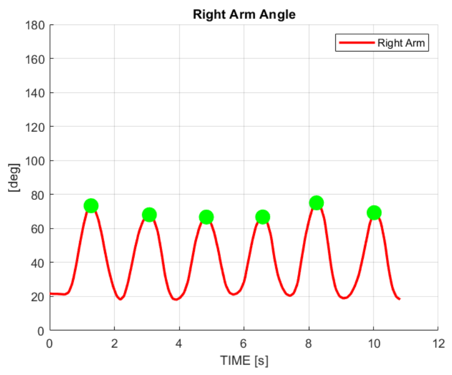


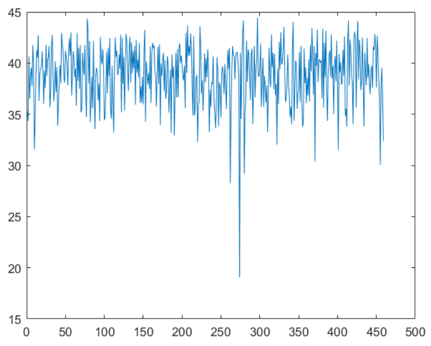
Figure 4.3: Comparing data positions quality with model complexity 0 - Lite (A) 1 - Full (B), and 2 - Heavy (C)

During this interval, the pose detection and estimation produced the performances, shown in figure 4.4:

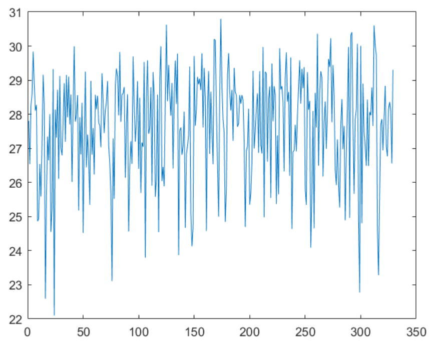
- 460 frames for the lite model, with an average framerate of 38.83, the solution registered seven correct repetitions and two errors and produced an irregular and jagged trajectory curve;
- 330 frames for the full model, with an average framerate of 27.64, the solution registered seven correct repetitions and no errors and produced a much more refined trajectory curve compared to the lite model;
- 94 frames for the heavy model, with an average framerate of 8.60, the solution registered five correct repetitions and no errors and produced the smoothest curve, compared to the lite and full models;

Therefore, while the heavy model produces the smoothest curve, as shown in Figure 4.3, its performance in terms of average framerate is insufficient. Conversely, the lite model offers excellent performance in terms of average FPS but lacks the

(A)



(B)



(C)

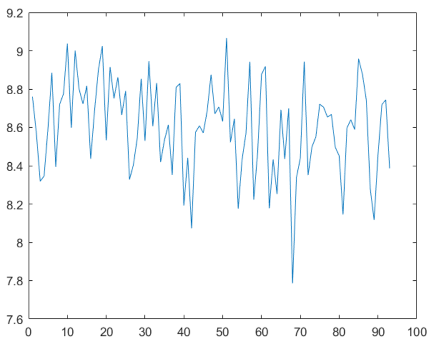


Figure 4.4: Comparing FPS performances with model complexity 0 - Lite (A) 1 - Full (B), and 2 - Heavy (C)

necessary precision for effective use. The full model is the best compromise in terms of performance and accuracy, so it was selected as the best option to set the default value of the model complexity. All data collected in this section are therefore collected using the full model.

Optimization of the average framerate

As detailed in section 4.1 the average framerate was calculated to analyse the solution performances. The FPS were calculated with:

```
    TIME=table2array(cell2table(timestamp));
    DELTA_TIME = (TIME(:)-TIME(1))*10^-6;
    framerate=1./(diff(TIME*10^-6));
    framerate_m=mean(framerate);
```

Where the variables represent:

- `TIME`: Contains the timestamps converted from microseconds to seconds.

- `DELTA_TIME`: Contains the time differences between each timestamp and the first timestamp, in seconds.
- `framerate`: Contains the instantaneous frame rates calculated as the reciprocal of the time differences.
- `framerate_m`: Contains the average (mean) frame rate computed from framerate.

The calculated results were also plotted to provide a visual representation of the FPS during the exercise. As shown in Figure 4.5, the initial average framerate was quite low, with an average FPS of 19.27 in version `0.1.0` of the solution. Setting `DEBUG` to `False`, that showed the rendering by MediaPipe pose detection and print console logs, as described in section 3.2, subsequent improvements were made, and in version `0.4.0`, there were significant enhancements, yielding a 24/25 FPS on average. However, this version still experienced performance drops due to a bug in the data persistence layer and the single-thread Python solution. In the latest version, `0.10.3`, bug fixes, code improvements, and the addition of a separate thread in the Python solution resulted in an average FPS of 28.93, calculated from more than 50 entries. The best result achieved was 29.72 FPS, though other results were also commendable, with the lowest FPS recorded in the latest version being 27.64.

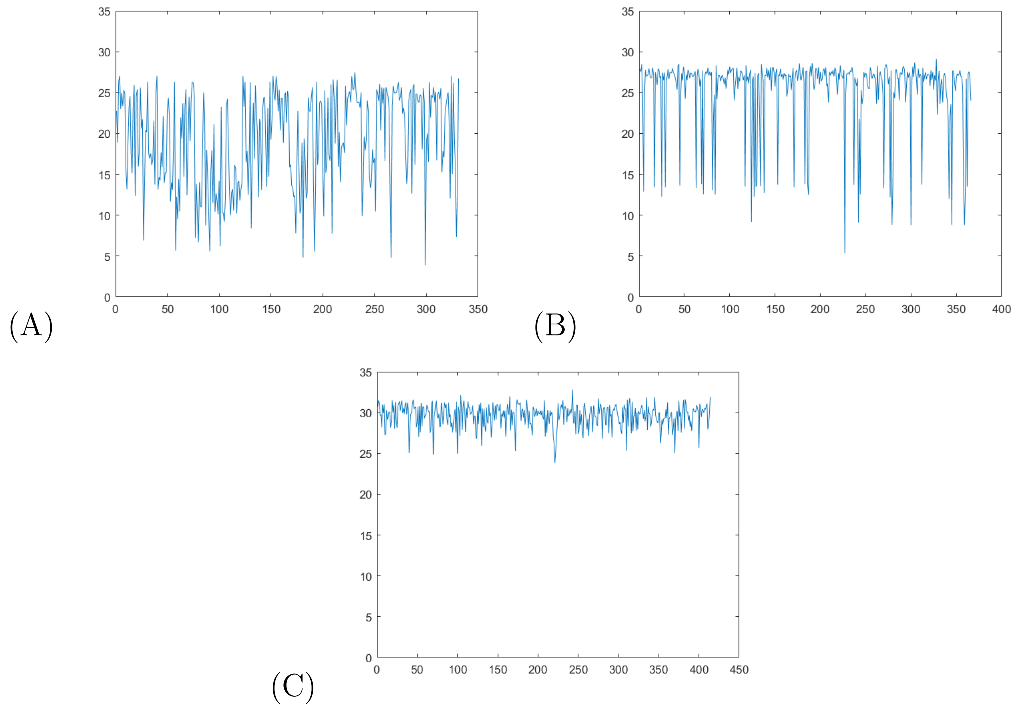


Figure 4.5: Average framerate evolution from version 0.1.0 (A, framerate avg 19.27), to version 0.4.0 (B framerate avg 25.31) and final version 0.10.3 (C framerate avg 29.61)

Effects of illumination on arms trajectories

The MATLAB script provides also the visual representation of the trajectories of the arms, plotting the angle between each arm and the corresponding user side. As mentioned in the introduction of this chapter, the trajectories can suffer from a suboptimal setting for the exercises and can indicate lag or incorrect pose estimations by the MediaPipe pose estimation solution. For example, if the user is illuminated only from an angle, leaving one side darker than the other, the trajectories of the arm will be imperfect, as shown in figure 4.6, which compares the trajectories of right ad left arm in simultaneous arm lift exercise.

The data recorded can also be of sub-optimal quality if the camera is against the light, as shown in figure 4.7, were the same exercise (simultaneous lateral arms lift) was performed first with optimal conditions and then with a strong backlight.

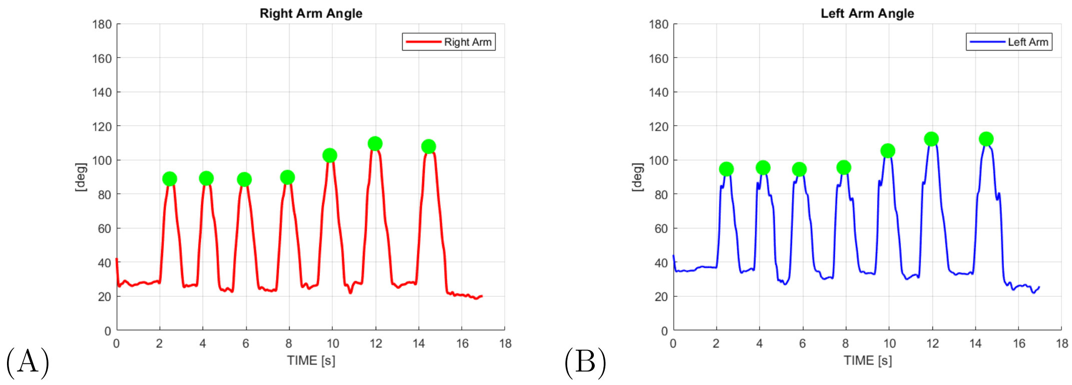


Figure 4.6: Comparing right side (A) and left side (B) in a exercise where the left side is poorly illuminated

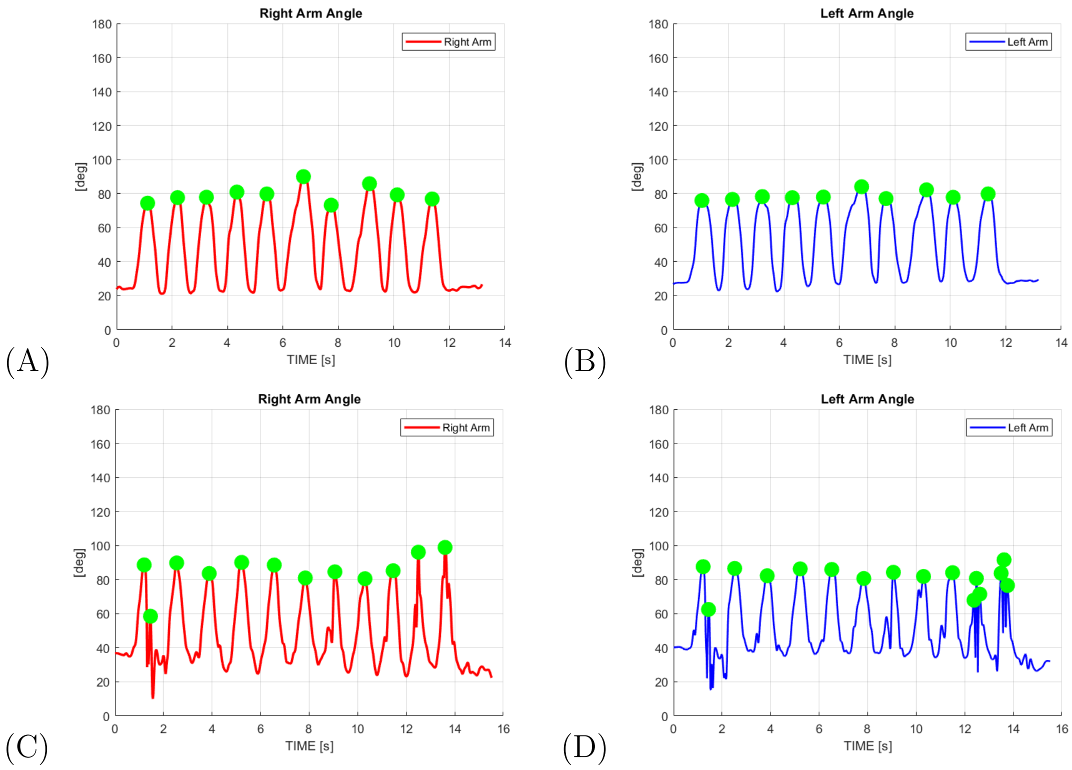


Figure 4.7: Comparing data positions quality with good lighting (A) and (B), and bad lighting (C) and (D)

4.3 Guidelines for effective data collection and use

Meaningful data collection hinges on users adhering to specific guidelines. The analysis has demonstrated that factors such as lighting conditions, background settings, user behavior, and clothing choices significantly impact data recording accuracy. When these variables are not controlled or standardized, they can introduce variability and diminish the quality and reliability of the collected data. Therefore, following established guidelines, described at the beginning of the section 4.2, ensures more consistent and valuable data, facilitating accurate analysis and informed decision-making.

Data collection serves not only to analyse the performance of the solution but, more importantly, to track the progress of patients over a defined period. Specifically, the collected data can be utilized to:

1. evaluate the progression of a patient, comparing the numbers of correct and wrong repetitions;
2. compare the effectiveness of tele-rehabilitation vs rehabilitation, in terms of successful repetitions, frequency of completed sessions, drop out rates and so on;
3. analyse the response time of the patient, comparing the `commandTimestamp` value with the timestamp of maximum lift by the patient.

Chapter 5

Future developments

5.1 General solution improvements

Self contained solution

To ensure the exergame and accompanying Python code can function independently of server connectivity, a self-contained solution should be developed. This standalone version would eliminate the dependency on a reachable server and remove the need for user login. While data collection would not occur in this scenario, the self-contained solution remains highly beneficial for rehabilitation purposes. Developing a self-contained solution aligns with the goal of making the exergame accessible and effective for rehabilitation purposes, even in environments where server connectivity is not guaranteed.

Versioning

To enhance the organization and maintainability of the exergame solution, it can be divided into different repositories. This separation allows for better management of the codebase, with each component being developed, tested, and maintained independently. By creating a specific repository for each solution node, including the Python code, Unity game, server, and web dashboard, the architecture can be more scalable and maintainable.

Security improvements

The current prototype was developed and maintained in local environments. However, if the solution is released and the server and database are centralized, a security layer needs to be added to protect sensitive information and ensure data encryption. Several measures can be adopted:

1. use JSON Web Tokens (JWT [76]) to create a temporary secure connection between the clients and the server;
2. release the solution over HTTPS by creating a SSL/TLS certificate [77];
3. use an external identity and access management solution to handle sensitive user information and data, such as passwords, names, diagnoses and so on;
4. implement regular data backups and establish a robust data recovery plan;
5. ensure that data stored in the database is encrypted. This helps protect sensitive information in case of unauthorized access to the database files;
6. set up comprehensive logging and monitoring to detect and respond to suspicious activities.

5.2 Game mechanics

New game mechanics can be added to the exergame, improving the user experience and rehabilitation efficiency. As proven by recent studies, group interaction can be an important factor in patients with Parkinson's disease [8], so it could be a good addition to evaluate a solution where more people can be tracked and rendered in the game.

The user's movements can be rendered using a three-dimensional avatar instead of a plain humanoid figure, providing a more immersive and engaging experience. The joint positions can be used to animate the skeleton of a three-dimensional character accurately, making the movements appear more realistic and lifelike. This approach could significantly enhance player engagement, particularly if players have the option to customize and select their avatars in advance. By allowing players to choose avatars, the system can create a more personalized and enjoyable experience, ultimately encouraging continued participation and improving the effectiveness of the exercise program.

To further enhance the versatility and efficiency of the exergame, the solution can be adapted to include exercises targeting the lower part of the body, so that the exergame can offer a more complete rehabilitation, catering to a broader range of physical therapy needs.

5.3 User Interface

Three-dimensional movement representation

The web page can be improved by adding a visual representation of the patient movements. A solution based on libraries such as three.js [78] or aframe.js [79] can be used to visualize the movements of the users in a three-dimensional environment on a web page, using WebGL API [80]. All the dimensions saved in the database are real-world dimensions measured in meters. The mentioned libraries also use three-dimensional positions represented in meters. Therefore, it should be possible to directly render the joints, considering that the positions saved from MediaPipe use the user's hip center as their origin.

Exercises and users related actions

To enhance the functionality of the system, CRUD (Create, Read, Update, Delete) operations can be implemented for both the exercise table and the patients table. This would provide users with the capability to perform comprehensive data management tasks.

By incorporating these CRUD operations, the system becomes more flexible and user-friendly, allowing administrators and authorized personnel to effectively manage the data related to exercises and patients.

Download a single exercise report

It should be given the possibility to download the reports of single exercises instead of a whole session. It should also be reviewed the exercise output file to check if there are other information that can be useful to the data analysis.

5.4 Server and Database

The server and database architecture can be improved to incorporate data analysis capabilities within the server layer itself. This would allow the generation of plot images depicting the movements of the patients, eliminating the necessity for script execution in MATLAB. By processing and visualizing data directly on the server, it ensures a more streamlined and efficient workflow.

Additionally, a functionality could be introduced to facilitate the download of all exercise information in a CSV or Excel file format. This feature would provide users with an easy way to access and analyse their data offline or integrate it with other applications for further analysis.

Furthermore, the system could be integrated into an existing solution, removing the requirement for users to create and register new accounts. This integration would enhance user convenience by leveraging existing authentication and user management systems, providing a seamless user experience.

Chapter 6

Conclusions

The development and implementation of an exergame solution, leveraging the MediaPipe framework, aimed at the rehabilitation of neuro impaired patients, has been comprehensively documented and analysed. This conclusion draws together the key findings and outcomes.

The data collected during exergame sessions were analysed using MATLAB, focusing on the trajectories of the right and left arms and the average FPS rate. Initial tests revealed performance bottlenecks, such as lower-than-expected FPS rates. However, through iterative improvements, performance was significantly enhanced, reducing data loss and providing more accurate and reliable analysis.

In comparing these findings to the initial goals, it is clear that the project has addressed the core requirements set out at the beginning. The solution effectively integrates real-time pose estimation, data collection, and analysis, within a scalable and maintainable architecture. The iterative development process, coupled with testing and evaluation, has resulted in a robust prototype that meets the needs of use.

The initial goal of creating an innovative rehabilitation tool has been achieved, with the potential for further enhancements and developments to expand its capabilities. The use of the MediaPipe framework for pose estimation has proven effective, offering a solid foundation for tracking and analyzing movements. However, it's important to note that while MediaPipe is robust and reliable under ideal conditions, its accuracy can diminish in less optimal environments. Factors such as poor lighting, occlusions, or unconventional body postures can affect the precision of pose estimation. Therefore, user training is essential to ensure that users are aware of the conditions required for accurate data capture. The data collected and analysed will provide valuable insights into the patient's progress and the effectiveness of the rehabilitation exercises.

Looking ahead, the proposed future developments offer a clear roadmap for enhancing the solution. By creating a self-contained version, introducing new game mechanics, and improving the user interface and server, the exergame can become more accessible and engaging for users. Upgrading the server and database capabilities will further streamline the data analysis process, providing real-time feedback and enhancing the overall user experience.

In conclusion, the project has successfully met its initial objectives, providing a solid foundation for future research and development. The continued evolution of this solution, driven by the proposed future developments, promises to deliver more effective and personalized rehabilitation programs, ultimately improving the quality of life for patients.

Bibliography

[1] Augusto Garcia-Agundez, Ann-Kristin Folkerts, Robert Konrad, Polona Caserman, Thomas Tregel, Mareike Goosses, Stefan Göbel, and Elke Kalbe. «Recent advances in rehabilitation for Parkinson's Disease with Exergames: A Systematic Review». In: *Journal of NeuroEngineering and Rehabilitation* 16 (2019). DOI: https://doi.org/10.1186/s12984-019-0492-1 (cit. on p. 1).

[2] Logan Wade, Laurie Needham, Polly McGuigan, and James Bilzon. «Applications and limitations of current markerless motion capture methods for clinical gait biomechanics». In: *PeerJ* 10 (Feb. 2022), e12995. ISSN: 2167-8359. DOI: 10.7717/peerj.12995. URL: https://doi.org/10.7717/peerj.12995 (cit. on pp. 1, 4).

[3] Bradley Scott, Martin Seyres, Fraser Philp, Edward K Chadwick, and Dimitra Blana. «Healthcare applications of single camera markerless motion capture: a scoping review». In: *PeerJ* 10 (2022). DOI: https://doi.org/10.7717/peerj.13517 (cit. on pp. 1, 4, 6).

[4] Jen-Li Chung, Lee-Yeng Ong, and Meng-Chew Leow. «Comparative Analysis of Skeleton-Based Human Pose Estimation». In: *Future Internet* 14.12 (2022). ISSN: 1999-5903. DOI: 10.3390/fi14120380. URL: https://www.mdpi.com/1999-5903/14/12/380 (cit. on pp. 1, 11).

[5] Yazdan Hashemi, Ghorban Taghizadeh, Akram Azad, and Saeed Behzadipour. «The effects of supervised and non-supervised upper limb virtual reality exercises on upper limb sensory-motor functions in patients with idiopathic Parkinson's disease». In: *Human Movement Science* 85 (2022), p. 102977. ISSN: 0167-9457. DOI: https://doi.org/10.1016/j.humov.2022.102977. URL: https://www.sciencedirect.com/science/article/pii/S0167945722000574 (cit. on p. 3).

[6] Chieh-Sen Chuang et al. «Effects of modern technology (exergame and virtual reality) assisted rehabilitation vs conventional rehabilitation in patients with Parkinson's disease: a network meta-analysis of randomised controlled trials». In: *Physiotherapy* 117 (2022). DOI: https://doi.org/10.1016/j.physio.2022.07.001 (cit. on p. 3).

[7] Elvira Maranesi et al. «The Effect of Non-Immersive Virtual Reality Exergames versus Traditional Physiotherapy in Parkinson's Disease Older Patients: Preliminary Results from a Randomized-Controlled Trial». In: *International Journal of Environmental Research and Public Health* 19.22 (2022). ISSN: 1660-4601. DOI: 10.3390/ijerph192214818. URL: https://www.mdpi.com/1660-4601/19/22/14818 (cit. on p. 3).

[8] Marcus Barth, Robert Möbius, Peter Themann, Erdem Güresir, Cornelia Matzke, Dirk Winkler, and Ronny Grunert. «Functional improvement of patients with Parkinson syndromes using a rehabilitation training software». In: *Frontiers in Neurology* 14 (2023). ISSN: 1664-2295. DOI: 10.3389/fneur.2023.1210926. URL: https://www.frontiersin.org/journals/neurology/articles/10.3389/fneur.2023.1210926 (cit. on pp. 3, 6, 58).

[9] Po-Jung Chen, Hui-Fen Hsu, Kuei-Min Chen, and Frank Belcastro. «VR exergame interventions among older adults living in long-term care facilities: A systematic review with Meta-analysis». In: *Annals of Physical and Rehabilitation Medicine* 66.3 (2023), p. 101702. ISSN: 1877-0657. DOI: https://doi.org/10.1016/j.rehab.2022.101702. URL: https://www.sciencedirect.com/science/article/pii/S1877065722000744 (cit. on p. 4).

[10] Patrick Manser, Fabian Herold, and Eling D. de Bruin. «Components of effective exergame-based training to improve cognitive functioning in middle-aged to older adults – A systematic review and meta-analysis». In: *Ageing Research Reviews* 99 (2024), p. 102385. ISSN: 1568-1637. DOI: https://doi.org/10.1016/j.arr.2024.102385. URL: https://www.sciencedirect.com/science/article/pii/S1568163724002034 (cit. on p. 4).

[11] Francisco-Javier Peláez-Vélez, Martina Eckert, Mariano Gacto-Sánchez, and Ángel Martínez-Carrasco. «Use of Virtual Reality and Videogames in the Physiotherapy Treatment of Stroke Patients: A Pilot Randomized Controlled Trial». In: *International Journal of Environmental Research and Public Health* 20.6 (2023). ISSN: 1660-4601. DOI: 10.3390/ijerph20064747. URL: https://www.mdpi.com/1660-4601/20/6/4747 (cit. on p. 4).

[12] Jan Stenum, Kendra M. Cherry-Allen, Connor O. Pyles, Rachel D. Reetzke, Michael F. Vignos, and Ryan T. Roemmich. «Applications of Pose Estimation in Human Health and Performance across the Lifespan». In: *Sensors* 21.21 (2021). ISSN: 1424-8220. DOI: 10.3390/s21217315. URL: https://www.mdpi.com/1424-8220/21/21/7315 (cit. on pp. 4, 6).

[13] Gil Serrancolí, Peter Bogatikov, Joana Palés Huix, Ainoa Forcada Barberà, Antonio J. Sánchez Egea, Jordi Torner Ribé, Samir Kanaan-Izquierdo, and Antoni Susín. «Marker-Less Monitoring Protocol to Analyze Biomechanical

Joint Metrics During Pedaling». In: *IEEE Access* 8 (2020), pp. 122782–122790. DOI: 10.1109/ACCESS.2020.3006423 (cit. on p. 4).

[14] Jan Stenum, Cristina Rossi, and Ryan T. Roemmich. «Two-dimensional video-based analysis of human gait using pose estimation». In: *PLOS Computational Biology* 17.4 (Apr. 2021), pp. 1–26. DOI: 10.1371/journal.pcbi.1008935. URL: https://doi.org/10.1371/journal.pcbi.1008935 (cit. on p. 4).

[15] Gökhan Güney, Talisa S. Jansen, Sebastian Dill, Jörg B. Schulz, Manuel Dafotakis, Christoph Hoog Antink, and Anne K. Braczynski. «Video-Based Hand Movement Analysis of Parkinson Patients before and after Medication Using High-Frame-Rate Videos and MediaPipe». In: *Sensors* 22.20 (2022). ISSN: 1424-8220. DOI: 10.3390/s22207992. URL: https://www.mdpi.com/1424-8220/22/20/7992 (cit. on pp. 5, 11).

[16] Talisa S Jansen, Gökhan Güney, Bergita Ganse, Mariana H G Monje, Jörg B Schulz, Manuel Dafotakis, Christoph Hoog Antink, and Anne K Braczynski. «Video-based analysis of the blink reflex in Parkinson's disease patients». In: *Biomedical engineering online* 23 (2024). DOI: https://doi.org/10.1186/s12938-024-01236-w. URL: https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/s12938-024-01236-w (cit. on pp. 5, 11).

[17] Hao-Ping Lin et al. «Exploring the Feasibility of Computer Vision for Detecting Post-Stroke Compensatory Movements». In: *2023 International Conference on Rehabilitation Robotics (ICORR)*. 2023, pp. 1–6. DOI: 10.1109/ICORR58425.2023.10304697 (cit. on pp. 5, 6, 11).

[18] Bruno Pereira, Bruno Cunha, Paula Viana, Maria Lopes, Ana S. C. Melo, and Andreia S. P. Sousa. «A Machine Learning App for Monitoring Physical Therapy at Home». In: *Sensors* 24.1 (2024). ISSN: 1424-8220. DOI: 10.3390/s24010158. URL: https://www.mdpi.com/1424-8220/24/1/158 (cit. on pp. 5, 6).

[19] Chang Soon Tony Hii, Kok Beng Gan, Nasharuddin Zainal, Norlinah Mohamed Ibrahim, Shahrul Azmin, Siti Hajar Mat Desa, Bart van de Warrenburg, and Huay Woon You. «Automated Gait Analysis Based on a Marker-Free Pose Estimation Model». In: *Sensors* 23.14 (2023). ISSN: 1424-8220. DOI: 10.3390/s23146489. URL: https://www.mdpi.com/1424-8220/23/14/6489 (cit. on pp. 5, 6).

[20] Valentin Bazarevsky, Ivan Grishchenko, and Eduard Gabriel Bazavan. *MediaPipe BlazePose GHUM 3D*. URL: https://storage.googleapis.com/mediapipe-assets/Model%20Card%20BlazePose%20GHUM%203D.pdf (cit. on pp. 5, 12, 50).

[21] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. *BlazePose: On-device Real-time Body Pose tracking.* 2020. arXiv: `2006.10204 [cs.CV]`. URL: `https://arxiv.org/abs/2006.10204` (cit. on pp. 5, 10, 11).

[22] Nicola Marotta, Dario Calafiore, Claudio Curci, Lorenzo Lippi, Valerio Ammendolia, Francesco Ferraro, Marco Invernizzi, and Alessandro de Sire. «Integrating virtual reality and exergaming in cognitive rehabilitation of patients with Parkinson disease: a systematic review of randomized controlled trials». In: *European Journal of Physical and Rehabilitation Medicine* 58 (2022), pp. 818–26. DOI: `https://doi.org/10.23736/S1973-9087.22.07643-2` (cit. on p. 6).

[23] Yoonsin Oh and Stephen Yang. «Defining exergames & exergaming». In: Jan. 2010. URL: `https://www.researchgate.net/publication/230794344_Defining_exergames_exergaming` (cit. on p. 7).

[24] Microsoft. *Microsoft Kinect.* 2024. URL: `https://learn.microsoft.com/it-it/windows/apps/design/devices/kinect-for-windows` (cit. on p. 7).

[25] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. «OpenPifPaf: Composite Fields for Semantic Keypoint Detection and Spatio-Temporal Association». In: *IEEE Transactions on Intelligent Transportation Systems* (Mar. 2021), pp. 1–14 (cit. on pp. 9, 11).

[26] Khanh LeViet and Yu-hui Chen. *Pose estimation and classification on edge devices with MoveNet and TensorFlow Lite.* 2021. URL: `https://blog.tensorflow.org/2021/08/pose-estimation-and-classification-on-edge-devices-with-MoveNet-and-TensorFlow-Lite.html` (cit. on p. 9).

[27] Google. *MoveNet: Ultra fast and accurate pose detection model.* 2021. URL: `https://www.tensorflow.org/hub/tutorials/movenet?hl=en` (cit. on p. 9).

[28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* 2019. arXiv: `1801.04381 [quant-ph]` (cit. on pp. 9, 11).

[29] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context.* 2015. arXiv: `1405.0312 [cs.CV]`. URL: `https://arxiv.org/abs/1405.0312` (cit. on pp. 9, 10).

[30] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.* 2019. arXiv: `1812.08008 [cs.CV]`. URL: `https://arxiv.org/abs/1812.08008` (cit. on pp. 9, 11).

[31] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. «2D Human Pose Estimation: New Benchmark and State of the Art Analysis». In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June 2014 (cit. on p. 10).

[32] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. *DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation.* 2016. arXiv: `1511.06645 [cs.CV]`. URL: `https://arxiv.org/abs/1511.06645` (cit. on pp. 10, 11).

[33] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. *RMPE: Regional Multi-person Pose Estimation.* 2018. arXiv: `1612.00137 [cs.CV]`. URL: `https://arxiv.org/abs/1612.00137` (cit. on pp. 10, 11).

[34] Camillo Lugaresi et al. *MediaPipe: A Framework for Building Perception Pipelines.* 2019. arXiv: `1906.08172 [cs.DC]`. URL: `https://arxiv.org/abs/1906.08172` (cit. on p. 10).

[35] Google. *Pose Landmarks Detection with MediaPipe Tasks.* Tech. rep. Google, 2023. URL: `https://colab.research.google.com/github/googlesamples/mediapipe/blob/main/examples/pose_landmarker/python/%5BMediaPipe_Python_Tasks%5D_Pose_Landmarker.ipynb` (cit. on pp. 11, 24).

[36] Google. *Pose landmark detection guide for Python.* Tech. rep. Google, 2024. URL: `https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker/python` (cit. on pp. 11, 23, 24).

[37] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, Bill Freeman, Rahul Sukthankar, and Cristian Sminchisescu. «GHUM and GHUML: Generative 3D Human Shape and Articulated Pose Models». In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (Oral).* 2020, pp. 6184–6193. URL: `https://openaccess.thecvf.com/content_CVPR_2020/html/Xu_GHUM__GHUML_Generative_3D_Human_Shape_and_Articulated_Pose_CVPR_2020_paper.html` (cit. on p. 11).

[38] Unity Technologies. *Unity.* 2024. URL: `https://unity.com/` (cit. on pp. 13, 14).

[39] Epic Games. *Unreal Engine.* 2024. URL: `https://www.unrealengine.com/en-US` (cit. on p. 13).

[40] Juan Linietsky and Ariel Manzur. *Godot Engine.* 2024. URL: `https://godotengine.org/` (cit. on p. 13).

[41] NuGet. *WebSocketSharp.* 2016. URL: `https://www.nuget.org/packages/WebSocketSharp` (cit. on p. 14).

[42] Newtonsoft. *JSON.Net.* 2024. URL: `https://www.newtonsoft.com/json` (cit. on p. 14).

[43] Trygve Reenskaug. *MVC*. Tech. rep. 2003. URL: `https://folk.universitetetioslo.no/trygver/themes/mvc/mvc-index.html` (cit. on pp. 14, 35).

[44] OpenJS Foundation. *Node.js*. 2024. URL: `https://nodejs.org/en` (cit. on p. 14).

[45] Axel Dalbard and Jesper Isacson. «Comparative study on performance between ASP.NET and Node.js Express for web-based calculation tools». PhD thesis. 2021. URL: `https://urn.kb.se/resolve?urn=urn:nbn:se:hj:diva-53664` (cit. on p. 14).

[46] inc. NPM. *NPM*. 2024. URL: `https://www.npmjs.com/` (cit. on p. 14).

[47] Inc. MongoDB. *MongoDB*. 2024. URL: `https://www.mongodb.com/` (cit. on p. 15).

[48] Mahmoud Moustafa Eyada, Walaa Saber, Mohammed M. El Genidy, and Fathy Amer. «Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments». In: *IEEE Access* 8 (2020), pp. 110656–110668. DOI: `10.1109/ACCESS.2020.3002164` (cit. on pp. 15, 38).

[49] Seyyed Hamid Aboutorabi[a], Mehdi Rezapour, Milad Moradi, and Nasser Ghadiri. «Performance evaluation of SQL and MongoDB databases for big e-commerce data». In: *2015 International Symposium on Computer Science and Software Engineering (CSSE)*. 2015, pp. 1–7. DOI: `10.1109/CSICSSE.2015.7369245` (cit. on pp. 15, 38).

[50] OpenJS Foundation. *jQuery*. 2024. URL: `https://jquery.com/` (cit. on pp. 15, 39).

[51] Bootstrap Team. *Bootsrap*. 2024. URL: `https://getbootstrap.com/` (cit. on pp. 15, 39).

[52] Wesley Eddy. *Transmission Control Protocol (TCP)*. RFC 9293. Aug. 2022. DOI: `10.17487/RFC9293`. URL: `https://www.rfc-editor.org/info/rfc9293` (cit. on pp. 16, 22).

[53] Alexey Melnikov and Ian Fette. *The WebSocket Protocol*. RFC 6455. Dec. 2011. DOI: `10.17487/RFC6455`. URL: `https://www.rfc-editor.org/info/rfc6455` (cit. on p. 16).

[54] Henrik Nielsen, Jeffrey Mogul, Larry M Masinter, Roy T. Fielding, Jim Gettys, Paul J. Leach, and Tim Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. June 1999. DOI: `10.17487/RFC2616`. URL: `https://www.rfc-editor.org/info/rfc2616` (cit. on p. 17).

[55] Kyzer R. Davis, Brad Peabody, and P. Leach. *Universally Unique IDentifiers (UUIDs)*. RFC 9562. May 2024. DOI: `10.17487/RFC9562`. URL: `https://www.rfc-editor.org/info/rfc9562` (cit. on pp. 18, 24, 25).

[56] Muhammad Ovais Ahmad, Jouni Markkula, and Markku Oivo. «Kanban in software development: A systematic literature review». In: *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. 2013, pp. 9–16. DOI: `10.1109/SEAA.2013.28` (cit. on p. 19).

[57] Kent Beck et al. *Manifesto for Agile Software Development*. 2001. URL: `http://www.agilemanifesto.org/` (cit. on p. 19).

[58] github. *GitHub*. 2020. URL: `https://github.com/` (cit. on p. 19).

[59] Tom Preston-Werner. *Semantic Versioning 2.0.0*. URL: `https://semver.org/` (cit. on p. 19).

[60] Sean Leonard. *The text/markdown Media Type*. RFC 7763. Mar. 2016. DOI: `10.17487/RFC7763`. URL: `https://www.rfc-editor.org/info/rfc7763` (cit. on p. 19).

[61] OpenJS Foundation. *ESLint*. 2024. URL: `https://eslint.org/` (cit. on p. 21).

[62] ganeshsar. *Multithreaded Unity Python MediaPipe Body/Pose*. URL: `https://github.com/ganeshsar/UnityPythonMediaPipeBodyPose` (cit. on pp. 22, 26).

[63] Chris Newman and Graham Klyne. *Date and Time on the Internet: Timestamps*. RFC 3339. July 2002. DOI: `10.17487/RFC3339`. URL: `https://www.rfc-editor.org/info/rfc3339` (cit. on p. 26).

[64] Creative Commons. *CC0 1.0 Universal Deed*. URL: `https://creativecommons.org/publicdomain/zero/1.0/` (cit. on p. 28).

[65] CGTrader. *CGTrader*. 2024. URL: `https://www.cgtrader.com/` (cit. on p. 28).

[66] Lennart Demes. *ambientCG*. 2024. URL: `https://ambientcg.com/` (cit. on p. 28).

[67] Music Technology Group of Universitat Pompeu Fabra. *Freesound*. 2024. URL: `https://freesound.org/` (cit. on p. 35).

[68] Microsoft. *Typescript*. 2024. URL: `https://www.typescriptlang.org/` (cit. on p. 35).

[69] OpenJS Foundation. *ExpressJS*. 2024. URL: `https://expressjs.com/` (cit. on p. 36).

[70] Roy T. Fielding, Mark Nottingham, and Julian Reschke. *HTTP Semantics*. RFC 9110. June 2022. DOI: `10.17487/RFC9110`. URL: `https://www.rfc-editor.org/info/rfc9110` (cit. on p. 37).

[71] Sergey Lyubka. *mongoose*. 2024. URL: `https://mongoosejs.com/` (cit. on p. 37).

[72] MongoDB. *ObjectId()*. 2024. URL: https://www.mongodb.com/docs/manual/reference/method/ObjectId/ (cit. on p. 39).

[73] Tim Berners-Lee and Daniel W. Connolly. *Hypertext Markup Language - 2.0*. RFC 1866. Nov. 1995. DOI: 10.17487/RFC1866. URL: https://www.rfc-editor.org/info/rfc1866 (cit. on p. 39).

[74] Yuu Yamashita. *pyenv*. 2019. URL: https://pypi.org/project/pyenv/ (cit. on p. 42).

[75] The MathWorks Inc. *MATLAB version: 24.1.0.2603908 (R2024a)*. Natick, Massachusetts, United States, 2024. URL: https://www.mathworks.com (cit. on p. 44).

[76] Michael B. Jones, John Bradley, and Nat Sakimura. *JSON Web Token (JWT)*. RFC 7519. May 2015. DOI: 10.17487/RFC7519. URL: https://www.rfc-editor.org/info/rfc7519 (cit. on p. 58).

[77] Eric Rescorla. *HTTP Over TLS*. RFC 2818. May 2000. DOI: 10.17487/RFC2818. URL: https://www.rfc-editor.org/info/rfc2818 (cit. on p. 58).

[78] Ricardo Cabello. *three.js*. 2024. URL: https://threejs.org/ (cit. on p. 58).

[79] Diego Marcos, Don McCurdy, and Kevin Ngo. *A-Frame*. 2024. URL: https://aframe.io/ (cit. on p. 58).

[80] Greggman. *WebGL Fundamentals*. 2015. URL: http://webglfundamentals.org/ (cit. on p. 59).