YOGA POSE DETECTION

PROJECT WORK REPORT

Submitted by

Saranya M

21BIT036

Under the Guidance of

Dr Rajdeepa B MCA MPhil PhD MSc(Yoga)NET

Associate Professor & Head of the Department

Department of Information Technology

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

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

of Bharathiar University

DEPARTMENT OF INFORMATION TECHNOLOGY

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous College-Affiliated to Bharathiar University

Accredited with 'A^{++}' grade by NAAC (4th Cycle)

College with Potential for Excellence

(Status Awarded by the UGC)

STAR College Status Awarded by DBT - MST

An ISO 9001:2015 Certified Institution

Coimbatore -641 014

APRIL 2024

DEPARTMENT OF INFORMATION TECHNOLOGY

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous College-Affiliated to Bharathiar University

Accredited with A^{++} grade by NAAC (4^{th} Cycle)

College with Potential for Excellence

(Status Awarded by the UGC)

STAR College Status Awarded by DBT - MST

An ISO 9001:2015 Certified Institution

Civil Aerodrome Post

Coimbatore -641 014

CERTIFICATE

This is to certify that this Project work entitled **"YOGA POSE DETECTION"** is a bonafide record of work done by **Saranya M (21BIT036)** in partial fulfillment of the requirements for the award of Degree of **Bachelor of Science in Information Technology** of Bharathiar University.

Faculty Guide **Head of the Department**

Submitted for Viva-Voce Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

I **SARANYA M (21BIT036)** hereby declare that this Project work entitled **"YOGA POSE DETECTION"** is submitted to PSG College of Arts & Science (Autonomous), Coimbatore in partial fulfillment for the award of Bachelor of Science in Information Technology, is a record of original work done by me under the supervision and guidance of **Dr Rajdeepa B MCA MPhil PhD MSc(Yoga)NET** Associate Professor and Head of the Department, Department of Information Technology, PSG College of Arts & Science, Coimbatore.

This Project work has not been submitted by me for the award of any other Degree/ Diploma/ Associate ship/ Fellowship or any other similar degree to any other university.

PLACE : Coimbatore **Saranya M**

DATE : **21BIT036**

ACKNOWLEDGEMENT

My venture stands imperfect without dedicating my gratitude to a few people who have contributed a lot towards the victorious completion for my project work.

I would like to thank **Shri Gopalakrishnan L Managing Trustee PSG & Sons Charities** for providing me with a prospect and surroundings that made the work possible.

I take this opportunity to express my deep sense of gratitude to **Dr Kannaian T** Secretary of PSG College of Arts & Science, Coimbatore for permitting and doing the needful towards the successful completion of this project.

I express my deep sense of gratitude and sincere thanks to **Dr Brindha D, Principal, MSc MPhil PhD MA (Yoga)** for her valuable advice and concern on students.

I am very thankful to **Dr Anguraj A MSc MPhil PhD Vice Principal(Academics)**, **Dr Umarani M MCom MPhil PhD Vice Principal (Student Affairs)** for their support.

I kindly and sincerely thank **Dr Rajdeepa B MCA MPhil PhD MSc(Yoga)NET Associate Professor and Head of the Department of Information Technology and my Project guide** for her whole hearted help to complete this project successfully by giving valuable suggestions.

This note of acknowledgement will be incomplete without paying my heartfelt devotion to my parents, my friends and other people, for their blessings, encouragement, financial support and the patience, without which it would have been impossible for me to complete the job.

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous College-Affiliated to Bharathiar University

Accredited with A^{++} grade by NAAC (4^{th} Cycle)

College with Potential for Excellence

(Status Awarded by the UGC)

STAR College Status Awarded by DBT - MST

An ISO 9001:2015 Certified Institution

Civil Aerodrome Post

Coimbatore -641 014

CERTIFICATE

This is to certify that this Project work entitled **"YOGA POSE DETECTION"** is submitted to PSG College of Arts & Science (Autonomous) ,Coimbatore, Affiliated to Bharathiar University in partial fulfillment for the award of Bachelor of Science in Information Technology, is a record of original work done by **Saranya M (21BIT036)** during December 2023 to April 2024 of her study in the Department of Information Technology, PSG College of Arts & Science affiliated to Bharathiar University under my supervision and guidance. This Project work has not formed the basis for the award of any other Degree / Diploma / Associate ship / Fellowship or any other similar degree to any other University.

Signature of the Guide
Dr. B. Rajdeepa
Associate Professor & HoD
Department of Information Technology
PSG College of Arts & Science
Coimbatore

Signature of the HOD
Dr. B. Rajdeepa
Associate Professor & HoD
Department of Information Technology
PSG College of Arts & Science
Coimbatore

SYNOPSIS

This project focuses on the detection of Yoga Pose system using Python. The system aims to accurately identify and analyze yoga poses from webcam by using various kind of libraries which is used to analyze and find out the yoga poses. The process involves collecting a diverse dataset of yoga poses. This dataset should include videos of individuals performing various yoga poses, with each pose labeled for training purposes.

Yoga pose detection using Python involves utilizing computer vision techniques to analyze images or video frames and identify specific yoga poses being performed by individuals. This process typically begins with gathering a dataset of images or videos containing various yoga poses along with their corresponding labels.

A machine learning model, trained on a diverse dataset of yoga poses, is employed to classify and track the user's current pose. The system is designed to be user-friendly, requiring minimal setup and no specialized equipment. We use Pose estimation libraries like Tensorflow, keras and mediapipe to extract the key points or joints of a person on a screen. These libraries help in understanding the body’s pose and positioning.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

Yoga has gained popularity as a result of the modern lifestyle's increased stress. The scope of this project is to develop a real-time yoga detection system using Convolutional Neural Network (CNN), OpenPose in Python. The system should be able to accurately identify the type of pose and action (such as forward bends, backward bends, twists, etc.) and the body part performing the action (such as arms, legs, torso, etc).

Python is chosen as the programming language for its versatility and extensive libraries in the field of machine learning. Libraries such as TensorFlow, OpenCV and mediapipe can aid in image processing and pose estimation. The proposed system not only aids practitioners in maintaining proper alignment but also offers a personalized and interactive experience.

By leveraging technology to monitor and correct yoga poses, individuals can enhance their home practice, reduce the risk of injury, and deepen their understanding of yoga postures.

Gathering a diverse dataset of images or videos containing individuals performing various yoga poses. These images or videos should cover a wide range of angles, lighting conditions, and body types to ensure the model's robustness. Annotating the dataset with labels indicating the specific yoga pose being performed in each image or frame. This step is crucial for supervised learning, where the model learns to associate visual features with corresponding labels during training.

1.2 Module description

- **Data Acquisition**
- **Data pre-processing and Feature extraction**
- **Gesture Classification**

Data Acquisition:

The different approaches to acquire data about the body gesture can be done in the following ways:

It uses electromechanical devices to provide exact body configuration, and position. Different position-based approaches can be used to extract information. But it is expensive and not user friendly. In vision-based methods, the computer webcam is the input device for observing the information of body poses.

The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices, thereby reducing costs. The main challenge of vision-based yoga pose detection ranges from coping with the large variability of the yoga pose appearance due to a huge number of body movements, to different skin-color possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.

In this project we need to create our own dataset, we need to collect images or videos of people performing yoga poses. Here we can use a camera or smartphone to record videos or capture images. It's essential to ensure that the data covers a diverse range of poses, lighting conditions, backgrounds, and individuals to make the model more robust.

In this Yoga Pose Detection we need to label each image or frame in the video with the corresponding yoga pose.

Data pre-processing and Feature extraction:

In this approach for yoga pose detection, initially we detect body pose from image that is acquired by webcam and for detecting a yoga pose, we used media pipe library which is used for image processing. So, after finding the yoga pose from image we get the region of interest (Roi)

then we cropped that image and convert the image to gray image using OpenCV library after we applied the gaussian blur.

The filter can be easily applied using open computer vision library also known as OpenCV. Then we converted the gray image to binary image using threshold and Adaptive threshold methods.

This project consists of images, we first need to load the images into your Python environment. We use libraries like OpenCV (cv2) or PIL (Python Imaging Library) for this purpose.

Data Preprocessing:

Before training the CNN model, preprocess the dataset. This involves:

Resizing: Resize images to a fixed size suitable for the input of the CNN model. This ensures uniformity across the dataset and helps in efficient computation.

Normalization: Normalize pixel values to a range between 0 and 1. This step ensures that the model's learning process is stable and helps in faster convergence.

Gesture Classification:

Convolutional Neural Network (CNN)

CNN is a class of neural networks that are highly useful in solving computer vision problems. They make use of a filter/kernel to scan through the entire pixel values of the image and make computations by setting appropriate weights to enable detection of a specific feature.

CNN is equipped with layers like convolution layer, max pooling layer, flatten layer, dense layer, dropout layer and a fully connected neural network layer. These layers together make a very powerful tool that can identify features in an image. The starting layers detect low level features that gradually begin to detect more complex higher-level features.

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

Model Training:

Train a CNN model using the preprocessed dataset. Steps involved in model training include:

- Model Architecture: Define the architecture of the CNN model. You can start with a simple architecture, such as a stack of convolutional layers followed by fully connected layers.
- Compile the Model: Compile the model with appropriate loss function, optimizer, and evaluation metric.
- Training: Train the model using the training dataset. Adjust hyper parameters like learning rate, batch size, and number of epochs to optimize the model's performance.
- Model Evaluation: Evaluate the trained model's performance on the validation set. Monitor metrics like accuracy, precision, recall, and F1-score to assess the model's performance

2. SYSTEM SPECIFICATIONS

2.1 Hardware Configurations:

The minimum specifications that are only suggested for yoga pose detection is a system with the in-built camera or any webcam of better quality and a system that could accept any kind of web-camera.

- System: DTJA08M
- Processor: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz
- Ram: 4.00GB
- Web cam

2.2 Software Specifications:

Software requirements includes the program language that we use in our project and the software that we do use in our project, such a way software that are required in this project are:

- Operating System: Windows 8 and Above
- IDE: VisualStudio Code
- Programming Language: Python
- Software used: Python 3.8.10
- Python libraries: OpenCV, NumPy, Keras,mediapipe,Tensorflow

2.3 Software descriptions

Python:

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C. Python is widely used in bigger organizations because of its multiple programming paradigms. They usually involve imperative and object-oriented functional programming. It has a comprehensive and large standard library that has automatic memory management and dynamic features. The software development companies prefer Python language because of its versatile features and fewer programming codes. Nearly 14% of the programmer use it on the operating systems like UNIX, Linux, Windows and Mac OS.

Model Training:

- Selecting an appropriate machine learning architecture for the task, typically a convolutional neural network (CNN), which excels at learning hierarchical features from images.
- Splitting the dataset into training, validation, and testing sets to evaluate the model's performance accurately.
- Training the CNN model on the labeled dataset using a suitable optimization algorithm (e.g., stochastic gradient descent) and loss function (e.g., categorical cross-entropy) to minimize prediction errors and improve accuracy.

Model Evaluation:

Evaluating the trained model's performance on the validation and test sets to assess its accuracy, precision, recall, and other relevant metrics. This step helps determine if the model generalizes well to unseen data and can reliably classify yoga poses.

PACKAGES:

NumPy

Python NumPy is a general-purpose array processing package that provides tools for handling n-dimensional arrays. It provides various computing tools such as comprehensive mathematical functions, linear algebra routines. NumPy provides both the flexibility of Python and the speed of well-optimized compiled C code. Its easy-to-use syntax makes it highly accessible and productive for programmers from any background.

After capturing frames from the webcam using OpenCV, we need to convert them into NumPy arrays. Preprocess these NumPy arrays before feeding them into the trained CNN model for inference.

OpenCV:

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To identify image pattern and its various features we use vector space and perform mathematical operations on these features.

OpenCV offers a wide range of functions for image processing tasks such as resizing, normalization, and color space conversion. These functions are useful for preprocessing webcam frames before feeding them into the pose detection model.

MEDIAPIPE:

Mediapipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. This cross-platform Framework works on Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano. Mediapipe is an open-source library developed by Google that provides a set of pre-trained machine learning models and tools for building applications that involve various forms of media data, including audio, video, and image processing. It's particularly popular for its computer vision and machine learning

capabilities, making it useful for tasks like face detection, hand tracking, pose estimation, and more.

Mediapipe provides a pose detection model that can be used to detect key landmarks on a person's body, which is useful for yoga pose detection. Capture frames from the webcam in real-time, process them using the Mediapipe pose detection model, and analyze the detected landmarks to infer yoga poses.

TENSORFLOW:

TensorFlow is an end-to-end open-source platform for Machine Learning (ML), backed by a comprehensive yet flexible ecosystem of tools, libraries, and communities. It is that magic well that allows developers to build and deploy ML-powered applications easily. This free software library is used for a variety of tasks with a focus on the inference and training of deep neural networks.

Capture frames from the webcam in real-time, process them using the pre-trained pose estimation model, and analyze the detected keypoints to infer yoga poses.

KERAS:

Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools for working with image and text data to simplify programming in deep neural network area. Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.

3. SYSTEM ANALYSIS

3.1 Existing System

The systems often required a pre-recorded video or a series of images for offline analysis. While they provided valuable insights into pose accuracy, they might lack real-time feedback and user interaction.

Disadvantages

- Some systems may be optimized for a specific set of yoga poses, and they might not perform well when confronted with a wide variety of poses. This limitation can impact the system's applicability to diverse yoga practices.
- Occlusions, where parts of the body are temporarily hidden from the camera's view, can pose challenges for accurate pose detection. The system may fail to accurately estimate poses when certain body parts are obscured.
- Existing systems may struggle to accurately detect poses during dynamic and fast movements, common in certain styles of yoga. Rapid changes in body positions can lead to pose misclassification or delays in detection.

3.2 Proposed System

The primary purpose of this project is to assist individuals in enhancing their home yoga practice, helping them to maintain proper form and alignment without the need for a physical instructor. The project can serve as an educational tool for individuals learning yoga. A proposed system for yoga pose detection using Python would integrate various components to achieve accurate and real-time detection of yoga poses from images or video streams.

Utilizing pose estimation algorithms, such as OpenPose or PoseNet, would enable the detection of crucial body landmarks in each image or frame, providing valuable input features alongside the raw image data.

Objectives of proposed system

- Utilize a standard webcam as the input device for capturing the user's live video feed
- Incorporate dynamic pose templates that adapt to variations in individual body shapes and sizes. This ensures a personalized experience, accommodating different levels of flexibility and expertise.
- Provide educational resources within the system, such as links to instructional videos, pose descriptions, and tips for improving yoga practice. This enhances the educational aspect of the system. Provides a natural interaction between humans and computers without the use of any extra devices, thereby reducing costs.
- Split the annotated dataset into training, validation, and test sets.
- Deploy the trained model to perform real-time yoga pose detection on live video streams or webcam feeds.
- Implement efficient algorithms for pose estimation, and model inference to achieve low-latency performance.

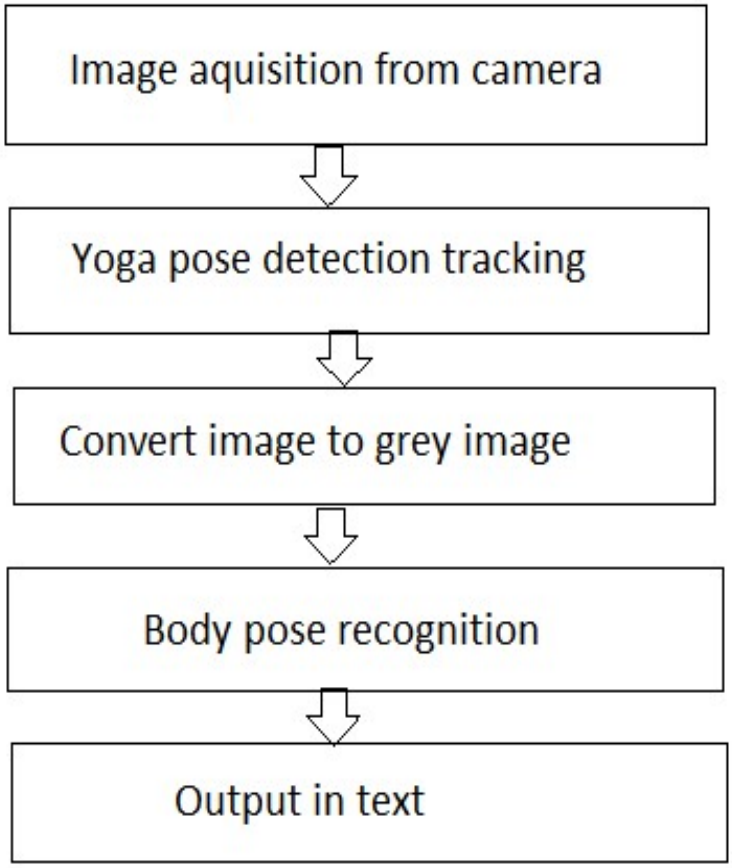
4. SYSTEM DESIGN

4.1 System flow diagram

A system flow diagram is a visual representation of the flow of data or processes within a system. It typically consists of a series of interconnected shapes that represent various components of the system, along with arrows that indicate the direction of data or process flow. The purpose of a system flow diagram is to provide a high-level overview of the system and its various components, as well as the relationships between them. This can be useful for identifying potential areas of improvement or optimization within the system.

This project collects the dataset from the webcam, processes it, convert the image into the grayscale image. So when user performs a pose, it display the name of the yoga pose

The system flow diagram for yoga pose detection is



4.2 Input Design

All images used for training and testing must be in the same resolution. The system must have access to a large dataset of images of yoga poses. The system should be able to handle different types of Pose.

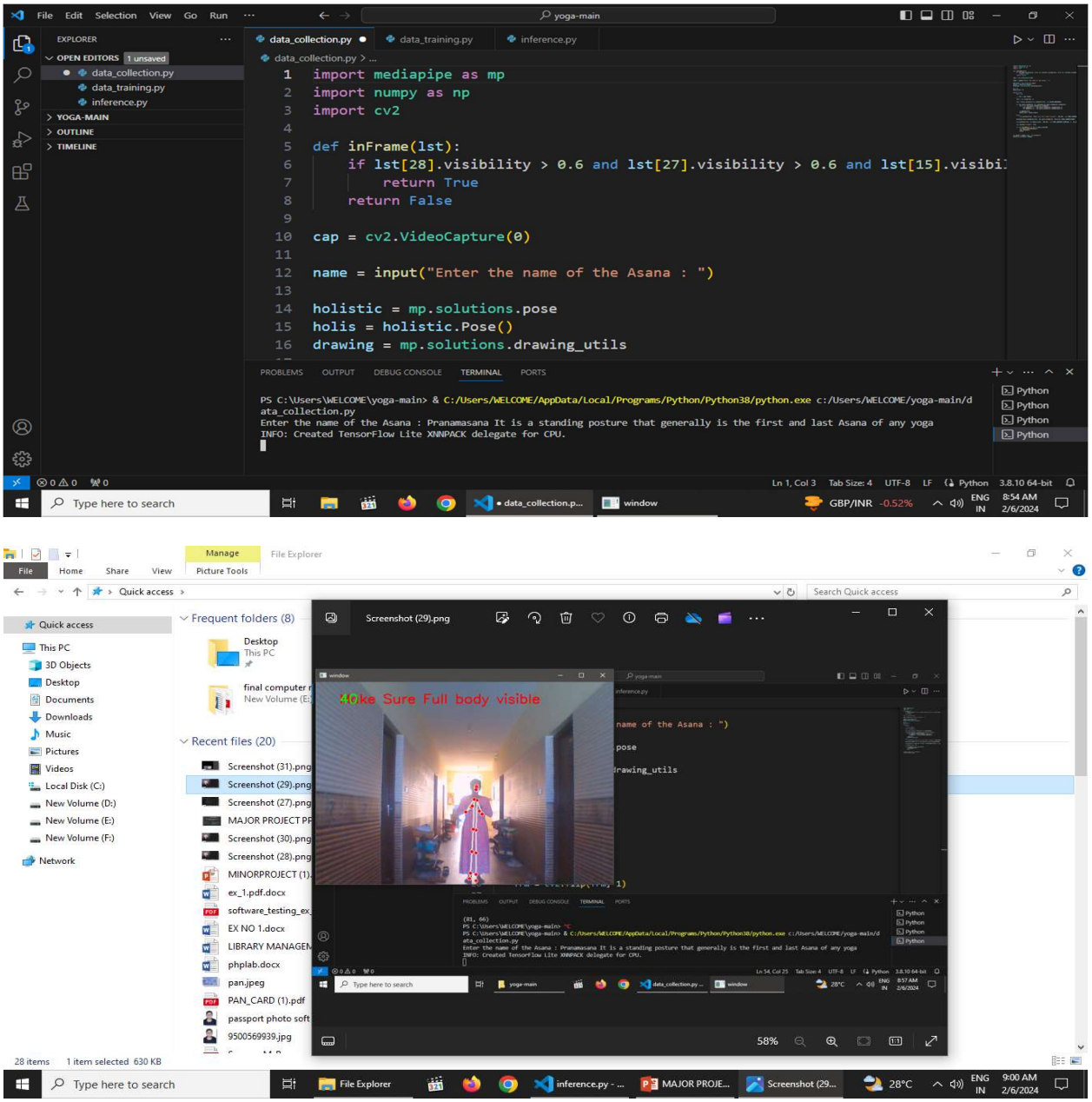


Fig: input code

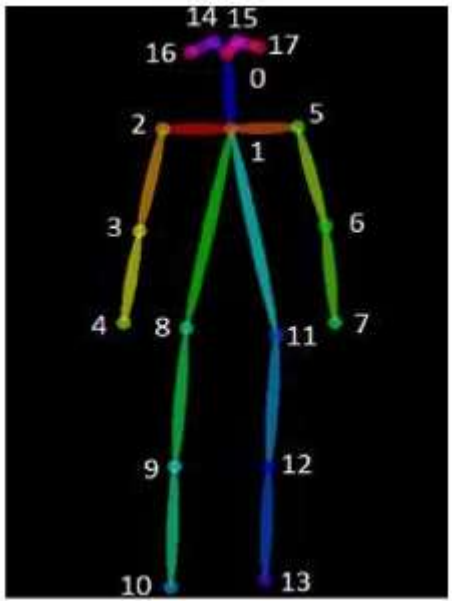


Fig: keypoints

Fig: skeleton image

4.3 Output Design

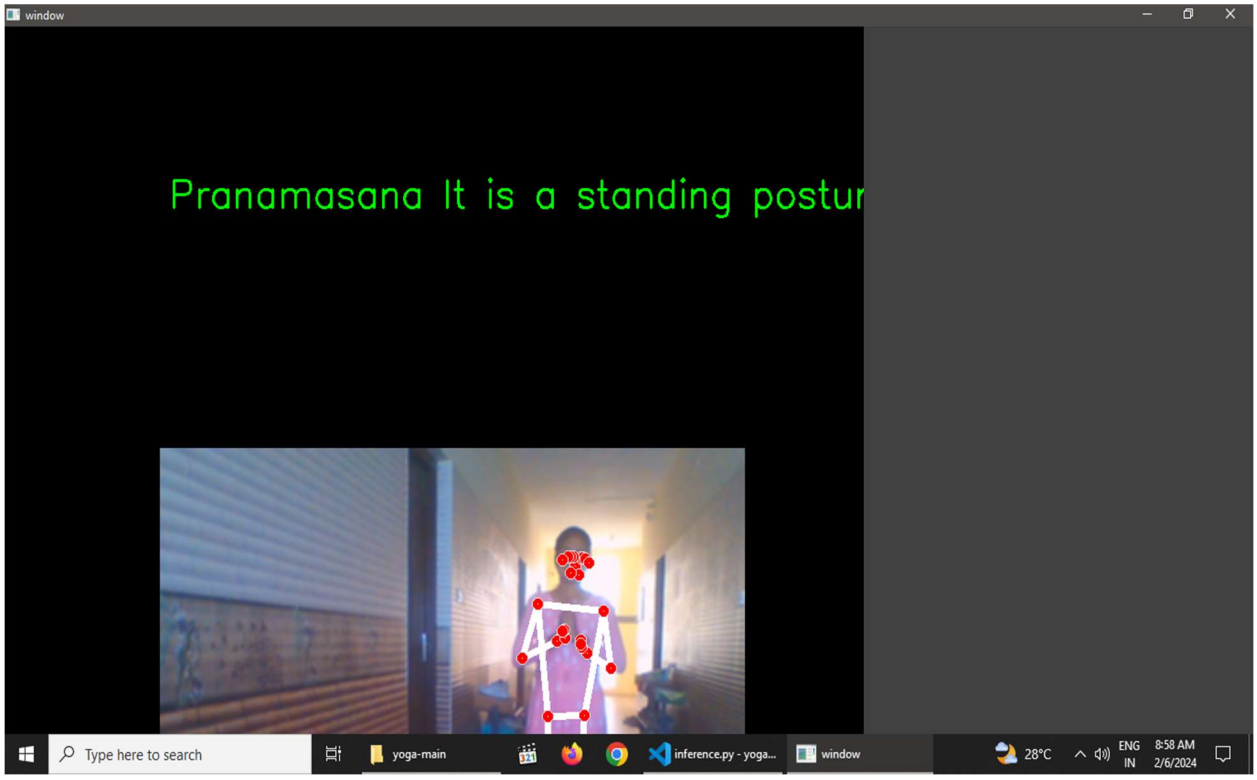


Fig: Output image

5. SYSTEM TESTING & IMPLEMENTATION

System testing is the stage of implementation that is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. System testing makes logical assumption that if all the parts of the system are correct, then the goal will be successfully achieved. System testing involves user training system testing and successful running of the developed proposed system. The user tests the developed system and changes are made per their needs. The testing phase involves the testing of developed system using various kinds of data. While testing, errors are noted and the corrections are made. The corrections are also noted for the future use.

1. Unit tests

Unit testing involves testing individual components or modules of the Object detection, Tracking and Alert system to ensure that they are functioning as intended. This type of testing typically focuses on testing the functionality of the system's algorithms and verifying that they produce the expected outputs for a given set of inputs. Unit tests are essential for detecting defects in the system's codebase and ensuring that individual components function correctly before they are integrated into the larger system.

Unit tests could be designed to validate the pose estimation algorithm's functionality, ensuring that it accurately detects key body landmarks (e.g., joints) in input images.

2. User acceptance testing

User acceptance testing is conducted to evaluate the system's ability to meet end-user needs and expectations. This testing typically involves creating real-world scenarios and testing how users interact with the system in those scenarios. The goal is to ensure that the system is intuitive and easy to use and that it meets the users' functional requirements.

Users would test the accuracy and reliability of the yoga pose detection system by providing input images or video streams containing various yoga poses.

3. Usability testing

Usability testing is conducted to evaluate the Object detection, Tracking and Alert system's user interface and overall user experience. This testing typically involves observing users as they

interact with the system and gathering feedback on how easy the system is to use and how well it meets their needs. The goal of usability testing is to ensure that the system is intuitive and easy to use, and that it provides a positive user experience.

Usability testing involves assessing the clarity, intuitiveness, and effectiveness of the application's user interface. Users interact with the UI to perform tasks related to yoga pose detection

4. Integration Testing:

Integrating testing is a systematic technique for constructing the software architecture to conduct errors associated with interfacing .Top-down integration testing ia a incremental approach to construction of software architecture Module are integrated by moving downward through the control hierarchy ,beginning with the main control module.

Integration testing ensures that the image processing module properly preprocesses input images or video frames before passing them to the pose estimation module.

6. CONCLUSION

The project is a simple demonstration of how CNN can be used to solve computer vision problems with an extremely high degree of accuracy. The yoga pose detection is obtained which has an accuracy of 95%. The project can be extended to other body poses by building the corresponding dataset and training the CNN. The main objective has been achieved, that is, the need for an interpreter has been eliminated.

If this issue is encountered, we need to either reset the histogram or look for places with suitable lighting conditions. The other issue that people might face is regarding their proficiency in knowing the ASL gestures. Bad gesture postures will not yield correct prediction.

7. SCOPE FOR FUTURE ENHANCEMENT

- In the future, the system can be further developed to include additional postures and features to improve accuracy and provide more comprehensive feedback.
- Additionally, in future this system is also able to produce a beep sound if the posture is wrong. This system has the potential to be used for fitness purposes, such as providing feedback to yoga practitioners, helping to improve posture and assisting in the development of strength and flexibility.
- The system can also be used in a variety of health and wellness applications such as physical therapy, post-operative rehabilitation, and injury prevention.

8. BIBLIOGRAPHY

Books Referred:

[1] Joseph Redmon and Anelia Angelova, Real-Time Grasp Detection Using Convolutional Neural Networks (ICRA), 2015.

[2] A. Quattoni, and A.Torralba. Recognizing Indoor Scenes. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

Website Referred:

- https://www.geeksforgeeks.org/libraries-in-python/
- Youtube- https://www.youtube.com/watch?v=sIRqrwZnuHE
- Youtube- https://youtu.be/u8APxuCICkA?si=Tzwhs06fJbJr9pQn
- Youtube- https://youtu.be/kCMJxOSiWrg?si=jBPvfC0vUgxDkxRX
- Youtube- https://youtu.be/n0hBK3_QT9A?si=g0Vu-rxCdLCrQasD

9. APPENDICES

A. Sample Code:

Data_collection.py:

```
import mediapipe as mp
import numpy as np
import cv2
def inFrame(lst):
        if lst[28].visibility > 0.6 and lst[27].visibility > 0.6 and
lst[15].visibility>0.6 and lst[16].visibility>0.6:
                return True
                return False
        cap = cv2.VideoCapture(0)
name = input("Enter the name of the Asana : ")
holistic = mp.solutions.pose
holis = holistic.Pose()
drawing = mp.solutions.drawing_utils
X = []
data_size = 0
while True:
lst = []
_, frm = cap.read()
frm = cv2.flip(frm, 1)
res= holis.process(cv2.cvtColor(frm, cv2.COLOR_BGR2RGB))
if res.pose_landmarks and inFrame(res.pose_landmarks.landmark):
```

```
                for i in res.pose_landmarks.landmark:
                        lst.append(i.x - res.pose_landmarks.landmark[0].x)
                        lst.append(i.y - res.pose_landmarks.landmark[0].y)

                X.append(lst)
                data_size = data_size+1

        else:
                cv2.putText(frm, "Make Sure Full body visible", (50,50),
    cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255),2)

        drawing.draw_landmarks(frm, res.pose_landmarks,
    holistic.POSE_CONNECTIONS)
    cv2.putText(frm,str(data_size),(50,50),cv2.FONT_HERSHEY_SIMPLEX,
1, (0,255,0),2)
    cv2.imshow("window", frm)
        if cv2.waitKey(1) == 27 or data_size>80:
                cv2.destroyAllWindows()
                cap.release()
                break
    np.save(f"{name}.npy", np.array(X))
    print(np.array(X).shape)
```

Data_training.py:

```
import os

import numpy as np

import cv2

from tensorflow.keras.utils import to_categorical

from keras.layers import Input, Dense

from keras.models import Model

 is_init = False

size = -1

label = []

dictionary = {}

c = 0

for i in os.listdir():

        if i.split(".")[-1] == "npy" and not(i.split(".")[0] == "labels"):

                if not(is_init):

                        is_init = True

                        X = np.load(i)

                        size = X.shape[0]

                        y = np.array([i.split('.')[0]]*size).reshape(-1,1)

                else:

                X = np.concatenate((X, np.load(i)))

                y=np.concatenate((y, np.array([i.split('.')[0]]*size).reshape(-1,1)))


                label.append(i.split('.')[0])
```

```
                dictionary[i.split('.')[0]] = c
                c = c+1
for i in range(y.shape[0]):
        y[i, 0] = dictionary[y[i, 0]]
y = np.array(y, dtype="int32")
y = to_categorical(y)
X_new = X.copy()
y_new = y.copy()
counter = 0
cnt = np.arange(X.shape[0])
np.random.shuffle(cnt)
for i in cnt:
        X_new[counter] = X[i]
        y_new[counter] = y[i]
        counter = counter + 1
ip = Input(shape=(X.shape[1]))
m = Dense(128, activation="tanh")(ip)
m = Dense(64, activation="tanh")(m)
op = Dense(y.shape[1], activation="softmax")(m)
model = Model(inputs=ip, outputs=op)
model.compile(optimizer='rmsprop',loss="categorical_crossentropy",
metrics=['acc'])
model.fit(X_new, y_new, epochs=80)
```

```
model.save("model.h5")
np.save("labels.npy", np.array(label))
```

Inference.py:

```
import cv2
import numpy as np
import mediapipe as mp
from keras.models import load_model
def inFrame(lst):
    if lst[28].visibility > 0.6 and lst[27].visibility > 0.6 and lst[15].visibility>0.6
and lst[16].visibility>0.6:
        return True
    return False
model  = load_model("model.h5")
label = np.load("labels.npy")
holistic = mp.solutions.pose
holis = holistic.Pose()
drawing = mp.solutions.drawing_utils
cap = cv2.VideoCapture(0)
while True:
    lst = [] _, frm = cap.read()
    window = np.zeros((940,940,3), dtype="uint8")
```

```
        frm = cv2.flip(frm, 1)
        res = holis.process(cv2.cvtColor(frm, cv2.COLOR_BGR2RGB))
        frm = cv2.blur(frm, (4,4))
        if res.pose_landmarks and inFrame(res.pose_landmarks.landmark):
                for i in res.pose_landmarks.landmark:
                        lst.append(i.x - res.pose_landmarks.landmark[0].x)
                        lst.append(i.y - res.pose_landmarks.landmark[0].y)
                lst = np.array(lst).reshape(1,-1)
                p = model.predict(lst)
                pred = label[np.argmax(p)]
                if p[0][np.argmax(p)] > 0.75:
                        cv2.putText(window, pred , (180,180),cv2.FONT_ITALIC,
1.3, (0,255,0),2)
                else:
                        cv2.putText(window, "Asana is either wrong not trained" ,
(100,180),cv2.FONT_ITALIC, 1.8, (0,0,255),3)
        else:
                cv2.putText(frm, "Make Sure Full body visible", (100,450),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,0,255),3)
        drawing.draw_landmarks(frm,res.pose_landmarks,holistic.POSE_CONNE
CTIONS,connection_drawing_spec=drawing.DrawingSpec(color=(255,255,255),
thickness=6),landmark_drawing_spec=drawing.DrawingSpec(color=(0,0,255),
circle_radius=3, thickness=3))
```

```
window[420:900, 170:810, :] = cv2.resize(frm, (640, 480))
cv2.imshow("window", window)
if cv2.waitKey(1) == 27:
        cv2.destroyAllWindows()
        cap.release()
        break
```

B. Screenshots

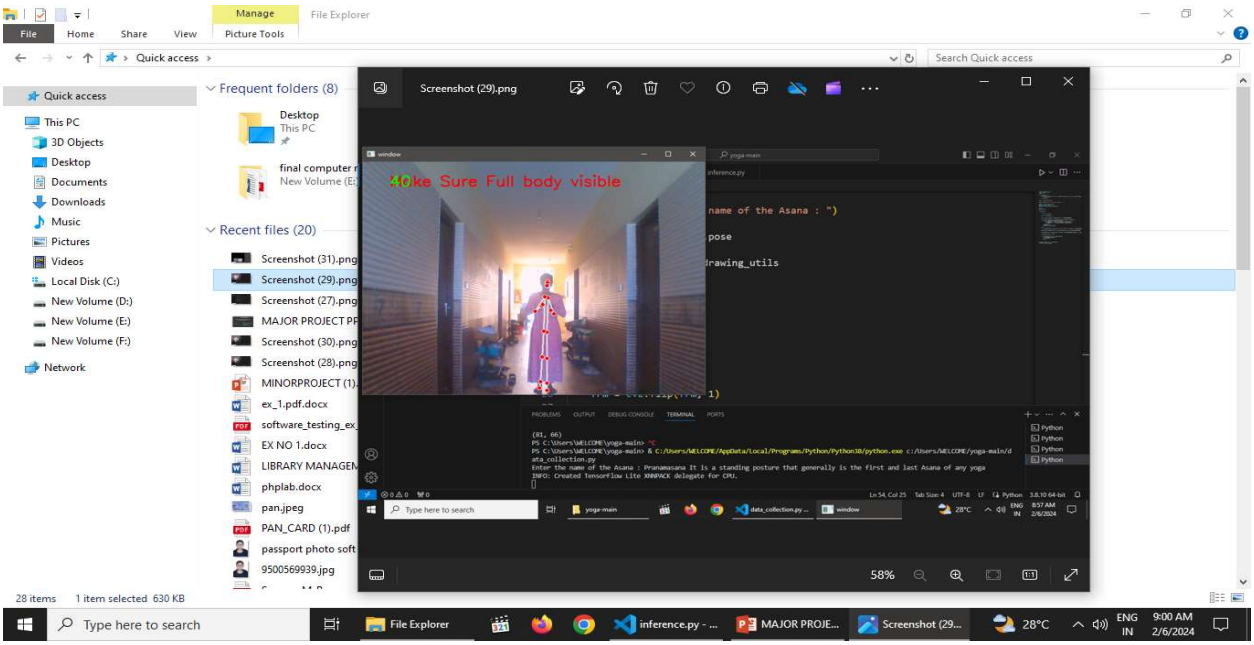


Fig: Collecting data from webcam

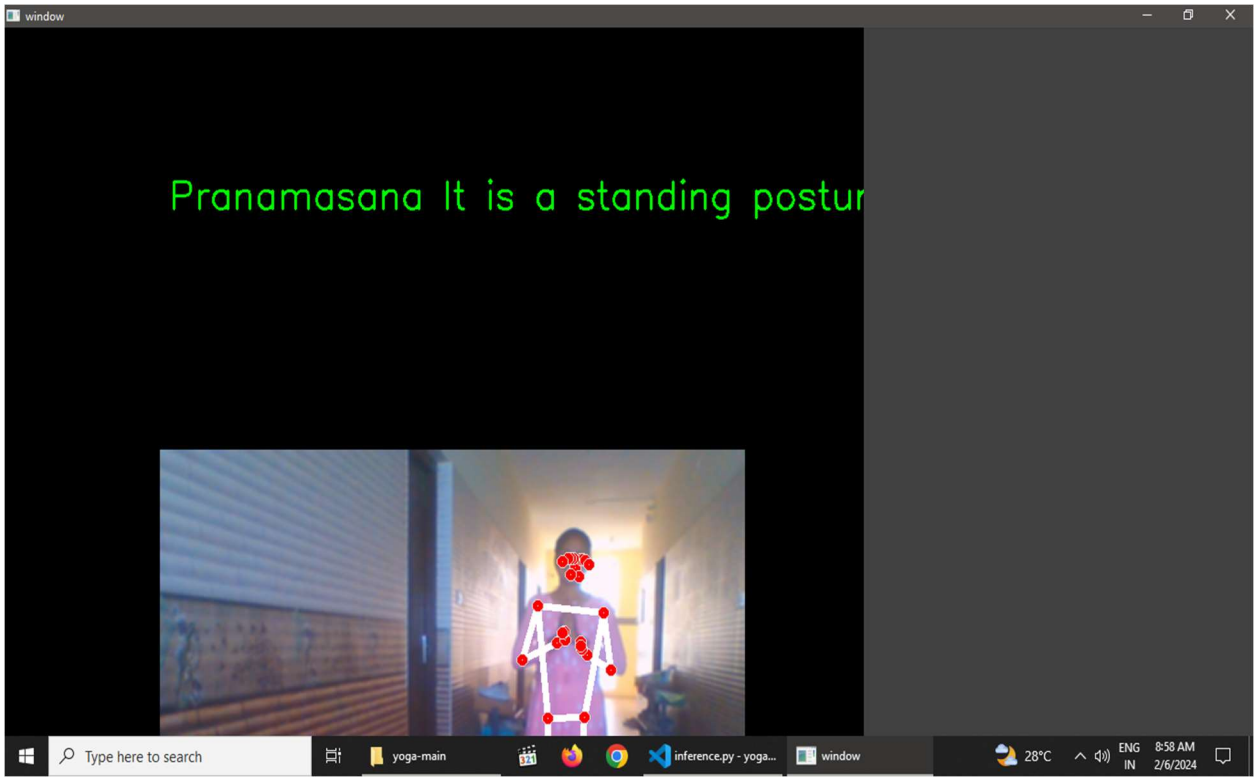


Fig: Output image

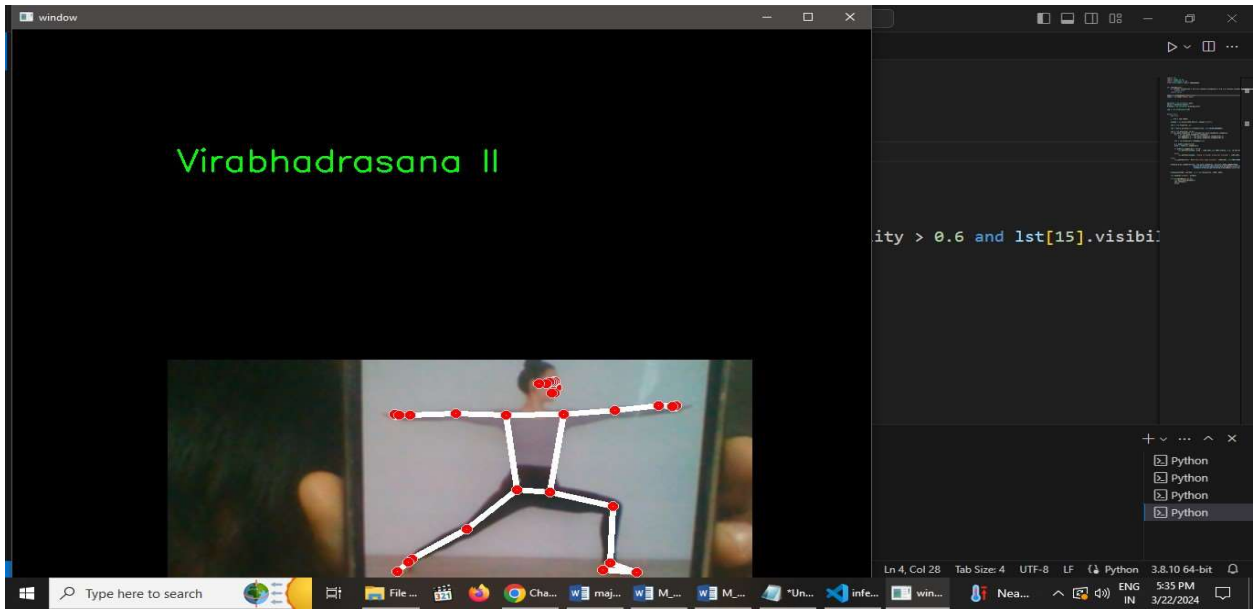


Fig: Output image(ii)

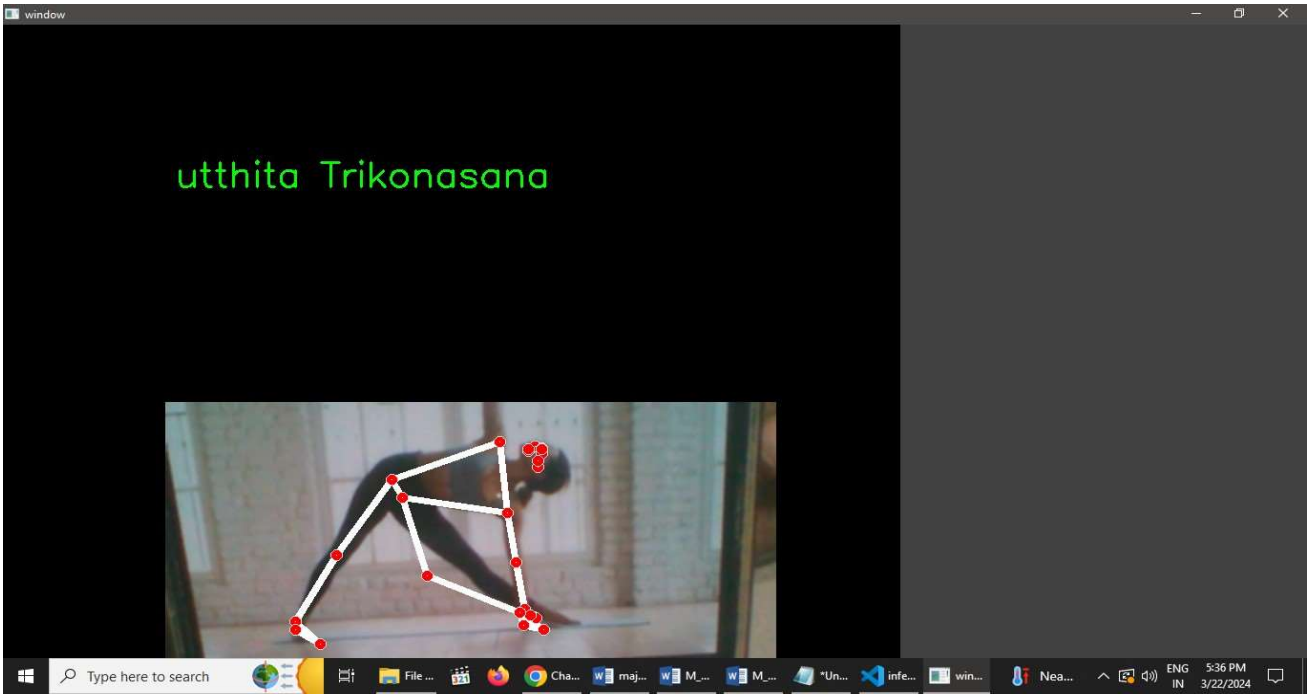


Fig: Output image(iii)

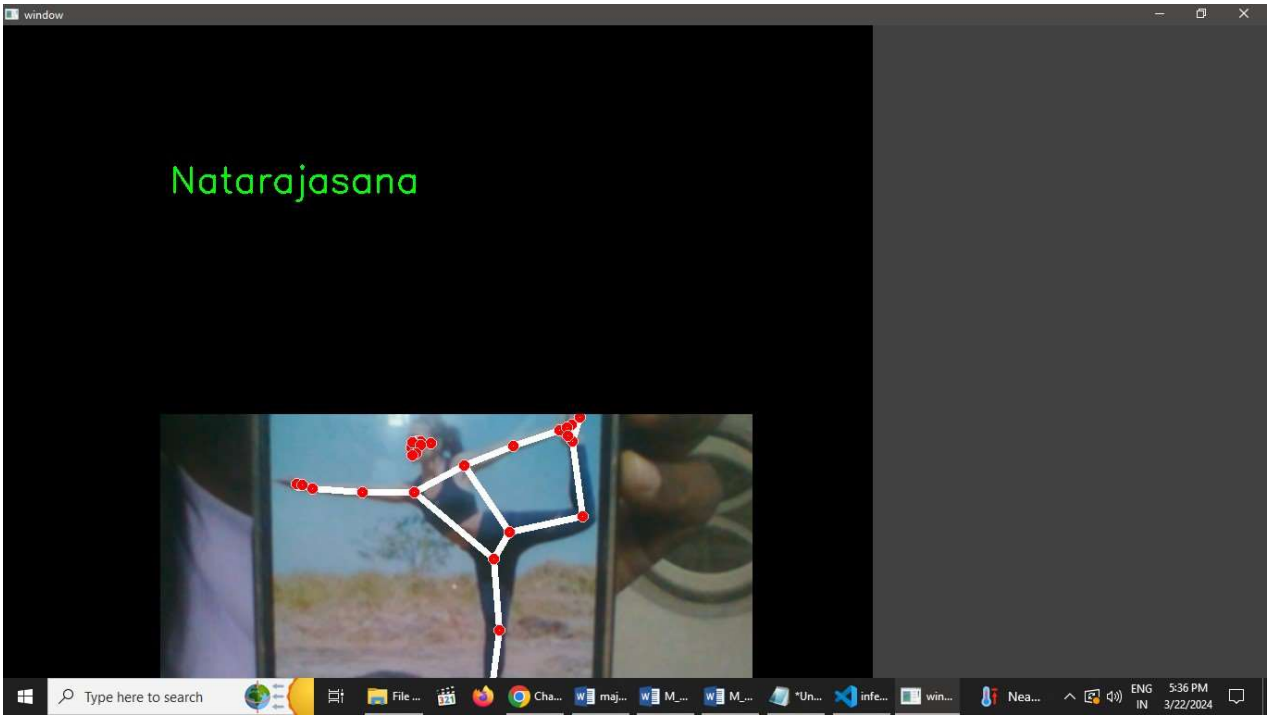


Fig: Output image(iv)

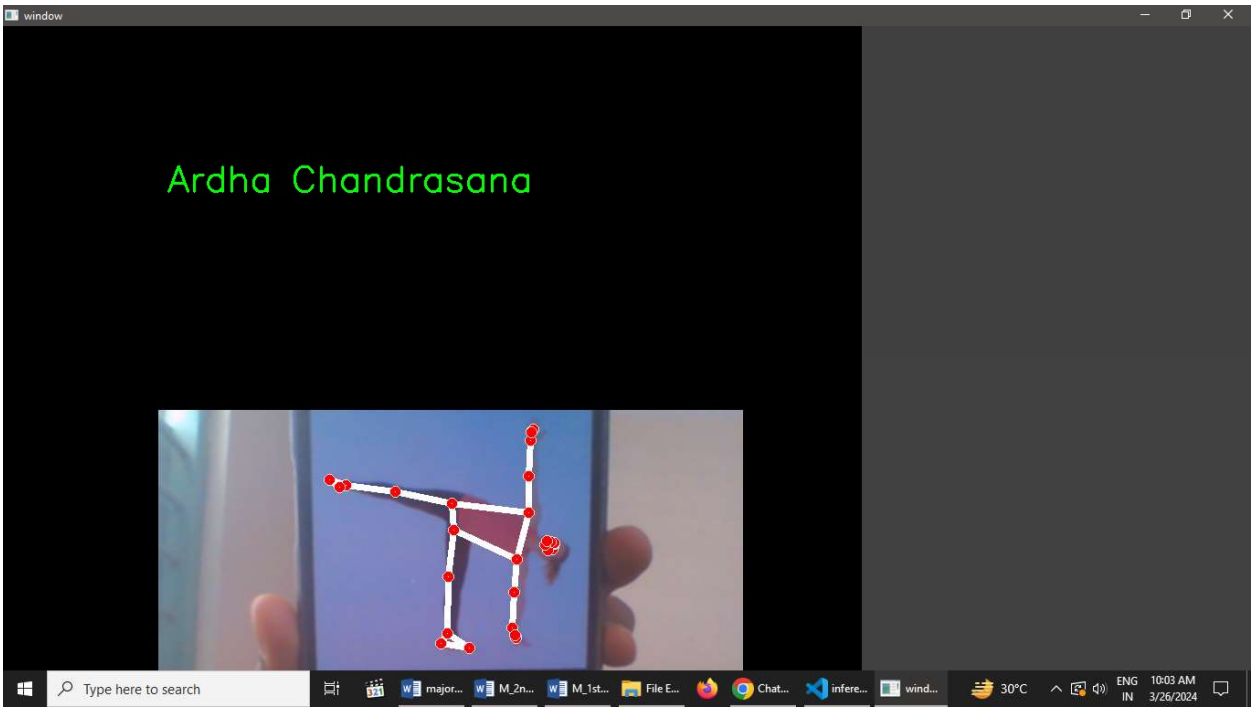


Fig: Output image(v)

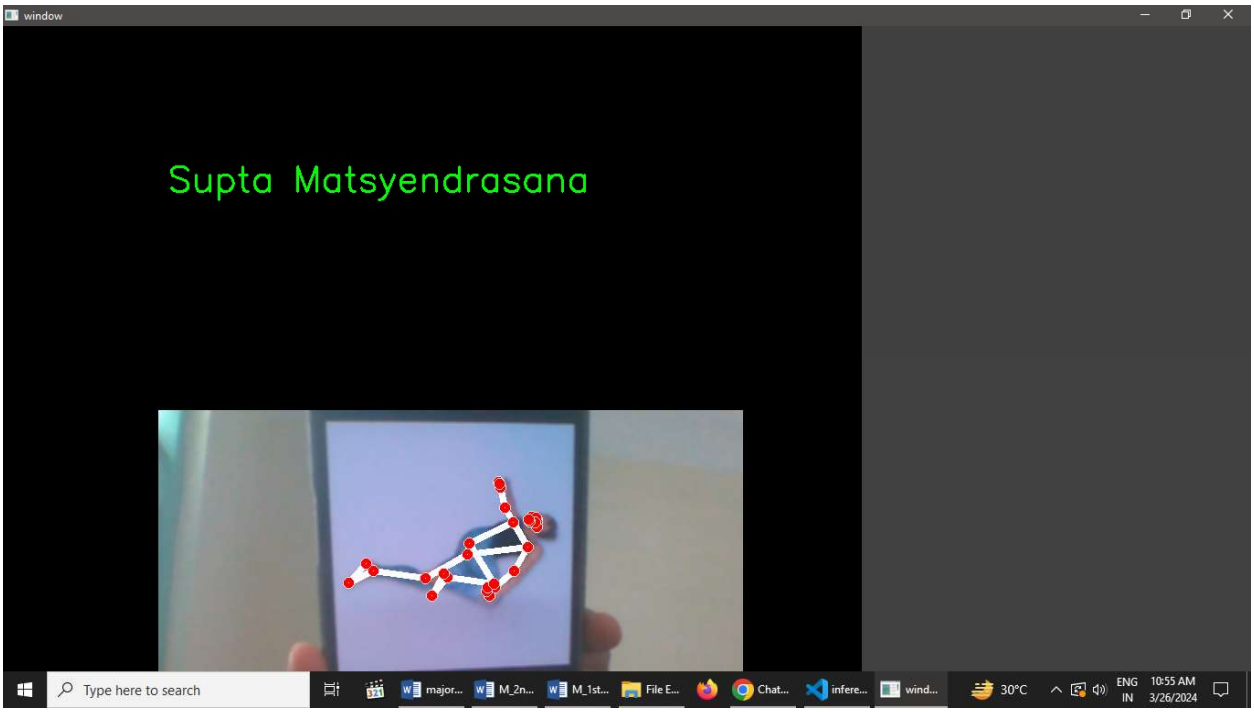


Fig: Output image(vi)

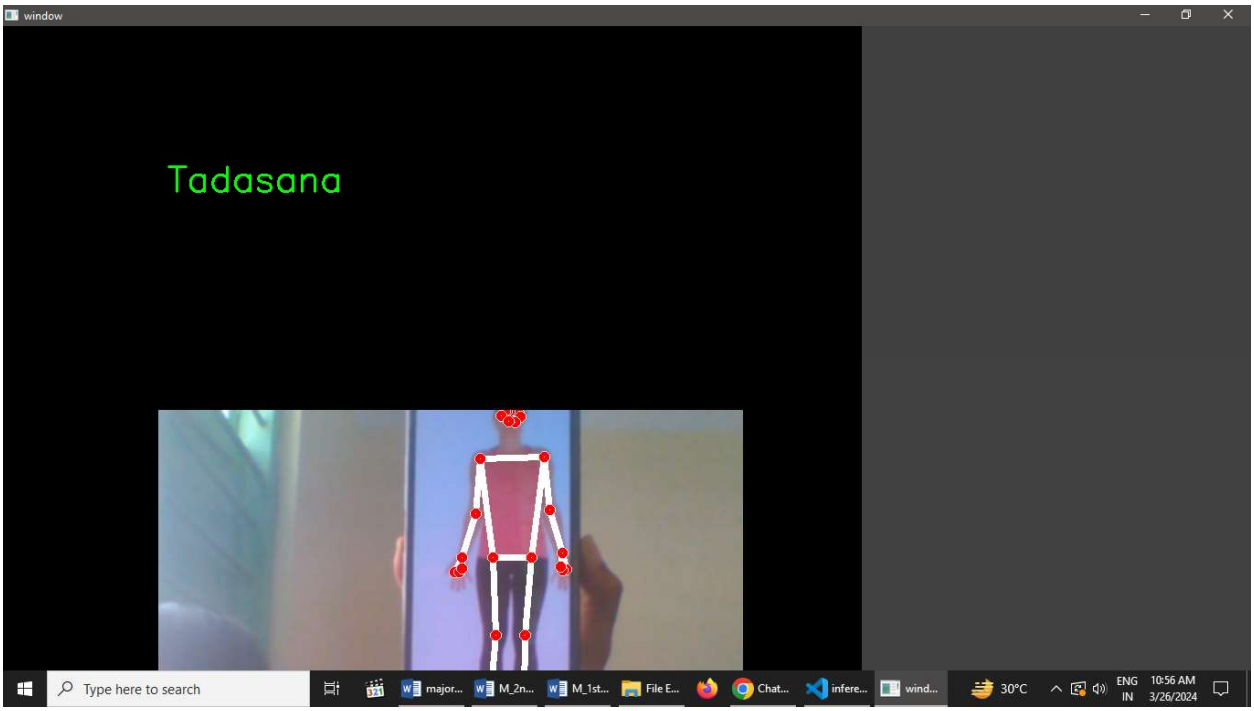


Fig: Output image(vii)

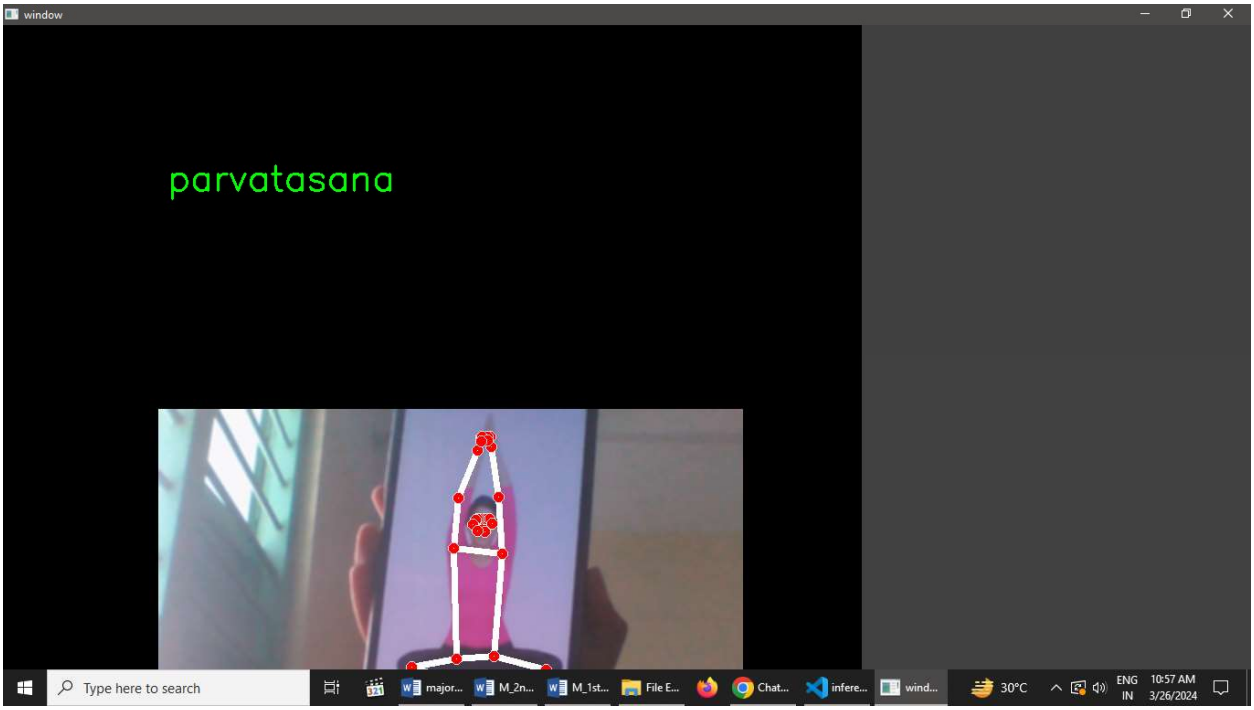


Fig: Output image(viii)