A

PROJECT REPORT

On

BODY POSTURE DETECTION

Submitted to the partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted by :
DHRITI NANDA (RA2011026030108)
SATYAM SINHA (RA2011026030102)
SHIVANSH GUPTA (RA2011026030111)

Supervised by :
Mrs. Geetanjali Tyagi
Assistant Professor

SRM Institute of Science and Technology
Delhi NCR Campus, Modinagar,
Ghaziabad (UP)-201204

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

BONAFIDE CERTIFICATE

Confirmed that this undertaking report named "**Body Posture Detection**" is the Bonafide work of "**Dhriti Nanda [RA2011026030108]", "Stayam Sinha [RA2011026030102]", "Shivansh Gupta [RA2011026030111]**" who did the task work under my watch. Ensured further, that apparently the work announced here doesn't frame some other venture report or paper based on which a degree or grant was presented on a prior event on this or some other up-and-comer.

SIGNATURE

Ms. Geetanjali Tyagi
(Guide)
ASSISTANT PROFESSOR
Dept. of Computer Science Engineering

SIGNATURE

Dr. Avneesh Vashistha

HEAD OF DEPARTMENT
Dept. of Computer Science Engineering

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

We would like to express our heartfelt appreciation to Mrs. Geetanjali Tyagi, Assistant Professor and Project Supervisor at SRM Institute of Science and Technology, Delhi-NCR Campus, Modinagar, for her invaluable insights and expertise in the subject matter, which motivated us to work diligently.

Our profound gratitude goes out to **Dr. Jitendra Singh** and **Mrs Abhilasha Singh,** Project Coordinators at SRM Institute of Science and Technology, Delhi-NCR Campus, Modinagar, for their enlightening guidance and skillful coordination, which served as a perpetual source of inspiration.

We would also like to extend our sincere thanks to **Dr. S. Vishwanathan**, Director of SRMInstitute of Science and Technology, Delhi-NCR Campus, Modinagar, for his unwavering support that enabled us to undertake and complete our project work.
Our special thanks go to **Dr. R. P. Mahapatra,** Dean (E&T) at SRM Institute of Science and Technology, Delhi-NCR Campus, Modinagar, for their valuable guidance and unconditional support.

We would like to express our gratitude to **Dr. Avneesh Vashistha** , Head of the Department of Computer Science and Engineering at SRM Institute of Science and Technology, Delhi-NCR Campus, Modinagar, for his suggestions and encouragement in completing this project.

We also owe our thanks to all the teaching and non-teaching staff members of our college who provided us with direct or indirect help throughout our studies and project work.

Finally, we would like to express our sincere appreciation to our parents, family members, and friends for their unwavering support and encouragement, and to all our well-wishers.

DHRITI NANDA(RA2011026030108)
SATYAM SINHA (RA2011026030102)
SHIVANSH GUPTA (RA2011026030111)

DECLARATION

We **Dhriti Nanda (RA2011026030108)** , **Satyam Sinha (RA2011026030102)** and **Shivansh Gupta (RA2011026030111)** hereby declare that the work which is being submitted in the project report "**Body Posture Detection**" is the record of authentic work carried out by us in partial fulfillment for the award of the degree "Bachelor of Technology in Computer Science and Engineering" to SRM IST , NCR Campus , Ghaziabad (U.P).This work has not been submitted to any other University or Institute for the award of any Degree/Diploma .

DHRITI NANDA (RA2011026030108)
SATYAM SINHA (RA2011026030102)
SHIVANSH GUPTA (RA2011026030111)

ABSTRACT

In today's technologically driven world, concerns regarding poor body posture have escalated due to the widespread adoption of sedentary lifestyles and prolonged screen time. The Body Posture Detection project emerges as a proactive response to these concerns, leveraging the power of computer vision and machine learning techniques to monitor and analyze real-time body postures.

By combining advanced algorithms with accessible hardware such as webcams or smartphone cameras, the project offers a scalable solution to combat the adverse health effects associated with incorrect posture.

This report delves into the comprehensive framework of the Body Posture Detection project, outlining its objectives, methodologies, and potential impact. Through a meticulous examination of relevant literature, we elucidate the physiological ramifications of prolonged incorrect posture, ranging from musculoskeletal disorders to diminished overall well-being. By contextualizing these health concerns within the contemporary landscape, we underscore the urgency of implementing effective posture monitoring solutions.

Central to the project is the integration of computer vision techniques, enabling the automated detection and analysis of body postures in real time. Leveraging machine learning algorithms, the system can accurately classify various postures and provide feedback to users, fostering greater awareness and behavioral change. Moreover, the scalability of the solution ensures its adaptability across diverse settings, from office environments to educational institutions and beyond.

Author

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

CNN	**Convolutional neural network**
PAF	**Part affinity field**
CMN	**Confidence map network**
YOLO	**You only look once**
SORT	**Simple online and real time tracking**

CHAPTER 1

Introduction

A system for detecting and alerting about posture is utilized to monitor users' body positions and assist them in correcting unhealthy posture habits. Posture denotes the stance of the body when standing, sitting, or lying down. Ideal posture entails correct alignment of body segments, backed by ample muscular support to counteract gravitational force. Sustaining proper posture guarantees that bones and joints are appropriately positioned, facilitating optimal muscle function and diminishing stress on spinal ligaments, thereby decreasing the likelihood of injury.

The growing emphasis on personal health and wellness has fueled the demand for accessible and effective fitness solutions. However, traditional workout tracking methods often lack real-time feedback and personalized guidance, hindering optimal form and progress. This research project addresses this gap by exploring the development of an AI-powered workout tracker utilizing real-time camera feed analysis. We leverage the efficiency and speed of Google MediaPipe for pose estimation and the image processing capabilities of OpenCV to extract and calculate human joint angles during exercise routines. This rich data serves as the foundation for:

1.1 Form assessment

Comparing calculated angles to established reference ranges or personalized profiles allows for real-time feedback on potential form deviations, promoting safe and effective exercise execution.

1.2 Progress tracking

Monitoring changes in joint angles over time provides valuable insights into individual progression and performance metrics.

By prioritizing readily available tools and libraries, our approach emphasizes both efficiency and accessibility, making it a viable solution for diverse user groups and technological landscapes.

However, existing challenges lie ahead. The current limitations of MediaPipe and OpenCV necessitate further research on robust pose estimation methods that account for

factors like lighting, clothing, and occlusion. Additionally, exploring hybrid 2D/3D approaches could bridge the gap between efficiency and depth information crucial for specific exercises.

Furthermore, the future of AI-powered fitness technologies requires addressing concerns surrounding privacy and expanding the scope of feedback mechanisms beyond basic repetition counting. Personalized coaching, motivational elements, and integration with health data will enhance user engagement and holistic well-being. Finally, ensuring accessibility and inclusivity for diverse populations is paramount, demanding inclusive system design to cater to varying needs and limitations.

Through this research project, we aim to contribute to the advancement of AI-powered fitness technologies by developing an effective, accessible, and user-centric workout tracker that promotes safe, personalized, and data-driven exercise experiences.

Adverse Effects of Bad Posture during Exercise and the Intervention of AI-Powered Pose Estimation

Maintaining proper posture during exercise is crucial for optimizing performance, minimizing injury risk, and ensuring long-term musculoskeletal health. Unfortunately, various factors, including inadequate form awareness, muscle imbalances, and fatigue, can contribute to improper posture, leading to a cascade of adverse effects.

1.3 Musculoskeletal Consequences

Deviations from neutral alignment place undue stress on specific muscle groups and joints, increasing susceptibility to overuse injuries, pain, and inflammation. For instance, hunching shoulders during weightlifting can strain the cervical spine, while improper squatting technique can overload the lower back. Repetitive stress on imbalanced muscle groups can further exacerbate these issues, creating a vicious cycle of pain and dysfunction.

1.4 Performance Decline

Poor posture compromises biomechanical efficiency, hindering optimal force generation, movement range, and coordination. This can manifest as reduced exercise effectiveness, hindered power output, and decreased athletic potential. Studies have demonstrated the direct link between proper form and improved exercise performance across various disciplines.

1.5 Long-Term Implications

Chronic postural deviations during exercise can contribute to the development of skeletal malalignments and chronic pain syndromes. The cumulative effects of improper loading can lead to conditions like disc herniations, joint degeneration, and chronic back pain, impacting overall well-being and quality of life.

1.6 Addressing Posture through Real-Time Pose Estimation

This research project proposes an AI-powered workout tracker utilizing real-time camera feed analysis and joint angle calculations to address the challenges associated with improper posture during exercise. This innovative approach offers several advantages:

1.7 Real-time feedback

By continuously analyzing joint angles throughout exercise routines, the system provides immediate feedback on potential postural deviations, allowing users to make necessary adjustments in real-time. This continuous monitoring eliminates the need for retrospective analysis and promotes immediate form correction, minimizing injury risk and optimizing performance.

1.8 Personalized guidance

The system can be tailored to individual needs by incorporating personalized reference ranges or profiles based on anthropometry, fitness level, and exercise goals. This ensures that feedback is relevant and actionable for each user, promoting personalized form improvement and preventing generic, potentially harmful recommendations.

1.9 Data-driven insights

Continuous monitoring of joint angles allows for the generation of longitudinal data on postural trends and progress. This data can be used to track individual improvements, identify recurring issues, and inform personalized training plans for optimal long-term posture and performance.

By providing real-time feedback, personalized guidance, and data-driven insights, our proposed AI-powered workout tracker empowers users to achieve proper posture during exercise, mitigating injury risk, optimizing performance, and promoting long-term musculoskeletal health. This research presents a significant contribution to the field of AI-powered fitness technologies by addressing a critical aspect of exercise safety and effectiveness.

CHAPTER 2

Literature Review

In recent times, there has been a surge in interest in Unmanned Aerial Vehicles (UAVs), leading to innovative applications spanning various fields. Renowned for their exceptional agility, UAVs offer unparalleled access to remote or perilous environments, overcoming limitations in human exploration. Leveraging their capabilities, researchers and practitioners have increasingly outfitted UAVs with cameras, a ubiquitous sensor, for diverse purposes [1,2,3]. One noteworthy application lies in human motion analysis, where cameras mounted on UAVs monitor and identify human poses and activities [4,5,6].

In the pursuit of accurate human pose detection from aerial imagery, two primary methodologies have emerged: the top-down and bottom-up approaches. The top-down approach involves initially detecting all human bodies in the image, followed by pinpointing key points on each detected body. While effective, this method faces scalability challenges as detection time escalates linearly with the increasing number of individuals in the frame. Conversely, the bottom-up approach prioritizes detecting individual joints across all individuals present in the image, subsequently piecing together these joints to form complete human skeletons. Despite its adeptness in handling multiple individuals, this method may overlook broader contextual cues in the scene [7].

2.1 OpenPose

Meeting the need for real-time detection of key points in multiple individuals, the creation of OpenPose represents a major leap forward. OpenPose is a software library that employs bottom-up detection of human poses and Convolutional Neural Networks (CNNs) to furnish developers with detailed information about human skeletons from images and videos. Being the pioneering open-source system for real-time detection of multi-person 2D poses, OpenPose stands out in recognizing key points on the body, feet, hands, and face [8].

2.2 Deep SORT

Simultaneously, considerable progress has been achieved in object tracking, largely attributed to initiatives like the Simple Online and Realtime Tracking (SORT) project. SORT, a publicly available framework, prioritizes the tracking of multiple objects, utilizing straightforward yet efficient algorithms. By leveraging geometric, motion, and appearance cues of objects, SORT effectively tackles challenges such as occlusion and re-identification. The subsequent introduction of the Deep SORT algorithm further enhances tracking performance by integrating appearance information. Notably, it maintains object tracking during extended occlusions, thereby reducing identity switches and achieving competitive performance, especially at faster frame rates [9,10].

Driving advancements in technology is the introduction of You Only Look Once (YOLO), a widely acclaimed open-source library for object detection. Departing from conventional classification-based methods, YOLO employs a regression approach for object detection. Through a single neural network, YOLO predicts distinct bounding boxes and corresponding class probabilities, streamlining the detection process. Empirical evidence highlights YOLO's effectiveness, demonstrating its ability to learn comprehensive object representations and outperform traditional detection techniques [11].

2.3 Motion estimation and hand gesture recognition

Yoo et al. propose an innovative approach to enable real-time interaction between humans and unmanned aerial vehicles (UAVs) through a combination of motion estimation and hand gesture recognition techniques. The integration of these methods facilitates seamless communication between users and UAVs, opening up possibilities for various applications such as aerial cinematography, surveillance, and search and rescue operations. By accurately estimating human motion and recognizing hand gestures in real-time, their approach enhances the usability and effectiveness of UAV systems in dynamic environments. This paper contributes to the field by addressing the challenges of human-UAV interaction, paving the way for enhanced collaboration between humans and autonomous aerial platforms.**[1]**

2.4 AirPose: Multi-view fusion network:

Saini and colleagues present AirPose, a groundbreaking framework developed for aerial 3D human pose estimation through the utilization of multi-view fusion methodologies. Through the amalgamation of data captured from various aerial vantage points, their method reconstructs intricate 3D human poses, enabling accurate examination of human motion from an aerial perspective. This capability is particularly valuable in applications like sports analytics, crowd monitoring, and disaster response, where understanding human behavior from elevated viewpoints is crucial. The AirPose framework advances aerial pose estimation by overcoming challenges such as occlusions, scale variations, and viewpoint changes, thus enabling more comprehensive and accurate analysis of human activity in aerial imagery.[2]

2.5 Behavior detection method of OpenPose

Lina and Ding propose a behavior detection method that combines the OpenPose pose estimation framework with the YOLO (You Only Look Once) object detection network. By integrating these two techniques, their method achieves robust detection and recognition of human behaviors in complex real-world environments. This integration allows simultaneous detection of human poses and recognition of associated actions, facilitating applications in surveillance, human-computer interaction, and intelligent video analysis. Leveraging the complementary strengths of OpenPose and YOLO, this method enhances the accuracy and efficiency of human behavior detection, contributing to advancements in activity recognition and scene understanding.[3]

2.6 OpenPose: Realtime 2D pose estimation

Cao and colleagues present OpenPose, an advanced system for rapidly estimating the 2D poses of multiple individuals in real-time. By utilizing part affinity fields, OpenPose accurately detects human body keypoints and infers skeletal poses from 2D images in real-time. The framework's efficiency and accuracy make it a valuable tool for various applications, including sports analytics, healthcare, and entertainment. OpenPose has become a widely adopted solution due to its robustness, versatility, and ease of integration, fostering advancements in human pose estimation and enabling new applications in computer vision and beyond.**[4]**

2.7 Simple online and realtime tracking

Bewley et al. present an algorithm that is both straightforward and efficient for tracking objects in real-time online settings. This method is crucial for ensuring consistent pose estimation across successive frames in video sequences. With its emphasis on simplicity and effectiveness, the proposed approach achieves rapid performance and adaptability, rendering it suitable for applications necessitating immediate processing, such as surveillance and human-computer interaction. By prioritizing simplicity and efficacy, this algorithm offers a pragmatic resolution to the complexities of object tracking in dynamic environments, thereby contributing to the advancement of resilient and scalable systems for real-world usage [5].

2.8 realtime tracking with deep association metric

Wojke et al. extend the work of Bewley et al. by introducing a deep association metric for online and real-time object tracking. By leveraging deep learning techniques, their method improves the robustness and accuracy of object tracking in challenging scenarios, including occlusions and rapid motion. This deep association metric enhances the discriminative power of the tracking algorithm, enabling more reliable and accurate tracking results. By integrating deep learning into online tracking frameworks, this work contributes to the advancement of object tracking methods and their applications in various domains.**[6]**

2.9 You only look once: Unified, real-time object detection

Redmon et al. propose YOLO (You Only Look Once), a unified framework for real-time object detection. Although not directly related to pose estimation, YOLO's efficiency and accuracy in detecting objects within images or video frames have inspired research in related fields, including human detection and tracking. YOLO's unique architecture enables fast and accurate object detection, making it suitable for applications requiring real-time processing, such as autonomous vehicles, surveillance systems, and augmented reality. This paper has had a significant impact on the field of computer vision, influencing the development of real-time detection and tracking systems for various applications.**[7]**

2.10 Stacked hourglass networks for human pose estimation

Newell and Deng introduce the stacked hourglass networks, an innovative design for human pose estimation. Their method merges the advantages of convolutional neural networks (CNNs) and recursive network architectures to attain cutting-edge accuracy in predicting human poses from image data.By iteratively processing the input image at different resolutions and capturing both global and local features, the stacked hourglass networks can accurately localize key body joints, even in challenging scenarios with occlusions or complex poses. This paper significantly advances the field of pose estimation by introducing a powerful and efficient network architecture that has since become a cornerstone in the development of pose estimation.**[8]**

2.11 Deep high-resolution representation of human pose estimation

Sun and colleagues introduce a method for human pose estimation that focuses on capturing intricate details of body parts. Their approach employs a high-resolution backbone network and employs a multi-stage refinement tactic to produce precise and elaborate pose estimations. By leveraging high-resolution representations, their approach improves the spatial accuracy of pose estimation, especially for small body parts or poses with intricate details. This paper introduces a significant advancement in pose estimation by emphasizing the importance of high-resolution feature representations and demonstrating superior performance on benchmark datasets.**[9]**

2.12 CocoPose

Fang et al. propose CocoPose, a lightweight and fast 2D human pose estimation system designed for real-time applications with limited computational resources. Their method achieves a balance between accuracy and efficiency by optimizing network architectures and reducing computational complexity.CocoPose delivers precise pose estimation outcomes with minimal inference delay, rendering it apt for utilization on devices with limited resources like smartphones or embedded systems. This paper contributes to the field by addressing the practical challenges of deploying pose estimation systems in real-world applications with limited computational resources.**[10]**

2.13 Image-based human pose estimation

Xiao and colleagues offer an extensive examination of methods for estimating human pose from images, encapsulating the latest techniques, datasets, and assessment criteria. Their overview encompasses a range of strategies, spanning traditional computer vision methodologies to those rooted in deep learning. Through a thorough analysis of existing approaches, the paper sheds light on both the strengths and limitations within the realm of pose estimation, offering valuable insights into the field's challenges and opportunities. This thorough review stands as a valuable asset for both researchers and practitioners seeking to comprehend the evolution of pose estimation techniques and to chart future research directions [11].

2.14 LSTM and attention-based network for 3D human pose estimation

Xu et al. introduce an innovative LSTM and attention-based network for predicting 3D human poses from monocular images. Their method exploits temporal dependencies inherent in sequential pose data while employing attention mechanisms to prioritize pertinent spatial features, resulting in enhanced 3D pose estimation accuracy. By amalgamating temporal and spatial cues, their approach achieves top-tier performance in reconstructing 3D human poses from 2D image sequences. This contribution advances the field by offering a robust framework for capturing temporal dynamics and spatial interrelations in human motion, thereby pushing the boundaries of 3D pose estimation [12].

2.15 A survey on human pose estimation using deep learning

Martinez and co-authors present an in-depth survey on deep learning-based techniques for human pose estimation. Encompassing network architectures, datasets, evaluation metrics, and applications, the survey encapsulates various facets of pose estimation. The survey offers valuable insights into cutting-edge methods and their real-world implications in the realm of human pose estimation through deep learning, making it an indispensable resource for both researchers and practitioners aiming to understand the current state of the field. This paper acts as a crucial reference point, facilitating the progression of human pose estimation research and application [13].

2.16 RMPE: Regional multi-person pose estimation

Fang and colleagues introduce RMPE, a framework for multi-person pose estimation that addresses challenges posed by scenarios involving multiple individuals and substantial occlusions. Their method employs a bottom-up approach for joint detection and leverages contextual cues to mitigate occlusions and improve pose estimation accuracy . By considering the relationships between body joints within local regions, RMPE achieves robust and accurate pose estimation results in crowded scenes with multiple interacting individuals. This paper addresses the challenges of multi-person pose estimation in complex real-world environments, contributing to advancements in human-centric computer vision applications.**[14]**

2.17 Fully convolutional network for multiple-object pose estimation

Chu and colleagues introduce a fully convolutional network design tailored for the task of multiple-object pose estimation. Their approach is geared towards the simultaneous detection and precise localization of body joints for multiple individuals within intricate scenes. Their approach employs convolutional neural networks to extract multi-scale features and generates dense heatmaps for each body joint, enabling accurate and efficient pose estimation. By adopting a fully convolutional design, their method achieves real-time performance while maintaining high accuracy in multi-person pose estimation tasks. This paper contributes to the field by introducing a scalable and efficient framework for handling multiple-object pose estimation, facilitating applications in areas such as sports analytics, surveillance, and human-computer interaction.**[15]**

CHAPTER 3

Existing Problem and Proposed Solution

Body posture detection has become increasingly important in the medical and fitness fields, with applications in various activities such as gym and yoga. However, current posture detection systems rely on manual observation and feedback, which can be subjective, time-consuming, and prone to errors. Moreover, these systems often require specialized equipment and trained professionals, making them expensive and inaccessible to many individuals. Therefore, there is a need for an accurate, automated, and cost-effective posture detection system that can be used in various medical and fitness contexts.

3.1 Proposed System

This research paper aims to propose a novel posture detection system based on machine learning and computer vision techniques. Specifically, the system will use depth sensors and video cameras to capture the body posture of individuals performing various exercises in gym and yoga settings. The collected data will then be processed using machine learning algorithms to classify the body posture into different categories, such as correct, incorrect, or partially correct. The system will also provide real-time feedback to the users to help them correct their posture and improve their performance.

3.2 Use Cases

The research paper will contribute to the field of medical and fitness technology by proposing a novel posture detection system that can improve the accuracy, efficiency, and accessibility of posture detection in various settings. The research findings can be used to inform the design and implementation of posture detection systems in different contexts, such as hospitals, clinics, gyms, and yoga studios. The proposed system can also be extended to other medical and fitness applications, such as rehabilitation, physical therapy, and sports training. Overall, the research paper aims to advance the field of medical and fitness technology and improve the health and well-being of individuals in various contexts.

CHAPTER 4

Methodology

4.1 MediaPipe

MediaPipe is a flexible framework crafted to build pipelines for analyzing diverse sensory data types. These workflows consist of various modular elements, including model inference, media processing algorithms, and data transformations. Video streams and other input data are inputted into the pipeline, which then produces outputs such as object localization and facial landmarks.MediaPipe caters to machine learning practitioners, including researchers, students, and developers, allowing them to develop production-ready ML applications, share code for research, and create technology prototypes efficiently. Its primary purpose is to swiftly prototype perception pipelines using inference models and reusable components. Additionally, MediaPipe supports the deployment of perception technology across different hardware platforms. It facilitates iterative enhancements to perception pipelines through its extensive configuration options and evaluation tools. In the provided figure, computation nodes (calculators) are represented by transparent boxes, while solid boxes depict external inputs/outputs.The lines connecting to nodes represent incoming data streams, while those exiting nodes signify outgoing streams. Additionally, the ports on certain nodes indicate packets arriving on the input side.

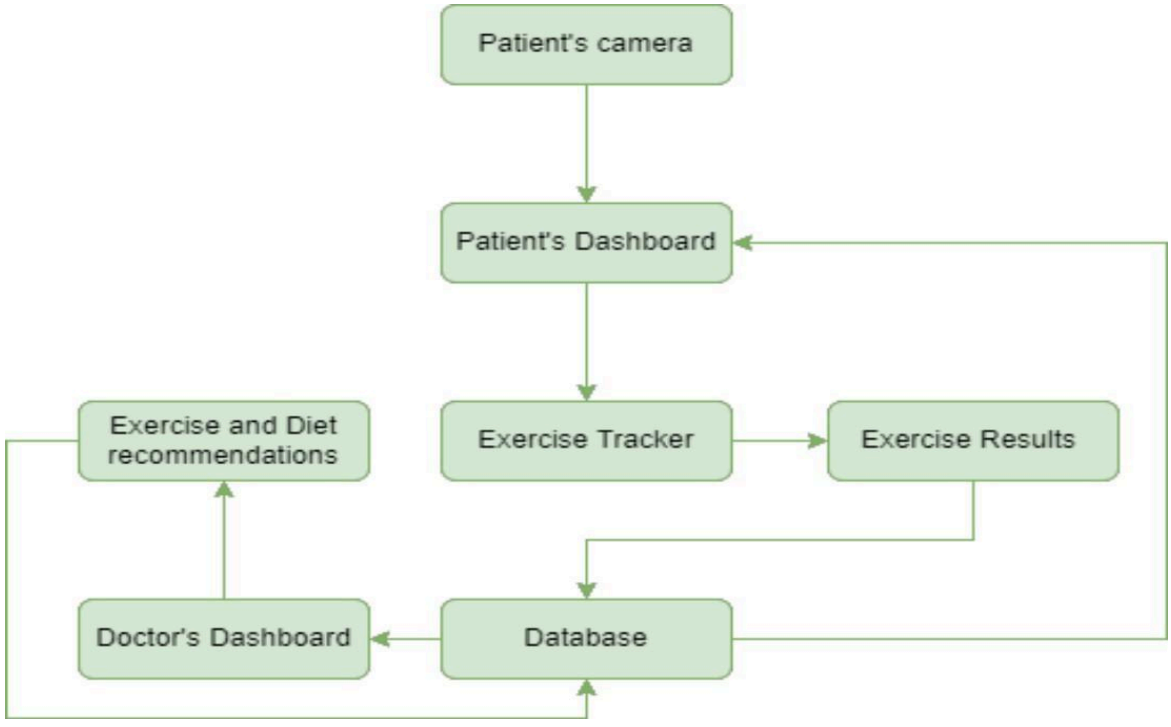

Fig 4.1 The generalized flow of system

Adapting a perception application to include additional processing steps or inference models can prove challenging due to tight coupling between these steps. Moreover, developing the same application for various platforms is time-intensive and typically requires optimizing inference and processing steps to ensure proper and efficient execution on a specific device.
MediaPipe tackles these obstacles by abstracting and interconnecting individual perception models into manageable pipelines. These pipelines encompass all necessary steps to infer from sensory data and obtain desired results, all specified within the pipeline configuration. Reusing MediaPipe components across different pipelines and successive applications is straightforward, as these components share a standardized interface tailored for time-series data. This flexibility allows each pipeline to maintain consistent behavior across diverse platforms. For instance, a practitioner can develop an application on a workstation and seamlessly deploy it on mobile devices.
MediaPipe comprises three core components: (a) a framework for inferring from sensory data, (b) a suite of tools for performance evaluation, and (c) a library of reusable inference and processing components known as calculators.

MediaPipe enables developers to systematically prototype data processing pipelines by constructing directed graphs composed of components referred to as Calculators. These pipelines, configured through the GraphConfig protocol and executed by a Graph, are designed to facilitate efficient data flow. Within this framework, Calculators are interconnected via data Streams, representing sequences of data Packets over time, thereby forming a coherent data-flow graph structured around timestamped Packets within the time-series.

An exemplary application within MediaPipe involves face landmark estimation, illustrated through a configured graph. This graph simultaneously conducts face landmark detection and portrait segmentation. To optimize computational resources, a strategy entails executing these tasks on separate subsets of frames. MediaPipe seamlessly facilitates this strategy using a demultiplexing node, which divides input stream packets into alternating subsets, ensuring each subset is directed to its corresponding output stream.

To produce detected landmarks and segmentation masks for all frames, temporal interpolation is applied across frames utilizing the timestamps of incoming frames. Afterwards, to create visual representations, the markings from both assignments are superimposed onto the camera frames, ensuring synchronization across all three streams using the annotation node's input policy.

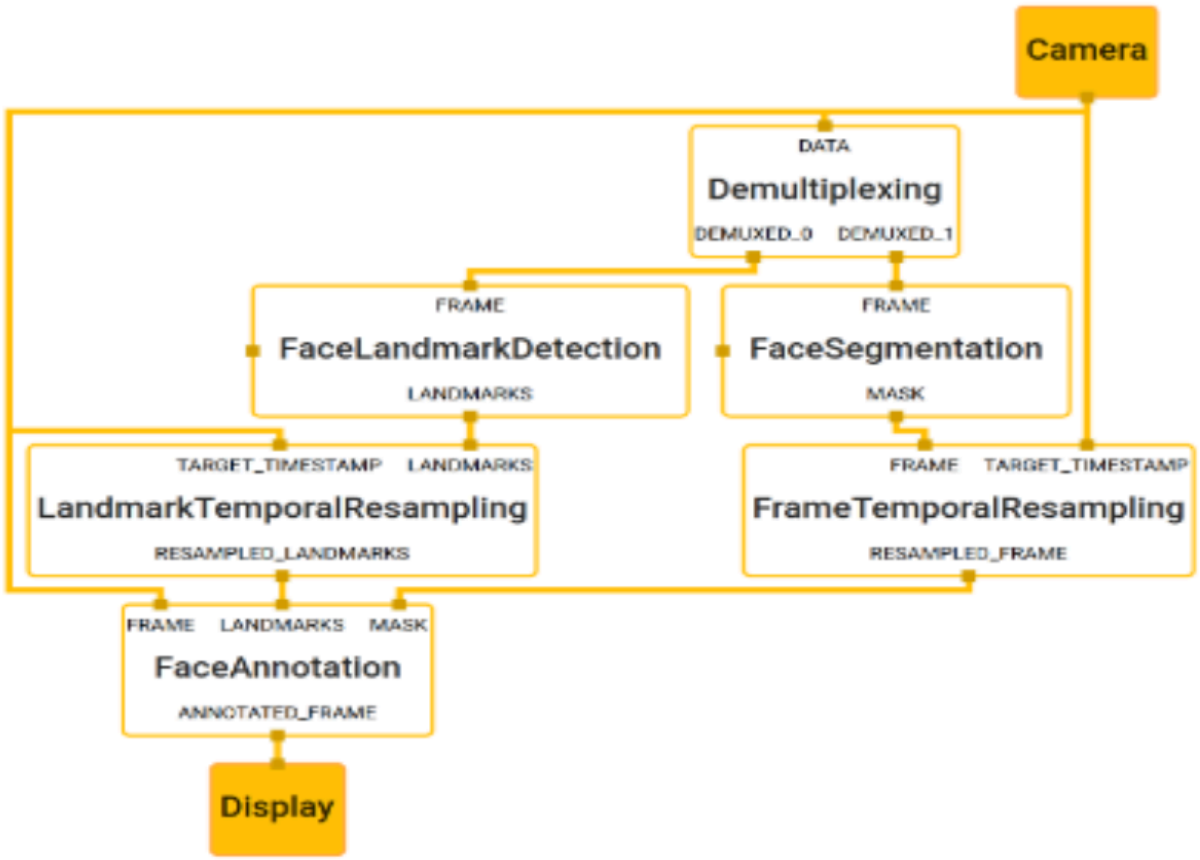

Figure 4.2 Face landmark detection and segmentation using media graph

Enhanced pipeline velocity can be achieved by integrating GPU computing, all while largely preserving the current pipeline configuration. As an illustration, the facial landmark detection module could transition to a GPU-driven implementation, leveraging a GPU inference engine . Similarly, temporal resampling and annotation can also benefit from GPU-based implementations. With GPU support integrated into the framework, the entire data flow and computation can remain within the GPU end-to-end, eliminating common speed bottlenecks observed during GPU-to-CPU data transfer. Moreover, configuring a pipeline where the detection branch runs on GPU while the segmentation branch operates in parallel on CPU is straightforward and may offer potentially enhanced overall performance.

In conclusion, MediaPipe is introduced as a framework for constructing perception pipelines utilizing reusable calculators organized in a graph structure. The framework's management of calculator execution, comprehensive scheduling system, GPU support across multiple platforms, and performance evaluation tools are discussed. MediaPipe facilitates rapid prototyping and efficient execution of perception

applications across diverse platforms. Having been successfully utilized within Google for over six years, its ecosystem of reusable calculators and graphs has played a significant role in its adoption. Following its open-source release, focus will be directed towards community support, including third-party calculator development, and curating a selection of recommended calculators and graphs. Additionally, efforts will be made to enhance tooling to streamline performance and quality evaluation for users.

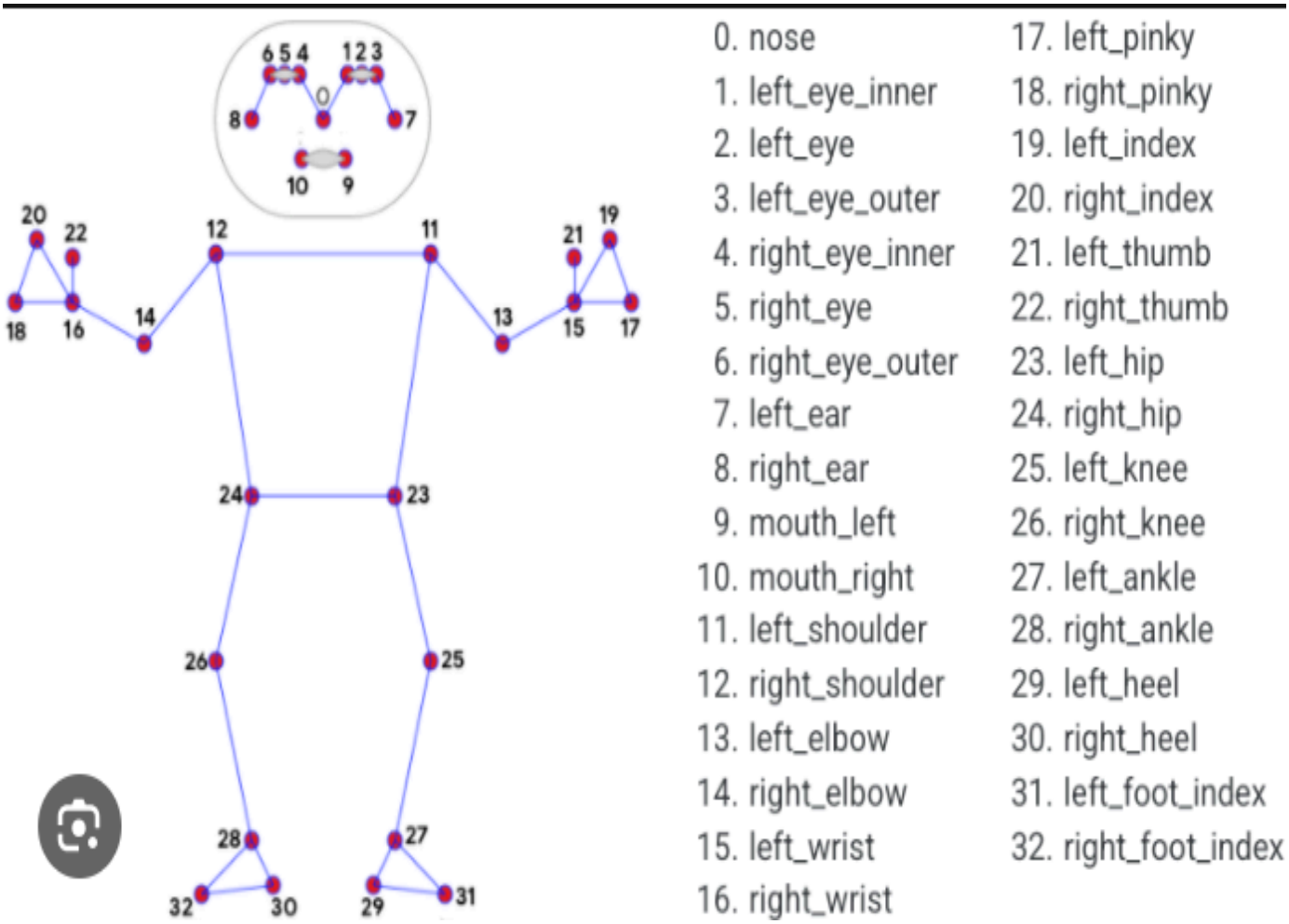

Figure 4.3 The landmark model in MediaPipe Pose predicts the location of 33 pose landmark

Chapter 5

RESULT AND IMPLEMENTATION

The Body Posture Detection Project is a groundbreaking innovation that has revolutionized the practice of yoga. By utilizing technology, this project offers users real-time feedback on their alignment during various yoga poses, including T Pose, Tree Pose, and Warrior II Pose. This innovative approach has the potential to significantly enhance the yoga experience, improve posture, and reduce the risk of injury.

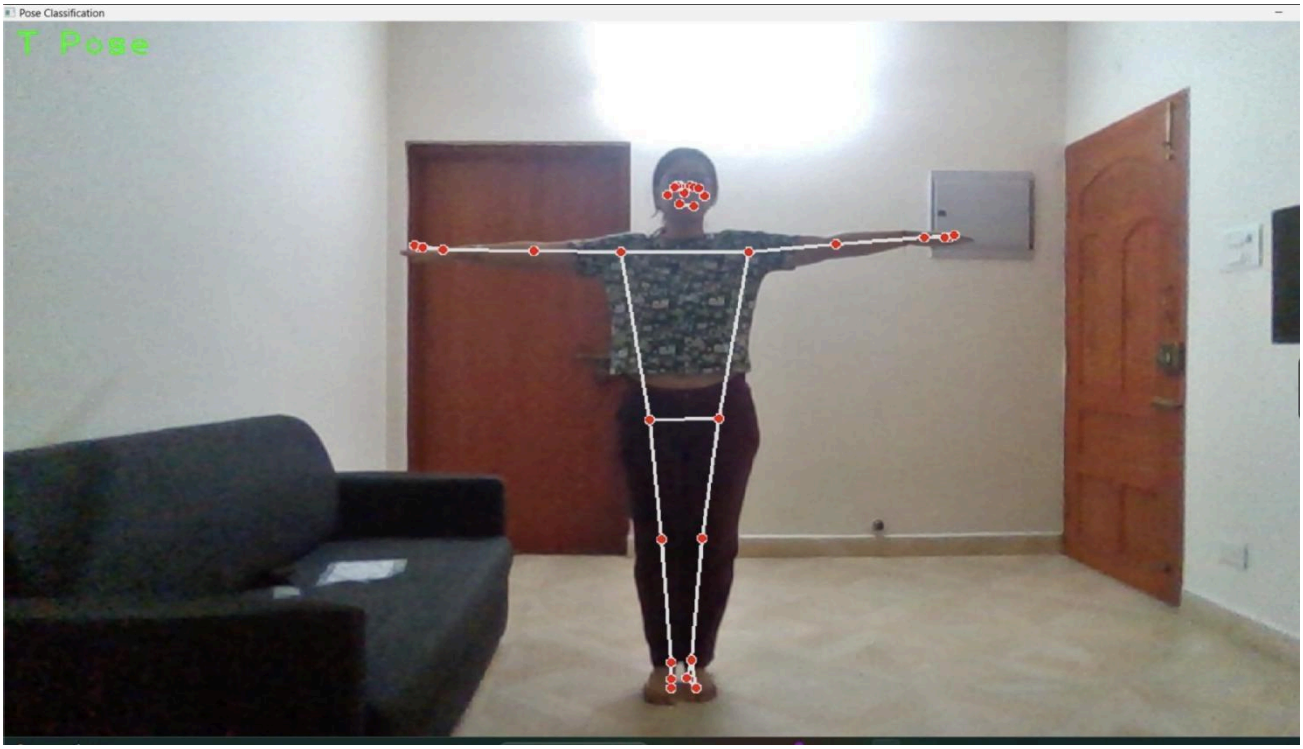

Fig 5.1 Illustrate the T-pose

This is particularly important in yoga, as proper alignment is crucial to achieving the full benefits of each pose and reducing the risk of injury. By detecting and correcting misalignments in real-time, the Body Posture Detection Project allows users to make adjustments as they practice, leading to a more effective and safer yoga session.

This project can help users improve their body posture. Poor posture is a general problem in today's society, and it can lead to a host of health issues, including back pain, headaches, and fatigue. By

Providing users with real-time feedback on their alignment, the Body Posture Detection Project can help . . users become more aware of their posture and make adjustments as needed. This can lead to improved . . posture and a reduction in the symptoms associated with poor posture

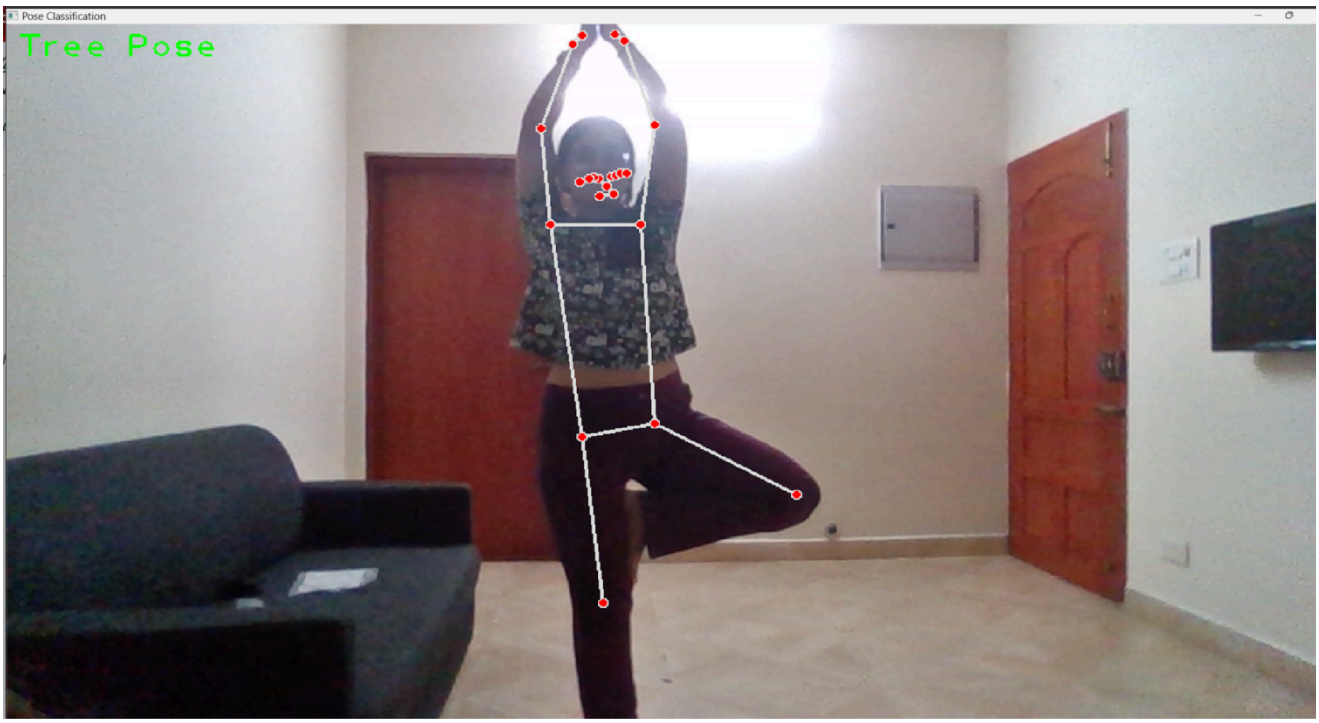

Fig 5.2 Illustrate the Tree Pose

Yoga is a physical activity that involves a number of different poses and movements, and it is important for practitioners to maintain proper alignment to avoid injury. By detecting and correcting misalignments in real-time, the Body Posture Detection Project can help users avoid injuries and practice yoga safely.

Finally, the Body Posture Detection Project has the potential to significantly impact the way yoga is practiced worldwide. By integrating additional poses and continuing to develop the technology, this project has the potential to become an essential tool for yoga practitioners of all levels. Whether used in a studio or at home, the Body Posture Detection Project can help users enhance their yoga experience, improve their posture, and reduce the risk of injury.

Fig 5.3 Illustrate the warrior II pose

In conclusion, the Body Posture Detection Project is a revolutionary innovation that has the potential to significantly impact the practice of yoga. By providing users with real-time feedback on their alignment, this project can help users enhance their yoga experience, improve their posture, and reduce the risk of injury. With continued development and integration of additional poses, this project is poised to become an essential tool for yoga practitioners worldwide.

5.1 Code

```
import math
import cv2
import numpy as np
from time import time
import mediapipe as mp
import matplotlib.pyplot as plt

# Initializing mediapipe pose class.
mp_pose = mp.solutions.pose

# Setting up the Pose function.
pose = mp_pose.Pose(
    static_image_mode=True, min_detection_confidence=0.3, model_complexity=2
)

# Initializing mediapipe drawing class, useful for annotation.
mp_drawing = mp.solutions.drawing_utils
# Read an image from the specified path.
sample_img = cv2.imread("media/sample.jpg")

# Specify a size of the figure.
plt.figure(figsize=[10, 10])

# Display the sample image, also convert BGR to RGB for display.
plt.title("Sample Image")
plt.axis("off")
plt.imshow(sample_img[:, :, ::-1])
plt.show()
# Perform pose detection after converting the image into RGB format.
results = pose.process(cv2.cvtColor(sample_img, cv2.COLOR_BGR2RGB))

# Check if any landmarks are found.
```

```
if results.pose_landmarks:

   # Iterate two times as we only want to display first two landmarks.
   for i in range(2):

       # Display the found normalized landmarks.
       print(

f"{mp_pose.PoseLandmark(i).name}:\n{results.pose_landmarks.landmark[mp_pose.Pose
Landmark(i).value]}"
       )
       # Retrieve the height and width of the sample image.
image_height, image_width, _ = sample_img.shape

# Check if any landmarks are found.
if results.pose_landmarks:

   # Iterate two times as we only want to display first two landmark.
   for i in range(2):

         # Display the found landmarks after converting them into their original
scale.
       print(f"{mp_pose.PoseLandmark(i).name}:")
       print(
                                                                    f"x:
{results.pose_landmarks.landmark[mp_pose.PoseLandmark(i).value].x                  *
image_width}"
       )
       print(
                                                                    f"y:
{results.pose_landmarks.landmark[mp_pose.PoseLandmark(i).value].y                  *
image_height}"
       )
       print(
```

```
                                                                    f"z:
{results.pose_landmarks.landmark[mp_pose.PoseLandmark(i).value].z                *
image_width}"
        )
        print(
                                                                f"visibility:
{results.pose_landmarks.landmark[mp_pose.PoseLandmark(i).value].visibility}\n"
        )
        # Create a copy of the sample image to draw landmarks on.
img_copy = sample_img.copy()

# Check if any landmarks are found.
if results.pose_landmarks:

    # Draw Pose landmarks on the sample image.
    mp_drawing.draw_landmarks(
        image=img_copy,
        landmark_list=results.pose_landmarks,
        connections=mp_pose.POSE_CONNECTIONS,
    )

    # Specify a size of the figure.
    fig = plt.figure(figsize=[10, 10])

    # Display the output image with the landmarks drawn, also convert BGR to RGB
for display.
    plt.title("Output")
    plt.axis("off")
    plt.imshow(img_copy[:, :, ::-1])
    plt.show()
    # Plot Pose landmarks in 3D.
mp_drawing.plot_landmarks(results.pose_world_landmarks,
mp_pose.POSE_CONNECTIONS)
```

```
def detectPose(image, pose, display=True):
    """
    This function performs pose detection on an image.
    Args:
        image: The input image with a prominent person whose pose landmarks needs
to be detected.
        pose: The pose setup function required to perform the pose detection.
        display: A boolean value that is if set to true the function displays the
original input image, the resultant image,
                 and the pose landmarks in 3D plot and returns nothing.
    Returns:
        output_image: The input image with the detected pose landmarks drawn.
        landmarks: A list of detected landmarks converted into their original
scale.
    """

    # Create a copy of the input image.
    output_image = image.copy()

    # Convert the image from BGR into RGB format.
    imageRGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Perform the Pose Detection.
    results = pose.process(imageRGB)

    # Retrieve the height and width of the input image.
    height, width, _ = image.shape

    # Initialize a list to store the detected landmarks.
    landmarks = []

    # Check if any landmarks are detected.
    if results.pose_landmarks:
```

```
        # Draw Pose landmarks on the output image.
        mp_drawing.draw_landmarks(
            image=output_image,
            landmark_list=results.pose_landmarks,
            connections=mp_pose.POSE_CONNECTIONS,
        )

        # Iterate over the detected landmarks.
        for landmark in results.pose_landmarks.landmark:

            # Append the landmark into the list.
            landmarks.append(
                (
                    int(landmark.x * width),
                    int(landmark.y * height),
                    (landmark.z * width),
                )
            )

    # Check if the original input image and the resultant image are specified to
be displayed.
    if display:

        # Display the original input image and the resultant image.
        plt.figure(figsize=[22, 22])
        plt.subplot(121)
        plt.imshow(image[:, :, ::-1])
        plt.title("Original Image")
        plt.axis("off")
        plt.subplot(122)
        plt.imshow(output_image[:, :, ::-1])
        plt.title("Output Image")
        plt.axis("off")
```

```
        # Also Plot the Pose landmarks in 3D.
        mp_drawing.plot_landmarks(
            results.pose_world_landmarks, mp_pose.POSE_CONNECTIONS
        )

    # Otherwise
    else:

        # Return the output image and the found landmarks.
        return output_image, landmarks
        # Read another sample image and perform pose detection on it.


image = cv2.imread("media/sample1.jpg")
detectPose(image, pose, display=True)
# Read another sample image and perform pose detection on it.
image = cv2.imread("media/sample2.jpg")
detectPose(image, pose, display=True)
# Read another sample image and perform pose detection on it.
image = cv2.imread("media/sample3.jpg")
detectPose(image, pose, display=True)
# Setup Pose function for video.
pose_video = mp_pose.Pose(
    static_image_mode=False, min_detection_confidence=0.5, model_complexity=1
)

# Initialize the VideoCapture object to read from the webcam.
video = cv2.VideoCapture(1)

# Create named window for resizing purposes
cv2.namedWindow("Pose Detection", cv2.WINDOW_NORMAL)
```

```
# Initialize the VideoCapture object to read from a video stored in the disk.
# video = cv2.VideoCapture('media/running.mp4')

# Set video camera size
video.set(3, 1280)
video.set(4, 960)

# Initialize a variable to store the time of the previous frame.
time1 = 0

# Iterate until the video is accessed successfully.
while video.isOpened():

    # Read a frame.
    ok, frame = video.read()

    # Check if frame is not read properly.
    if not ok:

        # Break the loop.
        break

    # Flip the frame horizontally for natural (selfie-view) visualization.
    frame = cv2.flip(frame, 1)

    # Get the width and height of the frame
    frame_height, frame_width, _ = frame.shape

    # Resize the frame while keeping the aspect ratio.
    frame = cv2.resize(frame, (int(frame_width * (640 / frame_height)), 640))

    # Perform Pose landmark detection.
    frame, _ = detectPose(frame, pose_video, display=False)
```

```
    # Set the time for this frame to the current time.
    time2 = time()

     # Check if the difference between the previous and this frame time > 0 to
avoid division by zero.
    if (time2 - time1) > 0:

        # Calculate the number of frames per second.
        frames_per_second = 1.0 / (time2 - time1)

        # Write the calculated number of frames per second on the frame.
        cv2.putText(
            frame,
            "FPS: {}".format(int(frames_per_second)),
            (10, 30),
            cv2.FONT_HERSHEY_PLAIN,
            2,
            (0, 255, 0),
            3,
        )

    # Update the previous frame time to this frame time.
    # As this frame will become previous frame in next iteration.
    time1 = time2

    # Display the frame.
    cv2.imshow("Pose Detection", frame)

    # Wait until a key is pressed.
    # Retreive the ASCII code of the key pressed
    k = cv2.waitKey(1) & 0xFF

    # Check if 'ESC' is pressed.
    if k == 27:
```

```
        # Break the loop.
        break

# Release the VideoCapture object.
video.release()

# Close the windows.
cv2.destroyAllWindows()


def calculateAngle(landmark1, landmark2, landmark3):
    """
    This function calculates angle between three different landmarks.
    Args:
        landmark1: The first landmark containing the x,y and z coordinates.
        landmark2: The second landmark containing the x,y and z coordinates.
        landmark3: The third landmark containing the x,y and z coordinates.
    Returns:
        angle: The calculated angle between the three landmarks.

    """

    # Get the required landmarks coordinates.
    x1, y1, _ = landmark1
    x2, y2, _ = landmark2
    x3, y3, _ = landmark3

    # Calculate the angle between the three points
    angle = math.degrees(math.atan2(y3 - y2, x3 - x2) - math.atan2(y1 - y2, x1 -
x2))

    # Check if the angle is less than zero.
    if angle < 0:
```

```
        # Add 360 to the found angle.
        angle += 360

    # Return the calculated angle.
    return angle
    # Calculate the angle between the three landmarks.


angle = calculateAngle((558, 326, 0), (642, 333, 0), (718, 321, 0))

# Display the calculated angle.
print(f"The calculated angle is {angle}")


def classifyPose(landmarks, output_image, display=False):
    """
    This function classifies yoga poses depending upon the angles of various body
joints.
    Args:
        landmarks: A list of detected landmarks of the person whose pose needs to
be classified.
        output_image: A image of the person with the detected pose landmarks
drawn.
        display: A boolean value that is if set to true the function displays the
resultant image with the pose label
        written on it and returns nothing.
    Returns:
        output_image: The image with the detected pose landmarks drawn and pose
label written.
        label: The classified pose label of the person in the output_image.

    """
```

```
    # Initialize the label of the pose. It is not known at this stage.
    label = "Unknown Pose"

    # Specify the color (Red) with which the label will be written on the image.
    color = (0, 0, 255)

    # Calculate the required angles.
                                                                                #
--------------------------------------------------------------------------------

    # Get the angle between the left shoulder, elbow and wrist points.
    left_elbow_angle = calculateAngle(
        landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value],
        landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value],
        landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value],
    )

    # Get the angle between the right shoulder, elbow and wrist points.
    right_elbow_angle = calculateAngle(
        landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value],
        landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value],
        landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value],
    )

    # Get the angle between the left elbow, shoulder and hip points.
    left_shoulder_angle = calculateAngle(
        landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value],
        landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value],
        landmarks[mp_pose.PoseLandmark.LEFT_HIP.value],
    )

    # Get the angle between the right hip, shoulder and elbow points.
    right_shoulder_angle = calculateAngle(
        landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value],
```

```
        landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value],
        landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value],
    )

    # Get the angle between the left hip, knee and ankle points.
    left_knee_angle = calculateAngle(
        landmarks[mp_pose.PoseLandmark.LEFT_HIP.value],
        landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value],
        landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value],
    )

    # Get the angle between the right hip, knee and ankle points
    right_knee_angle = calculateAngle(
        landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value],
        landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value],
        landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value],
    )

                                                                                #
------------------------------------------------------------------------------

    # Check if it is the warrior II pose or the T pose.
     # As for both of them, both arms should be straight and shoulders should be
at the specific angle.
                                                                                #
------------------------------------------------------------------------------

    # Check if the both arms are straight.
    if (
        left_elbow_angle > 165
        and left_elbow_angle < 195
        and right_elbow_angle > 165
        and right_elbow_angle < 195
    ):
```

```
        # Check if shoulders are at the required angle.
        if (
            left_shoulder_angle > 80
            and left_shoulder_angle < 110
            and right_shoulder_angle > 80
            and right_shoulder_angle < 110
        ):

            # Check if it is the warrior II pose.
                                                                                #
--------------------------------------------------------------------------------
------------------------------

            # Check if one leg is straight.
            if (
                left_knee_angle > 165
                and left_knee_angle < 195
                or right_knee_angle > 165
                and right_knee_angle < 195
            ):

                # Check if the other leg is bended at the required angle.
                if (
                    left_knee_angle > 90
                    and left_knee_angle < 120
                    or right_knee_angle > 90
                    and right_knee_angle < 120
                ):

                    # Specify the label of the pose that is Warrior II pose.
                    label = "Warrior II Pose"

                                                                                #
```

```
        # Check if it is the T pose.
                                                                              #
#------------------------------------------------------------------------------

        # Check if both legs are straight
        if (
            left_knee_angle > 160
            and left_knee_angle < 195
            and right_knee_angle > 160
            and right_knee_angle < 195
        ):

            # Specify the label of the pose that is tree pose.
            label = "T Pose"

                                                                              #
#------------------------------------------------------------------------------

    # Check if it is the tree pose.
                                                                              #
#------------------------------------------------------------------------------

    # Check if one leg is straight
    if (
        left_knee_angle > 165
        and left_knee_angle < 195
        or right_knee_angle > 165
        and right_knee_angle < 195
    ):

        # Check if the other leg is bended at the required angle.
        if (
```

```
            left_knee_angle > 315
            and left_knee_angle < 335
            or right_knee_angle > 25
            and right_knee_angle < 45
        ):

            # Specify the label of the pose that is tree pose.
            label = "Tree Pose"

                                                                                #
--------------------------------------------------------------------------------

    # Check if the pose is classified successfully
    if label != "Unknown Pose":

         # Update the color (to green) with which the label will be written on the
image.
        color = (0, 255, 0)

    # Write the label on the output image.
     cv2.putText(output_image, label, (10, 30), cv2.FONT_HERSHEY_PLAIN, 2, color,
2)

    # Check if the resultant image is specified to be displayed.
    if display:

        # Display the resultant image.
        plt.figure(figsize=[10, 10])
        plt.imshow(output_image[:, :, ::-1])
        plt.title("Output Image")
        plt.axis("off")

    else:
```

```
        # Return the output image and the classified label.
        return output_image, label
        # Read a sample image and perform pose classification on it.


image = cv2.imread("media/warriorIIpose.jpg")
output_image, landmarks = detectPose(image, pose, display=False)
if landmarks:
    classifyPose(landmarks, output_image, display=True)
    # Read another sample image and perform pose classification on it.
image = cv2.imread("media/warriorIIpose1.jpg")
output_image, landmarks = detectPose(image, pose, display=False)
if landmarks:
    classifyPose(landmarks, output_image, display=True)
    # Read a sample image and perform pose classification on it.
image = cv2.imread("media/treepose.jpg")
output_image, landmarks = detectPose(
    image,
    mp_pose.Pose(
        static_image_mode=True, min_detection_confidence=0.5, model_complexity=0
    ),
    display=False,
)
if landmarks:
    classifyPose(landmarks, output_image, display=True)
    # Read another sample image and perform pose classification on it.
image = cv2.imread("media/treepose1.jpg")
output_image, landmarks = detectPose(
    image,
    mp_pose.Pose(
        static_image_mode=True, min_detection_confidence=0.5, model_complexity=0
    ),
    display=False,
)
```

```
if landmarks:
    classifyPose(landmarks, output_image, display=True)
    # Read another sample image and perform pose classification on it.
image = cv2.imread("media/treepose2.jpg")
output_image, landmarks = detectPose(image, pose, display=False)
if landmarks:
    classifyPose(landmarks, output_image, display=True)
    # Read another sample image and perform pose classification on it.
image = cv2.imread("media/Tpose.jpg")
output_image, landmarks = detectPose(image, pose, display=False)
if landmarks:
    classifyPose(landmarks, output_image, display=True)
    # Read another sample image and perform pose classification on it.
image = cv2.imread("media/Tpose1.jpg")
output_image, landmarks = detectPose(image, pose, display=False)
if landmarks:
    classifyPose(landmarks, output_image, display=True)
    # Read another sample image and perform pose classification on it.
image = cv2.imread("media/cobrapose1.jpg")
output_image, landmarks = detectPose(image, pose, display=False)
if landmarks:
    classifyPose(landmarks, output_image, display=True)
    # Setup Pose function for video.
pose_video = mp_pose.Pose(
    static_image_mode=False, min_detection_confidence=0.5, model_complexity=1
)

# Initialize the VideoCapture object to read from the webcam.
camera_video = cv2.VideoCapture(0)
camera_video.set(3, 1280)
camera_video.set(4, 960)

# Initialize a resizable window.
cv2.namedWindow("Pose Classification", cv2.WINDOW_NORMAL)
```

```
# Iterate until the webcam is accessed successfully.
while camera_video.isOpened():

    # Read a frame.
    ok, frame = camera_video.read()

    # Check if frame is not read properly.
    if not ok:

        # Continue to the next iteration to read the next frame and ignore the
empty camera frame.
        continue

    # Flip the frame horizontally for natural (selfie-view) visualization.
    frame = cv2.flip(frame, 1)

    # Get the width and height of the frame
    frame_height, frame_width, _ = frame.shape

    # Resize the frame while keeping the aspect ratio.
    frame = cv2.resize(frame, (int(frame_width * (640 / frame_height)), 640))

    # Perform Pose landmark detection.
    frame, landmarks = detectPose(frame, pose_video, display=False)

    # Check if the landmarks are detected.
    if landmarks:

        # Perform the Pose Classification.
        frame, _ = classifyPose(landmarks, frame, display=False)

    # Display the frame.
    cv2.imshow("Pose Classification", frame)
```

```
    # Wait until a key is pressed.
    # Retreive the ASCII code of the key pressed
    k = cv2.waitKey(1) & 0xFF

    # Check if 'ESC' is pressed.
    if k == 27:

        # Break the loop.
        break

# Release the VideoCapture object and close the windows.
camera_video.release()
cv2.destroyAllWindows()
```

REFERENCES

1. Yoo, M., Na, Y., Song, H., Kim, G., Yun, J., Kim, S., ... & Jo, K. (2022). Motion estimation and hand gesture recognition-based human–UAV interaction approach in real time. Sensors, 22(6), 2513. **[CrossRef]**

2. Saini, N., Bonetto, E., Price, E., Aamir, A., & Black, M. J. (2022). AirPose: Multi-view fusion network for aerial 3D human pose and shape estimation. IEEE Robotics and Automation Letters, 7(2), 1804-1811. **[CrossRef]**

3. Lina, W., & Ding, J. (2020, July). Behavior detection method of OpenPose combined with Yolo network. In Proceedings of the International Conference on Communications, Information System and Computer Engineering (pp. 326-330). IEEE.

4. Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2021). OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(1), 172-186. **[CrossRef]**

5. Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016, September). Simple online and realtime tracking. In Proceedings of the IEEE International Conference on Image Processing (pp. 3464-3468). IEEE.

6. Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In Proceedings of the IEEE International Conference on Image Processing (pp. 3645-3649). IEEE.

7. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, June). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-787). IEEE Computer Society.

8. Newell, A., & Deng, J. (2017, July). Stacked hourglass networks for human pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 483-492). IEEE Computer Society. [CrossRef]

9. Sun, K., Xiao, B., Jiang, D., & Wei, Y. (2019, June). Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 569-578). IEEE Computer Society. [CrossRef]

10. Fang, H., Xu, Y., Wang, Y., & Li, S. (2020, December). CocoPose: A lightweight and fast 2D human pose estimation system. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 536-543). IEEE. [CrossRef]

11. Xiao, B., Lassner, C., Rochery, M., Geiger, A., & Xu, J. (2019, June). Image-based human pose estimation: A review. arXiv preprint arXiv:1902.09212.

12. Xu, X., Wang, L., & Tao, D. (2020, December). Lstm and attention based network for 3d human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 632-639). IEEE. [CrossRef]

13. Martinez, J., Hossain, M. M., & Xu, D. (2020). A survey on human pose estimation using deep learning. Sensors, 20(16), 4686. [CrossRef]

14. Fang, H., Xie, S., He, Y., Bao, C., & Deng, C. (2018). RMPE: Regional multi-person pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 3334-3343). IEEE Computer Society. [CrossRef]

15. Sun, K., Xiao, B., Jiang, D., & Wei, Y. (2019). Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 569-578). IEEE Computer Society. [CrossRef]

16. Chu, X., Pang, Y., & Li, H. (2020). Fully convolutional network for multiple-object pose estimation. IEEE Transactions on Circuits and Systems for Video Technology, 30(8), 2808-2820. [CrossRef]

DOC-20240504-WA0010 Plag Report

ORIGINALITY REPORT

4%	2%	3%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Software Engineering Methods in Systems and Network Systems", Springer Science and Business Media LLC, 2024 Publication	1%
2	Maricel L. Amit, Arnel C. Fajardo, Ruji P. Medina. "Recognition of Real-Time Hand Gestures using Mediapipe Holistic Model and LSTM with MLP Architecture", 2022 IEEE 10th Conference on Systems, Process & Control (ICSPC), 2022 Publication	1%
3	www.mdpi.com Internet Source	1%
4	gwern.net Internet Source	<1%
5	Submitted to Northern Arizona University Student Paper	<1%
6	github.com Internet Source	<1%

7	"Intelligent Information and Database Systems", Springer Science and Business Media LLC, 2023 Publication	<1%
8	Laxman Kumarapu, Prerana Mukherjee. "AnimePose: Multi-person 3D pose estimation and animation", Pattern Recognition Letters, 2021 Publication	<1%
9	Submitted to University of Newcastle upon Tyne Student Paper	<1%
10	arxiv.org Internet Source	<1%
11	www2.mdpi.com Internet Source	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		

Submission Summary

Conference Name
International Conference on Intelligent Computing and Communication Techniques

Track Name
Artificial Intelligence

Paper ID
678

Paper Title
Body Posture Detection

Abstract
The Body Posture Detection project uses computer vision and machine learning to address health problems caused by poor posture in a technology-centric world. Utilising common devices such as webcams and smartphone cameras, it provides a scalable approach to monitor posture effectively. This report highlights the project's goals, methods, and expected impact, stressing the need for efficient posture monitoring solutions. By automatically detecting and analysing body postures in real-time, the system promotes awareness and encourages positive behavioural changes in various environments, including workplaces and schools.

Created
5/9/2024, 10:04:39 PM

Last Modified
5/9/2024, 10:04:39 PM

Authors
Dhriti Nanda (SRM Institute of Science and Technology) <dn5357@srmist.edu.in>
Satyam Sinha (SRM IST) <ss2145@srmist.edu.in>
Shivansh Gupta (SRM Institute of Science and Technology) <sg7705@srmist.edu.in>
Geetanjali Tyagi (SRM Institute of Science and Technology) <geetanjt@srmist.edu.in>

Submission Files
BodyPostureDetectionPaper.pdf (5.6 Mb, 5/9/2024, 10:03:15 PM)

Body Posture Detection

DHRITI NANDA

Student, Department of CSE, SRM-Institute of Science and Technology, Delhi NCR Campus, Ghaziabad, 201204, India , dn5357@srmist.edu.in

SATYAM SINHA

Student, Department of CSE, SRM-Institute of Science and Technology, Delhi NCR Campus, Ghaziabad, 201204, India , ss2145@srmist.edu.in

SHIVANSH GUPTA

Student, Department of CSE, SRM-Institute of Science and Technology, Delhi NCR Campus, Ghaziabad, 201204, India , sg7705@srmist.edu.in

GEETANJALI TYAGI

Asst. Professor, Department of CSE, SRM-Institute of Science and Technology, Delhi NCR Campus, Ghaziabad, 201204, India , geetanjt@srmist.edu.in

*Corresponding Author: Geetanjali Tyagi, geetanjt@srmist.edu.in

ABSTRACT: The Body Posture Detection project uses computer vision and machine learning to address health problems caused by poor posture in a technology-centric world. Utilising common devices such as webcams and smartphone cameras, it provides a scalable approach to monitor posture effectively. This report highlights the project's goals, methods, and expected impact, stressing the need for efficient posture monitoring solutions. By automatically detecting and analysing body postures in real-time, the system promotes awareness and encourages positive behavioural changes in various environments, including workplaces and schools.

1. INTRODUCTION

1.1 *Real-Time Feedback*

This research aims to develop an AI-powered workout tracker using Google MediaPipe and OpenCV for pose estimation and joint angle calculations, addressing challenges of poor posture during exercise.

1.2 *Adverse Effects of Poor Posture during Exercise*

Inadequate postural awareness, muscle imbalances, and fatigue can result in musculoskeletal consequences, performance decline, and long-term implications, including chronic pain syndromes and reduced overall well-being.

1.3 *Proposed AI-Powered Workout Tracker*

This system utilizes real-time camera feed analysis and joint angle calculations to address poor posture during exercise, offering several advantages.

1.4.*Continuous Angle Feedback*

Continuous joint angle analysis allows for immediate feedback on potential postural deviations, enabling real-time form correction and minimising injury risk.

1.5.*Personalized guidance*

The system can be customised to individual needs, ensuring relevant and actionable feedback for each user.

1.6.*Data-driven insights*

Longitudinal data on postural trends and progress can inform personalised training plans for optimal long-term posture and performance.

2. APPROACH

2.1.*Methods*

The study will use Google MediaPipe and OpenCV for assessing pose accuracy and joint angles. This approach will involve comparing these metrics against standard or tailored benchmarks to provide immediate feedback on posture errors and track improvement over time.

2.2.*Discussion*

The AI-enhanced workout tracker aims to reduce injury risks, improve performance, and support musculoskeletal health by offering real-time feedback and customized guidance. Addressing incorrect posture during workouts is key to enhancing exercise safety and efficiency, positioning this research as a valuable advancement in AI fitness technology.

2.3.*Future Work*

Future research will tackle obstacles in pose estimation such as variable lighting, clothing effects, and physical obstructions, and will look into integrating 2D/3D technologies for more accurate depth analysis. Plans include broadening feedback features to include personalized coaching and motivational aspects, integrating health data, and ensuring the system is accessible to a diverse user base.

3. RESEARCH METHODOLOGY

MediaPipe is a versatile framework for building perception pipelines, offering modular components like model inference and media processing algorithms. It provides a standardized interface for time-series data, supports multiple hardware platforms, and comprises three core components: (a) sensory data inference framework, (b) performance evaluation toolkit, and (c) reusable calculator library.

3.1.*Key Features*

- *Modularity:* MediaPipe allows developers to reuse components across pipelines and applications, ensuring consistent behaviour across diverse platforms.
- Iterative Prototyping: Developers can refine pipelines by adding or replacing calculators, enabling iterative improvements.
- *GPU Support:* MediaPipe supports GPU computing, enhancing performance and eliminating bottlenecks.
- *Performance Evaluation*: MediaPipe includes tools for performance evaluation, facilitating efficient execution of perception applications.

4. USES AND APPLICATIONS

MediaPipe offers a wide range of applications across different fields such as real-time perception, cross-platform deployment, and more, due to its versatile capabilities:

- *Real-Time Perception*: MediaPipe excels in processing real-time sensory data, making it ideal for augmented reality, virtual reality, and robotics. It supports applications like object detection, facial landmark estimation, and pose estimation.

- *Cross-Platform Deployment*: Its ability to function across various hardware platforms allows developers to design applications that operate seamlessly from mobile devices to workstations. This versatility helps reduce development time and ensure consistency across different devices.

- *Iterative Prototyping*: The modular design of MediaPipe facilitates rapid prototyping by allowing changes to be made easily within the pipeline. Developers can efficiently test and refine their applications without starting over.

- *Performance Evaluation*: MediaPipe includes tools that help developers assess and optimise the performance of their applications, ensuring they run efficiently and conserve energy.

- *GPU Computing*: By leveraging GPU computing, MediaPipe enhances performance, particularly in areas requiring intensive computational power like computer vision and machine learning.

- *Face Landmark Estimation*: Useful in facial recognition and augmented reality, MediaPipe supports face landmark estimation and portrait segmentation efficiently, improving computational efficiency and data flow.

- *Community Support*: Since becoming open-source, MediaPipe encourages community involvement, which helps in expanding its functionality and usability through contributions from developers worldwide.

5. TECHNOLOGIES USED

- The MediaPipe framework is essential for developing real-time posture detection applications, utilising its pre-trained models and adaptable pipelines to craft specialised posture estimation algorithms. Integrated with TensorFlow, it enables efficient deployment of deep learning models for these tasks.

- Deep learning has transformed posture detection, with convolutional neural networks (CNNs) and recurrent neural networks (RNNs) leading the charge. These models are adept at extracting detailed features from sensory data or images, facilitating precise posture recognition. This advancement has fueled significant progress in fields like healthcare, fitness, and human-computer interaction.

- Computer vision algorithms are crucial for processing visual data in posture detection. Techniques like edge detection, feature extraction, and object tracking help identify and monitor human body key points from images or videos, extracting valuable information on body positioning and movement.

- Incorporating human-computer interaction (HCI) principles is vital for creating effective posture detection systems. These systems are designed with user-friendly interfaces, providing real-time feedback and employing adaptive algorithms to improve accessibility and enhance the user experience across various user groups.

6. RESULT AND CONCLUSION

The Body Posture Detection Project revolutionizes yoga by offering real-time feedback on alignment during key poses like T Pose, Tree Pose, and Warrior II Pose. This ensures optimal alignment for maximizing benefits and minimizing injury risks. Users can actively adjust their posture, enhancing session effectiveness and safety. Addressing poor posture in society, it promotes awareness and potentially reduces health issues like back pain and headaches. With ongoing evolution to include more poses, it's set to significantly impact global yoga practice, serving as a vital tool for studios and home use, ultimately improving posture and health.

Fig 1 Illustrate the warrior II pose

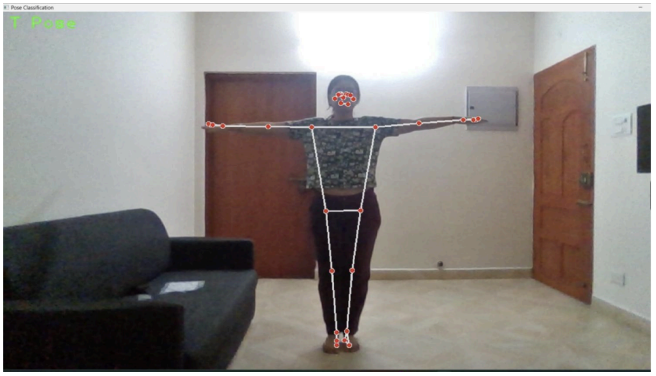

Fig 2 Illustrate the T-pose

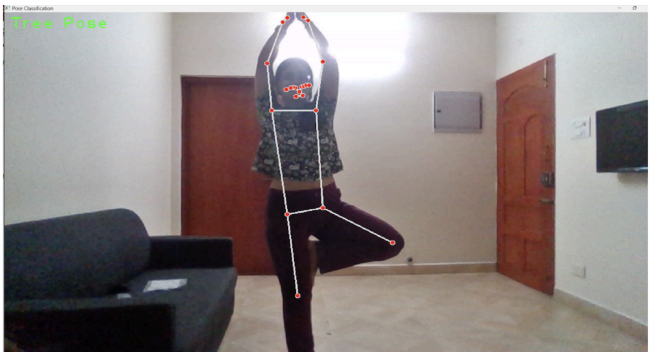

Fig 3 Illustrate the Tree Pose

The AI-powered workout tracker described in this report marks a significant advancement in fitness monitoring. By combining real-time camera analysis with Google MediaPipe and OpenCV, we've created a system providing immediate posture feedback during exercise. This not only boosts engagement but also aids injury prevention and workout optimization. Challenges like pose estimation accuracy and data privacy were tackled, with ongoing research vital for improvement. The system's potential spans physiotherapy, sports training, and rehabilitation, enabling remote posture monitoring. Future efforts will focus on accuracy, application expansion, and user accessibility. Through AI, this project pioneers digital health integration, shaping a healthier society.

7. ACKNOWLEDGEMENT

I might want to offer my most profound thanks to my guide, Ms. Geetanjali Tyagi for her significant direction, reliable consolation, convenient assistance and furnishing me with a fantastic air for exploring. All through the work, regardless of her bustling timetable, she has stretched out bright and warm help to me for finishing this exploration work.

8. REFERENCES

[1] Yoo, M., Na, Y., Song, H., Kim, G., Yun, J., Kim, S., ... & Jo, K. (2022). Motion estimation and hand gesture recognition-based human–UAV interaction approach in real time. Sensors, 22(6), 2513. [CrossRef]

[2] Saini, N., Bonetto, E., Price, E., Aamir, A., & Black, M. J. (2022). AirPose: Multi-view fusion network for aerial 3D human pose and shape estimation. IEEE Robotics and Automation Letters, 7(2), 1804-1811. [CrossRef]

[3] Lina, W., & Ding, J. (2020, July). Behavior detection method of OpenPose combined with Yolo network. In Proceedings of the International Conference on Communications, Information System and Computer Engineering (pp. 326-330). IEEE.

[4] Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2021). OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(1), 172-186. [CrossRef]

[5] Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016, September). Simple online and realtime tracking. In Proceedings of the IEEE International Conference on Image Processing (pp. 3464-3468). IEEE.

[6] Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In Proceedings of the IEEE International Conference on Image Processing (pp. 3645-3649). IEEE.

[7] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, June). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-787). IEEE Computer Society.

[8] Newell, A., & Deng, J. (2017, July). Stacked hourglass networks for human pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 483-492). IEEE Computer Society. [CrossRef]

[9] Sun, K., Xiao, B., Jiang, D., & Wei, Y. (2019, June). Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 569-578). IEEE Computer Society. [CrossRef]

[10] Fang, H., Xu, Y., Wang, Y., & Li, S. (2020, December). CocoPose: A lightweight and fast 2D human pose estimation system. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 536-543). IEEE. [CrossRef]

[11] Xiao, B., Lassner, C., Rochery, M., Geiger, A., & Xu, J. (2019, June). Image-based human pose estimation: A review. arXiv preprint arXiv:1902.09212

[12] Xu, X., Wang, L., & Tao, D. (2020, December). Lstm and attention based network for 3d human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 632-639). IEEE. [CrossRef]

[13] Martinez, J., Hossain, M. M., & Xu, D. (2020). A survey on human pose estimation using deep learning. Sensors, 20(16), 4686. [CrossRef]

[14] Fang, H., Xie, S., He, Y., Bao, C., & Deng, C. (2018). RMPE: Regional multi-person pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 3334-3343). IEEE Computer Society. [CrossRef]

[15] Sun, K., Xiao, B., Jiang, D., & Wei, Y. (2019). Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 569-578). IEEE Computer Society. [CrossRef]

[16] Chu, X., Pang, Y., & Li, H. (2020). Fully convolutional network for multiple-object pose estimation. IEEE Transactions on Circuits and Systems for Video Technology, 30(8), 2808-2820. [CrossRef]

Reserach_paper_Plag Report

ORIGINALITY REPORT

2%	1%	1%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	fastercapital.com Internet Source	1%
2	Submitted to University of Essex Student Paper	1%
3	Submitted to University of Nottingham Student Paper	1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		