EXCERSISE COUNTER USING MEDIAPIPE

BEWIN FELIX R A

SRIDHANUSH RAJAMANICKAM

ABSTRACT

This paper presents an adaptive counting approach that cap-tures weightlifting exercise repetitions despite fluctuations in the speed of performing. By having a hybrid counting approach that utilizes both state counting based on mean acceleration information and a sensor-based peak detection, reliable gym exercise counting with low transmission rates is achieved. Both approaches are running in parallel and enable the system to adapt to changes in the exercise speed in real-time and without losing counting information sent in the past.

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Sometimes we are so busy that we are not even able to care for our body and by caring we mean fitness, exercises and much more, and then we think that we need to make our exercise plan to maintain our body. So let's make a Python script to maintain this record for us. In this program, we add our exercises that we need to do. Exercise can help prevent excess weight gain or help maintain weight loss. When you engage in physical activity, you burn calories. The more intense the activity, the more calories you burn. Excercise delivers oxygen and nutrients to your tissues and helps your cardiovascular system work more efficiently. And when your heart and lung health improve, you have more energy to tackle daily chores. Regular trips to the gym are great, but don't worry if you can't find a large chunk of time to exercise every day. Any amount of activity is better than none at all. To reap the benefits of exercise, just get more active throughout your day — take the stairs instead of the elevator or rev up your household chores. Consistency is key.

1.2 OBJECTIVES

The main intention behind developing this interface is to maintain our health properly without going to gym or setting up any trainer to look into us. It monitors us even there is is no trainers monitors us.it doesn't need any wide range of area and it requires a few space to do our excersice.

1.3 OVERVIEW OF THE PROJECT

- Our project contains an user interface (UI)
- If you access it then you are able to choose two options like **"REAL TIME"** and **"TRAIN".**
- If you choose real time then you will be redirected to the excersice part.
- If you choose train option then you will be redirected to the demo video that how you want to do the excersice.
- If you are ready to do the excersice then the cam will open then estimates the position of the human body based on that your excersice will be counted.

CHAPTER 2
ANALYSIS AND DESIGN

2.1 FUNCTIONAL REQUIREMENTS

A project must be planed before the execution or implementation for better efficiency and accuracy. Functional requirements are the functions or features to be included in the system to satisfy the user needs and business needs. Functional requirements defines the internal working of the software.

SYSTEM REQUIREMENTS:

- VS CODE
- PYTHON

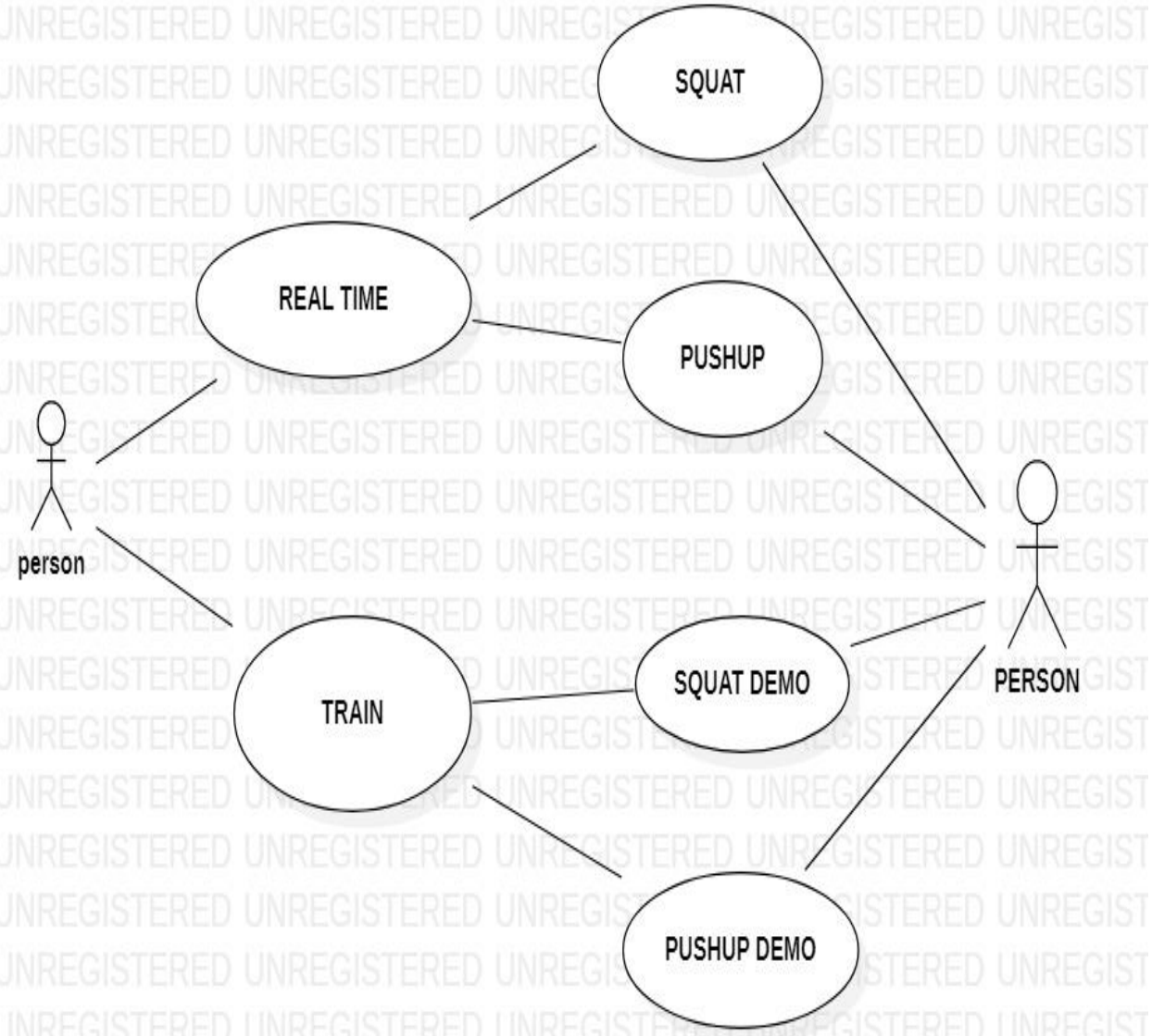
2.2 NON-FUNCTIONAL REQUIREMENTS

- NUMPY
- PANDAS
- TENSOR FLOW
- MEDIA PIPE
- TKINTER

2.3 ARCHIETECTURE

- Our project contains an user interface (UI)
- If you access it then you are able to choose two options like **"REAL TIME"** and **"TRAIN".**
- If you choose real time then you will be redirected to the excersice part.
- If you choose train option then you will be redirected to the demo video that how you want to do the excersice.
- If you are ready to do the excersice then the cam will open then estimates the position of the human body based on that your excersice will be counted.

2.4 USE-CASE DIAGRAM



CHAPTER 3
IMPLEMENTATION

3.1 MODULES DESCRIPTION

3.1.1 NUMPY MODULE

NumPy is a module for Python. The name is an acronym for "Numeric Python" or "Numerical Python". It is pronounced /ˈnʌmpaɪ/ (NUM-py) or less often /ˈnʌmpi (NUM-pee)). It is an extension module for Python, mostly written in C. This makes sure that the precompiled mathematical and numerical functions and functionalities of Numpy guarantee great execution speed. Furthermore, NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays. The implementation is even aiming at huge matrices and arrays, better know under the heading of "big data". Besides that the module supplies a large library of high-level mathematical functions to operate on these matrices and arrays. SciPy (Scientific Python) is often mentioned in the same breath with NumPy. SciPy needs Numpy, as it is based on the data structures of Numpy and furthermore its basic creation and manipulation functions. It extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier-transformation and many others. Both NumPy and SciPy are not part of a basic Python installation. They have to be installed after the Python installation. NumPy has to be installed before installing SciPy.NumPy is a library for the Python programming language, adding support for large,multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUSfiscally sponsored project. NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower thancompiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy. Python bindings of the widely used computer vision library OpenCV utilize NumPyarrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensionalarrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as a universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

3.1.2 PANDAS MODULE

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data. Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

3.1.3 TENSOR FLOW MODULE

The Tensorflow framework is an open end-to-end machine learning platform. It's a symbolic math toolkit that integrates data flow and differentiable programming to handle various tasks related to deep neural network training and inference. It enables programmers to design machine learning applications utilising multiple tools, libraries, and open-source resources.TensorFlow was created with extensive numerical computations in mind, not with deep learning in mind. Nevertheless, it proved valuable for deep learning development, so Google made it open-source. **TensorFlow takes data in tensors, multi-dimensional arrays with more excellent dimensions**. When dealing with enormous amounts of data, multi-dimensional arrays come in helpful. TensorFlow code is considerably easier to execute in a distributed way across a cluster of computers when utilising GPUs because the execution mechanism is in the form of graphs.

3.1.4 MEDIA PIPE MODULE

MediaPipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. This cross-platform Framework works in Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano.MediaPipe is a cross-platform pipeline framework to build custom machine learning solutions for live and streaming media. The framework was open-sourced by Google and is currently in the alpha stage.

What is MediaPipe?
MediaPipe is an open-source framework for building pipelines to perform computer vision inference over arbitrary sensory data such as video or audio. Using MediaPipe, such a perception pipeline can be built as a graph of modular components. MediaPipe is currently in alpha at v0.7, and there may still be breaking API changes. Stable APIs are expected by v1.0.

3.1.5 TKINTER MODULE

Tkinter is a **standard library** in python used for creating **Graphical User Interface (GUI)** for Desktop Applications. With the help of **Tkinter** developing **desktop applications** is not a tough

task. The **primary GUI toolkit** we will be using is Tk, which is Python's default GUI library. We'll access Tk from its Python interface called **Tkinter** (short for **Tk interface**). In Python, **Tkinter** is a standard **GUI** (graphical user interface) package. Tkinter is **Python's default GUI module** and also the most common way that is used for **GUI programming** in Python. Note that **Tkinter** is a set of **wrappers** that implement the Tk widgets as Python classes. kinter in Python helps in **creating GUI** Applications with a minimum hassle. Among various GUI Frameworks, Tkinter is the only framework that is built-in into **Python's Standard Library**. An important feature in favor of Tkinter is that it is **cross-platform**, so the same code can easily work on **Windows**, **macOS**, and **Linux**.

3.2 IMPLEMENTATION DESCRIPTION

- **NUMPY** helps to store the input data in an array format
- **PANDAS** helps the **NUMPY** to store the input in a multidimensional array.
- **TensorFlow** allows you to create dataflow graphs that describe how data moves through a graph.
- **MEDIA PIPE** is an important module in this project which classifies the poses of the hand and body and segments it and based on the actions input into it it makes the counts the actions
- **For example:** actoins of the squats,we need to make the sample images of front and back end images of the squat based on the samples it counts the excersice.
- **Tkinter** module is used to make the python program in a user interface format

3.3 TOOLS USED

3.3.1 VS CODE

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide justthe tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE

3.3.2 PYTHON

Python is commonly used for developing websites and software, task automation, dataanalysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

3.4 SAMPLE CODE

3.4.1 numpy module:

```
import numpy as np
from body_part_angle import BodyPartAngle
from utils import *


class TypeOfExercise(BodyPartAngle):
    def __init__(self, landmarks):
        super().__init__(landmarks)

    def push_up(self, counter, status):
        left_arm_angle = self.angle_of_the_left_arm()
        right_arm_angle = self.angle_of_the_left_arm()
        avg_arm_angle = (left_arm_angle + right_arm_angle) // 2

        if status:
            if avg_arm_angle < 70:
                counter += 1
                status = False
        else:
            if avg_arm_angle > 160:
                status = True

        return [counter, status]

    # def push_up_method_2():

    def pull_up(self, counter, status):
        nose = detection_body_part(self.landmarks, "NOSE")
        left_elbow = detection_body_part(self.landmarks, "LEFT_ELBOW")
        right_elbow = detection_body_part(self.landmarks, "RIGHT_ELBOW")
        avg_shoulder_y = (left_elbow[1] + right_elbow[1]) / 2

        if status:
            if nose[1] > avg_shoulder_y:
                counter += 1
                status = False

        else:
            if nose[1] < avg_shoulder_y:
                status = True

        return [counter, status]
```

```
def squat(self, counter, status):
    left_leg_angle = self.angle_of_the_right_leg()
    right_leg_angle = self.angle_of_the_left_leg()
    avg_leg_angle = (left_leg_angle + right_leg_angle) // 2

    if status:
        if avg_leg_angle < 70:
            counter += 1
            status = False
    else:
        if avg_leg_angle > 160:
            status = True

    return [counter, status]

def walk(self, counter, status):
    right_knee = detection_body_part(self.landmarks, "RIGHT_KNEE")
    left_knee = detection_body_part(self.landmarks, "LEFT_KNEE")

    if status:
        if left_knee[0] > right_knee[0]:
            counter += 1
            status = False

    else:
        if left_knee[0] < right_knee[0]:
            counter += 1
            status = True

    return [counter, status]

def sit_up(self, counter, status):
    angle = self.angle_of_the_abdomen()
    if status:
        if angle < 55:
            counter += 1
            status = False
    else:
        if angle > 105:
            status = True

    return [counter, status]

def calculate_exercise(self, exercise_type, counter, status):
    if exercise_type == "push-up":
```

```
            counter, status = TypeOfExercise(self.landmarks).push_up(
                counter, status)
        elif exercise_type == "pull-up":
            counter, status = TypeOfExercise(self.landmarks).pull_up(
                counter, status)
        elif exercise_type == "squat":
            counter, status = TypeOfExercise(self.landmarks).squat(
                counter, status)
        elif exercise_type == "walk":
            counter, status = TypeOfExercise(self.landmarks).walk(
                counter, status)
        elif exercise_type == "sit-up":
            counter, status = TypeOfExercise(self.landmarks).sit_up(
                counter, status)

        return [counter, status]
```

3.4.2 MEDIA PIPE MODULE

```
import mediapipe as mp
import pandas as pd
import numpy as np
import cv2
from utils import *


class BodyPartAngle:
    def __init__(self, landmarks):
        self.landmarks = landmarks

    def angle_of_the_left_arm(self):
        l_shoulder = detection_body_part(self.landmarks, "LEFT_SHOULDER")
        l_elbow = detection_body_part(self.landmarks, "LEFT_ELBOW")
        l_wrist = detection_body_part(self.landmarks, "LEFT_WRIST")
        return calculate_angle(l_shoulder, l_elbow, l_wrist)

    def angle_of_the_right_arm(self):
        r_shoulder = detection_body_part(self.landmarks, "RIGHT_SHOULDER")
        r_elbow = detection_body_part(self.landmarks, "RIGHT_ELBOW")
        r_wrist = detection_body_part(self.landmarks, "RIGHT_WRIST")
        return calculate_angle(r_shoulder, r_elbow, r_wrist)

    def angle_of_the_left_leg(self):
        l_hip = detection_body_part(self.landmarks, "LEFT_HIP")
        l_knee = detection_body_part(self.landmarks, "LEFT_KNEE")
        l_ankle = detection_body_part(self.landmarks, "LEFT_ANKLE")
        return calculate_angle(l_hip, l_knee, l_ankle)

    def angle_of_the_right_leg(self):
        r_hip = detection_body_part(self.landmarks, "RIGHT_HIP")
```

```
        r_knee = detection_body_part(self.landmarks, "RIGHT_KNEE")
        r_ankle = detection_body_part(self.landmarks, "RIGHT_ANKLE")
        return calculate_angle(r_hip, r_knee, r_ankle)

    def angle_of_the_neck(self):
        r_shoulder = detection_body_part(self.landmarks, "RIGHT_SHOULDER")
        l_shoulder = detection_body_part(self.landmarks, "LEFT_SHOULDER")
        r_mouth = detection_body_part(self.landmarks, "MOUTH_RIGHT")
        l_mouth = detection_body_part(self.landmarks, "MOUTH_LEFT")
        r_hip = detection_body_part(self.landmarks, "RIGHT_HIP")
        l_hip = detection_body_part(self.landmarks, "LEFT_HIP")

        shoulder_avg = [(r_shoulder[0] + l_shoulder[0]) / 2,
                        (r_shoulder[1] + l_shoulder[1]) / 2]
        mouth_avg = [(r_mouth[0] + l_mouth[0]) / 2,
                     (r_mouth[1] + l_mouth[1]) / 2]
        hip_avg = [(r_hip[0] + l_hip[0]) / 2, (r_hip[1] + l_hip[1]) / 2]

        return abs(180 - calculate_angle(mouth_avg, shoulder_avg, hip_avg))

    def angle_of_the_abdomen(self):
        # calculate angle of the avg shoulder
        r_shoulder = detection_body_part(self.landmarks, "RIGHT_SHOULDER")
        l_shoulder = detection_body_part(self.landmarks, "LEFT_SHOULDER")
        shoulder_avg = [(r_shoulder[0] + l_shoulder[0]) / 2,
                        (r_shoulder[1] + l_shoulder[1]) / 2]

        # calculate angle of the avg hip
        r_hip = detection_body_part(self.landmarks, "RIGHT_HIP")
        l_hip = detection_body_part(self.landmarks, "LEFT_HIP")
        hip_avg = [(r_hip[0] + l_hip[0]) / 2, (r_hip[1] + l_hip[1]) / 2]

        # calculate angle of the avg knee
        r_knee = detection_body_part(self.landmarks, "RIGHT_KNEE")
        l_knee = detection_body_part(self.landmarks, "LEFT_KNEE")
        knee_avg = [(r_knee[0] + l_knee[0]) / 2, (r_knee[1] + l_knee[1]) / 2]

        return calculate_angle(shoulder_avg, hip_avg, knee_avg)
```

3.4.3 TKINTER MODULE

1.
```
import tkinter as tk
from tkinter import *
import os
window = tk.Tk()

window.title("AI for workout counting")
window.geometry('500x700')
bg = PhotoImage(file = "file.png")
label1 = Label( window, image = bg)
label1.place(x = 0, y = 0)

label=tk.Label(window, text="For train",font = ('Arial',30))
label.pack(padx=30,pady=30)
```

```
def sit_up():
   os.system('python main.py -t sit-up')

def pull_up():
   os.system('python main.py -t pull-up')

def walk():
   os.system('python main.py -t walk')

def squat():
   os.system('python main.py -t squat')

def push_up():
   os.system('python main.py -t push-up')

#b1 = tk.Button(window, text='Sit up', width=25, command=sit_up,font = ('Arial',15))
#b2 = tk.Button(window, text='Pull up', width=25, command=pull_up,font = ('Arial',15))
#b3 = tk.Button(window, text='Walk', width=25, command=walk,font = ('Arial',15))
b4 = tk.Button(window, text='Squat', width=25, command=squat,font = ('Arial',15))
b5 = tk.Button(window, text='Push_up', width=25, command=push_up,font =
('Arial',15))

button = tk.Button(window, text='Stop', width=25, command=window.destroy,font =
('Arial',15),bg = 'red')

#b1.pack(padx=10,pady=10)
#b2.pack(padx=10,pady=10)
#b3.pack(padx=10,pady=10)
b4.pack(padx=10,pady=10)
b5.pack(padx=10,pady=10)
button.pack(padx=10,pady=10)

label=tk.Label(window, text="By                    \nSri Dhanush
\nURK21CS1039",font = ('Arial',10))
label.pack(padx=40,pady=40,side = BOTTOM,anchor='se')

window.mainloop()
```

```
2   import tkinter as tk
    from tkinter import *
    import os
    window = tk.Tk()

    window.title("AI for workout counting")
    window.geometry('500x700')
    bg = PhotoImage(file = "file.png")
    label1 = Label( window, image = bg)
```

```
label1.place(x = 0, y = 0)

label=tk.Label(window, text="Choose one option",font = ('Arial',30))
label.pack(padx=30,pady=30)

def sit_up():
    os.system('python main.py -t sit-up -vs videos/sit-up.mp4')

def pull_up():
    os.system('python main.py -t pull-up -vs videos/pull-up.mp4')

def walk():
    os.system('python main.py -t walk -vs videos/walk.mp4')

def squat():
    os.system('python main.py -t squat -vs videos/squat.mp4')

def push_up():

    os.system('python main.py -t push-up -vs videos/push-up.mp4')

#b1 = tk.Button(window, text='Sit up', width=25, command=sit_up,font =
('Arial',15))
#b2 = tk.Button(window, text='Pull up', width=25, command=pull_up,font
= ('Arial',15))
#b3 = tk.Button(window, text='Walk', width=25, command=walk,font =
('Arial',15))
b4 = tk.Button(window, text='Squat', width=25, command=squat,font =
('Arial',15))
b5 = tk.Button(window, text='Push_up', width=25, command=push_up,font =
('Arial',15))

button = tk.Button(window, text='Stop', width=25,
command=window.destroy,font = ('Arial',15),bg = 'red')

#b1.pack(padx=10,pady=10)
#b2.pack(padx=10,pady=10)
#b3.pack(padx=10,pady=10)
b4.pack(padx=10,pady=10)
b5.pack(padx=10,pady=10)
button.pack(padx=10,pady=10)

label=tk.Label(window, text="By                                    \nSri
Dhanush \nURK21CS1039 ",font = ('Arial',10))
label.pack(padx=40,pady=40,side = BOTTOM,anchor='se')

window.mainloop()
```

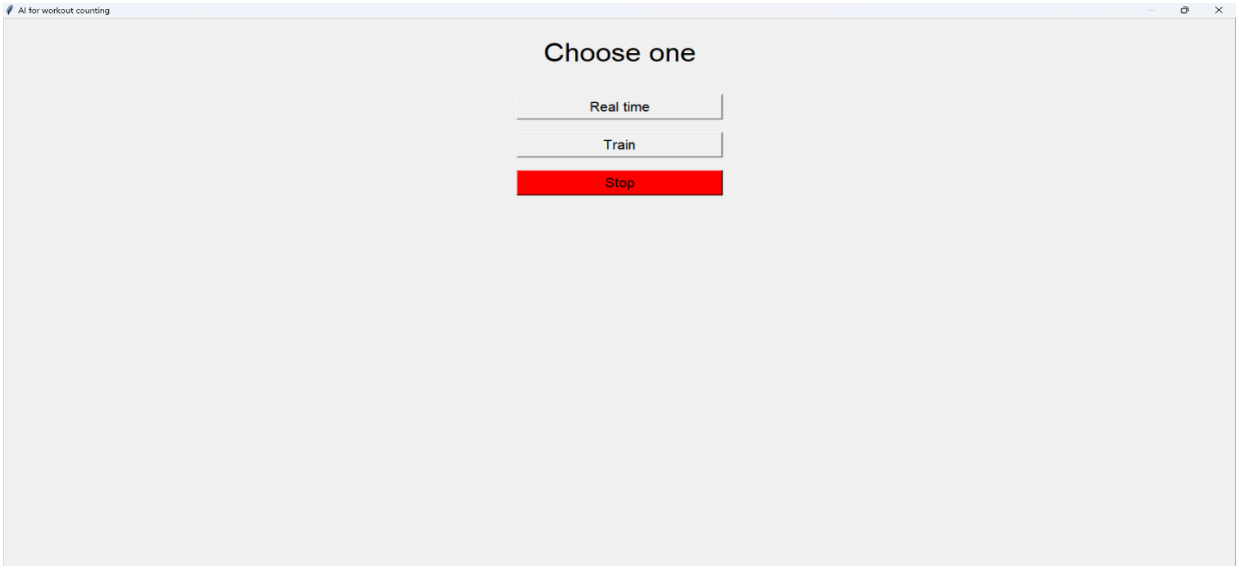
3.4.4 ORGANIZED MODULE

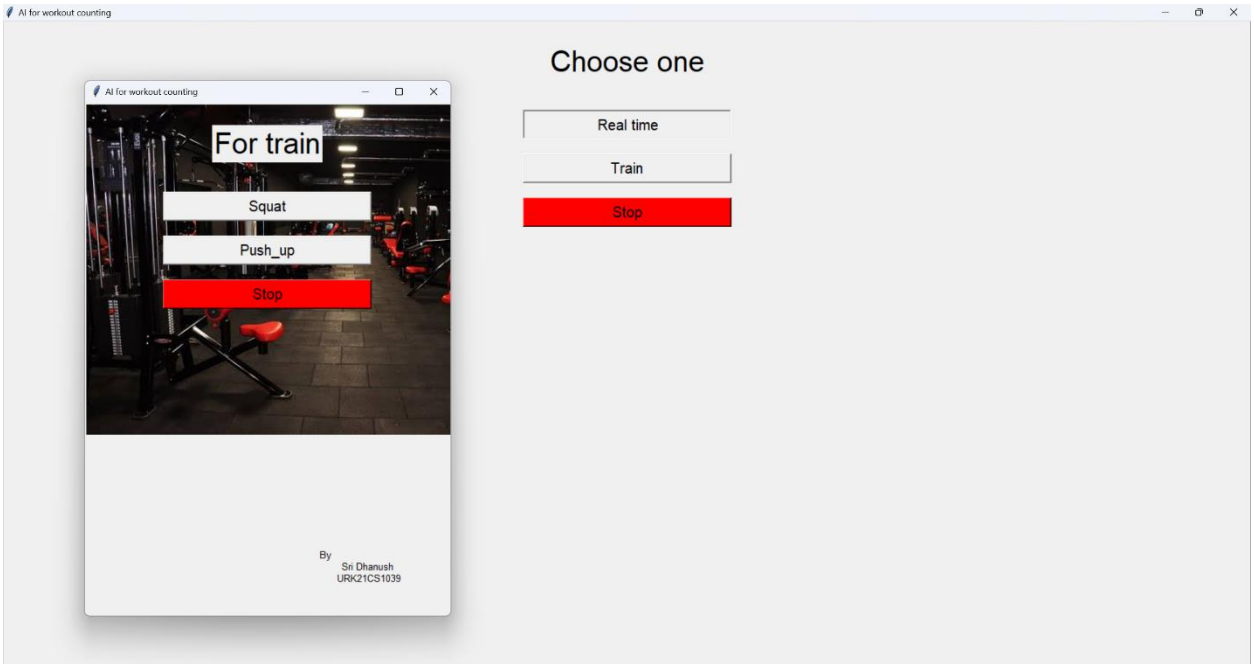
```
import tkinter as tk
from tkinter import *
import os
window = tk.Tk()

window.title("AI for workout counting")
```

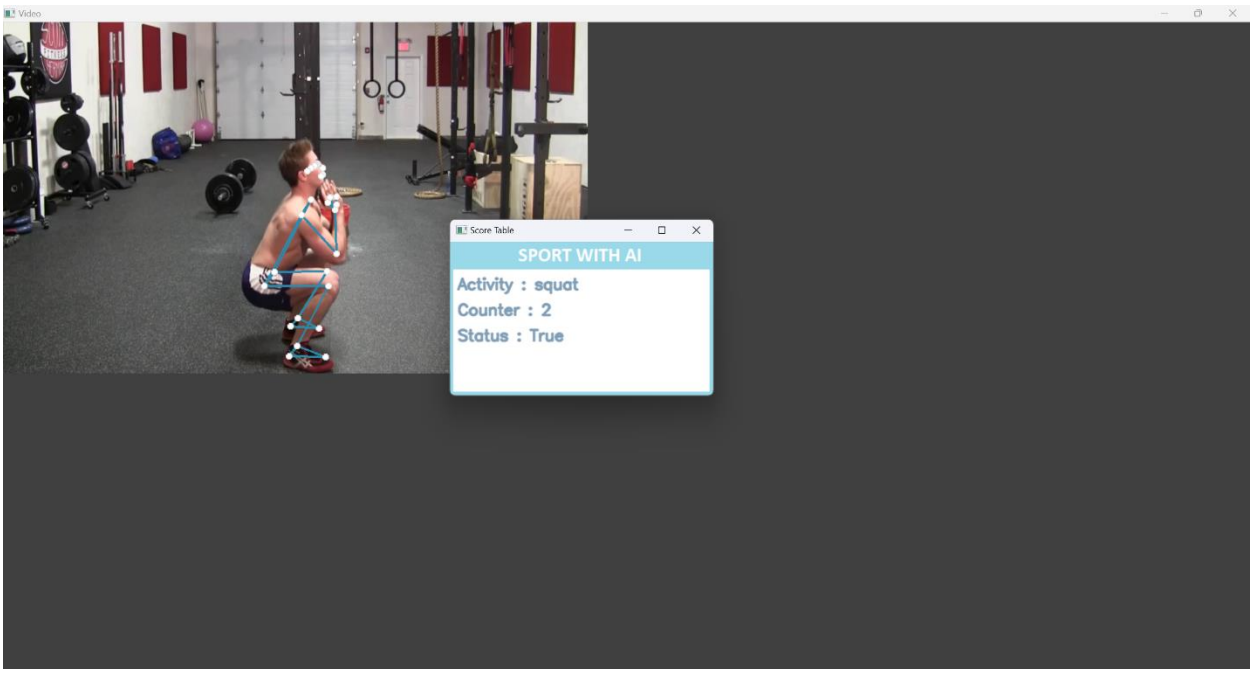
```
window.geometry('500x700')
bg = PhotoImage(file = "file.png")

label=tk.Label(window, text="Choose one",font = ('Arial',30))
label.pack(padx=30,pady=30)

def Real():
    os.system('python GUI.py')

def Train():
    os.system('python test.py')


b1 = tk.Button(window, text='Real time', width=25, command=Real,font =
('Arial',15))
b2 = tk.Button(window, text='Train', width=25, command=Train,font =
('Arial',15))

b1.pack(padx=10,pady=10)
b2.pack(padx=10,pady=10)

button = tk.Button(window, text='Stop', width=25, command=window.destroy,font
= ('Arial',15),bg = 'red')
button.pack(padx=10,pady=10)

window.mainloop()
```

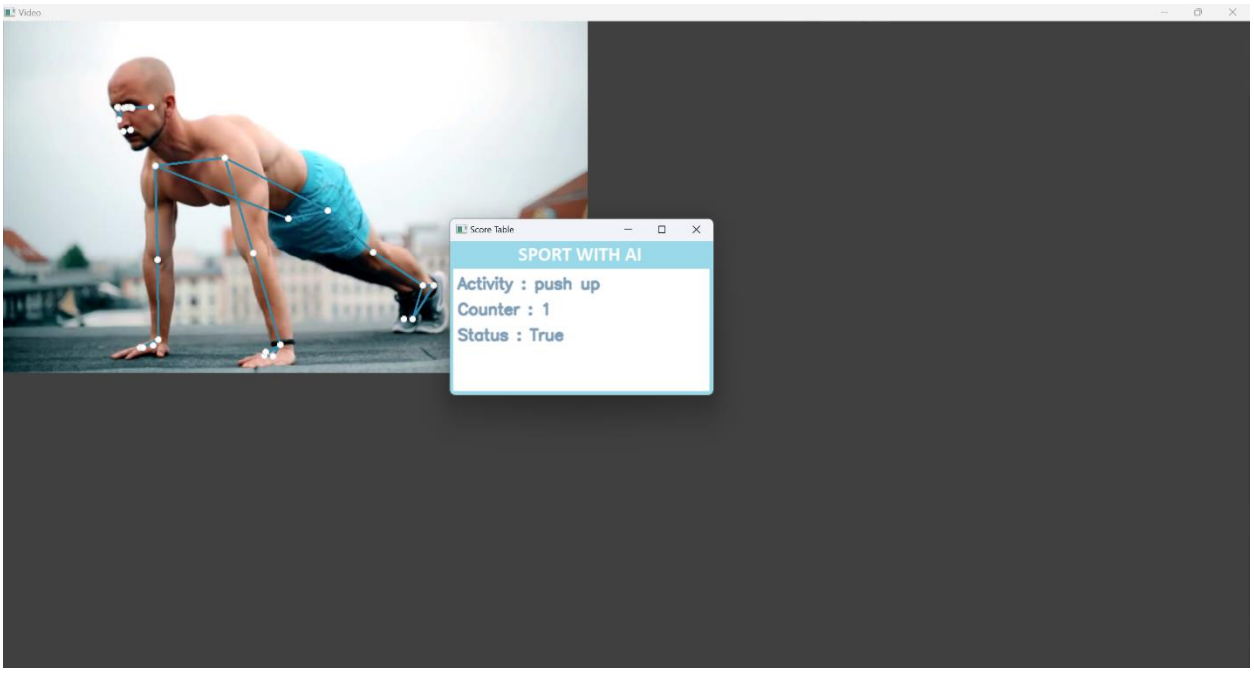
CHAPTER 4
TEST RESULTS/EXPERIMENT/VERIFICATION

4.1 RESULTS




AI for workout counting
Choose one
Real time
Train
Stop
AI for workout counting
For train
Squat
Push_up
Stop
By
Sri Dhanush
URK21CS1039


Video
Score Table
SPORT WITH AI
Activity : squat
Counter : 2
Status : True



CHAPTER 5

CONCLUSION

- This project is very helpful for maintain our health where it seems to be no gym in the area

- It provides precicely information that how many sets of workout you have done

- It doesn’t need any special trainer to watch the media pipe helps to take up the data and the tensor flow helps to organize the data and the numpy helps to store it and the tkinter module helps to show the result in an organized manner

THANK YOU