



# Design, Development and Evaluation of an Automated Video Analysis Tool for Martial Arts Technique

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Software Engineering and Internet Computing**

by

**Samir Duvelek, BSc.**

Registration Number 01426832

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Grechenig

Assistance: Dr. Dominik Hölbling, MSc, Bakk.

Vienna, 1<sup>st</sup> December, 2025

\_\_\_\_\_  
Signature Author

\_\_\_\_\_  
Signature Advisor





# Design, Development and Evaluation of an Automated Video Analysis Tool for Martial Arts Technique

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering and Internet Computing**

eingereicht von

**Samir Duvelek, BSc.**

Matrikelnummer 01426832

ausgeführt am  
Institut für Information Systems Engineering  
Forschungsbereich Business Informatics  
Forschungsgruppe Industrielle Software  
der Fakultät für Informatik der Technischen Universität Wien

**Betreuung:** Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Grechenig

Wien, 1. Dezember 2025



# Erklärung zur Verfassung der Arbeit

Samir Duvelek, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Dezember 2025

---

Samir Duvelek



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Danksagung

Ich möchte Professor Grechenig für die Betreuung bei dieser Arbeit danken.

Ich möchte mich bei Dr. Hölbing bedanken für die Unterstützung beim Anfertigen dieser Arbeit und für all die Möglichkeiten, die sich durch die Arbeit an diesem Thema ergeben haben.

Ich möchte Laura danken, die zur selben Zeit ihre Arbeit geschrieben hat und mir immer die Motivation gegeben hat, weiterzumachen.

Ich danke Mario, der mir immer geholfen hat, die Perspektive nicht zu verlieren, und mich daran erinnert hat, was eigentlich wichtig ist.

Ich danke Anton Kranzl, der mir schon vor all diesen Jahren so geholfen hat und damit vieles möglich gemacht hat.

Ich möchte Katharina danken, die von Beginn dieser Arbeit an für mich da war.

Meiner Tante Feiza, die eine Inspiration für mich ist und mir gezeigt hat, wie weit man kommen kann, wenn man die Arbeit investiert.

Meinen Onkeln und Vorbildern Mele und Zihno.

Alma, die die wichtigste Person beim letzten Push über die Ziellinie war.

Meiner Familie: Amina, Sinan, Amir, Inas, Dina, Suki, Bego, Wasi, Razi, Hamza, Ammar.

Babo, der mir gezeigt hat, dass das Wichtigste die Menschen um einen herum sind.

Mama, die für jeden meiner Erfolge verantwortlich ist.

Und zuletzt meinem Bruder Mustafa, meinem größten Vorbild und der Person, die mich immer und am meisten unterstützt hat



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Acknowledgements

I would like to thank Professor Grechenig for supervising this thesis. I would also like to thank Dr. Hölbing for supporting me throughout the preparation of this work and for all the opportunities that arose from working on this topic.

I want to thank Laura, who was writing her thesis at the same time and always motivated me to keep going.

My thanks go to Mario, who helped me not lose perspective and reminded me of what truly matters.

I would like to thank Anton Kranzl, who supported me all those years ago and made so much possible.

I want to thank Katharina, who was there for me from the very beginning of this work.

To my aunt Feiza, who is an inspiration to me and showed me how far one can go by investing in hard work.

To my uncles and role models Mele and Zihno.

To Alma, who was the most important person during the final push across the finish line.

To my family: Amina, Sinan, Amir, Inas, Dina, Suki, Bego, Wasi, Razi, Hamza, Ammar.

To Babo, who showed me that the most important thing in life is the people around you.

To Mama, who is responsible for every success I've had.

And lastly, to my brother Mustafa, my greatest role model and the person who has always supported me the most.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Kurzfassung

Herausragende Leistungen im Sport, insbesondere im Kampfsport, erfordern eine präzise technische Ausführung, um die bestmögliche Leistung hervorzubringen, das Verletzungsrisiko zu minimieren und taktische Vorteile zu erzielen. Die Analyse der Technik spielt eine entscheidende Rolle bei der Verbesserung der sportlichen Leistung, indem sie Fehler identifiziert und das Training gezielt unterstützt. Manuelle Technikanalyse ist jedoch zeitaufwendig und erfordert fachkundige Coaches und Experten. Während bestehende Forschung bereits automatisierte Technikanalysen auf Basis von 3D-Videoaufnahmen untersucht hat, sind solche Systeme auf teure Ausrüstung angewiesen und daher im täglichen Training kaum praktikabel. Neue Fortschritte im Bereich der Pose Estimation eröffnen nun die Möglichkeit, automatisierte Analysen basierend auf 2D-Videos durchzuführen, welche einfach mit einem Smartphone während des Trainings aufgezeichnet werden können.

Diese Arbeit stellt die Konzeption, Entwicklung und Evaluation einer Webanwendung vor, die eine automatisierte Analyse von Athleten hochgeladenen 2D-Videos ermöglicht. Das System wurde auf Basis aktueller Forschungsergebnisse und Anforderungen eines Experten für Kickboxen und Karate-Kumite entwickelt und unterstützt zwei Pose Estimation Modelle: MoveNet und OpenPose. Die Evaluation erfolgte mit einem Datensatz bestehend aus 3D- und 2D-Videos von 44 Athleten mit unterschiedlichem Leistungsniveau, die die Technik des doppelten Seitwärtskicks ausführten. Die Übereinstimmung zwischen der Analyse auf Basis von 2D- und 3D-Daten wurde mithilfe von Bland-Altman-Plots überprüft.

Die Ergebnisse zeigen eine hohe Übereinstimmung zwischen der 2D- und 3D-Analyse. Die Übereinstimmung war leicht höher bei der Analyse, die das OpenPose Model verwendete. Die Analyse mit dem OpenPose Model zeigte eine mittlere Abweichung von  $<0,04s$  bei Merkmalen für die Dauer der einzelnen Phasen der Technik,  $<15^\circ$  bei Merkmalen für Gelenkwinkel und  $<13\%$  bei relativen Körperpositionsmerkmalen. Diese Ergebnisse deuten darauf hin, dass das System ein kostengünstiges und praxisnahes Werkzeug für Athleten im täglichen Training darstellen kann.

**Keywords:** *Pose Estimation, Kampfsport, OpenPose, Künstliche Intelligenz, Motion Capturing, Videoanalyse, Sport*



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Abstract

Excelling in sports, especially combat sports, requires precise technical execution to enhance performance, reduce injury risk, and achieve tactical advantage. Technique analysis plays a key role in improving athletic performance by identifying flaws and informing training. However, manual technique analysis is time-consuming and requires expert input. While existing research has explored automated technique analysis using 3D video recordings, such systems rely on expensive equipment that is impractical for daily training. Recent advancements in pose detection now offer the potential for automated analysis based on 2D videos, which can be conveniently recorded using a smartphone.

This research presents the design, development, and evaluation of a web application that provides automated analysis of user-uploaded 2D videos. The system was developed using state-of-the-art research and requirements gathered from a subject matter expert in kickboxing and karate kumite. It supports two pose detection models, MoveNet and OpenPose. The system was evaluated using a dataset of 3D and 2D videos featuring 44 athletes of varying competitive levels performing the double side kick technique. Agreement between the 2D-based and 3D-based analyses from a preliminary study was assessed using Bland-Altman plots.

The evaluation showed strong agreement between the 2D and 3D analyses. The analysis using the OpenPose model displayed slightly better results than the analysis using the MoveNet model. The analysis using the OpenPose model showed a mean difference of  $<0.04s$  for relative phase durations,  $<15^\circ$  for joint angles, and  $<13\%$  for relative body position characteristics. These results indicate that the web application has the potential to serve as an inexpensive and practical tool for athletes in their daily training.

**Keywords:** *Pose Estimation, Martial Arts, OpenPose, Artificial Intelligence, Motion Capturing, Video Analysis, Sports*



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Contents

<b>Kurzfassung</b>	<b>xi</b>
<b>Abstract</b>	<b>xiii</b>
<b>Contents</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	2
1.2 Expected Results . . . . .	3
1.3 Fundamentals . . . . .	4
1.4 State of the art . . . . .	24
1.5 Research Questions . . . . .	26
<b>2 Methodology</b>	<b>29</b>
2.1 Literature Review . . . . .	30
2.2 Requirements Engineering . . . . .	30
2.3 System Design and Implementation . . . . .	31
2.4 Statistical Evaluation . . . . .	31
<b>3 Results</b>	<b>33</b>
3.1 Iterative Development . . . . .	33
3.2 Functional Description . . . . .	38
3.3 Architecture and Technical Description . . . . .	41
<b>4 Evaluation</b>	<b>51</b>
4.1 OpenPose . . . . .	52
4.2 MoveNet . . . . .	52
4.3 MoveNet and OpenPose . . . . .	53
<b>5 Discussion</b>	<b>63</b>
5.1 Identified Requirements . . . . .	63
5.2 Segmented Technique Performance . . . . .	64
5.3 Identifying Characteristics . . . . .	65
5.4 Quality of Analysis: . . . . .	66
	xv

<b>6 Conclusion</b>	<b>71</b>
<b>List of Figures</b>	<b>73</b>
<b>List of Tables</b>	<b>75</b>
<b>Bibliography</b>	<b>77</b>
Raw Data . . . . .	86

# CHAPTER 1

## Introduction

Excelling in sports requires, among other factors, a high degree of technical skill. Proper technique enhances efficiency, minimizes the risk of injury, and is the foundation for maximizing performance [1]. In combat sports particularly, technique execution also serves a tactical purpose [2].

Analyzing one's performance is a crucial tool for developing correct technique. Analysis helps pinpoint flaws in execution and can facilitate the creation of targeted training plans. There are also advantages in analyzing the opponent's technique. Studying opponents' technique can reveal their weaknesses and strengths, potentially gaining a competitive edge [3].

Despite the numerous benefits, technique analysis is a complex undertaking. In combat sports, each technique requires the accurate execution of multiple sequential actions in combination with correct posture and precise targeting in order to achieve maximum effectiveness [4]. This analysis of complex, rapidly performed motions demands a comprehensive, multidimensional approach. Manual analysis is time-intensive and requires highly skilled experts [5].

The analysis of sports technique has been the focus of biomechanical and kinematic research for many years [6]. Artificial intelligence applications can now facilitate semi- or fully automated analysis [7]. Automated analysis offers several benefits. It eliminates the need for a human judge to be present, significantly reducing costs and allowing athletes to perform analysis more frequently. This enables athletes to adjust their training earlier to achieve a better result. Moreover, automated analysis using sensors, such as motion capturing, can return much more precise data than what is visible to the human eye [8].

Several existing automated analysis systems rely on video recordings as the basis for evaluation 1.4. A lot of times these recordings are 3D recordings, created using expensive motion capture systems [8] and these systems are not fully automated, but rather semi-automated, where a domain expert needs to be present or some form of manual processing

is required. The necessity for expensive equipment and the need for a domain expert to be present is not practical for daily training.

Ideally, athletes would have access to technique analysis that is fully automated and does not require specialized equipment or expert supervision. Such a system would operate using only 2D video that can be recorded on a standard smartphone.

For a fully automated system to assess how well a technique was executed, the criteria for a good technique must be defined in a quantifiable manner. This is a complex task, as the criteria for a good technique can vary greatly between different techniques and sports [5]. Nevertheless, several studies have proposed such criteria for specific techniques, outlining measurable characteristics that indicate proper execution [9]. If these quantifiable characteristics could be reliably extracted from a 2D video recording, it would make it feasible to generate a fully automated analysis.

In recent years different models have been developed to detect human poses in 2D images [10] [11]. These models can be used to analyze every frame of a video and extract the pose of the athlete. This creates an abstracted representation of the athlete's performance, with the unnecessary noise of the image removed, with only the information critical for analysis remaining. This abstracted representation is much simpler to work with, and can be used to analyze the athlete's performance in a much more efficient way.

If the abstracted representation of the athlete's performance can be matched against the defined criteria for good technique, it can be used to create an application that provides detailed, automated analysis of an athlete's performance from 2D video alone. A high quality analysis based on 2D videos would be a great benefit for athletes, as it would allow them to analyze their performance more frequently and at a lower cost.

### 1.1 Problem Statement

This thesis focuses on the fully automated software-based analysis of the double side kick. The double side kick is a technique used in pointfighting kickboxing. Analyzing the double side kick is a complex process requiring tracking of multiple factors in order to model high performance [2]. These factors are defined as a number of characteristics strongly linked with competition success and positive expert evaluation of the technique [12] [2].

Extracting these characteristics requires extensive knowledge, expensive 3D motion capture technology, and a sufficient amount of time [12]. One of the studies exploring the double side kick employed the Vicon 3D motion capturing system to record 3D videos of various athletes performing the double side kick. The technique was then semi-automatically analyzed using 3D video recordings [12]. Performances were simultaneously recorded using 2D cameras.

The automated analysis developed in this thesis uses these 2D video recordings to create an analysis that is comparable to the one using 3D video. A comprehensive data set of

3D and 2D videos showing 44 athletes executing the double side kick is the foundation of the software application developed as part of this thesis. The 3D analysis results from previous research [9] [12] serve as the gold standard for verifying the 2D analysis.

The analysis in this thesis is fully automated; it requires no manual input from a domain expert. A lot of the existing studies require a laboratory setting and a domain expert to analyze the data, which makes the analysis expensive and time-consuming. The application in this thesis has the goal to support athletes and coaches in their daily training. Therefore, the analysis is based on 2D videos, which can be easily recorded with a smartphone. As the athlete often does not have access to a domain expert, the application also has to inform the athlete how to record themselves correctly and how to execute the technique to ensure a valid analysis. The application has to be readily available to the athlete and easy to use in a training environment, where athletes often only have access to limited technical equipment. Therefore the analysis tool is available as a web application, which can be accessed from any device with an internet connection.

By combining the existing scientifically proven performance characteristics with current advancements in the field of AI expert systems [13] and user-based development [14], it appears feasible to streamline the analysis process. This simplification could provide a broad set of athletes and coaches the opportunity to monitor their progress in training and performance status. While there are already studies that compare the use of 3D and 2D motion capture analysis in other sports, such as cross-country running [15] and youth baseball [16], no research was found for automatic or semi-automatic analysis of martial arts techniques.

This thesis is the first to develop a fully automated analysis of a martial technique based on published analysis and characteristics of the technique. The result of this thesis is a software application that provides athletes with easy access to scientifically grounded feedback on their performance.

## 1.2 Expected Results

Based on the existing literature, the following hypothesis can be proposed. There is no significant difference between the characteristics extracted from the 2D video as part of this thesis and the characteristics extracted from the 3D video as part of previous research. Specifically, the following characteristics are expected to show equivalence:

- (a) The duration of specific technique execution segments relative to the total execution time.
- (b) The vertical knee height at keypoints during technique execution, relative to the height of the trochanter major in a neutral standing position.
- (c) The distance between the knee and the frontal shoulder at specific execution points, relative to the same distance in a neutral position.

- (d) The angle formed by the hip and knee of the standing leg with the vector connecting the knee and hip of the kicking leg at key execution points.
- (e) The angle formed by the hip and heel of the standing leg with the vector connecting the heel and hip of the kicking leg at corresponding points.
- (f) The velocity of vertical knee elevation during a defined segment of the technique.
- (g) The velocity of the kick leg's movement over the course of a specific segment of the technique execution.

Given that the analysis is based on pose estimation models applied to 2D video, the accuracy of extracted characteristics is likely to depend on the performance of the underlying model. As both the OpenPose and MoveNet models are largely trained on similar datasets, the following hypothesis is proposed: there will be no significant difference between the analysis results produced by OpenPose and those produced by MoveNet. However, it is expected that the results derived from MoveNet will show slightly greater agreement with the 3D analysis. This expectation comes from the fact that MoveNet was trained on a dedicated dataset containing athletes and dynamic human motion, potentially offering improved performance in sports-specific contexts [11].

### 1.3 Fundamentals

This chapter provides the background knowledge needed to understand the results of this thesis. This includes fundamental information on the double side kick, web development, statistics and requirements engineering.

#### 1.3.1 Double Side Kick

Kickboxing is a martial art that can take various forms in competitive settings. The most common is the full-contact variant, where, similar to boxing, two athletes compete over several time-restricted rounds. Another form is point fighting, which can be compared to fencing. In this variant, a "point-stop" scoring system is used, where only the first valid strike is awarded points. Depending on the type of competition, the use of techniques can differ greatly [17].

The double side kick is a technique in which the athlete performs two consecutive front-leg kicks without lowering the leg between kicks. The second kick is executed with a forward thrust to close the distance and make contact with the target. This technique is commonly used in competition [18].

The figure 1.1 presents a visual representation of the technique. The keypoints of the technique were defined in a previous study [9], they will be described at a later stage as they are integral to the automated analysis.

Previous studies have identified characteristics that distinguish the execution of the double side kick between athletes with higher competitive achievement and those with lower achievement. Examples of characteristics are the angle created by the upper and lower leg at specific keypoints in the technique execution or the duration of specific parts of the technique [9]. The goal of this thesis is to develop a system that automatically detects these characteristics and supports athletes in refining their technique to align with the execution patterns associated with greater competitive success.

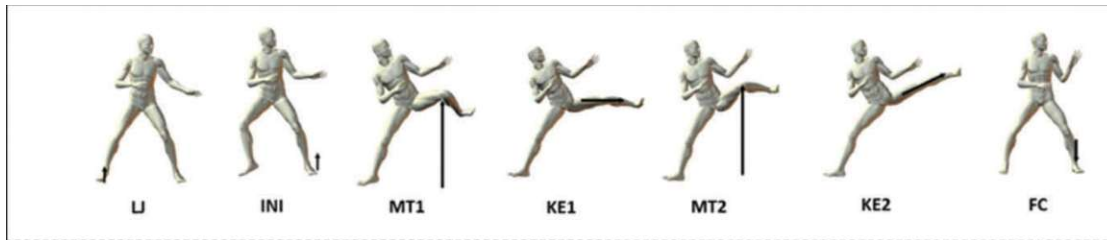


Figure 1.1: The double side kick with previously defined keypoints [9]

### 1.3.2 Statistics

The validation of this system's analysis is performed by comparing its results to those obtained from a system developed in a previous study [9], which used more expensive equipment and experimental conditions that are not suitable for daily use. The goal of this validation is to analyze the agreement between the analysis produced by the system developed in this thesis and the analysis generated by the system using more sophisticated equipment.

A common statistical technique used to assess the strength of the relationship between two variables is correlation. Several correlation methods exist, with the Pearson correlation being one of the most widely used. The result of a correlation analysis is the correlation coefficient, which ranges from  $-1$  to  $+1$ . The closer the coefficient is to either  $+1$  or  $-1$ , the stronger the linear relationship between the two variables. Correlation is often accompanied by linear regression analysis, where a line is fitted to the data to best predict one variable based on the other [19].

However, a high correlation does not necessarily imply agreement [20]. Correlation only describes the strength of the linear association between two datasets. For example, a system could consistently overestimate all measurements compared to another system. In that case, the correlation might still be perfect, while the agreement between the measurements would be poor.

To address this, Bland and Altman introduced the Bland–Altman plot to assess the agreement between two sets of measurements [21]. Their method determines levels of agreement by calculating limits of agreement, defined as the mean difference  $\pm 1.96$  times the standard deviation of the differences between paired measurements. In the corresponding plot, the y-axis shows the difference between measurement A and measurement

B, and the x-axis shows their average  $((A + B)/2)$ . Approximately 95% of the data points are expected to fall within these limits if the differences are normally distributed. The Bland–Altman plot itself does not determine whether the agreement is acceptable; it simply visualizes the bias and the range within which most differences occur. The recommended approach [20], and the one used in this study, is to define in advance the maximum acceptable difference based on domain-specific requirements and then use the plots to check whether these limits are exceeded.

### 1.3.3 Requirements Engineering

A requirement can be defined as a condition or capability that a system, component, or process must satisfy to meet the needs of a user or to achieve a specific objective [22]. Requirements engineering is the systematic process of eliciting, documenting, validating, and managing requirements. Requirements are sourced from stakeholders, individuals, or organizations that are affected in some form by the developed system [23].

The objective of requirements engineering is to ensure a complete understanding of the system to be developed. This includes functional requirements that specify what the system does, and non-functional requirements that specify how the system should behave [22].

Requirements engineering consists of four core activities [23]:

**Elicitation:** Identifying and collecting requirements from stakeholders using appropriate techniques. This activity aims to understand the goals, needs, and constraints of all parties involved.

**Documentation:** Representing the gathered requirements in a clear and structured form using natural language or conceptual models.

**Validation and Negotiation:** Assessing the documented requirements for completeness, consistency, and feasibility.

**Management:** Organizing and maintaining requirements throughout the development process.

For the elicitation of requirements, several techniques are commonly used [24]:

**Interviews:** Directly talking with individual stakeholders to obtain detailed information about their needs and expectations.

**Questionnaires:** Using structured sets of open or closed questions to gather input from a larger group of stakeholders.

**Domain Analysis:** Studying existing systems and their documentation to derive requirements from modern practices.

**Prototyping:** Developing early versions or mock-ups of the system to support stakeholder feedback and refine requirements iteratively.

The approach to requirements engineering depends strongly on the development methodology applied. In plan-driven approaches such as the Waterfall model [25], requirements are typically gathered and fixed at the beginning of the project. In contrast, agile methodologies such as Extreme Programming (XP) treat requirements elicitation as an ongoing process, continuously refined through stakeholder feedback and iterative development cycles [26].

The goal of requirements engineering is to achieve a mutual understanding between stakeholders and developers, ensuring that the system built aligns with actual user needs [22].

### 1.3.4 Architecture and Web Development

The system was designed with the intention of supporting athletes in their daily training. Therefore, the focus of design and development was to enable athletes to access the system through their device that they will most likely carry with them, their mobile phones. A web-based solution was selected as the most practical approach, because they can be accessed across platforms through a standard browser, providing the ability to be accessed from a modern smartphone while reducing development and maintenance effort compared to native applications that must support both iOS and Android environments [27].

At the same time, the system performs computationally intensive tasks, such as processing video and image data. These operations can be demanding in terms of hardware resources, particularly CPU and GPU performance, memory bandwidth, and storage throughput [28]. The exact hardware requirements of the system were not clear at the beginning of the project, so the first prototype was deployed in a cloud environment. This choice allows for flexible scaling and easier experimentation with resource configurations, which is essential for evaluating what kind of hardware capabilities would be needed for such an automated training system.

Two fundamental scaling strategies are supported in the cloud context [29]. Vertical scaling refers to increasing the resources available to a single machine such as adding more CPU, memory, or GPU capacity. Horizontal scaling refers to adding multiple instances of a service on a single or multiple machines to handle a growing number of computational tasks [29]. Cloud platforms make both scaling strategies straightforward by providing a simple interface to change configuration on the fly.

All the services of the system were containerized to further support both vertical and horizontal scaling. Packaging the service into an isolated container enables a simplified starting of multiple instances of the service as they are not dependent on each other. It also enables a new level of vertical scaling because the hardware resources of the container can be limited only using software without having to adjust the underlying hardware [30].

The system architecture was developed following modern software design principles such as starting with the most minimal possible system that solves the problem at hand

and then iterating on that solution [31]. This avoids over-engineering and helps with early feedback from stakeholders that can be quickly integrated into future iterations. Furthermore, the system was developed with modern distributed-systems considerations in mind. The considerations were how to handle concurrent access, manage distributed state, and ensure reliability under load [29].

This combination of web-based development with a cloud environment infrastructure, enables athletes to access the system through their mobile phones ensuring access during training, while enabling the system access to the hardware resources needed for the computationally intensive task of image processing.

### 1.3.5 Computer Vision

This section presents technologies that provide the foundation for automated technique analysis. It begins with an overview of Convolutional Neural Networks (CNNs), which are at the core of the pose detection models used in both the OpenPose [32] and MoveNet [11] systems. The section concludes with an overview of the two pose detection models and the datasets used to train these models.

#### Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a specialized type of neural network that excels in working with data where the spatial or temporal relationships between data points is significant. [33]. Such as time-series data where data points represent a chronological sequence of events. Or, image data where a group of neighboring data points can represent a group of pixels displaying a specific object. The architecture was first introduced by Fukushima [34], however it was not widely used due to the hardware limitations of the time [35]. Later, LeCun et al. [36] applied the architecture successfully for the classification of handwritten digits.

CNNs are characterized by the use of convolution operations in at least one of their layers, replacing the matrix multiplication common to other neural network architectures. Convolution is an operation that takes two functions of a real-valued argument as input and produces a third function as output. [33].

$$s(t) = x(t) * w(t)$$

Where  $s(t)$  is the output of the convolution,  $x$  and  $w$  are the input functions and  $*$  is the convolution operator.

The first argument of the convolution  $x(t)$  is referred to as the input and the second argument  $w(t)$  as the kernel or filter. The input is usually a multi-dimensional array, such as an image, and the kernel is a multi-dimensional array of values that are adjusted during the learning process. These are comparable to the weights in a traditional neural network. The output of the convolution  $s(t)$  is referred to as the feature map [33].

In the case of a two-dimensional grid, such as an image, convolution is applied over more than one axis at a time. The input and the kernel are two-dimensional arrays:

$$s(i, j) = \sum_m \sum_n x(m, n)w(i - m, j - n)$$

When the input data is an image, the kernel can be imagined as a sliding window that moves across the image, with every position of the window creating an output value, which contains aggregated information about the data inside the window at that position 1.2.

There are a variety of hyperparameters that influence how the convolution is applied to a dataset [35].

**Kernel Size:** or filter size, determines the number of data points considered during each individual convolutional operation. A larger kernel size aggregates more data points over a larger area, which can result in the loss of smaller features contained inside the data, with features that represent broader patterns inside the data being more pronounced. A larger kernel also leads to larger reductions in the dimensions of the output. In multi-layered architectures, subsequent layers that work with the output of this layer work with less data. This is computationally more efficient, but the later layers are operating on a more abstract version of the original data.

**Padding:** is used to expand the input data, so that information at the borders of the input data is preserved. Without padding, data points situated at the border of the data, would be involved in fewer convolutional operations compared to data points in the interior. This would lead to the output feature map being less influenced by the border data points. As a result, important information at the edges of the input might be underrepresented. The zero-padding approach inputs a series of zero values along the border of the input data.

**Stride:** determines how many datapoints the kernel shifts between every operation. In the case of an image, where every data point represents a pixel, the stride indicates how many pixels the kernel moves across the image for each step of the convolution. The impact of the stride value is similar to that of the kernel size. A higher stride leads to larger reduction in dimensions of the output, while reducing the computational cost of processing.

Parameter sharing and sparse connectivity are two characteristics of CNNs that differentiate them from traditional neural networks [33].

**Sparse Connectivity** means only a subset of the outputs in one layer is connected to the inputs in the next layer [33]. This is in contrast to a fully connected network, where each input is connected to every output in the previous layer [37]. This has the advantage of reducing the number of parameters that need to be learned, which, in turn, reduces the computational cost of training the network [33].

**Parameter Sharing:** means parameters or weights are shared between different inputs 1.3. This is in contrast to a fully connected network, where each input has its own set of

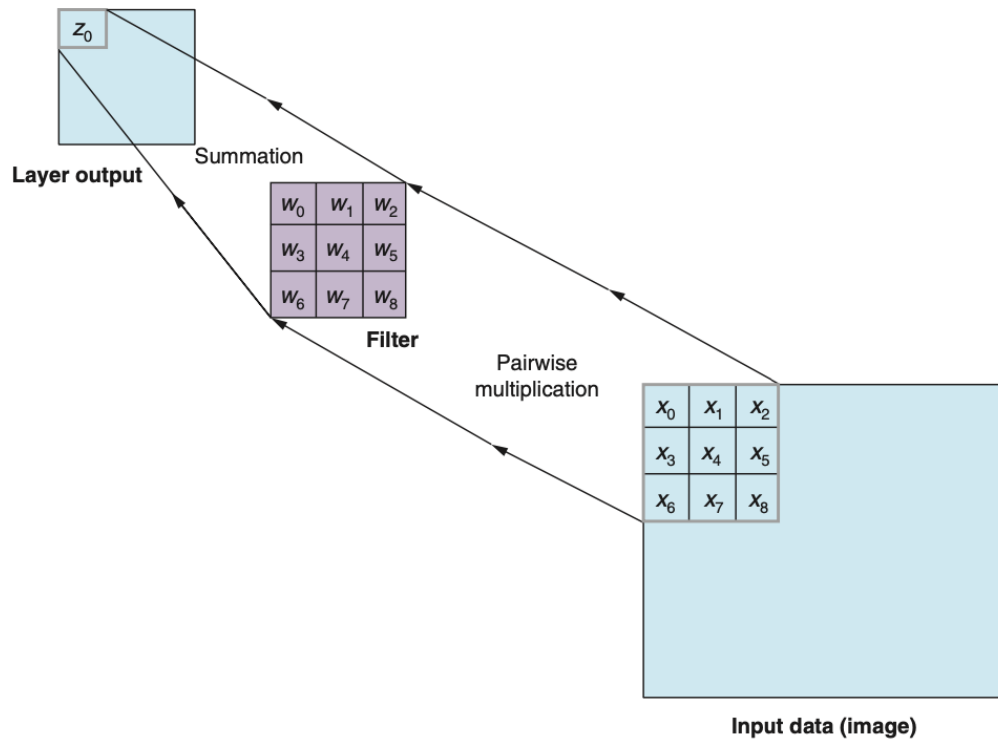


Figure 1.2: CNN sliding window [38]

weights [37]. This is useful in the context of images, as it is reasonable to assume that the same feature can occur at different locations in the image. By sharing the weights, the network can learn to detect similar features at different locations, without having to learn a new set of weights for each location [33].

### Architecture

CNN can be implemented using a variety of architectures [35]. OpenPose is built on top of a VGG-16 [39] architecture and MoveNet is built on top of a MobileNetV2 architecture [40]. This section will provide a general overview of individual components that make up a CNN and conclude with a description of how those components are used in the specific architectures.

**Componentes** There are three common type of layers in a CNN architecture [35]:

**Convolutional Layer:** In this layer the convolutional operation is applied to either the original input data or if the layer is positioned inside a multi-layered architecture, to the output of the previous layer. The outputs of the individual convolutional operations are input into a linear or non-linear activation function. Both VGG-16 and MobileNetV2

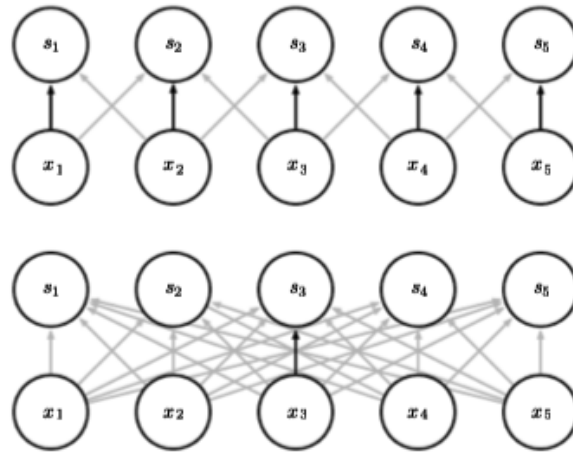


Figure 1.3: CNN parameter sharing [38]

make use of rectified linear unit activation functions (ReLU) and softmax activation functions [41] [40].

The ReLU function returns the non-negative part of its argument:

$$f(x) = \max(0, x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise} \end{cases}$$

The softmax function takes a vector of  $k$  real numbers and returns a probability distribution over  $k$  different possible outcomes [37]. For every value in the vector the softmax output value can be calculated using:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

where  $z$  is the input vector and  $j$  is the index of the output value. Given an output vector with three values  $[0.5, 0.9, 0.2]$  the output of the softmax function would be  $[0.309, 0.461, 0.229]$ .

**Pooling Layer:** This layer produces an aggregated value of its input by applying a pooling function. The pooling function is not applied to the entire dataset, as with the convolution operation, a filter of a specific size is applied over the data. The outputs of these individual pooling functions represent the output of the pooling layer.

The max pooling function operates by returning the maximum value from a given input region. For instance, when applied to a convolutional layer with a  $3 \times 3$  filter, the max pooling function will examine the 9 values within that filter and select the highest one as its output. The max pooling function is employed in VGG-16 [41] and MobileNetV2 [40].

Other pooling functions include average pooling or weighted average pooling. The average pooling function calculates the arithmetic mean of all values within the filter. For a  $3 \times 3$  filter, it would sum all 9 values and divide by 9. The weighted average pooling function assigns different weights to each value in the filter based on its distance from the center. Values closer to the center typically receive higher weights [37].

**Classification Layer:** This is the final stage of the CNN. It outputs the network's final output based on the features extracted by the layers beforehand. The input of this layer is represented as a one-dimensional scalar vector. This vector is passed through one or more fully connected layers that are akin to the inner working of a traditional neural network.

**VGG16** VGG is a CNN architecture developed by the Visual Geometry Group at the University of Oxford. It is characterized by a large amount of convolutional layers and a small filter size inside those layers [41].

The architecture has several different configurations that share an identical core structure 1.4. The input is always a fixed-size  $224 \times 224$  RGB image. The filter size inside the individual convolutional layers is  $3 \times 3$ , except for one configuration with a  $1 \times 1$  filter. The max-pooling layers use  $2 \times 2$  filters with a stride of 2 pixels. The fully connected layers are made up of two layers with 4096 nodes and one layer with a 1000 nodes. All layers use the rectified linear unit activation function, except for the output layer which uses the softmax function [41].

A modified version of the VGG-19 configuration is used as part of OpenPose [10].

**MobileNet** MobileNetV2 is a CNN architecture developed by Google with the aim of keeping the computational cost low while maintaining precision [40], the architecture achieves this using two concepts, depthwise separable convolution (DSC) and an inverted residual layer using a linear bottleneck.

**Depthwise separable convolution:** DSC works almost as well as regular convolutions but the computational cost is 8 to 9 times smaller [42]. DSC separates the convolutional operation into two distinct layers. The first layer is a depthwise convolution that applies a single filter to each input channel [40]. The second layer is a point wise convolution that applies a  $1 \times 1$  filter and creates a linear combination of the output of the depthwise convolution. The usual input channels for a 2D image are the three RGB channels [40].

Figure 1.5 depicts a normal convolution with an RGB image  $12 \times 12$  and an output of  $8 \times 8 \times 256$ , with 256 kernels of size  $5 \times 5 \times 3$ . The outputs of the individual kernels are stacked together to create the finished output.

Figure 1.6 depicts the first layer of a depthwise separable convolution where a  $5 \times 5 \times 1$  kernel is iterated for each individual channel, creating three individual  $8 \times 8 \times 1$  outputs. Stacking them together creates a  $8 \times 8 \times 3$  image that is used as input for the following pointwise convolution.

ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input ( $224 \times 224$ RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: Number of parameters (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figure 1.4: VGG architecture [39]

Figure 1.7 shows a  $1 \times 1 \times 3$  kernel used to combine a  $8 \times 8 \times 3$  channel input into a single  $8 \times 8 \times 1$  output.

Figure 1.8 shows 256  $1 \times 1 \times 3$  kernels that transform a  $8 \times 8 \times 3$  channel input into a single  $8 \times 8 \times 256$  output.

**Bottlenecks:** Bottleneck layers have a reduced size compared to the previous layer, while trying to keep the loss of information to a minimum [40]. Thereby reducing the computational cost while trying to keep accuracy high. MobileNetV2 inserts linear bottleneck layers inside the convolutional block. Linear refers to the activation function.

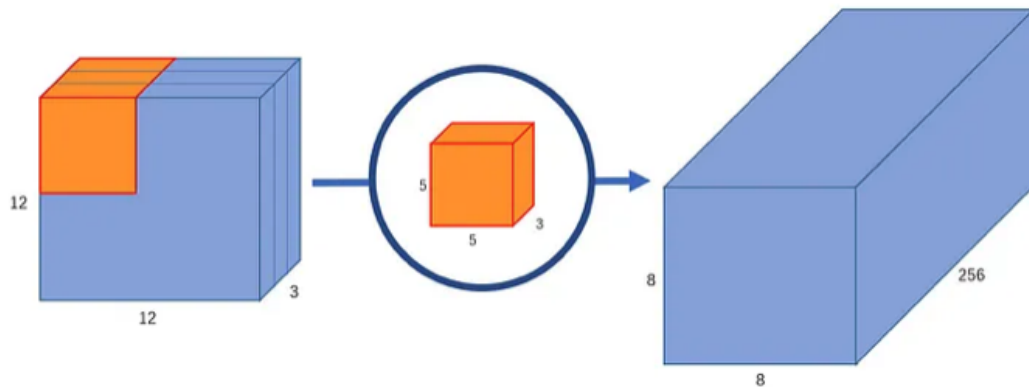


Figure 1.5: A normal convolution with  $8 \times 8 \times 256$  output [43]

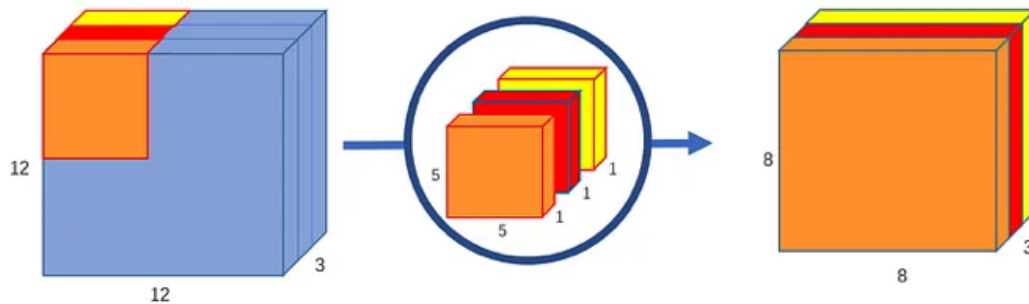


Figure 1.6: The first layer of the depthwise convolution, using 3 kernels to transform a  $12 \times 12 \times 3$  image to a  $8 \times 8 \times 3$  image [43]

A linear activation function leads to better performance of the architecture compared to a non-linear activation function [44].

**Residual layer:** Residual layers solve the issue of architecture with a large amount of layers losing accuracy, the more layers are added [45].

Given a function  $H(x)$  representing the mapping that is trying to be learned by a stack of layers inside a larger architecture. Residual learning replaces  $H(x)$  with the residual function  $H(x) - x$ . Inside the architecture this is achieved by adding a skip-connection that feeds the input directly into the output of several layers 1.9. The input and output are added together before being input into an activation function.

**Architecture:** The architecture of MobileNetV2 combines fully convolutional layers with residual bottleneck layers [40].

A bottleneck layer consists of a regular convolution with a  $1 \times 1$  filter and a RELU

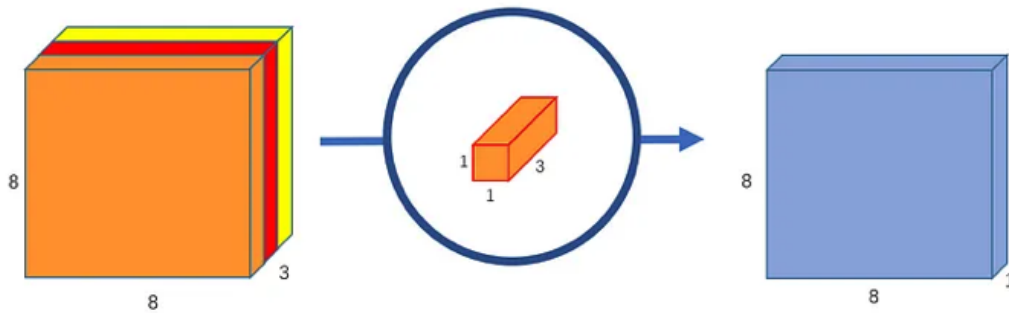


Figure 1.7: Pointwise convolution, transforms an image with 3 channels to a single channel output [43]

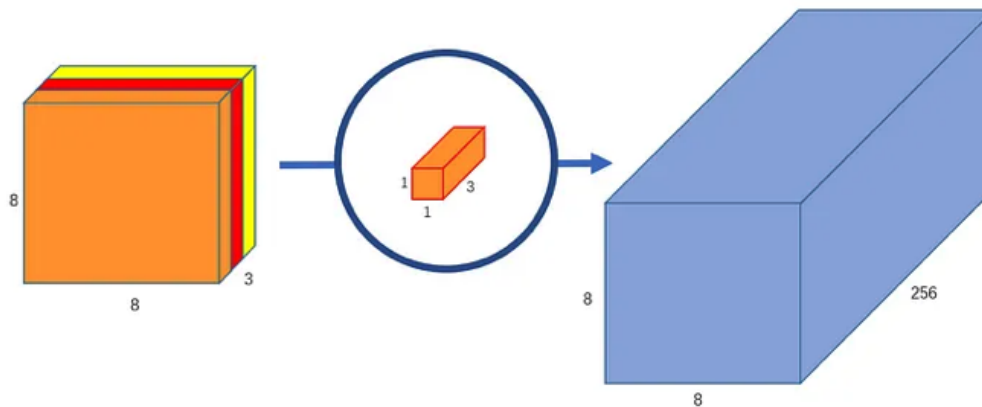


Figure 1.8: Pointwise convolution, transforms an image with 3 channels using 256 kernels to a 256 channel output [43]

activation function, followed by a depthwise convolution with a  $3 \times 3$  filter and a RELU activation function and a regular convolution with a  $1 \times 1$  filter and a linear activation function 1.10. The output is added to the input using the skip-connection [40].

Figure 1.11 presents the various layers that make up the MobileNetV2 architecture.

## Openpose

Openpose uses a bottom-up approach for pose detection. This approach first detects individual body parts or joints and then in a later stage groups them together to form a complete human pose. This approach is in contrast to the top-down approach, where a stand-alone person detector is used to recognize individual persons in the image and perform a pose estimation on them [10].

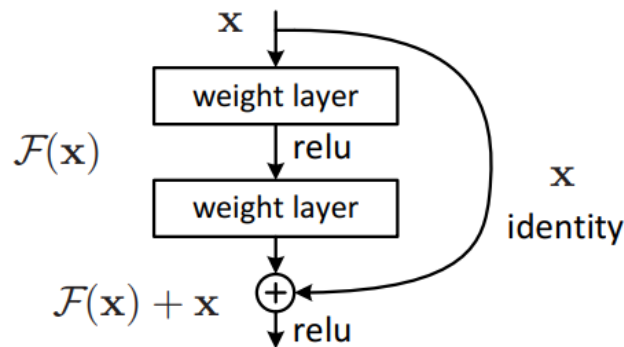


Figure 1.9: Residual layer [45]

Input	Operator	Output
$h \times w \times k$	1x1 conv2d, ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Figure 1.10: A bottleneck residual block with input size  $k$  and output size  $k'$ , stride  $s$  and ration between input and output  $t$  [40]

In the first step of the Openpose multistage process, the image is fed into a convolutional neural network (CNN). The network is a modified version of the VGG-19 network [39]. The VGG-19 network consists of 19 layers, those layers are divided into 16 convolutional layers and 3 fully connected layers. The VGG-19 convolutional layers use 3x3 filters. These filters are windows that slide over the image to create new features that contain information about the relationships between pixels that are close to each other. The pooling layers inside of the convolutional layers replace the output of the network with an aggregated value of the output. This reduces the size of the output and makes the network more efficient in terms of memory usage and processing time. The fully connected layers consist of neurons that are connected to all neurons in the previous layer. And are used to make the final prediction of the network. [39].

This initial output image  $F$  is fed into a series of stages, each composed of an individual CNN. The first stage takes only the initial output  $F$  of the first stage as input and predicts so-called part affinity fields (PAFs). PAFs are sets of 2D vector fields that encode the orientation and location of individual body parts. Subsequent stages take the initial output  $F$  and the output of the previous stage as input and predict refined predictions

Input	Operator	$t$	$c$	$n$	$s$
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Figure 1.11: MobileNetV2 architecture: Every line is a layer repeated  $n$  times.  $c$  is the number of output channels.  $s$  is the stride value.  $t$  is the ratio between input and output [40]

for PAFs.

The predicted PAFs of the last stage are combined with the initial output  $F$  and fed into another series of stages. Each stage being again made up of an individual CNN. These CNNs predict confidence maps. Each confidence map is a 2D heatmap that encodes the likelihood of a body part being present at any given pixel of the image.

The neural network that predicts confidence maps is trained on ground-truth confidence maps. These ground truth confidence maps are generated from images that are annotated with the exact location of body parts. Individual confidence maps  $S_{j,k}^x$  are created in the first step for each individual person  $k$  in the image. Let  $x_{j,k}$  element of  $R^2$  be the exact location of the body part  $j$  of person  $k$ . The value of the location element  $p$  of  $R^2$  in the confidence map  $S_{x,j}$  is then defined as:

$$S_{x_j}^x(p) = \exp\left(-\frac{\|p - x_{j,k}\|_2^2}{\sigma^2}\right)$$

Sigma is the hyperparameter that defines the spread of the Gaussian distribution. The

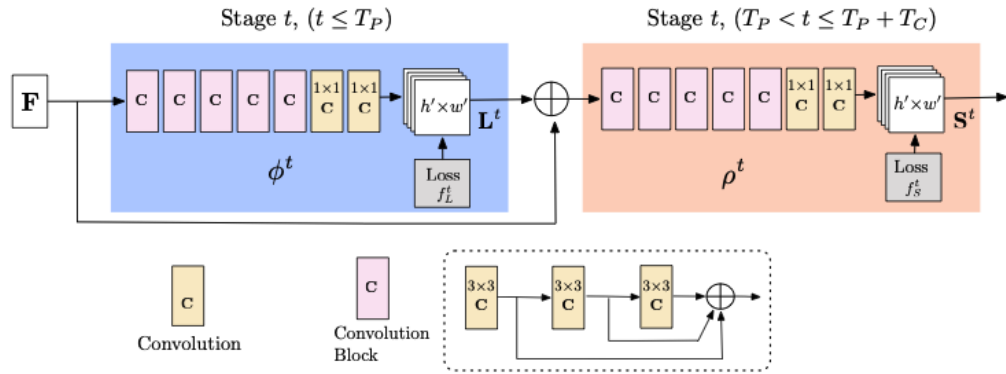


Figure 1.12: OpenPose architecture [32]

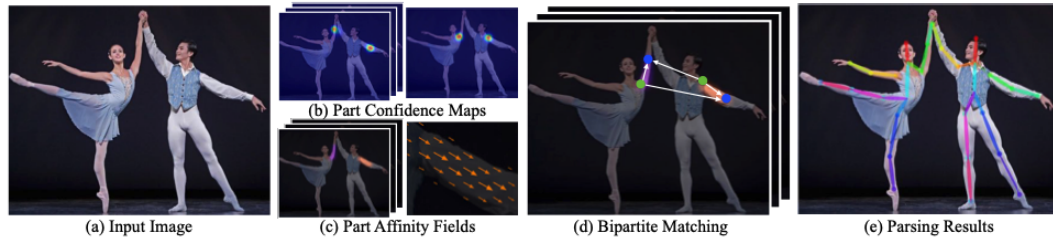


Figure 1.13: OpenPose methods [32]

final confidence map is then created by taking the maximum value of all individual confidence maps for each body part. This is done using the following formula:

$$S_j^*(p) = \max_k S_{j,k}^*(p)$$

The network creates confidence maps and predicts body part locations using nonmaximum suppression (NMS). NMS is a technique used in object detection to increase the accuracy of predictions. The prediction consists of bounding boxes and a confidence score. NMS works by selecting the bounding box with the highest confidence score and removing all other bounding boxes that have a high overlap with the selected bounding box (this paragraph needs to be rewritten) [46].

PAFs address the problem of associating body parts to the correct person. The ground truth PAFs that make up the training data are defined as follows. Let  $x_{j_1,k}$  and  $x_{j_2,k}$  be the locations of body parts  $j_1$  and  $j_2$  from the limb  $c$  of person  $k$ . The value in  $L_{c,k}^*(p)$  is defined as an unit vector pointing from  $j_1$  to  $j_2$  if  $p$  is on the line between  $j_1$  and  $j_2$ . Otherwise, the value is set to zero:

$$L_{c,k}^*(p) = \begin{cases} v & \text{if } p \text{ is on the limb between } x_{j_1,k} \text{ and } x_{j_2,k} \\ 0 & \text{otherwise} \end{cases}$$

with

$$v = \frac{x_{j_2,k} - x_{j_1,k}}{\|x_{j_2,k} - x_{j_1,k}\|}$$

The set of points that are located on the limb is defined as

$$0 \leq v \cdot (p - x_{j_1,k}) \leq l_{c,k} \text{ and } |v \cdot (p - x_{j_1,k})| \leq \sigma_l$$

$\sigma_l$  is the distance in pixels, that describes the width of the limb. The length of the limb is  $l_c$ ,  $k = \|x_{j_2,k} - x_{j_1,k}\|$  and  $v$  is the a vector perpendicular to the limb.

The ground-truth PAF is the average of all affinity fields of all people in the image:

$$L_c^*(p) = \frac{1}{n_c(p)} \sum_k L_{c,k}^*(p)$$

and  $n_c(p)$  is the number of nonzero vectors at point  $p$  for all people  $k$  in the image.

During the testing phase, the alignment of the predicted PAF vector for a limb connecting two body parts is compared to the vector created by connecting the two predicted body part locations. Let  $d_{j_1}$  and  $d_{j_2}$  be the predicted locations of body parts  $j_1$  and  $j_2$  and  $L_c$  be the predicted PAF vector for the limb connecting  $j_1$  and  $j_2$ . The confidence of the estimation is then defined as

$$E = \int_{u=0}^{u=1} L_c(p(u)) \cdot \frac{d_{j_2} - d_{j_1}}{\|d_{j_2} - d_{j_1}\|_2} du$$

With  $p(u)$  defined as the interpolation of body parts  $d_{j_1}$  and  $d_{j_2}$ :

$$p(u) = (1 - u)d_{j_1} + ud_{j_2}$$

### MoveNet

MoveNet uses a bottom-up architecture that consists of two components: a feature extractor and a set of prediction heads.

**Feature Extractor:** The feature extractor is a MobileNetV2 architecture with an attached Feature Pyramid Network (FPN) [11]. A feature pyramid is a structure used in image processing in which the same image is represented at different scales 1.14 to reveal more information about the image than is available with a single-scale representation [47].

Feature Pyramid Networks (FPN) are a type of neural network architecture that make use of feature pyramids to enhance the accuracy of object detection [48]. The architecture

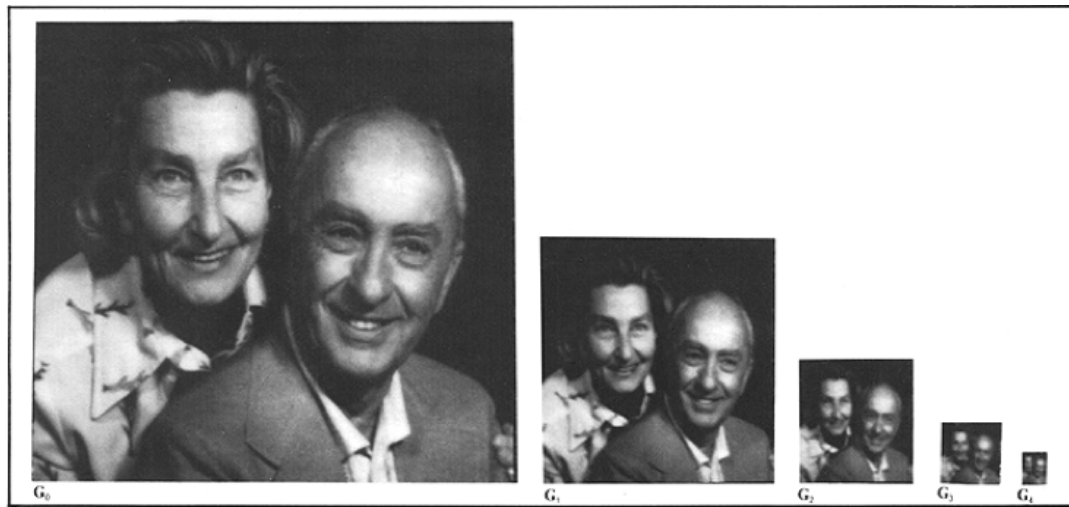


Figure 1.14: Image pyramid: same image represented at different scales [47]

takes an image of arbitrary size as input, and outputs proportional-sized feature maps. The architecture is created sequentially. First a bottom-up pathway creates feature maps at different scales using convolution and pooling approaches [48]. Then a top down approach creates higher scale features that are enhanced by merging features map from the bottom-up approach with top-down features of the same scale. This is done using lateral connections illustrated in figure 1.15. Prediction are created on every single top-down feature map. This kind of architecture significantly improves performance in object detection tasks [48].

**Prediction heads:** There are four prediction heads attached to the FPN, with each predicting a different kind of task. The prediction heads are a modified version of the CenterNet architecture [11].

CenterNet is an object detection system that models complex objects as single points, predicts their location and regresses to the remaining object properties, such as size, orientation and human pose [49]. This approach is in contrast to other systems that immediately predict a bounding box of the object [49]. The input is put into a fully convolutional network that produces a heatmap, with peaks in the heatmap representing object centers. The object bounding box height and width is predicted from image features at each peek 1.16.

For human pose estimation regression is used to predict the joint offsets in pixels [49].

Each of the four prediction heads predictions it own individual task [11]:

**Person center heatmap:** This head predicts the arithmetic mean of all keypoints belonging to one person. This is defined as the center of that person.

**Keypoint regression field:** This head predicts the initial set of keypoints for each person. Using keypoint regression starting from the object center

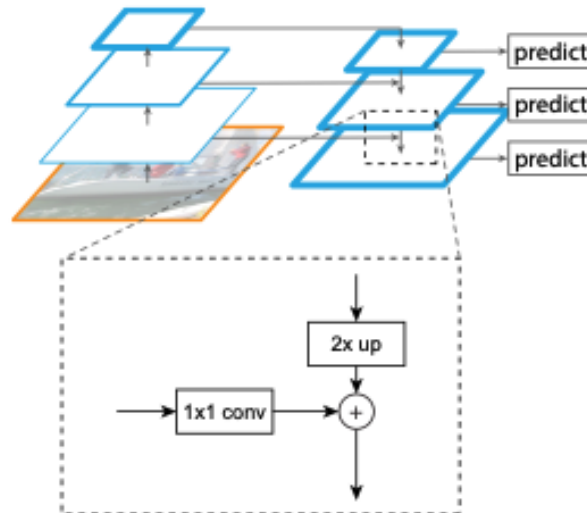


Figure 1.15: Feature Pyramid Network architecture [48]

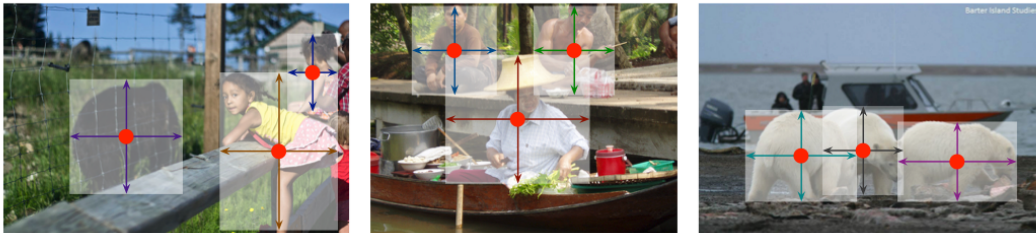


Figure 1.16: CenterNet prediction of center point and bounding box [49]

**Person keypoint heatmap:** This head predicts a refined set of keypoints by multiplying a weight that is proportional to the distance from the regressed keypoints. This is done to filter out background noise

**2D keypoint offset field:** This head predicts the final keypoints output by selecting the coordinates of the maximum from each keypoint heatmap.

Figure 1.17 illustrates the components described above making up the movenet architecture.

### Training Data

Both OpenPose [10] and MoveNet [11] use the Common Object in Context (COCO) dataset to train their respective models. The COCO dataset consists of 328 thousand images with 2.5 million individually labeled objects that appear in those images [50].

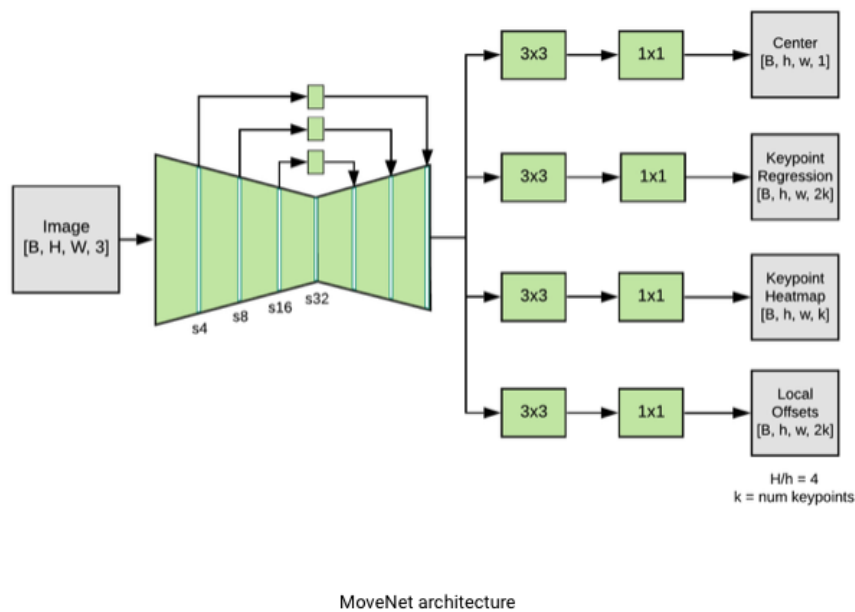


Figure 1.17: MoveNet architecture with FPN and prediction heads [11]

Most of the images in the dataset are gathered from the photo upload site and inside those images there are 91 different types of objects that are labeled inside the COCO dataset; however, only the labeled objects that identify people are relevant for training MoveNet and OpenPose [50].

The images are annotated using a three-part process 1.18 where the type of objects that appear in the image are labeled, then the location of these objects is marked with a single cross and finally the outline of the image is drawn into the image. All of this was done manually using a crowdsourcing approach [50]. All of the crowdsourcing tasks were done using workers from Amazon's Mechanical Turk, a marketplace for outsourcing processes that can be done virtually to a distributed workforce [51].

**Category labeling:** To determine which objects appear in individual images a hierarchical approach was used [52]. The 91 categories were grouped into 11 supercategories. To indicate if a type of objects is present in the image a worker picked from the 11 super-categories, and then picked from the subordinate categories the image icon of the category and dragged it onto the image on top of the object. In this stage only a single instance of the type of objects is marked, even if multiple of those type objects appear in the image. To ensure the labeling quality each image was processed by 8 workers. A type of objects was treated as present, if one of the workers marked is as present. The process took 20 thousand hours to complete [50].

**Instance spotting:** In this stage the individual marked objects were taken as a

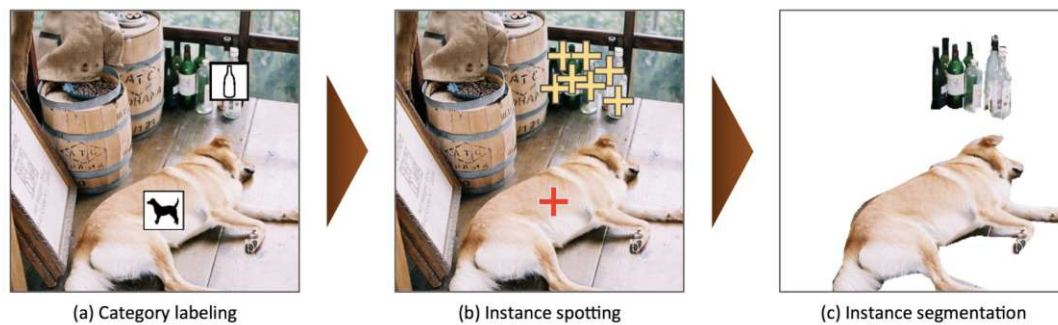


Figure 1.18: The annotation pipeline consists of (a) identifying the objects that appear in the image (b) locating and marking the object instances (c) the segmentation of each individual objects instance [50]

foundation to mark every instance of an objects appearing in the image. Each worker placed a cross on top of each individual object appearing in the image. Each image was labeled by 8 workers and this stage took 10 thousand hours to complete.

**Instance segmentation:** In this image, the real outlines of the identified objects are drawn into the image. For this, a modified version of an user interface developed by Bell et al. [53] was used.

In total 2.5 million object outlines were drawn, with each 1000 segmentation tasks taking roughly 22 hours to complete. Each object outline was only drawn by one worker; however, workers had to complete a training task beforehand and all outlines were subjected to a verification task, where multiple workers judged if the drawn outline matched the real object outline. Badly judged outlines were thrown out and workers that produced multiple badly judged outlines were removed from the project [50].

### Active dataset

In addition to the COCO dataset MoveNet also uses an internal Google dataset called Active for training [11]. The Active dataset contains about 24 thousand images sampled from YouTube fitness videos of people exercising. [54]. Each image is annotated with 17 body keypoints of the person appearing in the image [11].

The Active dataset is split into a training and validation dataset. MoveNet performs significantly better on the Active validation set compared to other pose detection models that were trained only using the COCO dataset. This is likely because the COCO dataset contains fewer images of people exercising. [11].

Evaluation was done by calculating the object keypoint similarity (OKS) [11] which is also used in evaluating the COCO validation set [55]. Every individual detected pose has a set of ground truth keypoints  $[x_1, y_1, v_1, \dots, x_k, y_k, v_k]$  where  $x, y$  are the locations of the individual keypoints and  $v$  is a visibility flag that can take one of three values. 0 means

## 1. INTRODUCTION

that the keypoint is not labeled, 1 means that the keypoint is labeled but not visible and 2 means that the keypoint is labeled and visible. [55]. The OKS is then calculated using:

$$OKS = \sum_i \left[ \exp -d_i^2 / 2s^2 k_i^2 \delta(v_i > 0) \right] / \sum_i [\delta(v_i > 0)]$$

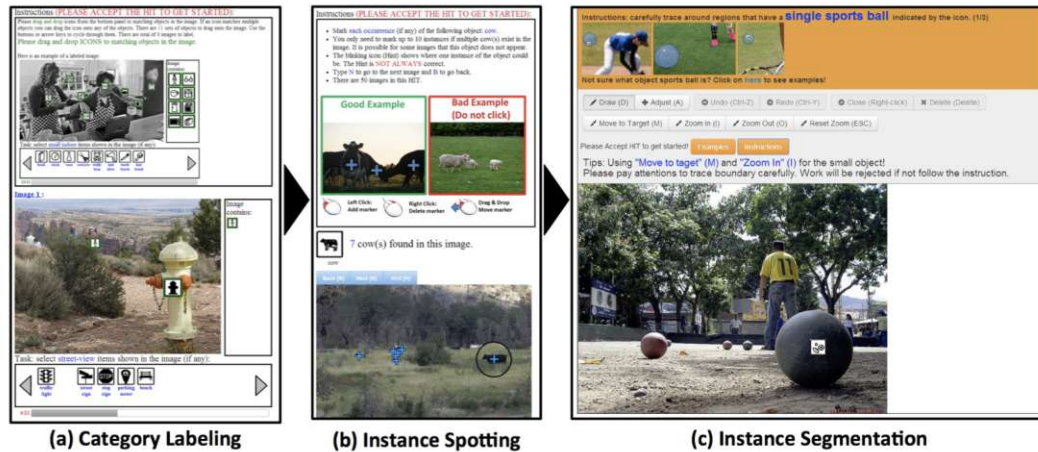


Figure 1.19: Image annotation user interface for the COCO dataset [50]

The previous section provided a detailed overview of the technologies that enable automated technique analysis. This foundation is essential for understanding how video data can be transformed into a form that allows for movement information analysis. However, while the technical capabilities of pose detection models are well documented, the question remains how the capabilities can be applied to real-world applications aimed at supporting athletes. The practical implementation and evaluation of such an application is guided by the following research questions.

### 1.4 State of the art

Existing research has leverages video technology to develop manual, semi-automated or fully-automated analysis systems. This chapter provides an insight into analysis based on 2D video, analysis based on 3D video, the competitive analysis of 2D and 3D video analysis and finally the use of the OpenPose [32] pose detection model in sports analysis.

**2D Video Analysis in Sports:** DeFroda et al. [56] utilized 2D video to analyze the pitching motion of youth baseball players. Their study compared the advantages and limitations of 2D and 3D video analysis, highlighting issues with analysis based on 2D videos. The study concludes that 2D video analysis has the limitations of suboptimal interrater/intrater reliability and the inability to assess kinetic variables.

Mostafa et al. [57] developed an innovative system for tracking football players using stitched together 2D videos from three digital cameras, This panoramic view of the game

and the object detection system YOLO v7 [58], enabled efficient player tracking on the field.

**3D Analysis in Combat Sports:** Rinaldi et al. [59] conducted an in-depth study of the karate punch technique of nine athletes recorded using eight infrared cameras (Vicon, Oxford, UK). Their analysis incorporated body markers that were placed on the athlete's bodies, as well as sensors in the floor measuring the ground reaction forces of front and back foot. Their study provided specific values for the flexion to be assumed at elbow and knees level, the range of motion of upper and lower arm as well as the maximum trunk rotational angular acceleration needed for an efficient punch technique.

**Comparative Studies of 2D and 3D Analysis:** Remedios et al. [60] analyzed 20 participants performing a functional lifting test, which consisted of lifting and lowering a box. The test was recorded with a 2D video camera and a 3D motion capture system (Vicon Nexus 2.6, Vicon, Oxford, UK). The same system that was used to provide the 3D analysis data used in this thesis. For pose estimation, they employed commercially available software (wrnch.ai, Montreal, Canada). The agreement between the 3D and 2D analysis was evaluated using Bland-Altman plots [21].

Peebles et al. [61] conducted a study on the running performance of 20 participants, employing both 2D and 3D video analysis techniques. The performers were filmed from a side angle running on a treadmill with markers strategically placed on various positions of their legs. Automated marker tracking systems were used instead of relying on a pose estimation model. The agreement of the 3D and 2D analysis was evaluated using Bland-Altman plots and intraclass correlation coefficients.

Schurr et al. [62] analyzed 26 adults performing single leg squats. They were recorded using three 2D video cameras and a 3D motion capture system. Their analysis focused on joint displacements at the trunk, hip, knee, and ankle in both the frontal and sagittal planes during the exercise. For the 2D analysis, still images taken at two keypoints were evaluated: peak knee flexion and neutral standing position. These images were captured in both the frontal and sagittal planes. To evaluate the agreement between the 3D and 2D analyses, the researchers employed Bland-Altman plots.

Chen et al. [63] analyzed the topspin stroke of various table tennis players. The video recording was done using a Codamotion infrared capturing system (Charnwood Dynamics Ltd, Leicestershire, UK). In addition to the camera setup, sensors were placed on the athletes themselves. The participants were categorized into groups based on their skill levels. The study's findings suggest a correlation between larger muscle strength under higher speed conditions and increased racket velocity during a forehand topspin stroke.

Maykut et al. [15] compared the use of 2D and 3D video in the context of analyzing the running kinematics of twenty-four cross-country runners. For analyzing the 2D videos the Dartfish Motion Analysis Software (Dartfish, Fribourg, Switzerland) was used. Three variables were considered for the analysis: (1) Peak contralateral pelvic drop angle, (2) peak hip adduction angle and (3) peak knee abduction angle. The study's findings

revealed a correlation between 2D and 3D analysis methods for one of the three examined variables: the peak hip adduction angle.

Pfister et al. [64] analyzed twenty adults jogging on a treadmill. The analysis was focused on comparing the results of 2D and 3D videos analysis. The 2D videos were recorded using the Kinect™ (Microsoft Corporation, Redmond, Washington, United States of America) sensor. The 2D videos recorded using the Kinect™ provided added depth information that is not available using common 2D digital cameras.

**OpenPose in Sports Analysis:** Liang et al. [65] conducted a study analyzing 250 videos of individuals using sign language. The research focused on examining hand movements to detect early-onset dementia. They used OpenPose’s hand-estimation algorithm for parts of the analysis.

Li et al. [66] analyzed the swing of ten university baseball players to automatically rate performance. The players were recorded from an side angle hitting a baseball and the videos were analyzed using OpenPose. To assess the quality of each swing, the researchers calculated various angles, including: shoulder, hip, arm and leg angles.

Jafarzadeh et al. [67] used OpenPose to evaluate a hurdle athlete that was recorded using three separate 2D video cameras. The automatically generated pose detection of OpenPose was evaluated against manually annotated keypoints of the athlete’s body.

The field of sports performance analysis is rapidly evolving, with researchers exploring the potential of cheaper and more accessible analysis using 2D video. The quality of this 2D analysis is frequently evaluated using the gold standard set using 3D video. And often pose detection models such as OpenPose provide the foundation for these automated analysis systems.

### 1.5 Research Questions

This thesis answers four interrelated research questions.

- **RQ<sub>1</sub>:** Which requirements can be identified for an application that provides automated analysis to athletes who want to improve their technique?
- **RQ<sub>2</sub>:** Is the application capable of generating an accurate, segmented model of the athlete’s performance from a 2D video?
- **RQ<sub>3</sub>:** Can the application accurately identify and track the predefined characteristics of the technique?
- **RQ<sub>4</sub>:** How does the performance of the 2D analysis compare to the 3D analysis? Are the variables extracted from the 2D video comparable to their counterparts in the 3D analysis data? And how does the quality of analysis compare between different 2D models?

The work in this thesis produces various artifacts in order to answer the research questions mentioned above. A requirements catalog is created that forms the basis for the application providing automated analysis. A set of mockups is created to guide the development of the application's user interface and also serve as the basis for discussion with the domain expert. An executable prototype capable of analyzing 2D videos is developed and documented. The analysis of the prototype is statistically evaluated against existing research results.

Depending on the research question, different methods are applied to answer them properly. Chapter 3 describes the methods used to answer each research question.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Methodology

This chapter outlines the research methodology and system development process of this thesis. It comprises four sections: Literature Review, Requirements Engineering, System Design and Implementation, and Statistical Evaluation.

The first section outlines the various areas covered as part of the literature review. The next section details the process that resulted in the final requirements list. This section also contains the definition for the characteristics, which are the foundation of the automated analysis. The following section describes the different prototyping methods used to create the final system. The final section explains the statistical methods used to evaluate the results of the automated analysis. This includes the specific threshold values defined for the various characteristics.

The figure 2.1 visualizes the process outlined above, displaying the resulting artifact for every applied scientific method as well as the research question that is answered by applying one of the scientific methods.

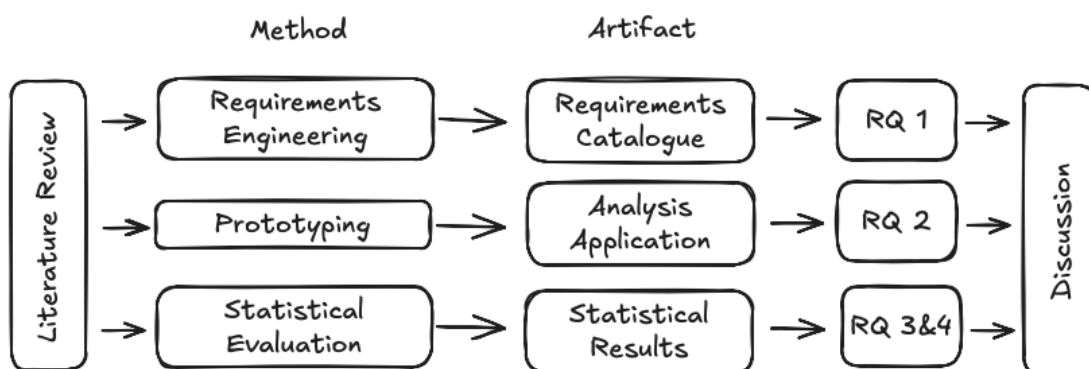


Figure 2.1: Phases of the work describing the methods and resulting artifacts.

### 2.1 Literature Review

The literature reviews provides the foundation for all aspects of this thesis. This section details the topics covered in the review.

**Double Side Kick:** The existing research on the analysis of the double side kick was evaluated, to understand the characteristics that provide the basis of the technique analysis. This understanding guided the identification of features that would need to be extracted from the input video in order to create an analysis.

**Automated Analysis:** The state of the art for automated analysis was researched to understand what frameworks are commonly used to process and analyze 2D videos. This research informed the search for technologies that would enable a accurate and performant analysis.

**Prototype Development:** A review of prototype creation and evaluation methods ensured scientific rigor in the thesis writing process and in the development of the resulting software artifact.

**Deployment:** The system created as part of this thesis requires specific hardware to run effectively. A server needs to have a GPU and in the case of OpenPose, specifically a Nvidia or AMD GPU [68]. There are options that enable the pose detection models to run on a server only containing a CPU, however the performance was deemed not good enough, to ensure an analysis that is fast enough so that it can be of use in day to day training [69]. Different deployment options were explored to find a solution that was able to provide the required performance while keeping the cost of running the system to a minimum.

### 2.2 Requirements Engineering

The requirements engineering process began with literature research and a qualitative analysis of the provided dataset. Semi-structured interviews [70] with experts in point fighting kickboxing and biomechanics were conducted and analyzed through qualitative content analysis [71] to extract essential requirements. Technical observations and expert feedback throughout the development process later informed adaptations and extensions to these requirements.

The foundation for motion analysis is the segmentation [72, 73] of the technique into distinct and measurable phases, with nodes that signify the transition between two phases. The literature review and domain expert interview provided phases and nodes for the double side kick. Those phases represented the foundation for analysis and were included in the iterative process of development.

## 2.3 System Design and Implementation

Based on the extracted requirements, a front- and back-end architecture was designed and an application was developed using the prototyping method [74]. This procedure allowed for practical demonstration of relevant parts of the system early in the development process. Multiple iterations were performed, incorporating expert feedback and try-and-error processes, to improve user experience and measurement accuracy.

**Exploratory Prototyping:** Multiple low fidelity prototypes were created in the process of creating the system. These prototype were presented to domain expert and the feedback that resulted was used to guide further development and the automated analysis functionality. Early prototypes only processed the 2D videos and output the extracted characteristics. Final prototypes provided a web interface and returned a video annotated with the analysis results.

**Evolutionary System Development:** A first version of the system that fulfilled all the predefined requirements was presented to domain experts, and the resulting feedback was used to make final adjustments and extensions to the system. This final iterative approach allowed for the refinement of the analysis, the improvement of user interfaces, and the addition of previously unforeseen but essential functionalities, resulting in a more useful and user-centric final product.

**Experimental Prototyping:** Different technologies and implementations were explored during the development process. These were evaluated on the basis of system performance and how close the resulting analysis was to the gold standard established by existing research. The areas explored include the use of different pose detection models, the deployment of the system to a variety of cloud providers, and evaluating the performance of different anomaly detection algorithms.

## 2.4 Statistical Evaluation

In a final step, the extracted data (from the 2D analysis) were statistically compared to the preexisting data (from Vicon 3D motion capturing). Statistical analysis and plot generation was performed using the Python programming language, Python Software Foundation. Python Language Reference, version 3.9). The differences between 3D and 2D variables were checked for normal distribution using Shapiro-Wilk tests. Similarly to previous studies exploring the agreement between 3D and 2D measurements in kinematic analysis [61, 62], Bland-Altman plots [21] were calculated for each of the dependent variables, in order to evaluate the agreement between the 3D and 2D analysis.

The y axis is constructed using the mean difference, because the x displays a small concentration range [75]. For 95%, limits of agreement were used.

For angle characteristics, the acceptable limits were set a priori at 20°. This decision was made by consulting the results of similar studies in which a subset of the results exceeds 20° [62] or where an average of 20° agreement [60] was reported.

## 2. METHODOLOGY

---

For the duration and distance characteristics, no comparable studies were found; therefore, it has been decided to calculate an equivalent of the 20° limits of agreement taken from [60]. It was decided to consider the minimum and maximum values for each of the individual characteristics and choose the 25 % value of the difference between minimum and maximum as the acceptable limit for that characteristic. This choice was made to ensure that in the case that the 3D analysis measures a performer to be either in the top 25 % or bottom 25 % of a characteristic, the 2D analysis agrees with the 3D analysis in so far that it puts the performer into the top half or bottom half for that characteristic, respectively. This is defined as the minimum viable case for a characteristic to be considered in the 2D analysis.

This means that for the duration variables the limits are for CH1 (0.162s), KI1 (0.05s), CH2 (0.19s), KI2 (0.108s).

And for the distance variables: For KHK1 (22), KI1 (28.5), KHK2 (16.5), DKS1 (15.5), DKS2 (20)

# CHAPTER 3

## Results

This chapter presents the artifacts created in the course of this thesis. The first section describes the iterative process that was used during the development of the system. This includes descriptions of the various iterations of prototypes that led to the final system, as well as the requirements and mockups that resulted from the iterative development. This is followed by a section that gives an overview of the final system's functionalities. This section focuses on the user interface of the system and how the analysis results are presented to the user. The next section presents a description of the system that focuses on the architecture and technical implementation. This includes the application details, as well as the deployment specifics. The final section presents the statistical evaluation of the system's automated analysis.

### 3.1 Iterative Development

This section details the different types of iterative processes used to create this thesis. The processes themselves are described, as well as the artifacts that resulted from them.

The overall development of the system followed these chronological stages:

- (a) Initial requirements gathered.
- (b) Movement variables and analysis characteristics defined.
- (c) Dataset prepared and standardized to single attempt videos.
- (d) Initial design mockups created.
- (e) Exploratory prototypes implemented and evaluated.
- (f) Final requirements refined based on prototype results.

- (g) Final mockups created.
- (h) Final prototype produced.

Each of the following subsections corresponds to one or more of these stages.

The first part describes the general requirements gathering process and presents the final requirements catalog. Then the phases and variables of the double side kick that represent the foundation for analysis are presented. The next part details the different stages of the prototyping process. The final part presents the different versions of the mockups that guided the development process.

#### 3.1.1 Requirements Gathering

- (a) *Initial requirements gathered.*
- (f) *Final requirements refined based on prototype results.*

The initial requirements were gathered by conducting semi-structured interviews with a domain expert, reviewing the existing research on the 3D analysis and doing an initial review of the state of the art for pose detection models. This process resulted in the initial requirement list presented in 3.1. This initial requirement list was extended over the course of the development process mainly through interviews with a domain expert, feedback from a domain expert after presenting prototypes of the system, and incorporating insights discovered over the course of development. The final requirements list 3.2 can be grouped into two categories. One is the group of requirements for the system. These include functionalities for the user to interact with the system and provide input of themselves performing the technique. The second group consists of requirements for the analysis. They define how the analysis results shall be presented to the user.

ID	Description
$C_1$	Users are able to upload a video of themselves performing the technique
$C_2$	The analysis extracts the same characteristics as the 3D analysis
$C_3$	The results of the analysis are presented to the user
$C_4$	The system is deployed as a web application
$C_5$	The system should be easily extended to other techniques
$C_6$	The analysis results should be explained to the user

Table 3.1: Initial requirements

#### 3.1.2 Analysis Characteristics

- (b) *Movement variables and analysis characteristics defined.*

The analysis is based on measurable characteristics of the double side kick. The characteristics are defined using literature research and domain expert interviews.

**Phases and nodes:** The double side kick is categorized into six functional phases and seven nodes as defined by Hölbling et al. [9]. The nodes are defined as the exact moments when one phase transitions to another; see Table 3.3.

**Variable definition:** After the definition and segmentation of the movement, the performance indicators had to be extracted and calculated at the nodes or within the phases, as described in [9]. They are grouped into the following four categories: (a) the relative duration of the functional phases, [9] (b) the relative position of relevant body parts (kicking legs knee height and distance between knee and front shoulder) , (c) the accumulated angles between the legs at the time of a node, [12] and (d) the velocity of body parts during a phase [12], see Table 3.4.

**Phase detection and motion analysis algorithms:** Based on the phase, node and performance indicator definitions, an advanced algorithm was developed, which solely relied on some anthropometric data and was able to automatically separate the phases and extract the variables. In an iterative process, the algorithm was then improved.

### 3.1.3 Data Preparation

*(c) Dataset prepared and standardized to single attempt videos.*

The provided dataset was transformed into a suitable form for initial analysis. This involved cutting the videos from their initial state, in which multiple attempts of the technique were depicted in the same video, to a version where every video only contains a single performance of the technique. This single performance corresponded to the attempt that was analyzed in the 3D analysis. Video enhancement was applied to individual videos that were too low in quality to be processed by the pose detection models. The enhancement was limited to adjusting the brightness and contrast of the existing video. The videos were annotated with the height of the athlete and the kicking leg used in the depicted performance. Throughout the development process, specific 2D videos were removed from the dataset due to not being able to be processed by either of the two pose detection models, even after enhancement.

### 3.1.4 Exploratory Prototyping

*(e) Exploratory prototypes implemented and evaluated.*

*(h) Final prototype produced.*

This section provides an overview of the different types of prototypes that were created over the course of development.

**Pose Detection Model:** Initial prototypes were created to determine which pose detection model to use as part of the system analysis. These prototypes processed the 2D videos from the provided dataset and output the detected pose for every individual image of the video. These initial systems were made up of prototypes using the pose detection models: OpenPose [10], PoseNet [76], and detectron2 [77] respectively. OpenPose was

### 3. RESULTS

picked as the pose detection model of choice. This selection stemmed from OpenPose's use in multiple publications [14], its performance [69], and its open-source nature [10] opening up the possibility of modifying the pose detection model to fit the system's needs.

In the final version of the system the MoveNet pose detection model was added and the user was provided with the option to pick which of the two models should be used during analysis.

**System Architecture:** The initial architecture of the system 3.1 was designed with a single frontend and two separate backend services. One service was written in C# [78] and responsible for communicating with the frontend, storing the uploaded videos and triggering the analysis. This service was always running and did not require a GPU to operate.

The second service was written in Python [79] and was responsible for creating the analysis and storing the analysis results. This service was a short-lived service, that was started when an analysis needed to be processed and shut down immediately after the analysis results were persisted.

This was done because of the large hardware requirements of the OpenPose system [80] and its reliance on a GPU to provide an adequate performance [69]. GPU resources are expensive [81] and by limiting the amount of time the service is running costs could be held low.

This approach was abandoned however, because the start-up of the short-lived service extended the total time of analysis to such an extent that it made it impractical for use in day to day training.

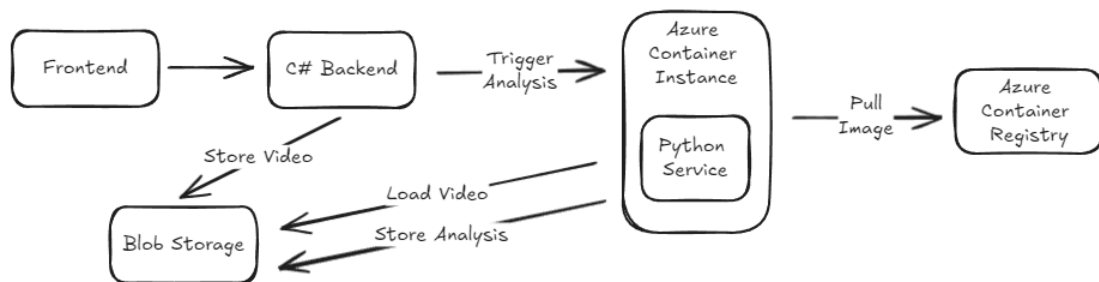


Figure 3.1: Initial architecture design

**Deployment:** The initial architecture of the system 3.1 was deployed inside of Microsoft's Azure cloud environment [82] and used Azure resources for storing data (Azure blob storage) and orchestrating the analysis (Azure container instances).

The system deployed the Python service using short-lived Docker containers [83]. Upon receiving an analysis request from the frontend, the C# service initiates the process by creating a Docker container for the Python service using Azure Container Instances and

Azure Container Registry resources [84]. The C# service triggers the container creation and sets an environment variable containing the key for the video in blob storage that requires analysis. The system then pulls the Python service image from the registry.

This approach enables efficient, on-demand processing while minimizing resource usage through short-lived containers. It was abandoned however because of performance reasons.

### 3.1.5 Design Mockups

(d) *Initial design mockups created.*

(g) *Final mockups created.*

This section presents the design mockups for the screens displayed to the athlete when interacting with the system.

**Upload Screen:** The upload screen, as shown in 3.2, serves as the users entry point to the system. It contains an interface that enables users to upload a video of themselves performing the technique and input the metadata needed for analysis. Additionally, the screen provides comprehensive instructions for both executing the technique correctly and recording the video properly.

Figure 3.2: Mockup: Video upload screen

**Analysis Screen:** The analysis screen, as shown in 3.3, presents the user with the results of their technique evaluation. The annotated video is displayed in the center, with the exact results presented in a table beside the video. A link is included below the

### 3. RESULTS

video that leads to an explanation of the different characteristics and a description of how the analysis results can be interpreted.

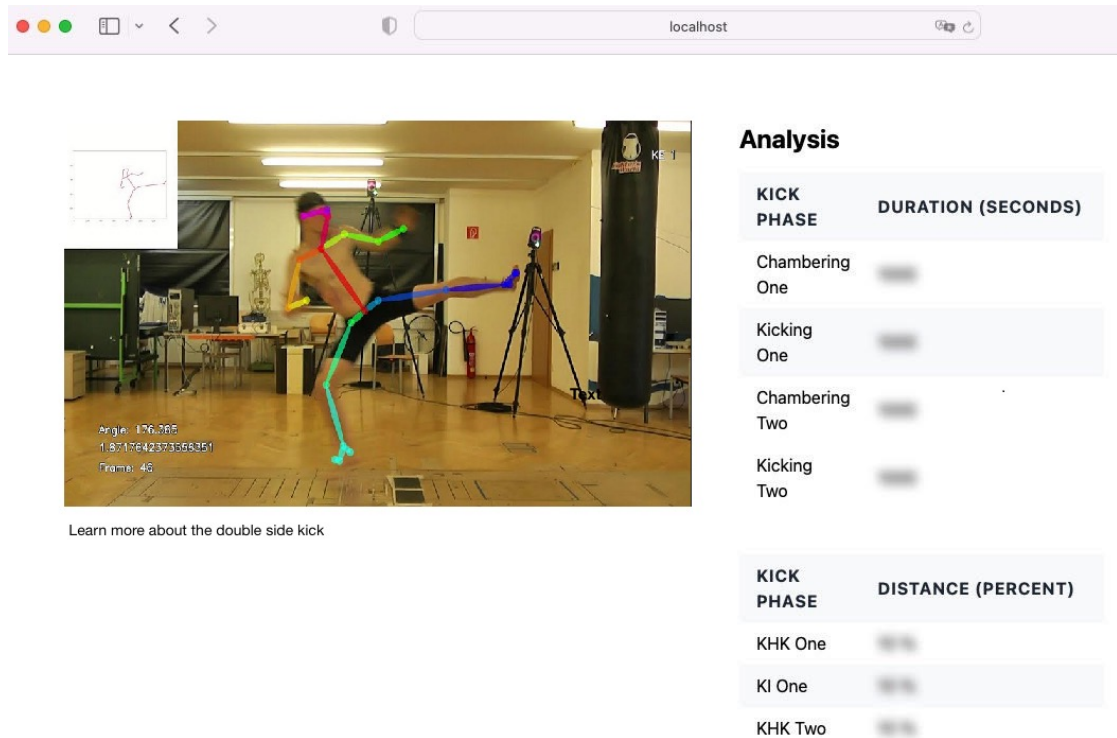


Figure 3.3: Mockup: Analysis results screen

**Recording instructions:** The recording instructions screen, as shown in 3.4, is accessible from the upload screen. The screen offers detailed step-by-step instructions on performing the technique correctly. It also provides guidance on setting up an optimal recording environment, including lighting and space considerations. Additionally, athletes can find a list of necessary equipment for effective video recording.

## 3.2 Functional Description

This section gives a functional overview of the final system. The system is accessible as a web application, currently deployed in an Azure cloud environment. The individual sections also contains information about which of the requirements were fulfilled by the specific functionality.

### 3.2.1 Upload Screen

**Implemented requirements:**  $C_1, C_2, C_4$

The upload screen, as shown in 3.5, enables the user to upload a video of themselves performing the technique, as well as input metadata required for analysis. The function-

How tall are you in cm:

Which foot are you kicking with?

Left

**Start Analysis**

## How to Film Yourself Performing a Double Side Kick

1. Find an open, well-lit space where you can move around freely.
2. Position your camera so that it captures your entire body from head to toe.
3. Wear well-fitting clothes.
4. Make sure nobody else appears in the video
5. Make sure the recording starts with you already in the starting position of the technique and that the recording ends shortly after you have finished the technique.
6. Start the technique facing the camera and make sure in the recording you move towards the right of the screen.

Figure 3.4: Mockup: Recording instructions screen

ality was extended in comparison to the initial mockups. The user was given the ability to choose between two different pose detection models: OpenPose and MoveNet. In addition, an interface was provided that lets the user decide what technique they want to have analyzed. The analysis of these other techniques in the backend is subject to future work however. The recording instructions for the video were moved to a separate screen, which is accessible via a link on the upload screen.

### 3.2.2 Analysis Screen

Implemented requirements:  $C_5$ ,  $C_6$ ,  $C_7$

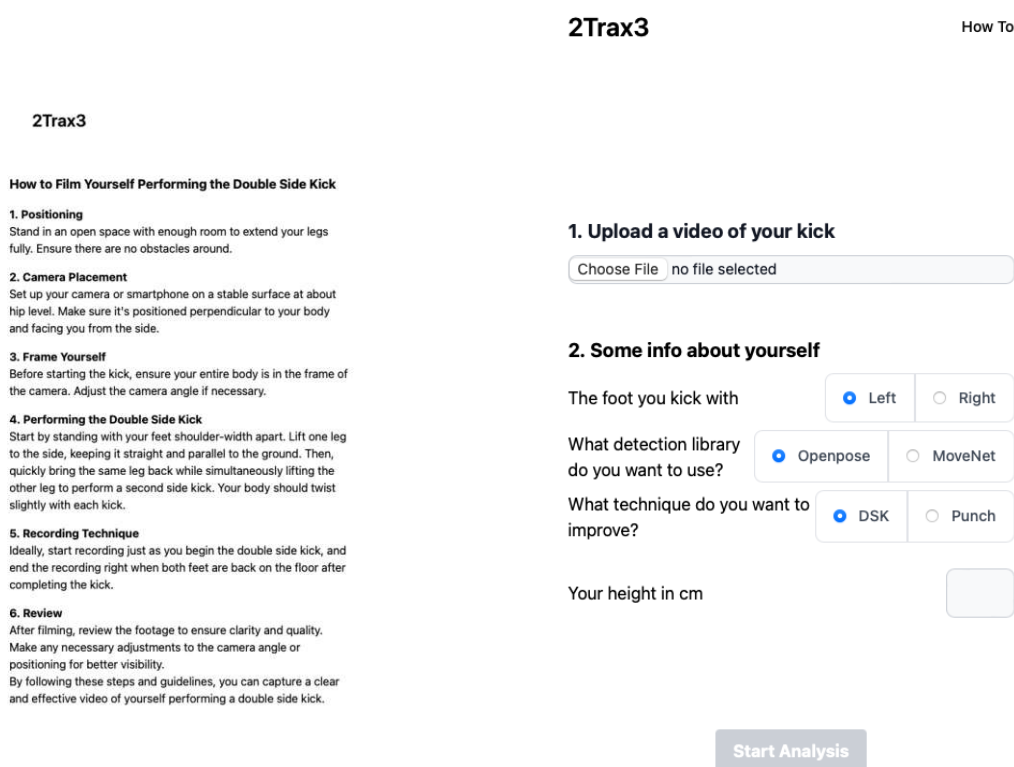


Figure 3.5: Recording instructions page (left) and upload screen page (right)

The analysis screen, as shown in 3.6, presents an annotated version of the user-uploaded video alongside a table displaying the analysis results. There are a few modifications made compared to the mockups. The analysis screen was extended to include a graphical timeline of the technique’s phases and nodes, offering users a quick overview of the analyzed performance. The structure of the table with the analysis results was changed to group the results into distinct categories, each having an icon linking to a detailed explanation of that category. This setup enables the user to receive a more focused explanation of the analysis results.

The annotated video has a slightly different look depending on the chosen pose detection model, however the conveyed information remains consistent. It displays the athlete’s pose with marked keypoints of the human physique connected by lines. The video briefly pauses at each detected node of the performance. In the upper left corner, a representation of the keypoints without the corresponding video frame provides a quick reference for the detected posture. These features collectively offer a comprehensive visual analysis of the user’s technique.

## 2Trax3

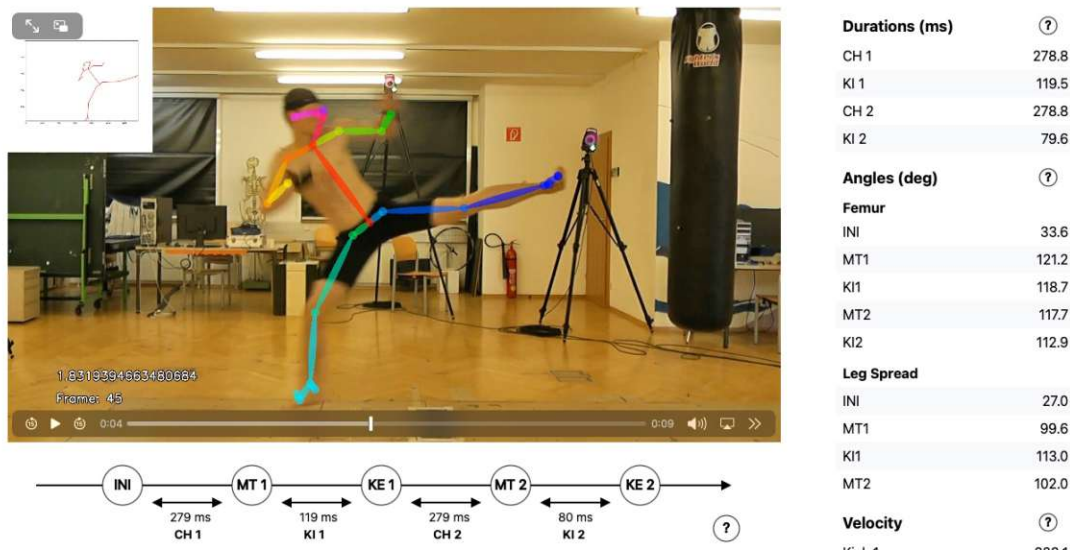


Figure 3.6: Analysis results screen

### 3.2.3 Recording instruction

#### Implemented requirements: $C_3$

The recording instructions screen, as shown in 3.5, provides the athlete with detailed instructions on how to record themselves performing the technique. Compared to the initial mockups, the instructions were moved to a dedicated screen. This change enhances user experience by allowing the user to solely focus on the recording instructions, while providing a cleaner design of the upload screen.

#### Annotated video: $C_2$

The annotated video serves as a visual representation of the analysis results. It is made up of the original video with keypoints of the user pose overlaid over each individual frame. At each of the five detected nodes of the performance, the video pauses briefly, achieved by duplicating the frame where the node is detected. Figure 3.7 displays an example of these paused moments. In the upper left corner of the video, a static representation of the keypoints appears without the corresponding video frame. This addition provides users with a quick reference to the model's output, allowing for easy comprehension of the detected pose independent of any other visual information.

## 3.3 Architecture and Technical Description

This section provides an overview of the system architecture and the technical details of the individual components making up the architecture. A high level overview of the

### 3. RESULTS

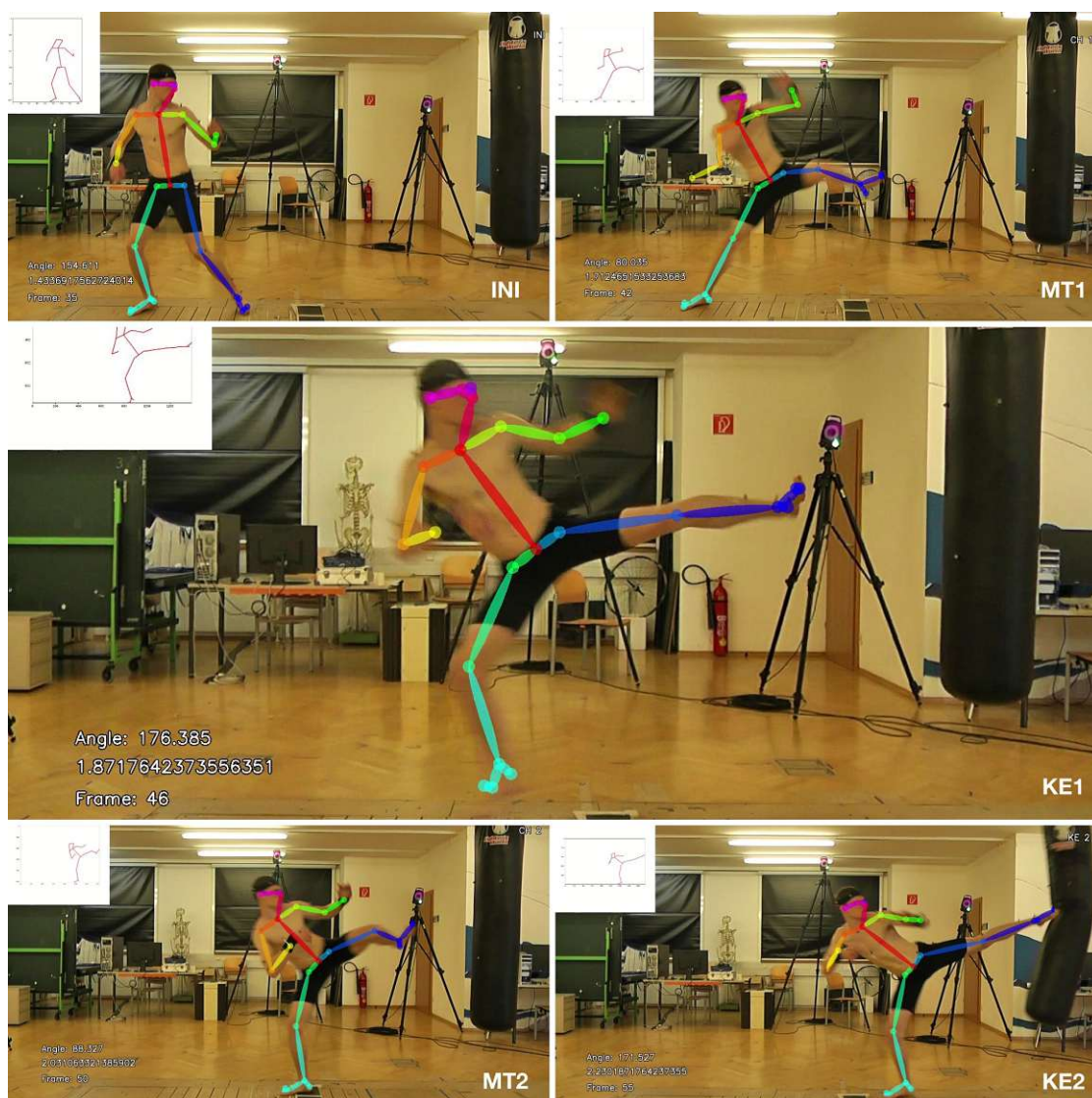


Figure 3.7: Annotated video of the performance

architecture is followed by the technical details of the backend, frontend and analysis. The section concludes with the deployment details of the system.

#### 3.3.1 Architecture

The architecture 3.8 consists of several key components: a frontend that provides the user interface for providing the 2D video input and receiving the analysis results, a backend that is responsible for communicating with the frontend and calculating the analysis results, blob storage for retaining original user input and analysis results, and an Azure container registry that is accessed during the deployment process. Detailed

descriptions of the frontend 3.3.2 and backend 3.3.3 are provided in separate sections.

**Blob Storage:** Blob or object storage is a storage approach where data is managed as individual objects or blobs. A blob is composed of user-accessible attributes, device-managed metadata and the data itself. The data is a collection of bytes of variable size that is treated as a single cohesive unit. This can be anything from a video, an image, or even an entire database [85].

The system in this thesis uses Azure Blob Storage [86] as its storage solution. There are different types of blobs that can be used to store data in Azure Blob Storage. Block blobs are used to store the videos and the analysis results. Block blobs are optimized for uploading large amounts of data efficiently. Each block blob consists of up to 50000 blocks of variable size. With each block having a maximum size of 4GB [86].

The backend is responsible for the communication with the blob storage. The backend communicates with the blob storage first by persisting the original user uploaded video. After the video is analyzed the analysis results comprised of the annotated version of the user uploaded video and the concrete analysis values are persisted in blob storage. The frontend sends the original video to the backend, and the backend returns a uniform resource identifier (URI) belonging to the blob that contains the analysis results. This URI is then sent in a consecutive call from the frontend to the backend to receive the analysis results. This approach enables the frontend to keep the URI in local storage and present the analysis results to the user even after a refresh of the website.

**Azure Container Registry:** The backend and frontend run as individual docker containers [87]. The images that provide the foundation for the backend and frontend containers are stored inside an azure container registry (ACR) instance. The registry is accessed during the initial start-up of the system. Two images are stored in the registry. One for the frontend and one for the backend. The details of the individual images are provided in the section 3.3.5.

### 3.3.2 Frontend

The frontend is a web application, written in Typescript using the Angular framework [88]. The frontend is structured as a single page application, when a user accesses the website, a compiled Javascript file, containing the frontend HTML, CSS and Javascript code, is returned from the server.

The frontend is structured as a collection of components. A component is made up of an HTML file that contains static content, a CSS file that provides the styling for the static content, and a Typescript file that provides interactivity and the ability to trigger communication with the backend.

There are in total four components: A component for the page where the user can upload their video. A component for the page where the analysis results are presented to the user. A component for the page where the recording and technique instructions are

provided to the user. And a component that represents the page containing the privacy information.

The components do not communicate with the backend directly. They access it over a separate singleton service that encapsulates the backend communication. There are three HTTP calls that are sent to the backend. A call that uploads the user uploaded video. For this call the backend returns a string that points to the location of the analysis results inside the blob storage. This string is stored in the local storage of the user's browser.

The string is used in two separate calls to the backend. One call returns the analysis results as a JSON file which contains the values of the various characteristics, making up the analysis of the technique. A second call returns the annotated version of the user uploaded video. This call is a HTTP range request [89]. This kind of request enables the frontend to ask the backend to only return parts of a specific resource. This enables the application to start playing the video without having to wait until the complete file is returned from the backend.

The project utilizes only Angular-specific packages, with the sole exception of the Tailwind CSS framework [90].

#### 3.3.3 Backend

The backend is a RESTful API, written in Python using the Flask framework [91]. The API provides three endpoints. One endpoint that receives the user uploaded video of the technique. This endpoint returns a URI that points to the analysis results in blob storage. And two endpoints that return the analysis results. One returns the analysis values as a JSON file, the other returns the annotated version of the user video.

The backend processes videos frame-by-frame using the OpenCV library [92]. Each frame undergoes pose detection using either OpenPose or MoveNet models, both of which return 18 keypoints of the detected person. These poses are then refined using a selected anomaly detection strategy, as detailed in the Analysis section 3.3.4. The resulting keypoint list is used to calculate the concrete characteristic values.

The calculated values are then used to create the annotated version of the video. The annotated version is created using OpenCV and PyPlot. Both the annotated video and analysis results are stored in blob storage and the URI is returned to the frontend.

When using MoveNet as the pose detection model, the analysis has to be started in a separate process, as the flask server is single threaded and the analysis is a blocking operation. The main thread checks if the analysis is finished by checking blob storage for the results, using the URI created at the beginning of the analysis process.

The analysis service is structured to allow modification of the pose detection model, anomaly detection strategy, and what technique is evaluated during analysis. This is achieved by encapsulating pose detection, anomaly detection, and technique-specific code into separate packages. Various runtime arguments can be provided to the analysis

service to control the different analysis parameters. This facilitates future iterations of the system to be extensible to analyze other techniques.

Access to the blob storage provided by security keys that are injected into the service using environment variables. The environment variables are set when building the backend container.

### 3.3.4 Analysis

The analysis is a two step-process, in the first step the extracted keypoints of the person undergo anomaly detection. In the second step the output of the detection is used as input for the analysis, which calculates values for various characteristics.

**Anomaly detection:** The anomaly detection uses predefined rules to identify and mark anomalies in the output of the pose detection models. These anomalies are then excluded from further calculation. The values are replaced using linear interpolation.

The input of the anomaly detection is a time series, created from either the x-coordinates (horizontal) or y-coordinates (vertical) of the kicking leg's ankle, knee or hip. These coordinates refer to the position in the 2D image being processed.

Two types of errors are identified in the output: (i) The body part cannot be detected in the frame, resulting in coordinates that are either missing or zero, (ii) the body part position is falsely detected and the coordinates are present but incorrect. The first type of error is handled by using the average between the first preceding non-zero coordinate and the next non-zero coordinate to interpolate the missing value. The second type of error is handled by a static rule set.

Each rule takes a consecutive sequence of coordinates as an input and returns a boolean indicating whether the sequence represents an anomaly or not. The rules check for differences in value between successive points in the sequence or they examine the monotony of the sequence.

An example for a rule is one that examines a sequence of three consecutive values as input. It checks if the second value is smaller than the first value, but that the third value is higher than both the first and the second value. To reduce false positives the difference between the first and second value must surpass a given threshold. For the threshold, the median difference in successive data points, calculated over the entire time series is used. If the differences all are higher than the threshold the rule marks the middle value in the sequence as an anomaly.

**Calculation characteristics:** The determination of knee angles begins with an initial estimation of the missing depth coordinates, which are not available in the 2D analysis compared to the output of the 3D analysis. This estimation involves calculating the distances between the knee and hip, as well as the knee and ankle of the kicking leg. These distances are then sorted in ascending order, and the upper value corresponding to the 95th percentile of each set is selected and for further calculations considered as

the "real" length of the knee and hip or knee and ankle. This estimation of the depth coordinates is repeated for each frame of the video analysis.

Next, a right triangle is constructed using the detected positions and the estimated depth coordinate. This allows the estimation of the depth coordinates. This allows for the generation of 3D vectors representing the line connecting the knee to the ankle, as well as the line connecting the knee to the hip. For the detection of the phases the angle between these 3D vectors is calculated to obtain the angle of the user leg at a specific frame of the video. These calculated angles undergo one more anomaly detection, utilizing a set of rules defined specifically for the angles.

The identified coordinates and calculated angles are used to detect the nodes that signify the beginning and conclusion of distinct technique phases. The INI node is determined by examining the y-coordinates of the kicking leg's angle, identifying the first monotonically increasing sequence of a predetermined length. For the MT1 node, the highest y-coordinate of the knee before the angle exceeds 110 degrees is detected. The KE1 node considers the first monotonically decreasing sequence of angles after the MT1 node. The same respective rules are applied to detect the MT2 and KE2 nodes.

The distance between the nose and ankle is used for computing velocity and distance characteristics. The upper value of the 95th percentile is selected to set the true height of the performer, as provided by the user, and map it to the corresponding coordinates in the video. Thereby converting the user input into pixel values. This enables an accurate estimation and calculation of the performer's velocity and distance metrics.

### 3.3.5 Deployment

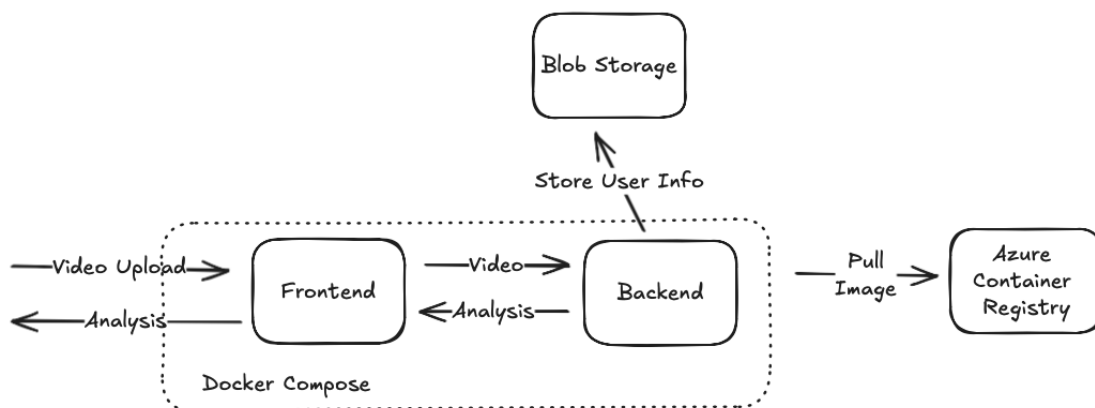


Figure 3.8: High-level architecture

The entire system is deployed inside an Azure Virtual Machine. Upon VM startup, two Docker containers—one for the frontend and one for the backend—are initiated using a docker-compose file. The frontend container is publicly accessible, while the backend container is only reachable by the frontend container within the Docker network.

There are three docker images that are pulled from the Azure Container Registry.

**OpenPose Compilation Image:** This image compiles OpenPose from source code. It uses `nvidia/cuda:11.6.2-cudnn8-devel-ubuntu18.04` as its base image. The image installs required dependencies, downloads OpenPose source code, and compiles it.

**Backend Image:** The OpenPose compilation image is the base for this image. This image first upgrades the Python version of the OpenPose compilation image to 3.11. This is done in order to accommodate Tensorflow and MoveNet which require a Python version 3.11 or greater. Then the required packages are installed and all Python source files are copied into the container. Finally, a Gunicorn webserver is started that hosts the Flask application.

**Frontend Image:** This image employs a two-step process. The build step uses `node:21.7.3-alpine3.18` as its base image. It installs Angular and its dependencies, copy source code, and compile the Angular application. The run step utilizes `nginx:1.25.4-alpine` as its base image. It starts an nginx web server that hosts the compiled Angular application from the previous step.

The virtual machine is of the Azure NVadsA10v5-series [93]. It contains NVIDIA A10 GPUs. The system could be deployed on other cloud providers or an local server, due to the fact that the main services are containerized. The server requires a powerful enough GPU to enable the use of the pose detection models.

### 3. RESULTS

ID	Title	Description
C1	Upload videos	The system shall make it possible for the user to upload videos of themselves performing the technique. The format of the video is limited to either MP4 or AVI. The size of the video is limited to 10MB.
C2	Recording instructions	The system shall provide instructions for the user on how to record themselves performing the technique. After reading the instructions the user shall have everything they need to setup the recording environment and create a video that enables the system to perform an accurate analysis.
C3	Technique instructions	The system shall provide instructions for the user on how to perform the technique. The instructions shall detail the different steps of the technique and the characteristics that are evaluated during analysis.
C4	Uploading user information	The system shall enable the user to input metadata about themselves that is necessary for analysis. The metadata shall include the height of the user and the leg they use for the kick movement.
C5	Annotating uploaded video	The system shall return an annotated version of the user uploaded video. The annotation shall include the pose of the user and mark keypoints of the technique. This shall provide the user with a visual representation of the analysis, giving them an understanding of how well the system detected their performance in the video.
C6	Displaying analysis results	The system shall display the analysis of the technique. The analysis shall include the pre-defined characteristics, provided in a structured manner to the user.
C7	Explaining analysis results	The systems shall provide an explanation of the analysis. The explanation shall detail the different characteristics, what they mean, and how the values relate to the quality of the technique.
C8	Extensibility of the system	The system architecture should enable the system to be easily extendable to analyze other techniques while using the same pose detection models.
C9	Deployment of the system	The system shall be deployed as a web application and be accessible over the internet.

Table 3.2: Final requirements list

Nodes	Description
<b>Node 1</b> Initialisation (INI)	Defined as the moment when the kicking leg's foot loses contact with the ground.
<b>Phase 1</b> Chambering 1 (CH1)	Consists of flexion of the knee, as well as flexion and abduction of the hip joint of the kicking leg.
<b>Node 2</b> Time of Measurement 1 (MT1)	Defined as the highest elevation of the knee before the kicking leg's knee angle surpasses 110°.
<b>Phase 2</b> Kicking Phase 1 (KI1)	Mainly consists of extension of the knee and hip joint, with ankle flexion of the kicking leg.
<b>Node 3</b> Knee Extension Maximum 1 (KE1)	The first kick ends with the maximum extension of the kicking leg's knee.
<b>Phase 3</b> Chambering 2 (CH2)	Re-chambering is similarly defined as CH1 and describes the preparation for the second kick.
<b>Node 4</b> Time of Measurement 2 (MT2)	Similarly defined as MT1, but after the first kick.
<b>Phase 4</b> Kicking Phase 2 (KI2)	Second kicking phase, with the same definition as KI1.
<b>Node 5</b> Knee Extension Maximum 2 (KE2)	Maximum extension of the kicking leg's knee and hip, leading to target contact.

Table 3.3: Nodes and phases of the double side kick.

Table 3.4: Performance indicators.

<b>Durations</b>	Relative duration of each phase, normalized by the duration from the nodes INI to KE2
<b>Distances</b>	(i) The vertical height of the kicking leg's knee normalized by trochanter major height in straight stance (KHK). (ii) The relative distance between the kicking leg's knee and nearest shoulder normalized by their distance in neutral standing position. Both extracted at every node
<b>Vector spreading Angle (VSA)</b>	(i) The angle between the femur bones of both legs (vector connecting knee and hips' center of rotation). (ii) The angle between both legs (vector connecting hip and ankles' center of rotation)
<b>Velocity</b>	The mean velocity of the kicking leg's knee vertical elevation during CH1 and CH2. And the mean velocity of the kicking legs' foot in target direction during phase KI2

# Evaluation

Only 26 of the full 44 videos from the preliminary study were suitable (due to quality issues) for analysis.

The differences between 3D and 2D measurements were all prechecked for normal distribution by Shapiro-Wilk test using a significance level of  $\alpha = 0.05$ . The following two sections provide the Bland-Altman plots and average mean difference agreement for the analysis results using OpenPose and the results gathered using MoveNet respectively.

The Bland-Altman plots show the agreement between the 2D and 3D measurements [21]. In each plot, the x-axis represents the average of the two measurements, and the y-axis shows their difference. The central line indicates the mean difference, while the two dashed lines represent the limits of agreement, calculated as the mean difference  $\pm 1.96$  times the standard deviation.

The area between the limits of agreement contains approximately 95% of the differences if the data is normally distributed. A wide interval indicates a greater disagreement between the two measurement methods. The mean and limits of agreement are expressed in the same unit as the measured quantity, for example, degrees. The distribution of points along the x-axis can show whether the variability depends on the size of the measurement value, suggesting that the difference in agreement changes depending on the value of the measurement.

Differences were calculated as  $(2D - 3D)$ , meaning that positive values indicate higher measurements in the 2D analysis, while negative values indicate higher measurements in the 3D reference. Thus, a positive mean difference suggests that the 2D method consistently measured higher values. For the plots comparing OpenPose and MoveNet, differences were calculated as  $(\text{OpenPose} - \text{MoveNet})$ .

## 4.1 OpenPose

As illustrated in figure Figure 4.1, the Bland-Altman plots estimated the average mean difference agreement between the 3D and 2D analysis for the phase duration in CH1 (-0.04 s; LOA -0.15 to 0.06), KI1 (0.04 s; LOA -0.05 to 0.12), CH2 (-0.04 s; LOA -0.16 to 0.08), and KI2 (-0.01 s; LOA -0.13 to 0.12).

Agreement for the femur angles at the node MT1 (-0.59°; LOA -20.20 to 19.03), KE1 (4.92°; LOA -23.47 to 33.31), MT2 (4.13°; LOA -13.64 to 21.89), KE2 (-11.59°; LOA -43.94 to 20.77) illustrated in figure Figure 4.2.

Agreement for the leg spreading angles at the node MT1 (4.25°; LOA -14.47 to 22.97), KE1 (6.84°; LOA -17.63 to 31.31), MT2 (-0.36°; LOA -16.04 to 15.33), KE2 (-7.08°; LOA -40.09 to 25.92) illustrated in figure Figure 4.3.

Agreement for the relative knee height at KHK1 (-5.78; LOA -29.76 to 18.20), KI1 (-3.30; LOA -23.95 to 17.36), KHK2 (-2.04; LOA -32.06 to 27.99) illustrated in figure Figure 4.4.

Agreement for the relative shoulder knee distance DSK 1 (0.48; LOA -11.75 to 12.71), DSK 2 (1.41; LOA -21.46 to 24.27) illustrated in figure Figure 4.5.

Some observations from the plots are that for the phase durations, the difference increases as the average measurement becomes larger. For the leg spreading angles, the 2D measurements tend to decrease relative to the 3D reference as the average measurement increases. For the relative knee height, the differences for KI1 and KHK2 show an upward trend with increasing average values.

## 4.2 MoveNet

For the phase durations, the Bland-Altman plots estimated the average mean difference agreement in CH1 (-0.02 s; LOA -0.22 to 0.18), KI1 (0.05 s; LOA -0.03 to 0.12), CH2 (-0.01 s; LOA -0.15 to 0.14), and KI2 (0.02 s; LOA -0.16 to 0.20) Figure 4.6.

Agreement for the femur angles at the node MT1 (-14.86°; LOA -43.86 to 14.14), KE1 (-6.83°; LOA -37.55 to 23.89), MT2 (-11.03°; LOA -49.22 to 27.15), KE2 (-17.17°; LOA -62.19 to 27.85) illustrated in figure Figure 4.7.

Agreement for the leg spreading angles at the node MT1 (-9.34°; LOA -41.00 to 22.33), KE1 (-6.22°; LOA -36.82 to 24.39), MT2 (-10.98°; LOA -34.85 to 12.89), KE2 (-8.47°; LOA -42.13 to 25.19) illustrated in figure Figure 4.8.

Agreement for the relative knee height at KHK1 (-15.04; LOA -45.14 to 15.05), KI1 (-7.96; LOA -38.40 to 22.49), KHK2 (-10.26; LOA -41.54 to 21.01) illustrated in figure Figure 4.9.

Agreement for the relative shoulder knee distance DSK 1(1.96; LOA -9.68 to 13.59), DSK 2 (0.78; LOA -20.54 to 22.10) illustrated in figure Figure 4.10.

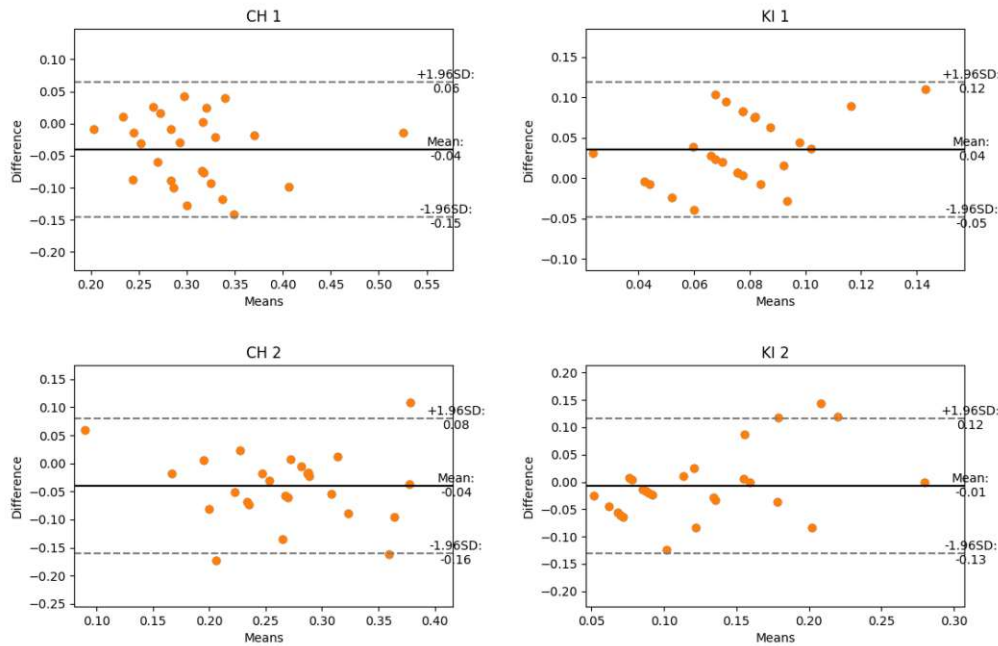


Figure 4.1: Bland-Altman plots for the four phases (OpenPose).

Some observations from the plots are that for the femur angles at MT2, the differences tend to increase as the average measurements become larger. For MT1, the variance in the differences appears higher for smaller average measurements. For the relative knee height, the differences for KI1 and KHK2 show an upward trend with increasing average values.

### 4.3 MoveNet and OpenPose

For the phase durations the Bland-Altman plots estimated the average mean difference agreement in CH1 (-0.02 s; LOA -0.21 to 0.17), KI1 (-0.01 s; LOA -0.11 to 0.09), CH2 (-0.03 s; LOA -0.20 to 0.14), and KI2 (-0.03 s; LOA -0.19 to 0.13) Figure 4.11.

Agreement for the femur angles at the node MT1 (-13.62°; LOA -20.45 to 47.70), KE1 (10.79°; LOA -23.56 to 45.14), MT2 (13.76°; LOA -18.81 to 46.33), KE2 (5.00°; LOA -35.97 to 45.98) illustrated in figure Figure 4.12

Agreement for the leg spreading angles at the node MT1 (13.18°; LOA -20.04 to 46.40), KE1 (13.02°; LOA -21.57 to 47.61), MT2 (9.95°; LOA -17.57 to 37.47), KE2 (2.20°; LOA -37.22 to 41.62) illustrated in figure Figure 4.13

Agreement for the relative knee height at KHK1 (8.83; LOA -29.85 to 47.51), KI1 (4.87; LOA -26.89 to 36.63), KHK2 (7.78; LOA -20.48 to 36.04) illustrated in figure Figure 4.14

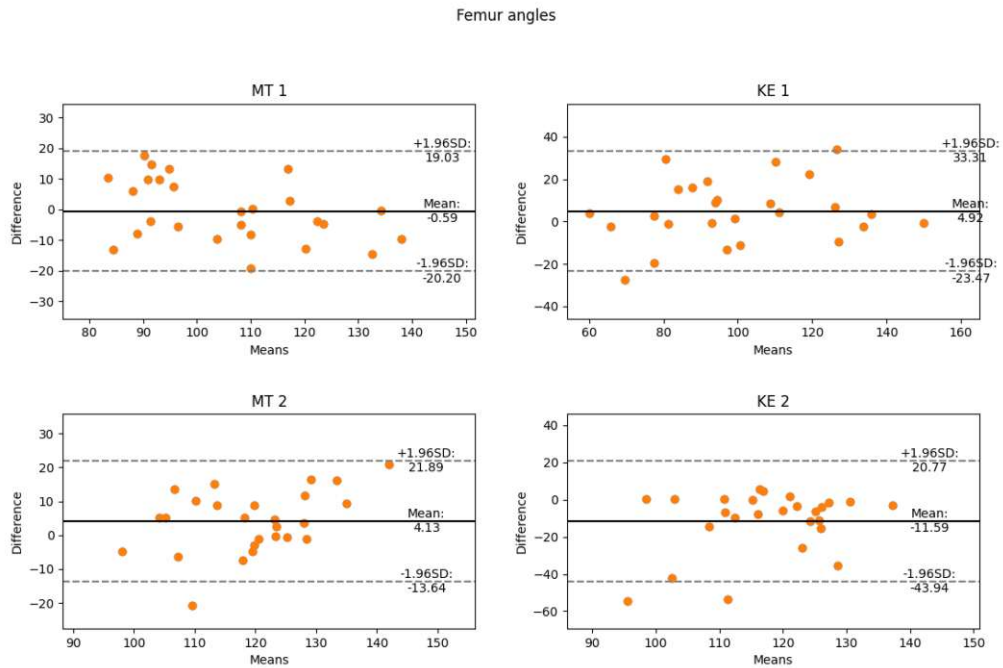


Figure 4.2: Bland-Altman plots for the femur angles (OpenPose).

Agreement for the relative shoulder knee distance DSK 1(-1.00; LOA -14.47 to 12.47), DSK 2 (1.22; LOA -14.97 to 17.40) illustrated in figure Figure 4.15

Some observations from the plots are that for the femur angles at MT1 and MT2, the differences tend to decrease as the average values become larger. The same pattern can be seen for the leg spreading angles. In addition, the variance appears higher for smaller average measurements, particularly in MT2 and KE2.

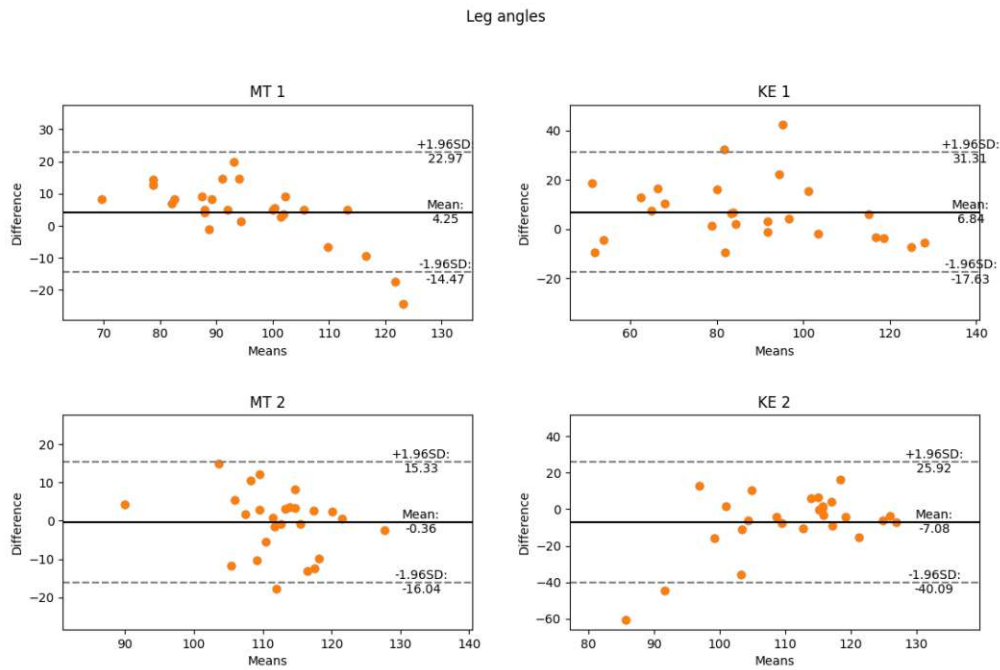


Figure 4.3: Bland-Altman plots for the leg spreading angles (OpenPose).

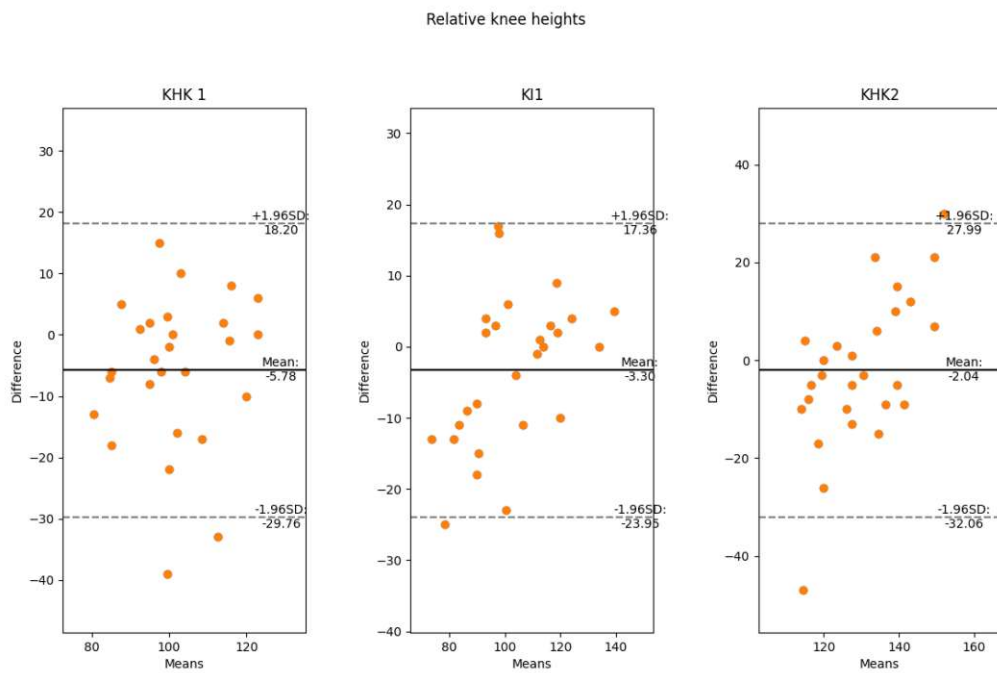


Figure 4.4: Bland-Altman plots for the relative knee height (OpenPose).

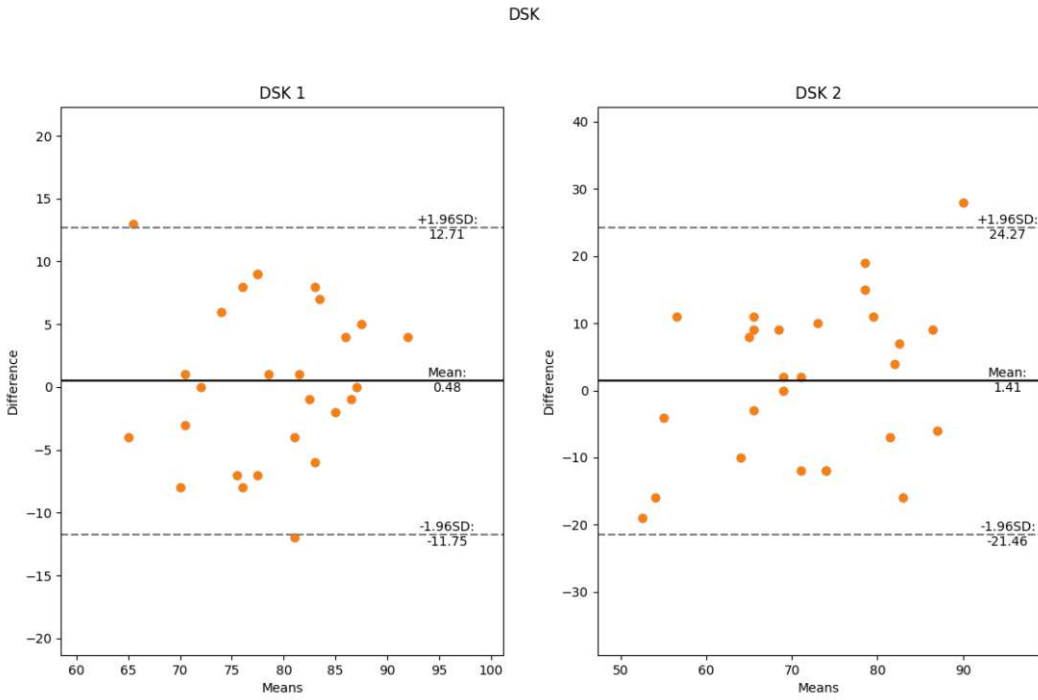


Figure 4.5: Bland-Altman plots for the shoulder knee distance (OpenPose).

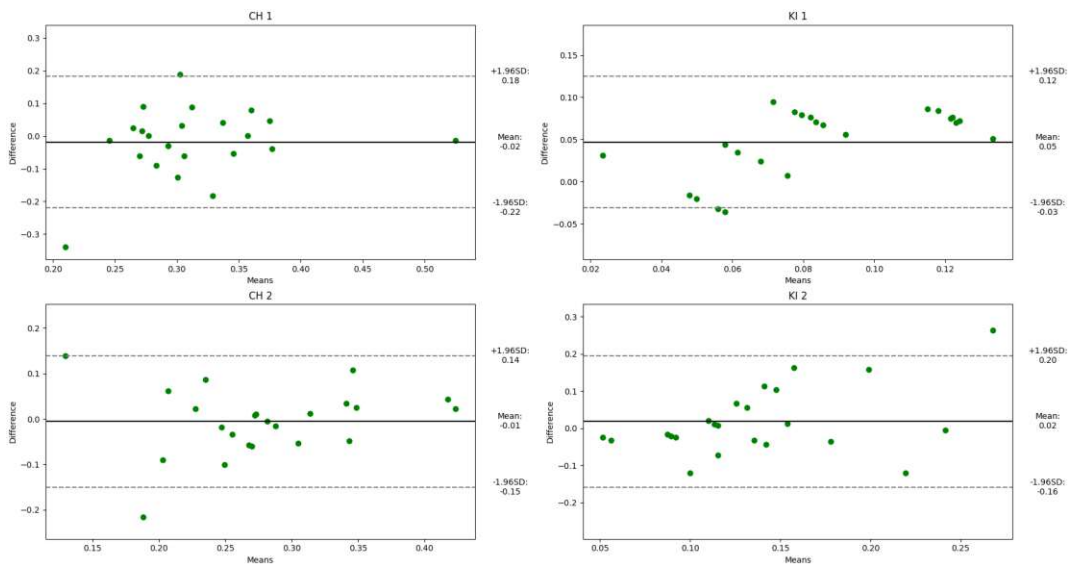


Figure 4.6: Bland-Altman plots for the four phases (MoveNet).

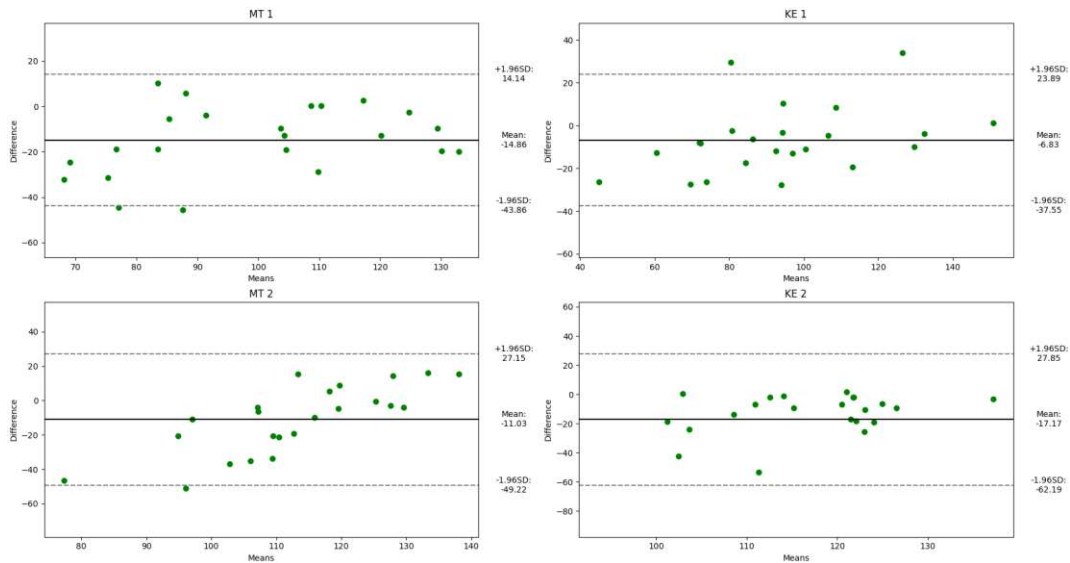


Figure 4.7: Bland-Altman plots for the femur angles (MoveNet).

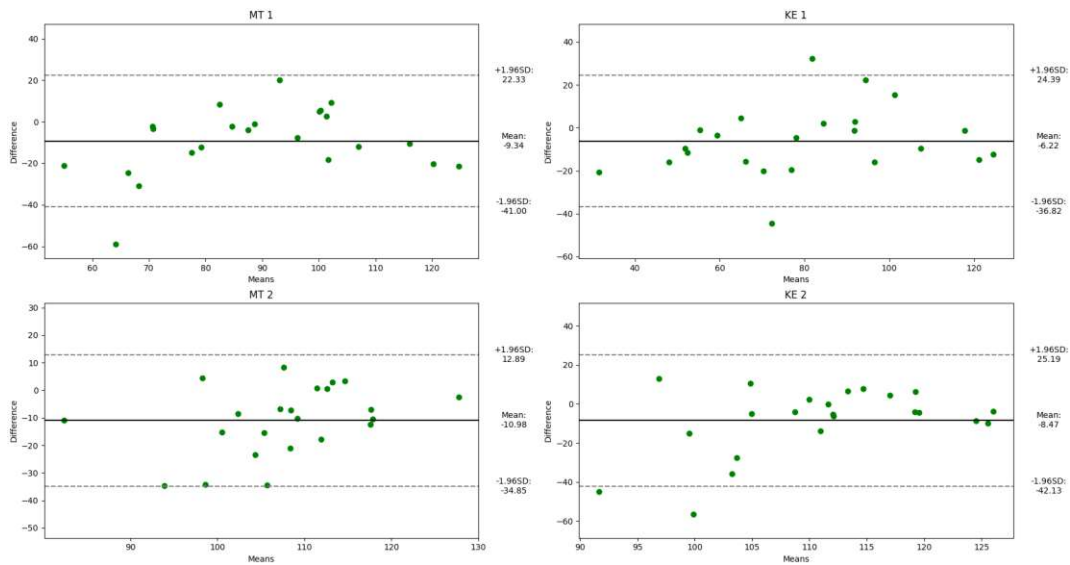


Figure 4.8: Bland-Altman plots for the leg spreading angles (MoveNet).

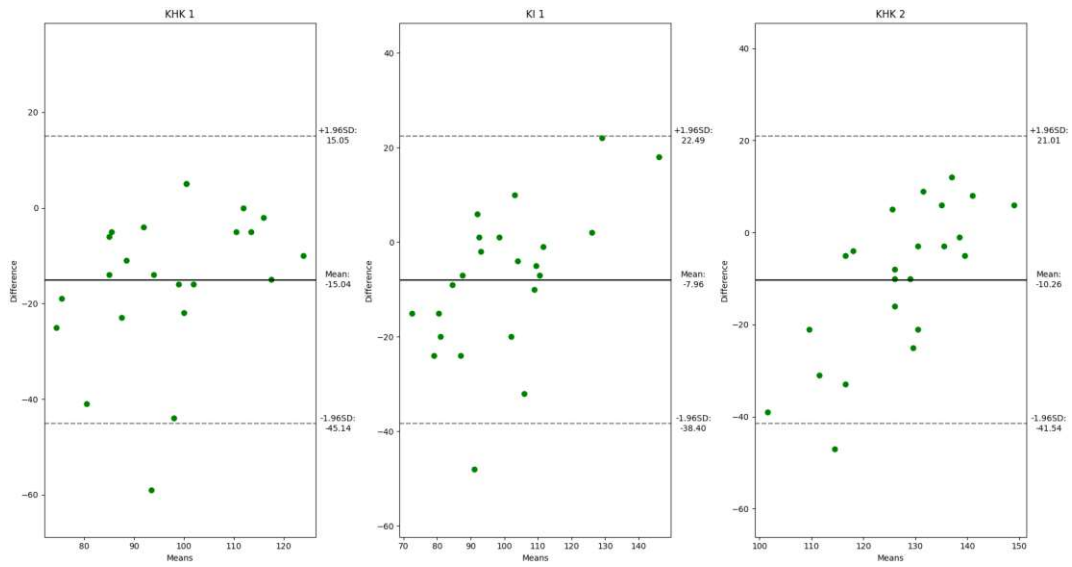


Figure 4.9: Bland-Altman plots for the relative knee height (MoveNet).

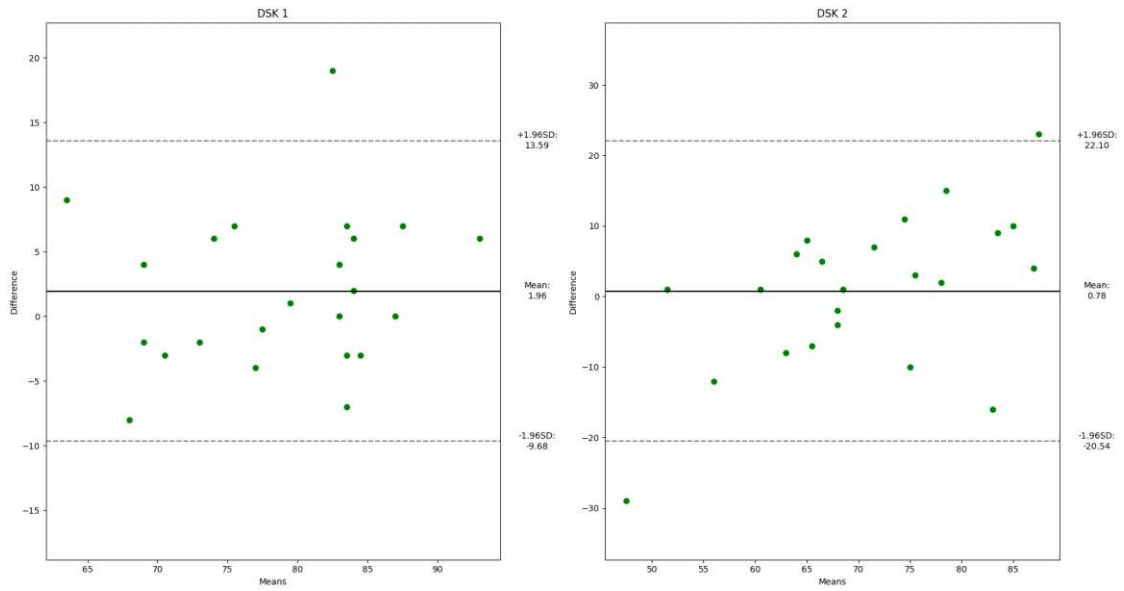


Figure 4.10: Bland-Altman plots for the shoulder knee distance (MoveNet).

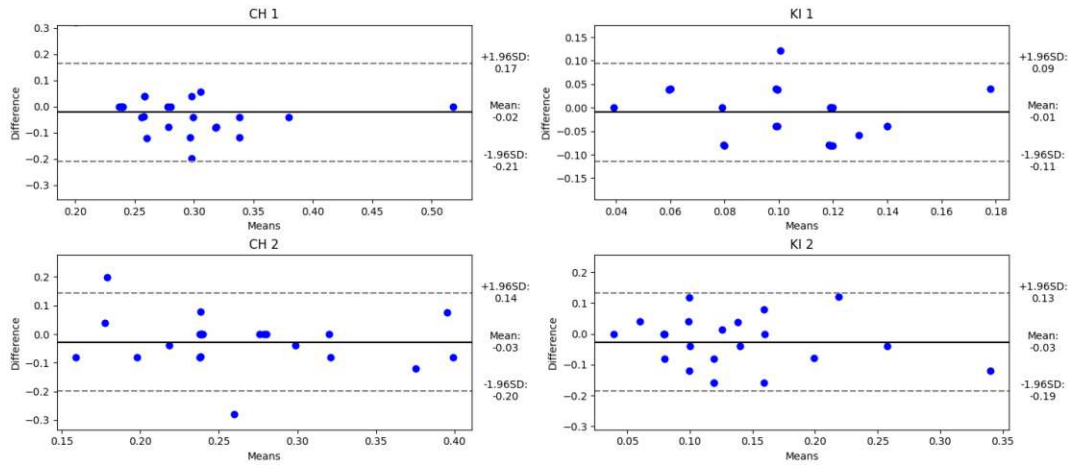


Figure 4.11: Bland-Altman plots for the four phases (MoveNet and OpenPose).

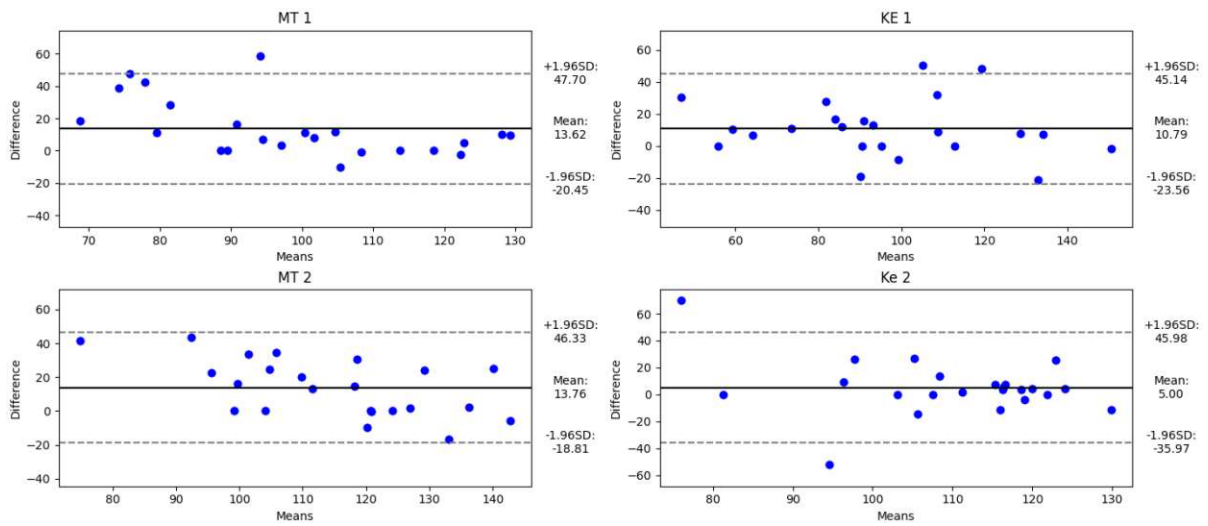


Figure 4.12: Bland-Altman plots for the femur angles (MoveNet and OpenPose).

## 4. EVALUATION

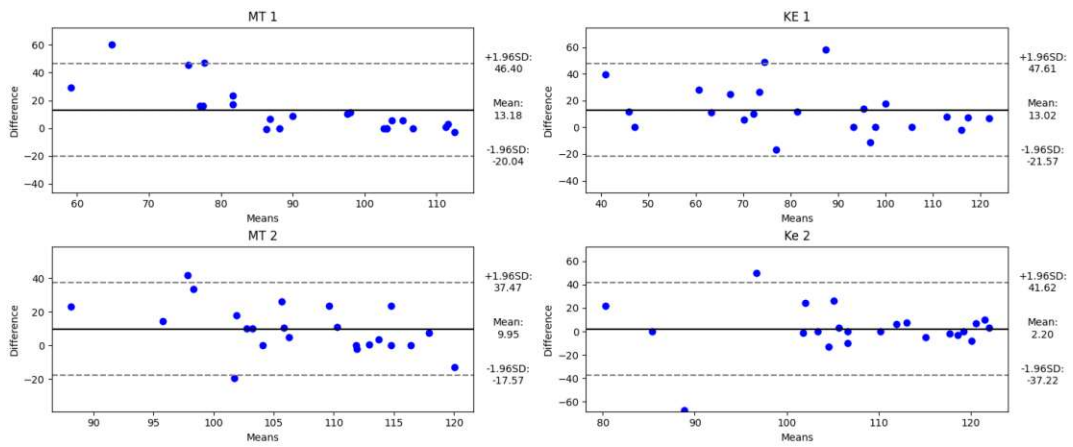


Figure 4.13: Bland-Altman plots for the leg spreading angles (MoveNet and OpenPose).

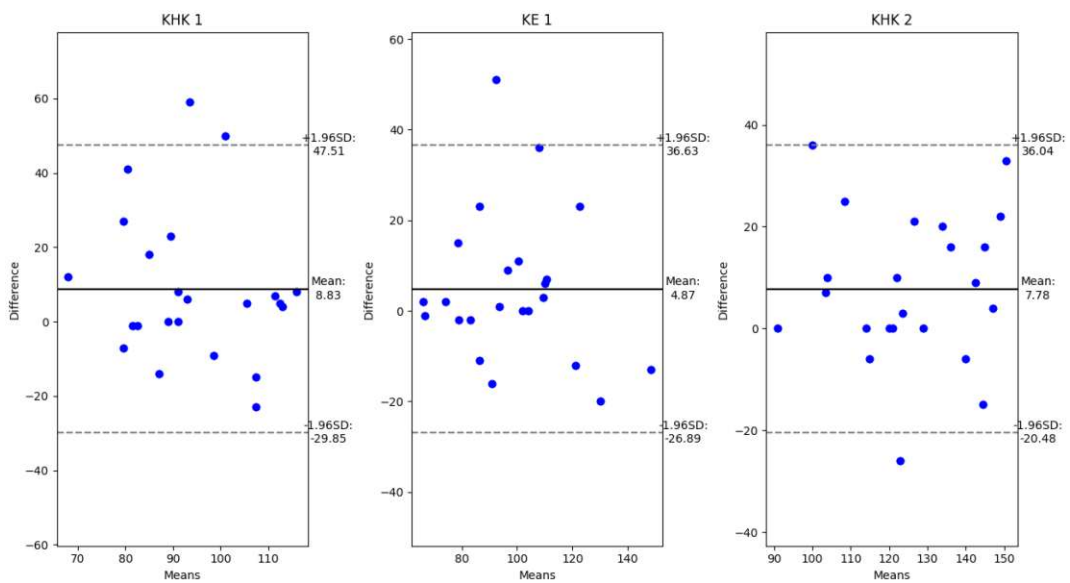


Figure 4.14: Bland-Altman plots for the relative knee height (MoveNet and OpenPose).

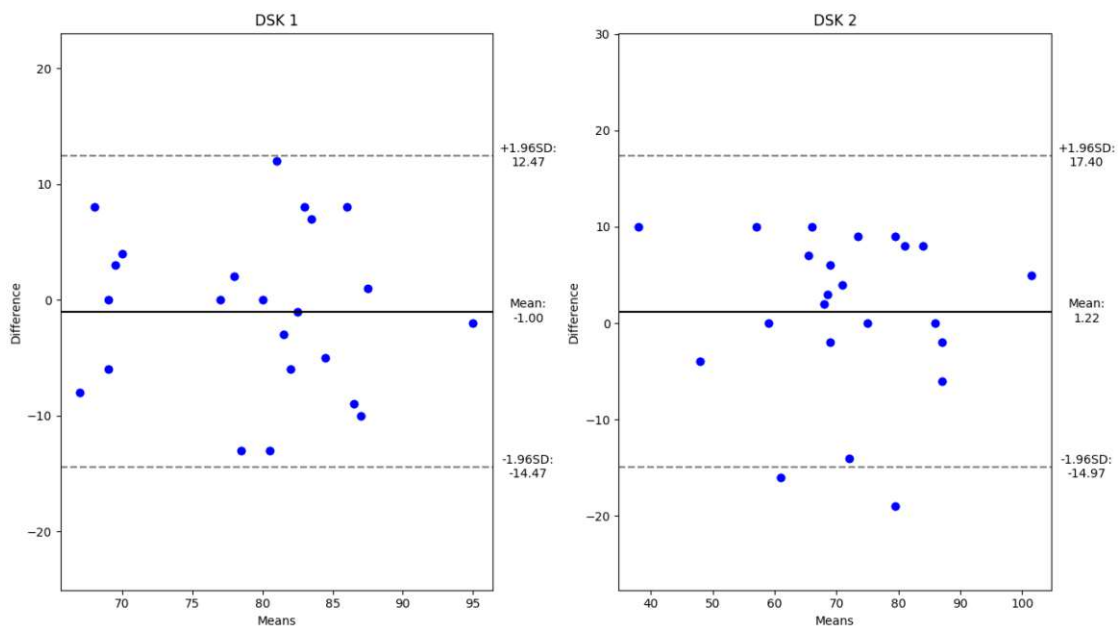


Figure 4.15: Bland-Altman plots for the shoulder knee distance (MoveNet and OpenPose).



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Discussion

In the course of this thesis a system was created that helps athletes in their day to day training by providing an automated analysis of their performance. The aim of developing the system was to answer four research questions, and the following sections provide answers based on all the results produced in the course of this thesis. The chapter concludes with an explanation of the limitations of this study.

## 5.1 Identified Requirements

**Research Question:** *Which requirements can be identified for an application that provides automated analysis to athletes wanting to improve their technique?*

The literature review, exploratory prototyping, and interviews with a domain expert led to general assumptions about the needs of the athletes using the system, as well as the system's own requirements. These assumptions were then specified as concrete requirements.

It was assumed that athletes would use the system as part of their daily training, where they most likely have access only to their mobile phones. This led to the requirement that the application be deployed as a web application, easily accessible via the internet, and capable of allowing athletes to upload videos of themselves performing the technique during training.

The development of various prototypes and analysis of the test data revealed the need for a specific camera setup, a certain quality of technique execution and additional information about the athlete to ensure optimal analysis results. This led to requirements for the system to provide recording and technique execution instructions, as well as the possibility for the athlete to provide additional information about themselves alongside the recording of the performance.

Literature research and domain expert feedback informed how the analysis results should be presented to the athlete. This resulted in the requirement that the analysis results are explained clearly, with an annotated version of the uploaded video provided to the athlete.

Finally, expert feedback led to the requirement that the system be easily extendable to support the analysis of other techniques.

## 5.2 Segmented Technique Performance

**Research Question:** *Is the application capable of generating an accurate, segmented model of the athlete's performance from a 2D video?*

Processing the training dataset revealed that not all videos could be analyzed due to quality issues of the recording, indicating that uploaded videos must meet a certain quality standard to ensure successful analysis. Even with the quality issues of the source material, which led to exclusion of 18 videos, the accuracy of the extracted data from the successfully processed videos is rather high.

The system faces unique challenges when compared to previous studies that report higher accuracy in 2D video kinematic analysis [62, 61]. In those studies, the performer remains in a fixed position within the video frame, either performing a single-leg squat or running on a treadmill. The main challenge of this study was the dynamic movement of the performer through the room.

Comparing the outputs of the two pose detection models on the test data, it seems that the MoveNet model is able to more consistently detect the correct pose from scenes where the performer is constantly moving. This can potentially be attributed to the MoveNet model being trained on an additional dataset that was created by sampling YouTube fitness videos [11]. However, the evaluation results show the OpenPose model performing slightly better in the final analysis. This could be due to the additional anomaly detection compensating for the OpenPose detection errors. Given the relatively recent release of the MoveNet model, this thesis represents one of the first works using the MoveNet model for the purposes of sports analysis. Further research is needed to determine if MoveNet is more suitable for analyzing videos depicting athletic activity.

All studies analyzing 2D videos face the challenge of missing depth parameters when compared to 3D analysis. There has been a study using the Kinect<sup>TM</sup> (Microsoft Corporation, Redmond, Washington, United States of America) 2D sensor for kinematic analysis [64], as the Kinect<sup>TM</sup> is capable of returning depth measurements. However, the results indicated that the measurements cannot reach the accuracy of a 3D motion capture system.

Recent research may help address the issue of missing depth information in 2D video analysis. Depth Pro is a recently released monocular depth estimation model [94]. Monocular depth estimation uses only a single image to obtain its depth parameters [95].

and Depth Pro does not need to be trained on domain-specific images—a feature known as zero-shot learning [94].

The figure 5.1 depicts a depth map generated from a frame of one of the videos in the dataset. This technology has the potential to be used in combination with the pose estimation models utilized in this thesis, to calculate more accurate body angle estimations by improving the accuracy of the depth estimations

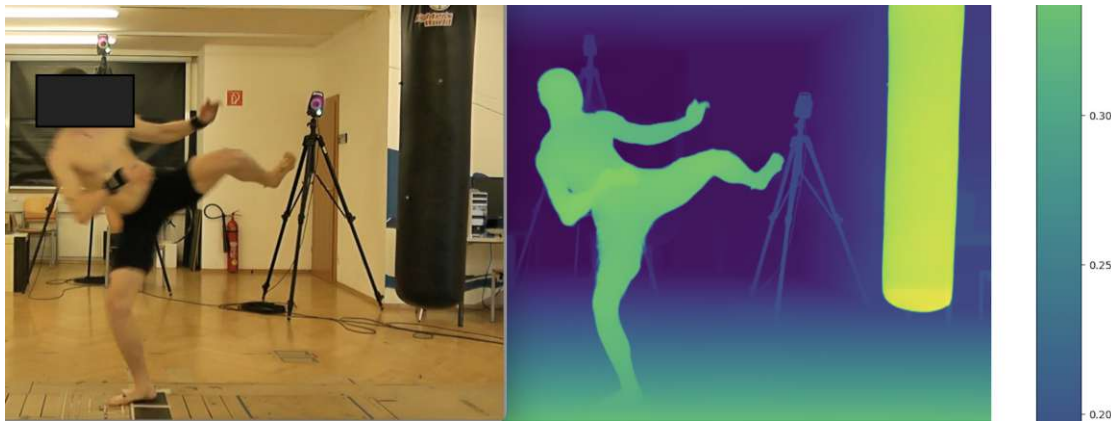


Figure 5.1: Depth map generated with Depth Pro

### 5.3 Identifying Characteristics

**Research Question:** *Can the application accurately identify and track the predefined characteristics of the technique?*

In the test data videos where the system successfully created a segmented model of the body, the system was quite adept at identifying the key moments of the technique, which are the nodes marking the transition between two phases. This is evident through the narrow limits of agreement for the phase durations, ranging from -0.22s to 0.20s for MoveNet and -0.16s to 0.12s for OpenPose across all phases. These results are also confirmed by manual review of the annotated analysis videos. Even for the chambering nodes where there is significant variation among athletes in terms of leg positioning and knee height during the preparation for the kick, the system is successful in identifying the correct frame.

However, the system struggled with characteristics related to leg angles during specific nodes. This is apparent in the broader limits of agreement observed for these characteristics. The limits of agreement range from  $-43.94^\circ$  to  $33.31^\circ$  for OpenPose and  $-62.19^\circ$  to  $27.85^\circ$  for MoveNet across all femur angles and ranging from  $-40.09^\circ$  to  $31.31^\circ$  for OpenPose and  $-42.13^\circ$  to  $25.19^\circ$  for MoveNet across all leg spreading angles. This makes sense as these characteristics are much more dependent on accurate depth values in order to calculate the angles correctly. As discussed in the previous section, recent

developments in depth estimation technologies may help address this limitation in future work.

Finally, characteristics based on the distance between body parts during specific nodes also showed reduced accuracy compared to the phase duration measures. This can again be attributed to the lack of depth information in 2D video as well as the need to know the athlete's height in order to calculate distance values.

## 5.4 Quality of Analysis:

**Research Question:** *How does the performance of the 2D analysis compare to the 3D analysis? Are the variables extracted from the 2D video comparable to their counterparts in the 3D analysis data? And how does the quality of analysis compare between different 2D models?*

Bland-Altman plots reveal no significant systematic bias in 2D measurements compared to 3D measurements for several key performance indicators. Based on the Bland-Altman plots for the OpenPose analysis results, the hypothesis can be accepted for performance indicators (i) Leg spreading vector at node MT2, (ii) phase duration of CH1 and CH2, (iii) relative shoulder distance DSK1 and relative knee height KI1. However, the hypothesis must be declined for the remaining distance, angle, and duration performance indicators. For the MoveNet analysis results, the hypothesis can be accepted for performance indicators (i) phase duration of CH2 and (ii) relative shoulder distance DSK1. Generally the 2D model analysis showed minimal average mean difference in calculation of phase durations and a higher mean difference and broader limits of agreement concerning joint angle and distance characteristics.

These results are comparable to findings from a similar study that compared motion data from 2D markerless pose estimation with that from a Vicon 3D motion capture system [60]. That study reported limits of agreement around 20° for knee flexion, shoulder flexion, and shoulder abduction. However, its focus was on participants performing a floor-to-waist functional lift, a considerably more constrained and stationary task than the double side kick examined in this thesis. In addition, the variety in how a functional lift is performed is much smaller than the variety in executions of the double side kick.

The study attributes one of the reasons for the lack of agreement between 2D and 3D data to the inaccurate estimation of joint centers by 2D pose estimation models. OpenPose and MoveNet estimate joint centers based on a general body outline. This can lead to inaccuracies that affect the calculation of joint angles and other spatial metrics [60].

Another study comparing 2D data collected using the Kinect™ sensor with Vicon 3D data of participants jogging on a treadmill reported a general underestimation of joint flexion and overestimation of joint extension by the Kinect™ device [64]. The study reported average mean differences between 3D and 2D angle measurements as approximately 19°. Although the Kinect™ was deemed unsuitable for clinical use in its current form, it was considered to have potential with future hardware and software advancements.

The use of a Kinect™ sensor compared to a normal camera could potentially compensate for the lack of depth parameters, but the need for such a sensor would not be practical for use in daily training. Furthermore, its accuracy was highest at lower movement speeds [64], making it less suitable for high-velocity movements such as martial arts kicks. In addition, depending on how the double side kick is performed there is the possible that key body points of the body are blocked or hidden by other parts of the performer's body. In such cases, even depth sensors would struggle to produce accurate estimations.

All videos in the dataset were recorded from a camera positioned sagittal to the plane of motion. Previous studies suggest that camera angle and position can influence agreement between 2D and 3D measurements [96] [60]. For tasks involving joint angle calculation, a sagittal camera placement is generally recommended [97], however studying the effects of different camera heights on analysis results or the effect of different camera angles on phase duration results could offer valuable insights into bridging the gap between 2D and 3D analysis.

As this thesis represents the first to assess phase duration measurements against 3D reference data, it is difficult to contextualize the results of the phase duration calculations. However, the low frame rate of the 2D test videos posed a limitation to potential accuracy, as some phases lasted only fractions of a second. The videos were recorded at 24 frames per second, with each frame representing 0.0417 seconds. Despite this limitation, the measurement results and manual review of the analyses suggest that the 2D models were capable of detecting the phases of the technique, with average mean differences across all phases for both OpenPose and MoveNet falling within 1–2 frames.

The hypothesis that there would be no significant differences between the analysis results using OpenPose and MoveNet models has to be rejected. For phase duration measurements, there is relatively strong agreement between the two models with minimal average mean differences and relatively narrow limits of agreement. For femur and leg spreading angles however, substantial mean differences and wide limits of agreement indicate strong differences in joint angle estimation between the models. Differences in relative knee height are also substantial. Both models show closer agreement in relative shoulder-knee distance estimates.

A review of the annotated test videos suggests that the MoveNet model struggled to correctly identify poses during moments of higher performer velocity, despite having been trained on additional exercise-related data [11]. This may be due to the generally lower image quality of the test videos, which could have affected the model's detection accuracy.

The expectation that the MoveNet model would perform better because of the additional training data was not met. When comparing the performance of OpenPose and MoveNet, OpenPose demonstrates generally tighter limits of agreement and smaller average mean differences across most metrics. For phase durations, both models perform comparably, with OpenPose showing slightly smaller average differences and narrower limits of agreement in CH1 and CH2. In femur angle estimation, OpenPose shows smaller

mean differences and narrower limits of agreement, suggesting greater accuracy and less variability. Similarly, OpenPose outperforms MoveNet in leg spreading angles. In terms of relative knee height, OpenPose again shows lower mean errors and tighter agreement compared to the more variable results from MoveNet. Finally, while both models perform similarly in estimating the relative shoulder-knee distance, OpenPose maintains slightly better agreement.

Overall, OpenPose offers superior performance with more accurate and stable results across the majority of evaluated kinematic parameters. While there are some studies that have compared OpenPose and MoveNet, most focus on single images across a variety of settings and contexts [98, 99]. However, in a study that analyzed gait kinematics using video data, OpenPose outperformed MoveNet in tracking walking movements [100], which is consistent with the findings in this study.

### 5.4.1 Limitations

This study only used one type of camera and one shooting angle for the recordings, which is similar to comparable 2D analysis studies [62, 61, 60]. It would be interesting to explore the effect of different recording setups and recording equipment on the analysis quality. Especially because it would better mirror the real world environment, where athletes using the automated analysis will not always have a similar recording quality and setup.

The videos in the provided data set were recorded in 24 frames per second. A higher frame rate would allow for a higher accuracy in measuring the characteristics. Only a subset of the test data was able to be used for validation, due to poor video quality of parts of the test data (recording equipment from approx. 2005). The effect of a higher quality video on analysis performance and accuracy should be explored in future studies.

**Cost of Analysis:** Besides effort and simplicity of use, the cost factor is essential for integration of video analysis in regular training. The analysis in this study relies on hardware that has a graphics processing unit. This type of hardware is generally more expensive [101]. Similar analysis can be performed on cheaper hardware, which only contains a central processing unit, however, this leads to a much longer analysis duration [65], which would prevent providing instant feedback to the user.

One option for reducing cost of deployment would be to switch from running the system on a virtual machine, to a serverless approach [102]. Azure Container Apps [103] provide the option to run a docker container as a serverless application. When a request comes in the container is started, and after the request is served, the container shuts down again. The majority of the costs of running the system would only be accrued when requests are actually being processed. This would represent a significant change in cost compared to the deployment on a virtual machine, where the system is accruing costs even if no requests are being processed.

This approach was explored during the creation of the system, however the performance of analysis was found to be not good enough to be used in an athlete's day to day

training. These performance issues might not be found in other solutions or with further technological advances in serverless technology.

The cost of storage is another factor that needs to be considered in a real-world deployment. A 60 frames per second, 4k video shot on an iPhone can take 400 megabytes of storage for one minute of video. If it is assumed that a user will upload multiple videos during one training session, the storage requirements can jump significantly with a rise in users. The cost of storage could be alleviated by not persisting the analysis, however the user experience would suffer because of the users not being able to review their previous analysis.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## Conclusion

While 3D analysis represents the gold standard for video analysis, due to its costly and time-consuming nature, it is not suited for use in day to day training. The system developed in the course of this thesis is able to provide quick feedback on video recorded using a normal mobile phone. This enables athletes to record their performance throughout training and in response to the provided analysis, adjust their technique immediately.

The system was developed in close contact with domain experts, ensuring the relevance and usefulness of the system in an athlete's training. The system was developed using the principles of prototyping and requirements engineering, creating a system that is extensible, performant, and able to be deployed in a variety of environments.

Based on the current findings, it can be inferred that the analysis produced by the system is close to the one produced by the more expensive 3D analysis. With the added benefit of offering a convenient and cost-effective means of technique analysis for helping to identify weaknesses in targeted training [104] [105]. It should be noted that the system's effectiveness and accuracy heavily rely on scientifically established performance indicators, as well as the quality of the source material. In light of these considerations, it can be concluded that the system serves as a valuable tool for various applications.

More research is needed to explore the impact of different camera setups, video quality, and recording environments on analysis accuracy and performance. New research in the field of depth estimation could further improve the accuracy of analysis. An approach combining pose estimation and depth estimation models could lead to an analysis that further narrows the gap between 3D and 2D analysis.

This thesis focused on one specific technique, in future work the system could be extended to have the ability to analyze a variety of different techniques. This would further enhance the usefulness in daily training.

## 6. CONCLUSION

---

In conclusion, this thesis demonstrates the feasibility of automated 2D video analysis for complex martial arts techniques. While challenges remain, the developed system represents a significant step towards making sophisticated biomechanical analysis more accessible to athletes and coaches in their daily training. As technology continues to evolve, the gap between 2D and 3D analysis may further narrow, potentially revolutionizing how athletes at all levels analyze and improve their performance.

# List of Figures

1.1	The double side kick with previously defined keypoints [9] . . . . .	5
1.2	CNN sliding window [38] . . . . .	10
1.3	CNN parameter sharing [38] . . . . .	11
1.4	VGG architecture [39] . . . . .	13
1.5	A normal convolution with $8x8x256$ output [43] . . . . .	14
1.6	The first layer of the depthwise convolution, using 3 kernels to transform a $12x12x3$ image to a $8x8x3$ image [43] . . . . .	14
1.7	Pointwise convolution, transforms an image with 3 channels to a single channel output [43] . . . . .	15
1.8	Pointwise convolution, transforms an image with 3 channels using 256 kernels to a 256 channel output [43] . . . . .	15
1.9	Residual layer [45] . . . . .	16
1.10	A bottleneck residual block with input size $k$ and output size $k'$ , stride $s$ and ration between input and output $t$ [40] . . . . .	16
1.11	MobileNetV2 architecture: Every line is a layer repeated $n$ times. $c$ is the number of output channels. $s$ is the stride value. $t$ is the ratio between input and output [40] . . . . .	17
1.12	OpenPose architecture [32] . . . . .	18
1.13	OpenPose methods [32] . . . . .	18
1.14	Image pyramid: same image represented at different scales [47] . . . . .	20
1.15	Feature Pyramid Network architecture [48] . . . . .	21
1.16	CenterNet prediction of center point and bounding box [49] . . . . .	21
1.17	MoveNet architecture with FPN and prediction heads [11] . . . . .	22
1.18	The annotation pipeline consists of (a) identifying the objects that appear in the image (b) locating and marking the object instances (c) the segmentation of each individual objects instance [50] . . . . .	23
1.19	Image annotation user interface for the COCO dataset [50] . . . . .	24
2.1	Phases of the work describing the methods and resulting artifacts. . . . .	29
3.1	Initial architecture design . . . . .	36
3.2	Mockup: Video upload screen . . . . .	37
3.3	Mockup: Analysis results screen . . . . .	38
3.4	Mockup: Recording instructions screen . . . . .	39
		73

3.5	Recording instructions page (left) and upload screen page (right) . . . . .	40
3.6	Analysis results screen . . . . .	41
3.7	Annotated video of the performance . . . . .	42
3.8	High-level architecture . . . . .	46
4.1	Bland-Altman plots for the four phases (OpenPose). . . . .	53
4.2	Bland-Altman plots for the femur angles (OpenPose). . . . .	54
4.3	Bland-Altman plots for the leg spreading angles (OpenPose). . . . .	55
4.4	Bland-Altman plots for the relative knee height (OpenPose). . . . .	55
4.5	Bland-Altman plots for the shoulder knee distance (OpenPose). . . . .	56
4.6	Bland-Altman plots for the four phases (MoveNet). . . . .	56
4.7	Bland-Altman plots for the femur angles (MoveNet). . . . .	57
4.8	Bland-Altman plots for the leg spreading angles (MoveNet). . . . .	57
4.9	Bland-Altman plots for the relative knee height (MoveNet). . . . .	58
4.10	Bland-Altman plots for the shoulder knee distance (MoveNet). . . . .	58
4.11	Bland-Altman plots for the four phases (MoveNet and OpenPose). . . . .	59
4.12	Bland-Altman plots for the femur angles (MoveNet and OpenPose). . . . .	59
4.13	Bland-Altman plots for the leg spreading angles (MoveNet and OpenPose). . . . .	60
4.14	Bland-Altman plots for the relative knee height (MoveNet and OpenPose). . . . .	60
4.15	Bland-Altman plots for the shoulder knee distance (MoveNet and OpenPose). . . . .	61
5.1	Depth map generated with Depth Pro . . . . .	65

# List of Tables

3.1	Initial requirements . . . . .	34
3.2	Final requirements list . . . . .	48
3.3	Nodes and phases of the double side kick. . . . .	49
3.4	Performance indicators. . . . .	50
1	Full Dataset: Phase Duration, Femur Angles, Leg Spreading, DKS, KHK/KI	86
2	OpenPose Full Dataset: Phases, Femur Angles, Leg Spreading, DKS, KHK/KI	87
3	MoveNet Full Dataset: Phases, Femur Angles, Leg Spreading, DKS, KHK/KI	88



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Bibliography

- [1] Benjamin Todd Drury, Thomas P Lehman, and Ghazi Rayan. Hand and wrist injuries in boxing and the martial arts. *Hand clinics*, 33(1):97–106, 2017.
- [2] Dominik Hoelbling, Martin Mattaeus Smiech, Dea Cizmic, Arnold Baca, and Peter Dabnichki. Exploration of martial arts kick initiation actions and telegraphs. *International Journal of Performance Analysis in Sport*, 21(4):507–518, 2021.
- [3] Ibrahim Ouergui, Slaheddine Delleli, Anissa Bouassida, Ezdine Bouhlel, Helmi Chaabene, Luca Paolo Ardigò, and Emerson Franchini. Technical-tactical analysis of small combat games in male kickboxers, 2021.
- [4] Tadeusz Ambroży, Łukasz Rydzik, Andrzej Kędra, Dorota Ambroży, Marta Niewczas, Ewa Sobilo, and Wojciech Czarny. The effectiveness of kickboxing techniques and its relation to fights won by knockout. *Archives of Budo*, 16, 10 2020.
- [5] Adrian Lees. Technique analysis in sports: a critical review. *Journal of sports sciences*, 20(10):813–828, 2002.
- [6] Bruce Elliott. Biomechanics: an integral part of sport science and sport medicine. *Journal of Science and Medicine in Sport*, 2(4):299–310, 1999.
- [7] A.C. Lapham and R.M. Bartlett. The use of artificial intelligence in the analysis of sports performance: A review of applications in human gait analysis and future directions for sports biomechanics. *Journal of Sports Sciences*, 13(3):229–237, 1995. doi: 10.1080/02640419508732232.
- [8] Sian Barris and Chris Button. A review of vision-based motion analysis in sport. *Sports Medicine*, 38:1025–1043, 2008.
- [9] Dominik Hölbling, Emanuel Preuschl, Michaela Hassmann, and Arnold Baca. Kinematic analysis of the double side kick in pointfighting, kickboxing. *Journal of sports sciences*, 35(4):317–324, 2017.

- [10] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021. doi: 10.1109/TPAMI.2019.2929257.
- [11] Ronny Votel and Na Li. Next-generation pose detection with movenet and tensorflow.js. <https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html>, 2024. Accessed: 2024-09-07.
- [12] Dominik Hölbling, Arnold Baca, and Peter Dabnichki. Sequential action, power generation and balance characteristics of a martial arts kick combination. *International Journal of Performance Analysis in Sport*, 20(5):766–781, 2020.
- [13] Caiming Zhang and Yang Lu. Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration*, 23:100224, 2021.
- [14] Maximilian Zorzetti, Ingrid Signoretti, Larissa Salerno, Sabrina Marczak, and Ricardo Bastos. Improving agile software development using user-centered design and lean startup. *Information and Software Technology*, 141:106718, 2022.
- [15] Jennifer N Maykut, Jeffery A Taylor-Haas, Mark V Paterno, Christopher A DiCesare, and Kevin R Ford. Concurrent validity and reliability of 2d kinematic analysis of frontal plane motion during running. *International journal of sports physical therapy*, 10(2):136, 2015.
- [16] Steven F DeFroda, Dai Sugimoto, Steven J Staffa, Donald S Bae, Ellen Shanley, Charles A Thigpen, and Peter K Kriz. Reliability of an observational biomechanical analysis tool in adolescent baseball pitchers. *International Journal of Sports Physical Therapy*, 16(6):1523, 2021.
- [17] Ed Yuncza. *Sport karate point sparring: An essential guide to the point fighting method*. Indomitable Pub., 2011.
- [18] Ibrahim Ouergui, Nizar Hssin, Emerson Franchini, Nabil Gmada, and Ezzedine Bouhleb. Technical and tactical analysis of high level kickboxing matches. *International Journal of Performance Analysis in Sport*, 13(2):294–309, 2013.
- [19] David Spiegelhalter. *The art of statistics: Learning from data*. Penguin UK, 2019.
- [20] Davide Giavarina. Understanding bland altman analysis. *Biochemia medica*, 25(2): 141–151, 2015.
- [21] J Martin Bland and DouglasG Altman. Statistical methods for assessing agreement between two methods of clinical measurement. *The lancet*, 327(8476):307–310, 1986.

- [22] Klaus Pohl. *Requirements engineering: An overview*. RWTH, Fachgruppe Informatik Aachen, 1996.
- [23] Klaus Pohl. *Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant*. Rocky Nook, Inc., 2016.
- [24] Didar Zowghi and Chad Coulin. Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and managing software requirements*, pages 19–46. Springer, 2005.
- [25] Winston W Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on Software Engineering*, pages 328–338, 1987.
- [26] Kent Beck. Embracing change with extreme programming. *Computer*, 32(10): 70–77, 1999.
- [27] Irena Petrijevcanin Vuksanovic and Bojan Sudarevic. Use of web application frameworks in the development of small applications. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 458–462. IEEE, 2011.
- [28] Costin-Emanuel Vasile, Andrei-Alexandru Ulmămei, and Călin Bîră. Image processing hardware acceleration—a review of operations involved and current hardware approaches. *Journal of Imaging*, 10(12):298, 2024.
- [29] Martin Kleppmann. *Designing data-intensive applications*, 2019.
- [30] Ian Miell and Aidan Sayers. *Docker in practice*. Simon and Schuster, 2019.
- [31] Chris Zimmerman. *The rules of programming: how to write better code*. " O'Reilly Media, Inc.", 2022.
- [32] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields, 2017.
- [33] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.
- [34] K Fukushima GI. A hierarchical neural network capable of visual pattern recognition. *Neural Network*, 1:90014, 1989.
- [35] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

- [36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [37] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.
- [38] Hobson Lane and Maria Dyshel. *Natural language processing in action*. Simon and Schuster, 2025.
- [39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. URL <https://arxiv.org/abs/1409.1556>.
- [40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.
- [41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [42] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [43] Chi-Feng Wang. A basic introduction to separable convolutions. <https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728>, 2024. Accessed: 2024-09-07.
- [44] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5927–5935, 2017.
- [45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [46] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855, 2006. doi: 10.1109/ICPR.2006.479.
- [47] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.
- [48] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

- [49] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.
- [50] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.
- [51] Amazon. Amazon mechanical turk. <https://www.mturk.com>, 2024. Accessed: 2024-10-04.
- [52] Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S Bernstein, Alex Berg, and Li Fei-Fei. Scalable multi-label annotation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3099–3102, 2014.
- [53] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Opensurfaces: A richly annotated catalog of surface appearance. *ACM Transactions on graphics (TOG)*, 32(4):1–17, 2013.
- [54] Google. Movenet model specifications. <https://storage.googleapis.com/movenet/MoveNet.SinglePose> Accessed: 2024-10-04.
- [55] Microsoft. Coco evaluation. <https://cocodataset.org/keypoints-eval>, 2024. Accessed: 2024-10-04.
- [56] Steven F DeFroda, Charles A Thigpen, and Peter K Kriz. Two-dimensional video analysis of youth and adolescent pitching biomechanics: A tool for the common athlete. *Current Sports Medicine Reports*, 15(5):350–358, 2016.
- [57] Abdelrahman Mostafa, Muhammad A Rushdi, Tamer A Basha, and Khaled Sayed. Footballtrace: An ai-based system for football player tracking with occlusion detection and trajectory correction. In *icSPORTS*, pages 194–201, 2023.
- [58] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.
- [59] Martina Rinaldi, Yasmien Nasr, Ghada Atef, Fabiano Bini, Tiwana Varrecchia, Carmela Conte, Giorgia Chini, Alberto Ranavolo, Francesco Draicchio, Francesco Pierelli, et al. Biomechanical characterization of the junzuki karate punch: indexes of performance. *European journal of sport science*, 18(6):796–805, 2018.
- [60] Sarah M Remedios and Steven L Fischer. Towards the use of 2d video-based markerless motion capture to measure and parameterize movement during functional capacity evaluation. *Journal of Occupational Rehabilitation*, 31(4):754–767, 2021.

- [61] Alexander T Peebles, Maddy M Carroll, John J Socha, Daniel Schmitt, and Robin M Queen. Validity of using automated two-dimensional video analysis to measure continuous sagittal plane running kinematics. *Annals of biomedical engineering*, 49: 455–468, 2021.
- [62] Stacy A Schurr, Ashley N Marshall, Jacob E Resch, and Susan A Saliba. Two-dimensional video analysis is comparable to 3d motion capture in lower extremity movement assessment. *International journal of sports physical therapy*, 12(2):163, 2017.
- [63] Ming-Zhu Chen, Xin Wang, Qi Chen, Yue Ma, Ivan Malagoli Lanzoni, and Wing-Kai Lam. An analysis of whole-body kinematics, muscle strength and activity during cross-step topspin among table tennis players. *International Journal of Performance Analysis in Sport*, pages 1–13, 2022.
- [64] Alexandra Pfister, Alexandre M West, Shaw Bronner, and Jack Adam Noah. Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis. *Journal of medical engineering & technology*, 38(5):274–280, 2014.
- [65] Xing Liang, Epaminondas Kapetanios, Bencie Woll, and Anastassia Angelopoulou. Real time hand movement trajectory tracking for enhancing dementia screening in ageing deaf signers of british sign language. In *Machine Learning and Knowledge Extraction: Third IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2019, Canterbury, UK, August 26–29, 2019, Proceedings 3*, pages 377–394. Springer, 2019.
- [66] Yung-Che Li, Ching-Tang Chang, Chin-Chang Cheng, and Yu-Len Huang. Baseball swing pose estimation using openpose. In *2021 IEEE International Conference on Robotics, Automation and Artificial Intelligence (RAAI)*, pages 6–9. IEEE, 2021.
- [67] Pouya Jafarzadeh, Petra Virjonen, Paavo Nevalainen, Fahimeh Farahnakian, and Jukka Heikkonen. Pose estimation of hurdles athletes using openpose. In *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–6. IEEE, 2021.
- [68] Github. Openpose doc - installation. [https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/installation/0\\_index.md](https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/installation/0_index.md), 2024. Accessed : 2024 – 10 – 04.
- [69] Github. Openpose doc - speed. [https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/06\\_maximizing\\_openpose\\_speed.md](https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/06_maximizing_openpose_speed.md), 2024. Accessed : 2024 – 10 – 04.
- [70] William C Adams. Conducting semi-structured interviews. *Handbook of practical program evaluation*, pages 492–505, 2015.

- [71] Philipp Mayring et al. Qualitative content analysis. *A companion to qualitative research*, 1(2):159–176, 2004.
- [72] Kurt Meinel and Günter Schnabel. *Bewegungslehre-Sportmotorik: Abriss einer Theorie der sportlichen Motorik unter pädagogischem Aspekt*. Meyer & Meyer Verlag, 2007.
- [73] U Göhner. Movement theory: Fundamental aspects. *Sport Science in Germany: An Interdisciplinary Anthology*, pages 191–200, 1992.
- [74] Christiane Floyd. A systematic look at prototyping. In *Approaches to prototyping*, pages 1–18. Springer, 1984.
- [75] J Martin Bland and Douglas G Altman. Measuring agreement in method comparison studies. *Statistical methods in medical research*, 8(2):135–160, 1999.
- [76] Dan Oved. Posenet. <https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5>, 2024. Accessed: 2024-10-04.
- [77] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [78] Microsoft Corporation. Csharp. <https://learn.microsoft.com/en-us/dotnet/csharp/>, 2023.
- [79] Python Software Foundation. Python. <https://www.python.org/psf-landing/>, 2023.
- [80] Openpose. Openpose hardware requirements. <https://github.com/tramper2/openpose/blob/master/doc/installat> 2023. Accessed: 2023-06-01.
- [81] The Full Stack. The full stack website. <https://fullstackdeeplearning.com/cloud-gpus/>, 2023. Accessed: 2023-06-01.
- [82] Microsoft. Azure cloud computing services. <https://azure.microsoft.com/en-gb/>, 2023. Accessed: 2023-05-31.
- [83] David Jaramillo, Duy V Nguyen, and Robert Smart. Leveraging microservices architecture by using docker technology. In *SoutheastCon 2016*, pages 1–5. IEEE, 2016.
- [84] Microsoft. Azure container instances. <https://azure.microsoft.com/de-de/products/container-instances>, 2023. Accessed: 2023-06-01.
- [85] Mike Mesnier, Gregory R Ganger, and Erik Riedel. Object-based storage. *IEEE Communications Magazine*, 41(8):84–90, 2003.
- [86] Microsoft. Azure blob storage. <https://azure.microsoft.com/en-us/products/storage/blobs>, 2024. Accessed: 2024-10-01.
- [87] Dirk Merkel et al. Docker: lightweight linux containers for consistent development and deployment. *Linux j*, 239(2):2, 2014.

- [88] Google. Angular. <https://angular.dev>, 2024. Accessed: 2024-10-01.
- [89] Mozilla. Http range requests. [https://developer-mozilla-org.translate.goog/en-US/docs/Web/HTTP/Range\\_requests?xtrsl=en\\_xtrtl=de\\_xtrhl=de\\_xtrpto=sc](https://developer-mozilla-org.translate.goog/en-US/docs/Web/HTTP/Range_requests?xtrsl=en_xtrtl=de_xtrhl=de_xtrpto=sc), 2024. Accessed : 2024 – 10 – 01.
- [90] Tailwind. Tailwind css. <https://tailwindcss.com>, 2024. Accessed: 2024-10-01.
- [91] Flask. Flask. <https://flask.palletsprojects.com/en/3.0.x/>, 2024. Accessed: 2024-10-01.
- [92] OpenCV. Opencv. <https://opencv.org>, 2024. Accessed: 2024-10-01.
- [93] Microsoft. Azure virtual machine specs. <https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/gpu-accelerated/nvadsa10v5-series?tabs=sizebasic>, 2024. Accessed: 2024-10-01.
- [94] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. *arXiv preprint arXiv:2410.02073*, 2024.
- [95] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021.
- [96] Colin D McKinnon, Michael W Sonne, and Peter J Keir. Assessment of joint angle and reach envelope demands using a video-based physical demands description tool. *Human Factors*, 64(3):568–578, 2022.
- [97] Brian D Lowe, Patricia Weir, and David Andrews. Observation-based posture assessment: review of current practice and recommendations for improvement. 2014.
- [98] BeomJun Jo and SeongKi Kim. Comparative analysis of openpose, posenet, and movenet models for pose estimation in mobile devices. *Traitement du Signal*, 39(1):119, 2022.
- [99] Prakarsh Kaushik, Bhanu Prakash Lohani, Anil Thakur, Amardeep Gupta, Akhilesh Kumar Khan, and Ashish Kumar. Body posture detection and comparison between openpose, movenet and posenet. In *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)*, volume 6, pages 234–238. IEEE, 2023.
- [100] Edward P Washabaugh, Thanikai Adhithiyam Shanmugam, Rajiv Ranganathan, and Chandramouli Krishnan. Comparing the accuracy of open-source pose estimation methods for measuring gait kinematics. *Gait & posture*, 97:188–195, 2022.
- [101] Microsoft. Azure virtual machine pricing. <https://azure.microsoft.com/en-us/pricing/details/virtual-machines/windows/pricing>, 2023. Accessed: 2023-06-01.
- [102] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, et al. Serverless computing: Current trends and open problems. *Research advances in cloud computing*, pages 1–20, 2017.

- [103] Microsoft. Azure container apps. <https://azure.microsoft.com/de-de/products/container-apps>, 2024. Accessed: 2024-10-01.
- [104] Dominik Hoelbling, Manfred Grafinger, Martin Mattaeus Smiech, Dea Cizmic, Peter Dabnichki, and Arnold Baca. Acute response on general and sport specific hip joint flexibility to training with novel sport device. *Sports Biomechanics*, pages 1–16, 2021.
- [105] Dominik Hoelbling, Manfred Grafinger, Arnold Baca, and Peter Dabnichki. The flexibility trainer: Feasibility analysis, prototype-and test station development for a sports device for hip-joint flexibility and strength enhancement. In *8th International Conference on Sport Sciences Research and Technology Support (ICSPORTS)*, pages 22–29, 2020.

## Raw Data

Table 1: Full Dataset: Phase Duration, Femur Angles, Leg Spreading, DKS, KHK/KI

ID	Phase Duration				Femur Angles				Leg Spreading				DKS		KHK/KI		
	tCH1	tKI1	tCH2	tKI2	MT1	KE1	MT2	KE2	MT1	KE1	MT2	KE2	DKS1	DKS2	KHK1	KI1	KHK2
1	0.352	0.056	0.368	0.112	142.80	134.78	120.84	128.24	135.36	128.68	123.15	115.26	67	80	115	115	137
2	0.308	0.072	0.332	0.120	81.49	58.16	123.51	131.15	65.57	41.80	108.16	121.72	85	69	81	67	127
4	0.372	0.036	0.060	0.280	139.95	150.39	131.57	133.67	130.37	130.69	118.97	130.53	59	51	96	142	137
6	0.456	0.080	0.268	0.164	88.15	82.30	99.86	117.22	82.80	71.95	112.98	113.14	86	82	105	106	117
7	0.340	0.016	0.240	0.160	85.18	89.42	125.60	138.02	78.37	83.42	123.78	114.04	87	85	90	95	128
11	0.208	0.056	0.300	0.092	88.15	82.00	105.15	110.66	71.66	61.23	87.85	107.09	84	68	93	86	121
12	0.264	0.084	0.308	0.136	84.19	76.35	121.31	130.07	78.59	55.93	110.64	111.79	81	80	101	81	112
13	0.308	0.084	0.324	0.108	92.00	79.69	101.59	98.27	86.61	78.12	103.16	100.00	80	90	101	98	109
14	0.288	0.044	0.440	0.108	124.28	122.74	126.21	131.21	110.87	112.11	122.92	128.85	79	67	101	123	132
16	0.420	0.044	0.192	0.084	108.61	106.00	121.89	135.85	100.00	92.39	120.85	121.31	72	69	91	104	120
17	0.228	0.052	0.296	0.076	110.17	109.54	125.25	138.80	97.66	93.55	128.96	127.92	80	61	94	111	137
18	0.328	0.024	0.216	0.104	119.71	131.87	122.20	128.17	102.96	120.48	103.01	115.43	73	57	80	134	153
21	0.316	0.108	0.272	0.104	93.37	83.32	119.88	114.34	89.14	56.55	111.06	110.83	87	79	76	66	107
22	0.396	0.076	0.268	0.152	91.10	76.04	100.54	115.62	84.94	62.70	111.26	117.91	83	91	95	83	114
23	0.364	0.008	0.268	0.064	134.34	134.28	130.37	128.12	121.29	118.45	112.42	108.88	72	62	100	120	160
24	0.380	0.072	0.336	0.164	92.86	87.06	121.61	146.25	85.34	58.17	115.81	128.09	90	76	82	78	118
25	0.288	0.064	0.292	0.100	115.88	103.60	115.57	120.22	97.66	90.33	111.74	114.83	72	71	89	102	91
26	0.276	0.040	0.256	0.096	108.54	96.01	120.94	122.76	85.80	74.02	113.12	116.09	71	68	120	101	167
29	0.252	0.036	0.176	0.096	110.39	107.83	122.31	124.00	93.54	94.57	116.11	117.38	81	77	126	126	144
30	0.252	0.060	0.296	0.160	110.74	89.47	109.19	113.53	89.47	80.43	103.55	110.13	78	60	108	106	144
33	0.356	0.048	0.284	0.072	125.95	108.99	129.04	122.83	112.98	104.49	121.20	110.81	70	70	115	114	149
37	0.532	0.088	0.396	0.244	86.02	66.87	105.73	114.58	72.45	56.01	96.05	99.52	79	61	94	75	129
38	0.336	0.072	0.248	0.100	126.63	104.47	115.33	123.62	97.56	83.26	114.27	121.17	73	74	123	118	110
39	0.320	0.076	0.296	0.148	114.13	93.20	102.58	115.36	100.06	80.35	106.71	114.71	74	62	115	113	147
40	0.268	0.088	0.412	0.152	99.29	98.48	121.08	119.94	83.66	86.61	112.12	107.43	86	69	74	82	121
42	0.300	0.044	0.300	0.196	78.35	65.69	110.42	102.72	83.10	65.65	112.97	90.38	83	64	96	94	125

Table 2: OpenPose Full Dataset: Phases, Femur Angles, Leg Spreading, DKS, KHK/KI

ID	Phase Duration				Femur Angles				Leg Spreading				DKS		KHK/KI		
	tCH1	tKI1	tCH2	tKI2	MT1	KE1	MT2	KE2	MT1	KE1	MT2	KE2	DKS1	DKS2	KHK1	KI1	KHK2
1	0.279	0.119	0.279	0.199	133.123	132.630	137.414	122.065	111.025	121.255	113.292	114.993	63	68	115	115	137
2	0.278	0.079	0.198	0.238	98.970	62.116	123.175	120.151	73.837	60.509	110.940	112.726	90	69	81	67	127
4	0.279	0.119	0.119	0.279	125.341	149.650	152.524	118.276	113.105	125.282	121.322	123.408	72	62	96	142	137
6	0.357	0.040	0.238	0.040	101.420	101.383	113.505	107.608	91.991	88.241	112.243	105.818	84	91	105	106	117
7	0.319	0.119	0.159	0.279	91.055	99.632	124.971	84.607	86.698	85.400	111.360	69.244	86	78	90	95	128
11	0.199	0.079	0.278	0.079	97.800	80.801	115.324	110.871	86.007	68.520	92.035	91.265	88	70	93	86	121
12	0.280	0.120	0.320	0.280	98.975	91.613	118.292	118.518	85.526	68.858	118.748	118.207	82	84	101	81	112
13	0.333	0.100	0.433	0.133	99.493	95.693	106.866	98.513	101.333	79.514	108.659	101.768	72	84	101	98	109
14	0.279	0.040	0.279	0.119	120.549	129.617	129.711	129.929	115.728	118.280	109.957	113.668	72	64	101	123	132
16	0.278	0.119	0.198	0.040	98.824	95.066	117.108	110.141	102.738	91.125	103.042	117.036	80	59	91	104	120
17	0.239	0.080	0.239	0.080	110.407	143.486	141.424	135.658	102.548	109.008	126.514	124.163	87	69	94	111	137
18	0.239	0.119	0.239	0.080	100.370	122.454	124.798	124.117	108.036	116.842	113.596	116.067	82	53	80	134	153
21	0.318	0.079	0.199	0.040	89.531	55.790	99.183	107.532	88.145	47.027	111.879	106.601	87	86	76	66	107
22	0.278	0.079	0.199	0.119	78.014	78.909	95.661	101.011	93.254	73.076	99.629	107.421	82	75	95	83	114
23	0.237	0.039	0.276	0.039	134.008	137.642	139.754	126.353	111.826	115.006	110.930	97.960	72	43	100	120	160
24	0.361	0.161	0.281	0.080	85.058	67.744	114.167	110.973	90.312	74.564	115.106	121.801	94	104	82	78	118
25	0.200	0.040	0.120	0.040	118.542	90.455	120.717	121.845	106.758	93.295	114.752	119.204	69	86	89	102	91
26	0.318	0.079	0.238	0.040	107.823	124.462	125.512	68.418	90.100	116.576	107.789	55.316	77	78	120	101	167
29	0.238	0.119	0.158	0.079	123.485	130.371	133.936	120.328	95.030	98.935	118.726	114.186	74	65	126	126	144
30	0.278	0.080	0.278	0.159	105.617	98.572	118.073	119.108	94.388	87.239	115.642	126.559	79	71	108	106	144
33	0.279	0.040	0.279	0.080	121.101	113.163	127.916	117.181	106.488	102.492	121.775	116.965	71	72	115	114	149
37	0.518	0.080	0.359	0.160	95.624	64.638	120.800	119.181	85.177	51.569	111.059	110.225	87	70	94	75	129
38	0.236	0.079	0.197	0.079	113.757	112.854	124.207	81.322	103.064	105.530	104.068	85.380	82	85	123	118	110
39	0.360	0.120	0.280	0.120	105.886	92.586	107.882	115.077	103.698	86.553	108.345	116.644	66	46	115	113	147
40	0.237	0.198	0.316	0.158	93.636	99.765	119.873	112.003	98.355	77.203	115.697	101.123	80	88	74	82	121
42	0.240	0.120	0.240	0.160	88.585	95.305	104.049	103.169	103.125	97.905	116.406	103.394	79	73	96	94	125

Table 3: MoveNet Full Dataset: Phases, Femur Angles, Leg Spreading, DKS, KHK/KI

ID	Phase Duration				Femur Angles				Leg Spreading				DKS		KHK/KI		
	tCH1	tKI1	tCH2	tKI2	MT1	KE1	MT2	KE2	MT1	KE1	MT2	KE2	DKS1	DKS2	KHK1	KI1	KHK2
1	0.398	0.080	0.319	0.119	122.99	124.72	135.14	117.87	114.02	113.71	112.60	108.91	71	70	110	127	143
2	0.278	0.079	0.278	0.278	56.77	31.85	88.50	113.01	44.44	21.18	92.86	117.43	82	67	82	65	117
4	0.319	0.080	0.199	0.159	120.19	151.47	127.58	114.39	110.04	118.43	97.85	120.55	68	52	119	155	152
11	0.397	0.040	0.199	0.159	91.06	99.63	124.97	84.61	86.70	85.40	111.36	69.24	87	89	66	71	96
12	0.280	0.160	0.320	0.400	82.56	79.63	84.66	91.87	69.55	57.66	76.90	91.93	91	75	78	92	118
13	0.278	0.159	0.358	0.119	52.02	68.10	84.31	112.89	54.13	54.85	103.84	111.53	85	90	60	75	99
16	0.238	0.079	0.278	0.198	95.36	103.32	92.53	121.81	92.46	102.61	88.46	120.21	80	59	91	104	120
17	0.318	0.119	0.239	0.239	98.82	95.07	117.11	110.14	102.74	91.13	103.04	117.04	79	62	76	108	116
18	0.239	0.119	0.239	0.080	110.41	143.49	141.42	135.66	102.55	109.01	126.51	124.16	87	69	94	111	137
21	0.358	0.159	0.238	0.159	89.53	55.79	99.18	107.53	88.15	47.03	111.88	106.60	80	88	83	67	100
22	0.358	0.040	0.278	0.079	59.55	68.06	54.11	91.65	70.07	67.23	76.59	103.96	83	75	87	99	114
23	0.237	0.039	0.276	0.039	124.55	130.50	145.73	121.84	110.82	117.06	112.87	111.13	64	33	115	140	138
24	0.040	0.040	0.361	0.120	74.07	60.81	70.55	41.11	73.09	46.54	81.49	71.67	96	99	83	80	82
25	0.320	0.120	0.400	0.120	118.54	90.46	120.72	121.85	106.76	93.30	114.75	119.20	69	86	89	102	91
26	0.278	0.119	0.238	0.199	108.73	92.68	110.92	120.74	83.59	58.27	97.66	122.43	77	69	112	92	134
29	0.277	0.119	0.238	0.079	64.80	80.05	103.11	116.99	34.68	49.93	92.64	89.92	87	79	76	90	124
30	0.239	0.040	0.080	0.040	97.83	83.15	105.01	111.67	85.52	75.71	111.77	116.55	77	61	103	95	128
33	0.358	0.119	0.279	0.040	123.49	104.29	126.10	120.80	101.12	88.38	114.07	118.53	68	66	111	107	145
37	0.518	0.160	0.439	0.239	67.29	54.15	120.93	113.60	68.99	40.13	100.57	110.16	75	67	103	73	129
38	0.276	0.158	0.158	0.079	113.76	112.85	124.21	81.32	103.06	105.53	104.07	85.38	92	77	64	67	136
39	0.400	0.160	0.280	0.160	94.95	75.71	91.51	101.75	92.41	60.30	98.09	109.40	72	50	108	107	138
40	0.356	0.158	0.435	0.119	54.78	86.55	99.78	110.48	52.70	67.16	104.81	102.48	83	80	62	84	121
42	0.240	0.120	0.240	0.160	88.59	95.31	104.05	103.17	103.13	97.91	116.41	103.39	85	69	90	93	122