Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Vision Based Jump Rope Counter

Author : Ayush Gupta

Supervisor : DR Kenneth Dawson-Howe

A report submitted to the University of Dublin, in partial fulfilment of the requirements for the degree of Bachelors in Computer Engineering.

Abstract

Vision Based Jump Rope Counter

By

AYUSH GUPTA

Supervisor: DR. KENNETH DAWSON HOWE

SCHOOL OF COMPUTER SCIENCE AND STATISTICS

A jump rope counting system that can take videos of a subject jumping rope as an input to accurately count the number of jumps performed while skipping rope and also counting the different kinds of jumps being performed such as High Knees, Criss Cross and Jumping Jacks is developed. To count the number of jumps accurately, pose of the subject is estimated using Human Pose Estimation to evaluate the movement of the limbs of the subject. The position of the ankle of the subject is analysed throughout the video to detect a valid jump and the angle formed by the wrists, ankle, knee and hip are used to detect the occurrence of the High Knee Jump, Jumping Jacks and Criss Cross Jumps.

A novel rope detection technique is developed in this project which utilizes the detection of contours within a window, tracking the location of the wrists. The rope detector checks for the rope every 90 frames(3 seconds for 30fps video) to continue incrementally counting the number of jumps taking place. In case of presence of no rope, the counter is stopped until a rope is detected again. Since the videos used for development are of myself, Ground Truth had to be generated to accurately evaluate the performance of the system.

Working with non-static background posed a problem during the development of the project but was resolved using appropriate background subtraction techniques. Under ideal conditions the application counts the number of normal jumps with 98.5% accuracy on average. The application detects High knees with 97.48% accuracy on average and Jumping Jacks with a 95.85% accuracy.

Acknowledgement

Firstly, I would like to thank my project supervisor Kenneth Dawson-Howe for his help and guidance throughout the year. His expertise and guidance were extremely helpful in developing this project which I was very passionate about.

I would like to thank my family and friends for the continuous support and encouragement 1000's of miles away from home at all times.

Table of Contents

1 Introduction

This chapter Introduces the project, providing detailed description of the goals of this project. It also discusses the motivation behind this project and the road map for the rest of this project.

1.1 Project Overview

The goal of this project was to create a computer vision application which when given a video of someone jumping rope as an input, processes the video to give the count for the number of jumps performed by the subject. The application is able to detect the following kinds of jumps:

- Normal Jump
- High Knees
- Criss Cross
- Jumping Jack

Additionally, the application was developed to work with backgrounds of static and non-static nature. Static background is a non-moving background in which there is no movement in the background. A non-static background is a background which can have moving elements in the background such as moving tree branches, moving leaves and moving cars.

In the application, the jumps can be performed in any order and the number of jumps will be accurately predicted for the performed activity given that the jumps the subject is fully visible within the frames of the video.

Figure 1-1 Jumping Jack

Figure 1-2: High Knee Jump

Figure 1-3: Normal Jumps

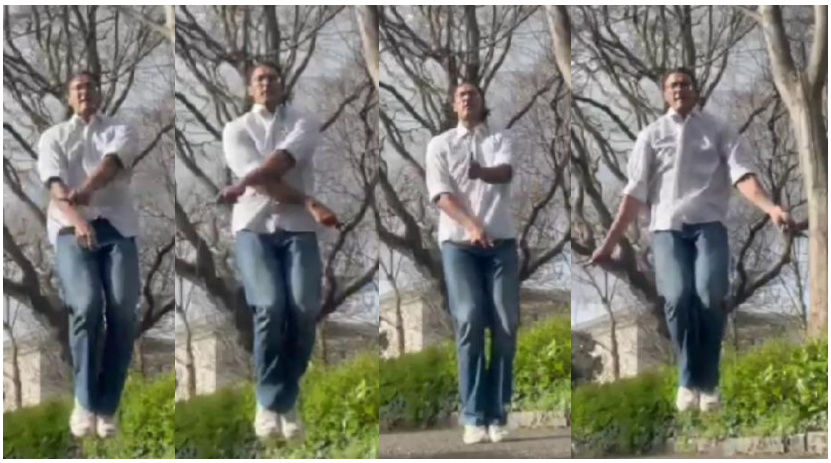

Figure 1-4: Criss Cross

The figures shown above depict the different kinds of jumps which are being detected by the system. The count for each kind of jump increases when the respective jump is observed to have taken place. For this a novel rope detection technique was developed which detects the presence of rope within a window tracking the movement of the wrist as seen in Figure 1-5

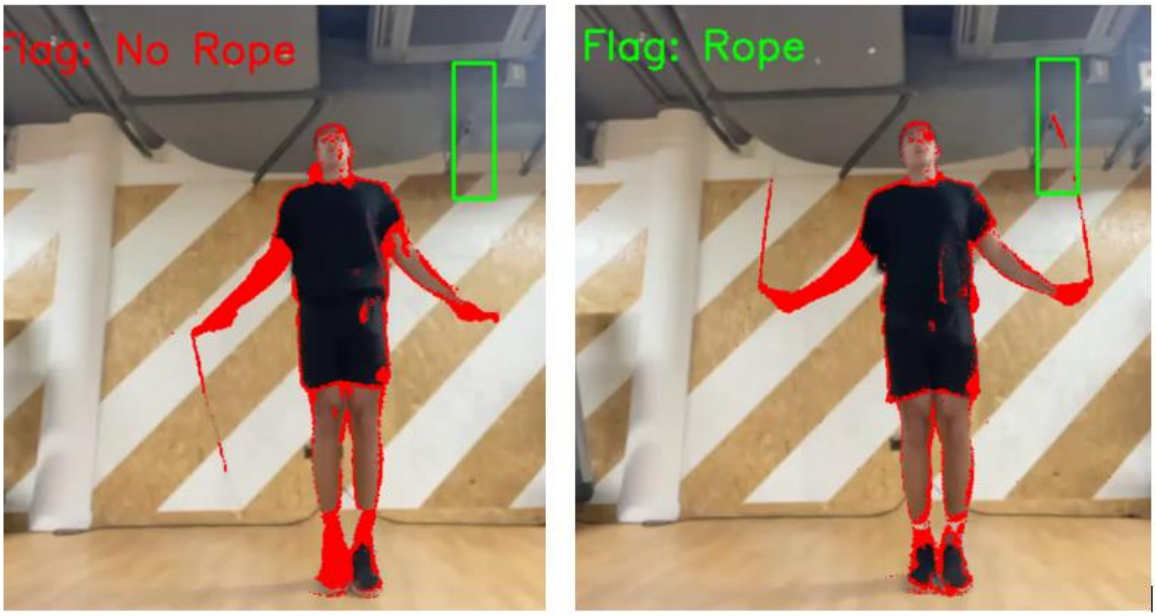

Figure 1-5: Rope Detection

And Finally this is what the final application looks like:

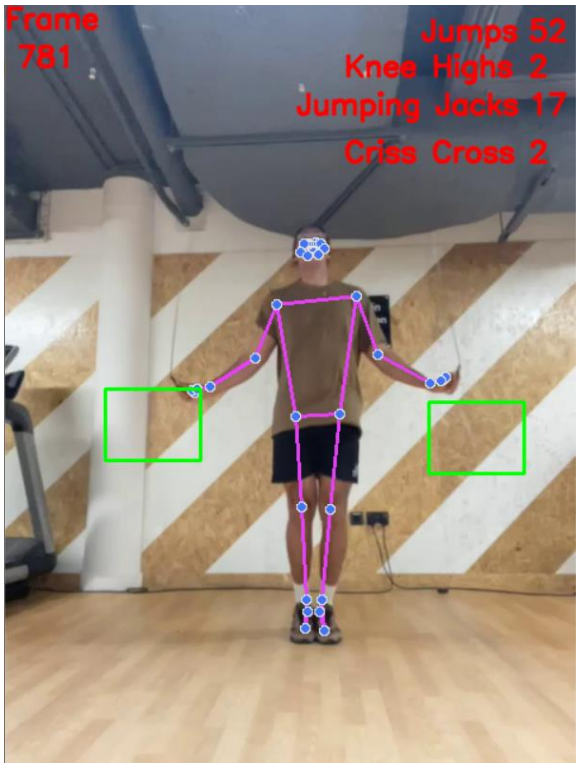

Figure 1-6: Final Application

1.2 Motivation

Jumping rope is a popular exercise that is not only fun but also offers numerous health benefits. It is a great way to improve cardiovascular health, coordination, and endurance. [1] However, counting the number of jumps accurately can be a challenge, especially when individuals are performing different types of jumps, such as High Knees or Criss Cross Jumps. Inaccurate counting can hinder progress and make it difficult to set and achieve fitness goals. Moreover, manually counting the number of jumps can be tedious, inaccurate and take a person's mind off the exercise, which can discourage individuals from engaging in this form of exercise regularly.

Therefore, the motivation behind developing a project on jump rope counting using computer vision is to address these challenges by providing an automated and efficient solution that accurately counts the number of jumps while also identifying and recording different types of jumps. Computer vision technology enables the system to recognize and differentiate between various jump rope movements, including different types of jumps and their combinations. This capability makes it easier for individuals to track their progress accurately, measure their performance, and set achievable goals.

Overall, the motivation behind this project is to make jumping rope a more accessible and enjoyable form of exercise while also demonstrating the potential of technology in improving health and wellness.

1.3 Defining a Jump

Before moving forward it is necessary to define a valid jump. A valid jump is defined to have taken place when the rope goes underneath the foot of a jumper and comes back over again. A person who jumps rope can jump different heights based on the kind of jumps they're performing and have different positions of hands and legs when compared to other people. The video which would be analysed has to be a front view of the person using the rope i.e. the person jumping the rope should face the camera while recording themselves.

1.4 Road Map

In this section I would provide a brief overview of the content discussed in the subsequent chapters of this thesis.

Chapter 2 discusses the solutions for the problem which are present currently, description about the academic literature, techniques which have been used during the implementation of this application such as person detection, jump detection and rope detection.

Chapter 3 will discuss in detail the method used for detecting jumps in a video sequence which will include detecting presence of a skipper in each frame, identification of a jump taking place and updating the counter accordingly.

Chapter 4 will discuss in detail the methodology behind detection of rope in the application in a video sequence. It makes use of contours and background models to achieve the final result.

Chapter 5 will discuss the classification techniques used for detecting different kinds of jumps such as jumping jack, Criss Cross Jump and High Knee jumps.

Chapter 6 Evaluates the techniques developed and discusses the problems encountered.

Chapter 7 will conclude the report with a review of the system and discuss the current limitations of the system. Future work is also discussed.

2 Background

The previous chapter introduced the project and discussed the motivation behind the project. This chapter discusses the necessary background information which acquaints the reader with currently available technology, literature on the subject and techniques that are used in the project

2.1 Present solutions

At the writing of this thesis there are 2 different kinds techniques with which are used count the number of jumps performed by a subject. One of these techniques makes use of sensors such as gyroscopes and accelerometers present in handheld devices and in special smart rope as seen in [2]. The other solution is a vision based approach which is relatively new and make use of OpenPose which was open sourced in 2018.

2.1.1 Applications which make use of sensors

Since jumping rope is such a popular way of exercising, it should be no surprise that their exist jump ropes which have inbuilt jump counters in them. They make use of an IMU [3],which is an electronic device that measures and reports specific force, angular rate using a combination of accelerometers and gyroscopes. The IMU is placed inside the handles of the rope and measures the rotation of the handles as the user jumps m allowing the device to track the number of jumps, duration of the workout.

A microcontroller is also present inside the handles of the rope which processes the gyroscopic data received from the IMU. The microcontroller has logic programmed inside it which analyses the data to accurately count the number of jumps, the duration of the workout and the number of calories burned, based on the weight and height details provided by the user [2].

The skipping rope with inbuilt counters is a device which has been available in markets since early 2000s and hence been a recent development. It was brought to the market because of the advancement in portable technology and reduced size of gyroscopes and accelerometers [2] . With inbuilt display, the users get a live feedback in terms of the number of jumps they're doing, allowing user to track their progress and set fitness goals. They can be used in variety of settings, including indoors and outdoors allowing them to be versatile since they can be used both outdoors and indoors, by people of varying skill levels.

Jump detection using mobile devices, such as iPhones, utilizes various sensors, including the accelerometer and gyroscope integrated into the system, to measure the device's orientation as seen in the step counting technology using gyroscopes [4] . The incorporation of the gyroscope and accelerometer enables the detection of numerous phenomena, including orientation, shake detection, pedometer functionality, tilt detection, rotation, and stabilization (utilized for image and

video stabilization) [5].By harnessing the capabilities of these pre-existing sensors, mobile phones can accurately count jumps by collecting and analysing sensor data in three-dimensional space. The accelerometer gathers information pertaining to linear acceleration along the x, y, and z axes, while the gyroscope measures angular velocity along these same axes. Upon acquisition of this data, a jump detection algorithm is employed wherein the value of the vector product of the acceleration detected by the accelerometer is calculated. A jump is registered when this value surpasses a pre-determined threshold. Upon successful detection of a valid jump via the sensors, the count is incremented and recorded within the final application

2.1.2 Vision based solutions

At the time of writing this report there are 2 applications [6] [7] which make use of computer vision to count the number of jumps being performed while jumping rope. The two projects found on GitHub are briefed in the following passage.

2.1.2.1 Cleaner Pose Estimation using Open Pose:

The author makes use of python, OpenCV and OpenPose's Pose estimation model [8]. OpenPose's pose estimation model is used to extract body keypoints, a filter is applied to clean the noise, the data is used to detect the number of jumps. [6]

The above implementation makes use of Deep neural network libraries in OpenCV and does not go over the logic which has been applied in the final implementation for detecting the number of jumps. Since the application was using previous version of OpenPose it relied heavily on the performance of the GPU in the system, making it less accessible for mobile users in its current state.

2.1.2.2 Jump Rope Counter

GitHub user Danimaaz made a jump rope counter using deep learning [7], more specifically convolution neural networks. Their implementation involved collecting videos of themselves jumping, labelling them in up down and landing positions, training a neural network on it then. Using a neural network for hyper specific tasks like this poses the issue of overtraining which is observed in their implementation. Movements which are not jump related are being also picked up as jump but the implementation is definitely insightful and gives us a good starting point to base our work off of.

2.2 Present Literature

This section discusses the available literature which most closely resembles the application developed. At present there is very limited literature which counts the number of jumps being detected using computer vision. Several papers and articles have been published which formulate techniques that could be used to count the number of jumps performed while jumping rope using

sensors such as an IMU [3] or accelerometers and gyroscopes. A few of these papers are briefed below:

2.2.1 Paper 1

"Accelerometer-Based Automated Counting of Ten Exercises without Exercise-Specific Training or Tuning" by Samuel Zelman et al [9]

This study aimed to create an automatic repetition counting system to measure repeating movements during physical therapy [9].The researchers used accelerometers attached to smartphones to track repetitive motions during different exercises. The study found the periodic motions to be tracked very well and faced difficulty when it encountered irregular movements. Threshold crossing with low pass filtering was used along with fast Fourier transform [10] in this study but the former gave the better results.

2.2.2 Paper 2

"Validation of an Inertial Measurement unit for the measurement of jump count and height" by Kerry MacDonald et al[4].

This study aimed to validate the use of an Inertial Measurement Unit(IMU) the collection of total jump count and assess the validity of an IMU for the measurement of jump height as well. It was concluded that the number of jumps detected by the IMU device were 5% more than the number of jumps recorded by manual inspection. This study concludes that IMU is a valid measuring tool for jump counting [4].

2.2.3 Paper 3

"Determining jumping performance from a single body-worn accelerometer using machine learning" by Mark white et al [11]

This study looks at the jump performance for countermovement jump which is used to monitor an athlete's training. This study is using a machine learning model based on characteristic features extracted from a single body-worn accelerometer. This study finds an alternative to the use of force plate which is standard for performance testing, since it cannot be used by athletes on the field when they are performing [11].

2.2.4 Paper 4

"Video-Based Rope Skipping Repetition Counting with ResNet Model)" by Xinxin Li, Jiawen Wang [12]

A model for counting the number of repetitions was proposed using the ResNet Model [13] and a counting algorithm. The study classifies the motion of the subject as upward and downwards to detect the occurrence of a jump. Similar technique was observed to be used in 2.1.2.2. Detects the number of jumps very accurately but does detect different kinds of jumps.

2.3 Theory of Techniques used.

This section discusses the theory behind the various computer vision techniques that were used for the implementation of this project.

2.3.1 Gaussian Background model

The MOG (Mixture of Gaussians) background model is a widely used method for detecting moving objects in computer vision applications. It is a statistical based on modelling the background image as a mixture of Gaussian distributions and detecting any pixels that deviate significantly from this distribution as foreground pixels [14]. It does so estimating the probability distribution of each pixel in a video sequence over time.

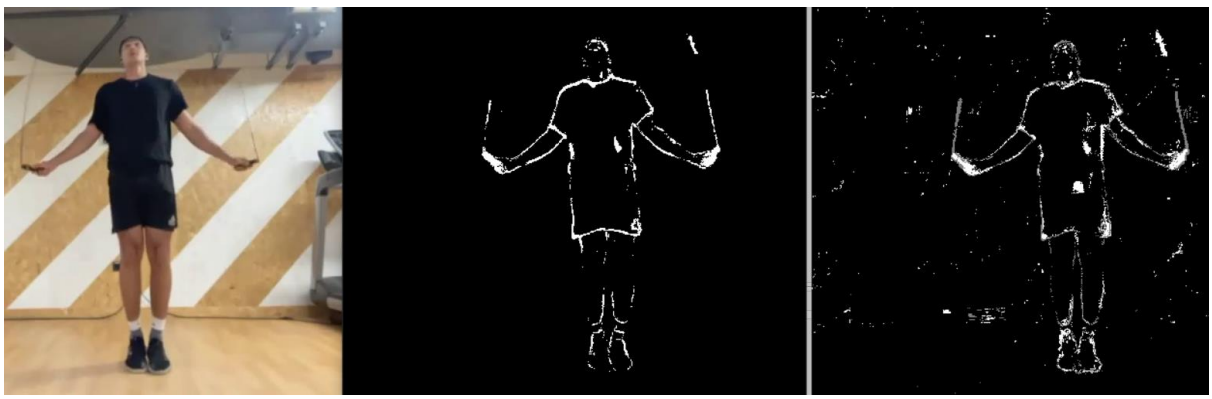

Figure 2-1: From Left to Right: (a)Normal Jump (b) Subtraction Model used in the system(b) Subtraction Model with additional parameters

Based on the K-Nearest Neighbours(KNN) algorithm, the cv2.BackgroundSubtractorKNN() uses the algorithm to model the background and detect moving objects in a video sequence. The algorithm works by storing a set of historical pixel value for each pixel location in the video sequence. Each new pixel value is then compared to the historical set of values for that location. If the pixel value deviates significantly from the historical set, it is considered a foreground pixel.

From the images in the figures given below it is observed how the background model behaves for different parameters.

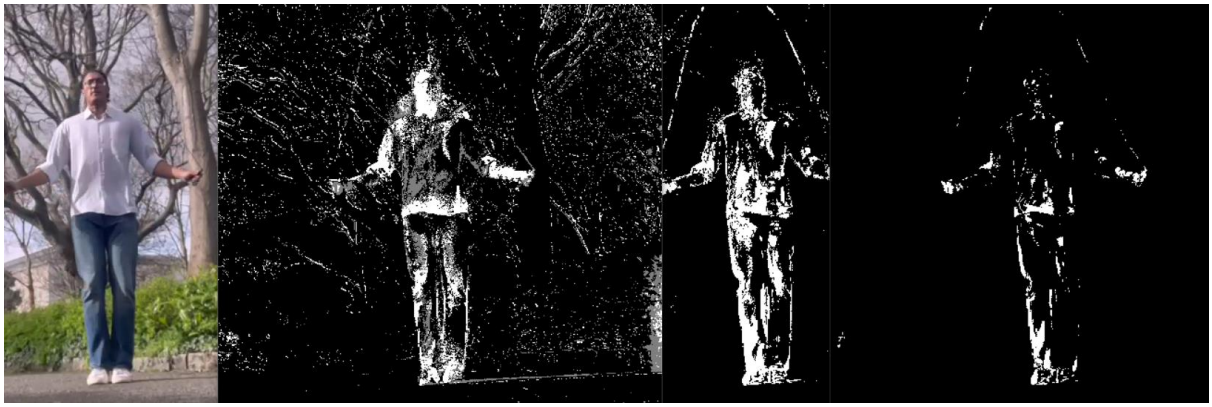

Figure 2-2 From Left to Right: (a)Normal Jump (b) Subtraction Model with no additional parameters (c) Subtraction Model with additional parameters (d) Subtraction Model used in application

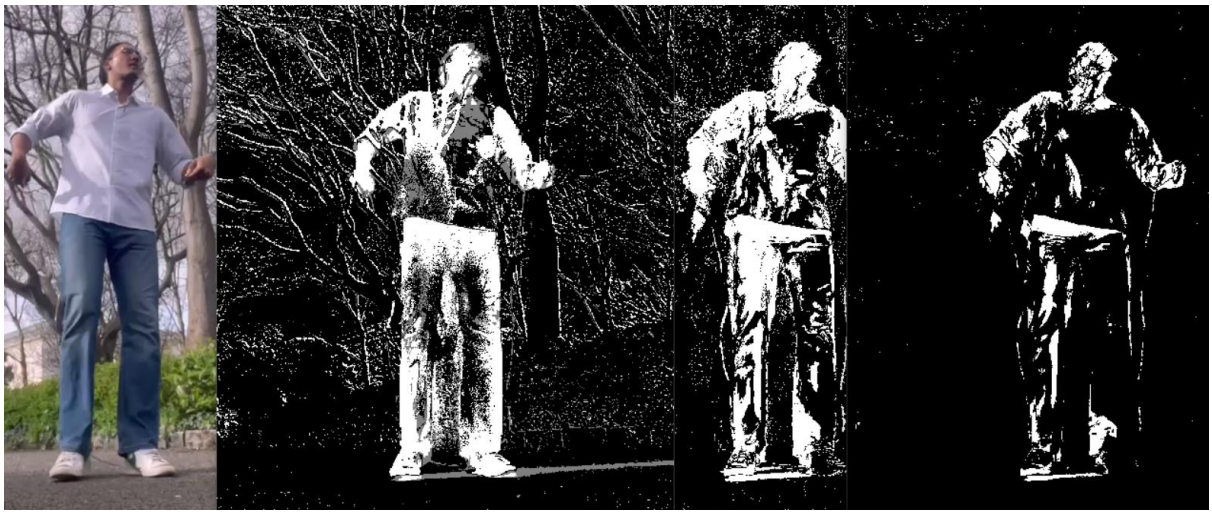

Figure 2-3 From Left to Right: (a)Normal Jump (b) Subtraction Model with no additional parameters (c) Subtraction Model with additional parameters (d) Subtraction Model used in application

Figure 2-4: From Left to Right: (a)Normal Jump (b) Subtraction Model with no additional parameters (c) Subtraction Model with additional parameters (d) Subtraction Model used in application

```
knn1fgbg = cv2.createBackgroundSubtractorKNN()
knnfgbg = cv2.createBackgroundSubtractorKNN(detectShadows=False,history=5)
knn2fgbg = cv2.createBackgroundSubtractorKNN(detectShadows=False,dist2Threshold = 2000,history=5)
```

Figure 2-5: Code for the Different parameters

The parameter detctectShadows detects the shadows in the background image received and has been set to False as the application doesn't want to deal with shadows.

Knn algorithm in background subtraction is more robust towards gradual illumination changes in the scene and is able to handle shadows. These features make KNN a very viable technique for the implementation, as it wishes to combat the changing illumination and shadows cast by surroundings or shadows affecting the processing [15]

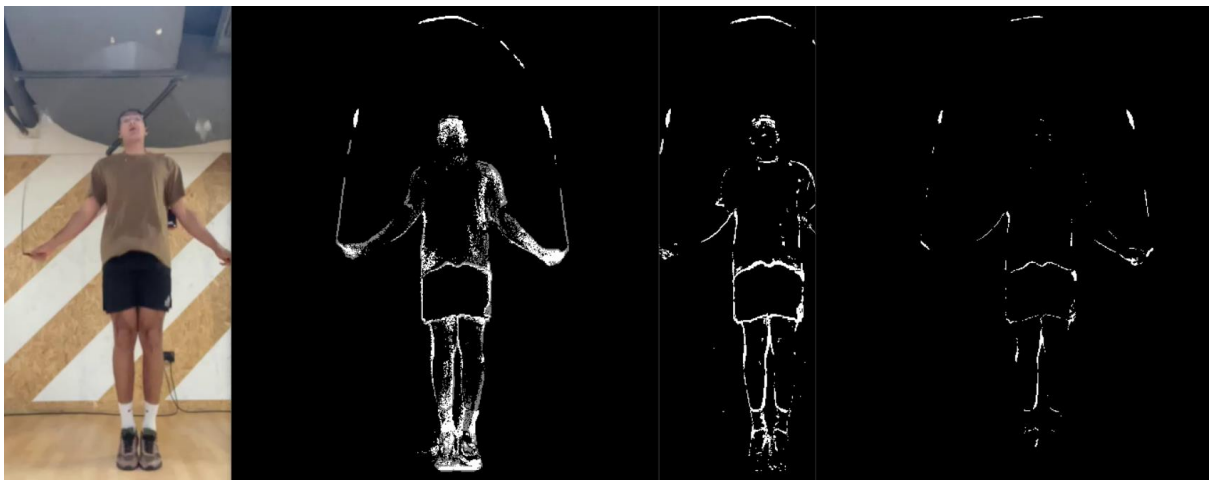

Figure 2-6: From Left to Right: (a)Normal Jump (b) Subtraction Model with no additional parameters (c) Subtraction Model with additional parameters (d) Subtraction Model used in application

Hence KNN based Background subtraction is an effective implementation of the K nearest Neighbours technique, handling illumination changes and shadows effectively. It should be noted that the choice of appropriate parameters depends on the complexity of the scene and the requirements of the application. Keeping this in mind, through trial and error of using different parameters, the perfect parameters within the scope of this project were found.

2.3.2 Grayscale Images and Binary Thresholding

Grayscale images are essential to video and image processing due to their simplicity and computational efficiency. Grayscale images only contain shades of grey, ranging from block to white, keeping no colour information. RGB images on the other hand stores 3 channels of colours(red, green and blue) which combine to form the final colour of the pixel.

Figure 2-7 From Left to Right: (a) RGB image (b) Grayscale Image

Grayscales images are primarily used due to their computational efficiency, allowing simplification of image processing tasks. They also effectively capture the contrast and edge information in an image and hence is a necessary and important step in the pipeline for edge detection.

RGB images are converted to grayscale images without significant loss of information, which makes their usage very viable. The conversion process typically involves taking a weighted average of the red, green, and blue colour channels to produce a single grayscale value for each pixel. Different weightings can be used to produce different grayscale representations of the same image, which can have a significant impact on subsequent image processing steps [16].

Figure 2-8 From Left to Right: (a) RGB Frame (b) Grayscale Frame (c) Binary Image frame

In the above figure Grayscale images are binarized using a thresholding technique, where pixels above a certain thresholds are set to white and below are set to black, helping in segmenting the image into foreground(1,white) and background (0,black) based on the threshold set. This

project makes use of Binary Thresholding which has a fixed value. On Trial and error the right binary threshold for the image is 127.

Figure 2-9From Left to Right: (a) grayscale image (b)Adaptive Thresholded Binary Image (c) Binary Thresholded Binary Image

Figure 2-9 Shows the performance of the different binary thresholding techniques on grayscale image.

Grayscale images are hence used extensively in image processing due to the improved efficiency without losing much information and as gateway to binary thresholding. Binary thresholding is hence used to separate the foreground from the background of an image into two and also find edges. Although choosing the right threshold can be tricky, advanced techniques such as Adaptive Thresholding and Otsu Thresholding [17] can be used to increase the desired accuracy.

2.3.3 Contours

Contours are sets of curves that represent the boundaries of an object in an image. Contour detection and it's analysis plays a key role in this application in form of shape detection. Contours are represented as a sequence of points that lie on the object's boundary, connecting the points to form a curve that approximates the object's shape [18].

Figure 2-10: (a) RGB Image (b) Contour drawn and detected in red

Image segmentation and object detection are important applications of contours, as they are used to define the boundaries of the segmented regions and separate them from the rest of the image [19]. In this project, contours are detected in the image frame after using a background subtraction model to finally see the moving pixels. The contours formed by those pixels isolates the pixels which isolate the rope and allows its detection.

Figure 2-10 shows the contours detected by the application after thresholding and applying background subtraction model. The contours are drawn and in their locations on the original frame. The contours are detected using the background subtraction model image received from the previous chapter[2.3.1]

It should be noted that the use of contours makes the application less robust due to the presence of noise and variations in lightning and background and hence pre-processing techniques such as thresholding are used to pre-process the image and extract reliable contours.

2.3.4 Arctan For Calculating angles

To caluclate the various angles that were anlyzed in the application, revrse tangent, also known as arctangent, a mathematical function which calculates the angle between the positive x-axis and a given point on the plane was used. This function is the inverse of the tangent function and is denoted as

$$\tan x \ \ or \ \ \arctan x$$

Reverse tangent function is used in various applications such as robotics, computer graphics and signal processing to name a few. This function is used to calculate the angle between twio vectors, determining the orientation of a 3D object.

Here, this technique is used to determine the angle between various detected keypoints that are received from the human pose estimation model. The function takes in the first point, the middle point and the final point for which the angle is to be detected. Angle is measured between the lines connecting the points and is then finally returned in degrees. The code is provided below to check the angle

```
def calculate_angle(a, b, c):
    # Convert input points to numpy arrays
    a = np.array(a)  # First point
    b = np.array(b)  # Middle point
    c = np.array(c)  # Last point

    # Calculate the angle in radians between the lines connecting b to a and b to c
    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] - b[0])

    # Convert angle from radians to degrees
    angle = np.abs(radians * 180.0 / np.pi)

    # Make sure angle is not greater than 180 degrees
    if angle > 180.0:
        angle = 360 - angle

    return angle
```

Figure 2-11: Code snippet depicting the angle calculation method used

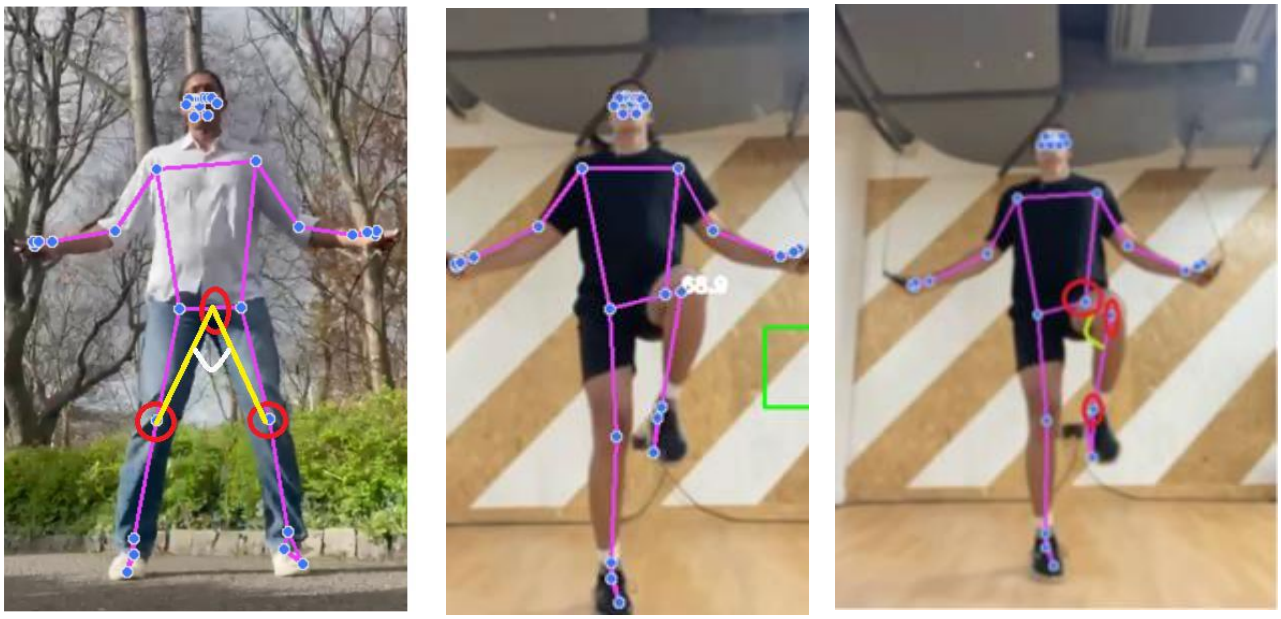

Figure 2-12 From Left to Right: (a) Angle for Jumping Jack, (b) Angle detected for High Knee, (c) Angle for High Knee

3 Jump Detection

In this section, the design and implementation of methodology for detecting jumps while skipping rope in the application is discussed. The process involves several steps, including detecting the presence of the subject (the skipper), applying human pose estimation to track the movement of keypoints throughout the video, and finally, discussing the logic used to determine if a jump has occurred.

3.1 Detecting Human (Human Pose estimation)

To detect a person in the scene, Human Pose Estimation using the Mediapipe library from TensorFlow Google [8] provides a robust and efficient approach for detecting and tracking human subjects in videos. The Mediapipe library utilizes a deep neural network architecture that is trained on a large labelled dataset of human poses [20], allowing for accurate localization of body parts and estimation of joint position and orientations. The detected human joint positions were used as key points for the analysis of the movement of different body parts.

The estimated joint positions are used for pose estimation. The model calculates the 3D or 2D pose of the human subject by estimating the joint positions and orientations relative to a global or local coordinate system. The pose estimation step involves combining the detected body parts and joint locations to reconstruct the complete human pose, including the positions and orientations of all the body joints. Figure 3-1

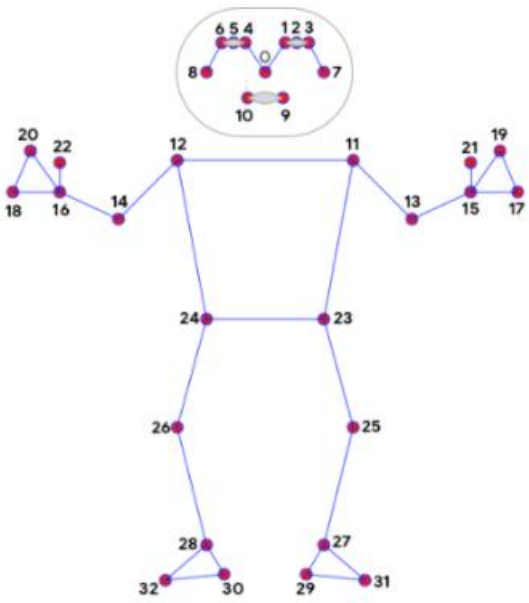

Figure 3-1: Human Pose Estimation Reference points

In the context of detecting when a human is jumping, the Human Pose Estimation technique was utilized to detect and track the joint positions and orientation of human body during a jumping action. The key points corresponding to the head, neck, shoulders, wrists, hips, knees and ankles were detected, and the joint locations were estimated with high accuracy. The estimated joint positions were then used to infer the jumping action of the human subject in the video frames. The jumping action is detected based on the change in joint positions and their orientation during the jumping action. The joint positions and orientations from multiple consecutive frames were analysed to capture the temporal dynamics of the jumping action. This temporal analysis helped to further improve the accuracy and robustness if the jumping action detection as it considered the changes in joint position and orientations over time. [21]

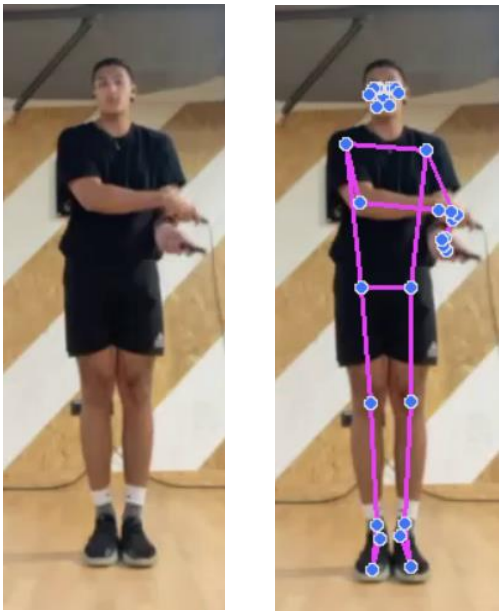

Figure 3-2 (a) Normal Video Frame (b) Pose estimation frame

3.2 Key-points Analysis

Once the human pose estimation is applied and the presence of subject is detected, movement of the keypoints are tracked using HPE throughout the motion of the subject in the video. This allows closer observation of the behaviour of each keypoint and help identify the keypoints tracking which would lead to the best results.

3.2.1 Comparing Keypoints

This section compares the behaviour of various keypoints throughout the video in order to find a consistent metric to correctly count the number of jumps taking place. This is done by tracking the x and y coordinate corresponding to the various keypoints, storing these values and observing their behaviour through the frames using graphs. In the following curves, the motion of a subject performing normal jumps is considered. The subject performs 27 jumps during the duration of this video which is being analysed.

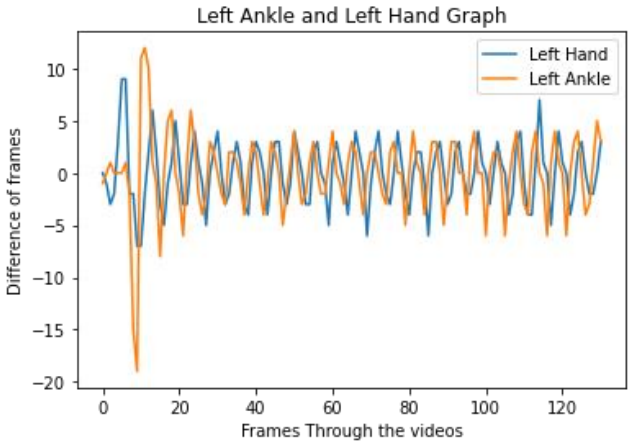

Figure 3-3: Movement Graph of Left Ankle And Left Hand (Y axis)

In the figure 3-3, a comparison between the behaviour observed by the left hand keypoint and left ankle keypoint is made. It's a difference of frames curve in which the relative movement of the keypoints is observed and is not influenced by their position in the video frame but rather their movement. From this curve we can conclude that the keypoint tracking the y coordinate corresponding to the keypoints of the left hand and the left ankle behave similarly.

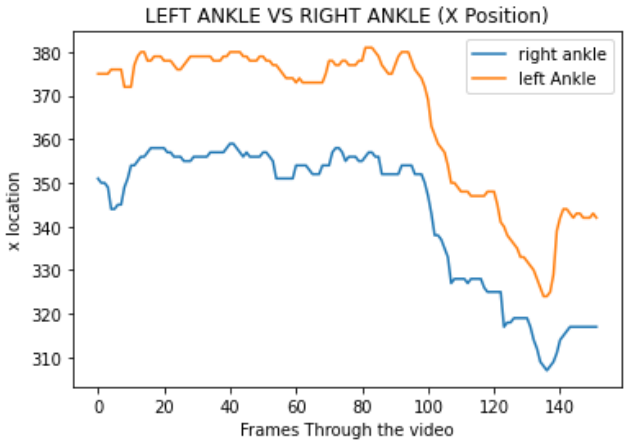

Figure 3-4: X Coordinate Comparison of Left and Right Ankle

The above Figure 3-4 represents the behaviour of keypoint corresponding to the x axis representing the left ankle and the right ankle keypoints throughput the video. It is observed that left ankle is always greater in magnitude than right ankle and that is due to the placement of the origin in the image frame as shown in Figure 3-5

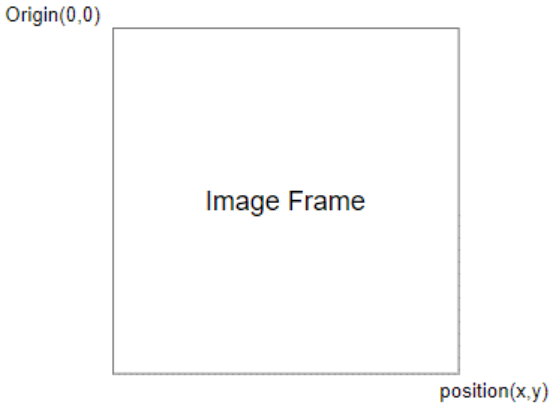

Figure 3-5: Origin in the image

Hence the x location of a keypoint which is located on the left side will always be greater than its corresponding right point unless they are crossed over.

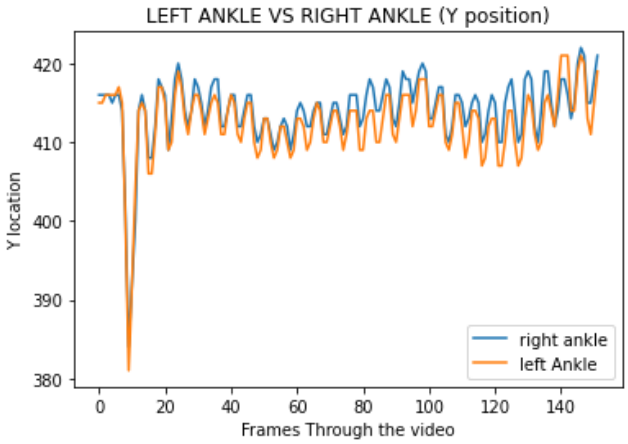

Figure 3-6: Y coordinate comparison between left and right ankle

The Figure 3-6 shown above depicts the relationship present between the Y coordinate corresponding to the Left Ankle and the Right Ankle. Both the curves appear to almost identical in nature when tracking their y coordinates which is a contrast from Figure 3-4 seen before comparing the left and right ankle's x coordinate.

Similar conclusions can be drawn when looking at the hip and the knee as keypoints in terms of behaviour of the x and y coordinate corresponding to the keypoint.

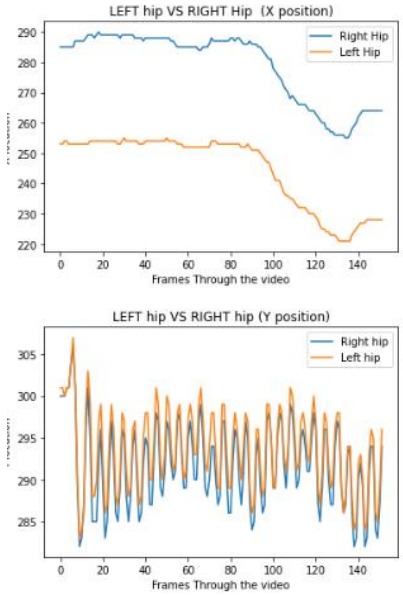

Figure 3-7: X and Y coordinate comparison for Left and Right Hip

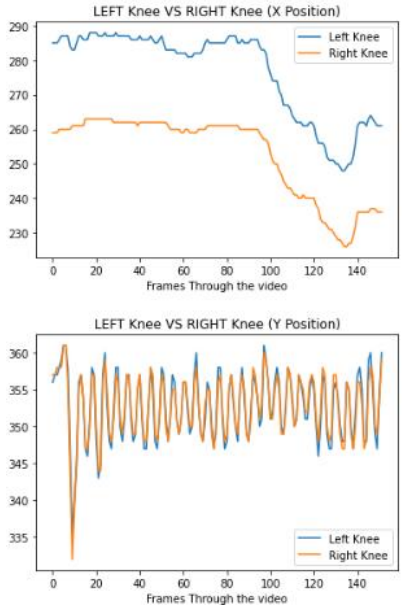

Figure 3-8: X and Y coordinate comparison for the Left and Right Knee

3.2.2 Keypoints Inference

Upon applying human pose estimation, Detection of local keypoint was observed, as shown in the Figure 3-1. These keypoints were tracked throughout the live video as the subject moved. By analysing these points individually, certain inferences are drawn that help identify keypoint tracking which provides the most accurate count of jumps.

The behaviour of the keypoints in the form of graphs tracking their movement was studied and attempted to be interpreted. It is natural to observe the ankles at the bottom of the feet if one wishes to see if a person is jumping. Same logic is applied here when looking at a keypoint to observe, hence ankles are observed. The analysis of the motion of the keypoints for the following keypoints is observed:

- Ankle
- Knees
- Hip
- Wrist

Upon analysing the graphs corresponding to the ankles, a clear pattern is observed. The ankles followed a sinusoidal path on the curve with respect to time and position. The curves exhibited changes in amplitude throughout its motion, which are attributed to the height of the jumps performed by the subject. Despite these changes in amplitude, the sinusoidal nature of the curve was maintained and local minimum points were observed. This trend was also evident when tracking other aforementioned keypoints.

The y-position curves for both ankles were examined and it was found that they followed similar paths, which allowed the researchers to draw the same conclusions. It was noted that the amplitude of the curves could be altered by performing different types of jumps, but the sinusoidal nature of the curve would remain present during jumping actions. This characteristic of curves in the y-coordinate space was consistent for all of the keypoints that were analysed. As all of the curves exhibited similar behaviour, the number of jumps detected using each keypoint was compared and their performance was evaluated. In contrast, when examining the x-positions of the ankles and the above mentioned keypoints, no conclusion could be draw. A difference in amplitude was observed between both curves, but they both followed similar movements. The x-coordinate of the subject's ankle represented their lateral position within the frame and was useful for tracking their movement throughout the video.

3.3 Technique implemented

As previously mentioned, when observing the number of jumps performed by a subject, we focus on the movement of their ankles. Individuals observe the upward and downward motion of the ankles and subconsciously count the number of jumps. Similarly, in this application the number of jumps performed by the subject are counted by observing the number of times their feet touch the ground.

This is achieved by counting the number of minima observed in the y-coordinate curve corresponding to the ankle, with each minimum point representing the location of the ankle keypoint at its lowest point and one period depicting a jump. Given how coordinates are detected within the frame, we can identify the lowest points on the curve during each period as being when the ankle is in contact with the ground, i.e., its lowest position and also its local minima on the graph representing the motion . Therefore, an algorithm is implemented that increases the jump count whenever a local minima was encountered.

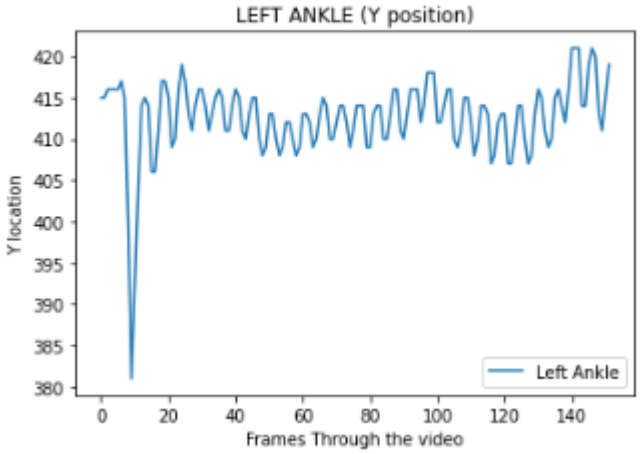

Figure 3-9: Behaviour of y coordinate of the left ankle through a video

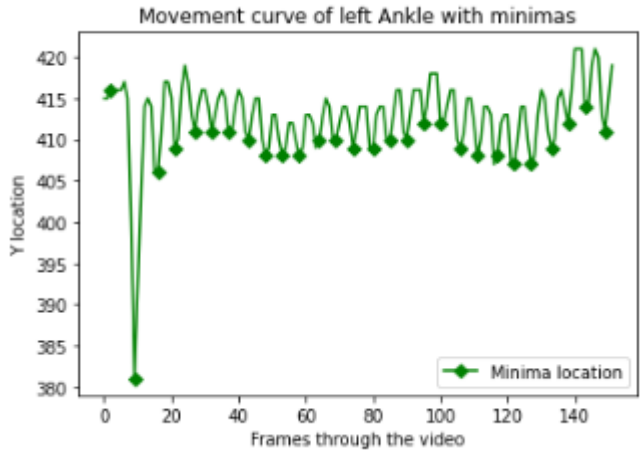

Figure 3-10: Labelled minimas detected in the Y coordinate movement of left Ankle

From the Figure 3-10 shown above it can be seen that the frames which are being identified as minimas are marked on the curve defining the motion of the ankle. Marking the curve to see where the minima is located can help us identify that there is a misclassification of the first jump between frame in 0 and 10. The number of jumps being detected by using the ankle key point is 28. Similar performance is observed in identifying the number of jumps using different keypoints as seen in the figures depicted below.

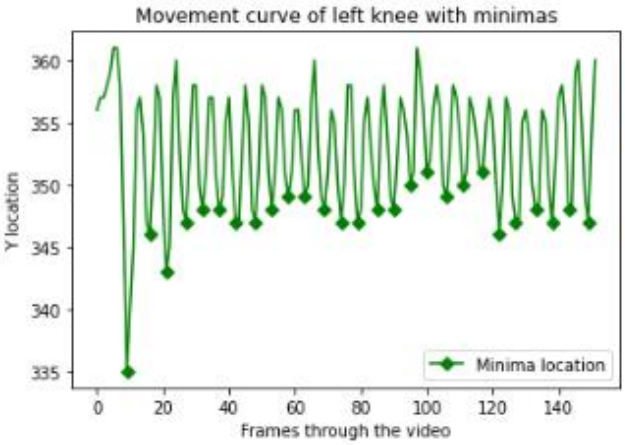

Figure 3-12: Labelled minimas detected in the Y coordinate movement of Left Knee

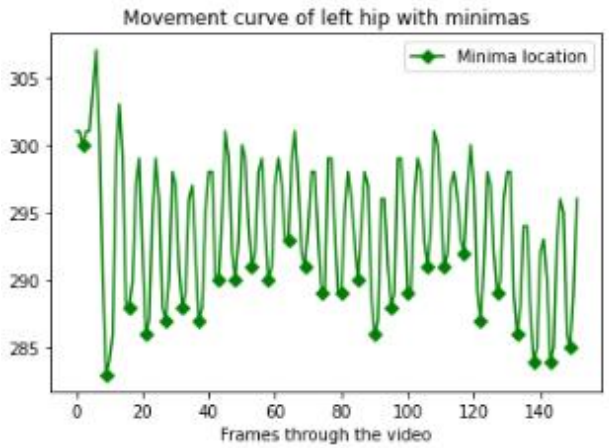

Figure 3-11: Labelled minimas detected in the Y coordinate movement of Left Hip

Hence, it can be concluded that to detect the occurrence of a jump, the technique discussed above can be used effectively to count the number of jumps. Under ideal conditions, tracking of the ankle, hip and knee keypoints yield very similar results. As the keypoints tracking of these 3 points result in similar results, the left ankle keypoint is used in the final application to detect the number of jumps being detected.

This concludes the jump detection chapter, in the next chapter technique behind the detection of rope is discussed in detail.

4 Rope Detection

The preceding section covered the detection of jumps in the application, including the underlying technology, counting the number of jumps, and discussing the process. This section will focus on the methodology used to detect the presence of a rope and how rope detection was utilized. A flow diagram is provided below to illustrate the logic behind the methodology to be discussed subsequently.

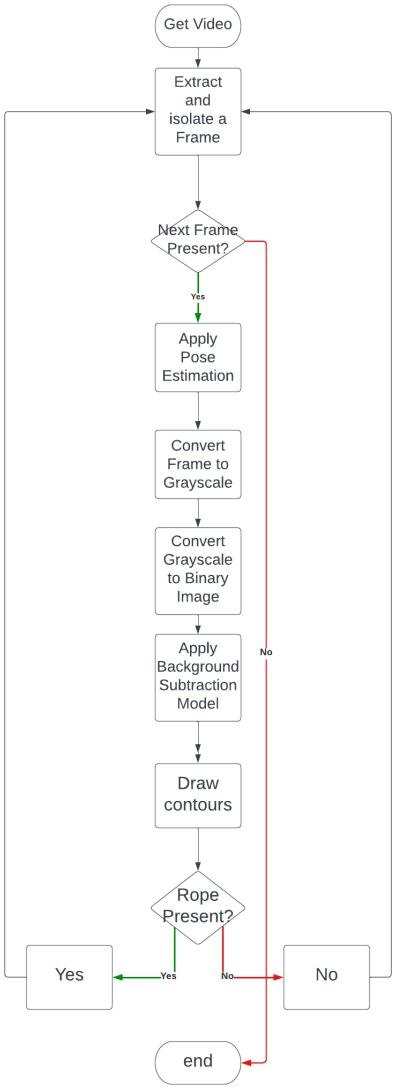

Figure 4-1: Rope Detection logic Flow Diagram

All the points mentioned in the above flow diagram (**Error! Reference source not found.**) can be found in the subsequent pages of the thesis, under Rope detection.

4.1 Need for detecting rope

First and foremost, it is imperative to understand the rationale behind detecting the rope in this application. Given that the application in question is designed to count jumps using a jump rope, accurately detecting the presence of a rope in motion is crucial for the proper function of the application. In order for a jump (Figure 4-2 : Frame) to be validly detected, it is necessary to ensure that the detection occurs when the rope passes beneath the feet of the individual jumping. Utilizing background subtraction models presents challenges in accurately locating the rope and determining whether its passing beneath the feet of the jumper, due to the rapid movement of the rope and variations in its surrounding in different types of video backgrounds, namely static and dynamic backgrounds. Consequently, there is a risk that the rope may not be detected with a high degree of accuracy.

Figure 4-2 : Frame

To ensure accurate detection of a rope in the frames used to validate a jump, a specific technique has been employed. It is important to establish the parameters of the algorithm, specifically whether the rope is being tracked in every frame and throughout its entire motion, or if the focus is on the rope passing through a designated window. In this application, the objective is to track the rope as it passes through a specific portion of the video, as determined by the location of the wrist in the hand.

In the subsequent sections, processing techniques employed to detect the presence of a rope in a user's hand are discussed in detail. We will elucidate the functionality of the application in detecting a rope when one is present and demonstrate its behaviour in the absence of a rope, illustrating how the application responds under such circumstances.

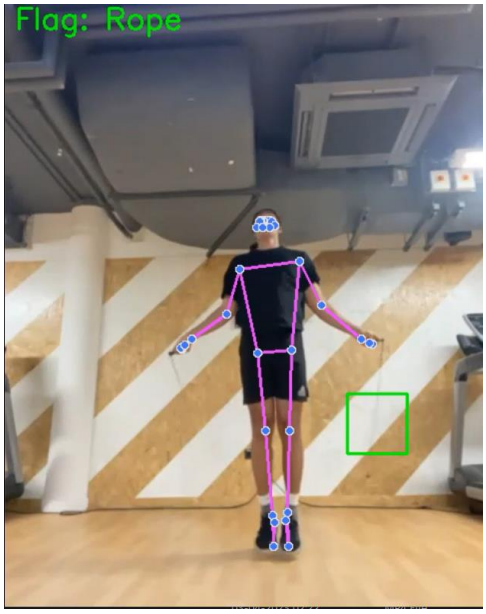

Figure 4-3: Working Rope Detection

4.2 Image conversions and background models(Frame pre-processing)

In order to accurately detect the presence of a rope in a video of a subject jumping rope, its necessary to employ a series of sophisticated processing techniques. To begin the analysis, a frame from a live video is extracted in which a subject is actively engaged in jumping rope, with the rope in hand and their body suspended in mid-air during the activity. This frame serves as a representative example for the analysis and provides a basis for further processing. (Figure 4-2 : Frame)

Once the frame in question is obtained, Google's advanced Media Pipe based pose estimation technique is applied to the image. This tool enables us to obtain a series of key points that can be used to track the movement of various body parts throughout the video and the motion of jumping rope (Figure 4-4: Subject with Human Pose Estimation) . For further information on the capabilities and functionality of Media Pipe, kindly refer to the previous section where it was discussed in more detail [3.1]

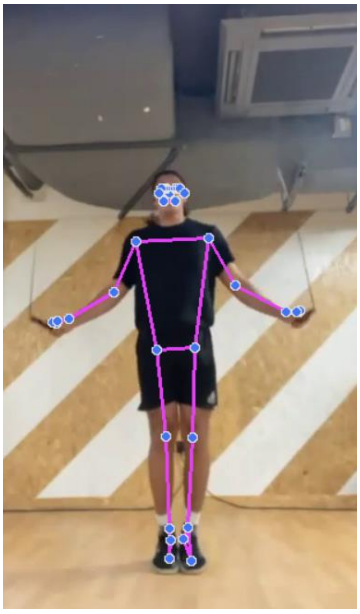

Figure 4-4: Subject with Human Pose Estimation

On applying Media Pipe's human pose estimation technique to the frame (Figure 4-4: Subject with Human Pose Estimation), location of the position of the wrists can be measured accurately. These wrist locations are then used to define a rectangular window within which presence of a rope is detected. Given that the location of wrists changes over time as the individual jumps and decides to move, it is essential to update the position of the checking box accordingly.

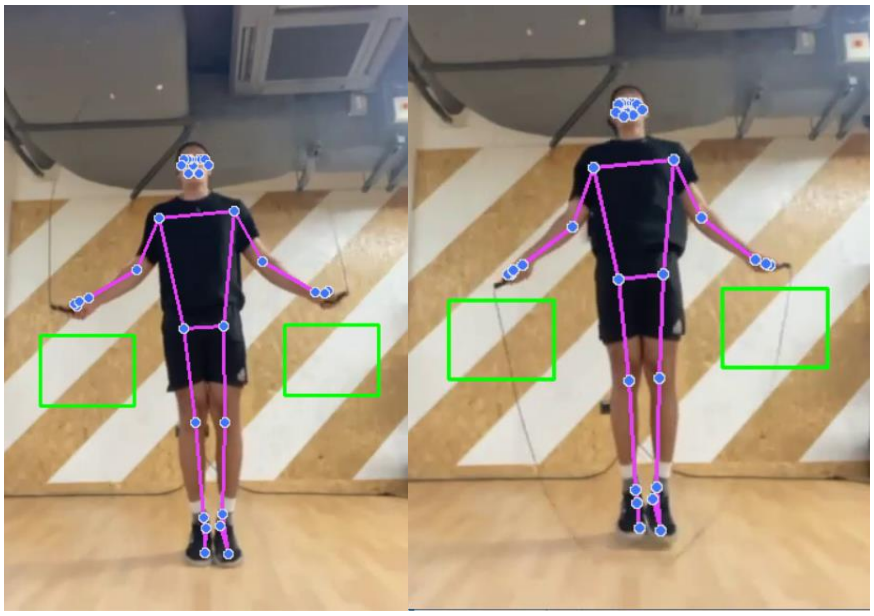

Figure 4-5: Region of interest under wrists

To ensure that the system remains accurate and responsive, the position of the window is rechecked and adjusted every 90 frames. This frequent updating enables us to account for any lateral movements performed by the subject jumping rope, as well as changes in their position while they're in the frame. For instance, if a jumper is jumping in location 1 in front of the camera for 200 frames and then moves to a new location within the frame by frame number 210, the window described by the location of the wrist to detect the rope would become redundant if it remained stationary in position 1. Consequently, updating the location of the window every 90 frames enables us to continually check for key points within whose vicinity we can detect the presence of the rope.

4.3 Video pre-processing (frame pre-processing)

Once the location of the window has been determined, it is necessary to perform additional image processing on the video in order to accurately detect the presence of the rope. This is because a computer is unable to inherently differentiate between a rope and other moving objects. As such, the processing of the video in such a manner enables distinction between these two types of objects and accurate detection of the presence of a rope.

4.3.1 Frame Conversions

To accomplish this, we first convert our colour video into grayscale[2.3.2]This conversion is necessary because the large colour spectrum of 256x256x256 makes our working space complex in terms of the number of colours and amount of information we must process [16]. By converting the video into grayscale, we are able to simplify this information and make it more efficient for us to work with. This enables us to more effectively process the video and accurately detect the presence of a rope.

The grayscale conversion process reduces the information in the image from a complex colour spectrum of 256x256x256 to a single channel of 256 based on the greyness of the image pixels. This reduction in complexity enables us to more efficiently process the video stream without sacrificing any important information that could hinder the performance of our system.

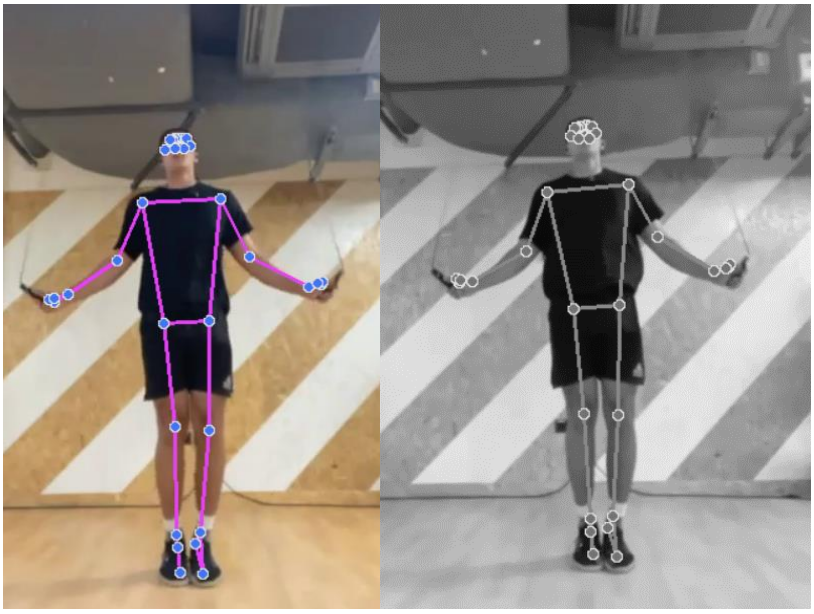

Figure 4-6: Normal Frame Converted to Grayscale

Once the videos are converted to grayscale, a technique known as thresholding is used to further process the image. Specifically, use of binary thresholding is made here to condense the image into two distinct types of pixels: foreground and background [Grayscale Images and Binary Thresholding 2.3.2]. This thresholding process enables distinction between objects of interest and their surroundings, but the application still lacks the ability to differentiate between the actual background and the foreground of interest.

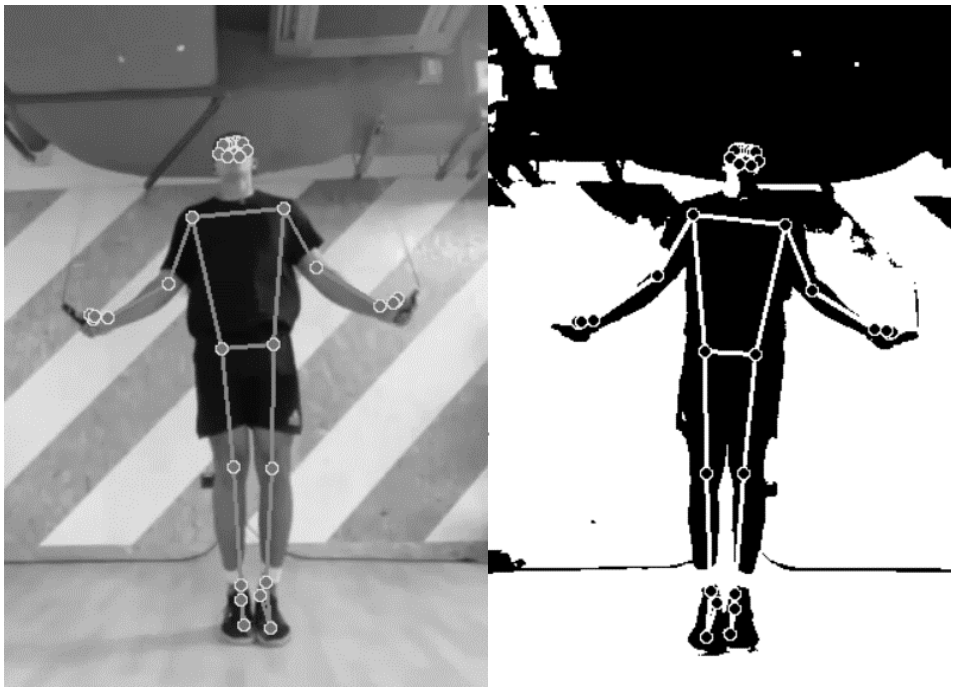

Figure 4-7: Grayscale Image converted to Binary Threshold Image

4.3.2 Applying Background Model

To address this issue, a background subtraction model is applied to grey scale image. In the analysis, k-nearest neighbours (KNN) background subtraction model is used [22]. This powerful technique enables accurate identification and isolation of moving pixels within the image that are of interest to us i.e. the foreground pixels. It can be seen in the images below; the rope and subject are clearly demarcated by white pixels representing the foreground. By utilizing this background subtraction model, the moving pixels are isolated and separated them from non-moving pixels, enabling focus on the pixels of interest.

Figure 4-8: Progression from normal frame to Knn Background Model

4.4 Working with Contours and Flags

Once the pixels of interest, the foreground are successfully identified using the sophisticated KNN background subtraction model, the pixels are aggregated and edges are drawn around them. These edges are commonly referred to as contours[2.3.3] and represent a curve that joins all continuous pixels points within the image that share similar colour and intensity values Figure 4-9. By accurately identifying these contours and determining their locations within the image, the position of various shapes and objects can be mapped, including the hand, torso, and rope.

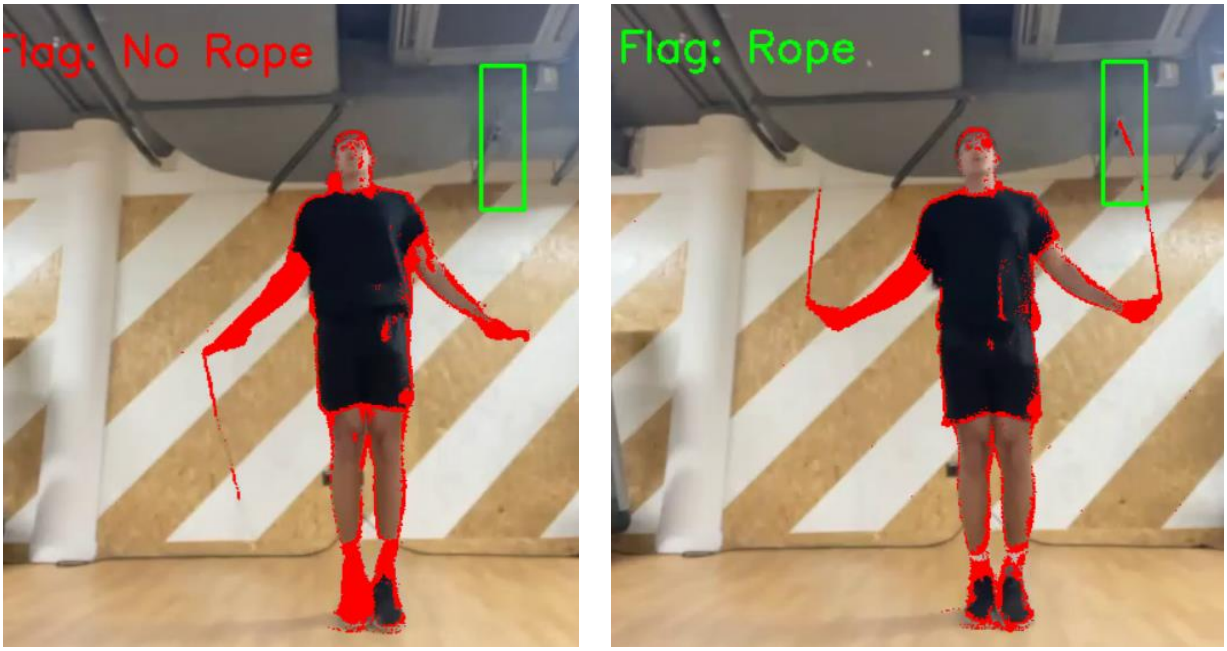

Figure 4-9: Contours Drawn on the Subject

After locating these contours, the next step is to check whether they are present within the window that have previously been defined using human pose estimation techniques [3.1]. If a contour is detected within this window, a flag is set to indicate that a rope contour has successfully passed through the window. This flag remains true for as long as the contour is present within the window and switches to false when no contour is detected.

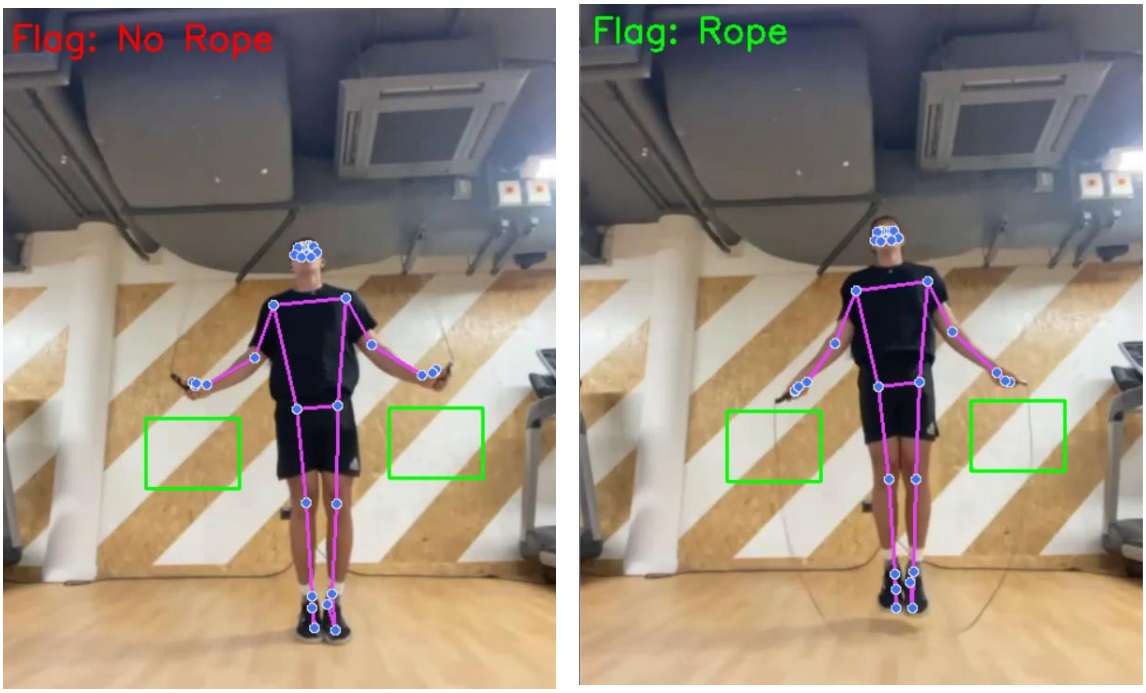

Figure 4-10: Flag Detection using contours

In this implementation of this system, additional care was taken to place windows of interest on either side of the wrist. This enables effective check for the presence of the rope in both hands, which is particularly important in edge cases where the rope may only be present in one hand. Once the regions of interest are successfully located and the respective flags for both regions are set, these flags are carefully monitored throughout the video to determine whether a rope is present.

In this implementation, flags are checked every 90 frames. This provides sufficient time for the jumper to detect the rope passing through the region of interest in both hands. By carefully monitoring these flags and checking their values at regular intervals, accurate detection of the presence of a rope within our regions of interest is observed.

4.5 In case of No Rope:

In the event that no rope is detected in a video of an individual jumping rope, the process described above remains largely unchanged until the final stages. Our analysis begin by obtaining a video and carefully extracting a frame from it. This frame serves as the basis for the subsequent processing and analysis.

Figure 4-15 No Rope: Normal Jump

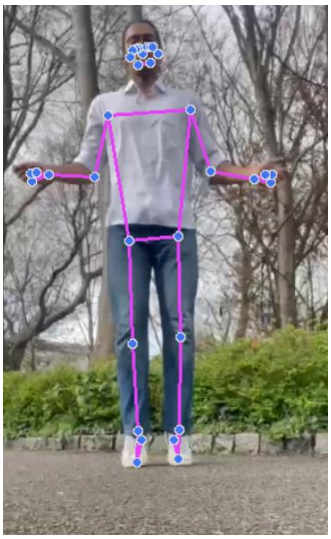

Figure 4-14 No Rope: Applied Human Pose Estimation

Figure 4-11 No Rope : Grayscale

Figure 4-12 No Rope: Background Subtraction for KNN

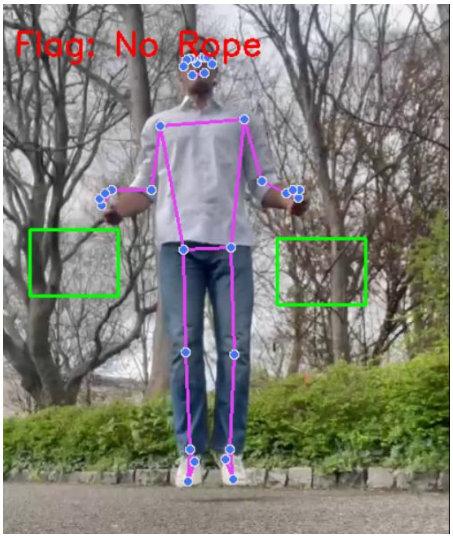

Figure 4-13: No rope Detected

Once the image frames are obtained, Human Pose Estimation Techniques is applied to the image in order to accurately determine the location of the wrists. These wrist locations are then used to define a regions of interest within the image. Next, image is converted to grayscale in order to simplify the information contained within it and make it more efficient for us to process

Finally, KNN background subtraction model is applied to the grayscale image. Application of the background subtractor model using KNN enables accurate identification and helps isolate moving portions of the image that are of interest. On obtaining the KNN background subtracted frame image, search for contours passing through our regions of interest is initiated.

In the absence of a rope, no contours should be detected within these regions of interest. As it can be seen in this figure above, when no rope is present, the flags for both regions of interest are false. Given that no rope is detected in the video, the counter does not increase.

This illustrates the behaviour of the application when no rope is detected in the video and demonstrates its ability to accurately distinguish between videos with and without a rope present.

5 Jump Classification

5.1 Introduction

In this section methodologies behind detecting different kinds of jumps is discussed. In this project, detection of different types of jumps was an objective, which makes the system novel when compared to different types of applications already available in terms of detecting different types of jumps. I discuss the different types of jumps as follows:

- Crossover Jump
- High Knees
- Jumping Jacks

The logic used behind these different kinds of jumps would be discussed in detail in the subsequent pages of the thesis

5.2 Criss Cross Jump

5.2.1 Description

A crossover jump is a jump-rope technique in which the arms cross in front of the body as the rope passes underfoot. As the rope descends, the jumper must quickly cross one arm in front of the other so that the rope passes cleanly under the crossed arms. The arms are then uncrossed as the rope returns, and the jump is repeated. The figure below depicts how the jump progress.

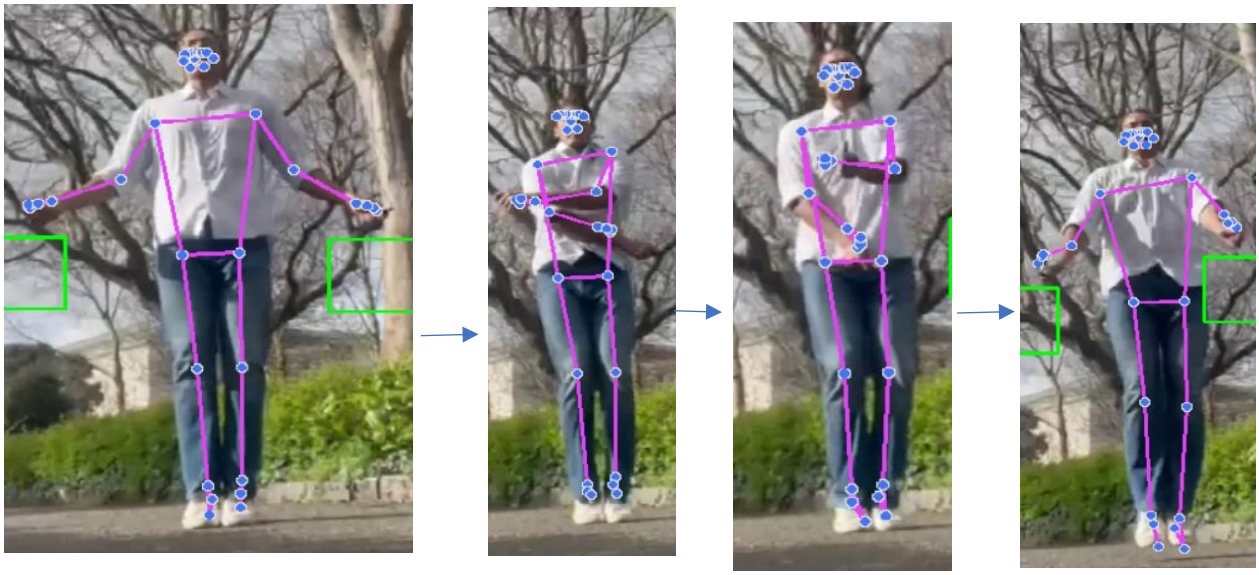

Figure 5-1: Criss Cross Jump

5.2.2 Methodology used

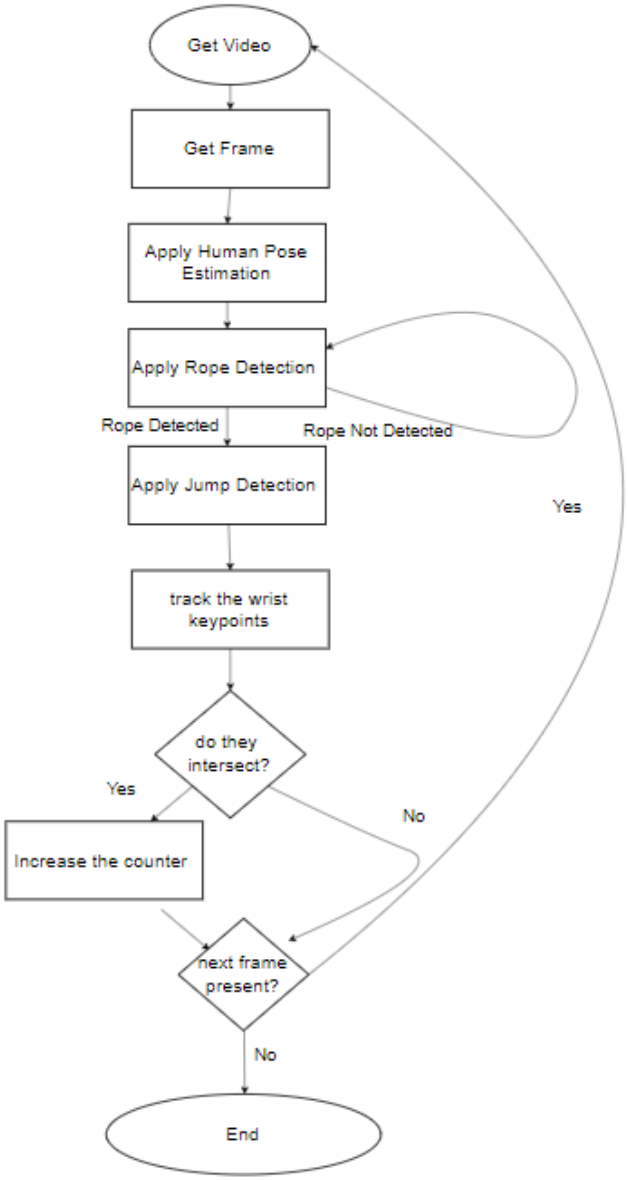

Figure 5-2: Flow Chart for Criss Cross

In our application, human pose estimation model from Mediapipe (3.1) is utilised to detect crossover jumps. Prior to this, we had implemented jump detection by tracking the ankles throughout the video and detecting the presence of a rope using a background model based on KNN. This allowed extraction of information about the rope and helped determine if it passed through a region of interest that tracked the user's wrists on both hands.

Having already implemented jump detection and rope tracking in our application, keypoints highlighted were used and detected by the human pose estimation model. By tracking the movement of both wrists, we were able to detect crossovers. We analysed curves representing the movement of various keypoints and determined that the crossing over of the x-coordinates of the

wrist keypoint allowed us to detect when a crossover jump occurred. The video frame had its axis defined as shown in the attached figure below.

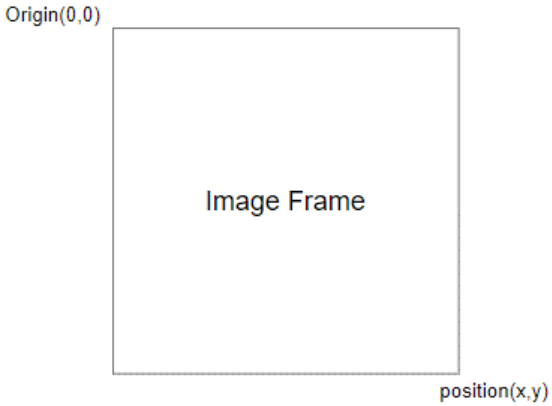

Figure 5-3: Frame Definition with origin defined

Considering the origin and axis as shown in Figure 3-5, it was evident that the x-coordinate value corresponding to the left wrist would always be greater than that corresponding to the right wrist due to the position of the subject in the video. This enabled us to pinpoint moments when the hands crossed in front of the jumper. A valid crossover jump was detected when the hands crossed over one another and then returned to their natural x-coordinate positions, with the right wrist being greater than the left wrist keypoints.

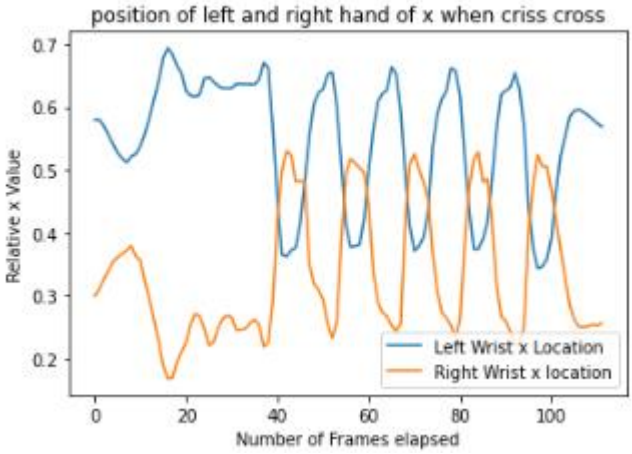

Figure 5-4: X coordinate corresponding to the left and right wrist's position during Criss Cross Jump

It can be seen from the figure above how the x coordinates corresponding to the left and right wrist keypoints intersect throughout the motion. When this intersection is detected between the left wrist x position and the right wrist x position, the counter for the number of Criss Cross Jump increases.

Figure 5.4, the subject can be seen performing normal jumps. By tracking the keypoints from this array of figures, we were able to analyse the curves obtained for the x-coordinates corresponding to the left and right wrists. It was observed that the curves never crossed over and that the x-coordinate of the right wrist was always less than that of the left wrist. This confirmed the effectiveness of our technique for detecting crossover jumps.

Figure 5-5: Performing a normal Jump

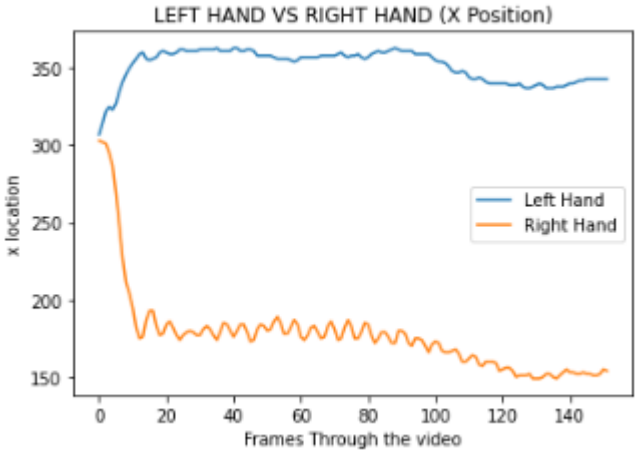

Figure 5-6: X coordinate comparison of left and right Hand movement during Normal Jump

5.3 High Knees Jump

In the previous section, the methodology behind detection of Criss Cross Jump using jump rope was defined. This section will introduce High Knee jumps in context of skipping rope and discuss the methodology behind the Classification technique developed.

5.3.1 Definition

A high-knee jump was defined as having taken place when one knee was off the ground and brought back down again. When both knees have performed this action, one valid high knee is said to have occurred.

5.3.2 Methodology

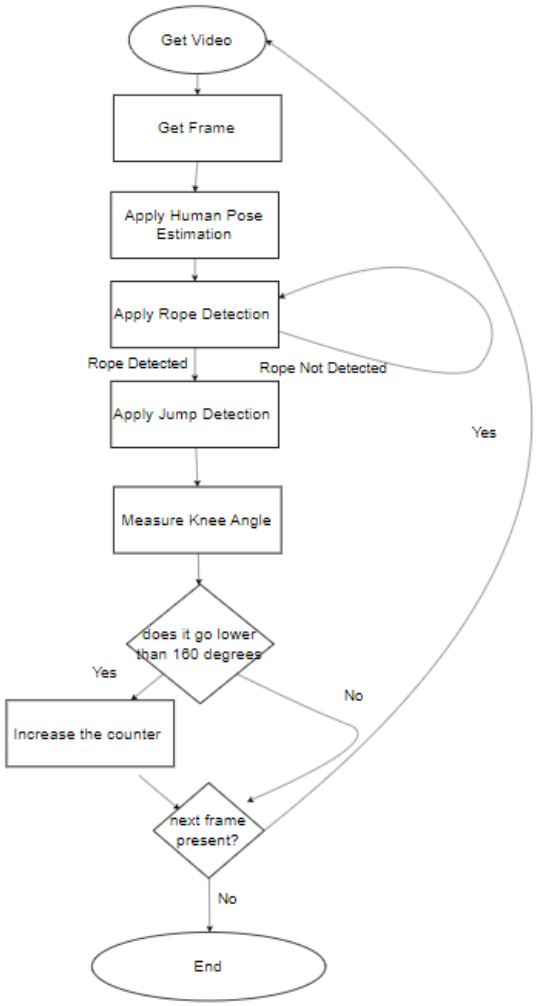

Figure 5-7: Flow Chart for High Knees

The application utilizes a human pose estimation model from Mediapipe [23] to detect crossover jumps. Prior to this, we had implemented jump detection by tracking the ankles throughout the video and detecting the presence of a rope using a background model based on KNN. This allowed us to extract information about the rope and determine if it passed through a region of interest that tracked the user's wrists on both hands.

In order to identify high knee jumps, we applied a logic that involved analysing the angle formed by the left hip, left knee, and left ankle which are highlighted in red and angle between them is highlighted with the yellow angle.

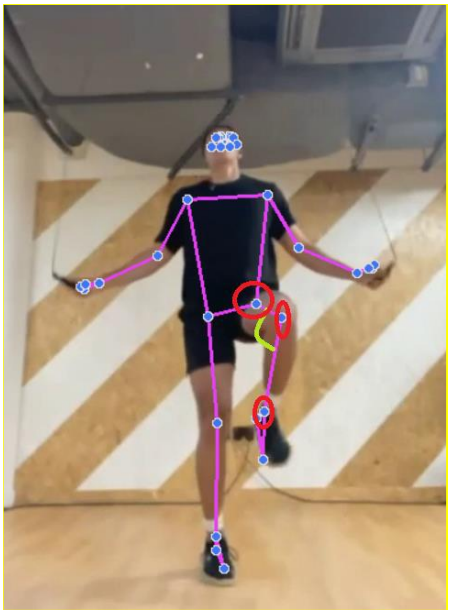

Figure 5-8: Angle which is tracked at the knee

Arctan technique to calculate angles between these three points was utilised [2.3.4] .The angles were then plotted against time to determine if there was a relationship between the angle being a certain degree and the occurrence of a high knee jump.

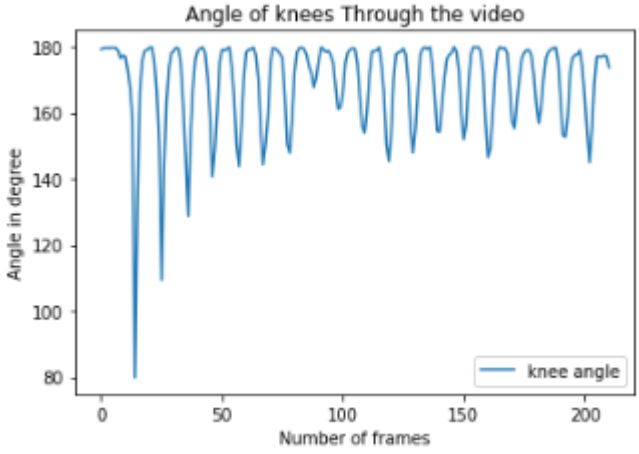

Figure 5-9: Angle made by the knees through the videos

Figure 5-9 Depicts a sinusoidal curve in which the amplitudes vary based on the angle formed by the three points. By comparing the angles between these points with the ground truth of our sample training videos, it was that determined that a valid high-knee jump occurred when the angle between these three points was less than 160°. This was due to the fact that when the subject jumped normally in front of the camera, the angle between these points was very close to 180°, as they were aligned in a straight line due to the natural positioning of the hip above the ankle as shown in the Figure 5-9 below:

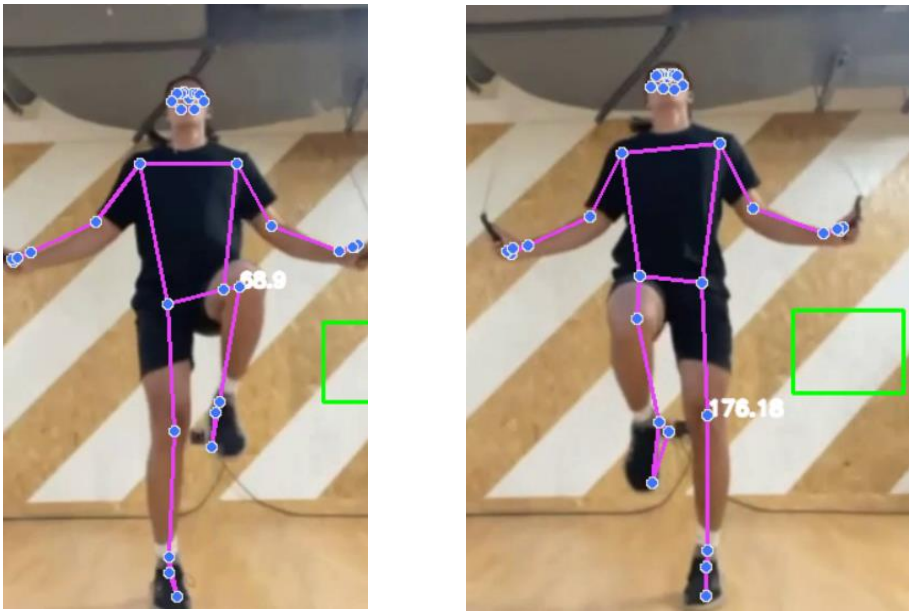

Figure 5-10 : Angle Made while Doing High Knees

During a high-knee jump, the angle between the three points (right knee, centre of the pelvis, and left knee) contracted as the position of the knees changed relative to these points. This change was recorded and analysed to detect the high-knee jump. The counter for high-knee jumps increased only when the angle between these three points decreased below 160° and then returned above 160°.

It is worth noting that there may be some confusion regarding how high-knee jumps were counted. While some may consider a high-knee jump to have occurred when both feet land on the ground, for the purpose of this project, the counter for high-knee jumps was updated when one foot followed suit with the jump.

5.4 Jumping Jack

In the previous section, the methodology behind detection of High Knee Jump using jump rope was defined. This section will introduce Jumping Jack jumps in context of skipping rope and discuss the methodology behind the Classification technique developed.

5.4.1 Description

To execute a valid jumping jack, an individual would begin by holding a jump rope and jumping with both feet conjoined while swinging the rope around their body. As the rope descended, the individual would jump their feet apart while simultaneously extending their arms outwards and upwards over their head, mimicking the motion of a standard jumping jack exercise. The individual would then jump their feet back together as the rope completed its rotation and continue to repeat this motion for several repetitions or for a predetermined duration.

5.4.2 Methodology

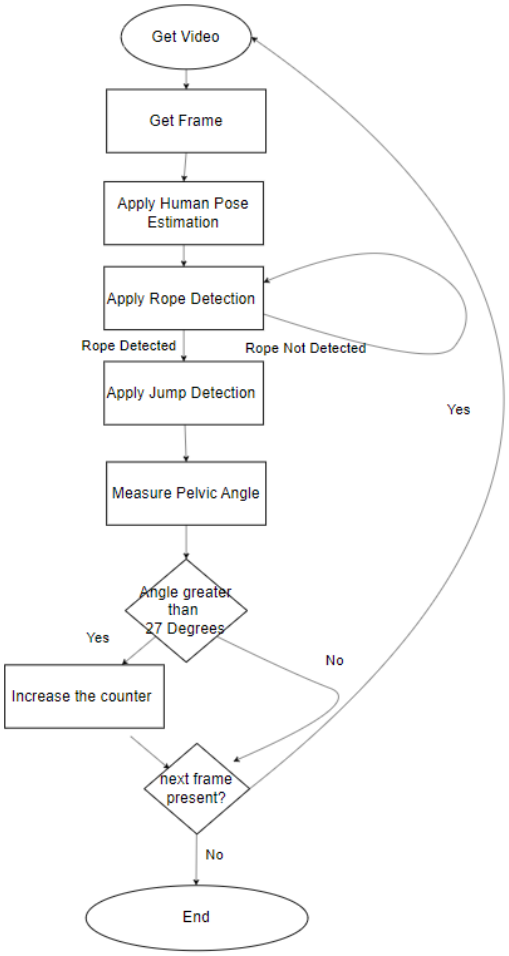

Figure 5-11: Flow Diagram for Jumping Jack

In order to detect jumping jacks while performing jumps, keypoints obtained from Mediapipe's human pose estimation are utilised. Analysis of the angle between the two knees and the centre of the pelvis, as shown in Figure 5-12 below was done. Since the centre of the pelvic region was not automatically tracked by Mediapipe, its location was calculated using the x and y coordinates of the left and right hip.

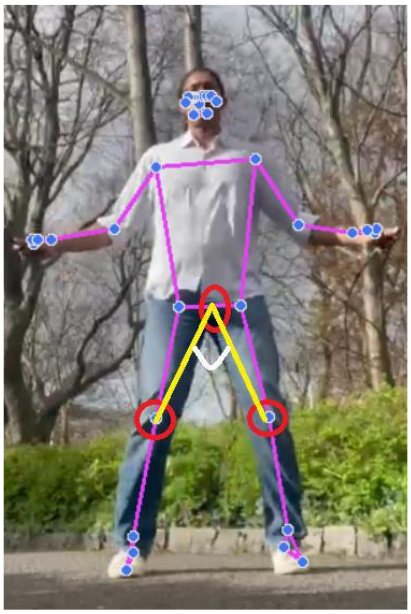

Figure 5-12: Angle Drawn for Detecting a valid Jumping Jack

Using the coordinates of all three points, the angle formed by the right knee, centre of the pelvis, and left knee (shown in red in Figure 5-12) was tracked. By analysing this angle throughout the video, it was determined that an angle of 27° worked best. The counter for the number of jumping jacks increased when the angle between these three points exceeded 27° and then decreased again. The graph below depicts the angles formed to detect the occurrence of a jumping jack throughout the video, and it can be observed why the 27° angle works well.

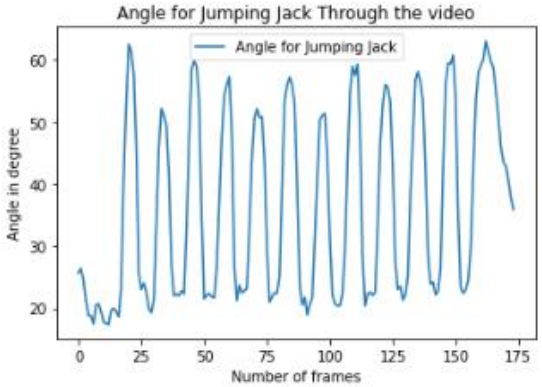

Figure 5-13: Angle Observed at the centre of Pelvis through the Jumping Jack video

In figure below, it can be observed how the angle between the left knee, right knee, and centre of the pelvic area changed for a person performing normal jumps. It was noted that the angle did not change significantly because the feet remained conjoined, resulting in an angle that remained close to 12 degrees for most jumps.

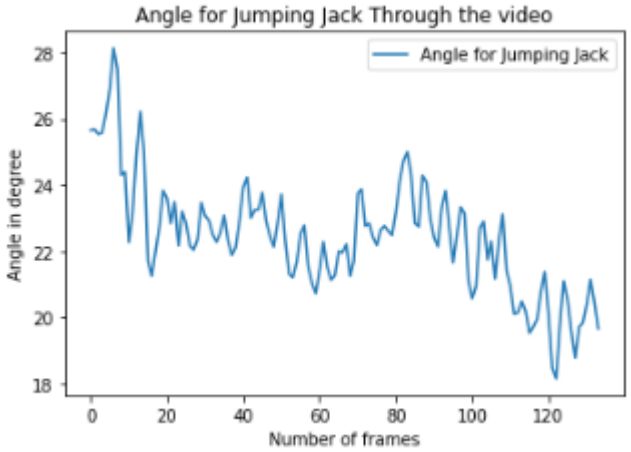

Figure 5-14: Angles Detected at the centre of the pelvic region for Normal Jumps

6 Evaluation

Previous sections discussed the methodology behind the whole application and saw how each section played a part in classifying the different kinds of jumps taking place. Now, in this section the evaluation criteria and evaluation metrics for the final application developed have been discussed.

For the development of this project videos of myself have been used. The project testing has been done on the same videos and new videos were also recorded for further testing of the project. The ground truth was self-generated and has been used to test the algorithm. Kinovea has been used to annotate the videos and get the ground truth for the testing of the application.

Video is being processed between 15 frames per second using OpenCV and the Ground Truth is based on the videos recorded at 30 frames per second(fps). The count of the frame is used actively in this evaluation to easily substitute passage of time during the processing of the video.

6.1 Jump Detection

6.1.1 Accuracy

In section [3.3] a discussion on the methodology behind the jump detection technique is done. In this section testing of the jump detection technique on various parameters and evaluate the performance and robustness of the technique is done. For the testing of the technique, 4 videos have been used in the evaluation of the technique used.

Table 6-1: Normal Jump Performance

Video Name	Actual Jumps	Detected Jumps	Accuracy
Normal Jumps 1	56	58	96.42%
Normal Jumps 2	78	77	98.71%
JumpRopeChairAngle	59	60	98.3%
HouseJumpCut	23	23	100 %
Total	216	218	Avg : 98.35%

In the table shown above, the number of jumps performed in each video has been aggregated against the number of jumps detected by the algorithm to get the average accuracy of 98.35% for the technique implemented.

6.1.2 Timing

The table below shows us the ground truth for one of the videos used for evaluating the project. Every jump has an associated number with its name which signifies the jump number. The Ground Truth Frames column represents the frame number at which the respective jump was recorded when the ground truth was formed. Frame numbers are used in this analysis as they are easier to understand and represent well the passage of time in this analysis.

Table 6-2 : Ground Truth for Normal Jump

JumpState	Timing	Ground Truth Frames	Algorithm Detected Jump Frame
DownNJ1	1.40	21	3
DownNJ2	1.77	26	18
DownNJ3	2.23	33	25
DownNJ4	2.60	39	32
DownNJ5	3.00	45	39
DownNJ6	3.33	49	45
DownNJ7	3.76	56	52
DownNJ8	4.16	62	58
DownNJ9	4.60	69	64
DownNJ10	5.00	75	70
DownNJ11	5.33	79	76
DownNJ12	5.66	84	82

The frame number corresponding to the jump detected by the application is the **Algorithm Detected Jump Frame column** in the above Table [Table 6-2 : Ground Truth for Normal Jump]. It consists of the frame number at which the application has flagged the occurrence of a valid Normal Jump.

"The Algorithm Detected Jump Frame" and "**Ground Truth Frames**" columns, both consist of the frame numbers for the jumps detected by the application and the recorded ground truth. The frame numbers received from these two columns of data are aggregated against each other to observe and understand when the jumps are taking place within the provided video.

The figures depicted below represent the time(frames here) at which each jump is detected. A scatter plot is drawn between the ground truth and the detected truth. A comparison between the hip and ankle keypoints is made below to demonstrate how both the keypoints are detecting the jumps as seen earlier how well both these points perform in counting the number of jumps.

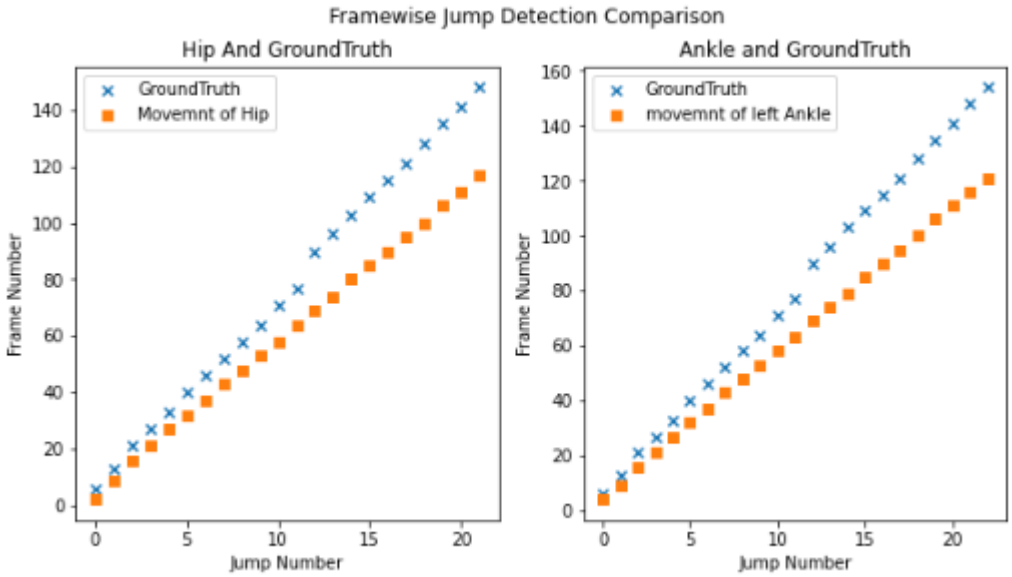

Figure 6-1 Comparison of performance for Normal Jump Video

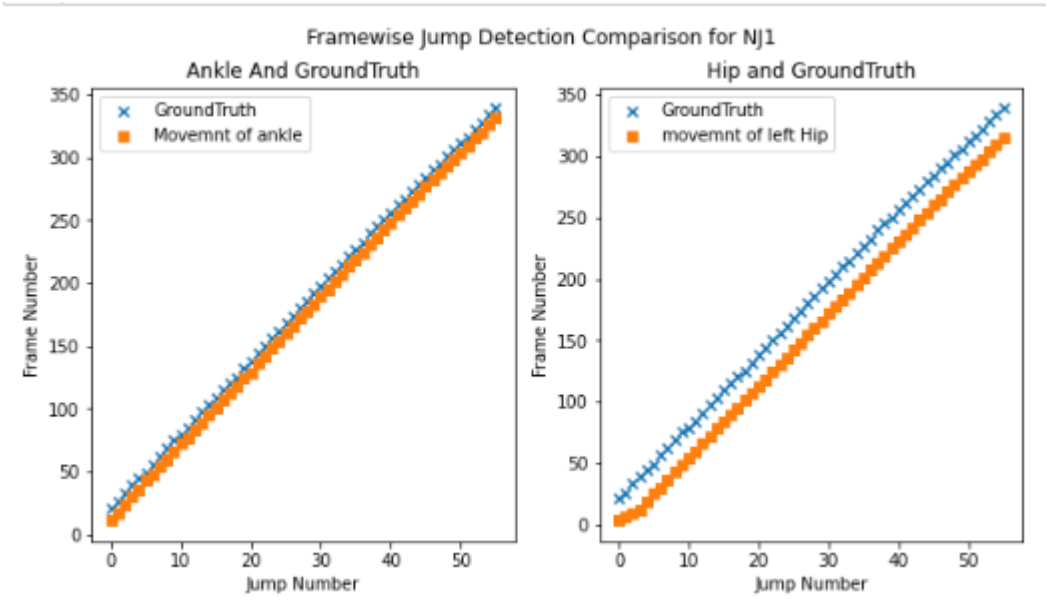

Figure 6-2 Comparison of performance for the video Normal Jump 1

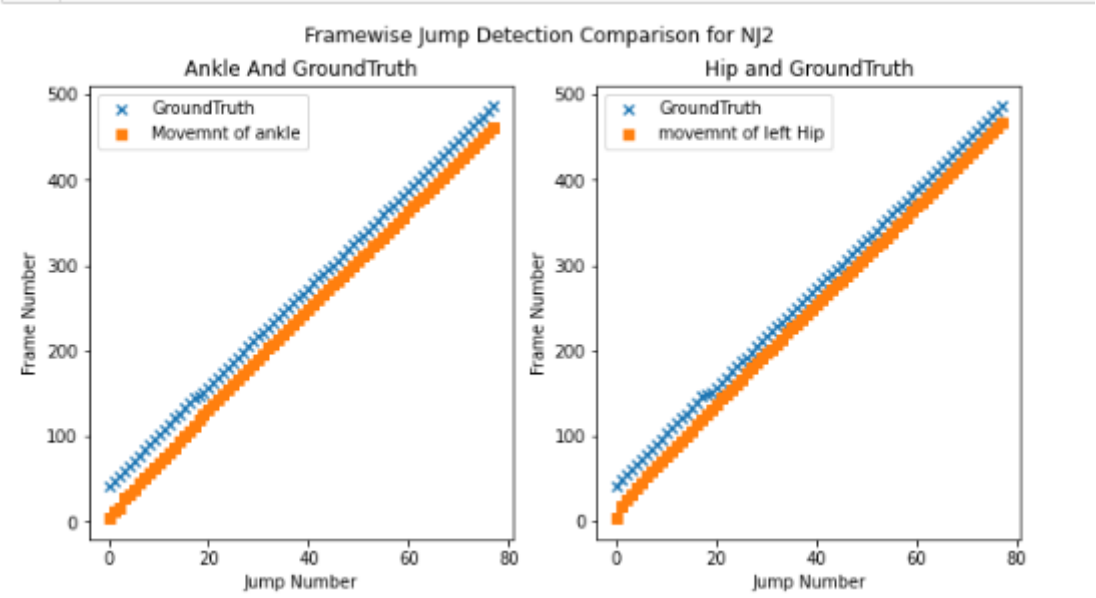

Figure 6-3: Performance Comparison for Normal Jump 2

The above figures exhibit with help of a scatter plot the timestamps at which Normal Jumps are being detected in the videos which have been used for the testing of the application. The gap between the curves depicts how close to the actual frames are the jumps being detected. Hence, on closer observation of the curves, it is concluded for some videos the average time difference between the detection of the actual jump happening it being detected is as follows:

- Ankle : **15** frames i.e. **500ms**
- Hip : **21** frames i.e. **700ms**

6.1.3 Limitation

The application achieves high success rate when the video being analysed is well labelled and has the ideal conditions such as no moving background, singular person in the video, good video quality, visible and identifiable features such as identifiable wrists and legs. In those situations the model has performed extremely well and hence it's a robust solution when the conditions are ideal.

However, the application is far from perfect as it does miscount the number of jumps when the subject moves closer and away from the camera or makes irregular movements such as walking around within the frame and/or spinning. The irregular movements such as walking away and towards the camera, walking around in the frame have been accounted for and has its respective mitigation in place. A threshold value is used in the implementation which was determined by trial and error given the dimension of the videos processed.

6.2 Rope Detection

This section discusses the evaluation of the rope detection technique. The ground truth for the rope detection was generated for different videos and checked. The presence of rope is detected every 90 frames, 3seconds in the videos used since they were recorded at 30fps. The ground truth records all the frames in which the rope appears to be in the window where the contour for the rope are being detected. As the check for a rope occurs every 90 frames, the observed frames in the ground truth serve as a basis of check every 90 frames. If a rope is detected to be in the window where the contours are being checked then that is labelled as a correct classification. So for a video of 45 seconds there are 15 checks present for the rope.

6.2.1 Accuracy

The table below shows the number of times the Rope has been detected in the frame in the various videos.

Table 6-3: Performance of Rope Detection

Name of video	Total number of frames	Checks done	Number of checks in which rope is detected	Accuracy
House Jump Cut	1385	15	15	100%
Jumping Jack 2	1007	11	9	81%
High Knees 3	1245	13	12	92%
Mixed Jump	1758	19	17	89%
Normal Jumps	854	9	9	100%

This shows that the number of times ropes are being detected in the window are very accurate, only missing the windows sometimes. The average accuracy in the videos for which the ground truth was recorded is 92.4%.

	Rope Detection Frame Number
0	24
1	25
2	37
3	48
4	59
5	69
6	70
7	80
8	90
9	101
10	111
11	122
12	132
13	143
14	154
15	163
16	164
17	174
18	175
19	185

Figure 6-4: Ground Truth for the Rope detector

6.2.2 Limitations

The methodology for evaluating the rope detection technique is stated above [6.2]. There are videos in the testing dataset where the foreground and background are not being able to distinguished by the algorithm due to the non-static nature of the background in those video, because of which the background pixels are being misclassified as rope contours. This serves as a

limitation for the detection of rope in our application and requires the development of a more robust rope detection technique in which the contours from the background are not detected as rope.

Additionally, due to the lateral movement of the subject in frame it can be observed that rope is not being detected since the rope is moving away from the window within which these checks are made : this is encountered in the 90 frames before updating of the windows location.

6.3 Jump Classification

This section discusses the methodology used to test the different Jump classification techniques which were discussed above [5].

6.3.1 High Knees

6.3.1.1 Accuracy

The table below shows the number of High knee jumps which were detected in the various videos using the algorithm against the number of jumps in the ground truth. The mean accuracy achieved using this technique for detecting the number of High Knee jumping technique is 97.48%.

Table 6-4: High Knees Performance Summary

Video name	Actual Jumps	Detected Jumps	Accuracy
High Knee 1	15	16	93.33%
High Knee 2	31	32	96.77%
High Knee 3	16	16	100%
High Knee 4	20	20	100%
High Knee 5	37	36	97.3%
Total	119	120	97.48%

6.3.1.2 Timing

	JumpState	Timing	Ground Truth Frames	Algorithm Detected Jump Frame
2	DownHK1	4.70	70	89
3	DownHK2	5.43	81	101
4	DownHK3	6.10	91	112
5	DownHK4	6.76	101	124
6	DownHK5	7.50	112	135
7	DownHK6	8.16	122	146
8	DownHK7	8.86	132	158
9	DownHK8	9.50	142	169
10	DownHK9	10.13	151	181
11	DownHK10	10.76	161	192
12	DownHK11	11.36	170	204
13	DownHK12	12.03	180	215
14	DownHK13	12.69	190	226
15	DownHK14	13.36	200	237
16	DownHK15	14.03	210	261
17	DownHK16	14.73	220	272

Figure 6-5: Ground Truth for High Knee

The above table depicts the Ground Truth for one of the High Knee Videos. In this table the jumps and their count are mentioned in the Jump State Column. Frames depict the frame number at which the jump occurred while recording the Ground Truth. This frame number captured in the ground truth is used here to observe when the jump has taken place during the video. Algorithm Detected Jump Frame shows when the jump is detected in the video using the algorithm developed.

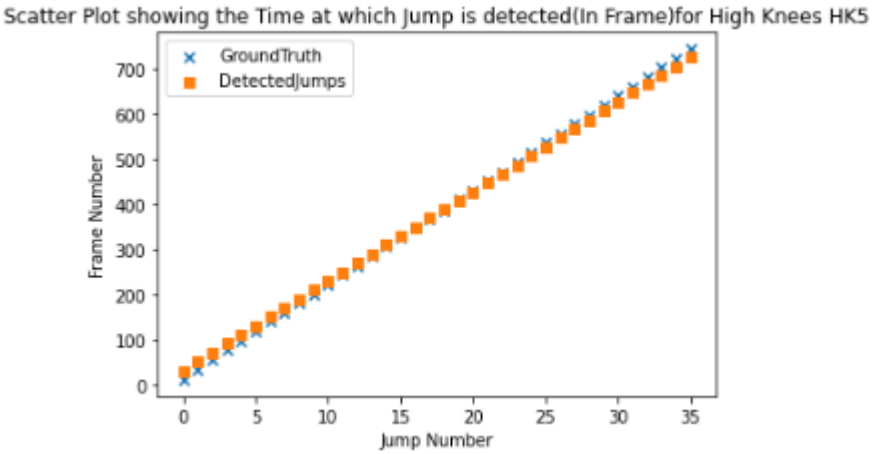

Figure 6-6: Performance comparison for High Knees 5

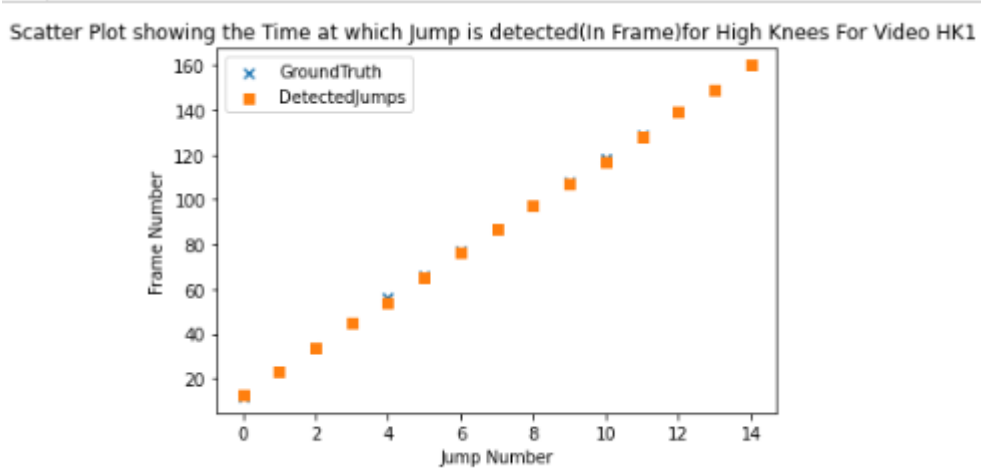

Figure 6-7: Performance comparison for High Knees 1

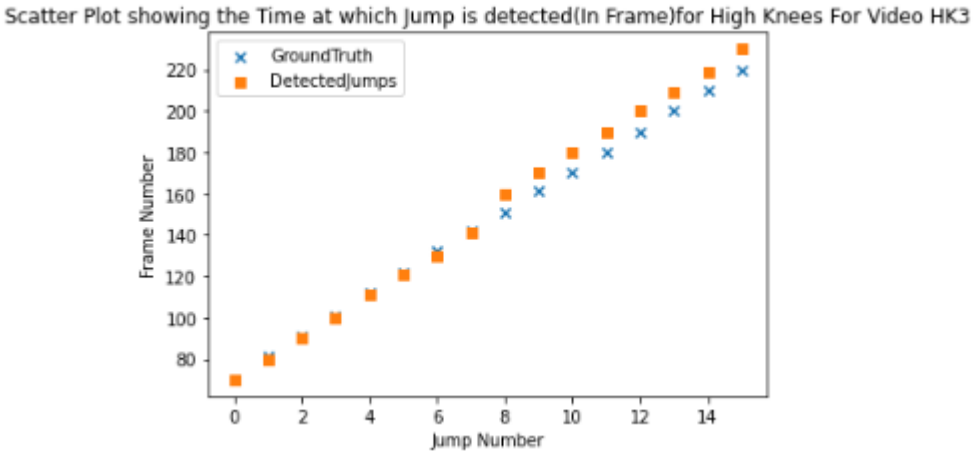

Figure 6-8: Performance comparison for High Knees 4

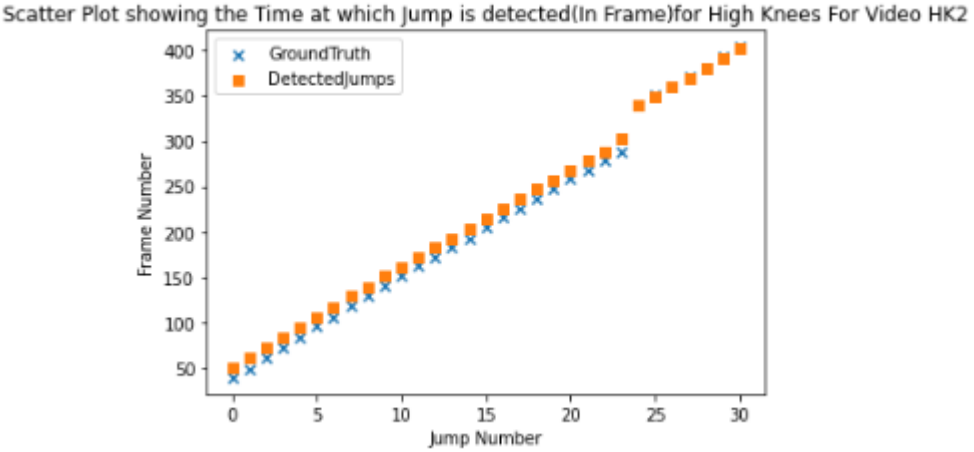

Figure 6-9 5: Performance comparison for High Knees 3

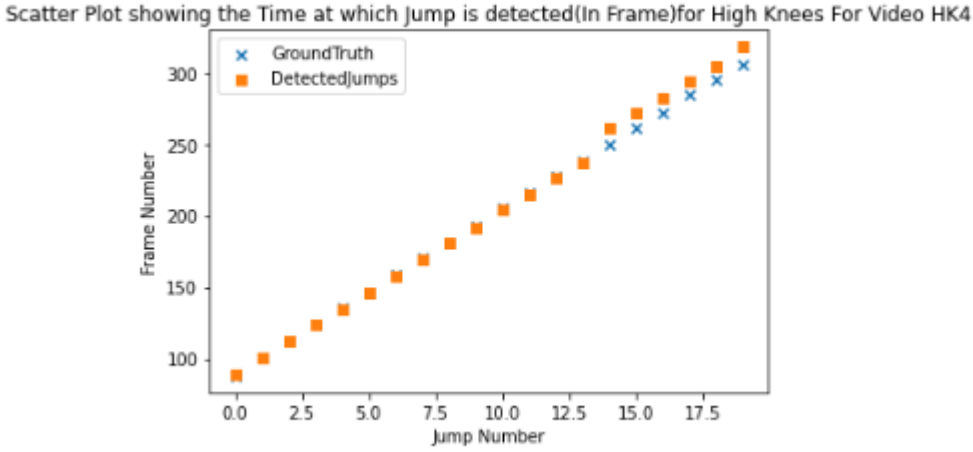

Figure 6-10: Performance comparison for High Knees 2

Figure depicts a scatterplot which helps us to understand the time at which each jump is being detected as compared to the Ground Truth for the jumps. The closer they are on the y-axis the closer they are to each other in terms of time when they are detected. It is observed from the above figure that the timings being detected are quite accurate when an adjustment to the frame numbers has been made. There is an average difference of 5 frames which is 0.1second for a video which has a frame rate of 48 in the timing of the jump detection.

6.3.2 Jumping Jacks

6.3.2.1 Accuracy

Table 6-5: Jumping Jacks Performance Summary

Name	Actual Jumps	Detected Jumps	Accuracy
Jumping Jacks 1	12	12	100%
Jumping Jacks 2	23	21	91.3%
Jumping Jacks 3	13	13	100%
Jumping Jacks Main	12	13	92%
Total	60	59	95.85%

The above table shows the performance of the algorithm in detecting the number of jumping jacks being performed in a video. The results are from 5 different videos in which the number of jumping jacks being performed are counted and aggregated against the count detected by the algorithm. The final average accuracy of the algorithm in detecting the performance of Jumping Jacks is 95.85%.

6.3.2.2 Timing

Table 6-6: Ground Truth Jumping Jack

	JumpState	Timing	Ground Truth Frames	Algorithm Detected Jump Frame
2	JJ1	1.36	20	19
3	JJ2	2.20	33	32
4	JJ3	3.06	45	44
5	JJ4	3.86	57	57
6	JJ5	4.73	70	69
7	JJ6	5.62	84	82
8	JJ7	6.42	96	95
9	JJ8	7.22	108	107
10	JJ9	8.12	121	120
11	JJ10	8.95	134	133
12	JJ11	9.75	146	146
13	JJ12	10.58	158	157

The table shown above depicts the Ground Truth of a video in which jumping jacks are being performed. Based on the ground truth generated for the video, a comparison of timing of jump detection by the algorithm is made to better understand the performance of the system. In the given table Jump State represents the number of Jumping Jack jumps in the ground truth of the video. Ground Truth Frames are the frames at which the jumping jacks were recorded to have taken place while making the ground Truth. Algorithm Detected Jump Frame is the frame which has the count for the detected Jumping Jacks

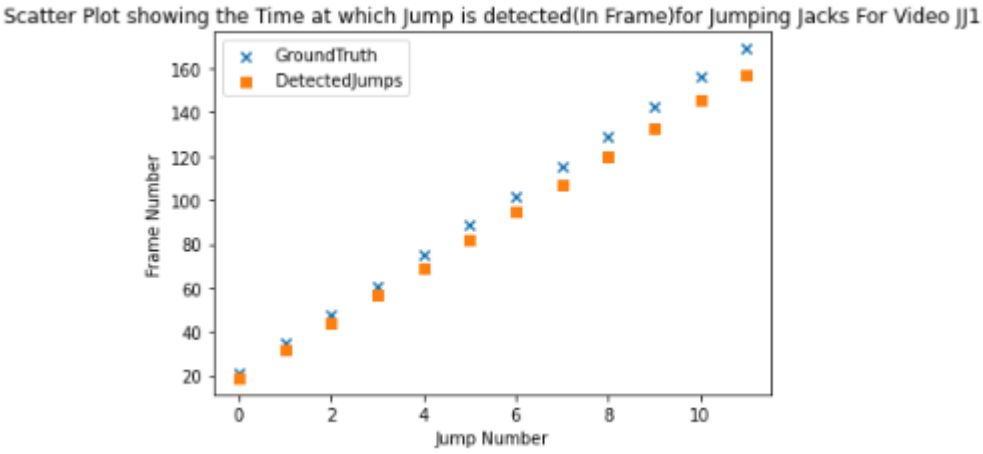

Figure 6-11: Timing for Jumping Jack video 1

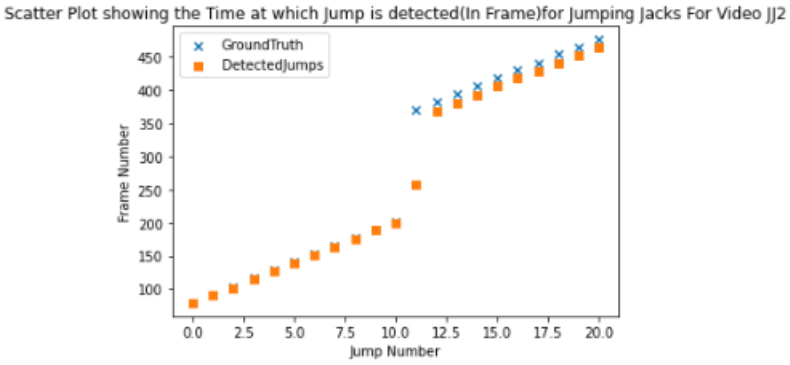

Figure 6-12: Timing for Jumping Jack video 2

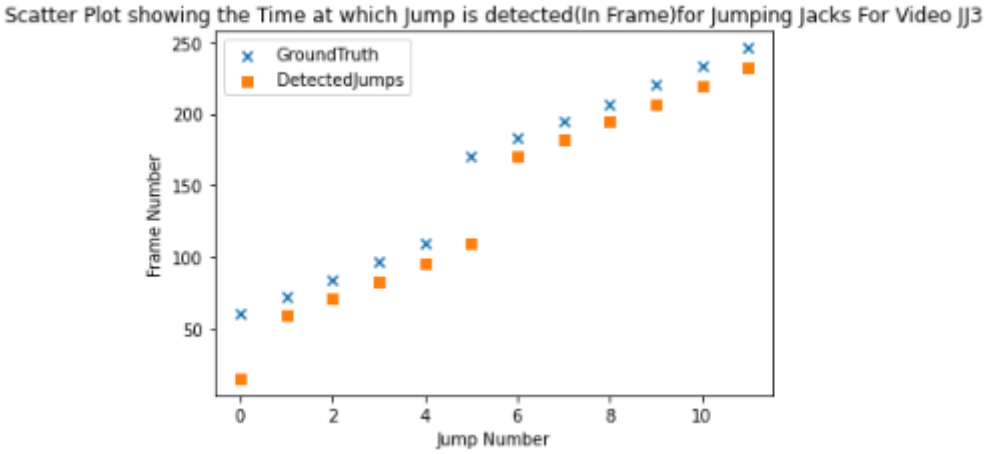

Figure 6-13: Timing for Jumping Jack video 3

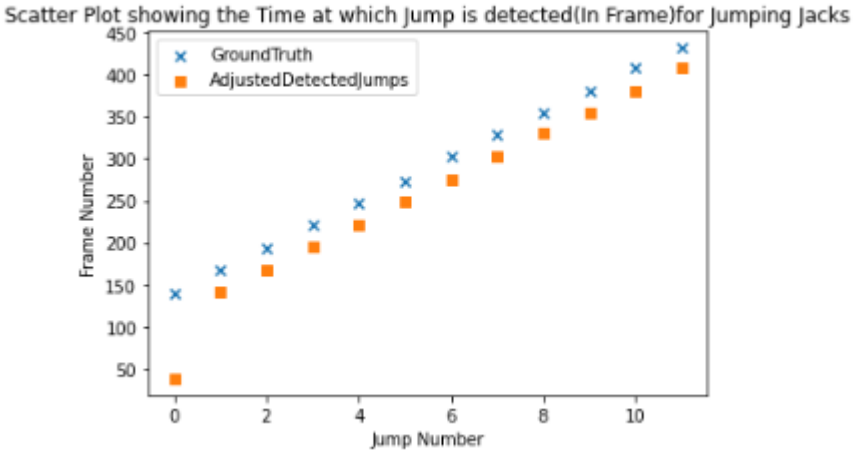

Figure 6-14: Timing for Jumping Jack video 4

The above curve compares the frame number at which the jumps are detected with the frame number at which the jumps are labelled in the Ground Truth. It is observed from the above figure that even with adjustments made due to the varying timeframe, a significant difference of [] frames

are observed on average in timing the detection of the jump. This means that in the video the jumps are being detected on an average [] frames after the jump has taken place i.e. after []seconds.

6.3.3 Criss Cross

Criss Cross Jump detection technique has not been tested as extensively as the other jump classification techniques since it is an advanced jump rope technique and I wasn't able to perform it consecutively during the collection of Ground Truth Videos. Although that is an indicator of poor testing, the logic applied for a valid cross rope is such that a valid Criss Cross Jump will always be detected due to the nature of the logic which is used. For a Criss Cross jump to occur the left hand should cross over the right hand for it to be performed. As long as the subject is visible in the frame, all the valid Criss Cross jump will be detected.

6.3.3.1 Accuracy

Table 6-7: Performance summary for Criss Cross Jump

Name	Total jumps	Predicted Jumps	Accuracy
Cross Rope	5	5	100%

The above table shows the performance of the algorithm in detecting the number of Criss cross being performed in a video. Only 1 video is tested due to the lack of ground truth. It can be observed a 100% accuracy is achieved here but the algorithm is quite prone to false positives.

6.3.3.2 Timing

	JumpName	Frames	JumpDetectionFrame
3	Cross 1	90	94
5	Cross 2	119	122
7	Cross 3	146	150
9	Cross 4	173	176
11	Cross 5	201	204

Figure 6-15: Criss Cross Ground Truth

The figure shown above depicts the Ground Truth of a video in which Criss Cross jumps are being detected. Frames represent the frame number at which the cross jump is observed to have occurred and JumpDetectionFrame signifying the frame at which the respective Criss Cross Jump is performed. The values from these curves was aggregated against each other to figure out the time at which the frames were being detected.

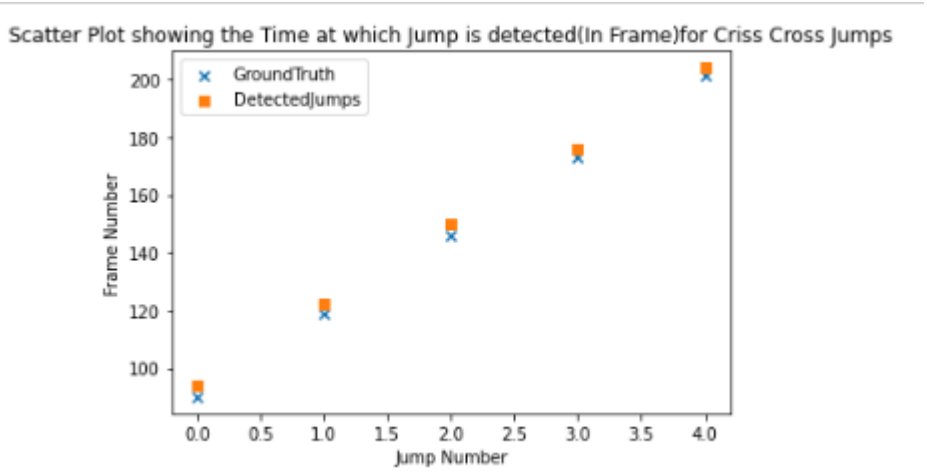

Figure 6-16: Timing for Criss Cross Video

The above Figure 6-16: Timing for Criss Cross Video compares the frame number at which the jumps are detected with the frame number at which the jumps are labelled in the Ground Truth. This means that in the video the jumps are being detected on an average 8 frames after the jump has taken place i.e. after 266ms

6.3.4 Shortcomings

This section discusses the shortcomings which are encountered in the 3 above mentioned jump classification techniques

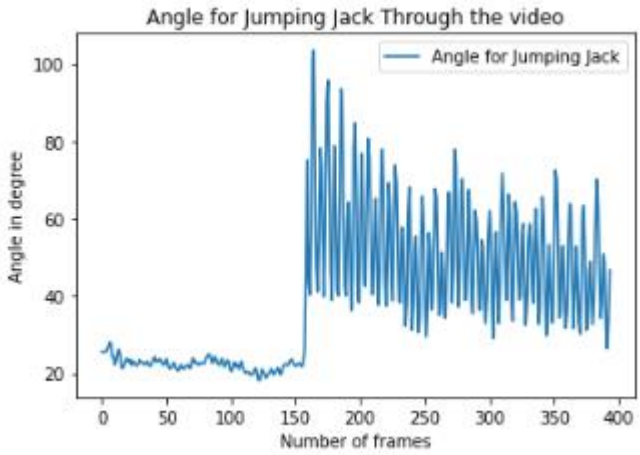

Figure 6-17: Angles detected for the pelvic area for a High Knee Jump

The above figure depicts the angle made recorded at the centre of pelvic region for a video in which High Knees occur. It can be observed that when the High Knees start appearing at 150 frames, the angle which is used to measure the angle for detecting Jumping Jack. Since both Jumping jack and High Knee both use angles to detect the number of jumps which have occurred, it is often

encountered that one of the other technique has been mistakenly counted up because of the erratic movement.

Additionally, another problem is present in the Criss Cross Jump Detection algorithm. Since the algorithm for Criss Cross Jump detects a valid Jump to have taken place when a cross over between the x coordinate corresponding to the left and right wrist is encountered. The problem arises when the count for Criss Cross jump increases when the subject crosses their hands above their heads for adjusting the rope when they are not jumping.

7 Conclusion

7.1 Review of application

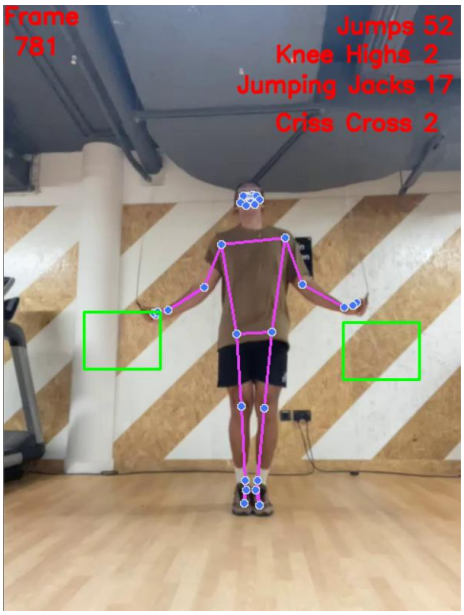

Figure 7-1: Final Application

The goal of this project was to develop a system which can accurately count the number of jumps which were performed in the input video along with the different kinds of jumps which were done as well. The system developed accurately counts the following jumps:

- Jumping Jacks
- Normal Jumps
- Criss Cross Jumps
- Normal Jumps

The project was broken down into 3 core functionalities that worked together to make the final system, those being detecting the occurrence of a valid jump, detecting the presence of a valid rope within the application and classification of different kinds of jumps when the first two functionalities are built and their necessary presence is fulfilled.

The jump detection technique made use of Human Pose Estimation to find the joint keypoints for the subjects, which were utilised to detect the occurrence of a jump. The movement of the left ankle and the hip of the subject was observed throughout the videos to detect the occurrence as well the number of jumps.

The rope detection technique made use of the background subtraction model to isolate the moving foreground pixels for further analysis. The moving pixels were contoured and checked if they pass through a given window. The contours are checked If they exist within a window which tracks the movement of the wrist every 90 frames. The detection of these contours validates the presence

of a jump rope and hence the system can move onto the next step of classifying the kind of jump taking place.

Jump classification is done after the previous two functionalities are detected to be present. The process of jump classification takes place using the keypoints obtained from the Human Pose Estimation model. The angle made between these keypoints is used to count the number of Jumping Jacks, High Knees and Criss Cross Jumps.

7.2 Conclusions

The system gives very good accuracy of 98.5% on an average for detecting the normal jumps, 97.48% accuracy for on average for detecting High Knees and Jumping Jacks with 95.85% accuracy. The system is certainly quite rigorous since it is also able to detect the jumps in less than ideal situations as well. But It is to be noted that no system is perfect and this system is no different. The system seems to work poorly when they user is making sudden movements laterally or walking towards or away from the camera. The number of jumps are effected only for the number of frames in which these activities are taking place and the resumes to its normal well-functioning when the user stops erratic movements. Moreover the system faces difficulties in correctly differentiating a rope from any other moving object within the frame as it utilises the concept of contours which is in turn formed using the moving and changing pixels in each frame. Hence the system works best under static background conditions and works reasonably well for applications in which the background is not static .

7.3 Current limitations

The project was not without its limitations. Most of the time in the development of this system went into developing a technique for detecting the presence of a rope in the video. 3 different techniques were used out of which the current implementation gave the best result. The techniques are mentioned in the [Rope detection technique using Centre of Mass 8.1] section.

Since the rope detection technique uses contours, it inherently is not able to differentiate between an actual rope in the hand of the subject within the window or a moving leave in the background. The current technique for the detection of rope works extremely well in a controlled environment but will fail in backgrounds with a lot of non-static background motion. A more robust technique which can distinguish a rope from the other objects in the background will make this application better.

7.4 Further work

Naturally there are many things which can be improved about the system. A few of them will be discussed in the following passage

Due to the relatively poorer performance in a non-static background for the rope detector a more robust technique can be further developed for the detection of presence of a rope.

A new approach for counting the different kinds of jumps can be developed since High knees and Jumping Jacks both rely on angles made while moving the legs, it was observed that the count for one of the jumps was increasing even if they weren't being performed. A new technique to detect the occurrence of either of the jumps can increase the robustness of the system.

There are several many different kinds of jump rope techniques which can be programmed into the system. This would make the system more complete in terms of functionality it offers and would definitely be a very good value add.

The efficiency of the system could be improved to detect the number of jumps more accurately and closer to them happening. This can be done by implementing better data structures to store and manipulate different values and also by using more efficient computer vision techniques.

A mobile application which has all these features discussed above will be something that will be worked on, a robust mobile application with good emphasis on design can be a very helpful application for the jump rope enthusiasts. Moreover, having the system on a mobile application allows the system to reach a wider range of audience which would benefit from the varying level of difficulty of jumps which can be identified.

8 Appendix

8.1 Rope detection technique using Centre of Mass

An alternative technique was tried to determine the presence of a rope, well technically the motion of the wrists along with a rope. The contours for the hand were isolated and the centre of mass for the whole blob was observed. Movement of the y axis coordinate corresponding to the centre of mass was looked at and it was observed that the model was not robust enough to be used as a final solution since the movement was negligible and difficult to detect. However this technique can be utilised for videos of lower resolution and this technique would perform very well in it.

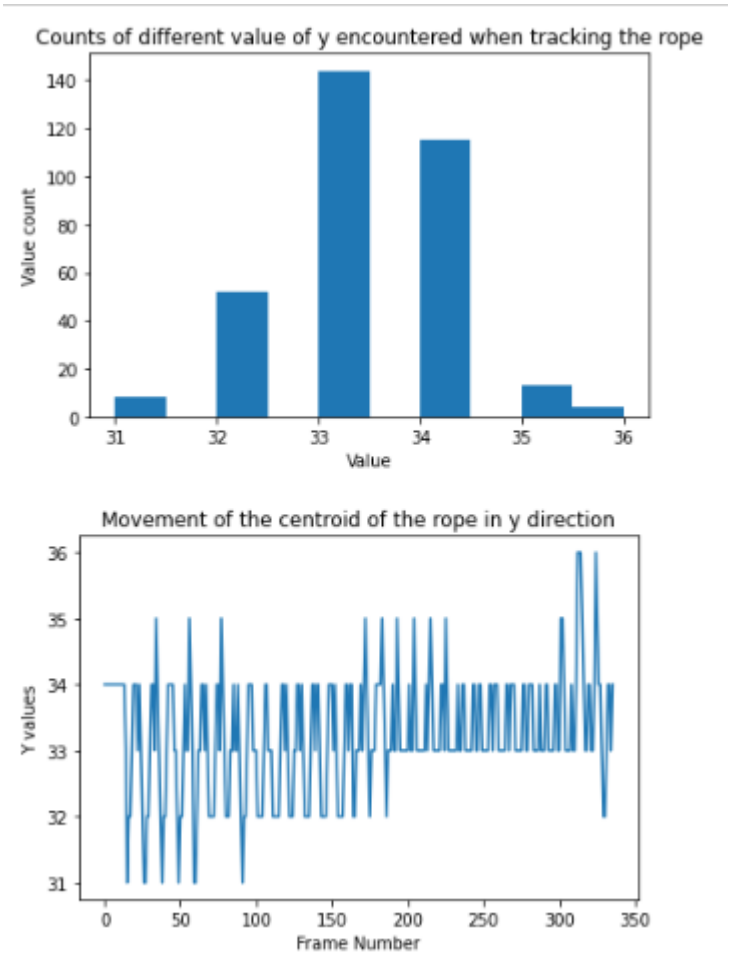

Figure 8-1: Centre of Mass for rope detection

9 Bibliography

[1] V. K. S. e. al, “The Effect of Skipping rope Exercise on Physical and Cardiovascular fitness among Collegiate Males,” 2019.

[2] T. S. J. Rope, “Tangram Factory,” [Online]. Available: https://tangramfactory.com/main/en/products.html. [Accessed 14 04 2023].

[3] Arrow, “What is IMU? Inertial Measurement Unit Working & Applications,” [Online]. Available: https://www.arrow.com/en/research-and-events/articles/imu-principles-and-applications. [Accessed 14 04 2023].

[4] M. B. R. &. J. M. Mahoney, “A comprehensive comparison of simple step counting techniques using wrist and ankle mounted accelerometer and gyroscope signals,” *Journal of Medical Engineering & Technology.*

[5] S. M. o. T. S. (. A. E. Majid Dadafshar, “Analog.com,” maxim integrated, 2014. [Online]. Available: https://www.analog.com/en/technical-articles/accelerometer-and-gyroscopes-sensors-operation-sensing-and-applications.html. [Accessed 14 4 2023].

[6] A. ojas, “Cleaner Pose estimation using openPose,” Medium, [Online]. Available: https://adityaojas.medium.com/cleaner-pose-estimation-using-open-pose-6d239cc33fe6. [Accessed 12 4 2023].

[7] Danimaaz, *Jump Rope Counter,* https://github.com/danimaaz/Jump-Rope-Counter, 2021.

[8] G. H. T. S. S.-E. W. Y. S. Zhe Cao, “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,” 2019.

[9] M. D. T. T. T. X. Samuel Zelman, “Accelerometer-Based Automated Counting of Ten,” 2020.

[10] A. C. Dirican, “Step Counting Using Smartphone Accelerometer and Fast Fourier Trransform,” 2017.

[11] M. White, “Determining jumping performance from a single body-worn accelerometer using machine learning,” Swansea, 2021.

[12] J. W. Xinxin Li, “Video-Based Rope Skipping Repetition Counting with ResNet Model,” 2021.

[13] Microsoft Research, “Deep Residual Learning for Image Recognition,” 2015.

[14] “Using Background Subtraction Methods in Image Processing,” *Analytics India Magazine,* pp. https://analyticsindiamag.com/using-background-subtraction-methods-in-image-processing/", 2021.

[15] C. K. e. al, “A hybrid framework combining background subtraction and deep neural networks for rapid person detection,” 2018.

[16] G. W. C. Christpher Kanan, “Color-to-Grayscale: Does the Method Matter in Image Recognition?,” 2012.

[17] Docs OpenCV, “Image Thresholding,” [Online]. Available: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html. [Accessed 14 4 2023].

[18] D. OpenCv, “Contour Features,” [Online]. Available: https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html. [Accessed 14 4 2023].

[19] LearnOpenCV, “Contour Detection using OpenCV,” [Online]. Available: https://learnopencv.com/contour-detection-using-opencv-python-c/. [Accessed 16 4 2023].

[20] J.-W. K. e. al, “Human Pose Estimation Using MediaPipe Pose and Optimization Method Based on a Humanoid Model,” 2023.

[21] Y. X. e. al, “ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation,” 2022.

[22] P. S. R. H. Tibor Trnovszkýa, “Comparison of background subtraction methods on near Infra-Red,” in *TRANSCOM 2017*, 2017.

[24] “IMU Principles and Applications,,” *Arrow,* 06 December 2021.