A REPORT

ON

Recognizing human actions from RGB videos using pose estimation

By

Name (s) of the student (s)	Registration No.
Mounish Sai Gowtham M	AP21110010031
Lakshmi Sai Prakash P	AP21110010041
Sai Gopal L	AP21110010042
Khyathi Vardhan R	AP21110010168

Prepared in the partial fulfillment of the
Summer Internship Course

Under the guidance of

Dr. M Naveen Kumar Mahamkali

SRM UNIVERSITY, AP

(July, 2023)

Certificate

Date: 02-08-23

This is to certify that the work present in this Project entitled “Recognizing human actions from RGB videos using pose estimation ” has been carried out by Mounish Sai, Lakshmi Sai Prakash,Sai Gopal,Khyathi Vardhan under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology / Master of Technology in School of Engineering and Sciences.

Supervisor

(Signature)

Dr. M. Naveen Kumar Mahamkali

Designation - Assistant Professor

Affiliation - SRM University, AP

Acknowledgement:

I would like to take this opportunity to express my sincere gratitude and appreciation to all the individuals and organizations who have contributed to the successful completion of this project report titled "Recognizing human actions from RGB videos using pose estimation".

I am also thankful to the faculty mentor DR. M Naveen kumar Sir for their constant encouragement and constructive feedback, which have significantly enriched the quality of this project report.

Abstract :

Recognizing human actions from RGB videos using pose estimation is the name of the project. As a part of it MediaPipe is used for Extracting Joint Positions and Features. Mounish, Prakash, Gopal, and Khyathi are the members of the project group, which was led by Naveen Kumar Sir. The primary goal of the project's development is to extract joint locations from the RGB videos that are input from the photos, videos, and webcam feed. The project also determines the distance between the extracted joint positions in pixel units.

Table of Contents

1. Introduction :

Recognition of human actions is currently a prominent focus within the realm of computer vision research. The significance of human action recognition from RGB videos extends across diverse fields and applications, promising to deepen our comprehension of human behavior and enhance the efficacy of various systems. In recent years, there has been a growing interest in using skeleton data, which captures the joint positions of a human body, to perform human action recognition. This is because skeleton data is less sensitive to variations in appearance and can capture the temporal dynamics of human actions.

This project aims to develop a human action recognition system using pose estimation that utilizes computer vision techniques to extract joint positions and calculate distances between joints from various input sources, including videos, images, and real-time webcam feeds, which are the main inputs for creating the human action recognition system. By integrating the OpenCV and Mediapipe libraries with Python, the system will enable accurate and real-time analysis of human body movements. Our approach is based on recent advances in deep learning and is designed to overcome the challenges associated with recognizing complex and diverse human actions.

Because of its many applications, human pose estimation has emerged as a critical area of study in computer vision. Traditional methods heavily relied on manual feature engineering and complex algorithms, limiting accuracy and efficiency. With advancements in deep learning and computer vision libraries, such as OpenCV and Mediapipe, it is now possible to accurately estimate joint positions and calculate distances between them, making human pose estimation more accessible and efficient. The significance of this research work is to increase the performance of vision-based action recognition by extracting key points from RGB video sequences. It is noted that the extracted key points form a 2D skeleton pose

1.1 Outline of the Project:

- Setup and Environment Configuration
- Image and Video Processing
- Joint Position Estimation
- Distance Calculation
- Real-Time Pose Estimation and Output
- Testing

2. Main Text:

2.1 Method of Treatment:

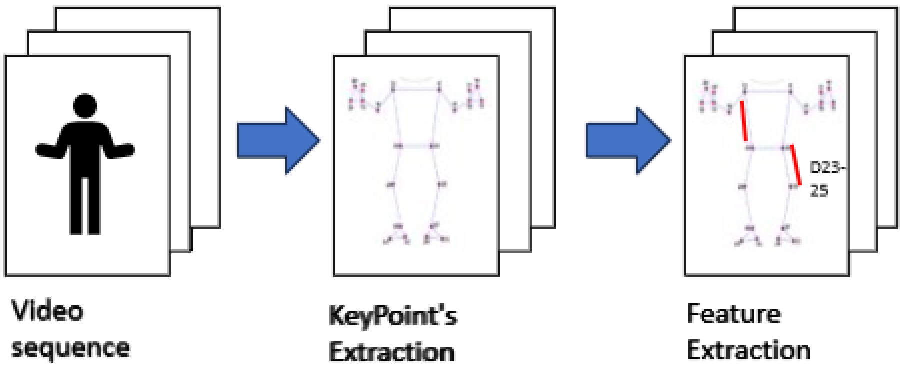


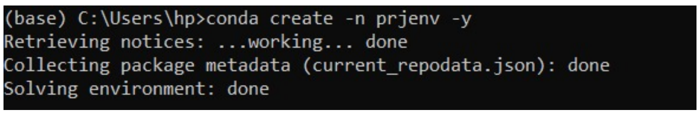
Fig 1. proposed scheme

The proposed approach includes gathering the video sequence , key points extraction and finally feature extraction. which will be explained in a detailed way in the upcoming sections.

2.2 Methodology

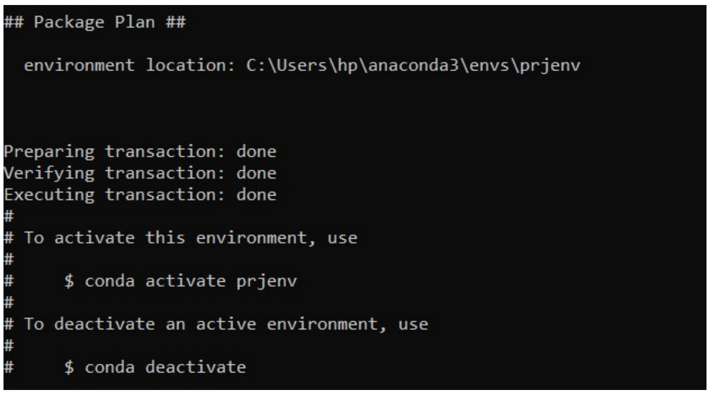
2.2.1 Environmental Setup:

- Initially Anaconda software should be installed on the local system.
- A new environment should be created on the anaconda prompt.



```
(base) C:\Users\hp>conda create -n prjenv -y
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

fig.2 creating a new environment



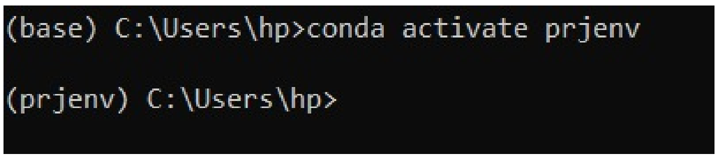
```
## Package Plan ##

  environment location: C:\Users\hp\anaconda3\envs\prjenv



Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate prjenv
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

fig. 3 Activating the environment



```
(base) C:\Users\hp>conda activate prjenv

(prjenv) C:\Users\hp>
```

fig. 4 New environment has been created

- Import the libraries that are necessary for execution of the code.

2.2.2 Libraries/Packages:

- **Pose Estimation Algorithm :**

MediaPipe pose is the Pose Estimation algorithm that has been used in this project. It is a part of the MediaPipe library, developed by Google and an open-source framework that provides real-time pose estimation solutions. MediaPipe Pose is designed to detect and track human body key points (joints) in images and videos, enabling applications such as human action recognition, gesture recognition, and more. The library is implemented in C++ and offers APIs for various programming languages, including Python.

Key features of MediaPipe Pose:

- The primary goal of MediaPipe Pose is to detect and track key body points in the human body, such as the nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles.
- This pose estimation model, used by MediaPipe Pose is based on a deep learning neural network, which enables it to perform on various devices, including CPUs, GPUs, and mobile devices.
- It is able to handle multi-person scenarios, that is it can detect and track the poses of multiple individuals in a single image or frame.
- MediaPipe Pose supports multiple platforms, making it accessible for both desktop and mobile applications.
- It is an open-source project, which means developers can access the source code, modify it, or contribute to the library's improvement.
- Complex multimodal applications can be built by combining MediaPipe pose with other Mediapipe solutions such as MediaPipe Hands, MediaPipe Holistic, etc.,

command line: pip install mediapipe

- OpenCV :

It is the well known library that is used to process the images and perform computer vision tasks, such as face detection, object detection, and much more. it can also be used for modifying the videos.

command line: pip install opencv-python

- Argparse :

Argparse is a python module that is used to parse the command line arguments.

- Math:

Math is a python library that is used to perform mathematical operations.

- Time:

Time is a python library that is used to perform various functions to work with time-related tasks such as retrieving the current time

- Numpy:

It is a python library which comes handy while working with arrays.

command line: pip install numpy

To evaluate our proposed methodology

- We are going to use the UT Kinect dataset.
- We are creating our own dataset of recording our own videos.

2.3 Working:

In this section, we will cover the detailed steps of how the project works, from getting the inputs to creating a feature vector that encapsulates the location information of the video.

2.3.1 Input files:

- The project starts by taking video files or webcam channel as input. OpenCV, an open source computer vision library, is used for this.
- Video files and webcam streams are basically a sequence of frames, where each frame is a 2D matrix of pixels representing the colors of the image.
- The pixel matrix, often called image matrix or pixel board, is the basic information structure used to represent the colors of digital images.
- This is a two-dimensional grid of pixels, where each pixel corresponds to a small point image and stores information about its color.
- The pixel matrix is necessary to understand and process the colors of the image.
- Each pixel contains color information which is a combination of red, green and blue components (RGB).
- The RGB color model is the most common color representation used in digital images.
- It uses three main color channels: red, green and blue.
- Colors are created by varying the intensity of these three channels.
- The pixel matrix is arranged in a grid with rows and columns.
- Each cell of the matrix corresponds to one pixel.
- The dimensions of the matrix are determined by the image resolution, e.g. 1920x1080 Full HD picture.
- Each pixel of the matrix stores color information in the form of three values: intensity of red (R), green (G) and blue (B).
- In general, each channel is represented by 8 bits, allowing 256 intensity levels (0 up to 255). This is often referred to as 8-bit color depth.
- Image colors are obtained by combining different intensities R, G and B.
- Each color space has its own way of representing colors in the pixel matrix.

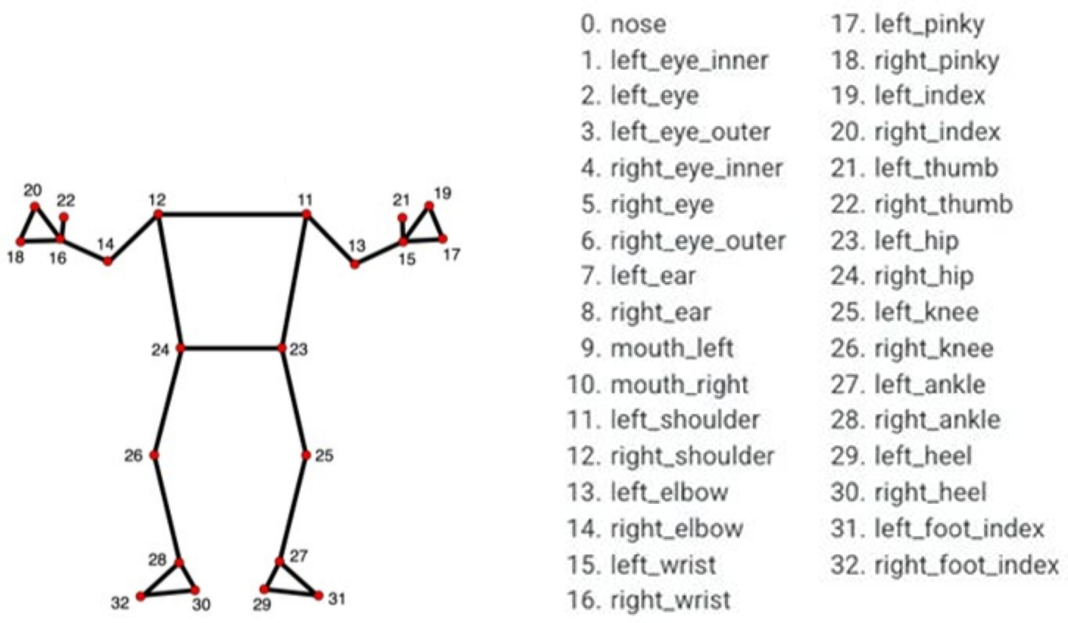
2.3.2 Conversion to RGB format:

Mediapipe Pose works with images in RGB (red, green, blue) format, while OpenCV reads video files in BGR (blue, green, red) format by default. Input frames must be converted from BGR to RGB to be compatible with Mediapipe Pose. This conversion is done using OpenCV's cvtColor () function with the COLOR_BGR2RGB flag. This simple operation switches the red and blue color channels.

- The cvtColor () function is an important function provided by the OpenCV library (OpenSource Computer Vision Library) for color space conversion in image processing.
- It allows you to convert an image from one color space to another. A commonly used conversion is from BGR color space to RGB color space.
- This is a key feature when working with digital images, as different algorithms and libraries may expect or produce images in different color spaces.
- COLOR_BGR2RGB is one of the many predefined conversion codes (flags) of OpenCV. This specifies that the conversion should be from the BGR color space to the RGB color space.
- In the BGR color space, pixel values are arranged as blue, green, red. Instead, in the RGB color space, pixel values are arranged as red, green, blue.
- This flag basically rearranges the color channels, swapping red and blue to convert the image from BGR to RGB.

2.3.3 Identification of landmarks:

Mediapipe Pose is then used to detect human body landmarks or joints in each frame of the video. These landmarks correspond to certain points on the human body, such as shoulders, elbows, wrists, hips, knees and ankles. Facial landmarks (normal features 0-10) are not considered in this project.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Fig 5. skeleton structure representation

2.3.4 Drawing connections:

Using the detected landmarks, connections are made between certain pairs of joints and the skeletal structure of the human body is effectively created in each frame. This step helps to visualize the results of the pose estimation in the video frames, which makes the pose easier to understand and interpret.

2.3.5 Distance matrix from the coordinate points:

The coordinates (x, y) of each connection point are extracted from the observed landmarks. These coordinates represent the spatial location of each joint in the frame and provide precise information about the location of the person shown in the video.

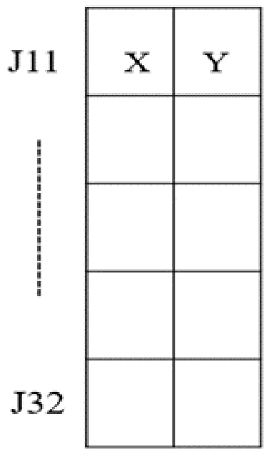


Fig 6. Image representation of coordinates

2.3.6 Distance calculation:

For each frame of the video, the distance between all connection pairs is calculated. The distance between two joints (x1, y1) and (x2, y2) is calculated using the Euclidean distance formula: "distance = sqrt ((x2 - x1)^2 (y2 - y1)^2)". This step creates a matrix where each element represents the distance between two joints of the given frame.

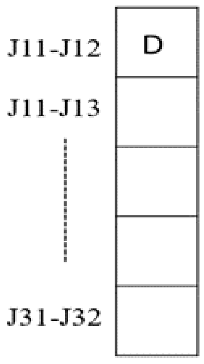


Fig 7. Image representation of the distance of a single frame

2.3.7 Structure of the feature matrix:

All distance matrices obtained for each frame are combined into a single matrix. The size of this matrix is 231 x M, where M is the number of video frames. The number 231 means the total number of unique common combinations (21 joints and 2 combinations per common pair, no diagonal).

2.3.8 Feature vector:

To obtain a more compact representation of all pose information in the video, the feature matrix is transformed into a 1D matrix known as a feature vector. This transformation is achieved by concatenating the rows of the matrix into a single vector. The resulting feature vector is a short but comprehensive summary of the pose information of the video, making it suitable for further analysis and processing.

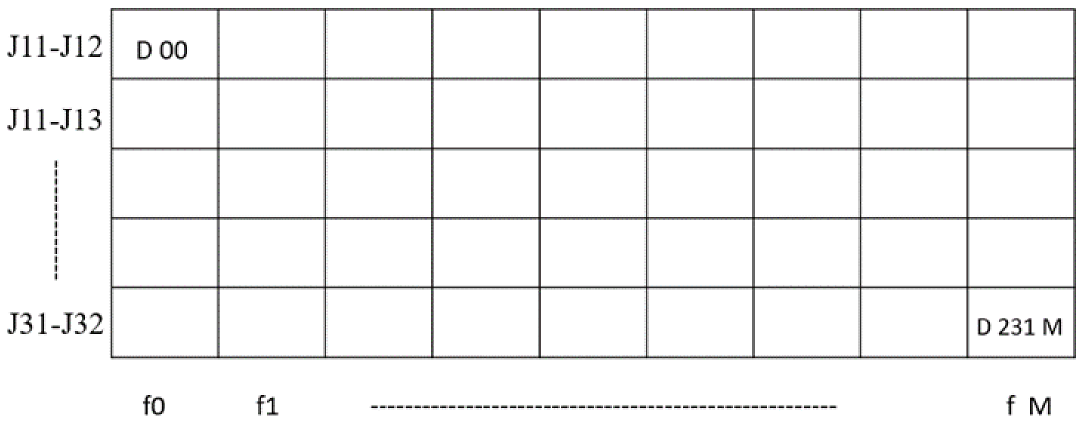


Fig8. Image representation of feature matrix

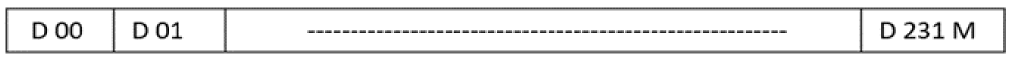


Fig9. Image representation of feature vector

Future Work

Training ML model

- We are having a dataset consisting of 10 videos and each video is performed twice by 10 subjects. This means we are having a total of 200 action samples in the dataset.

- Here we are going to train and evaluate a machine learning model. We need to split the dataset into two parts, they are Training set and Testing set.

Training set:-

This was a subset of dataset where we are going to train the 100 training samples

Testing set:-

The other subset of the dataset will be used to evaluate the performance of the trained model. The remaining 100 samples are used for the testing.

Action Comparison:-

In both training and testing each action is associated with a specific video.

In the training set we are comprehending the sample and we are comparing it with the testing set. Therefore the training model is predicting the action in frame.

By this it will display the name of the action video that has been compared.

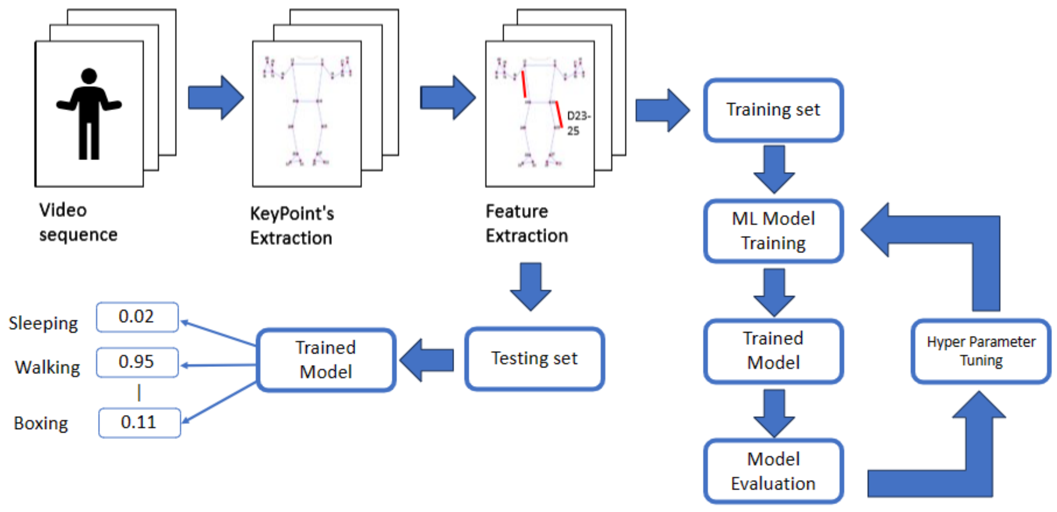
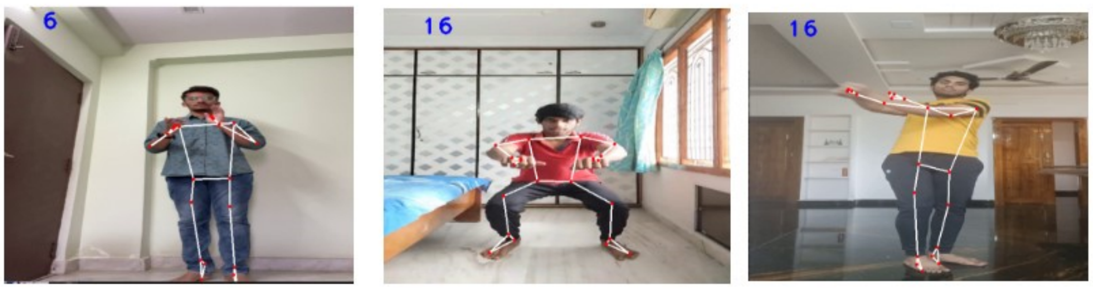
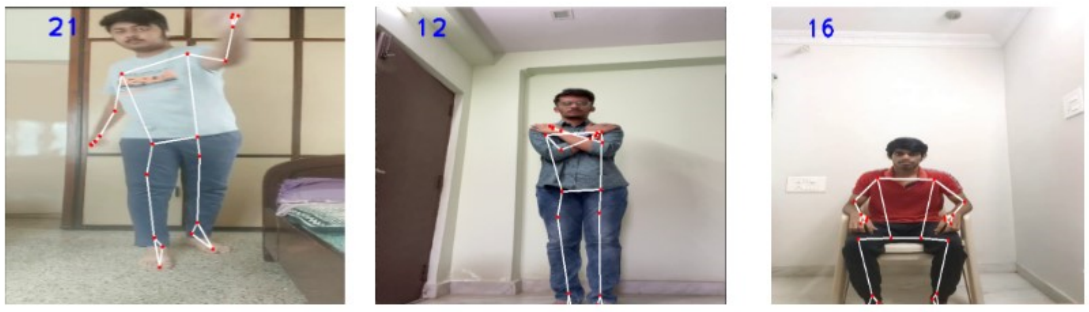


Fig10. ML model

Outcomes



1.Claps *2.squat* *3.Baseball swing*

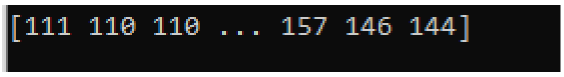


4.Throw *5.Arm Cross* *6.Sit to Stand*

Fig 11. overlaid landmarks representation

The figure 11 depicts how landmarks are overlaid on the videos

- The above picture is the outcome where the distances between all the joints of every frame is stored in a 1D array which is a feature vector.



```
[111 110 110 ... 157 146 144]
```

Fig 12. feature vector

Source Code: code

4. Conclusion:

The goal of this project is to use pose estimation, which determines joint positions accurately and estimates the distances between them to obtain the feature vector, to develop a flexible and effective human action recognition system. With potential applications across several domains, our goal is to enable accurate and real-time analysis of human body movements through the use of Python's Mediapipe and OpenCV libraries. Our goal is to use this project to further pose estimation technology and make it more useful across a range of fields.

5. References

posenet - https://github.com/rwightman/posenet-python

openpose - https://viso.ai/deep-learning/openpose/

UT Kinect dataset - https://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html