FACHHOCHSCHULE DORTMUND

Project Thesis
on

Landing Screening of the Volleyball players from the National Team U17 and U19 with Landing Error Scoring System scoring tool using Computer Vision

Author: Catari Suresh Babu, Vignesh
Matriculation Number: 7213383

Supervisor 1:
Prof. Dr. Jaitner, Thomas
thomas.jaitner@tu-dortmund.de
Institut für Sport and Sportwissenschaft
Technische Universität Dortmund

Supervisor 2:
Prof. Dr. Weyers, Stephan
stephan.weyers@fh-dortmund.de
Fachhochschule Dortmund

A thesis submitted in partial fulfillment of the requirements
for the degree of

Master of Science
in
Digital Transformation

April 2023

Table of Contents

List of Figures

List of Abbreviations

ACL: Anterior Cruciate Ligament

AGI: Artificial General Intelligence

AI: Artificial Intelligence

ANI: Artificial Narrow Intelligence

ASI: Artificial Super Intelligence

FPS: Frame rate Per Second

IC: Initial Contact

LESS: Landing Error Scoring System

ML: Machine Learning

ROI: Region of Interest

1. Introduction

“Computer vision is a foundational technology that can revolutionize the way we live and work. From healthcare to agriculture, education to entertainment, computer vision has the potential to transform entire industries, and we're only scratching the surface of what's possible.”

- *Sundar Pichai, CEO of Google, and Alphabet Inc.*

"Computer vision is at the heart of how we’re thinking about the intelligent cloud and the intelligent edge. That means everything that Microsoft is doing, whether it’s HoloLens, Windows Hello, or Kinect, is all going to be infused with these capabilities to understand the environment around us."

- *Satya Nadella, CEO of Microsoft.*

In this chapter, the motivation for the project work on Computer Vision and Injury Prevention is presented in section **1.1**, followed by a discussion of the problems that the project aims to handle and solve in section 1.2. Additionally, section 1.3 outlines the limits and potential use cases of this Project thesis.

1.1 Motivation

In the past few years, there has been a remarkable evolution in the field of (AI) and its application in everyday life. AI has the potential to enable machines to perform tasks that were previously only possible for humans, such as perceiving, reasoning, rationalizing, and problem-solving within a given context or environment. This results in greater efficiency and accuracy in interacting with the environment [1].

“Essentially, AI is the field of computer science that involves enabling computers to behave like humans or perform tasks that usually require human intelligence [1].”

The current categorization system recognizes four main types of AI: reactive, limited memory, theory of mind, and self-awareness**.** Reactive AI belongs to the basic category, where it receives input and provides a predictable output. Limited memory AI learns from previous experiences and uses observations to make outputs. Theory of Mind AI can make decisions like a human mind, while Self-aware AI is the most advanced and can understand its emotions and those of others **[1,2].** AI systems can be categorized based on the complexity of the problem they can solve. In response to a specific request, ANI, also known as Weak AI, can solve an issue within a limited range of skills. Siri, which is available on cellphones, is an example of ANI. AGI, often known as Strong AI, refers to robots that can do tasks as well as humans. The Pillo robot, for example, can diagnose and give medications. ASI involves machines that can execute jobs that humans cannot. The Alpha 2 robot is an ASI effort; it can manage a smart home and control domestic equipment [1].

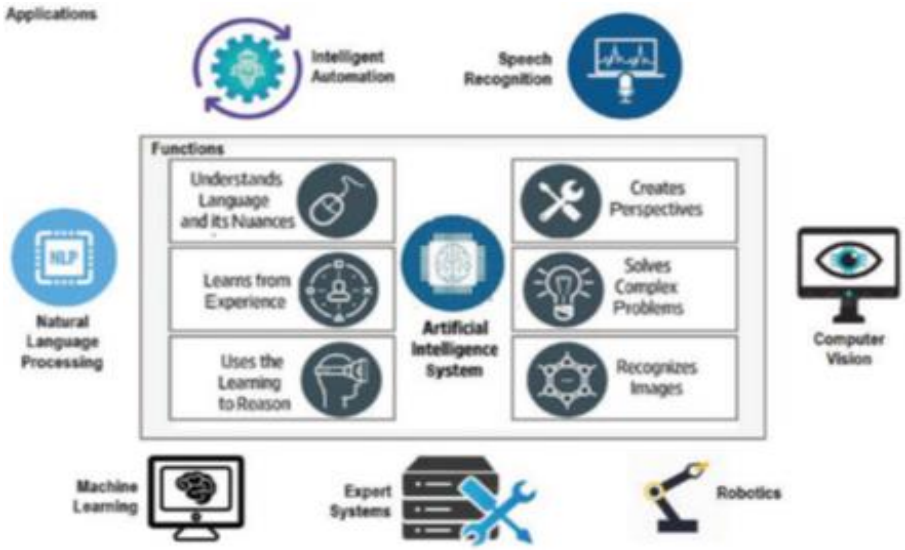

Figure 1 Functions of Artificial Intelligence

Figure 1 depicts the fundamental functions and characteristics of an AI system, as well as associated subfields to demonstrate their importance in enabling the execution of these functions [1,2].

The application of AI are

- Computer vision
- Natural language processing
- Robotics
- Machine learning and deep learning
- Expert systems
- Speech or voice recognition
- Intelligent automation

Although there is interrelation among these subfields, our focus and project scope is centered around Computer Vision. Therefore, we will be delving deeper into Computer Vision specifically, even though real-world implementations typically involve one or more of these subfields.

Computer vision, sometimes just called "vision," is a cutting-edge area of computer science that deals with making it possible for machines to see, comprehend, interpret, or otherwise modify what is being observed [1]. To interpret text that has been retrieved from pictures, computer vision technology uses deep learning algorithms and, in certain situations, NLP approaches. Building tasks like picture categorization, object identification, posture estimation tracking, and image editing has been easier and more precise thanks to deep learning's developments, opening the door to explore more sophisticated applications [1,2].

The science of computer vision is concerned with creating algorithms and systems to precisely ascertain the posture and movement of articulated bodies [1]. The study focuses on articulated body posture estimation, which entails locating joints and body parts in an image of a human [3]. There are several technological restrictions, such as real-time node image processing and whole-body tracking. Google's MediaPipe Pose is a ML approach to high-fidelity based on body posture monitoring that infers 33 3D landmarks and applies a background bounding boxes over the entire body. In August 2019, Google released MediaPipe, an open-source framework that heavily utilizes AI methods including Item Tracking, Pose Estimation, Face Recognition, Multi-node Tracking on Human Body, and many others [3,4].

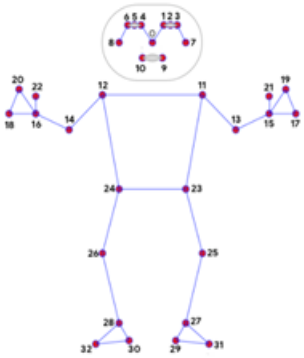

Figure 2 Pose Landmarks [4]

Figure 2 displays how MediaPipe Pose's landmark model predicts the locations of 33 posture landmarks. Please refer to Appendix A for more details about the BlazePose 33 landmarks. A full-body segmentation mask that is expressed as a two-class delineation is an optional component of MediaPipe Pose (human or background) [4]. With the aid of MediaPipe, which has several applications, we could use it in this project as a screening tool for the LESS, which was primarily used for the study of athletes.

LESS is a score system that was created as a clinical evaluation tool with the primary goal of identifying people who were at risk for lower extremity injuries. This tool's inability to be evaluated in real time and need for video cameras are two of its drawbacks [5,6]. However still, the LESS can effectively assess adjustments made to landing technique because of an injury-prevention program. The LESS can be promptly scored by a physician, but it needs two video cameras and video analysis. If the LESS were changed to permit real-time analysis in a clinical context, its usefulness may be increased. The Landing Error Scoring System-Real Time, or LESS-RT, is a modified LESS, and the aim of this study was to assess its dependability. The LESS-RT was created to offer information on a person's movement style during a jump-landing assignment and to be assessed in real time in a clinical setting [5, 6].

The development of Artificial Intelligence in technology and the well-known LESS scoring tool inspired me to carry up this project for the identification of landing of the athletes. The project processes and the Python code for working with computer vision with the aid of OpenCV, MediaPipe libraries, and the idea of posture estimation for identifying the landmarks of each node of the human body are shown in the subsection.

1.2 Problem Statement

Several individuals study recordings of each player's landing and figure out the angle of displacement of each item in the Landing Error Scoring System to manually do LESS analysis for the diagnosis of injuries. When computations are made pixel by pixel, it may take a long time and lose precision. Also, the videos could be of poor quality and clarity, the analysis could solely be done in two dimensions, and it might be difficult to discover an appropriate estimate with just the naked eye. To address this issue, significant technological breakthroughs should be put to use to make evaluation quicker and more efficient, letting us eliminate the reliance on human judgment.

1.3 Objective of the Project

In this project, the stated problems are reduced with the combination of Computer Vision and LESS technologies simplifying the extremely complicated math into a straightforward, more precise technique to recognize each athlete's landing. The study also illustrates the reliability of incorporating modern technologies into a traditional scoring system and achieving the optimum outcome with it.

1.4 Scope of the project

This project has the stipulation that the evaluation could only be accomplished in 2D because of the constraints that it could only be read frame by frame of the entire video, significantly slowing down the video in terms of frames per second. However, this adds a benefit for our evaluation to find the precise displacement of each LESS item. A select few evaluations are also carried out in both projection and lateral views.

2. Literature Review

This chapter is composed of two sections: section 2.1 presents a thorough overview of the literature survey and describes the results of the literature research. Section 2.2 emphasizes the gap as well as the distance that is absent in such works, as well as their limits, and responds to the query, "Isn't there any other project work on this issue previously done?" Additionally, providing an overview of key ideas and concepts relevant to this issue, as well as the approach used and the motivation that prompted me to commence on this project.

2.1 Theoretical Foundation

Several studies have been conducted utilizing the LESS scoring system, which is mostly used to examine athletic injuries such as ACL injuries. The ACL is a ligament in the knee that connects the femur (thigh bone) to the tibia (shin bone) (shin bone). It is one of the four primary ligaments of the knee joint, and it is important for knee joint stability by restricting excessive forward movement of the tibia relative to the femur. The ACL is also involved in the rotational stability of the knee joint. ACL injuries are frequent in sports that involve pivoting, leaping, or rapid changes in direction, and can result in knee instability and discomfort. Padua et al. concluded that the LESS would be a valid and reliable way to recognize potentially high-risk muscle movements throughout a jump-landing activity. A motion capture technology was recently employed to examine the biomechanics of the injury process. The drop vertical jump (DVJ), for example, has been used to assess risky movements for ACL injury after landing after a leap [14]. Females with a larger knee valgus angle and external knee valgus movements during a drop-landing activity were at an elevated risk for sustaining a noncontact ACL injury, according to Hewett et al. In addition, Haddas et al. observed that females, as compared to males, demonstrated biomechanical variables that may enhance their risk of ACL damage while landing from a 0.30-m height, particularly when landing after exhaustion. Moreover,

while laboratory-based motion analysis systems are the gold standard, the Landing Error Scoring System (LESS) is a low-cost clinical evaluation tool that utilizes two conventional video cameras to evaluate landing biomechanics. [14, 15, 16, 18]. More than 25% of persons with ACL injuries do not return to former activity levels despite successful surgery and rehabilitation. 65% of people stop playing soccer within 7 years after an ACL damage. Returning to sports such as soccer within 2 years after an ACL damage is also difficult for more than one-third of skeletally immature youngsters. 2 Patients have reported moderate to severe difficulty in walking (31%) and activities of daily life (44%), regardless of therapy [5, 6].

Teenage female players are the most at risk for poor long-term outcomes, and over 200,000 ACL tears occur in the United States each year, with more than 22% of these athletes requiring contralateral knee surgery or revision surgery within 5 years after the index treatment [5, 6, 14]. Since surgery and rehabilitation may not prevent long-term morbidity, there is an urgent need to prevent ACL injuries in young soccer players. Prospective risk factors for harm should be determined before taking preventative measures. Most postulated risk variables for ACL damage, including as sex, hormone fluctuations, notch width, and static postural alignment, are not adjustable through preventive measures. Yet, faulty lower extremity biomechanics are adjustable and important aspects to investigate since they result in aberrant knee loads. In particular, 3-dimensional knee loading, including knee-extension moment, proximal anterior tibial shear force, and knee valgus at comparable levels of competitiveness and always on natural grass [5, 6].

2.1.1 Methods

2.1.1.1 Layout and the Participants

A few articles show that a prospective cohort design to evaluate the LESS as a predictor of ACL injury in elite-youth athletes and the number of participants has an approximately equal number of male and female participants with ages ranging from 11 to 20 years, mean body mass index ([BMI]=21.50.9 kg/m2). At the initial test session, all participants were free of any injury or sickness that would have prevented them from participating in competitive soccer. Before the first test session, participants and their legal guardians supplied signed informed assent and consent, respectively [5, 6, 14].

2.1.1.2 Test Procedures

Participants must complete a baseline questionnaire and mobility assessment at the start of each soccer season, which included a jump-landing activity scored using the LESS (Landing Error Scoring System) via video review. At the start of their returning seasons, returning athletes were retested using similar techniques. Participants were prospectively followed for ACL damage, and a member of the study team visited each participant once a week throughout the monitoring period to document injuries. Coaches and athletes were encouraged to identify any participant who had missed a sport-related activity the previous week owing to an injury or an unclear cause. A member of the study team followed up with all individuals who had possible injuries [5, 6, 14, 21].

All individuals who had reported ACL injuries filled out a questionnaire to confirm the condition and learn more about the circumstances surrounding the injury. All reported ACL injuries were confirmed following surgical repair and were documented on this self-reported questionnaire. Contact that is non-intrusive and indirect ACL injuries were defined as injuries that occurred without direct contact with the lower extremity from an external entity now of injury. Injuries caused by noncontact mechanisms of damage were defined as those that did not include contact with the participant, whereas injuries caused by contact with a body component other than the knee, such as the trunk, were characterized as those caused by indirect contact mechanisms of injury [5, 6, 14, 21].

2.1.1.3 Experiment

During each test session, participants completed three attempts of a standardized jump landing task. The participant began the exercise by standing on a 30-cm-high box that was placed half a body height away from a landing site marked by a line on the ground. The instructions given to participants were to jump forward such that both limbs exited the box at the same moment, to land just over the line, and to jump swiftly for maximum height after landing. They rehearsed until they felt confident in their ability to complete the assignment successfully. If individuals leaped vertically from the box or did not jump for maximum height upon landing, trials were eliminated and redone. To record frontal and sagittal images of all jump-landing attempts, a few digital video cameras should be set on the front and both sides of the participants [5, 6, 14, 21].

2.1.2 Result

In the preseason baseline questionnaire, no participant who experienced an ACL damage reported a previous lower limb injury [5, 6]. Female athletes are more prone to get ACL injuries as a result of a multifactorial neuromusculoskeletal phenomena [15, 16, 18]. The main difference between the female athletes was dependent on their years of experience, and the peak knee abduction moment within 40 ms of the IC before the FP was considerably greater in female university athletes than in female leisure athletes [14].

2.2 Turning Gaps and Limitations into Solutions: A Comprehensive Gap Analysis

The key constraints encountered while reviewing all the articles published by Padua et al. regarding the LESS screening tool for ACL injury prevention were that they had to capture digital recordings of all participants and afterwards evaluate the film with the assistance of research assistants. This slows the review process, as well as the need for extra research assistants and additional storage space. Moreover, Eckard et al. and Harato et al. said that to estimate the LESS scoring system, they employed marker-based technologies such as the Qualisys marker-based motion analysis system. Because marker-based motion analysis tracks and records motion data by using reflective markers put on the subject's body or equipment.

The primary drawback of the marker-based motion system is that it requires time to set up, has limited mobility, and is costly [22, 23].

To address the issues of marker-based motion analysis, storage, delays in estimating findings, and additional human resources, this research proposes a solution by incorporating AI into the LESS scoring system for ACL injury prevention using Computer Vision and Deep Learning. As computer vision algorithms can monitor characteristics on the subject's body in real-time and as a marker-less motion capture system without the requirement for markers [24, 25], it may be used in real-time analysis. This inspired me to work on this issue for my project thesis, and it assisted me in proposing a technique to estimate the LESS score in real-time while eliminating the need to save the video.

3. Methodology

The elaboration of the architectural implementation of the project, the choice of a landmark for each of the LESS items, the estimation of the angle between the landmarks as the coordinates, the explanation of the implementation of the code, the preparation of the data, and finally the comparison and assessment of the accomplishments of the athlete are discussed in this section.

3.1 Project Architecture

The proposed project is composed of a variety of interconnected building blocks, such as obtaining the video, estimating the coordinates for each landmark on the athlete's body, then calculating the angle, and finally doing the analysis and estimating the results. The project's simplified architectural design diagram is shown in Figure 3, and each block's function and significance are described in the subsections.

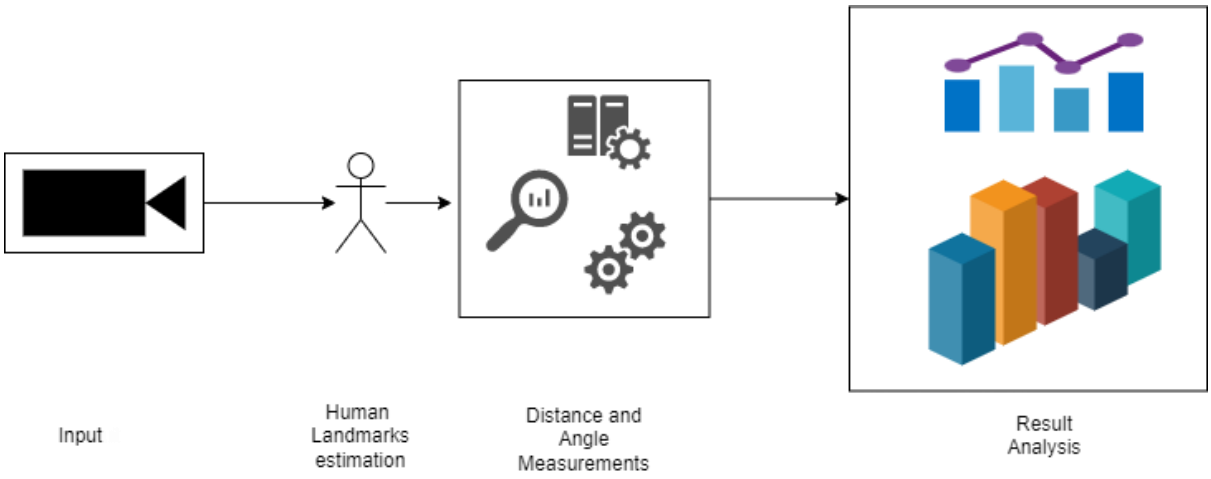

Figure 3 Project Architectural Diagram

3.1.1 Input

To do object tracking, the athlete's video is utilized as the input, and the continuous films are loaded frame by frame. The practice of determining an object's precise location as it is running or through a series of image frames in a video is known as object tracking [7]. The process for tracking an object in a video is shown in Figure 4. Since OpenCV is a free computer vision and machine learning software package, the project's athletes serve as the object to be recognized in each frame and estimate the ROI of humans with the help of OpenCV [8].

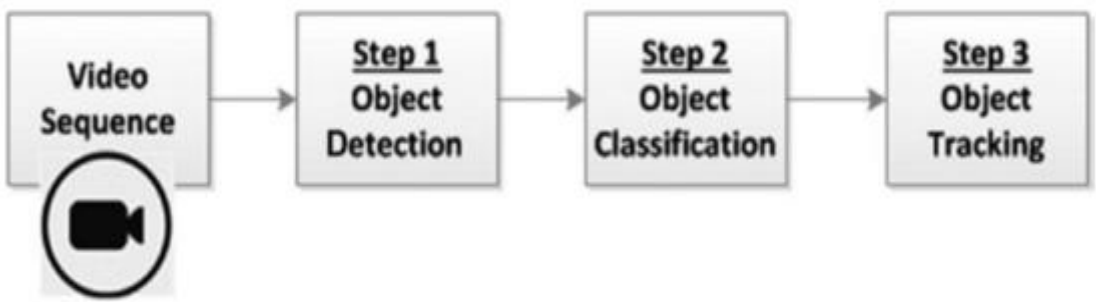

Figure 4 Object Tracking from a video [7]

A video consists of a series of sequential image frames, hence the bounding box application must be extended and applied to each frame. The object being detected must be present in every frame for object tracking to produce the desired results. A strong matching formula can prove that an object is present in both frames. Finding interesting items in the video sequence and grouping their pixels is the initial phase in the object-tracking process. Many techniques concentrate on the identification of moving objects because they are frequently the main source of information. This practice is also known as tracking by detection [7].

3.1.2 Estimating coordinates for each landmark of the Athletes body

We were able to monitor human posture with the use of Google's Real-Time Body Pose Tracking with MediaPipe by using ML to infer 33 2D landmarks of a body from a single frame, or what is known as Blazepose. BlazePose is ideally suited for fitness applications since it precisely localizes more keypoints. Moreover, although our method provides real-time performance on smartphones with CPU inference, existing state-of-the-art approaches generally rely on robust desktop settings for inference. By utilizing GPU inference, BlazePose may execute subsequent ML models, such as face or hand tracking, with super-real-time speed [9].

3.1.3 Estimating the distance and angle between the landmarks

Analytic geometry is used to compute the angle α between the two lines for the coordinate, coordinates are (x1, y1), (x2, y2), and (x3, y3), where

$$\alpha = (\text{atan}(y3, y2, x3, x2) - \text{atan}(y1, y2, x1, x2))$$

Figure 5 depicts the estimated angle between three locations, and Figure 6 represents the angle estimation between 2 landmarks.

Figure 5 Angle estimation of 3 landmarks

Figure 6 Angle estimation of 3 landmarks

According to the analytic geometry is used to compute the angle θ between the two coordinates, coordinates are (x1, y1), and (x2, y2) where

$$\theta = \text{acos}\, x \left(\frac{(y2 - y1) * (-y1)}{\sqrt{(((x2 - x1) ** 2 + (y2 - y1) ** 2) * y1)}}\right) \text{--- (1)}$$

From (1), the angle θ formed by two coordinates was,

$$\alpha = \quad \theta \text{ x } \frac{180}{\pi}$$

The coordinates are (x1, y1), and we need to calculate the distance (d) between the two points (x2, y2)

$$d = \sqrt{((x2-x1)^2+(y2-y2)^2)}$$

Figure 7 illustrates the distance between two landmarks. The Python code for predicting the angle between three coordinates and two coordinates, as well as calculating the distance between the landmarks, is presented in the subsequent section of this chapter.

3.1.4 Analysis

The estimates are based on the parameters of the LESS scoring tool that were created by modifying the BlazePose model. The coordinates of each node in each frame of the movie are the crucial information. Also, the coordinates will continue to change dynamically for each frame in response to the actions of every movement made by the athlete. The primary score components are estimated by calculating the angle and distance using the coordinates [5, 6].

3.2 LESS Operational descriptions for each items

When performing a jump-landing maneuver, the Landing Error Scoring System (LESS) is a field evaluation tool for spotting potentially dangerous movement patterns (also known as "errors"). Using 3-dimensional motion analysis, Padua et al. showed that the LESS has concurrent validity and that high levels of inter- and intra-rater reliability may be achieved [5, 6].

The LESS hasn't, however, been extensively studied as a potential method for screening. Consequently, the aim of our study was to investigate the reliability of the LESS for identifying elite-youth soccer players at risk for ACL injuries. While more movement faults are represented by higher LESS scores, we predicted that in this cohort, ACL damage would be more likely to occur [5, 6].

The general steps to carry out a LESS evaluation are as follows:

1. Prepare the essential tools: You will need a measuring tape or ruler, a LESS scoring sheet, and a camera or video camera to film the landing in order to complete an LESS evaluation.
2. Keep a landing log: Take a side-on recording of the landing you wish to analyze. Make sure you can see the athlete's feet, knees, hips, shoulders, and head, as well as the rest of their body.
3. Take a measurement between the athlete's feet. As the athlete lands, measure the space between his or her feet using a measuring tape or ruler. This distance should be measured starting at the middle of the heel on one foot and ending at the heel on the other.
4. A successful landing: To evaluate the landing, use the LESS score matrix. Six categories are listed on the page, and each is graded as 0 or 1.

- **Initial contact**: When the foot was flat on the ground, the first contact for the knee flexion, hip flexion, trunk flexion, medial knee position, and lateral trunk flexion was obtained. This indicates that the athletes' toe and heel should face the ground to prevent any asymmetries in the feet's landing. Nevertheless, during ankle plantar flexion, the first point of contact between the toe and the ground is considered.

  Furthermore, the operational standards for the first interaction were

  a. Knee Flexion: If the knee was flexed less than 30 degrees at the time of initial contact, a score of 1 was given; otherwise, a score of 0 was given.

b. Hip Flexion: At the starting position, each side's tight should be vertically straight to the hip of the athletes; thus, the knee and hip should be straight to 180 degrees. If this criterion is met, the score is 1, otherwise it is 0.
c. Trunk Flexion: If the athlete's trunk is vertically straight to the hip during the first contact of the trunk flexion and there is no flexion in the upper body, the athlete receives a score of 1, otherwise they receive a zero.
d. Ankle Plantar Flexion: In ankle plantar flexion, as previously stated, the initial contact is taken when either the toe or heel touches the ground surface. In this case, if the toes touch the ground first before the heel (i.e., Toe to Heel), then secure the score of 1; otherwise, score of 0 is assigned, either the landing is flat or heel to toe landing.
e. Medial knee position: The medial knee, together with the ankle, should be facing 180 degrees straight down when the body is in its original position. The ankle and ground are at an angle of 180 degrees with the patella's center of mass. Then 1; otherwise, 0.
f. Lateral trunk flexion: At the initial point of ground contact, if the hip has moved in any way, the result is 1, else it is 0. As shown in Figure 6, if any point moves forward or backward, it indicates that the lateral trunk has been displaced and receives a score of 1, otherwise it receives a score of 0.
g. Initial Symmetrical Contact: The first symmetrical contact is calculated when both feet fall toe to heel at the same moment, in which case the score would be 0, but in the event of an asymmetrical landing in which the toes land before the heel or vice versa, the estimated value would be 1.

- **Stance width:** When an athlete's feet (measured from toe to toe) are farther apart than their shoulders, the stance is seen to be broader. Conversely, if the athlete's feet (measured from toe to toe) are closer together than their shoulders, the stance is deemed to be narrower.
- **Maximum Displacement:** The maximum displacement that an athlete can make after landing is thought to be their upper body and knee flexion. When it comes to the LESS scoring system, each item has specific criteria to be evaluated, such as
    a. Knee flexion displacement: In this case, the requirement to score 1 is that the knee flexes less than 40° between the initial landing and the maximum knee flexion; otherwise, it is a score of 0.
    b. Hip flexion displacement: If the thigh is not in a straight line with the trunk between the beginning and maximum knee displacement, it is regarded as 1, and if it is not, then regarded as 0.
    c. Trunk flexion displacement: The maximum displacement is calculated here when the trunk does not flex more than the difference between the initial contact and the maximum knee displacement, in which case the score would be 1, otherwise 0.
    d. Medial knee displacement: To receive a score of 1, the patella's center must be 180 degrees away from the ankle now the knee is displaced to its maximum.
- **Foot positional rotation:** An internal rotation is one in which the foot is rotated more than 30 degrees inward relative to the optimal foot position between the initial and maximum displacement. An external rotation is one in which the foot is rotated more

than 30 degrees outward relative to the optimal foot position between the initial and maximum displacement.

- **Joint displacement** is estimated by comparing the trunk, hip, and knee displacements of the athletes. This is classified into three categories: 0. Soft: When the athletes' demonstration is huge when all three displacements are considered, 1: Average: When the participants' displacement is on the average level and not at the maximum, 2: Stiff: When participants' displacement is extremely little, the performance is poor.
- The **overall impression** is produced based on the examiner's view on the participants' performance.
    a. Excellent: when the landing performance was gentle and there was no frontal or traversal plane motion,
    b. Average: Meets the fundamental requirements and has an average displacement.
    c. Poor: When there was a lot of frontal and traversal plane motion, or when the landing was firm with no bending.

The examiner also looks at the following fundamental criteria:

a. Body position: This category evaluates the position of the athlete's body upon landing, including their back, hips, knees, and ankles.
b. Arm position: This category evaluates the position of the athlete's arms and hands upon landing.
c. Landing position: This category evaluates the athlete's foot posture upon landing, including the distance between their feet and the angle of their toes.
d. Balance: This category evaluates the athlete's ability to retain their balance after landing.

Landing Error Scoring System Item	Operational Definition of Error	Scoring
Knee flexion: initial contact	The knee is flexed less than 30° at initial contact.	0 = Absent 1 = Present
Hip flexion: initial contact	The thigh is in line with the trunk at initial contact.	0 = Absent 1 = Present
Trunk flexion: initial contact	The trunk is vertical or extended on the hips at initial contact.	0 = Absent 1 = Present
Ankle plantar flexion: initial contact	The foot lands heel to toe or with a flat foot at initial contact.	0 = Absent 1 = Present
Medial knee position: initial contact	The center of the patella is medial to the midfoot at initial contact.	0 = Absent 1 = Present
Lateral trunk flexion: initial contact	The midline of the trunk is flexed to the left or the right side of the body at initial contact.	0 = Absent 1 = Present
Stance width: wide	The feet are positioned greater than shoulder width apart (acromion processes) at initial contact.	0 = Absent 1 = Present
Stance width: narrow	The feet are positioned less than shoulder width apart (acromion processes) at initial contact.	0 = Absent 1 = Present
Foot position: external rotation	The foot is externally rotated more than 30° between initial contact and maximum knee flexion.	0 = Absent 1 = Present
Foot position: internal rotation	The foot is internally rotated more than 30° between initial contact and maximum knee flexion.	0 = Absent 1 = Present
Symmetric initial foot contact: initial contact	One foot lands before the other foot or 1 foot lands heel to toe and the other foot lands toe to heel.	0 = Absent 1 = Present
Knee-flexion displacement	The knee flexes less than 45° between initial contact and maximum knee flexion.	0 = Absent 1 = Present
Hip-flexion displacement	The thigh does not flex more on the trunk between initial contact and maximum knee flexion.	0 = Absent 1 = Present
Trunk-flexion displacement	The trunk does not flex more between initial contact and maximum knee flexion.	0 = Absent 1 = Present
Medial-knee displacement	At the point of maximum medial knee position, the center of the patella is medial to the midfoot.	0 = Absent 1 = Present
Joint displacement	Soft: the participant demonstrates a large amount of trunk, hip, and knee displacement.	0 = Soft
	Average: the participant has some, but not a large amount of, trunk, hip, and knee displacement.	1 = Average
	Stiff: the participant goes through very little, if any, trunk, hip, and knee displacement.	2 = Stiff
Overall impression	Excellent: the participant displays a soft landing with no frontal-plane or transverse-plane motion.	0 = Excellent
	Average: all other landings.	1 = Average
	Poor: the participant displays large frontal-plane or transverse-plane motion, or the participant displays a stiff landing with some frontal-plane or transverse-plane motion.	2 = Poor

e. Landing stability: This category evaluates the athlete's ability to absorb the shock of landing.

f. Steps: This category evaluates any further steps the athlete makes after landing.

To get the total LESS score, add the scores from each category. A perfect landing would receive a score of 0, but a landing with several faults would receive a higher score. All of these grading criteria were taken from the journal published by the author Padua.

5. Provide input: When you've scored the landing, give feedback to the athlete. Find places where they may enhance their technique and give assistance on how to do so.
6. Rep the procedure: To achieve a more accurate assessment of the athlete's landing technique, you may need to record and evaluate many landings.

By following these steps, you may utilize the LESS evaluation to identify areas for improvement in an athlete's landing technique and give specific comments to assist them improve their performance [5,6].

Each individual LESS item's operational Definitions are stated in Figure 8 [5].

3.3 Landmarks for each LESS's operational Item:

1. Knee flexion displacement:

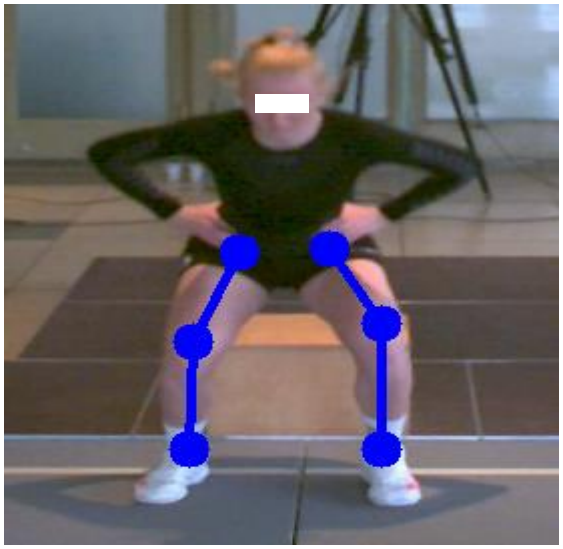

Figure 9 Font-view: Knee Flexion

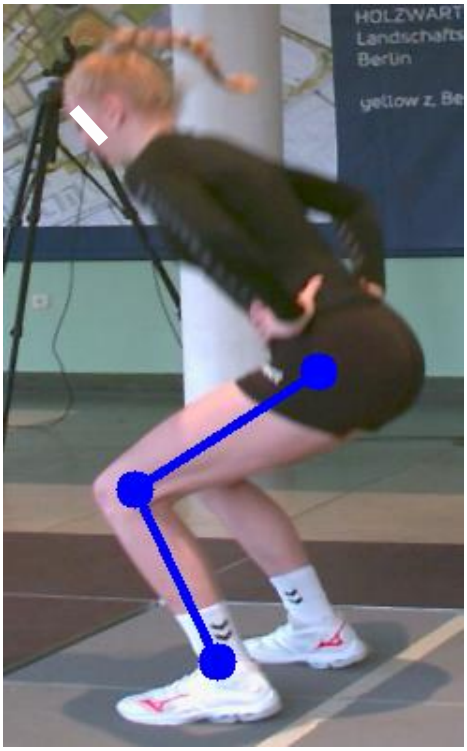

Figure 10 Side-view: Knee Flexion

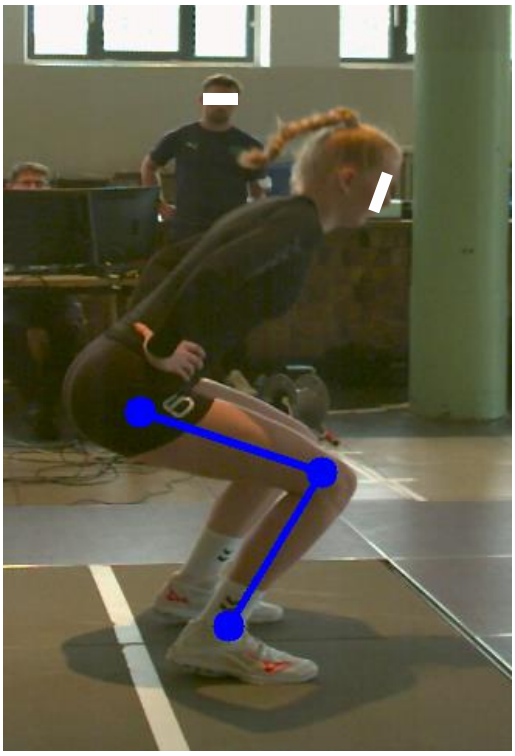

Figure 11 Side-view: Knee Flexion

The knee flexion is evaluated using the BlazePose with the left:(23,25,27) and right: (24,26,28) landmarks. Figure 9 depicts the front perspective, while Figures 10 and 11 depict the side views.

2. Hip Flexion Displacement:

The hip flexion is evaluated using the BlazePose with the left:(23,25), and right:(24,26) landmarks. Figure 12 depicts the front view.

Figure 12 Front-view: Hip Flexion

3. Trunk Flexion Displacement

The trunk flexion is evaluated using the BlazePose with the left: (11, 23, 25), and right: (12, 24, 26) landmarks. Figure 13 depicts the front perspective, while Figures 14 and 15 depict the side views.

Figure 13 Front-view: Trunk Flexion

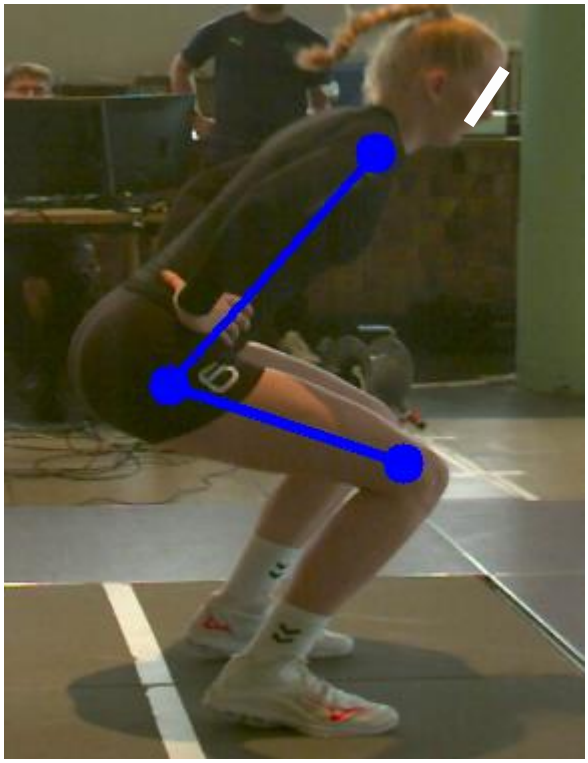

Figure 14 Side-view: Trunk Flexion : Right

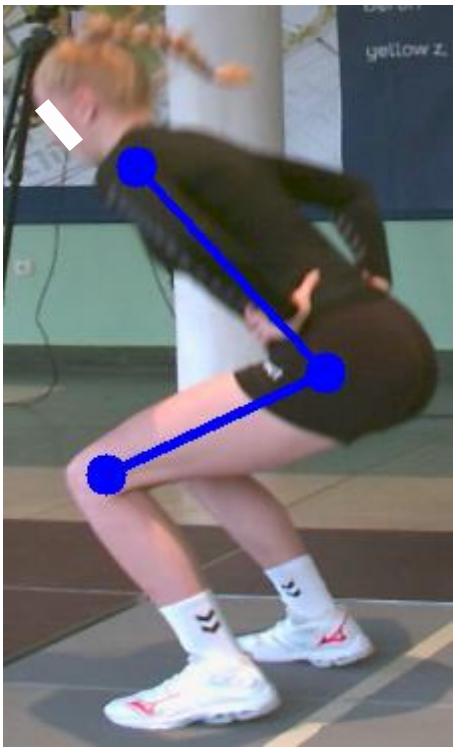

Figure 15 Side-view: Trunk Flexion : Left

4. Ankle Plantar Flexion:

The BlazePose with the landmarks for the left:(25,27,31), and right:(26,28,32) is used to estimate the Ankle Plantar Flexion. The side perspectives are depicted in Figures 16 and 17.

Figure 16 Side-view: Ankle Flexion : Right

Figure 17 Side-view: Hip Flexion : Left

5. Medial Knee Displacement

The BlazePose with the landmarks for the left:(25,27), and right:(26,28) is used to estimate the medial knee flexion. Figure 18 depicts the front view.

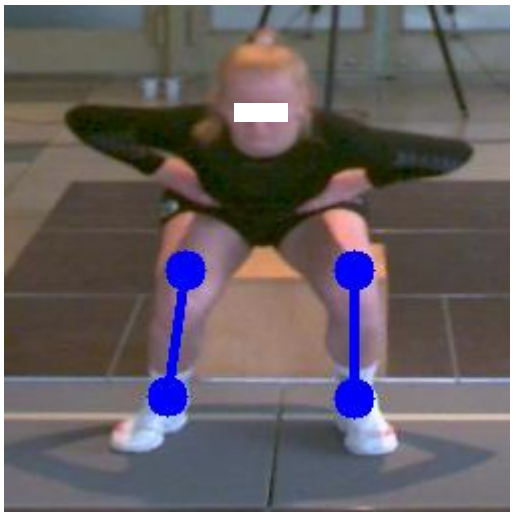

Figure 18 Front -view: Medial Knee Flexion

6. Lateral Trunk Flexion:

The BlazePose with the landmarks for the left:(11,23), and right:(12, 24) is used to estimate the Lateral Trunk flexion. Figure 19 depicts the front view.

Figure 19 Front-view: Laternal Trunk

7. Stance width:

The BlazePose is used to determine the Stance Width, given the shoulder width:(11,12) and foot distance: (31,32). Figure 20 depicts the front view.

Figure 20 Front-view Stance Width

8. Foot Position:

The BlazePose with the landmarks for the left:(29,31), and right:(30, 32) is used to estimate the foot location. Figure 21 depicts the front view.

Figure 21 Front-view: Foot Rotation

9. Symmetrical Foot Contact:

The BlazePose is used to estimate the Symmetric Foot Contact, which includes landmarks for the Ankles (27,28), Heels (29,30), and Toes (31,32). The front perspective is seen in Figure 22.

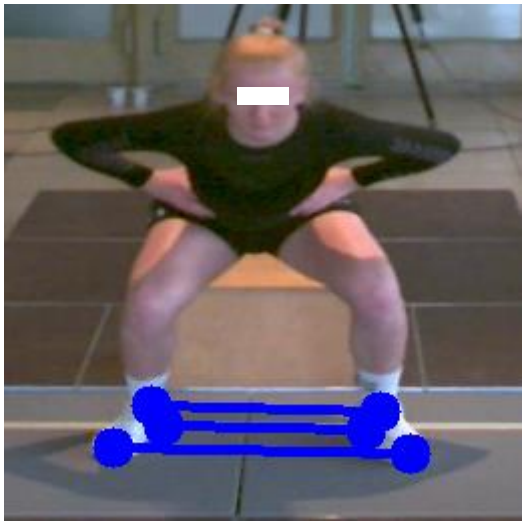

Figure 22 Front-view: Symmetrical Foot-Land Contact

3.4 Python code

Python is an established programming language because to its extensive libraries, ease of use, huge community, cross-platform support, and compatibility for Computer Vision. Python also has a plethora of libraries, including that of OpenCV, for example, is a robust library that provides a variety of image processing and computer vision methods [26, 27]. To take advantage of all the benefits of Python and computer vision, I used Python as the programming language in this project to estimate the degree of flexion, primarily when determining the degree of displacement owing to the movement or action made by the athletes when taking on the challenge of jumping. Furthermore, this code is quite simple to use for calculating the angle and distance between the coordinates, and it is designed to be straightforward to understand. The programming logic utilized in this project could be accessed in Appendix B.

3.5 Data Generation

Every one of the parameter estimates are derived and represented in the video using the OpenCV library, as well as Google's BlazePose for estimating the movements of the people taking part, and all of the angle estimations are generated using the analytic geometry and distance formula and a python program, which is referenced in Appendix B. The figure calculated was the extent of dislocation or flexibility of the player's specific anatomical component. The resulting numbers are then manually recorded in an excel spreadsheet file and utilized for further estimate, analysis, and comparison between the outcomes.

Below are some specifics on how each LESS operational item's outcome is achieved.

1. Knee Flexion: The value predicted using analytic geometry is then diminished by 180° in both directions to produce the proper calculation.
2. Hip Flexion: The values determined using analytic geometry is then lowered by 180° on both sides to achieve the proper calculation.
3. Trunk Flexion: The value calculated using analytic geometry is then reduced with 360° on the left side to make the angle estimates balance to the right side.
4. Ankle Plantar Flexion: The value computed using analytic geometry is then subtracted with 270° on the left side and 90° on the right side, primarily to make the angle calculation symmetrical in both directions.
5. Medial Knee Position: The calculation using analytic geometry is then subtracted by 180° on both sides to achieve the proper estimation.
6. Lateral Trunk Flexion: The value predicted using analytic geometry is then subtracted with 170° on both sides to correct the angle estimation based on the upper section of the human anatomy.
7. Stance Width: The distance in this case is calculated using the Euclidean distance in two dimensions.
8. Foot Position: To acquire the reliable estimate, the value calculated using analytic geometry is subtracted by 180° on both sides.
9. Symmetric Foot: The value calculated using analytic geometry is then subtracted by 90° for each estimation, such as ankle, toes, and heel.

10. Displacement at the joint and overall: These findings are assessed based on individual evaluation, however only the greatest displacement and satisfaction of all the LESS criteria are considered.

The documents for the result obtained using computer vision and the result obtained using Qualisys 3D can be found below.

3.6 Qualisys 3D analysis and Computer Vision 2D analysis Performance and Result Comparison

Even though the LESS score was estimated in 2D in this project, the total final estimation of the OpenCV was nearly equivalent to the Qualisys 3D analysis, and OpenCV was far more efficient in predicting the Trunk displacement and Ankle plantar displacement than Qualisys. While the Qualisys employs 3D to anticipate the angle measurement, it cannot predict all types of landings, and when it comes to the Ankle, the accuracy in diagnosing the direction of the Foot rotation was a little insufficient. Resultantly, OpenCV was far ahead in determining all the human posture projections.

One of most notable difference between Qualisys and OpenCV was in identifying the Medial-Knee displacement, in which the similarity was 50% owing to the processing of the object in different dimensions. Overall, the computer vision algorithm experience was immersive and participatory, akin towards the 3D analysis. The outcomes of both analyses are recorded in an excel file, which is included 

for reference.

4. Result

This experiment analyzed a total of ten participants, all of them were girls who played in under 17 and 19 teams in national volleyball, with an average height of 1.79 m and a weight of roughly 67.4 kg. It's evident that most of the players whose Low score was influenced during the landing, particularly the initial landing and regulating them while executing the challenge, were affected during the landing. A few people were capable of controlling their performance; however their knee flexion had not been stronger than some others.

The key issue in obtaining the result was pinpointing the initial contact because there is no symmetrical foot landing and most of the participants' first initial surface touch was on their right leg, followed by their left leg. To summarize the end results, 1 person outperformed all others and had a reliable performance, 5 participants' achievement was satisfactory with an

average landing to a little stiff, and 4 individuals' initial and total performance was stiff and less than the average. Consequently, the final comparison result excel sheet was supplied in section 3.5.

5. Discussion

This chapter includes the most significant conclusions from the project's experiments, as well as the major challenges and technological limits encountered. Additionally highlighted and thoroughly defined the future advancement of how some of the constraints can be addressed.

5.1 Key findings

Without any of the standards of the LESS scoring system, it is very difficult to quantify each flexion and displacement of the athletes based on examining all of their performance. In the absence of Computer Vision, examining the videos alone was unable to evaluate every jump. Also, with the help of Computer Vision, the algorithm made video analysis much more straightforward and effective by tweaking the FPS and identifying the landmark, as well as integrating each of the location, making this endeavor a greater success and reducing the need for an extra principal investigator to assess the results of the contestants.

Upon examining the video, it is apparent that many of the erroneous activities resulted in a disastrous landing; a few respondents were sliding to one side by putting extra weight on their dominant thigh and shoulder. Also, I observed how most of the participants placed greater importance on their dominant side, which leads to asymmetry while performing. And while the stances of a few people were appropriate, but the knee and trunk flexion were quite modest, leading to the conclusion that the performance was not up to standard when compared to everybody else.

5.2 Limitations and Future Improvements:

There were a few obstacles faced while working on my project thesis, primarily due to technological issues and the operating norms of the LESS scoring method, and these difficulties are detailed below,

1. Exploration of data: The main difficulty I confront is that I must complete all of the analysis in the recorded video because there is no way to invite athletes and undertake real-time analysis using digital cameras.
2. Constraints of the scoring system: The LESS rating method is primarily on on the evaluator's subjective judgement of movement quality, which can result in variability in score across evaluators. Moreover, the LESS scoring system only evaluates a subset of landing mechanics, such as knee valgus and trunk flexion, and may miss key parts of landing mechanics. Also, there is no further advice to undertake any workout in order to overcome the injuries and perform better in the future test. Such recommendations

are not incorporated to the LESS score system for participants who have problems with the Jump landing test.

3. Constraints of computer vision: The precision in detecting the athlete's specific landmarks fluctuated from high to low due to the athlete's movements in each frame of the video while doing the evaluation. OpenCV is intended solely for 2D image processing and requires native support for 3D vision. Although OpenCV has some deep learning functionality, it is not as extensive as other deep learning frameworks such as TensorFlow or PyTorch.
4. Additionally, there is no application to input and maintain the findings from the evaluation in the database, and jotting it down in an excel spreadsheet is challenging to keep track of.

Several of the difficulties highlighted above may be able to be addressed during my master's thesis in the following months. Figure 23 depicts the system's architectural diagram, and the essential important principles will be presented in detail below

1. The idea is to subsequently create an application, either a desktop or mobile application, that will save the data and evaluate the results of the participants.
2. Furthermore, by incorporating AI into the application, the application may be able to recommend certain treatment or therapists practice by each of the participants on a regular basis in order to overcome the irregularity movement and make them physically and intellectually strong.

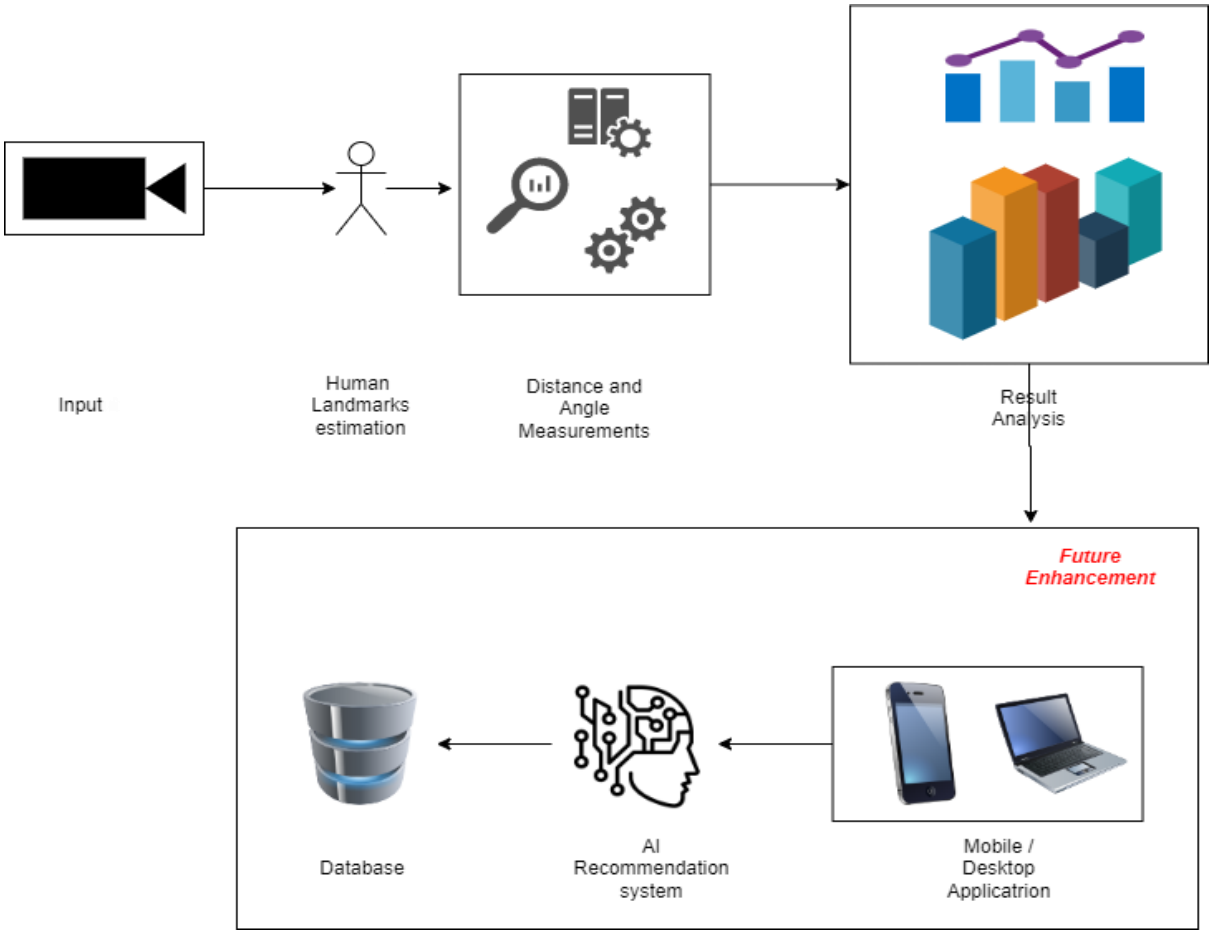

Figure 23 System Architecture for Future Implementation

3. The data would additionally be saved in the database for future research, dashboard presentation, and regular streak monitoring of their productivity.

6. Conclusion

The results revealed that Computer Vision may be able to use LESS scoring to forecast the landing of sports players. The above project's major goal is to conclude that the LESS analysis can be performed using the AI algorithm and can estimate everything in 2 dimensions without implementing any complex dimensional space, hence removing the need to depend on extra expert aid to analyze the jump-landing. Furthermore, this research demonstrates that the evaluations may be able to execute in both the projectional and lateral perspectives, although the predicted angle's ranges may differ owing to dimension space.

References

[1] Gollapudi, S. (2019). Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs (1st ed.) [ISBN-13 (pbk): 978-1-4842-4260-5]. Apress. https://doi.org/10.1007/978-1-4842-4261-2

[2] Shakya, S., Du, K. L., & Haoxiang, W. (2022). Proceedings of Second International Conference on Sustainable Expert Systems: ICSES 2021 (Vol. 351). Springer Publishing. https://doi.org/10.1007/978-981-16-7657-4

[3] Anilkumar, A., K.T., A., Sajan, S., & K.A., S. (n.d.). Pose Estimated Yoga Monitoring System. In International Conference on IoT Based Control Networks & Intelligent Systems - ICICNIS 2021. International Conference on IoT Based Control Networks & Intelligent Systems - ICICNIS 2021, Kerala, India. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3882498

[4] Google LLC [Google's MediaPipe on GitHub]. (n.d.). MediaPipe Pose. Retrieved February 26, 2023, from https://google.github.io/mediapipe/solutions/pose.html

[5] Padua, D. A., DiStefano, L. J., Beutler, A. I., De La Motte, S. J., DiStefano, M. J., & Marshall, S. W. (2015). The Landing Error Scoring System as a Screening Tool for an Anterior Cruciate Ligament Injury–Prevention Program in Elite-Youth Soccer Athletes. Journal of Athletic Training, 50(6), 589–595. https://doi.org/10.4085/1062-6050-50.1.10

[6] Padua, D. A., Boling, M. C., DiStefano, L. J., Onate, J. A., Beutler, A. I., & Marshall, S. W. (2011). Reliability of the Landing Error Scoring System-Real Time, a Clinical Assessment Tool of Jump-Landing Biomechanics. Journal of Sport Rehabilitation, 20(2), 145–156. https://doi.org/10.1123/jsr.20.2.145

[7] Gollapudi, S. (2019). Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs. Apress. https://doi.org/10.1007/978-1-4842-4261-2

[8] OpenCV. (n.d.). OpenCV. Retrieved March 7, 2023, from https://opencv.org/

[9] Valentin Bazarevsky, Ivan Grishchenko, & Google LLC. (2020, August 13). On-device, Real-time Body Pose Tracking with MediaPipe BlazePose. Retrieved March 7, 2023, from https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html

[10] ATan2 function—ArcGIS Pro | Documentation. (n.d.). ArcGIS Pro. Retrieved March 7, 2023, from https://pro.arcgis.com/en/pro-app/latest/help/analysis/raster-functions/atan2-function.htm#:~:text=ATan2%20function%20converts%20rectangular%20coordinates,of%20rectangular%20to%20polar%20coordinates.&text=(where%20%CE%B8%20is%20the%20angle,matrix%20(based%20on%20sign)

[11] Lobo, A., Lobo, J., Joy, A., & Mirza, I. A. (2022). Yoga Correction Using Machine Learning. 2022 2nd Asian Conference on Innovation in Technology (ASIANCON). https://doi.org/10.1109/asiancon55314.2022.9909255

[12] Kwon, Y., & Kim, D. (2022). Real-Time Workout Posture Correction using OpenCV and MediaPipe. Journal of Korean Institute of Information Technology, 20(1). https://doi.org/10.14801/jkiit.2022.20.1.199

[13] Danielsson, P. (1980). Euclidean distance mapping. Computer Graphics and Image Processing, 14(3), 227–248. https://doi.org/10.1016/0146-664x(80)90054-4

[14] Harato, K., Morishige, Y., Niki, Y., Kobayashi, S., & Nagura, T. (2021). Fatigue and recovery have different effects on knee biomechanics of drop vertical jump between female collegiate and recreational athletes. Journal of Orthopaedic Surgery and Research, 16(1). https://doi.org/10.1186/s13018-021-02893-6

[15] Hewett, T. E., Myer, G. D., Ford, K. R., Heidt, R. S., Colosimo, A. J., McLean, S. G., Van Den Bogert, A. J., Paterno, M. V., & Succop, P. (2005). Biomechanical Measures

of Neuromuscular Control and Valgus Loading of the Knee Predict Anterior Cruciate Ligament Injury Risk in Female Athletes: A Prospective Study. American Journal of Sports Medicine, 33(4), 492–501. https://doi.org/10.1177/0363546504269591
[16] Hewett, T. E., Myer, G. D., Kiefer, A. W., & Ford, K. R. (2015). Longitudinal Increases in Knee Abduction Moments in Females during Adolescent Growth. Medicine and Science in Sports and Exercise, 47(12), 2579–2585. https://doi.org/10.1249/mss.0000000000000700
[17] Padua, D. A., Marshall, S. W., Boling, M. C., Thigpen, C. A., Garrett, W. E., & Beutler, A. I. (2009). The Landing Error Scoring System (LESS) Is a Valid and Reliable Clinical Assessment Tool of Jump-Landing Biomechanics. American Journal of Sports Medicine, 37(10), 1996–2002. https://doi.org/10.1177/0363546509343200
[18] Hewett, T. E., Ford, K. R., Hoogenboom, B. J., & Myer, G. D. (2010). Understanding and preventing acl injuries: current biomechanical and epidemiologic considerations - update 2010. North American Journal of Sports Physical Therapy : NAJSPT. https://www.rehabeducation.com/wp-content/uploads/2015/02/NAJSPTHewettACL.pdf
[19] Haddas, R., James, C. R., & Hooper, T. L. (2015). Lower Extremity Fatigue, Sex, and Landing Performance in a Population With Recurrent Low Back Pain. Journal of Athletic Training, 50(4), 378–384. https://doi.org/10.4085/1062-6050-49.3.61
[20] Haddas, R., Sawyer, S. F., Sizer, P. S., Brooks, T. J., Chyu, M., & James, C. R. (2017). Effects of Volitional Spine Stabilization and Lower-Extremity Fatigue on the Knee and Ankle During Landing Performance in a Population With Recurrent Low Back Pain. Journal of Sport Rehabilitation. https://doi.org/10.1123/jsr.2015-0171
[21] Dar, G., Yehiel, A., & Benzoor, M. C. (2019). Concurrent criterion validity of a novel portable motion analysis system for assessing the landing error scoring system (LESS) test. Sports Biomechanics, 18(4), 426–436. https://doi.org/10.1080/14763141.2017.1412495
[22] Eckard, T. G., Miraldi, S. F. P., Peck, K. Y., Posner, M. T., Svoboda, S. J., DiStefano, L. J., Padua, D. A., Marshall, S. W., & Cameron, K. L. (2021). Association Between Automated Landing Error Scoring System Performance and Bone Stress Injury Risk in Military Trainees. Journal of Athletic Training, 57(4), 334–340. https://doi.org/10.4085/1062-6050-0263.21
[23] Harato, K., Morishige, Y., Kobayashi, S., Niki, Y., & Nagura, T. (2022). Biomechanical features of drop vertical jump are different among various sporting activities. BMC Musculoskeletal Disorders, 23(1). https://doi.org/10.1186/s12891-022-05290-0
[24] Lawin, F. J., Byström, A., Roepstorff, C., Rhodin, M., Almlöf, M., Silva, M., Andersen, P. H., Kjellström, H., & Hernlund, E. (2023). Is Markerless More or Less? Comparing a Smartphone Computer Vision Method for Equine Lameness Assessment to Multi-Camera Motion Capture. Animals, 13(3), 390. https://doi.org/10.3390/ani13030390
[25] Needham, L., Evans, M., Cosker, D., & Colyer, S. L. (2021). Development, evaluation and application of a novel markerless motion analysis system to understand push-start technique in elite skeleton athletes. PLOS ONE, 16(11), e0259624. https://doi.org/10.1371/journal.pone.0259624
[26] OpenCV: OpenCV-Python Tutorials. (n.d.). Retrieved November 21, 2022, from https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
[27] Welcome to Python.org. (n.d.). Python.org. Retrieved November 21, 2022, from https://www.python.org/doc/
[28] Z. (n.d.). Free Online Diagram Editor. diagrameditor.com. Retrieved March 26, 2023, from https://www.diagrameditor.com/

Appendix A. BlazePose 33 landmarks

0. Nose
1. Left eye inner
2. Left eye
3. Left eye outer
4. Right eye inner
5. Right eye
6. Right eye outer
7. Left ear
8. Right ear
9. Mouth left
10. Mouth right
11. Left shoulder
12. Right shoulder
13. Left elbow
14. Right elbow
15. Left wrist
16. Right wrist
17. Left pinky #1 knuckle
18. Right pinky #1 knuckle
19. Left index #1 knuckle
20. Right index #1 knuckle
21. Left thumb #2 knuckle
22. Right thumb #2 knuckle
23. Left hip
24. Right hip
25. Left knee
26. Right knee
27. Left ankle
28. Right ankle
29. Left heel
30. Right heel
31. Left foot index
32. Right foot index

Appendix B. Python code

PoseEstimationModule.py

```
#Importing libraries
import cv2
import mediapipe as mp
import math
##

class poseEstimation():

    # Initialization Method
    def __init__(self, static_image_mode=False,
                 model_complexity=1,
                 smooth_landmarks=True,
                 enable_segmentation=False,
                 smooth_segmentation=True,
                 min_detection_confidence=0.5,
                 min_tracking_confidence=0.5):
        self.static_image_mode = static_image_mode
        self.model_complexity = model_complexity
        self.smooth_landmarks = smooth_landmarks
        self.enable_segmentation = enable_segmentation
        self.smooth_segmentation = smooth_segmentation
        self.min_detection_confidence = min_detection_confidence
        self.min_tracking_confidence = min_tracking_confidence

        self.mpDraw = mp.solutions.drawing_utils
        self.mpPose = mp.solutions.pose
        self.pose = self.mpPose.Pose(self.static_image_mode,
                                     self.model_complexity,
                                     self.smooth_landmarks,
                                     self.enable_segmentation,
                                     self.smooth_segmentation,
                                     self.min_detection_confidence,
                                     self.min_tracking_confidence)
    ##

    # Detecting the coordinates of each frame and storing back as frame
    # for further estimation
    def detectCoordinates(self, video, draw=True):

        # Converting the video from BGR to RGB to make it compatible for
        # Mediapipe package
        videoRGB = cv2.cvtColor(video, cv2.COLOR_BGR2RGB)
        ##

        # Pose detection by processing
        self.processed = self.pose.process(videoRGB)
        self.landmarks = self.processed.pose_landmarks
        ##

        # If pose joins are present then we have to draw the landmarks of
        # the joins
```

```
        if self.landmarks:
            if draw:
                self.mpDraw.draw_landmarks(
                    video, self.landmarks, self.mpPose.POSE_CONNECTIONS)

        return video
        ##
    ##

    # Fetching each coordinates for each frame and storing it in
    # an array, mainly this array helps us to check that in each frame
    # we determine the angle for the estimation
    def findCoordinates(self, video, draw=True):
        self.coordinates = []

        # If pose joins are present then we have to draw the landmarks of
        # the joins
        if self.landmarks:
            for index, lm in enumerate(self.landmarks.landmark):
                height, width, channel = video.shape
                # To convert the ratio of the image (in decimal) to pixel
value
                x, y = int(lm.x*width), int(lm.y*height)
                ##

                # Appending all the joints coordinates into an array
                self.coordinates.append([index, x, y])
                ##

                # To mark the coordinates into the frame
                if draw:
                    cv2.circle(video, (x, y), 5, (255, 0, 0), cv2.FILLED)
                ##
        return self.coordinates
        ##
    ##

    # Determining the angle for the future verification whether it is
    # successfull or not
    def toFindAngleThreeCoordinates(self,frame,i1,i2,i3, draw = True):

        # To get the coordinates/landmarks of 3 points
        x1,y1 = self.coordinates[i1][1:]
        x2,y2 = self.coordinates[i2][1:]
        x3,y3 = self.coordinates[i3][1:]
        ##

        # Angle Calculation
        # Also needs to convert to degree from radian
        angle = math.degrees(math.atan2(y3-y2,x3-x2)- math.atan2(y1-y2,x1-
x2))
        ##

        # Incase angle goes to negative then we must do some adjustment by
        # adding 360 to it
        if angle <0:
            angle += 360
```

```
        ##

        # Connect all the landmarks with the coordinates into each frame
        if draw:
            cv2.line(frame, (x1,y1),(x2,y2),(255,0,0),3)
            cv2.line(frame, (x3,y3),(x2,y2),(255,0,0),3)
            cv2.circle(frame, (x1,y1), 10, (255,0,0), cv2.FILLED)
            cv2.circle(frame, (x2,y2), 10, (255,0,0), cv2.FILLED)
            cv2.circle(frame, (x3,y3), 10, (255,0,0), cv2.FILLED)
        ##

        return angle
    ##

    # Calculate angle for pose inclination
    def findAngleTwoCoordinates(self,frame,i1,i2, draw = True):

        # To get the coordinates/landmarks of 2 points
        x1,y1 = self.coordinates[i1][1:]
        x2,y2 = self.coordinates[i2][1:]
        ##

        theta = math.acos( (y2 -y1)*(-y1) / (math.sqrt(
            (x2 - x1)**2 + (y2 - y1)**2 ) * y1) )
        angle = int(180/math.pi)*theta

        # Connect all the landmarks with the coordinates into each frame
        if draw:
            cv2.line(frame, (x1,y1),(x2,y2),(255,0,0),3)
            cv2.circle(frame, (x1,y1), 10, (255,0,0), cv2.FILLED)
            cv2.circle(frame, (x2,y2), 10, (255,0,0), cv2.FILLED)
        ##

        return angle
    ##

    # Calculate the distance between two points
    def findDistanceTwoCoordinates(self, frame, i1, i2, draw = True):

        # To get the coordinates/landmarks of 2 points
        x1,y1 = self.coordinates[i1][1:]
        x2,y2 = self.coordinates[i2][1:]
        ##

        distance = math.sqrt((x2-x1)**2+(y2-y1)**2)

        # Connect all the landmarks with the coordinates into each frame
        if draw:
            cv2.line(frame, (x1,y1),(x2,y2),(255,0,0),3)
            cv2.circle(frame, (x1,y1), 10, (255,0,0), cv2.FILLED)
            cv2.circle(frame, (x2,y2), 10, (255,0,0), cv2.FILLED)
        ##

        return distance
    ##
```

```
AngleEstimation.py

# Importing libraries
import cv2
import PoseEstimationModule as pm

def main(video, pEstimate, finalCheck, partSelection,viewSelection =
0,rotate = False):


    maxr = 0
    maxl = 0
    maxrt = 0
    maxlt = 0

    def angleDisplay(text, angle,coordinate1 = (30,50),
                     coordinate2 = (30,120)):
        #cv2.rectangle(frame, (0, 150), (1280, 0), (0, 0, 0), cv2.LINE_AA)
        cv2.putText(frame, str(text), coordinate1,
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 204, 102), 5)

        cv2.putText(frame, str(angle), coordinate2,
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 10)

    # Looping through each frame of the video
    while True:
        # Checking whether the frame has valid image and getting
        # the current frame
        success, frame = video.read()

        # Main checkpoint to verify whether the frame obtained is not none
        # mainly to avoid the assertion error while in the end of the video
        # if no frame is left the window opened are closed
        if frame is not None:
            # Resizing the video
            if partSelection == 4:
                frame = cv2.resize(frame, (900, 900))
                # Rotating the video if required
                if rotate:
                    frame = cv2.rotate(frame,
cv2.ROTATE_90_COUNTERCLOCKWISE)
            elif partSelection == 1 or partSelection == 3:
                if viewSelection != 3:
                    frame = cv2.resize(frame, (900, 820))
                    # Rotating the video if required
                    if rotate:
                        frame = cv2.rotate(frame,
                                           cv2.ROTATE_90_COUNTERCLOCKWISE)
                else:
                    frame = cv2.resize(frame, (1280, 720))
            else:
                frame = cv2.resize(frame, (1280, 720))
            ##

            # Detecting the coordinates of each frame and storing back as
frame
```

```
            # for further estimation
            frame = pEstimate.detectCoordinates(frame, False)
            ##

            # Fetching each coordinates for each frame and storing it in
            # an array, mainly this array helps us to check that in each
frame
            # we determine the angle for the estimation
            xyList = pEstimate.findCoordinates(frame, False)
            ##

            # Looping through the array of coordination for the detected
            # coordinates
            if len(xyList) != 0:

                # Estimating angle to verify whether it is successfull or
not
                # Knee Flexion
                if partSelection == 1:

                    if viewSelection == 1:
                        # Angle for Right Side Leg
                        angleRightBottom =
pEstimate.toFindAngleThreeCoordinates(frame, 24, 26, 28, True)
                        if maxr < abs(round(angleRightBottom - 180)):
                            maxr = abs(round(angleRightBottom - 180))
                        else:
                            maxr = maxr
                        angleDisplay('Knee flexion: Right' ,
abs(round(angleRightBottom - 180)),(30,50),(30,120))
                        angleDisplay('Maximum Knee Displacement: Right' ,
maxr,(340,50),(340,120))

                    elif viewSelection == 2:
                        # Angle for Left Side Leg
                        angleLeftBottom =
pEstimate.toFindAngleThreeCoordinates(frame, 23, 25, 27, True)
                        if maxl < abs(round(angleLeftBottom -180)):
                            maxl = abs(round(angleLeftBottom -180))
                        else:
                            maxl = maxl
                        angleDisplay('Knee flexion: Left' ,
abs(round(angleLeftBottom -180)),(30,50),(30,120))
                        angleDisplay('Maximum Knee Displacement: Left ' ,
maxl,(340,50),(340,120))

                    else:
                        # Angle for Left Side Leg
                        angleLeftBottom =
pEstimate.toFindAngleThreeCoordinates(frame, 23, 25, 27, True)
                        # Angle for Right Side Leg
                        angleRightBottom =
pEstimate.toFindAngleThreeCoordinates(frame, 24, 26, 28, True)
                        if maxl < abs(round(angleLeftBottom -180)):
                            maxl = abs(round(angleLeftBottom -180))
                        else:
                            maxl = maxl
```

```
                        if maxr < abs(round(angleRightBottom - 180)):
                            maxr = abs(round(angleRightBottom - 180))
                        else:
                            maxr = maxr
                        angleDisplay('Knee flexion: Left' ,
abs(round(angleLeftBottom -180)),(30,50),(30,120))
                        angleDisplay('Knee flexion: Right' ,
abs(round(angleRightBottom - 180)),(340,50),(340,120))
                        angleDisplay('Maximum Knee Displacement: Left ' ,
maxl,(640,50),(640,120))
                        angleDisplay('Maximum Knee Displacement: Right' ,
maxr,(940,50),(940,120))
                ##
                # Hip Flexion
                elif partSelection == 2:
                    # Angle for Left Side Thigh
                    angleLeftThigh =
pEstimate.findAngleTwoCoordinates(frame, 23, 25, True)
                    # Angle for Right Side Thigh
                    angleRightThigh =
pEstimate.findAngleTwoCoordinates(frame, 24, 26, True)

                    if maxl < abs(round(angleLeftThigh - 180)):
                        maxl = abs(round(angleLeftThigh - 180))
                    else:
                        maxl = maxl
                    if maxr < abs(round(angleRightThigh - 180)):
                        maxr = abs(round(angleRightThigh - 180))
                    else:
                        maxr = maxr
                    angleDisplay('Hip flexion: Left' ,
abs(round(angleLeftThigh - 180)),(30,50),(30,120))
                    angleDisplay('Hip flexion: Right' ,
abs(round(angleRightThigh - 180)),(340,50),(340,120))
                    angleDisplay('Maximum Hip Displacement: Left ' ,
maxl,(640,50),(640,120))
                    angleDisplay('Maximum Hip Displacement: Right' ,
maxr,(940,50),(940,120))
                ##
                # Trunk Flexion
                elif partSelection == 3:
                    if viewSelection == 1:
                        # Angle for Right Side Trunk
                        angleRightTrunk =
pEstimate.toFindAngleThreeCoordinates(frame, 12, 24,26, True)
                        if maxr == 0:
                            maxr = round(angleRightTrunk)
                        else:
                            if maxr > abs(round(angleRightTrunk)):
                                maxr = abs(round(angleRightTrunk))
                            else:
                                maxr = maxr
                        angleDisplay('Trunk flexion' ,
(round(angleRightTrunk)),(30,50),(30,120))
                        angleDisplay('Maximum Hip Displacement: Right' ,
maxr,(340,50),(340,120))
                    elif viewSelection == 2:
```

```
                        # Angle for Left Side Trunk
                        angleLeftTrunk =
pEstimate.toFindAngleThreeCoordinates(frame, 11, 23, 25, True)
                        if maxl == 0:
                            maxl = round(360- angleLeftTrunk)
                        else:
                            if maxl > abs(round(360 - angleLeftTrunk)):
                                maxl = abs(round(360 - angleLeftTrunk))
                            else:
                                maxl = maxl
                        angleDisplay('Trunk flexion' , (round(360 -
angleLeftTrunk)),(30,50),(30,120))
                        angleDisplay('Maximum Hip Displacement: Left ' ,
maxl,(340,50),(340,120))
                    else:
                        # Angle for Left Side Trunk
                        angleLeftTrunk =
pEstimate.toFindAngleThreeCoordinates(frame, 11, 23, 25, True)
                        # Angle for Right Side Trunk
                        angleRightTrunk =
pEstimate.toFindAngleThreeCoordinates(frame, 12, 24, 26, True)
                        angleDisplay('Trunk flexion: Left' ,
(round(angleLeftTrunk)),(30,50),(30,120))
                        angleDisplay('Trunk flexion: Right' ,
(round(angleRightTrunk)),(340,50),(340,120))
                ##
                # Ankle Plantar Flexion
                elif partSelection == 4:
                    if viewSelection == 1:
                        angleRightAnkle =
pEstimate.toFindAngleThreeCoordinates(frame, 26,28,32, True)
                        absAngleRightAnkle = (round(angleRightAnkle -
90,2))
                        angleDisplay("Ankle Plantar flexion: Right",
absAngleRightAnkle,(30,50),(30,120))
                    else:
                        angleLeftAnkle =
pEstimate.toFindAngleThreeCoordinates(frame, 25,27,31, True)
                        absAngleLeftAnkle = (round(270- angleLeftAnkle,2))
                        angleDisplay("Ankle Plantar flexion: Left",
absAngleLeftAnkle,(30,50),(30,120))

                ##

                # Medial knee positon
                elif partSelection == 5:
                    angleLeftMedialKnee =
pEstimate.findAngleTwoCoordinates(frame, 25,27, True)
                    angleRightMedialKnee =
pEstimate.findAngleTwoCoordinates(frame, 26,28, True)
                    angleDisplay('Medial Knee: Left' ,
abs(round(angleLeftMedialKnee-180)),(30,50),(30,120))
                    angleDisplay('Medial knee: Right' ,
abs(round(angleRightMedialKnee-180)),(340,50),(340,120))
                ##
                # Lateral Trunk Flexion
                elif partSelection == 6:
```

```
                        angleRightLateralTrunk =
pEstimate.findAngleTwoCoordinates(frame, 12, 24, True)
                        absAngleRightLateralTrunk =
abs(round(angleRightLateralTrunk - 170))
                        angleDisplay("Lateral Trunk flexion: Right",
absAngleRightLateralTrunk,(520,50),(520,120))
                        angleLeftLateralTrunk =
pEstimate.findAngleTwoCoordinates(frame, 11, 23, True)
                        absAngleLeftLateralTrunk =
abs(round(angleLeftLateralTrunk - 170))
                        angleDisplay("Lateral Trunk flexion: Left",
absAngleLeftLateralTrunk,(30,50),(30,120))
                ##
                # Stance width
                elif partSelection == 7:
                    footDistance =
pEstimate.findDistanceTwoCoordinates(frame, 31, 32, True)
                    angleDisplay('Foot width' , round(footDistance),
(30,50),(30,120))
                    shoulderWidth =
pEstimate.findDistanceTwoCoordinates(frame, 11, 12, True)
                    angleDisplay('Shoulder width' ,
round(shoulderWidth),(340,50),(340,120))

                    if footDistance > shoulderWidth:
                        cv2.putText(frame, "Foot is wider than shoulder",
(640,50),
                                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 204,
102), 5)
                    elif footDistance == shoulderWidth:
                        cv2.putText(frame, "Foot and shoulder is verticle
to ground", (640,50),
                                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 204,
102), 5)
                    else:
                        cv2.putText(frame, "Foot is narrow than shoulder",
(640,50),
                                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 204,
102), 5)
                ##
                # Foot position
                elif partSelection == 8:
                    # Angle for Left Side Ankle
                    angleLeftAnkle =
pEstimate.findAngleTwoCoordinates(frame, 29,31, True)
                    if round(angleLeftAnkle)-180 > 0:
                        angleDisplay(
                            'Foot position - Left : Internal Rotation',
                            abs(round(angleLeftAnkle)-
180),(30,50),(30,120))
                    elif round(angleLeftAnkle)-180 < 0:
                        angleDisplay(
                            'Foot position - Left : External Rotation',
                            abs(round(angleLeftAnkle)-
180),(30,50),(30,120))
                    else:
                        angleDisplay(
```

```
                            'Foot position - Left : No Rotation',
                            abs(round(angleLeftAnkle)-
180),(30,50),(30,120))
                    # Angle for Right Side Ankle
                    angleRightAnkle = pEstimate.findAngleTwoCoordinates(
                        frame, 30,32, True)
                    if round(angleRightAnkle)-180 > 0:
                        angleDisplay(
                            'Foot position - Right : Internal Rotation',
                            abs(round(angleRightAnkle)-
180),(30,150),(30,220))
                    elif round(angleRightAnkle)-180 < 0:
                        angleDisplay(
                            'Foot position - Right : External Rotation',
                            abs(round(angleRightAnkle)-
180),(30,150),(30,220))
                    else:
                        angleDisplay(
                            'Foot position - Right : No Rotation',
                            0,(30,150),(30,220))
                ##
                # Symmetric foot contact
                elif partSelection == 9:
                    angleSymmetricalAnkels =
pEstimate.findAngleTwoCoordinates(frame, 27, 28, True)
                    angleSymmetricalToes =
pEstimate.findAngleTwoCoordinates(frame, 31, 32, True)
                    angleSymmetricalHeels =
pEstimate.findAngleTwoCoordinates(frame, 29, 30, True)
                    angleDisplay('Ankles Symmatry' ,
abs(round(angleSymmetricalAnkels-90)),(30,50),(30,120))
                    angleDisplay('Toes Symmetry' ,
abs(round(angleSymmetricalToes-90)),(340,50),(340,120))
                    angleDisplay('Heels Symmetry' ,
abs(round(angleSymmetricalHeels-90)),(640,50),(640,120))
                ##
                # Joint displacement
                elif partSelection == 10:
                   # Angle for Left Part of the Body
                   angleLeftTop =
pEstimate.toFindAngleThreeCoordinates(frame, 11, 23, 25, True)
                   # Angle for Right Part of the Body
                   angleRightTop =
pEstimate.toFindAngleThreeCoordinates(frame, 12, 24, 26, True)
                   if maxlt < abs(round(angleLeftTop -180)):
                       maxlt = abs(round(angleLeftTop -180))
                   else:
                       maxlt = maxlt
                   if maxrt < abs(round(angleRightTop - 180)):
                       maxrt = abs(round(angleRightTop - 180))
                   else:
                       maxrt = maxrt
                   angleDisplay('Maximum Trunk Displacement: Left ' ,
maxl,(30,50),(30,120))
                   angleDisplay('Maximum Trunk Displacement: Right' ,
maxr,(640,50),(1140,120))
                   # Angle for Left Side Leg
```

```
                    angleLeftBottom =
pEstimate.toFindAngleThreeCoordinates(frame, 23, 25, 27, True)
                    # Angle for Right Side Leg
                    angleRightBottom =
pEstimate.toFindAngleThreeCoordinates(frame, 24, 26, 28, True)
                    if maxl < abs(round(angleLeftBottom -180)):
                        maxl = abs(round(angleLeftBottom -180))
                    else:
                        maxl = maxl
                    if maxr < abs(round(angleRightBottom - 180)):
                        maxr = abs(round(angleRightBottom - 180))
                    else:
                        maxr = maxr
                    angleDisplay('Maximum Knee Displacement: Left ' ,
maxl,(30,150),(30,220))
                    angleDisplay('Maximum Knee Displacement: Right' ,
maxr,(640,150),(1140,220))
                    #####
                ##
                # Close all the windows
                else:
                    cv2.destroyAllWindows()
                    break

            # Display the result text in each frame
            cv2.imshow("Pose Estimation", frame)
            ##

            # Breaking the loop to close the window by pressing 'q' key
            # Masking the 28 bit of key pressed with the unique code
            if cv2.waitKey(10) & 0xFF == ord('q'):
                cv2.destroyAllWindows()
                break
            ##
            elif cv2.waitKey == ord('p'):
                cv2.waitKey()

        ##
        # Close all the windows
        else:
            cv2.destroyAllWindows()
            break
        ##

# Main function
if __name__ == "__main__":


    # Basic variables used for further estimation
    successLanding = 0
    passCheckLeftBottom = []
    passCheckRightBottom = []
    passCheckLeftTop = []
    passCheckRightTop = []

    finalCheck = ['leftBottom', 'rightBottom', 'leftTop', 'rightTop']
```

```
    ##

    # Creating an object for the postestimation for reuse the logic of
    # angle estimation, to fetch and detect the coordinates of each frame
    pEstimate = pm.poseEstimation()
    ##
    partSelection = int(input(
        "Enter the number for evaluate: 1. Knee Flexion, 2. Hip Flexion, 3.
Trunk Flexion, 4. Ankle Plantar Flexion, 5. Medial knee positon, 6. Lateral
Trunk Flexion, 7. Stance width, 8. Foot position, 9. Symmetric foot
contact, 10. Joint displacement:\n"))

    if partSelection == 4:
        viewSelection = int(input(
            "Enter the number for view for evaluation: 1. Right Side View,
2. Left Side View:\n"))
        if viewSelection == 1:
            # Video as input for the initiating the estimation in Right
side
            video = cv2.VideoCapture('Pose Estimation/016/Dop Jump 1/Drop
Jump Bilateral Markerless 1_Miqus_6_25467.avi')
        else:
            # Video as input for the initiating the estimation in left side
            video = cv2.VideoCapture('Pose Estimation/016/Dop Jump 1/Drop
Jump Bilateral Markerless 1_Miqus_10_25459.avi')
        ##
        # Calling the main function
        main(video, pEstimate, finalCheck, partSelection, viewSelection,
True)
    elif partSelection == 1 or partSelection == 3:

        viewSelection = int(input(
            "Enter the number for view for evaluation: 1. Right Side View,
2. Left Side View:\n"))
        if viewSelection == 1:
            rotation = True
            # Video as input for the initiating the estimation in Right
side
            video = cv2.VideoCapture('Pose Estimation/020/Dop Jump 1/Drop
Jump Bilateral Markerless 1_Miqus_6_25467.avi')
        elif viewSelection == 2:
            rotation = True
            # Video as input for the initiating the estimation in left side
            video = cv2.VideoCapture('Pose Estimation/020/Dop Jump 1/Drop
Jump Bilateral Markerless 1_Miqus_10_25459.avi')

        else:
            rotation = False
            # Video as input for the initiating the estimation
            video = cv2.VideoCapture('Pose Estimation/020/Dop Jump 1/Drop
Jump Bilateral Markerless 1_Miqus_3_25465.avi')
        # Calling the main function
        main(video, pEstimate, finalCheck, partSelection, viewSelection,
rotation)
        ##
    else:
```

```
        # Video as input for the initiating the estimation
        video = cv2.VideoCapture('Pose Estimation/020/Dop Jump 1/Drop Jump
Bilateral Markerless 1_Miqus_3_25465.avi')

        # Calling the main function
        main(video, pEstimate, finalCheck, partSelection)
        ##
##
```