A Project Report on

AI Fitness Tracker

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

in partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

K. Sai Prajwal (19BD1A056R)

Abhiram Krishnam (19BD1A0562)

Vijay Cherukuri (19BD1A0567)

Under the guidance of

Ms. G. Naga Sai Sree Suma
Assistant Professor
Department of CSE

Department of Computer Science and Engineering

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

Approved by AICTE, Affiliated to JNTUH

3-5-1206, Narayanaguda, Hyderabad – 500029

2022-2023

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(Accredited by NBA & NAAC, Approved By A.I.C.T.E., Reg by Govt of Telangana State & Affiliated to JNTU, Hyderabad)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled Fitness Correction Using Python being submitted by

Sai Prajwal	**(19BD1A056R)**
Abhiram Krishnam	**(19BD1A0562)**
Vijay Cherukuri	**(19BD1A0567)**

In partial fulfilment for the award of **Bachelor of Technology** in **Computer Science and Engineering** affiliated to the **Jawaharlal Nehru Technological University, Hyderabad** during the year 2022-23**.**

Internal Guide **Head of the Department**

(Ms. G. Naga Sai Sree Suma) **(Mr. Para Upendar)**

Submitted for Viva Voce Examination held on ___________________________

External Examiner

Unit of Keshav Memorial Educational Society

#: 3-5-1026 Narayanaguda Hyderabad 500029.

040-3261407 www.kmit.in e-mail: principal@kmit.in

Vision of KMIT

Producing quality graduates trained in the latest technologies and related tools and striving to make India a world leader in software and hardware products and services. To achieve academic excellence by imparting in depth knowledge to the students, facilitating research activities and catering to the fast growing and ever-changing industrial demands and societal needs.

Mission of KMIT

- To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms and prepare students to become successful professionals.
- To establish industry institute Interaction to make students ready for the industry.
- To provide exposure to students on latest hardware and software tools.
- To promote research based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce a spirit of nationalism which will enable the student to develop, understand lndia's challenges and to encourage them to develop effective solutions.
- To support the faculty to accelerate their learning curve to deliver excellent service to students.

Vision & Mission of CSE

Vision of the CSE

To be among the region's premier teaching and research Computer Science and Engineering departments producing globally competent and socially responsible graduates in the most conducive academic environment.

Mission of the CSE

- To provide faculty with state-of-the-art facilities for continuous professional development and research, both in foundational aspects and of relevance to emerging computing trends.
- To impart skills that transform students to develop technical solutions for societal needs and inculcate entrepreneurial talents.
- To inculcate an ability in students to pursue the advancement of knowledge in various specializations of Computer Science and Engineering and make them industry-ready.
- To engage in collaborative research with academia and industry and generate adequate resources for research activities for seamless transfer of knowledge resulting in sponsored projects and consultancy.
- To cultivate responsibility through sharing of knowledge and innovative computing solutions that benefit the society-at-large.
- To collaborate with academia, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities.

PROGRAM OUTCOMES (POs)

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

3. Design/development of solutions: Design solutions for complex engineering problem and design system component or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create select, and, apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to societal, health, safety. legal und cultural issues and the consequent responsibilities relevant to professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: An ability to analyse the common business functions to design and develop appropriate Information Technology solutions for social upliftment.

PSO2: Shall have expertise on the evolving technologies like Python, Machine Learning, Deep Learning, Internet of Things (IOT), Data Science, Full stack development, Social Networks, Cyber Security, Big Data, Mobile Apps, CRM, ERP eetc..

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates will have successful careers in computer related engineering fields or will be able to successfully pursue advanced higher education degrees.

PEO2: Graduates will try and provide solutions to challenging problems in their profession by applying computer engineering principles.

PEO3: Graduates will engage in life-long learning and professional development by rapidly adapting changing work environment.

PEO4: Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism and ethical responsibility.

PROJECT OUTCOMES

P1: Ability to design Posture detection model which detects the posture of the user based on the body.

P2: Ability to use OpenCV and Image Processing techniques such as BlazePose and MediaPipe

P3: Ability to develop user-friendly UI for using the application

P4: Ability to evaluate and analyze body pose to maintain count constantly

LOW - 1
MEDIUM - 2
HIGH - 3

PROJECT OUTCOMES MAPPING PROGRAM OUTCOMES

PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
P1	2	2		3								
P2			3					2	1		1	3
P3				3			2					
P4								2	3	2	3	

PROJECT OUTCOMES MAPPING PROGRAM SPECIFIC OUTCOMES

PSO	PSO1	PSO2
P1		3
P2	2	1
P3		2
P4	1	

PROJECT OUTCOMES MAPPING PROGRAM EDUCATIONAL OBJECTIVES

PEO	PEO1	PEO2	PEO3	PEO4
P1	3	2		
P2	3	1		
P3		2		
P4			2	3

DECLARATION

We hereby declare that the project report entitled “AI FITNESS TRACKER” is done in the partial fulfillment for the award of the Degree in Bachelor of Technology in Computer Science and Engineering affiliated to Jawaharlal Nehru Technological University, Hyderabad. This project has not been submitted anywhere else.

Sai Prajwal (19BD1A056R)

Abhiram Krishanm(19BD1A0562)

Vijay Cherukuri(19BD1A0567)

ACKNOWLEDGMENT

We take this opportunity to thank all the people who have rendered their full support to our project work.

We render our thanks to **Dr. Maheshwar Dutta**, B.E., M Tech., Ph.D., Principal who encouraged us to do the Project.

We are grateful to **Mr. Neil Gogte**, Director for facilitating all the amenities required for carrying out this project.

We express our sincere gratitude to **Mr. S. Nitin**, Director and **Dr. D. Jaya Prakash**, Dean Academics for providing an excellent environment in the college.

We are also thankful to **Mr. Para Upendra**, Head of the Department for providing us with both time and amenities to make this project a success within the given schedule.

We are also thankful to our guide **Ms. G. Naga Sai Sree Suma**, for her valuable guidance and encouragement given to us throughout the project work.

We would like to thank the entire CSE Department faculty, who helped us directly and indirectly in the completion of the project. We sincerely thank our friends and family for their constant motivation during the project work.

Sai Prajwal (19BD1A056R)

Abhiram Krishnam(19BD1A0562)

Vijay Cherukuri (19BD1A0567)

CONTENT

ABSTRACT

Nowadays, because of busy schedules, people have no time to go to the gym, and even if they manage to find a gym nearby having a gym trainer besides all the time to correct postures while doing exercise is impossible unless people do not opt for a personal trainer. Even if they assist a personal trainer for them they would have to adjust their time accordingly and both of these methods are quite costly and not everyone can afford it, also during this global pandemic, people are stuck at home and have no access to go to the gym or can’t even take a risk of getting in contact with a personal trainer. Performing exercises let it be exercise or doing yoga proper body posture is important, if not performed properly it can lead to crucial problems such as poor joint alignment, increased shear forces on the spine, compression of discs and joints, less space for nerves to course through the body due to compression, reduced blood flow, etc. to prevent such injuries and pains, and to track gym exercises repetitions we came up with a system using Python and OpenCV. It is an AI Fitness tracker. It tracks users’ body movements using human pose estimation. This in turn keeps track of repetitions of gym exercises and can also be used for detection of wrong body posture while doing yoga.

LIST OF FIGURES

CHAPTER - 1

1.INTRODUCTION

1.1 Background

The aim of the proposed research work is to develop and implement the AI model which can do analysis and keep count of human physical exercise postures using Deep Learning and Computer Vision in actual scenarios. The research is proposed for the selected exercises like curls, squat, sit-ups, lifts, raises and push-ups. The physical exercises are the part of human rehabilitation process which can be performed at all possible situations for relaxation.

With reference to the World Health Organization, one third of the world population are not doing proper exercise practices which leads to the physical instability and impacts huge on the fitness of the human being World Health Organization (2020). To support this statement the pandemic has worsened the exercise situations. People are really in need of an application which can support and motivate them to do exercises at their own without external factors like Gym, Trainers, and so on. The need of personal trainer application arises with the solid goal to train themselves in the existing situations. This Application will interact instantly according to the postures performed by the user in all the possible exercise environments. The merits of Deep Learning and Computer Vision will help to solve the exercise identification and analysis for this application of research for the mentioned exercises.

There are certain technology limitations like full body tracking and real time node image processing. Recently, Google's MediaPipe framework can track the full body node on the human body from a computer or phone camera in actual scenarios. MediaPipe is a commonly available open-source framework released by Google in August 2019 which is widely used for AI techniques like object tracking, face recognition and multi-node tracking on human body, and so on.

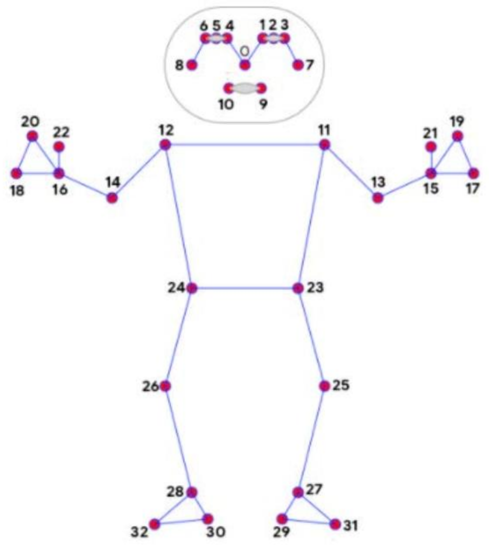


Figure 1.1: Blaze Pose Model Topology

By applying the merits of the MediaPipe, the Blazepose model as shown in Figure 1.1, is used to track the body nodes and postures from a normal camera connected to the processor and detect gestures with a Convolutional Neural Network (CNN) in real time. The node detection is depending on the Blazepose model, and this research work is focused on developing the AI model to do the exercise posture analyzation and Blazepose model is a successive tool for identification of human body nodes. The uniqueness of the proposed work is simplicity of the system which uses one camera from a processor and then a developed computer application is capable of all the analytical work for the human exercises.

The proposed research work complies with time and other limitations which does not intend to be done analyzation for the all the exercises done at the exercise environments. This work identifies and analyzes fourteen different exercises and provides good hypothesis that similar practice can be performed to analyze other exercises in a larger scope.

1.2 Problem Statement

There are many individuals conducting physical exercise without the knowledge of the right postures. This proposed work helps person to identify and analyze their physical exercise posture using simple and interactive system which will be user-friendly. The analyzation will instantly show the accuracy of the exercise postures and suitable counting for the exercises performed using the dashboard. This project aims to solve the identification problems of people who are performing physical exercise. The identification and analysis of the physical exercise done by humans are performed by using deep learning models. The deep neural networks are trained to work as like the trainers in the gym to help the human beings to perform physical exercise effectively. 1.3 Objectives

The objectives of the research are:

1. To create an Intelligent system using deep learning model to identify the human physical posture.
2. To develop a computer vision model to analyze the correct form of physical exercise.
3. To propose an AI system which can assist the users to perform the physical exercise in the correct way by keeping count of their physical output.

1.3 Scope of Study

Even though the proposed work is designed for the exercise posture analysis, there are some important limitations which are not took up as research constraints. For instance, it is not possible to analyse all the exercise postures without user's selection. In this project user will select the listed exercises before performing it. It's a complicated work, if exercise is done without prior selection because of mixing of the same gesture actions of the exercises. The different combinations of nodes from BlazePose model are used to analyse the exercises. The selection of the node is important feature of the analyzation. The implementation of the concept is done by the Python tool and TensorFlow framework based on Graphical Processing Unit (GPU).

1.4 Summary

The thesis is organized as follows. Chapter one illustrates the background of this project, as well as the problem statement, project objectives and scope of work. Chapter two gives an overview of and a detailed literature review of object detection and Exercise Posture detection. Chapter three indicates the research methods to conduct this task. Chapter four presents the results of the proposed method and result discussion. Chapter five concludes the whole thesis and points out possible future works for this project.

CHAPTER - 2

2. SOFTWARE REQUIREMENTS SPECIFICATIONS

2.1 What is SRS?

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.). The SRS phase consists of two basic activities:

Problem/Requirement Analysis: The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

Requirement Specification: Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity. The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

2.2 Role of SRS

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium though which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

2.3 Requirements Specification Document

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application. There are a set of guidelines to be followed while preparing the software requirement specification document. This includes purpose, scope, functional and non-functional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behaviour of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS. This section introduces the requirement specification document for Fitness Posture Correction using OpenCV which enlists functional as well as non-functional requirements.

2.4 Software and Hardware

The implement of the BlazePose model requires CPU and GPU hardware. The CPU and GPU are the commonly used hardware modules used for deployment of deep learning model and computer vision techniques. GPUs are high speed computing embedded hardware used for processing tensors. GPUs executes blocks faster than the CPU. Besides the hardware, software is required to deploy the proposed models. The software and IDE are necessary thing for any tool to develop models. The software like Python and its IDEs will allow efficient open-source packages which are used for implementing deep learning applications. The Python version of 3.7.2 and Pycharm Community Edition 2020.2.3 IDE is used for implementing the proposed exercise analyzation work. The important packages used are

TensorFlow, Keras, MediaPipe, OpenCV, Numpy, Math functions, Tkinter, time, Image TK, Img.

2.5 Summary

This chapter discussed different approaches of computer vision and other deep learning techniques for identification and analyzation of object tracking, pose estimation and sign detection. Subsequently, this chapter also explains overview of the BlazePose model and its architecture’s outline for better identification of human pose. Then, this chapter also gives the idea about model implementation using available hardware and software with current technological points.

CHAPTER – 3

3. LITERATURE SURVEY

3.1 Introduction

This chapter will discuss the earlier research works, before this proposed work, conducted by various authors who all tried to recognize various objects and pose modules in the human body to find postures with the unique approaches. The features of this proposed work will represent the comparison works which are related to Google's MediaPipe Framework, and establishments of all other possible approaches to recognize exercise postures and the uses of the different systems in people's everyday life.

3.2 Computer Vision approaches for Object Detection

Computer Vision approach is the one of the most used technologies that presents gave the best results for the tasks like object tracking, contour detection, edge detection, image processing, image segmentation, machine vision counter actions and so on. The OpenCV is an early relied open-source library technology developed by the Intel, and it focuses fully on the real time compute vision related projects.

There are many research works done prior to the node analyzation of the human body parts. The study proposed by (Silver & Xin, 1997) has tracked dynamic turbulent 3D features random datasets. Figure shows colour amalgamation of the different object in simulation set. The important feature of the algorithm is the Boolean set characteristics to make computer vision system for acquiring 3D visualization. This work shows solid proof dynamic tracking of the objects in the different frames acquired by the camera.

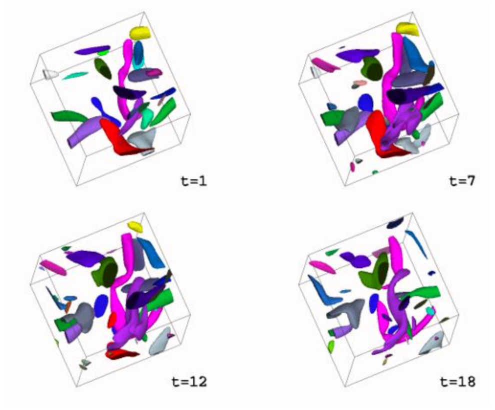


Figure 3.1: Simulation of coherent Structures

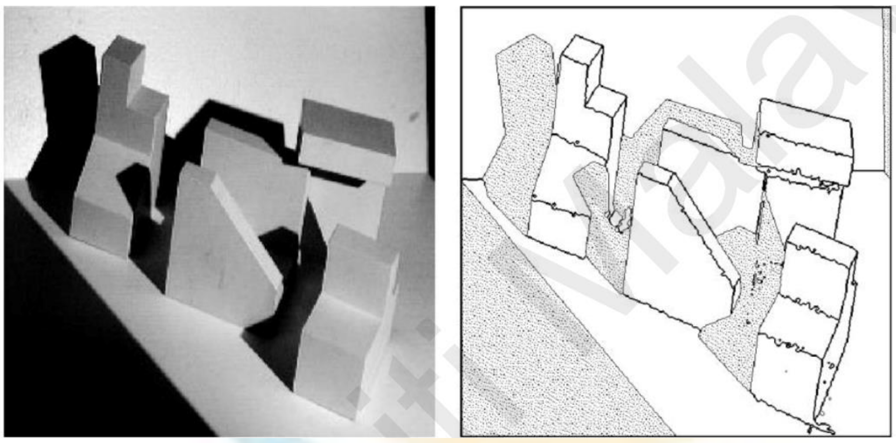


Figure 3.2: Segmentation of the testing image

Another research work proposed by (Ohya, Kosaka, & Kak, 1998) has its own navigation robot which uses single camera for computer vision and ultra-sonic sensors to support the system to obtain the objectives. The model developed by (Ohya et al., 1998) non-stop navigation position correction system to monitor the obstacle before the prototype robot. This concept clearly shows computer vision calculations like edge detection, positioning, 3D model can be used for self-assuring autonomous systems. Similarly, the work done by (Frigui & Krishnapuram, 1999) has a new approach called Robust Competitive Agglomeration (RCA) to cluster the different object patterns. This approach has many merits in robust classification feature to identify the objects. Figure 3.2 shows the image segmentation of the certain objects. The feature classification is based on the correct detection, over segmentation, under segmentation, and noise cross.

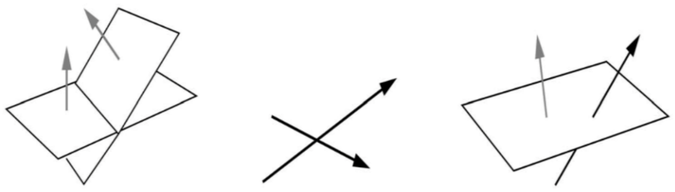


Figure 3.3: Angel relations in model

Another research work performed by (Xufei, Kanungo, & Haralick, 2005) Error propagation statistical method for the identifying the positions of the building edges. This methodology acquits angle of the spatial models from the image captured by the camera. The patterns in the algorithm dynamically recognise the 3D parameters in the image which built by the error propagation method. Figure 3.3 shows the spatial angle calculation of the model. The work accomplished by the (Pun, Alecu, Chanel, Kronegg, & Voloshynovskiy, 2006) have Brain Computer Interaction (BCI) system which connects the Electro Encephalogram (EEG) through the system to develop computer vision model. The model uses EEG signals which direct proof of physiological signs which cannot be faked. The important aspect of the proposed work by (Pun et al., 2006) is physical signs were optimized for developing human interactive computer vision system. Similarly, the results of the research work by (Lee & Park, 2009) for developing Remote Control System (RCS) model uses human hand moves for establishing control system. This system detects finger counting based on the contour model designed in the flow process. This helps the user to control the systems with simple hand gestures before sitting in front of the camera module. Figure 3.4 shows the iterative results of the finger counting used for RCS.

The frame less event-based system for fast vision technique developed by (Perez- Carrasco et al., 2010) has unique of identifying features from the video frames. The conventional method using convolution method to extract features for texture recognition was boycotted and special sensory system was designed for finding classification, implementation of the event-based system reduces computational process for 25-30 frames of video images, and event based sustains to enhancing features for the classification as computational efficient computer vision system. Likewise, the work achieved by the Google AI blog (2020) used non-linear diffusion analysis to filter the micro nucleus organisms. The non-linear filtering allows the algorithm to perform seed evaluation to segment the organisms and this technique further intensifies the clustering data. Figure 3.5 shows the filtered image of the organisms.

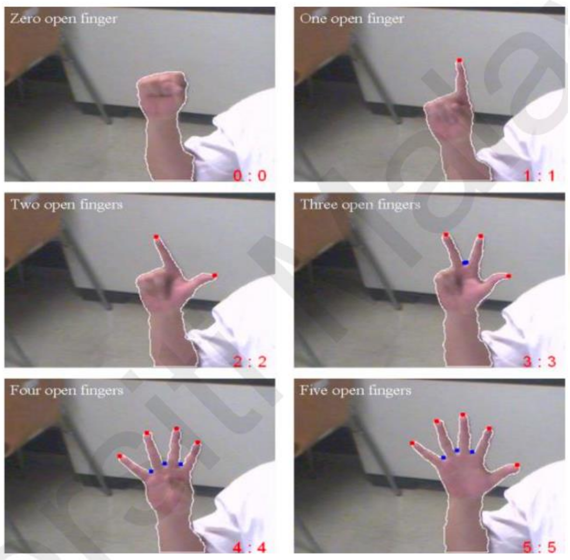


Figure 3.4: Example of finger counting

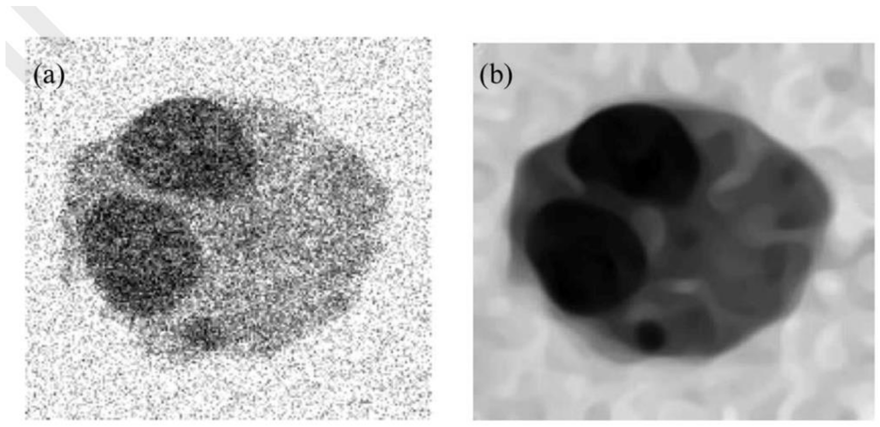


Figure 3.5: Non-linear filter image of micronucleus

3.3 Approaches for Exercise Detection

The human body metabolism is enhanced by performing the process of physical exercise postures and relaxes the body for rehabilitation (Bakar, Samad, Pebrianti, Mustafa, & Abdullah, 2015). The physical exercises can be performed based on the flexibility of the human beings. The flexibility attained during the performance of the physical exercises will improve the muscle functionalities. Evidently, the exercises are performed based on the gait movements, depends on the level of difficulty, and every exercise has dissimilar body nodes to analyze based on the gait action (Kavian & Nadian-Ghomsheh, 2020). The gait action based on the movements of the exercise can be evaluated by selecting correct nodal action of range for each exercise. The selected body nodes for the certain exercise will have clear unique line of records (Kavian & Nadian-Ghomsheh, 2020). The identification of the exercises performed by humans are identified by the Gait activity of the exercise (Z. Zhang, Wang, & Cui, 2018). The gait activity of the exercise will have the specific curve-fit characteristics naturally. The possibility of the gait activity recognition rate for the nodal analysis is attained by the linear analysis of the image (Tao, Li, Wu, & Maybank, 2007). The Figure 3.6 shows the averaged gait image of the same person after the linear analysis of the image from the database.

Figure 3.6: Averaged gait images

(Tao et al., 2007) performed the possibilities of Tensor Discriminant Analysis for classifying the gait of the human by processing sequence of images. Another research work done by (Yang, Li, Zeng, & Wang, 2021) have the two branches of CNN for the node recognition to perform pose estimation for developing open pose detection method. Although, the methodology of nodal post estimation by the open pose detection method gives the fine results, they failed to analyze the exercises in dynamic way. Figure 3.7 shows the enhanced view of the pose estimation without analyzation and human-computer interaction. The existing systems, for deeply analyzing the posture of the exercises are done with the use of certain externally fitted system like Radio Frequency Identification (RFID) modules, Wi-fi signal routers, Image sensor, accelerometer, gyroscope and so on (Z. Liu, Liu, & Li, 2020) (Ermes, PÄrkkÄ, MÄntyjÄrvi, & Korhonen, 2008) (Mekruksavanich & Jitpattanakul, 2020; Nagarkoti, Teotia, Mahale, & Das, 2019) (W. Zhang, Su, & He, 2020). (Z. Liu et al., 2020) performed the monitoring of the Human physical exercises with the heavy act kinetic camera, RFID tag system, transceiver, reader/writer, and processor which is the complicated system than that of human pose estimation system (Yang et al., 2021). The Figure 3.8 gives the design of the complicated system proposed by (Z. Liu et al., 2020).

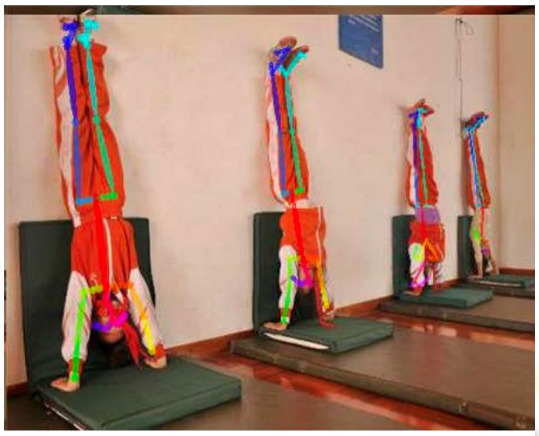


Figure 3.7: Enhanced view of Open Pose

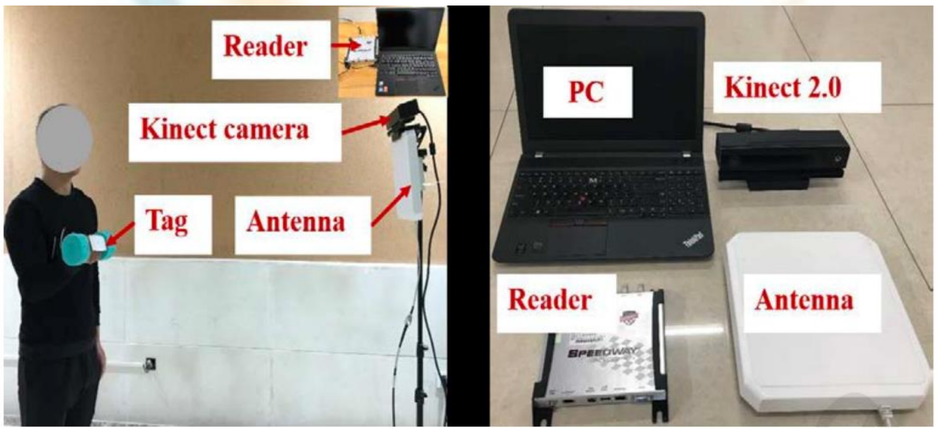
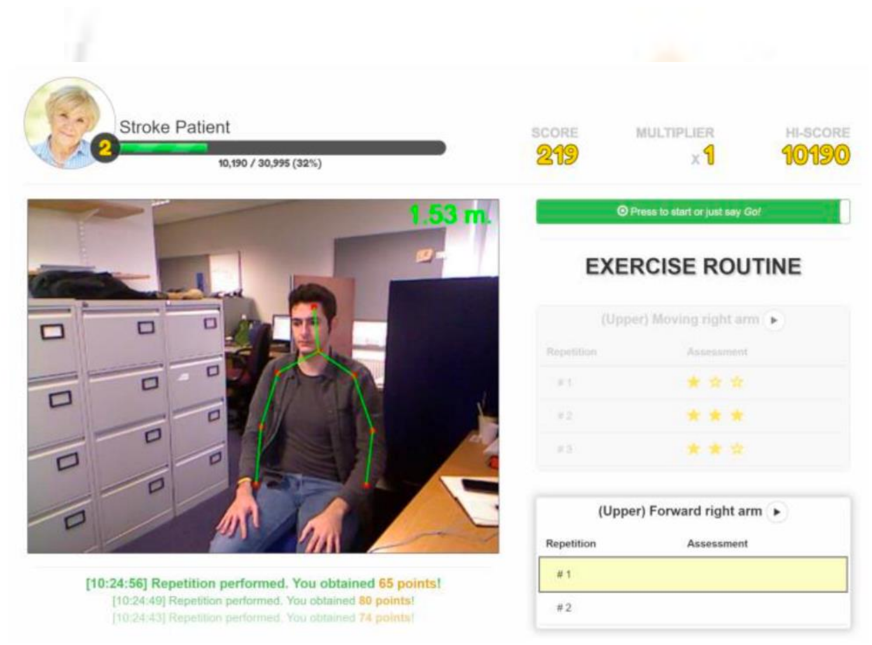


Figure 3.8: Complicated experiment layout

The computerized recognition of human physical exercises analyzation is possibly done by the help of computer vision (Ar & Akgul, 2014). The results achieved by both (Z. Liu et al., 2020) and (Ar & Akgul, 2014) are optimized to get good results and exercise analyzation was attained with complex designs. The advancement of the open-source computer vision gives better results for effectively analyzing the exercises using Remote rehabilitation process (Schez-Sobrino, Vallejo, Monekosso, Glez-Morcillo, & Remagnino, 2020). The developed the rehabilitation process as gamified interactive system (Schez-Sobrino et al., 2020). Figure 3.9 shows the dynamic exercise analyzation system for the stroke patients.



(Upper) Moving right arm

Repetition	Assessment
# 1	★ ☆ ☆
# 2	★ ★ ★
# 3	★ ★ ☆

(Upper) Forward right arm

Repetition	Assessment
# 1	
# 2	

[10:24:56] Repetition performed. You obtained 65 points!
[10:24:49] Repetition performed. You obtained 80 points!
[10:24:43] Repetition performed. You obtained 74 points!

Figure 3.9: Gamified system for rehabilitation process

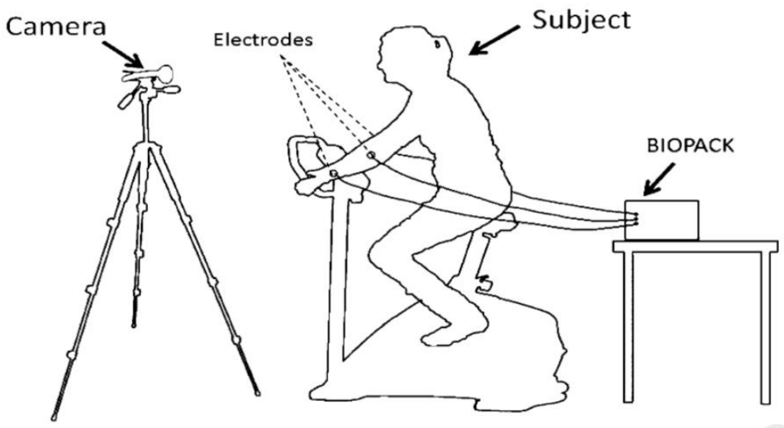


Figure 3.10: Non-invasive experimental setup

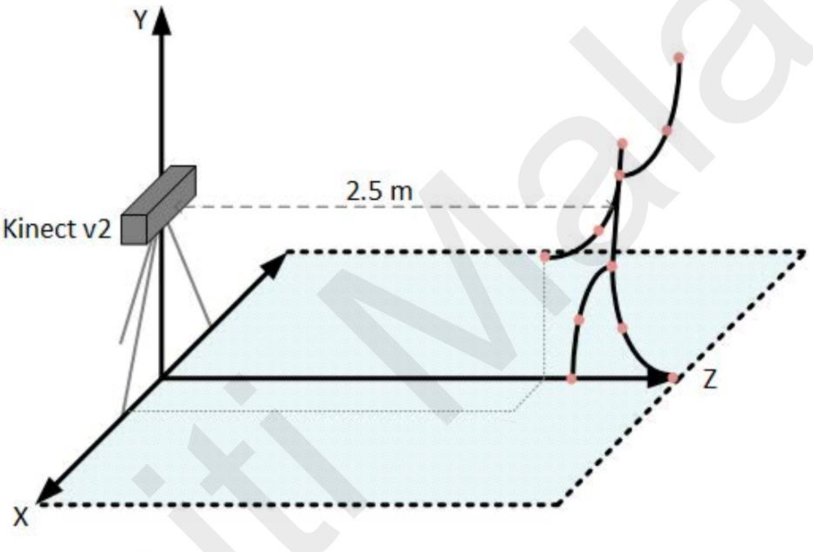


Figure 3.11: Enhanced view of Open Pose

The system can produce optimal results when it uses DTW algorithm to get the exercise pattern from the patients (Schez-Sobrino et al., 2020) (Schez-Sobrino, Monekosso, Remagnino, Vallejo, & Glez-Morcillo, 2019). The assessment done by (Schez-Sobrino et al., 2020) promotes remote rehabilitation with 3-D transformation (Posture and rotation). (Schez-Sobrino et al., 2019) engraved exercise rehabilitation for stroke survivors without the interactive counting performance feature. Remote and contactless recordings of all physical human signals will directly enroot health application (Monkaresi, Calvo, & Yan, 2014). (Monkaresi et al., 2014) proposed a contactless heart rate monitoring system using Webcam. Figure 2.14 shows the experimental view of the study for implementing machine learning approach of contact less heart rate monitoring. The image acquisition, processing and feature extraction using a webcam simplifies the complex system design (Yang et al., 2021) (Monkaresi et al., 2014). The motion capture recognition for human exercises is possible with the skeletal data of the motion (Vox & Wallhoff, 2017). The (Vox & Wallhoff, 2017) done a classification of few exercises using SVM architecture, but model normalized the output with mixed pattern of exercise recognition ended up with loss of accuracy. Figure 2.15 shows the skeletal data acquisition through the kinetic camera. The evaluation for few knee exercises can be spontaneously acquired to attain the performance using histogram analysis (Pattamaset, Charoenpong, & Charoensiriwath, 2020). (Pattamaset et al., 2020) performed axis segmentation pattern to classify the skeletonization of the typical exercise. Figure 3.11 shows the histogram analysis for knee movement of the exercise. The disadvantage of automatic system proposed by (Pattamaset et al., 2020) ends up with reduced overall accuracy due to complicated axis extension calculations. The tracking of physical counteractions can also be done by routing Wi-Fi module system (N. Liu, 2017). (N. Liu, 2017) proposed a Wi-gym framework which requires coverable transmitting capabilities at exercise environment to pre-train the accuracy and complexity of system of will increase computational loss for many exercise classes. Similarly, CNNs for exercise recognition and classification enforces strong pattern label as input for the calculations (Bakchy et al., 2018; Huang et al., 2020; Rungsawasdisap et al., 2019).

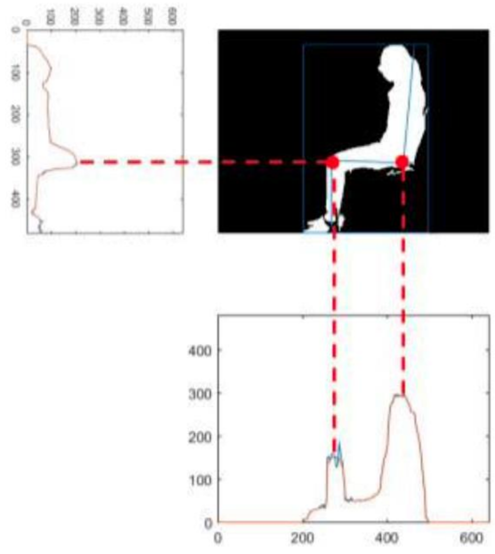


Figure 3.12: Histogram analysis of binary image

3.4 Pose Estimation Methods

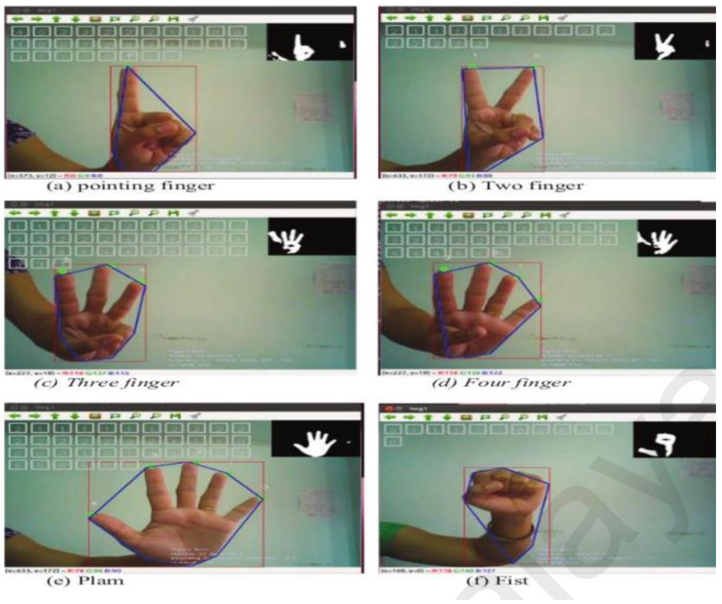


Figure 3.13: Contour detection of Open Hand

The contour detection is key feature for the most computer vision works approaches as discussed in earlier works. The detection of contours possibly gives good results in classifying, tracking, and identifying the objects. Although, the contour detection has been nice approach, further robust technique is required which focuses on real time computer vision applications.

The contour detection generates unusual identification of problems. First, it is challenging behind all combination of postures to identify the exercise postures of the humans. Second, if we have forms of the different postures, we cannot detect the posture at the correct blob, at the actual analyzation position. In addition to that, the processing acquires more images as many frames acquired by the camera.

The model proposed by the (Gurav et.al,2015) clearly shows that the hand postures for steady signs detection is done, which cannot be used for the clear posture attainments. Figure 3.12 clearly shows the steady signs of hand and its contrast image. Even though, the camera acquires the image at the speed of 30fps, the concept of contour detection syncs and gives better output on the contrasted background and the object which is tracked for the contours need to be close to the camera.

In this research, computer vision and deep learning methods are used for identification and analyzation of physical exercises. The proposed system follows certain line of action (nodes) by BlazePose model Google AI Blog (2020) of the human body parts to analyze the exercise. The selected device is merely a normal webcam to capture video input to take depth data of exercise.

CHAPTER -4

4. METHODOLOGY

4.1 Input

The input for the system is video frame images. The RGB image input is used by the deep learning model to track the nodes. The users are needed to do exercise in the correct form in front of the video capturing object (camera or webcam). The camera should be placed in a good lighting condition. The system can work typically on the exercise environments. The system cannot predict output except when the ROI of human pose is not covered in the view of the computer vision machine. The system gives better analyzation results when frame rate of the camera is minimum 30 fps.

4.2 Feature Extraction

The exercises which are selected for analysis will have mixture of postures which cannot be predicted or analysed without selection particular exercise before performing it. The exercise should be selected by the user from the dashboard of the system. The system can identify and analyse the exercises without errors. The user should know the pre-requisite of performing knowledge of the exercise.

4.3 Pose Estimation

The pose estimation is performed with the help of the MediaPipe framework, a deep learning model. The BlazePose model from MediaPipe will identify the nodes of human pose. The framework of the model supports to display 33 points as its features. The proposed will recognize all the 33 points. The customized node selection was done for each exercise. These nodes are used to estimate the exercise postures for analyzation.

4.4 Overview of Blazepose Model

Pose estimation of human body from a certain action of gestures plays vital role in identifying the physical nodes and enables quick learning process in the applications of augmented reality, human body gesture control method, sign language recognition and for quantifying the human physical exercise postures. The BlazePose is a Deep Learning model which can detect the node topology in difference appearances, environments, degree of flexibility, mixtures of postures and so on.

4.4.1 Topology of Blazepose Model

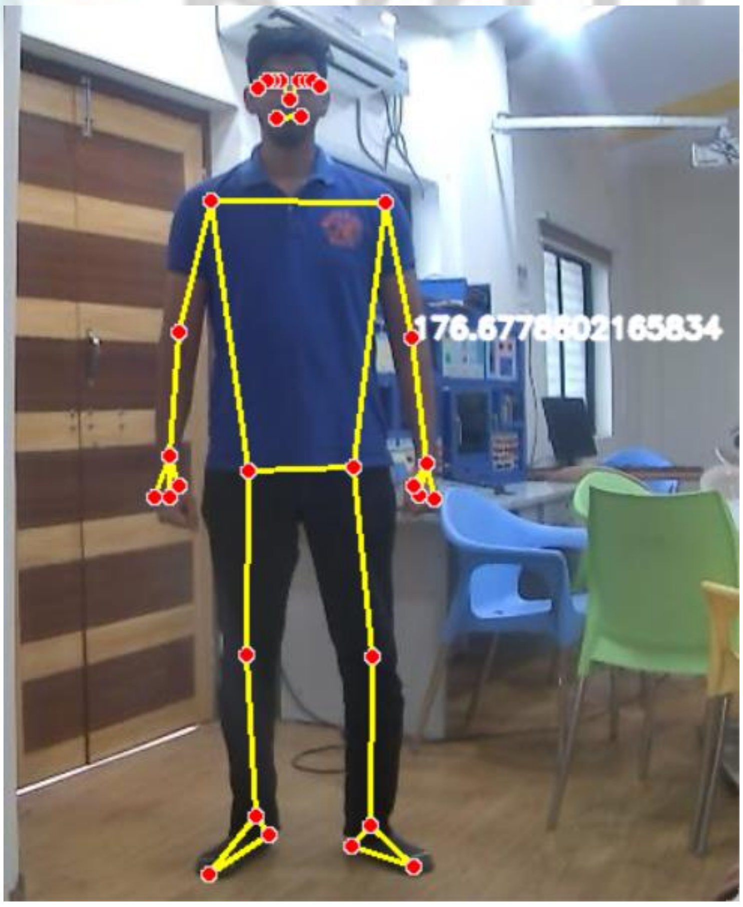


Figure 4.1: Topology of BlazePose model implemented in human pose

The topology of 33 points from the human body are taken as nodes for reference in this work. These nodes allow to predict the poses alone which gives good results across human model in accordance with number of frames per second. Figure 3.13 shows the implementation of topology of the BlazePose model in human body.

4.4.2 Implementation of Tracking Model

Figure 3.14 explains the pipeline of the tracking model. The images acquired by the camera module will be analyzed by the pose detector calculation, based on the calculation up on the image the pose alignment is shifted as output. The pose tracking is performed and aligned for every image frames. The predictions of the pose detector are done based on the heatmap function combined with regression algorithm. The entropy loss of the function is expressed as:

$l = {}^{1} \sum^{N} \sum^{W} \sum^{H} [p^{n} \log p^{\hat{n}} + (1 - p^{n}) \log(1 - p^{\hat{n}})]$ .... (1)

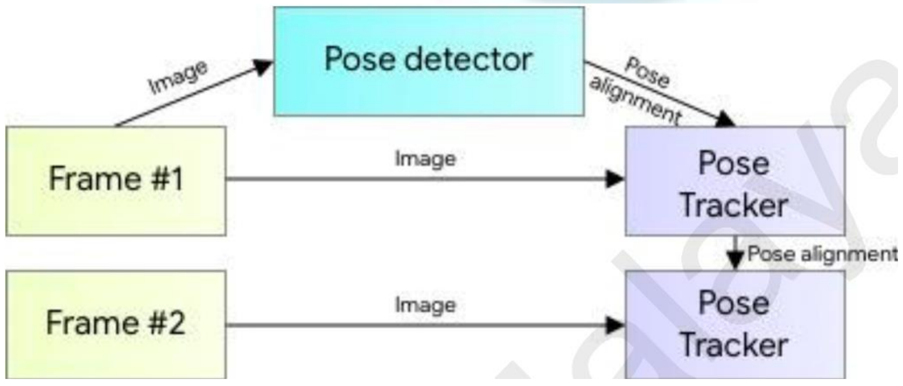


Figure 4.2: Pipeline of tracking model

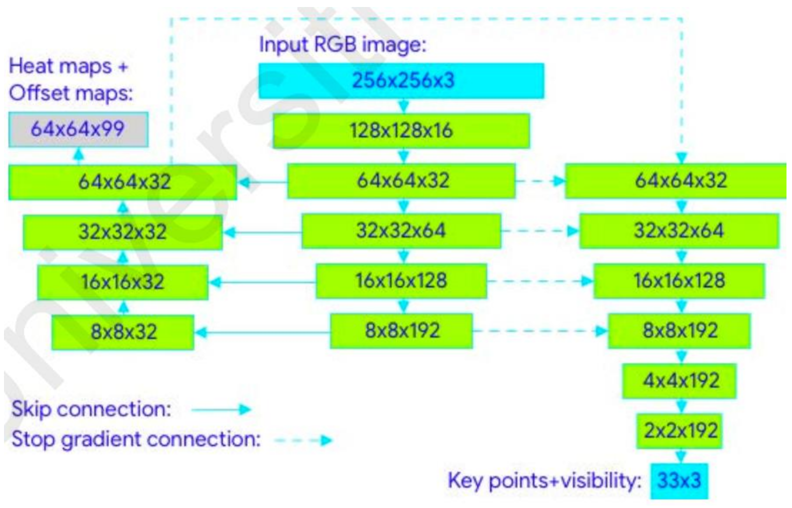


Figure 4.3: Architecture of System

Figure 2.20 explains the architecture of the tracking model with the heat map, offset function maps establishments, along the convolution process of the input RGB image from the camera, and the process establishment the regression analysis through the image. The regression output of the nodes at all locations of the human body is expressed as following loss:

N

$l = {}^{1}\sum \sum \|Mn\,(i, j) - Mn(i, j)\|^{2}$.. (2)

N

Based on the human pose, the implementation of BlazePose model produces better results for estimating the nodes. These nodes are further used for the identification and analyzation of the exercise postures for this proposed research work.

4.5 Analysis

Pose estimation is only performed with the selected nodes from the customized BlazePose model. The minimum of two or three nodes are packed together for each exercise. The nodes play major role for the exercise analyzation process. The key points are the coordinates of each node in every frame of the video image frames. The coordinates will change dynamically for every frame according to the action of the human pose. These coordinates are used to calculate the two important features angle and distance.

4.6 Accuracy and Counting

The selection of key points and other condition based on the exercise we are estimating the accuracy. Based on the accuracy the counting is performed. The counting technique is not same for all the exercises.

4.6.1 Angle Calculation between Nodes

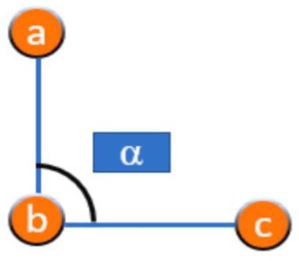


Figure 4.4: View of angle calculation between nodes

To calculate the angle, three nodes are bundled together. The combination of nodes for angle calculation is picked up based on the exercise pattern and some exercise will have more than one angle. The coordinate points (x_a, y_a) (x_b, y_b) (x_c, y_c) are used to find the α

in Figure 4.1. The equation for calculating the angle (α) is expressed as:
$\alpha = (\mathrm{atan}(y_c - y_b), (x_c - x_b) - \mathrm{atan}(y_a - y_b), (x_a - x_b))$................(3)

The negative angle indication is eliminated due to the error of negativity. The error of negativity is eliminated by performing ($\alpha + 360^\circ$) calculation, this will help the system to analyse the right form of the exercise.

To calculate the distance, two nodes are bundled together. The combination of nodes for distance calculation is selected based on the exercise pattern and some exercise will have more than one distance. The coordinate points (x_a, y_a) (x_b, y_b) are used to find the β in Figure 3.3. The equation for calculating the distance (β) is expressed as:

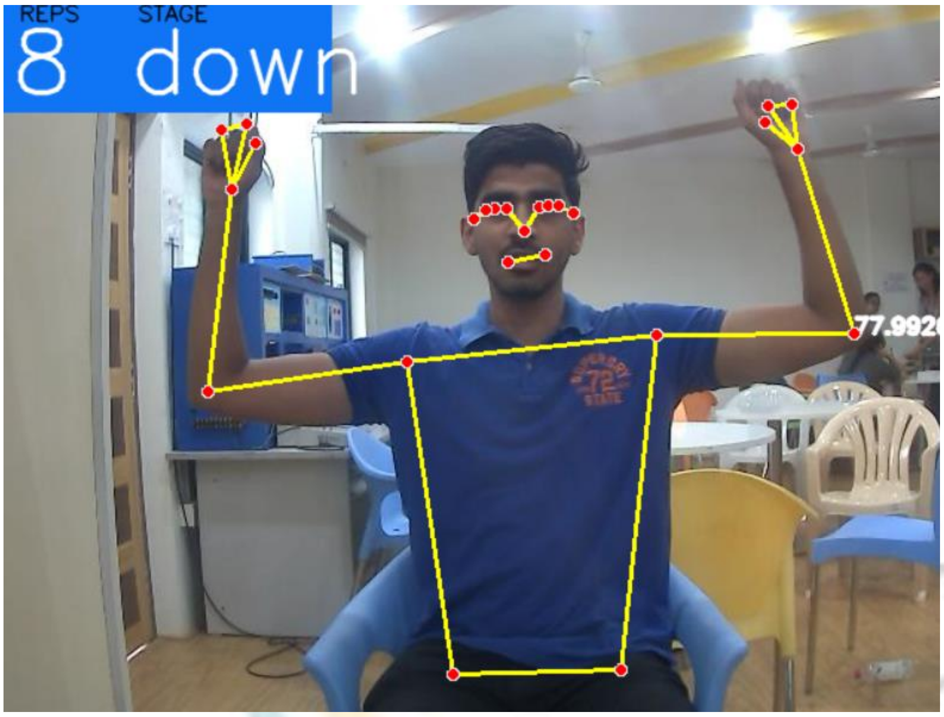
$\beta = \sqrt{} (y_a - y_b)^2 + (x_a - x_b)^2$...(4)

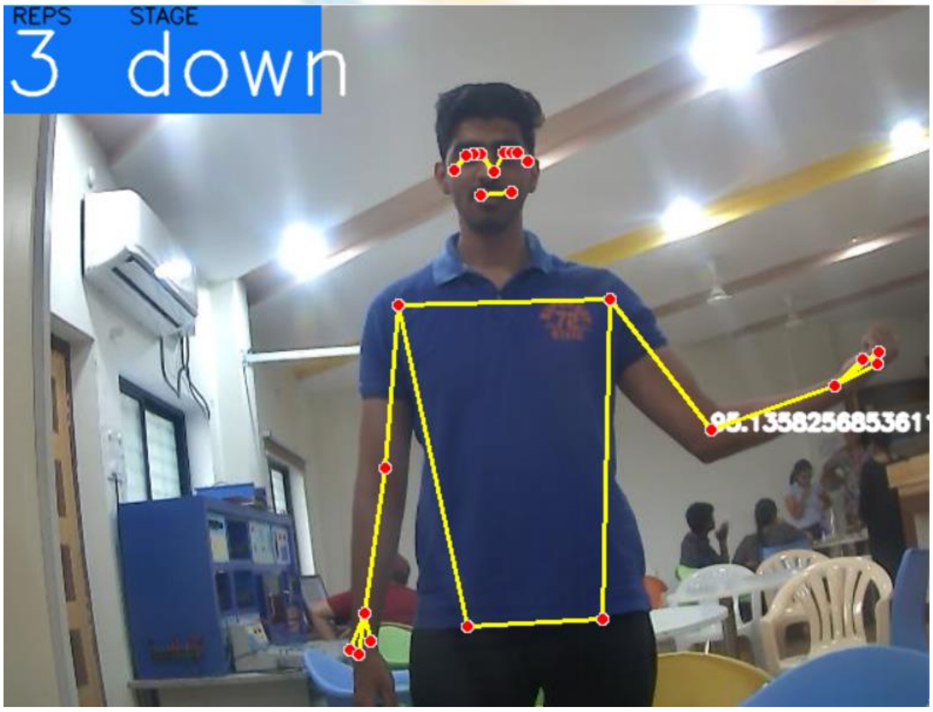
Now, the α and β are calibrated to α_1 and β_1 to find error range of the exercise by a threshold Ʈ. Ʈ will be defined based on instances of the exercises which is defined for estimating the accuracy of the performed exercise. The critical condition for each exercise is expressed as:

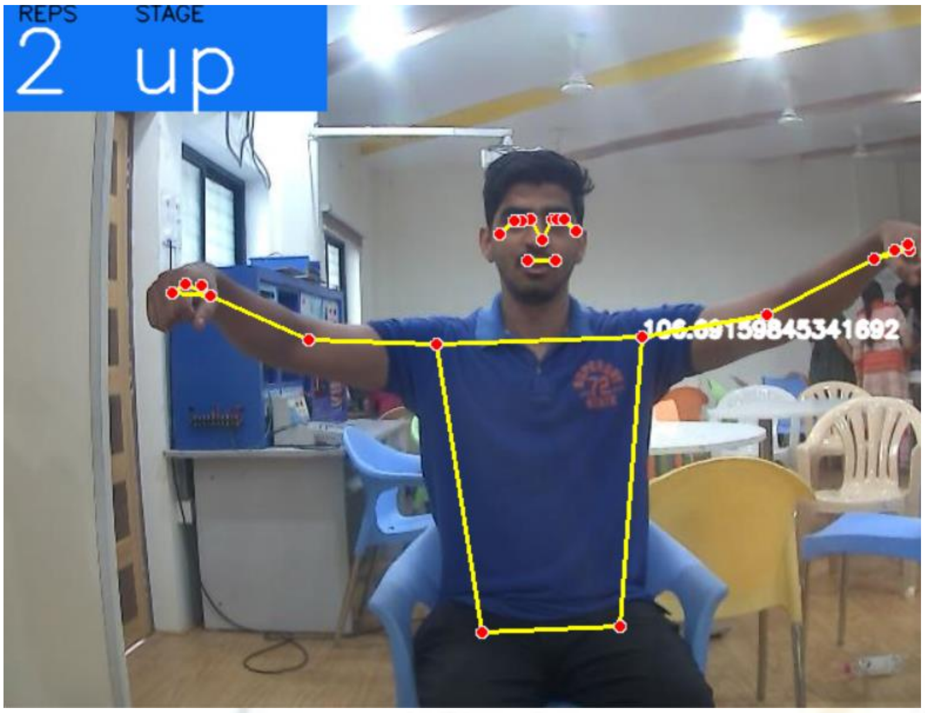
$1 \geq \alpha_1$ (or) $\beta_1 \geq$ Ʈ...(5)
The posture experimentation used to define Ʈ for each exercise. The accuracy and counting will be calculated based on the Ʈ.

4.7 Node Selection for Exercises

The node selection for exercises is based on the angle and distance calculation between nodes. The nodes are selected comparing to correct posture of the exercise. The selection of nodes for each exercise after performing many trial and error postures of the exercise postures in the experimentation process. The following figures shows the view of the key point selection for every exercise.







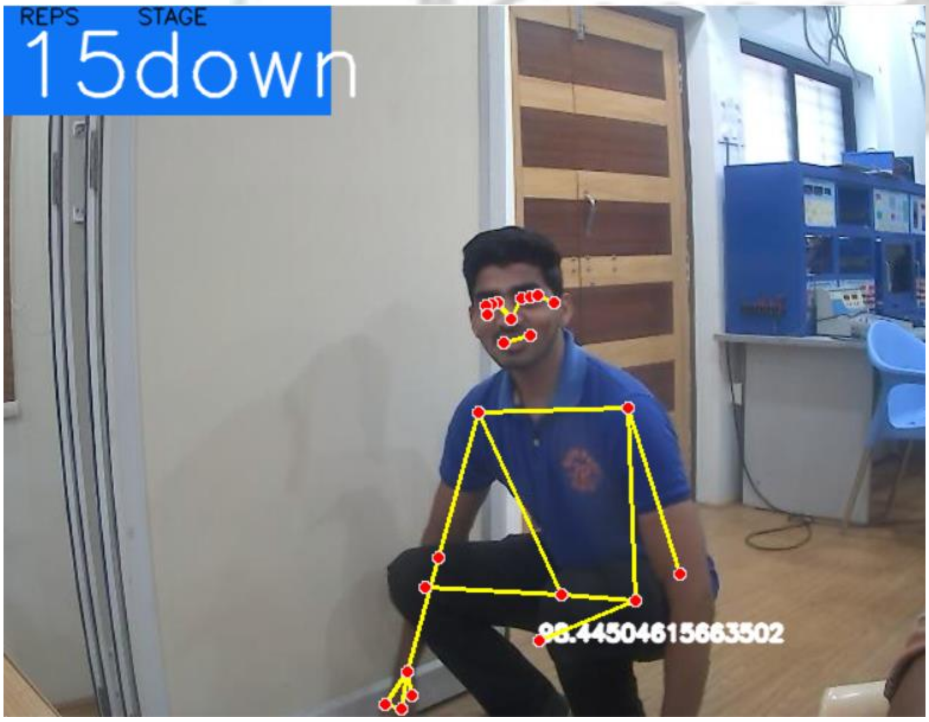


Fig – 4.5 Selection Nodes

CHAPTER - 5

5. SYSTEM DESIGN

5.1 Introduction to UML

The Unified Model Language allows the software engineer to express an analysis model using the model notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective.

Each view is defined by a set of diagrams, which is as follows:

1.User Model View

This view represents the system from the users' perspective. The analysis representation describes a usage scenario from the end-users' perspective.

2.Structural Model View

In this model, the data and functionality are arrived from inside the system. This model view models the static structures.

3.Behavioural Model View

It represents the dynamic of behavioural as parts of the system, depicting he interactions of collection between various structural elements described in the user model and structural model view.

4.Implementation Model View

In this view, the structural and behavioural as parts of the system are represented as they are to be built.

5.Environmental Model View

In this view, the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

5.2 UML Diagrams

5.2.1 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running/operating. So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we must discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So, to model the entire system numbers of use case diagrams are used. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analysed to gather its functionalities use cases are prepared and actors are identified. In brief, the purposes of use case diagrams can be as follows:

a. Used to gather requirements of a system.

b. Used to get an outside view of a system.

c. Identify external and internal factors influencing the system.

d. Show the interacting among the requirements are actors.

The following topics describe model elements in use-case diagrams:

- Use cases
  A use case describes a function that a system performs to achieve the user’s goal. A use case must yield an observable result that is of value to the user of the system.

- Actors

An actor represents a role of a user that interacts with the system that you are modelling. The user can be a human user, an organization, a machine, or another external system.

- Subsystems
  In UML models, subsystems are a type of stereotyped component that represent independent, behavioural units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modelling.

- Relationships in use-case diagrams
  In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behaviour between the model elements.

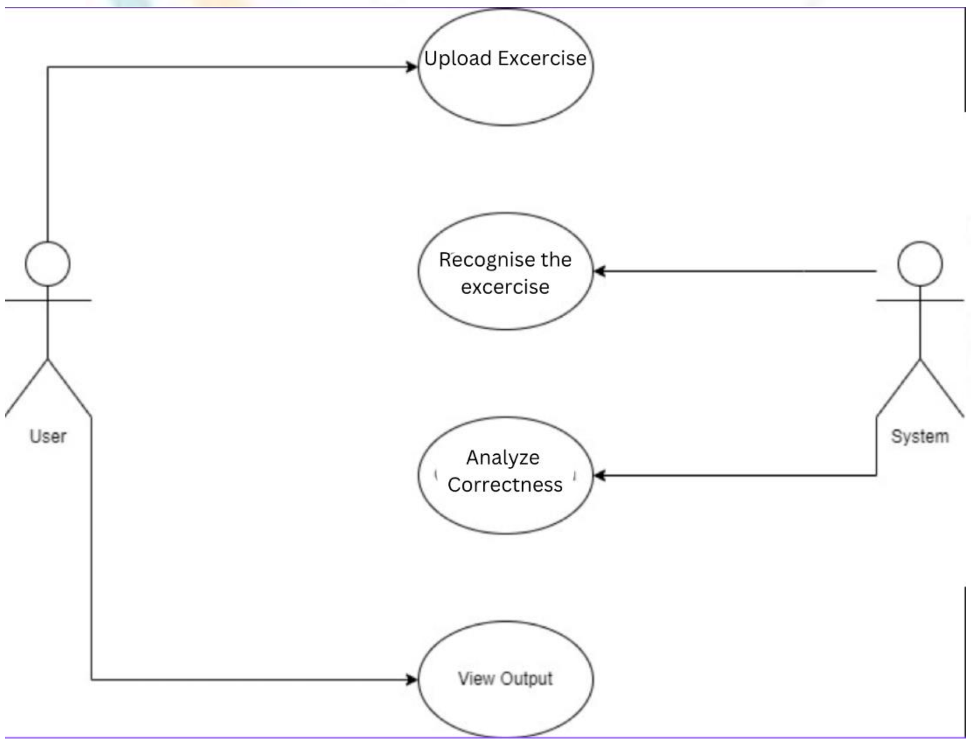


Fig – 5.1 Use Case Diagram

5.2.2 Activity Diagram

- Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. It is also suitable for modeling how a collection of use cases coordinate to represent business workflows.
- Identify candidate use cases, through the examination of business workflows.
- Identify pre- and post-conditions (the context) for use cases.
- Model workflows between/within use cases.
- Model complex workflows in operations on objects.
- Model in detail complex activities in a high level activity diagram.

The following diagram depicts the activity diagram of the Fitness Correction Using ML.

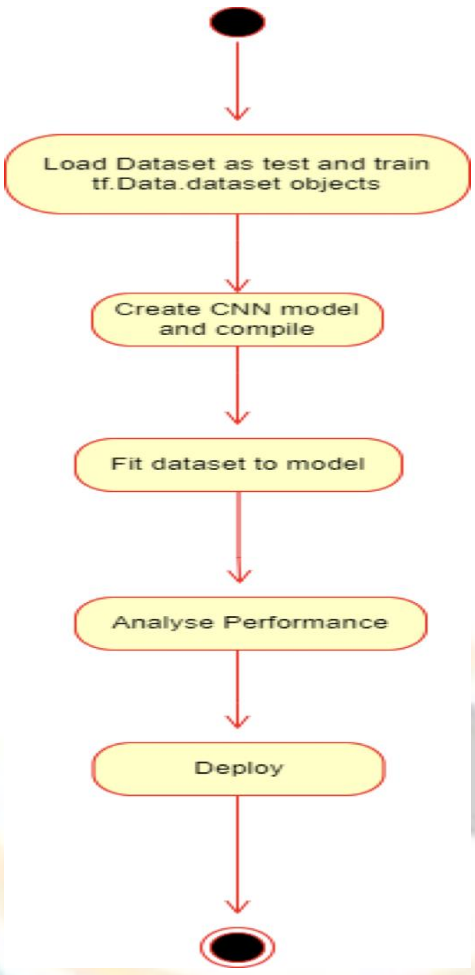
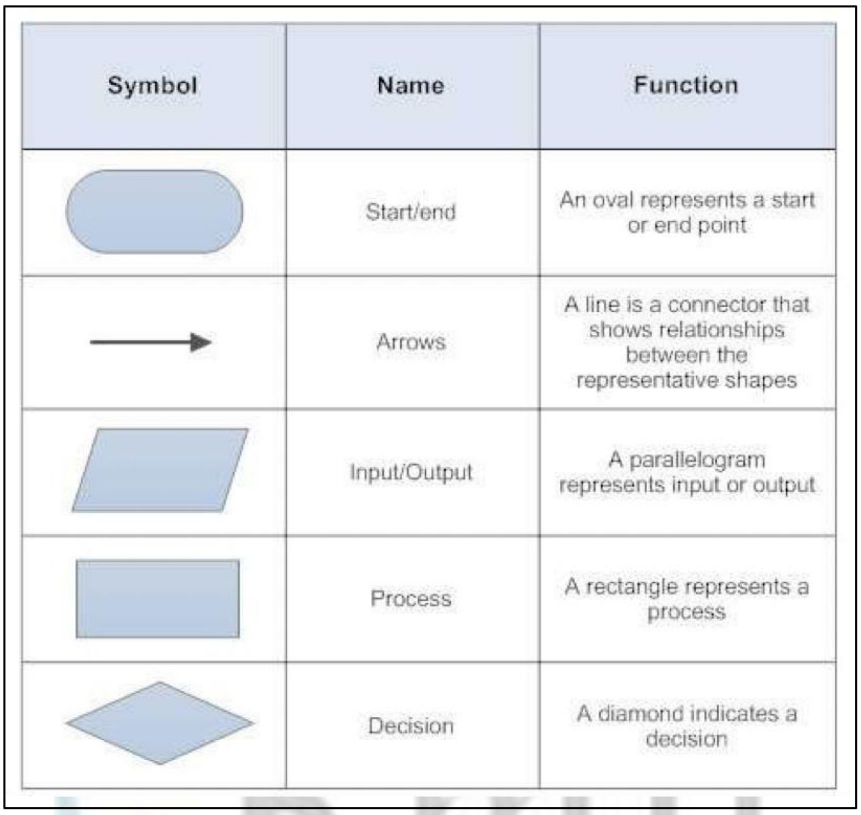

Load Dataset as test and train tf.Data.dataset objects
Create CNN model and compile
Fit dataset to model
Analyse Performance
Deploy

Fig – 5.2 Activity Diagram

5.2.3 Data Flow Diagram

A data-flow diagram is a diagram that shows how data flows through a system or process DFD also includes data upon every entity's output and input, as well as the operation itself. There are no steps involved or loops in a data-flow diagram, so there is no transmission. It's a modelling language used in the creation of object-oriented software. It's also used to build a system overview that may be cancelled afterward.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point
→	Arrows	A line is a connector that shows relationships between the representative shapes
▱	Input/Output	A parallelogram represents input or output
▭	Process	A rectangle represents a process
◇	Decision	A diamond indicates a decision

The flowing block explains the data flow in checking for the repetitions of the right angles and movement of the exercise.

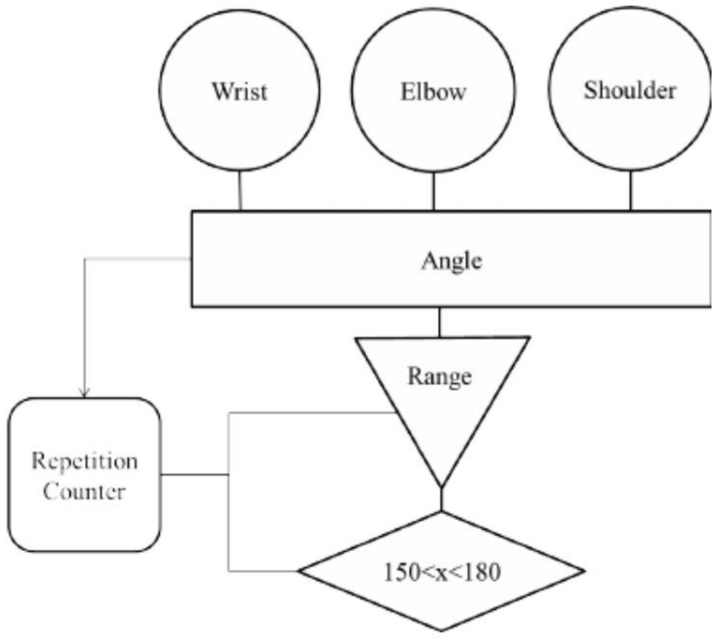


Fig – 5.3 Data Flow Diagram

5.2.4 Architecture

This chapter explains the architecture of the implemented system, nodal point selection for each exercise and implementation of exercise analyzation using computer vision technique. The proposed system consists of several blocks of work. The Figure 3.1 shows the standard functional block diagram of the simplified system. The following part explains the importance of the few blocks of the system.

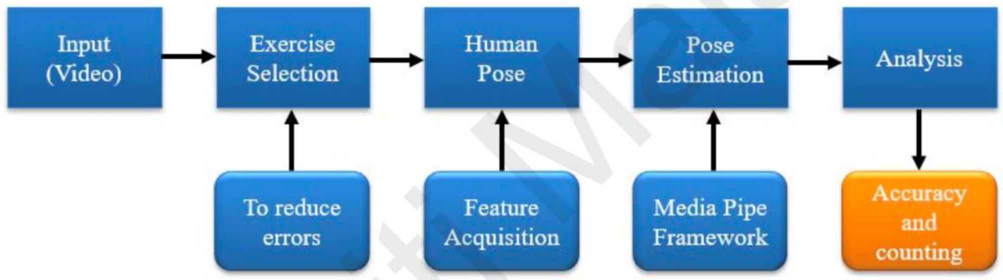


Fig – 5.4 Architecture Diagram

CHAPTER - 6

6.IMPLEMENTATION

6.1 Libraries Used

JupyterLab: JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

Mediapipe: Mediapipe is an open-source library developed by Google that provides a framework for building various real-time multimedia processing pipelines. It offers a wide range of pre-built modules and tools for tasks such as hand tracking, pose estimation, face detection, object tracking, and more. Mediapipe is commonly used in computer vision applications, including gesture recognition, augmented reality, and human-computer interaction.

NumPy: NumPy is a fundamental library for scientific computing in Python. It provides efficient and high-performance multidimensional array objects, along with a collection of functions for manipulating and operating on these arrays. NumPy is widely used for numerical computations, linear algebra, Fourier transforms, random number generation, and other mathematical operations. It serves as the foundation for many other scientific and data analysis libraries in Python.

Tkinter: Tkinter is the standard Python interface to the Tk GUI toolkit, which allows developers to create graphical user interfaces (GUIs) for their applications. It provides a set of tools and widgets for building windows, buttons, menus, text boxes, and other GUI components. Tkinter is included in Python's standard library, making it readily available for creating cross-platform desktop applications with a native look and feel. It is known for its simplicity and ease of use, making it a popular choice for beginners in GUI programming with Python.

6.2 Code:

```
import cv2
import mediapipe as mp
import numpy as np
import tkinter as tk
from PIL import Image, ImageTk
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose


def calculate_angle(a,b,c):
    x = np.array(a) # First
    b = np.array(b) # Mid
    c = np.array(c) # End

    radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
    angle = np.abs(radians*180.0/np.pi)

    if angle >180.0:
        angle = 360-angle


    return angle
```

```
from tkinter import *
import tkinter.font as font
from PIL import ImageTk, Image

def squat():
    cap = cv2.VideoCapture(1)

    # Curl counter variables
    counter = 0
    stage = None
    dir=0

    ## Setup mediapipe instance
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as
pose:
        while cap.isOpened():
            ret, frame = cap.read()

            # Recolor image to RGB
            #frame = cv2.resize(frame, (1600, 900))
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image.flags.writeable = False

            # Make detection
            results = pose.process(image)

            # Recolor back to BGR
            image.flags.writeable = True
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # Extract landmarks
            try:
                landmarks = results.pose_landmarks.landmark
            # Get coordinates
            shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,landmarks[mp_pose.PoseLandmark
.LEFT_HIP.value].y]
            elbow =
[landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x,landmarks[mp_pose.PoseLandm
ark.LEFT_KNEE.value].y]
            wrist =
[landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].x,landmarks[mp_pose.PoseLand
mark.LEFT_ANKLE.value].y]

            # Calculate angle
            angle = calculate_angle(shoulder, elbow, wrist)

            # Visualize angle
            cv2.putText(image, str(angle),
                        tuple(np.multiply(elbow, [640, 480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
                            )
             # Curl counter logic
            if angle > 140:
                stage = "up"
            if angle < 110 and stage =='up':
                stage="down"
                counter +=1

        except:
            pass

        # Render curl counter
        # Setup status box
```

```
            cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)

            # Rep data
            cv2.putText(image, 'REPS', (15,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, str(counter),
                        (10,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

            # Stage data
            cv2.putText(image, 'STAGE', (95,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, stage,
                        (90,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)




            # Render detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                    mp_drawing.DrawingSpec(color=(0,0,255), thickness=2,
circle_radius=2),
                                    mp_drawing.DrawingSpec(color=(0,255,255), thickness=2,
circle_radius=2)
                                     )

            cv2.imshow('Mediapipe Feed', image)

            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()

def lat():
    cap = cv2.VideoCapture(1)

    # Curl counter variables
    counter = 0
    stage = None
    dir=0

    ## Setup mediapipe instance
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as
pose:
        while cap.isOpened():
            ret, frame = cap.read()

            # Recolor image to RGB
            #frame = cv2.resize(frame, (1600, 900))
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image.flags.writeable = False

            # Make detection
            results = pose.process(image)

            # Recolor back to BGR
            image.flags.writeable = True
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # Extract landmarks
            try:
                landmarks = results.pose_landmarks.landmark

                # Get coordinates
            shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,landmarks[mp_pose.PoseLandmark
.LEFT_HIP.value].y]
            elbow =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.Pose
Landmark.LEFT_SHOULDER.value].y]
            wrist =
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseLand
mark.LEFT_ELBOW.value].y]

            # Calculate angle
            angle = calculate_angle(shoulder, elbow, wrist)

            # Visualize angle
            cv2.putText(image, str(angle),
                        tuple(np.multiply(elbow, [640, 480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
                            )
             # Curl counter logic
            if angle > 75:
                stage = "up"
            if angle < 30 and stage =='up':
                stage="down"
                counter +=1

        except:
            pass

        # Render curl counter
        # Setup status box
        cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)
```

```
            # Rep data
            cv2.putText(image, 'REPS', (15,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, str(counter),
                        (10,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

            # Stage data
            cv2.putText(image, 'STAGE', (95,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, stage,
                        (90,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)



            # Render detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                    mp_drawing.DrawingSpec(color=(0,0,255), thickness=2,
circle_radius=2),
                                    mp_drawing.DrawingSpec(color=(0,255,255), thickness=2,
circle_radius=2)
                                     )

            cv2.imshow('Mediapipe Feed', image)

            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()
```

```
def bicepcurl():
    cap = cv2.VideoCapture(1)

    # Curl counter variables
    counter = 0
    stage = None
    dir=0

    ## Setup mediapipe instance
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as
pose:
        while cap.isOpened():
            ret, frame = cap.read()

            # Recolor image to RGB
            #frame = cv2.resize(frame, (1600, 900))
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image.flags.writeable = False

            # Make detection
            results = pose.process(image)

            # Recolor back to BGR
            image.flags.writeable = True
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # Extract landmarks
            try:
                landmarks = results.pose_landmarks.landmark

                # Get coordinates
            shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.Pose
Landmark.LEFT_SHOULDER.value].y]
            elbow =
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseLand
mark.LEFT_ELBOW.value].y]
            wrist =
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_pose.PoseLand
mark.LEFT_WRIST.value].y]

            # Calculate angle
            angle = calculate_angle(shoulder, elbow, wrist)

            # Visualize angle
            cv2.putText(image, str(angle),
                        tuple(np.multiply(elbow, [640, 480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
                            )
             # Curl counter logic
            if angle > 120:
                stage = "down"
            if angle < 50 and stage =='down':
                stage="up"
                counter +=1

        except:
            pass

        # Render curl counter
        # Setup status box
        cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)
```

```
            # Rep data
            cv2.putText(image, 'REPS', (15,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, str(counter),
                        (10,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

            # Stage data
            cv2.putText(image, 'STAGE', (95,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, stage,
                        (90,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)



            # Render detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                    mp_drawing.DrawingSpec(color=(0,0,255), thickness=2,
circle_radius=2),
                                    mp_drawing.DrawingSpec(color=(0,255,255), thickness=2,
circle_radius=2)
                                     )

            cv2.imshow('Mediapipe Feed', image)

            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()
```

```
def push():
    cap = cv2.VideoCapture(1)

    # Curl counter variables
    counter = 0
    stage = None
    dir=0

    ## Setup mediapipe instance
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as
pose:
        while cap.isOpened():
            ret, frame = cap.read()

            # Recolor image to RGB
            #frame = cv2.resize(frame, (1600, 900))
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image.flags.writeable = False

            # Make detection
            results = pose.process(image)

            # Recolor back to BGR
            image.flags.writeable = True
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # Extract landmarks
            try:
                landmarks = results.pose_landmarks.landmark

                # Get coordinates
            shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.Pose
Landmark.LEFT_SHOULDER.value].y]
            elbow =
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseLand
mark.LEFT_ELBOW.value].y]
            wrist =
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_pose.PoseLand
mark.LEFT_WRIST.value].y]

            # Calculate angle
            angle = calculate_angle(shoulder, elbow, wrist)

            # Visualize angle
            cv2.putText(image, str(angle),
                        tuple(np.multiply(elbow, [640, 480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
                            )
             # Curl counter logic
            if angle > 120:
                stage = "up"
            if angle < 100 and stage =='up':
                stage="down"
                counter +=1

        except:
            pass

        # Render curl counter
        # Setup status box
        cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)
```

```
            # Rep data
            cv2.putText(image, 'REPS', (15,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, str(counter),
                        (10,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

            # Stage data
            cv2.putText(image, 'STAGE', (95,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, stage,
                        (90,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)



            # Render detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                    mp_drawing.DrawingSpec(color=(0,0,255), thickness=2,
circle_radius=2),
                                    mp_drawing.DrawingSpec(color=(0,255,255), thickness=2,
circle_radius=2)
                                     )

            cv2.imshow('Mediapipe Feed', image)

            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()
```

```
def spress():
    cap = cv2.VideoCapture(1)

    # Curl counter variables
    counter = 0
    stage = None
    dir=0

    ## Setup mediapipe instance
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as
pose:
        while cap.isOpened():
            ret, frame = cap.read()

            # Recolor image to RGB
            #frame = cv2.resize(frame, (1600, 900))
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image.flags.writeable = False

            # Make detection
            results = pose.process(image)

            # Recolor back to BGR
            image.flags.writeable = True
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # Extract landmarks
            try:
                landmarks = results.pose_landmarks.landmark

                # Get coordinates
            shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.Pose
Landmark.LEFT_SHOULDER.value].y]
            elbow =
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseLand
mark.LEFT_ELBOW.value].y]
            wrist =
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_pose.PoseLand
mark.LEFT_WRIST.value].y]

            # Calculate angle
            angle = calculate_angle(shoulder, elbow, wrist)

            # Visualize angle
            cv2.putText(image, str(angle),
                        tuple(np.multiply(elbow, [640, 480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
                            )
             # Curl counter logic
            if angle < 120:
                stage = "down"
            if angle > 150 and stage =='down':
                stage="up"
                counter +=1

        except:
            pass

        # Render curl counter
        # Setup status box
        cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)
```

```
            # Rep data
            cv2.putText(image, 'REPS', (15,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, str(counter),
                        (10,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

            # Stage data
            cv2.putText(image, 'STAGE', (95,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, stage,
                        (90,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)



            # Render detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                    mp_drawing.DrawingSpec(color=(0,0,255), thickness=2,
circle_radius=2),
                                    mp_drawing.DrawingSpec(color=(0,255,255), thickness=2,
circle_radius=2)
                                     )

            cv2.imshow('Mediapipe Feed', image)

            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()
```

```
def situps():
    cap = cv2.VideoCapture(1)

    # Curl counter variables
    counter = 0
    stage = None
    dir=0

    ## Setup mediapipe instance
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as
pose:
        while cap.isOpened():
            ret, frame = cap.read()

            # Recolor image to RGB
            #frame = cv2.resize(frame, (1600, 900))
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image.flags.writeable = False

            # Make detection
            results = pose.process(image)

            # Recolor back to BGR
            image.flags.writeable = True
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # Extract landmarks
            try:
                landmarks = results.pose_landmarks.landmark

                # Get coordinates
            shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.Pose
Landmark.LEFT_SHOULDER.value].y]
            elbow =
[landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,landmarks[mp_pose.PoseLandmark
.LEFT_HIP.value].y]
            wrist =
[landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x,landmarks[mp_pose.PoseLandm
ark.LEFT_KNEE.value].y]

            # Calculate angle
            angle = calculate_angle(shoulder, elbow, wrist)

            # Visualize angle
            cv2.putText(image, str(angle),
                        tuple(np.multiply(elbow, [640, 480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
                            )
             # Curl counter logic
            if angle > 90:
               stage = "down"
            if angle < 50 and stage =='down':
               stage="up"
               counter +=1

         except:
            pass

         # Render curl counter
         # Setup status box
         cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)
```

```
            # Rep data
            cv2.putText(image, 'REPS', (15,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, str(counter),
                        (10,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

            # Stage data
            cv2.putText(image, 'STAGE', (95,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
            cv2.putText(image, stage,
                        (90,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)



            # Render detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                    mp_drawing.DrawingSpec(color=(0,0,255), thickness=2,
circle_radius=2),
                                    mp_drawing.DrawingSpec(color=(0,255,255), thickness=2,
circle_radius=2)
                                     )

            cv2.imshow('Mediapipe Feed', image)

            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()
```

```
def details():
    top= Toplevel()
    top.title("Team Details")
    deetlabel1 = Label(top,text="K. Sai Prajwal").pack()
    deetlabel2 = Label(top,text="Abhiram Krishnam").pack()
    deetlabel3 = Label(top,text="K. Sai Prajwal").pack()


root = Tk()
#root.geometry("500x500")
myFont = font.Font(size=12)

myimg=ImageTk.PhotoImage(Image.open("mainphoto.jpg"))
mylabel=Label(image=myimg)
mylabel.grid(row=0,column=0,rowspan=6)

DetailsButton=Button(root,text="Team Details",command=details)
DetailsButton.grid(row=6,column=0)

BicepButton = Button(root, text="Bicep Curl",command=bicepcurl,height = 2, width =
13,bg='#0052cc', fg='#ffffff')
BicepButton['font'] = myFont
BicepButton.grid(row=0,column=1)

SquatButton = Button(root,text="Squat",command=squat,height=2, width = 13,bg='#0052cc',
fg='#ffffff')
SquatButton['font'] = myFont
SquatButton.grid(row=1,column=1)
```

```
LatButton = Button(root,text="Lateral Raise",command=lat,height=2, width =
13,bg='#0052cc', fg='#ffffff')
LatButton['font'] = myFont
LatButton.grid(row=2,column=1)

PushupButton = Button(root,text="Push Up",command=push,height=2, width =
13,bg='#0052cc', fg='#ffffff')
PushupButton['font'] = myFont
PushupButton.grid(row=3,column=1)

PushupButton = Button(root,text="Push Up",command=push,height=2, width =
13,bg='#0052cc', fg='#ffffff')
PushupButton['font'] = myFont
PushupButton.grid(row=3,column=1)

ShoulderButton = Button(root,text="Shoulder Press",command=spress,height=2, width =
13,bg='#0052cc', fg='#ffffff')
ShoulderButton['font'] = myFont
ShoulderButton.grid(row=4,column=1)

SitUpsButton = Button(root,text="Sit Ups",command=situps,height=2, width =
13,bg='#0052cc', fg='#ffffff')
SitUpsButton['font'] = myFont
SitUpsButton.grid(row=5,column=1)


root.mainloop()
```

6.3 Screenshots

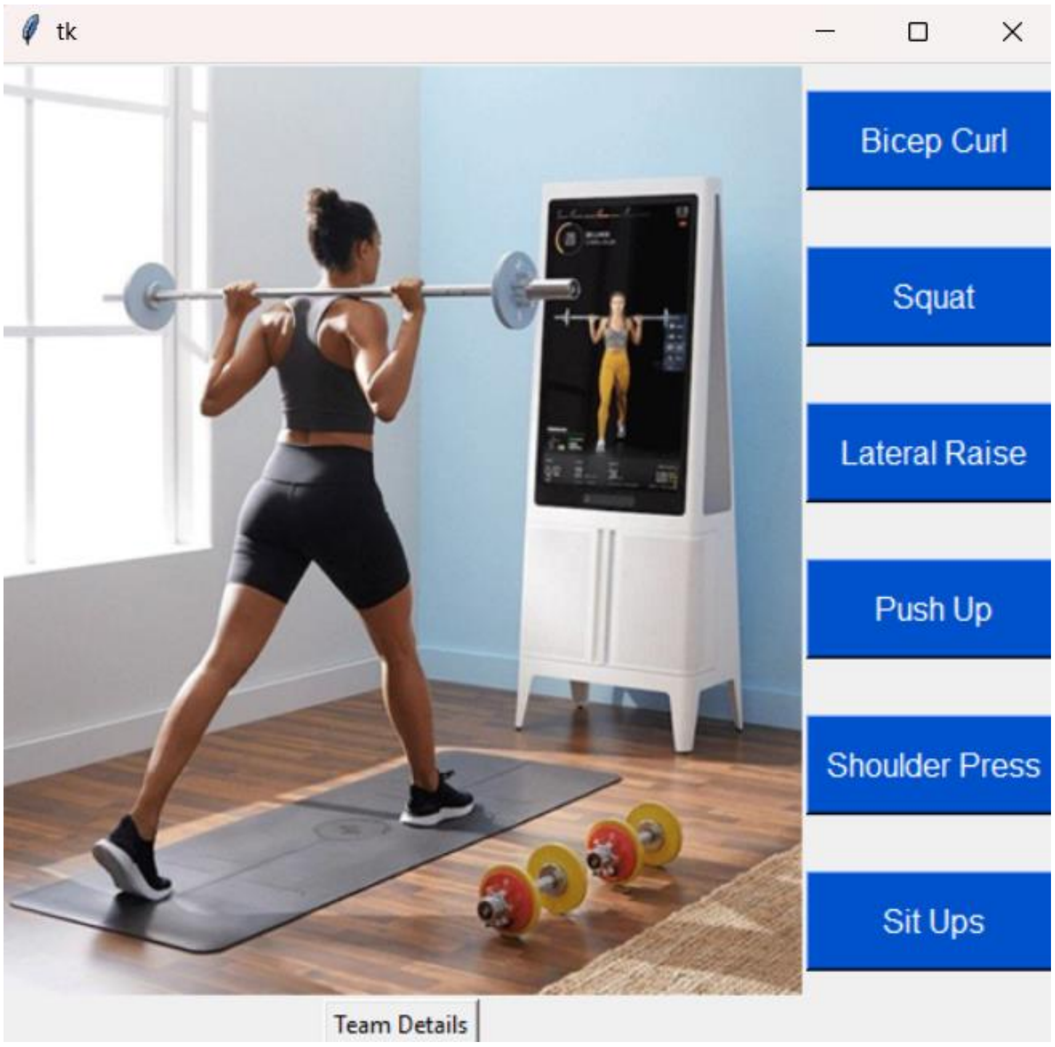


Fig – 6.1 UI Of Exercise Menu

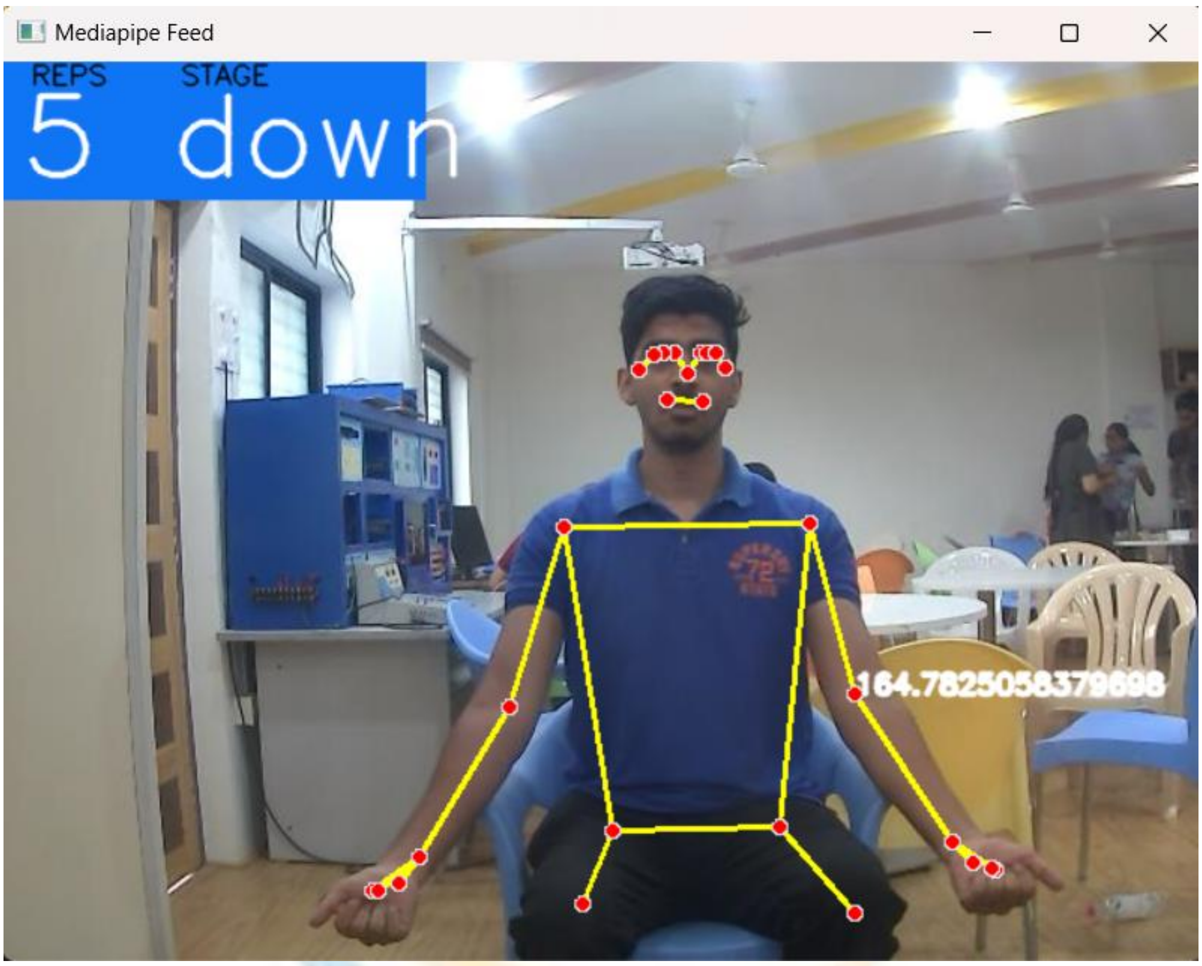


Fig – 6.2 UI Of Exercise Window

CHAPTER – 7

7. TESTING

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification

7.1 Software Validation

Validation is process of examining whether the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which attempts all that user needs from this software?".
- Validation emphasizes on user requirements.

7.2 Software Verification

Verification is the process of confirming if the software is meeting the business requirements and is developed adhering to the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by firmly following all design specifications?”
- Verifications concentrate on the design and system specifications.

7.3 Target Of The Test Are

- Errors -These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, considered as an error.
- Fault - When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause system to fail.
- Failure - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

7.4 Black-Box Testing

It is carried out to test functionality of the program. It is also called ‘Behavioral’ testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested ‘ok’, and problematic otherwise.

7.5 Black-Box Testing Techniques

Equivalence class - T he input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.

- Boundary values - T he input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
- Cause-effect graphing - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- Pair-wise Testing - The behavior of software depends on multiple parameters. In pair wise testing, the multiple parameters are tested pair-wise for their different values.
- State-based testing - The system changes state on provision of input. These systems are tested based on their states and input.

7.6 White-Box Testing

It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as 'Structural' testing
In this testing method, the de sign and structure of the code are known to the tester.

Programmers of the code conduct this test on the code.

The following are some White box testing techniques

- Control-flow testing - The purpose of the control-flow testing to set up a test case which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- Data-flow testing - This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

7.7 Testing Levels

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified. Testing separately is done just to make sure that there are no hidden bugs or issues left in the software.

7.8 Unit Testing

While coding, the programmer performs some tests on that unit of program to know if 64 it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

7.9 Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passes and data updating etc.

7.10 System Testing

The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- Functionality testing - Tests all functionalities of the software against requirement.
- Performance testing - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.
- Security & Portability - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

7.11 Acceptance Testing

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

- Alpha testing - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.
- Beta testing - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not as yet the delivered product. Developers expect

that users at this stage will bring minute 65 problems, which were skipped to attend.

7.12 Regression Testing

Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code.

7.13 End-to-End Testing

End-to-End testing is a type of Software testing that not only validates the software system under test but also check its integration with external interfaces. It uses actual production like data and test environment to simulate real-time settings. End-to-End testing is also called Chain testing. End-to-End design framework consists of three parts: Build User functions, Build conditions, Build test cases.

CHAPTER – 8

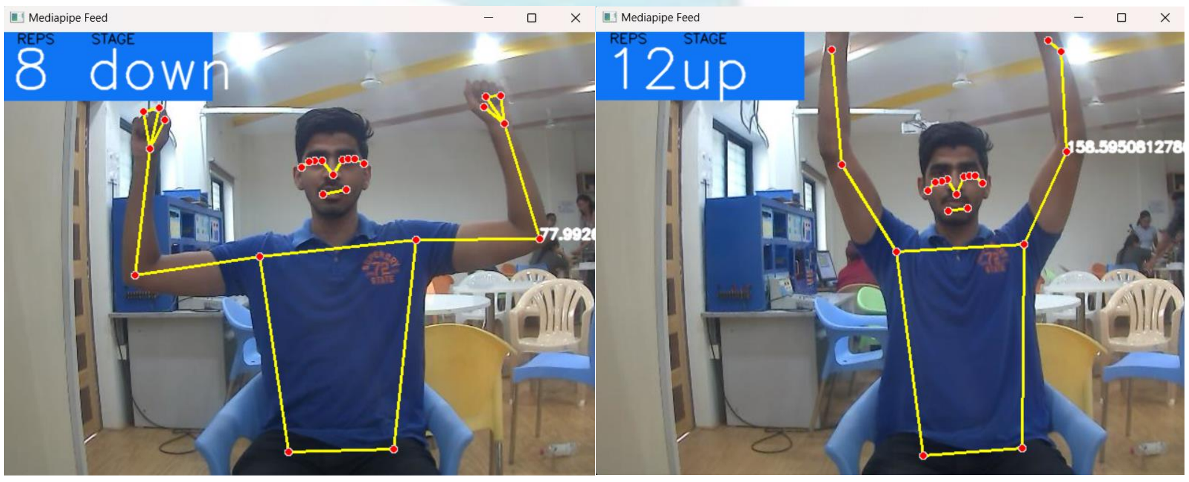
8. RESULTS

8.1 Introduction

This chapter explains the results of exercise analyzation which is analyzed by the proposed system. The discussion will have briefing about the counting methodology, characteristic curves of the accuracy, predictions, and performance of the model in the user environment.

8.2 Output of the implemented system

The BlazePose model and methodologies proposed in this work are implemented using Python frameworks. The Exercise Dashboard was created using Tkinter GUI and results were displayed as Human Interactive model. The following Figures clearly shows the interactive system which display accuracy, counting methodology dynamically according to the changes in the human exercise postures.



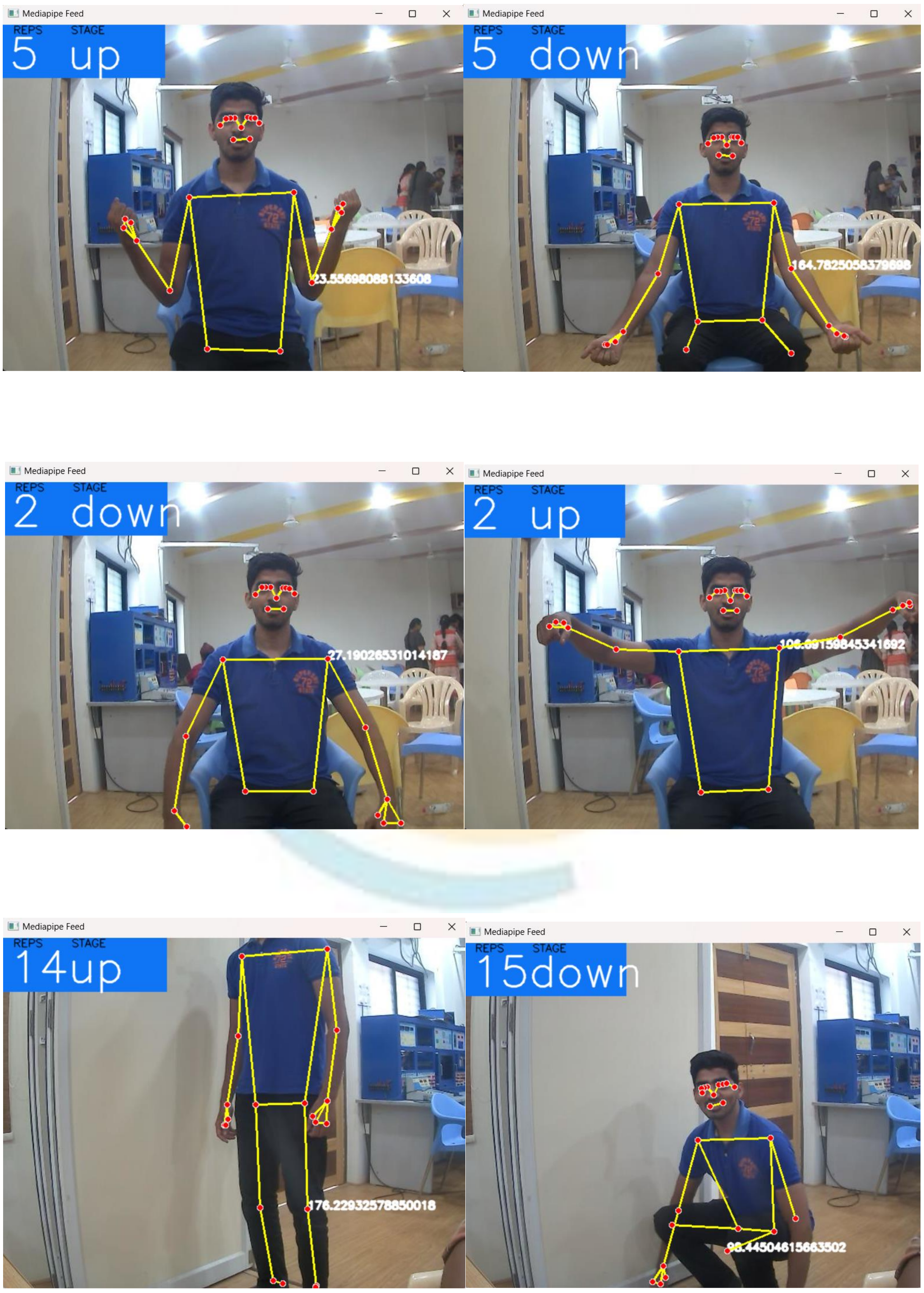


Fig – 8.1 Output of Various Exercises

8.3 Robustness Of The System

The implemented system was tested with the different users. The system can identify and analyze exercise for all age groups. The pre-processing technique like regularization is not used, because the nodal analysis of human pose is used to estimate the pose. The system mainly works on the data acquired by the camera, it works better under the better lighting environment and only one user can perform exercise to verify the results. The input video frames are resized to 1280x720 pixel size to clearly identify the nodal points. The Postures which are performed outside the Region of Interest cannot led exercise analyzation process due to missing of nodal points. The User can perform exercise within range of 3m to 5m distance away from the camera, exceeding the range could not give better results. The range is calculated based on the few trials, beyond the 5m distance away the user will not be in the Region of Interest of the camera, and the system could not identify nodal points and perform analyzation for the performed exercise.

8.4 Summary

This chapter, initially, explains the results as output of the exercises performed in user environments. chapter provides information on accuracy estimation based on the calculations and gives the comparison of the counting methods for exercises. At last, this chapter discusses the robustness of the implemented system.

CHAPTER –9

9. FUTURE ENHACEMENTS

The model performs well on the user environments, but still there are some limitations for the methods. The following steps can be the limitations which can be extended, for further research:

1) The exercise postures are analysed in this project, based on the user’s selection. The new design approach is needed to overcome this limitation. This is a complicated work that needs additional research, due to because of the mixture or similarity of the same exercise postures.
2) This project has completed an analysis for 6 exercises. However, there are lot more exercises are out there, which need experimentation of postures to analyse those posture patterns.
3) The proposed computer vision techniques failed to analyse some of the exercises like Jump Rope, Jump squat and etc. Therefore, further extension of works is needed to analyse these exercises.
4) The application needs to have a video graphic depicting the correct variation of the workout in case the user is performing it in an incorrect manner.
5) There needs to be conversion of the application from a python executable to a mobile application which unitizes the camera and can also log all historic data on an android app can help the application very user-friendly.

CHAPTER – 10

10. CONCLUSION

The identification and analyzation of exercise is achievable only with complex invasive system design. The few non-invasive methods are also proposed earlier this works are mostly computationally expensive. The human exercise pose should be analyzed with simple system, that should give better accuracy. The extraction of features is a crucial process for the better analyzation.

In this research project, the proposed system identifies and analyses the exercise postures non-invasively with simple design. At the same time, a total of 14 types of exercises are analysed in most of the exercise environments. The system gives better output characteristics, the experiment adopted computer vision and deep learning model to accomplish the expected outcomes. We conducted intensive testing on different users to evaluate the performance of the proposed system. The model dynamically performs well and give better output.

CHAPTER – 11

11. REFERENCES

Alegria, E. C., & Serra, A. C. (2000). Automatic calibration of analog and digital measuring instruments using computer vision. *IEEE Transactions on Instrumentation and Measurement, 49*(1), 94-99. doi:10.1109/19.836317

Ar, I., & Akgul, Y. S. (2014). A Computerized Recognition System for Home- Based Physiotherapy Exercises Using an RGBD Camera. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 22*(6), 1160-1171. doi:10.1109/TNSRE.2014.2326254

Bakar, M. Z. A., Samad, R., Pebrianti, D., Mustafa, M., & Abdullah, N. R. H. (2015, 27-29 Nov. 2015). *Computer vision-based hand deviation exercise for rehabilitation.* Paper presented at the 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE).

Bakchy, S. C., Mondal, M. N. I., Ali, M. M., Sathi, A. H., Ray, K. C., & Ferdous, M. J. (2018, 8-9 Feb. 2018). *Limbs and Muscle Movement Detection using Gait Analysis.* Paper presented at the 2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2).

Engbers, E. A., & Smeulders, A. W. M. (2003). Design considerations for generic grouping in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*(4), 445-457. doi:10.1109/TPAMI.2003.1190571

Ermes, M., PÄrkkÄ, J., MÄntyjÄrvi, J., & Korhonen, I. (2008). Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions. *IEEE Transactions on Information Technology in Biomedicine, 12*(1), 20-26. doi:10.1109/TITB.2007.899496

Frigui, H., & Krishnapuram, R. (1999). A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(5), 450-465. doi:10.1109/34.765656

Huang, J., Lin, S., Wang, N., Dai, G., Xie, Y., & Zhou, J. (2020). TSE-CNN: A Two- Stage End-to-End CNN for Human Activity Recognition. *IEEE Journal of Biomedical and Health Informatics, 24*(1), 292-299. doi:10.1109/JBHI.2019.2909688

Kavian, M., & Nadian-Ghomsheh, A. (2020, 18-20 Feb. 2020). *Monitoring Wrist and Fingers Range of Motion using Leap Motion Camera for Physical Rehabilitation.* Paper presented at the 2020 International Conference on Machine Vision and Image Processing (MVIP).

Kocak, D. M., Lobo, N. d. V., & Widder, E. A. (1999). Computer vision techniques for quantifying, tracking, and identifying bioluminescent plankton. *IEEE Journal of Oceanic Engineering, 24*(1), 81-95. doi:10.1109/48.740157

Lee, D., & Park, Y. (2009). Vision-based remote control system by motion detection and open finger counting. *IEEE Transactions on Consumer Electronics, 55*(4), 2308-2313. doi:10.1109/TCE.2009.5373803

Liu, N. (2017, 24-26 Nov. 2017). *WiGym: Exercise recognition and counting using Wi- Fi signals.* Paper presented at the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS).

Liu, Z., Liu, X., & Li, K. (2020, 6-9 July 2020). *Deeper Exercise Monitoring for Smart Gym using Fused RFID and CV Data.* Paper presented at the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications.

Mekruksavanich, S., & Jitpattanakul, A. (2020, 11-14 March 2020). *Exercise Activity Recognition with Surface Electromyography Sensor using Machine Learning Approach.* Paper presented at the 2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON).

Monkaresi, H., Calvo, R. A., & Yan, H. (2014). A Machine Learning Approach to Improve Contactless Heart Rate Monitoring Using a Webcam. *IEEE Journal of Biomedical and Health Informatics, 18*(4), 1153-1160. doi:10.1109/JBHI.2013.2291900

Nagarkoti, A., Teotia, R., Mahale, A. K., & Das, P. K. (2019, 23-27 July 2019).

Realtime Indoor Workout Analysis Using Machine Learning & Computer Vision. Paper presented at the 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).

Ohya, I., Kosaka, A., & Kak, A. (1998). Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. *IEEE Transactions on Robotics and Automation, 14*(6), 969-978. doi:10.1109/70.736780

Pattamaset, S., Charoenpong, T., & Charoensiriwath, S. (2020, 29 Jan.-1 Feb. 2020).

Evaluation of Physical Exercise for Osteoarthritis of the Knee through Image Processing Technique. Paper presented at the 2020 12th International Conference on Knowledge and Smart Technology (KST).

Perez-Carrasco, J. A., Acha, B., Serrano, C., Camunas-Mesa, L., Serrano-Gotarredona, T., & Linares-Barranco, B. (2010). Fast Vision Through Frameless Event-Based Sensing and Convolutional Processing: Application to Texture Recognition. *IEEE Transactions on Neural Networks, 21*(4), 609-620. doi:10.1109/TNN.2009.2039943

Pun, T., Alecu, T. I., Chanel, G., Kronegg, J., & Voloshynovskiy, S. (2006). Brain- computer interaction research at the computer vision and multimedia laboratory, University of Geneva. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 14*(2), 210-213. doi:10.1109/TNSRE.2006.875544

Rungsawasdisap, N., Lu, X., Yimit, A., Zhang, Z., Mikami, M., & Hagihara, Y. (2019). *Squat Movement Recognition Using Convolutional Neural Network*.

Schez-Sobrino, S., Monekosso, D. N., Remagnino, P., Vallejo, D., & Glez-Morcillo, C. (2019, 9-10 May 2019). *Automatic recognition of physical exercises performed by stroke survivors to improve remote rehabilitation.* Paper presented at the 2019 International Conference on Multimedia Analysis and Pattern Recognition (MAPR).

Schez-Sobrino, S., Vallejo, D., Monekosso, D. N., Glez-Morcillo, C., & Remagnino, P. (2020). A Distributed Gamified System Based on Automatic Assessment of Physical Exercises to Promote Remote Physical Rehabilitation. *IEEE Access, 8,* 91424-91434. doi:10.1109/ACCESS.2020.2995119