# TRIBHUVAN UNIVERSITY

# INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering


Minor Project Final

On

## PERSONAL TRAINER USING HUMAN POSE VISUALIZATION

[Code No: CT654]

By: -

Abinash Nath Pandey – KAT076BCT007

Kathmandu, Nepal

Falgun, 2079

March, 2023

# TRIBHUVAN UNIVERSITY

# INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

## “PERSONAL TRAINER USING HUMAN POSE VISUALIZATION”

[Code No: - CT654]

**PROJECT REPORT SUBMITTED TO**

**THE DEPARTMENT OF COMPUTER ENGINEERING**

**IN PARTIAL FULFILLMENT FOR THE REQIUREMENT**

**FOR THE BACHELOR OF ENGINEERING**


By: -

Abinash Nath Pandey – KAT076BCT007

Kathmandu, Nepal

Falgun, 2079

March, 2023

# TRIBHUVAN UNIVERSITY

# INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

## CERTIFICATE

The undersigned certify that they have read and recommended to the Department of Computer Engineering, a minor project work entitled "**Personal Trainer using Human Pose Visualization**" submitted by **Abinash Nath Pandey-- KAT076BCT007** in partial fulfillment of the requirements for the degree of Bachelor of Engineering.

---

**Er. Anju Khanal**

(Project Supervisor)

Department of Computer Engineering

Kathmandu Engineering College

---

**Er. Krista Byanju**

(Project Coordination)

Department of Computer Engineering

Kathmandu Engineering College

---

**Er. Sudeep Shakya**

(Head of Department)

Department of Computer Engineering

Kathmandu Engineering College

# ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my project coordinator **Er. Krista Byanju, Er. Sudeep Shakya Head of Department, Er. Kunjan Amatya Deputy Head of Department** of Computer Engineering Department, **Er. Hirendra Man Pradhan** and Principal of Kathmandu Engineering College, for providing me with this golden opportunity to do this wonderful project on the topic of "**Personal Trainer using Human Pose Visualization"**. This project has helped me to reflect on my ways of learning new things and work in any sort of projects.

I would like to thank to all teachers including **Year Coordinator Er. Nabin Neupane** also **Project Supervisor Er. Anju Khanal** for giving me feedbacks and guidance for this project. I am really grateful to all the teachers who helped me in providing information on various topics related to the project and for encouraging me to strive to the completion of the project.

Without the help of everyone, this project couldn't be completed. So, I am really grateful to everyone for providing guidance and support.

# ABSTRACT

Fitness has evidently become an unceasing trend because of the shift of focus to a healthier lifestyle and the promotion of the new offerings. This become a part of people's lifestyle. Different technologies have been introduced so far to help improve the human body. Artificial Intelligence is one of the technologies that has invaded practically all functional areas of business over the years and has become a popular project today in the field of computer vision. Pose estimation is among the most popular solutions that AI has to offer; it is used to determine the position and orientation of the human body given an image containing a person [1]. The human pose estimation is using AI/ ML in which the system is fed with sample data or trained models and hence can hence localize joints in the human body over a video or an image. This developed a system prototype device that functions as a friendly personal exercise trainer that understands the physical techniques and strategies involved to achieve his/her success. This offers a workout program that monitors the user'sworkout in real time, counts down the reps and alerts the user when the move is performedincorrectly. The System is designed/ developed with respect to single users. It does not require the background to be of any plain color; thus, can detect and deliver required resultsin real time. The human pose estimation method is based on CNN, MediaPipe and LinearRegression. This shows the joints of the bones with some random values and provides counting of right and wrong values along with posture of exercise while performing it.

***Key words**: Fitness, Artificial Intelligence, Machine Learning, Human Pose Estimation, Mediapipe*

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| **3D:** | 3-Dimensional |
| **AI:** | Artificial Intelligence |
| **CNN:** | Convolutional Neural Network |
| **CPU:** | Central Processing Unit |
| **CUDA:** | Compute Unified Device Architecture |
| **GPU:** | Graphics Processing Unit |
| **GUI:** | Graphical User Interface |
| **MATLAB:** | Matrix Laboratory |
| **ML:** | Machine Learning |
| **MMX:** | Matrix Math Extension |
| **OPENCL:** | Open Computing Language |
| **PC:** | Personal Computer |
| **RGB:** | Red Green Blue |
| **ROI:** | Region of Interest |
| **SMID:** | Single Instructions Multiple Data |
| **SSE:** | Streaming SMID Extension |
| **STL:** | Standard Template Library |

# CHAPTER 1: INTRODUCTION

## 1.1. BACKGROUND

Exercises such as squats, deadlifts, and shoulder presses are beneficial to health and fitness, but they can also be very dangerous if performed incorrectly. The heavy weights involved in these workouts can cause severe injuries to the muscles or ligaments. Many people work out and perform these exercises regularly but do not maintain the proper form (pose). This could be due to a lack of formal training through classes or a personal trainer or could also be due to muscle fatigue or using too much weight. For my course project, this seek to aid people in performing the correct posture for exercises by building Pose Trainer, a software application that detects the user's exercise pose and provides useful feedback on the user's form, using a combination of the latest advances in pose estimation and machine learning. Goal for Pose Trainer is to help prevent injuries and improve the quality of people's workouts with just a computer and a webcam.

The first step of Pose Trainer uses human pose estimation, a difficult but highly applicable domain of computer vision. Given visual data, which could be an RGB image and/or a depth map, a trained model predicts a person's joints as a list of skeletal key points. Pose estimation is critical for problems involving human detection and activity recognition, and can also aid in solving complex problems involving human movement and posture. We use a state-of-the-art pose estimation deep neural network, Mediapipe [2], within Pose Trainer for inference. The second part of our application involves detecting the quality of a user's predicted pose for a given exercise. This approach this usingheuristic-based and machine learning models, using the poses and instruction of personal trainers and other qualified professionals as the ground truth for proper form.

Our full application consists of the previously described two main components, combined into an end-to-end application that can take a video of an exercise and provide useful exercise form feedback to the user. This present a Pose Trainer desktop application: future work involves moving Pose Trainer to mobile devices to create a true on-the-go personal trainer.

## 1.2. PROBLEM STATEMENTS

Exercising without any knowledge can lead to serious injuries. Moreover, the training sessions will not be as effective if you don't know what type of exercise is suitable for you. There are many cases where exercise has done more harm than good. And there are even more cases where people did not get the results that they desired. This is because of poor knowledge on the topic. A personal trainer can help in this situation but they charge a lot and may not be affordable for everyone.

## 1.3. OBJECTIVES

- To design and implement a system that monitors the user's exercise in real time.
- To design and implement a system that provides users with some exercise plans and tracks user’s progress.

## 1.4. SCOPES AND APPLICATIONS

### 1.4.1. SCOPES

This project takes into consideration and covers the following areas:

- The advent of AI and its subsets including computer vision involved in this project is helpful in modernizing fitness industry at an unprecedented rate.

- Gyms can incorporate this system to increase efficiency and speedup the fitness achieving rate.

- In remote areas, where fitness knowledge is less and there is no availability of fitness centers, this project can educate the people about importance of fitness and even help them in achieving it.

### 1.4.2. APPLICATIONS

This project takes into consideration and covers the following areas:

- Pose estimation can be used to track Human activities such as Walking, Running Sleeping, Drinking. It provides some information about a person. Activity estimation can enhance security and surveillance systems.

- One of the most interesting applications of human pose estimation is Motion Transfer. We've seen in movies or games, 3D graphics character's physical movement is like real humans or animals. By tracking the human pose, the 3D rendered graphics can be animated by the human's movement.

- To train the movement of a robot, human pose estimation can be used. Instead of manually programming a robot to follow a certain path, a human pose skeleton is usedto train the robot's joint movement

- Nowadays VR or Virtual reality gaming is very popular. In virtual reality gaming, a 3d pose is estimated by one or more cameras, and the game character moves according to the human's action [3].

# CHAPTER 2: LITERATURE REVIEW

## 2.1. RESEARCH AND RELATED WORKS

This identify several state-of-the-art methods for pose estimation that accurately estimate human poses under a variety of sensor configurations, shots, and counts of individuals per shot. Toshev et al. [3] were the first to use a deep neural network to improve pose detection, finding the location of each body joint using regression on CNN features. Newell et al. [4] introduce a stacked hour glass neural network architecture that uses repeated bottom-up and top-down processing to achieve accurate single pose predictions. Wei et al. [5] propose a different architecture, using multiple convolution networks to refine joint estimates over sequential passes. Instead of RGB camera data, Shotton et al. [6] use single depth maps captured by the Microsoft Kinect to predict 3D positions of joints through an object recognition approach. Bogo et al. [7] estimate 3D pose, as well as 3D mesh shape, using just single RGB images.

Pose estimation allows us to analyze the static posture of humans, which will provide valuable information regarding posture correctness. Zell et al. [8] use an interesting approach for analysis of physical movements, where the body is represented as a mass spring system and used to find the forces and torques that travel through the joints of the body. We have found that, by using exercise specifications and feedback from professionals, we can take a simpler approach to physical analysis, analyzing the angles and distances between joint key points to provide important feedback to users without needing a full physical simulation.

3D hand pose estimation is a very challenging task due to the high dexterity of the humanhand (i.e., 21 degrees of freedom). We briefly review the state-of-the-art of 2D and 3D pose estimation methods that were successful over the past few years. The successful breakthrough in pose estimation was related to human pose estimation as in (Tompson etal. 2014) and (Toshev and Szegedy. 2014). The idea of belief maps was mentioned by (Wei et al., 2016) for 2D human pose estimation applying convolutional pose machines. Many papers were published based on this idea of heat maps to improve the accuracy and to provide further extensions to 3D (Garcia-Hernando et al., 2018).

(Tompson et al., 2014) presented the idea of hand pose recovery implementing CNNs in real-time inferring intermediate heat map features to extract accurate 3D pose with the help of an inverse kinematic model on depth datasets. (Wan et al., 2017) proposed a deep 3D hand pose estimation approach that uses depth images. In this approach, depth feature maps from CNNs were divided into regions to form a tree structured region ensemble network, these regions are passed through fully connected layers for 3D coordinate regression and for better accuracy.

One of the best state-of-the-art techniques for heat map generation is mentioned in (Newell, Yang, and Deng, 2016). The authors use a stacked hourglass network to retain 2D heat map information. (Wan et al., 2017) also worked on a similar idea of heat maps on depth images for 2D and3D, in addition to 3D directional vectors for 3Dregression of hand pose. The further extensions to stacked hourglass network can be observed in (Zhou et al., 2017). The 2D CNNs are trained to infer heat maps and the authors extend the network to output the 3D pose by adding a regression network [9].

Steven Chen and Richard R. Yang (2020) [10] introduced a Pose Trainer, an end-to-end computer vision application that used a pose estimation via geometry, and machine learning to provide personalized feedback on fitness exercise form. They used the output of pose estimation to evaluate videos of exercises through human pose key points. They worked with four different exercises, recording training videos for each, and used both geometric heuristic algorithms to provide personalized feedback on specific exercise improvements, as well as machine learning algorithms to automatically determine posture correctness using only labeled input videos.

# CHAPTER 3: RELATED THEORY

## 3.1. ALGORITHMS AND TERMS

### 3.1.1. MEDIAPIPE

MediaPipe is Google's open-source framework, used for media processing. It uses a new approach to human pose perception, called BlazePose [11]. BlazePose provides human pose tracking by employing machine learning (ML) to infer 33, 2D landmarks of a body from a single frame. In contrast to current pose models based on the standard COCO topology, BlazePose accurately localizes more key points, making it uniquely suited for fitness applications.

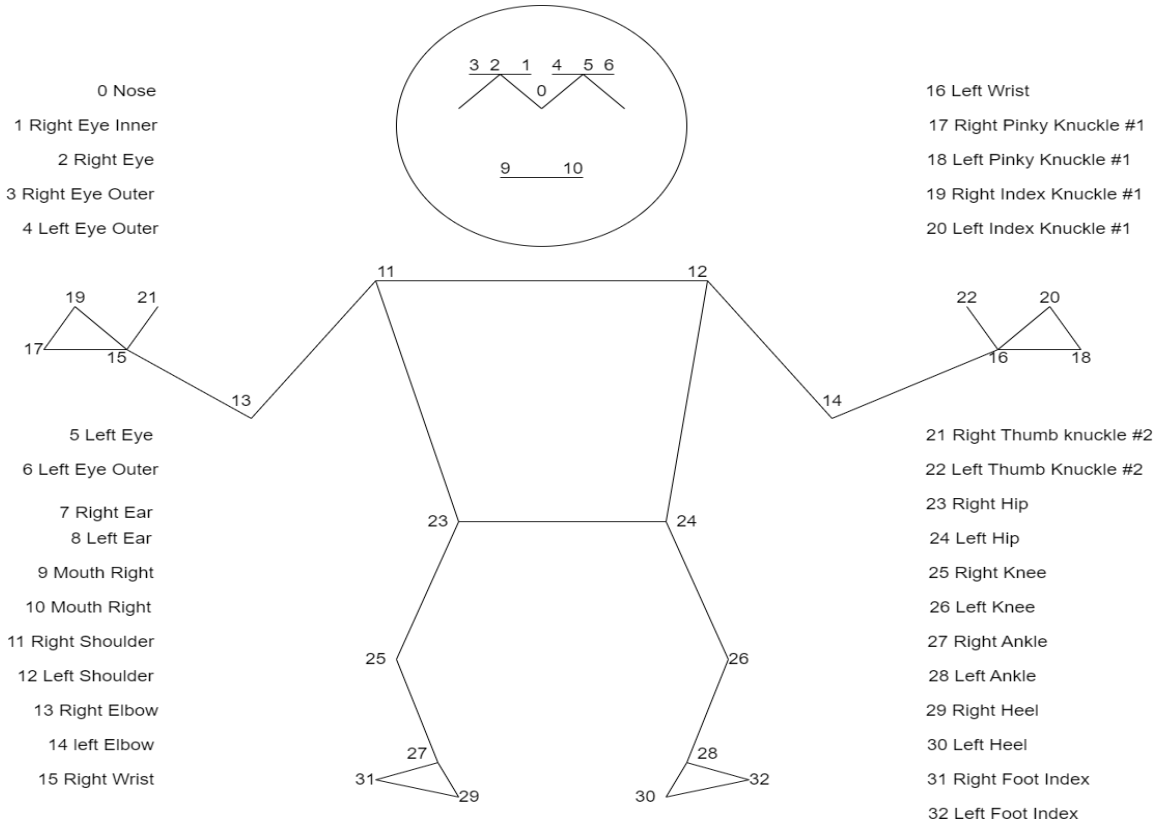


*Figure 3.1.1: Key points topology as COCO*

*(Colored with green) Supersets [11]*

### 3.1.2. BLAZEPOSE ML PIPELINE

For pose estimation, Mediapipe utilizes their proven two-step detector-tracker ML pipeline. Using a detector, this pipeline first locates the pose region-of-interest (ROI) within the frame. The tracker subsequently predicts all 33 pose key points from this ROI. Note that for video use cases, the detector is run only on the first frame. For subsequent frames this derive the ROI from the previous frame's pose key points as discussed below.

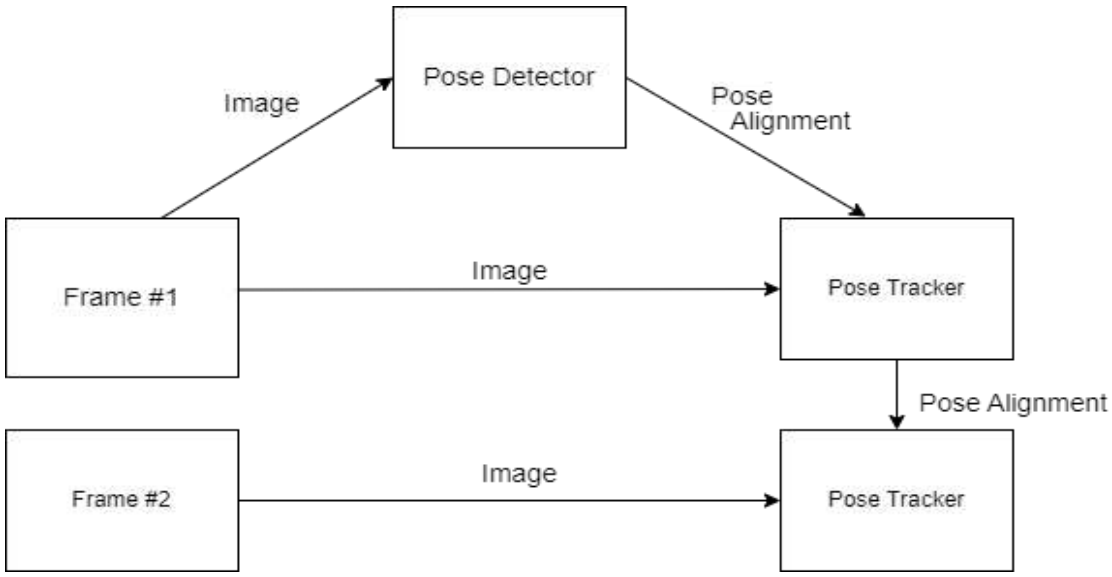


*Figure 3.1.2.1: Human pose estimation*

*Pipeline overview [11]*

- Using a detector, the pipeline first locates the person/pose region-of-interest (ROI) within the frame.
- The tracker subsequently predicts the pose landmarks and segmentation mask within the ROI using the ROI-cropped frame as input.

For video use cases the detector is invoked only as needed, i.e., for the very first frameand when the tracker could no longer identify body pose presence in the previous frame. For other frames the pipeline simply derives the ROI from the previous frame's pose landmarks.

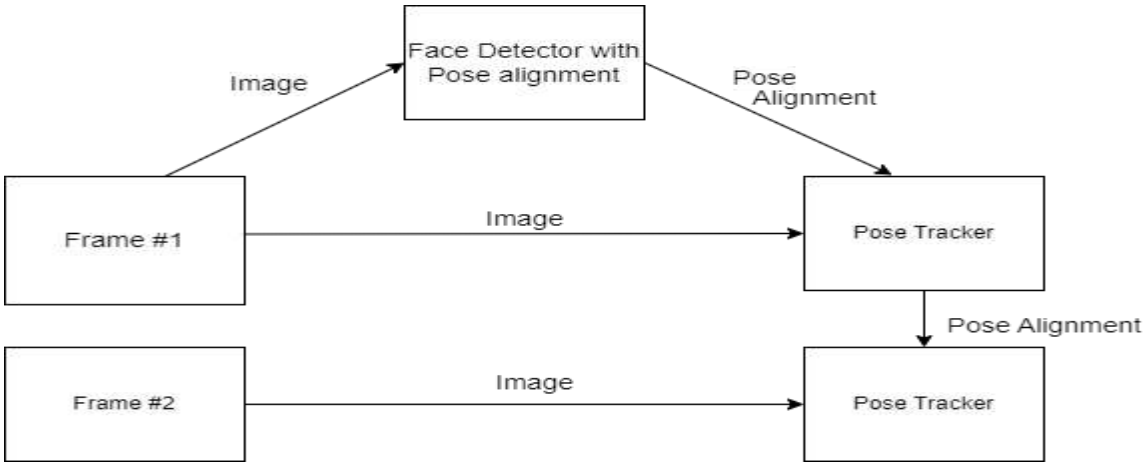


*Figure 3.1.2.2: Human pose estimation pipeline*

*Detailed view [12]*

The pipeline consists of a lightweight body pose detector followed by a pose tracker network. At the first stage, an object detection model is run to locate the presence of a human or to identify their absence. After the person has been detected, the pose estimation module can process the localized area containing the person and predict the position of the key points. The object detection module (face detector in the case of Blaze Pose) can be used only to kick start the pose tracking in the first frame while the subsequent tracking of the person can be done using exclusively the pose predictions after some pose alignment, parameters for which are predicted using the pose estimation model. The tracker predicts key point coordinates, the presence of the person on the current frame, and the refined region of interest for the current frame. When the track indicates that there is no human present, it re-runs the detector network on the next frame.

### 3.1.3. POSE DETECTION

For real-time performance of the full ML pipeline consisting of pose detection and tracking models, each component must be very fast, using only a few milliseconds per frame. To accomplish this, the strongest signal to the neural network about the position of the torso which is the person's face (due to its high-contrast features and comparably small variations in appearance). Therefore, mediapipe achieves a fast and lightweight pose detector by making the strong (yet for many mobile and web applications valid) assumption that the head should be visible for our single-person use case.

Face detector is trained, inspired by MediaPipe another model Blaze Face, and used as a proxy for a pose detector that only detects the location of a person within the frame. In contrast to the Face Mesh and MediaPipe Hand tracking pipelines, where we derive the ROI from predicted key points, for the human pose tracking mediapipe explicitly predicts two additional virtual key points that firmly describe the human body center, rotation and scale as a circle. Inspired by Leonardo's Vitruvian man, it predicts the midpoint of a person's hips, the radius of a circle circumscribing the whole person, and the incline angle of the line connecting the shoulder and hip midpoints. This results in consistent tracking even for very complicated cases, like specific yoga asanas. The figure below illustrates the approach.

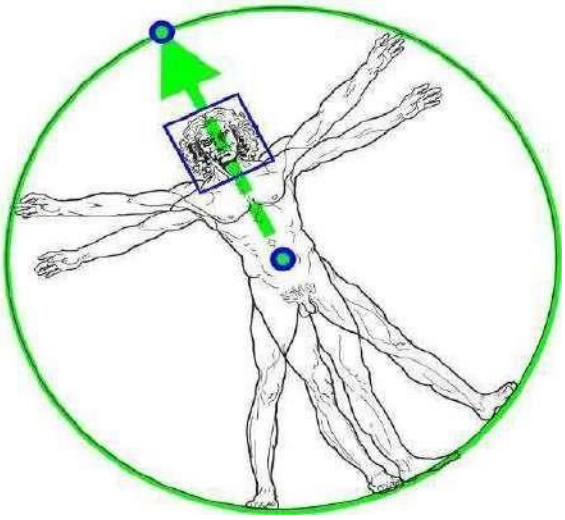

Figure 3.1.3: Vitruvian man aligned via pose detector vs

Face detection boundary box [11]

### 3.1.4. POSE TRACKER

The pose estimate component of the pipeline predicts the location of all 33-person key points with three degree of freedom each (x, y location and visibility) plus the two virtual alignment key points, described above. Unlike current approaches that employ compute-intensive heat maps prediction, the model uses a regression approach that is supervised by a combined heat map/offset prediction ofall key points, as shown below.

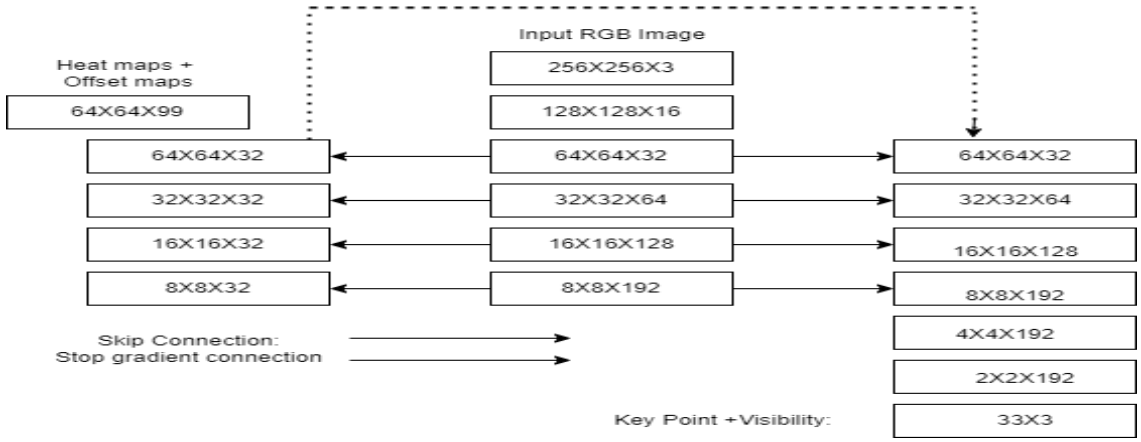


*Figure 3.1.4.1: Tracking network architecture regression with heat map supervision [11]*

Specifically, during training we first employ a heat map and offset loss to train the center and left tower of the network. We then remove the heat map output and train the regression encoder (right tower), thus, effectively using the heat map to supervise a lightweight embedding.

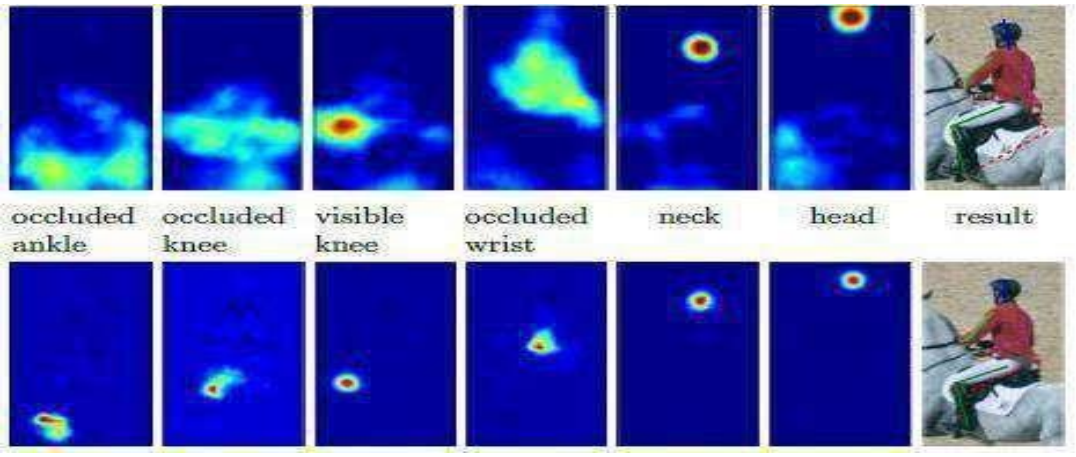


*Figure 3.1.4.2: Sample Heat maps [13]*

BlazePose adopts a combined heat map, offset, and regression approach, as shown in Figure 4. We use the heat map and offset loss only in the training stage and remove the corresponding output layers from the model before running the inference. Thus, we effectively use the heat map to supervise the lightweight embedding, which is then utilized by the regression encoder network. This approach is partially inspired by Stacked Hourglass approach of Newell et al. [14], but in this case, a tiny encoder- decoder heat map-based network and a subsequent regression encoder network is stacked. An encoder-decoder network architecture predicts the heat maps for all joints, followed by another encoder that regresses directly to the coordinates of all joints.

## 3.2. LINEAR REGRESSION

Linear Regression is an ML algorithm used for supervised learning. Linear regression performs the task to predict a dependent variable (target) based on the given independent variable(s). So, this regression technique finds out a linear relationship between a dependent variable and the other given independent variables.

In Regression, we plot a graph between the variables which best fits the given data points, using this plot, the machine learning model can make predictions about the data. If there is a single input variable(x), such linear regression is called simple linear regression. And if there is more than one input variable, such linear regression is called simple linear regression. The linear regression model givesa sloped straight line describing the relationship within the variables.

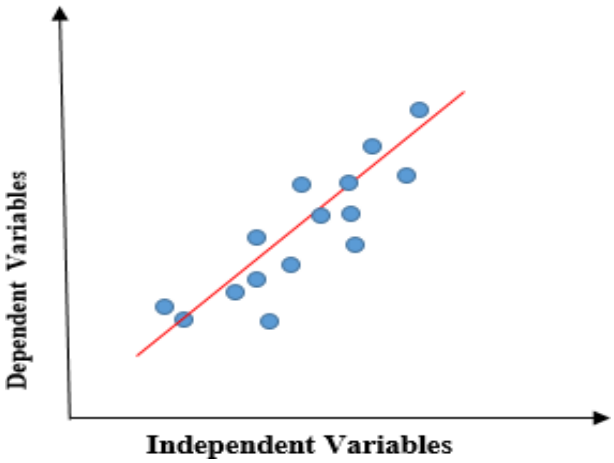


Figure 3.2: Regression Graph [15]

The above graph presents the linear relationship between the dependent variable and the independent variables. When the value of x (independent variable) increases, the value of y (dependent variable) is likewise increasing. The red line is referred to as the best fit straight line. Based on the given data point, we try to plot a line that models the points best. To calculate best-fit line linear regression uses a traditional slope-intercept form.

$$y = mx + b \ldots\ldots\ldots (1)$$

$$y = a_0 + a_1.x \ldots\ldots\ldots\ldots (2)$$

Where,

y = dependent variable

x = independent variable

$a_0$ = intercept of the line

$a_1$ = linear regression coefficient

### 3.2.1. ENCODER-DECODER NETWORK

An Encoder-Decoder Neural Network is a modular neural network that consists of an encoder network and a decoder network. A popular image segmentation model is based on an encoder network structure. In the encoder part, down sampling is adopted to reduce the input spatial resolution, so as to generate a lower resolution feature mappings (which is computationally efficient and can effectively distinguish different categories); in the decoder part, these feature representations are up sampled and restored to the full resolution.

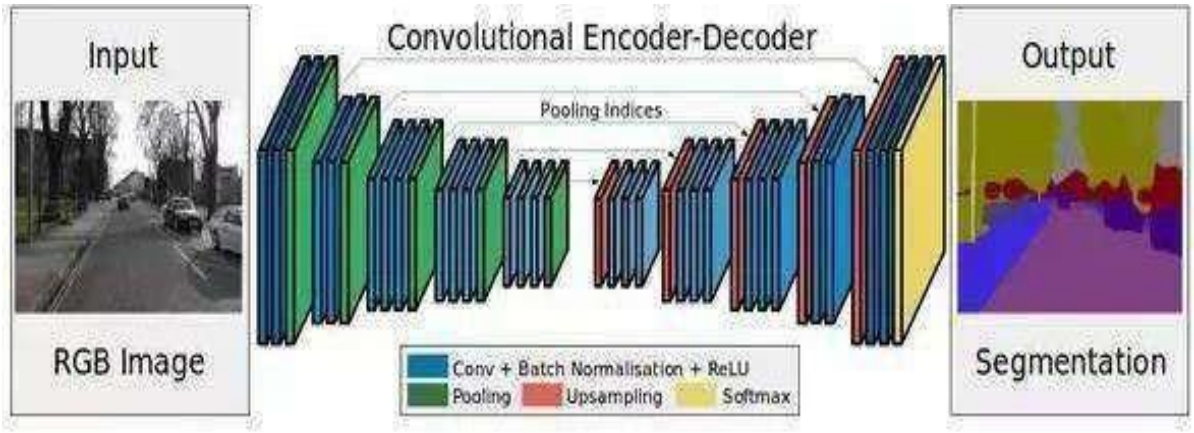


*Figure 3.2.1.1: Encoder Decoder CNN Network [16]*

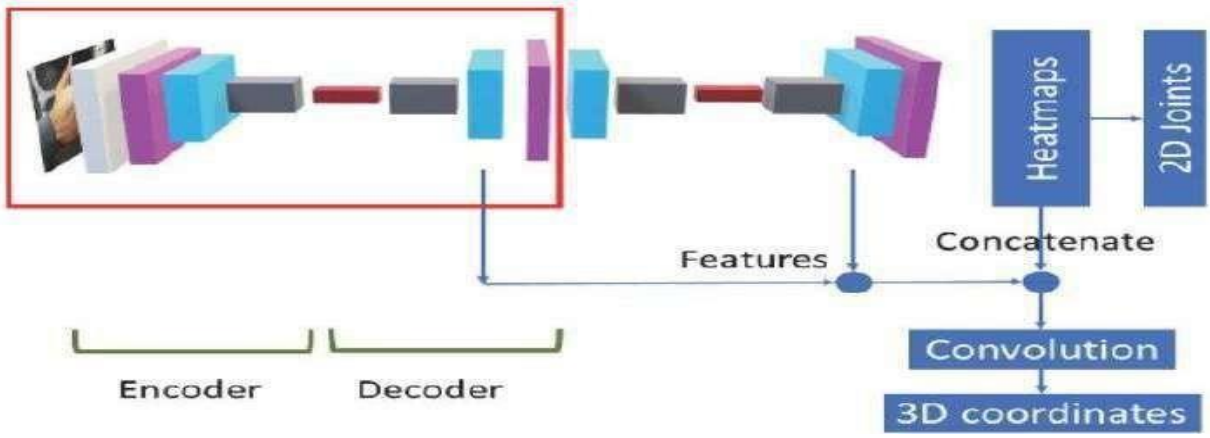


*Figure 3.2.1.2: Stacked Encoder Decoder network [16]*

### 3.2.2. CNN

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network, we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

The role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction. The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc. Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a "class." For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains any of those animals.

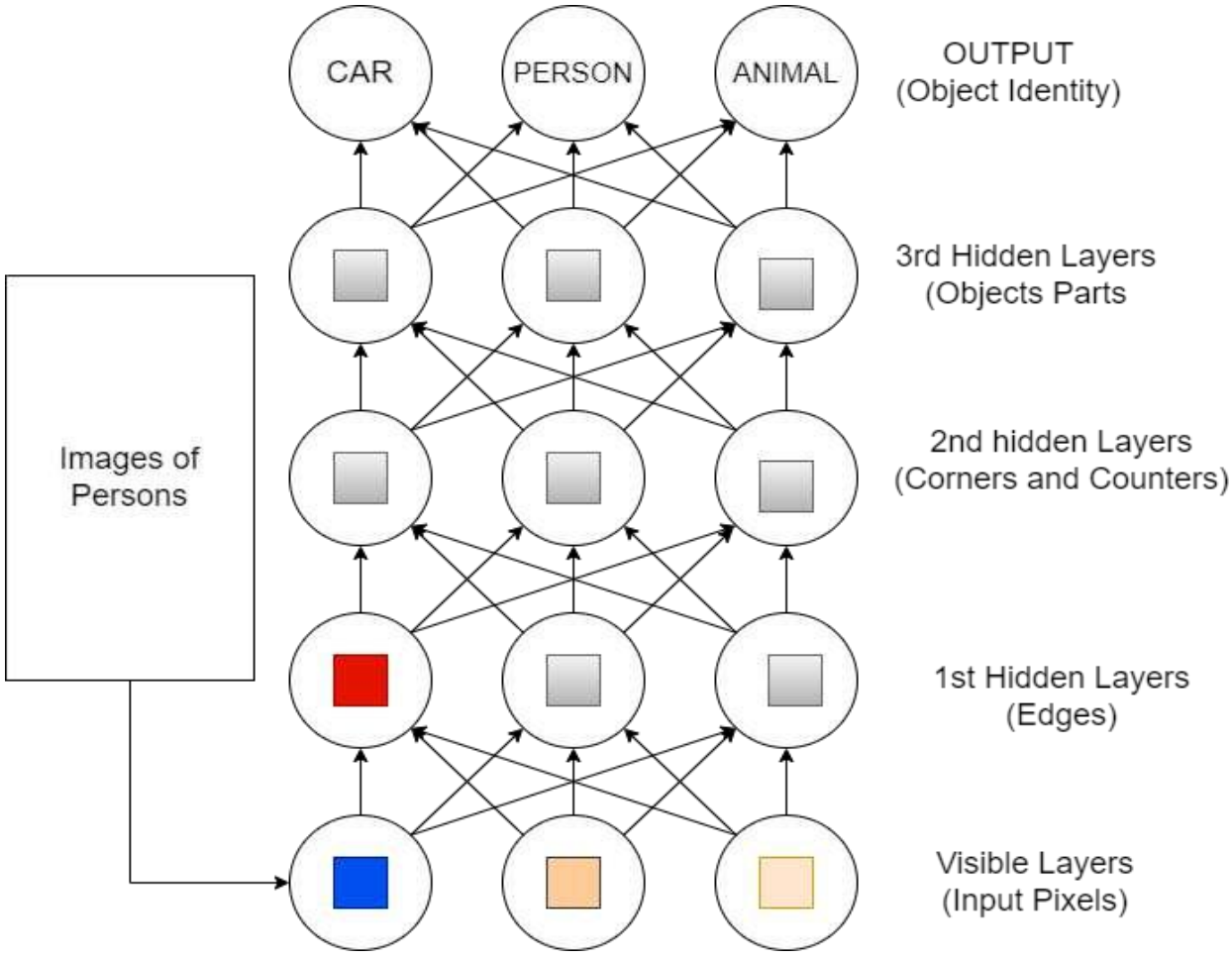


*Figure 3.2.2: CNN Classifier*

## 3.3. TECHNICAL DETAILS

### 3.3.1. PYTHON

Python is a powerful general-purpose programming language. It is used in web development, data science, creating software prototypes, and so on. It was created by Guido van Rossum and first released in 1991. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**FEATURES**

- Easy to Learn**:** Python is easy to learn and use. It is developer-friendly and high-level programming language.

- Expressive Language: Python language is more expressive means that it is more under standard able and readable.

- Interpreted Language: Python is an interpreted language i.e.; interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for engineering.

- Cross-Platform Language: Python can run equally on different platforms such as Windows, Linux, UNIX and Macintosh etc. So, we can say that Python is a portable language.

- Free Open Source: Python language is freely available at official web address. The source-code is also available. Therefore, it is open source.

- Large Standard Library: Python has a large and broad library and provides rich set of module and functions for rapid application development.
- Extensible: It implies that other languages such as C/C++ can be used to compile the codes and thus, it can be used further in our python code.

### 3.3.2. OPENCV

OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly aimedat real-time computer vision. It was originally developed by Intel, and was later supported by WillowGarage then it sees (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, findsimilar images from an image database, remove red eyes from images taken using flash, follow eyemovements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL containers.

### 3.3.3. NUMPY

NumPy is a library for the Python programming language, adding support for a large, multi-dimensional arrays and matrices, along with collection of high-level mathematical functions to operate on these arrays. It is a fundamental package for scientific computing in Python.

## 3.4. FEASIBILITY

### 3.4.1. TECHNICAL FEASIBILITY

Personal trainer using human pose estimation is technically feasible as it is powered by computer vision. Computer vision is a mature technology and is widely used in different areas. All the technologies required are easily available. Other publicly available data or datasets will be used if needed. Thus, the project is technically feasible.

### 3.4.2. FINANCIAL FEASIBILITY

The freely available software's and data will be enough for this project. Thus, this project requires zero cost for its completion.

### 3.4.3. SCHEDULE FEASIBILITY

The availability of libraries, tutorials, frameworks related to this project and the familiarity of the team members with these technologies makes the schedule feasible. So, the project can be completed within the marked deadline.

### 3.4.4. OPERATIONAL FEASIBILITY

The project will be a desktop application, and the user will require only a PC and a webcam. Thus, it is feasible to operate in real-time.

# CHAPTER 4: METHODOLOGY

## 4.1. PERSONAL TRAINER PROCESS WORKFLOW

There are different steps involved in making Personal Trainer. The process involves data collection, pre-processing, feature selection, classification techniques application, and evaluating performance measures.

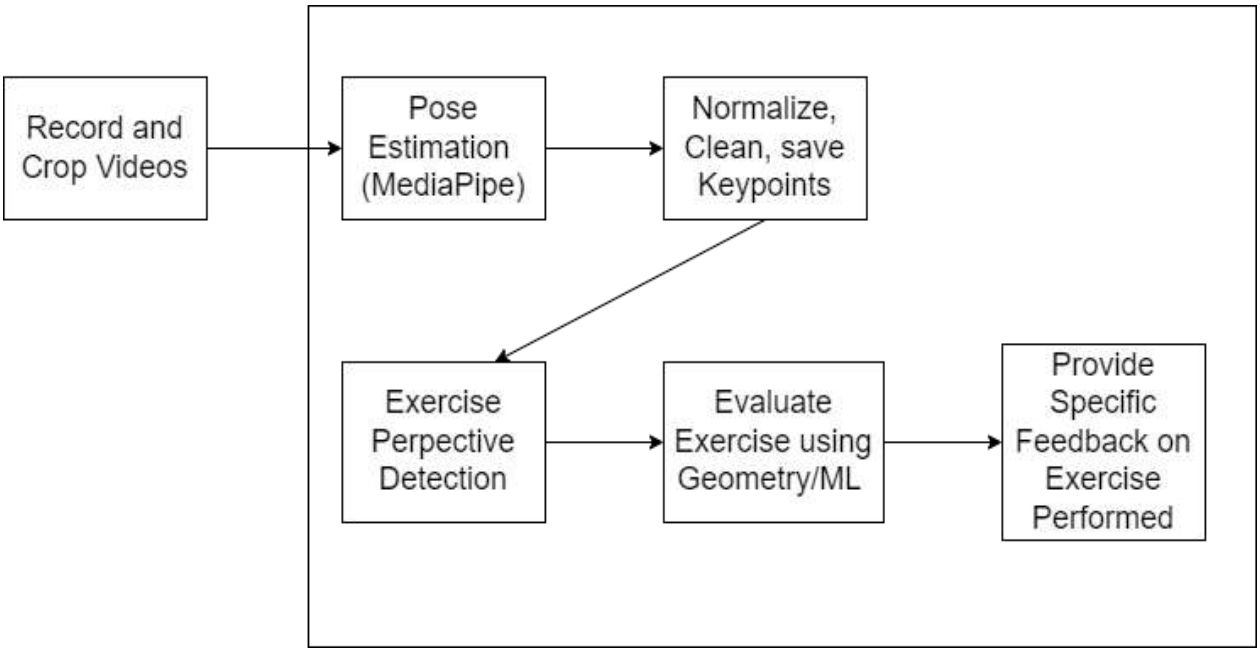


*Figure 4.1: System Design Workflow*

### 4.1.1. POSE ESTIMATION

For pose estimation, we chose to use the pre-trained model, MediaPipe (Blaze Pose) for pose detection. MediaPipe Pose is a ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation masks on the whole body from RGB video frames Blaze Pose research. Current state-of-the-art approaches rely primarily on powerful desktop environments for inference, whereas this method achieves real-time performance on most modern

mobile phones, desktops/laptops and even on the web.

MediaPipe is both accurate and efficient, while also scalable to multiple people without scaling up the run-time. Our approach provides human pose tracking by employing machine learning (ML) to infer 33, 2D landmarks of a body from a single frame. In contrast to current pose models based on the standard COCO topology, BlazePose accurately localizes more key points, making it uniquely suited for fitness applications. In addition, current state-of-the-art approaches rely primarily on powerful desktop environments for inference, whereas our method achieves real-time performance on mobile phones with CPU inference. If one leverages GPU inference, BlazePose achieves super-real-time performance, enabling it to run subsequent ML models, like face or handtracking.

### 4.1.2. KEY POINTS NORMALIZATION

Based on either the detection stage or the previous frame key points, mediapipe align the person so that the point between the hips is located at the center of the square image passed as the neural network input. It estimates rotation as the line L between mid-hip and mid-shoulder points and rotate the image so L is parallel to the y-axis. The scale is estimated so that all the body points fit in a square bounding box circumscribed around the body, as shown in Figure 3.1.4. On top of that, we apply 10% scale and shift augmentations to ensure the tracker handles body movements between the frames and distorted alignment. To support the prediction of invisible points, we simulate occlusions (random rectangles filled with various colors) during training and introduce a per-point visibility classifier that whether a particular point is occluded and if the position prediction is deemed inaccurate. This allows tracking a person constantly even for cases of significant occlusions, like upper body only or when the majority of person body is out of scene as shown on Figure 4.1.2.

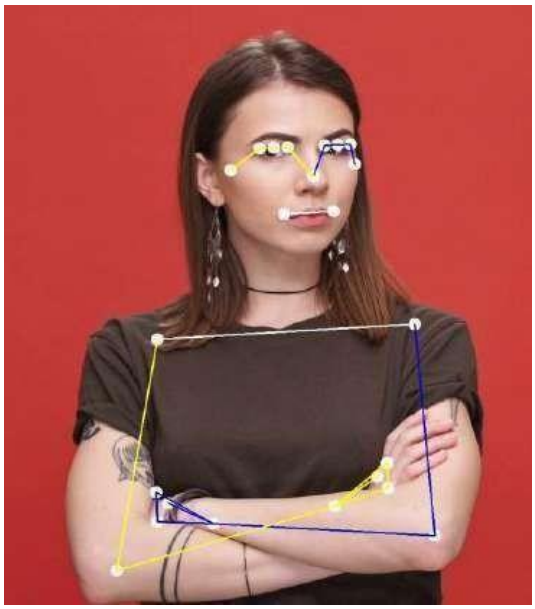

*Figure 4.1.2: Blaze Pose results on upper-body case [12]*

### 4.1.3. PERSPECTIVE DETECTION

For certain exercises, we resolve the ambiguity in camera perspective. For example, the bicep curl exercise is recorded from the side of the body, and could be performed with either the left orright arm. We identify which arm is performing the exercise in the video by measuring which keypoints are most visible (left or right-side key points) throughout all frames of the exercise. This accurately detects the perspective of all of our input videos.

### 4.1.4. GEOMETRY EVALUATION

Next, we compute body vectors from key points of interest, and use personal training guidelines and our own recorded videos to design geometric heuristics, evaluating on the body vectors.
The upper arm and forearm. In the start position with the weight down, the angle should be nearly $180^0$. As the weight is lifted, the angle should decrease.

Example: Bicep Curl.

For instance, in a bicep curl, we identify two heuristics of interest. First, the upper arm should be kept steady and not move significantly. We quantify this by the angle between the upper arm vector and the torso vector. If the upper arm is held steady, then it should be parallel to the torso with minor variations for the entire video. Significant movement of the arm will result in a large change in the angle between the two vectors.

Second, a proper, complete curl requires the weight to be brought up until the bicep is fully contracted, beyond the midway points ($90^0$ between upper arm and forearm) that is commonly stopped at. The improper from is typically a result of the user using weights that are too heavy, which our application will point out. We quantify this problem by the minimum angle achieved between until when the user stops, and increase again as the weight is brought down. Thus, finding the minimum angle across the video sequence will how far the user brought up the weight.

Thought our analysis of annotated video data, we find that if the angle between the shoulder and hip exceed $190^0$ or is less than $170^0$ then the user is bending their upper body. If the minimum angle between the upper arm vector forearm vectors is not below $80^0$, then the user is not curling the weight all the way up. Using the quantified measures and thresholds, we alert the user specifically which thresholds they exceed, and offer suggestions to improve their form.

## 4.2. SOFTWARE DEVELOPMENT MODEL

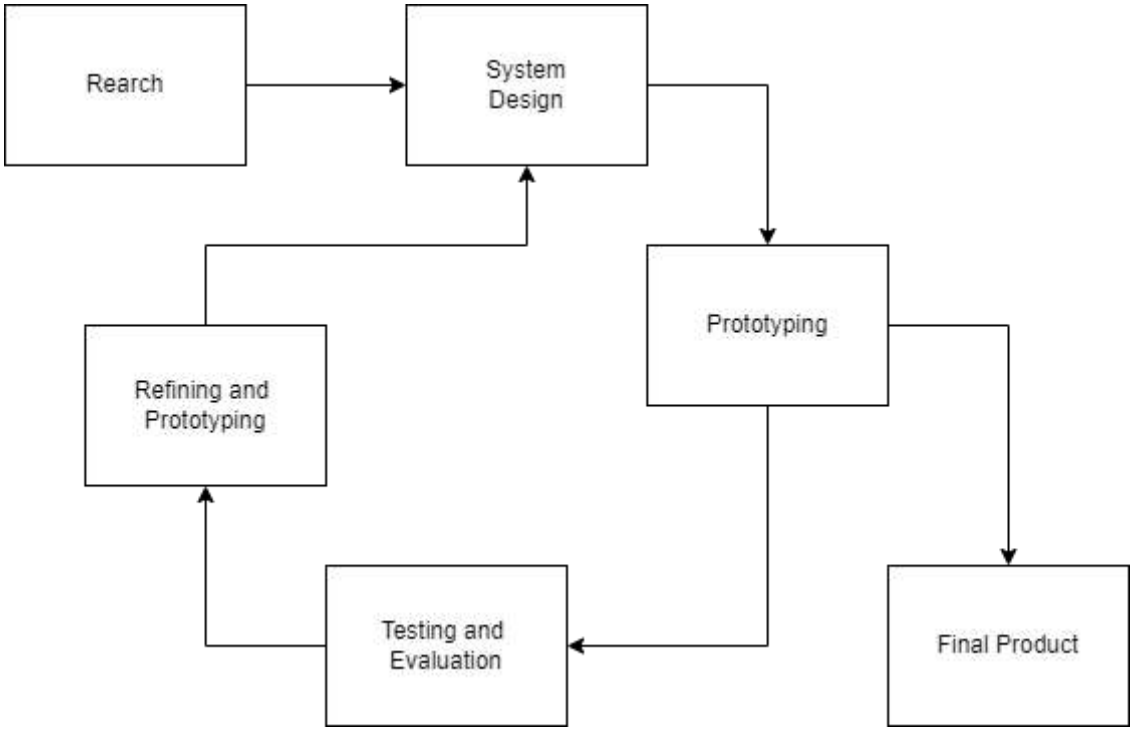


*Figure 4.2.1: Prototype Model*

Since we were not sure about our final design, we approached our project using the prototype life cycle model. The prototype life cycle model suggests that before developing the actual software, a working prototype of the system should be built. During the research phase we discovered multiple frameworks to implement our system. Many of these frameworks required a high-end GPU in order to get acceptable frames. However, we were able to find a framework named Mediapipe that used very few resources. With the help of this framework, we were able to run oursystem on computers without GPU. In the system design phase, we made a quick design for ourfirst and only prototype. Then we made a prototype that could count the repetitions of bicep curls.After building the first prototype we tested it and found out it needed some minor tweaking in thegeometric analysis part. In the refining and prototyping phase, we fixed the geometric analysis

part and the repetitions were being counted properly**.**

Then we made a prototype that could count the repetitions of bicep curls. After building the first prototype we tested it and found out it needed some mirror tweaking in the geometric analysis part. In the refining and prototyping phase, we fixed the geometric analysis part and repetitions were being counted properly. Then we were able to make a final design of the system with the helpof the prototype. And finally, the system was built that could detect wrong and right posture doneduring the exercise sessions for four different exercises.

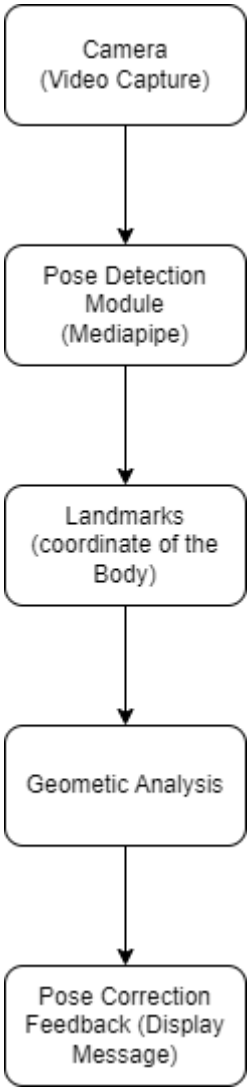


*Figure 4.2.2: Prototype Process Personal Trainer*

## 4.3. CLASS DIAGRAM

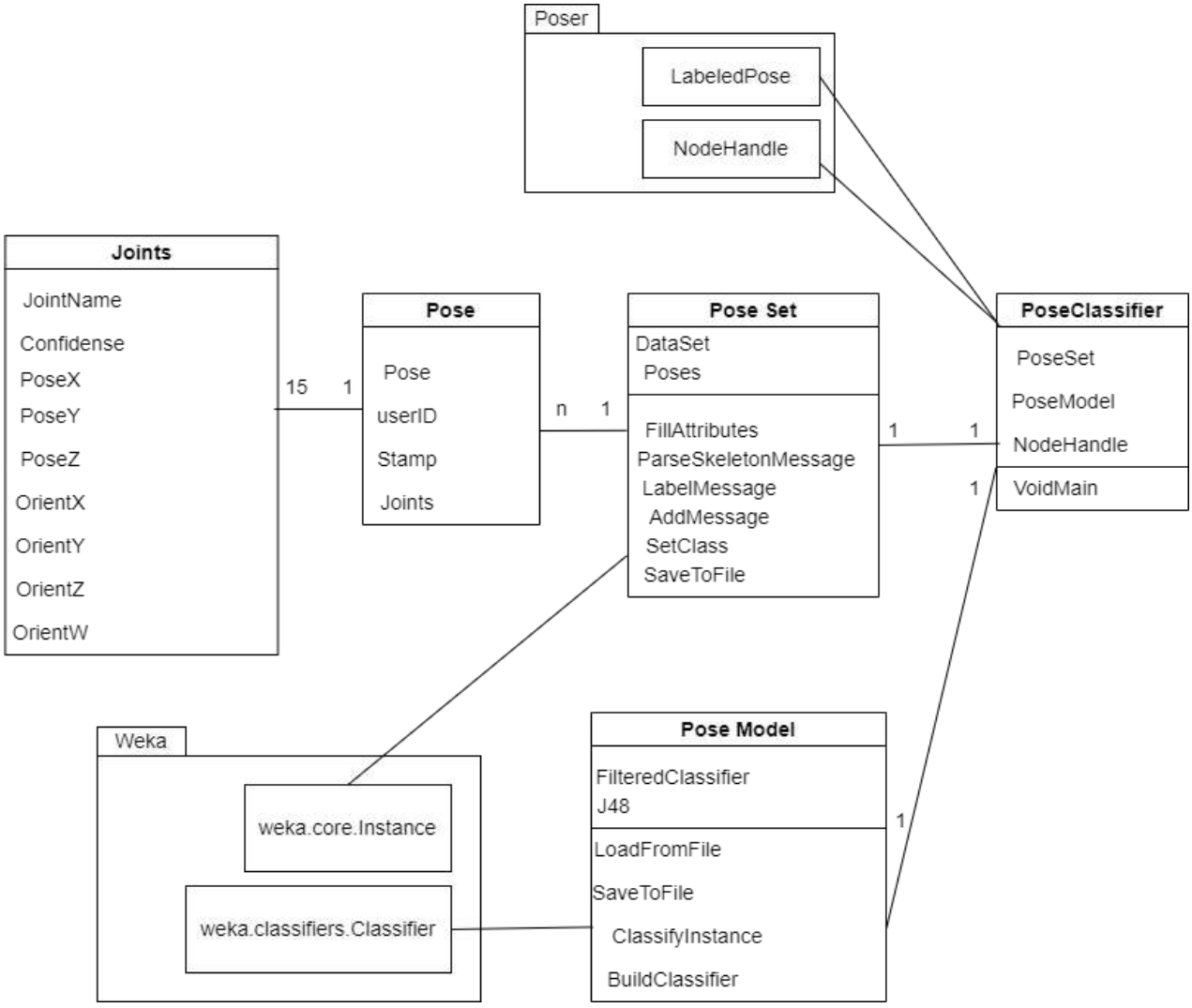


*Figure 4.3: Class Diagram for Human Pose*

### 4.3.1. USE CASE DIAGRAM

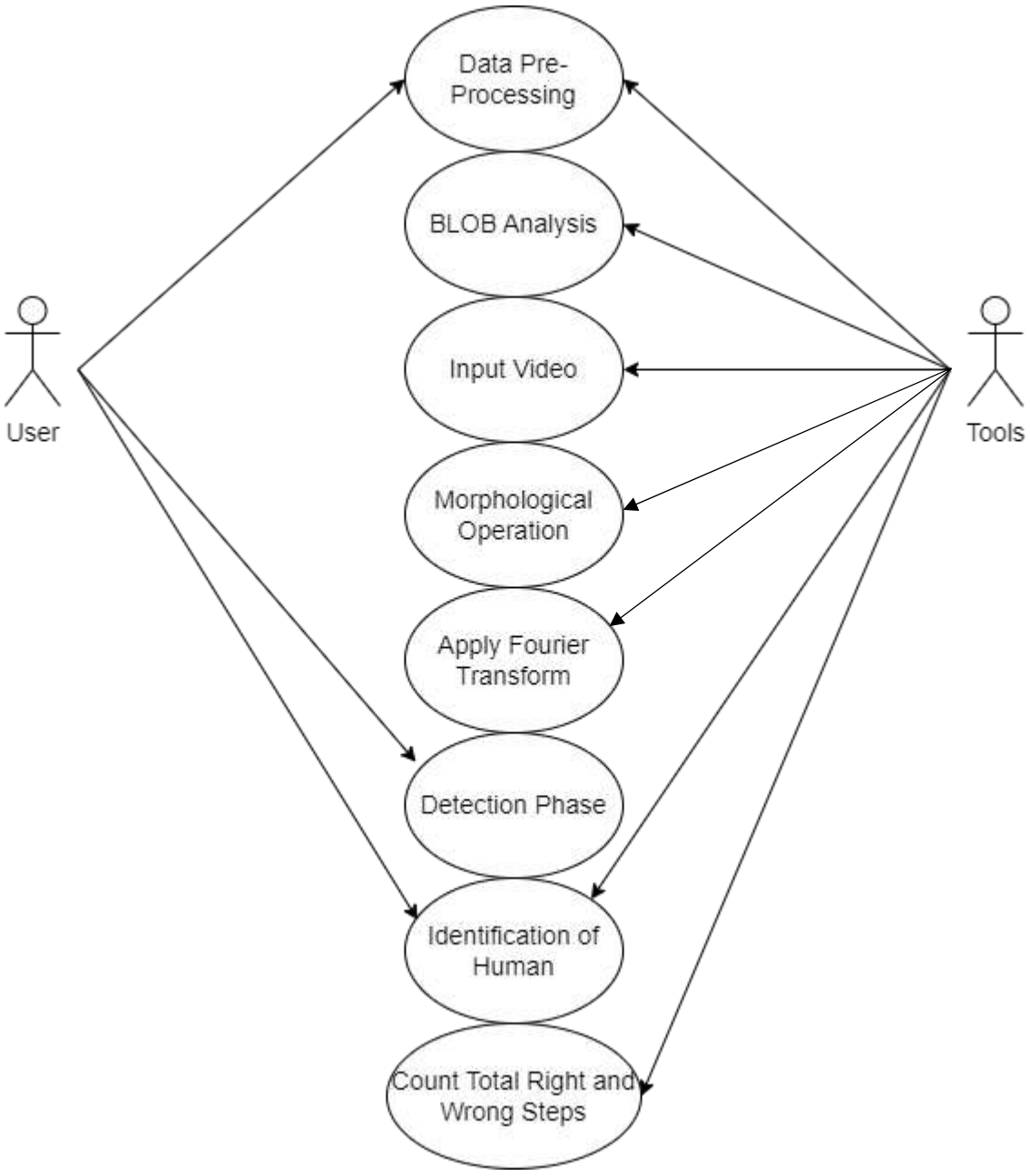


*Figure 4.3.1: Use Case Diagram*

## 4.4. SYSTEM DESIGN

### 4.4.1. BLOCK DIAGRAM

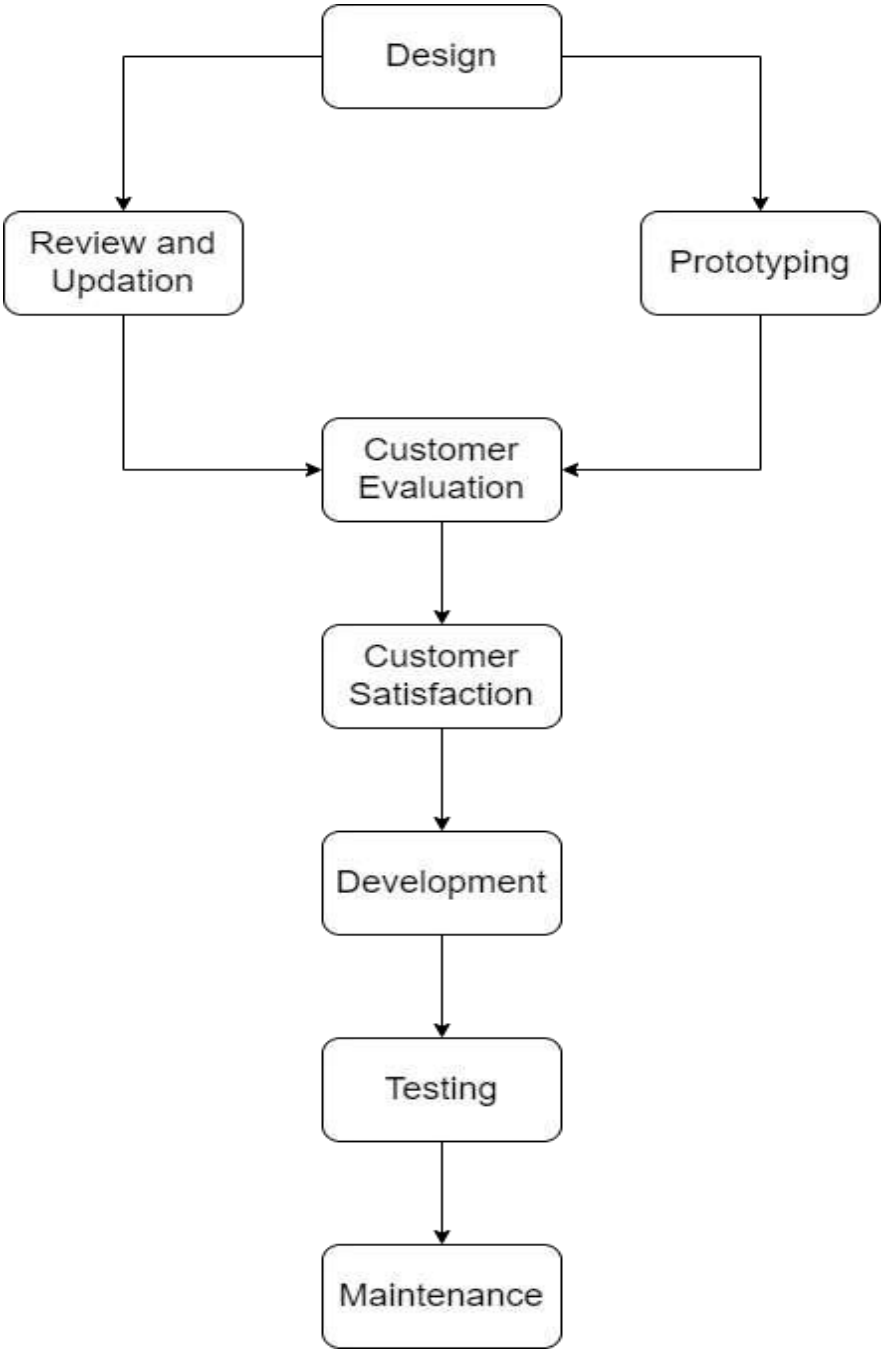


*Figure 4.4.1: Block Diagram of Personal Trainer*

### 4.4.2. FLOWCHART

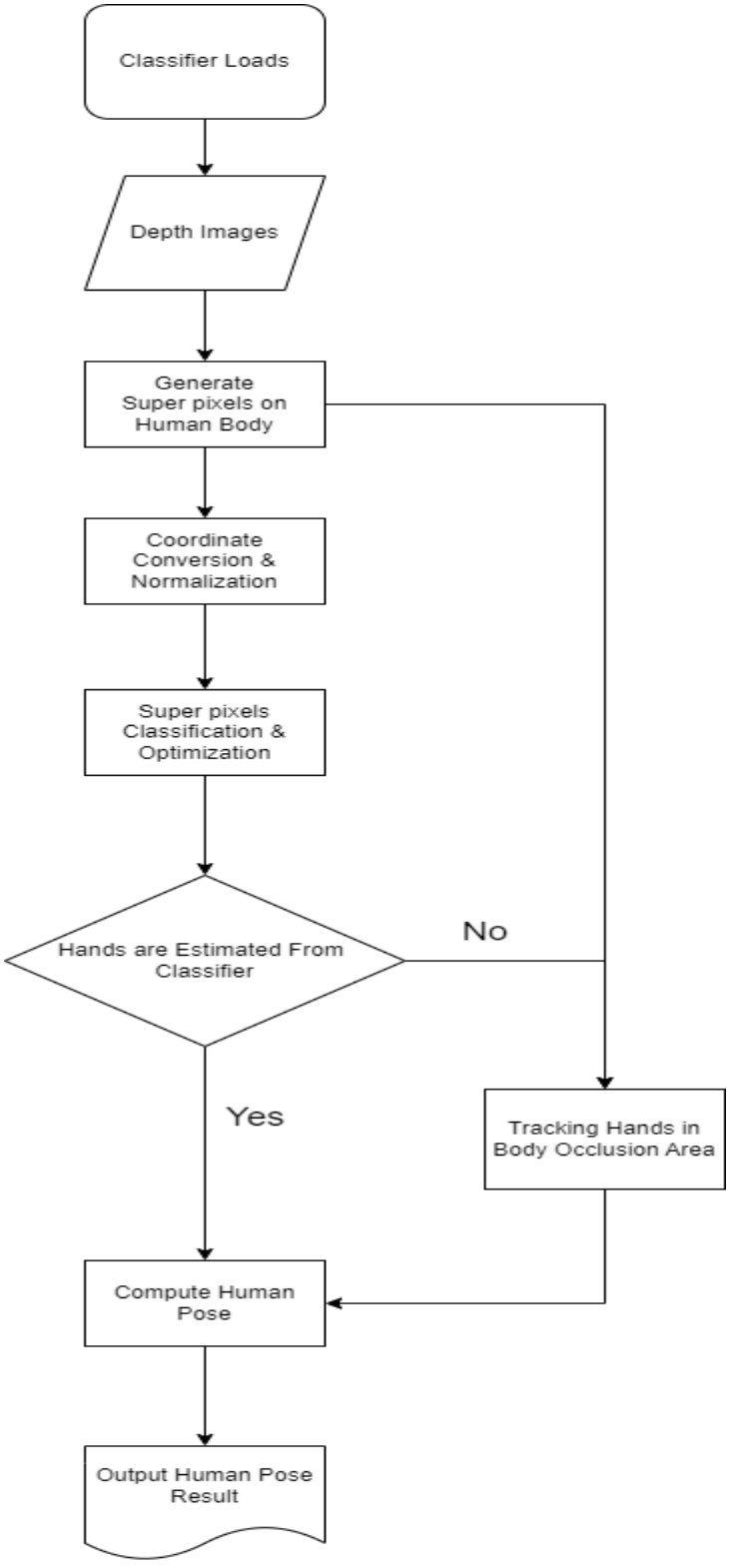


*Figure 4.4.2: Flowchart of Pose Visualization*

### 4.4.3. DATA FLOW DIAGRAM

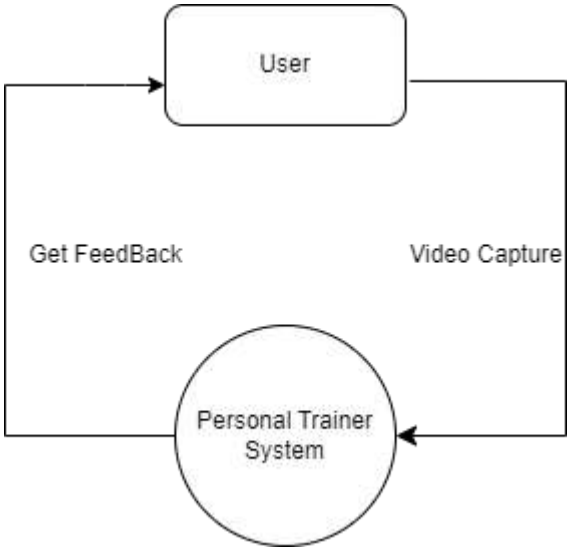


*Figure 4.4.3.1. Level 0 Data Flow Diagram*

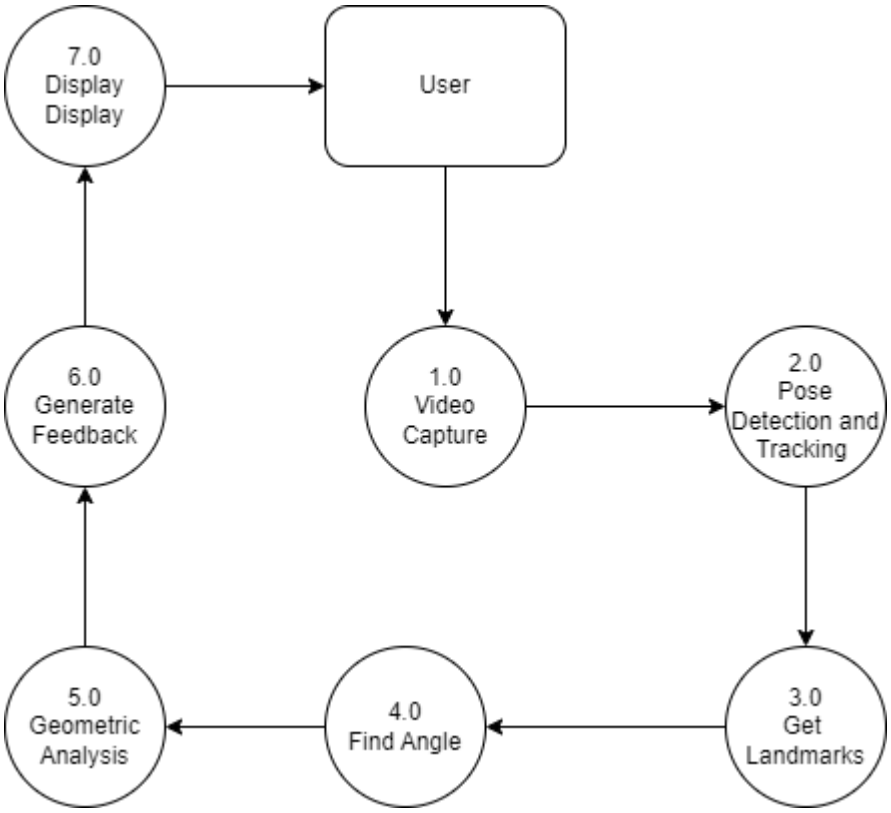


*Figure 4.4.3.2. Level 1 Data Flow Diagram*

## 4.5. UML DIAGRAM

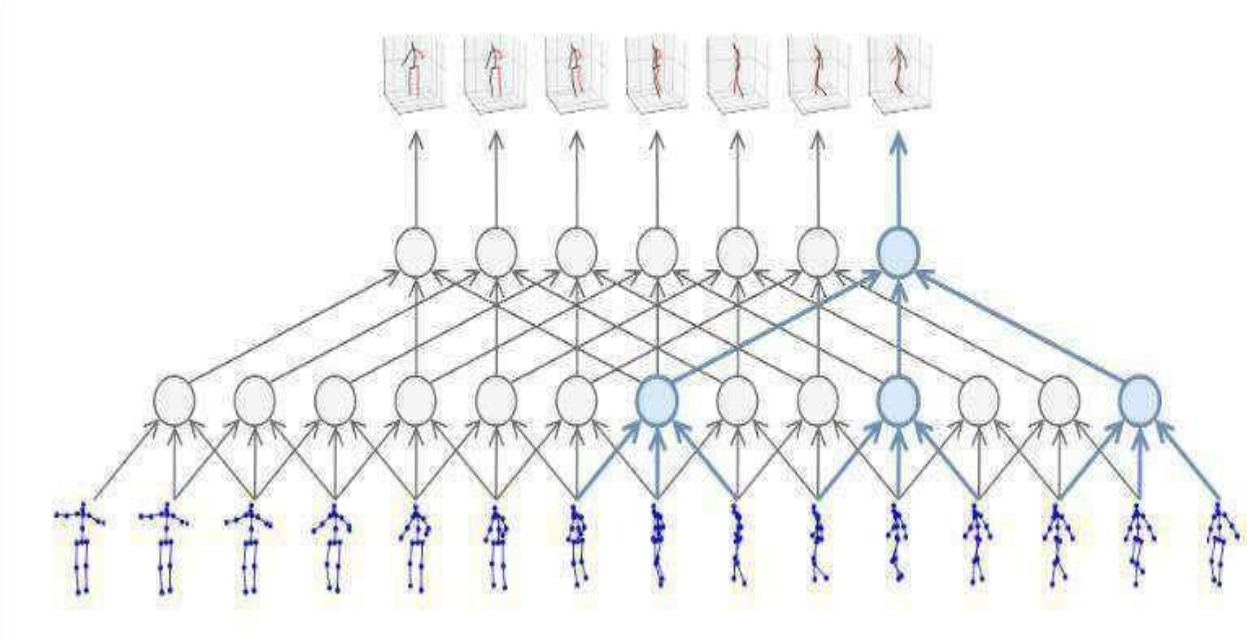

*Figure 4.5.1. UML Diagram of Pose (1)*

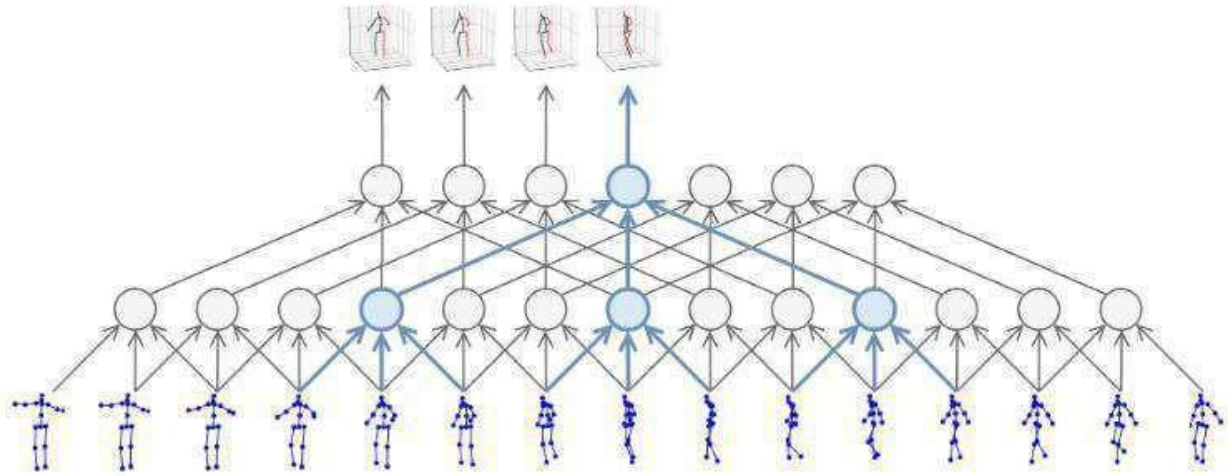

*Figure 4.5.2. UML Diagram of Pose (2)*

# CHAPTER 5: RESULTS AND DISCUSSION

We present results of Pose Trainer on four different exercises: bicep curl, pushup, squat and sit-up. For each exercise, we took a geometric heuristic approach to count repetitions and identify correct posture.

## 5.1. BICEP CURL

When the hip angle is near about 180 degrees i.e., when a person is standing straight while doing bicep curls then the posture is correct and we increment the right counter. Otherwise, the posture becomes incorrect and we increment the wrong counter.

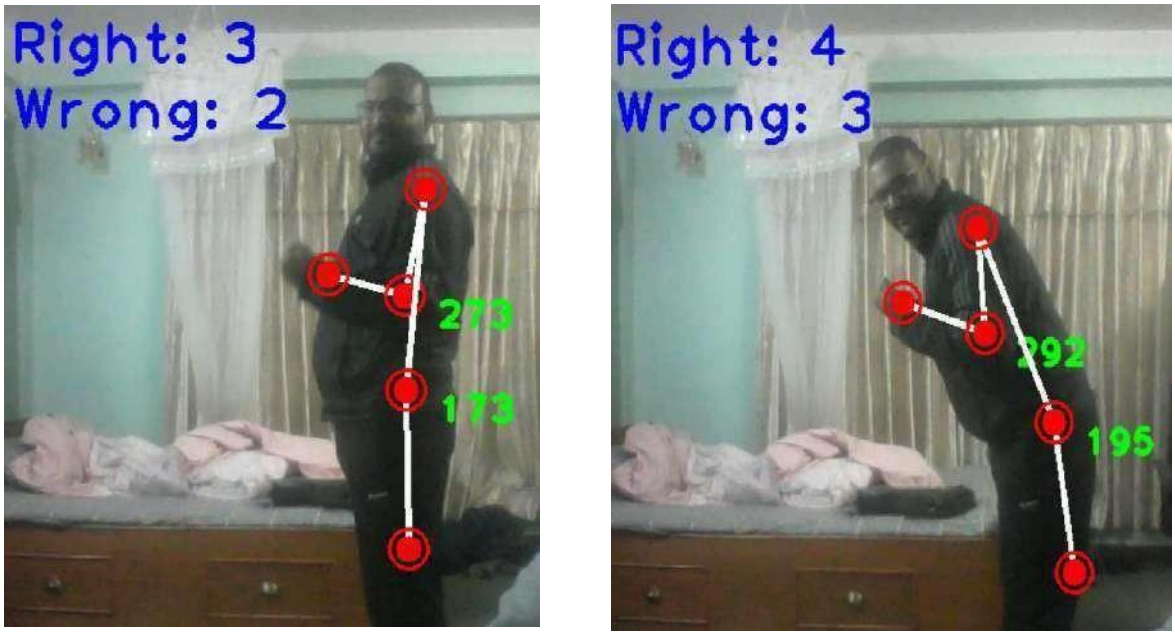


*Figure 5.1: Bicep Curl right and wrong posture*

## 5.2. PUSH-UP

When the right hip angle is near about 170 degrees or left hip angle is near about 190 when a person is performing pushups then the posture is correct and we increment the right counter. Otherwise, the posture becomes incorrect and we increment the wrong counter.

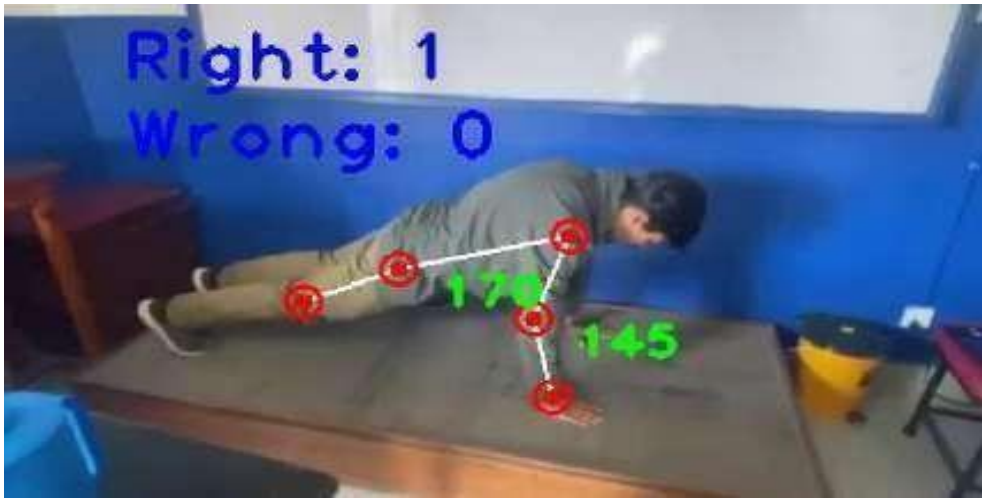


*Figure 5.2.1: Push up right and wrong posture (1)*


*Figure 5.2.2: Push up right and wrong posture (2)*

## 5.3. SQUAT

When the right knee angle goes from around 180 degrees to around 250 degrees or the leftknee angle goes from around 180 degrees to around 110 degrees while performing squats,we increment the counter.


*Figure 5.3: Squat right posture*

## 5.4. SIT-UP

When the right hip angle goes from around 120 degrees to around 80 degrees or the left hip angle goes from around 240 degrees to around 280 degrees while performing sit ups,we increment the counter.

## 5.5. CONCLUSION

In conclusion, Mediapipe was very effective for this project. It used less resources by performing detection and tracking rather than only detection, which made it able to be implemented in a low-spec computer. The final system consists of four different exercises for each of which training videos were recorded and geometric heuristic algorithms were used to automatically determine posture correctness.

All in all, the system helps us to exercise in a good posture. The input fed into the system is a video feed and the output is the number of wrong and right postures done during the exercise session.

# CHAPTER 6: LIMITATIONS AND FUTURE ENHANCEMENTS

## 6.1. LIMITATIONS

The limitations that exist in the project are as follows:

- Limited exercise for evaluation
- Cannot recognize the activity being done.
- There is lack of Security.

## 6.2. FUTURE ENHANCEMENTS

In future, the project can be enhanced to:

- A fully functional Application.
- Provide more feedback for corresponding exercises
- Add more exercises for evaluation.
- Make the application recognize the activity being performed.
- Make the application more secure.

# SCREENSHOTS

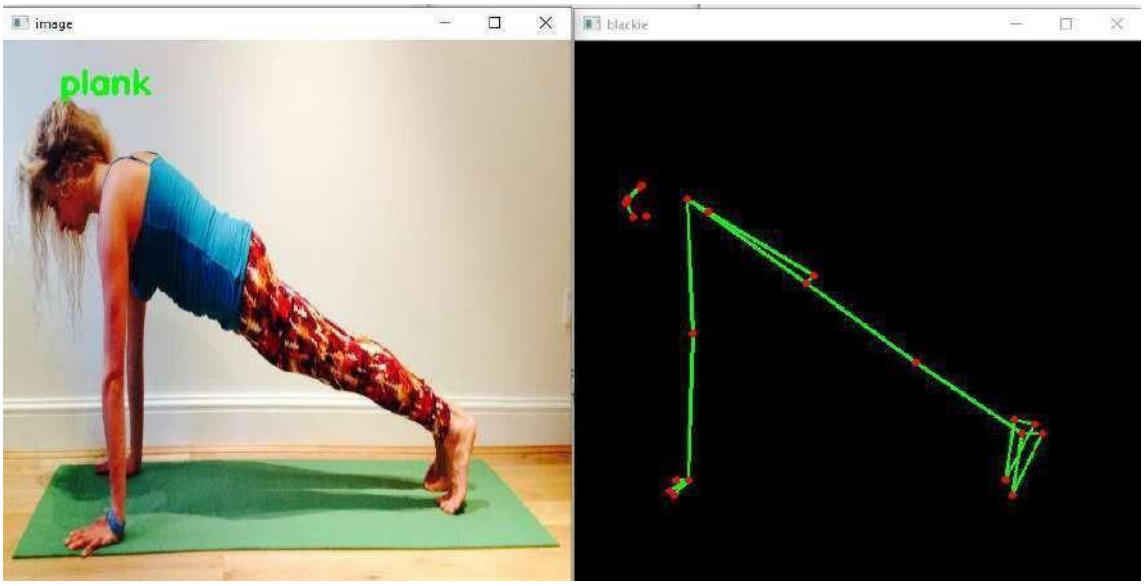


*Figure G1: Result of Models (1)*

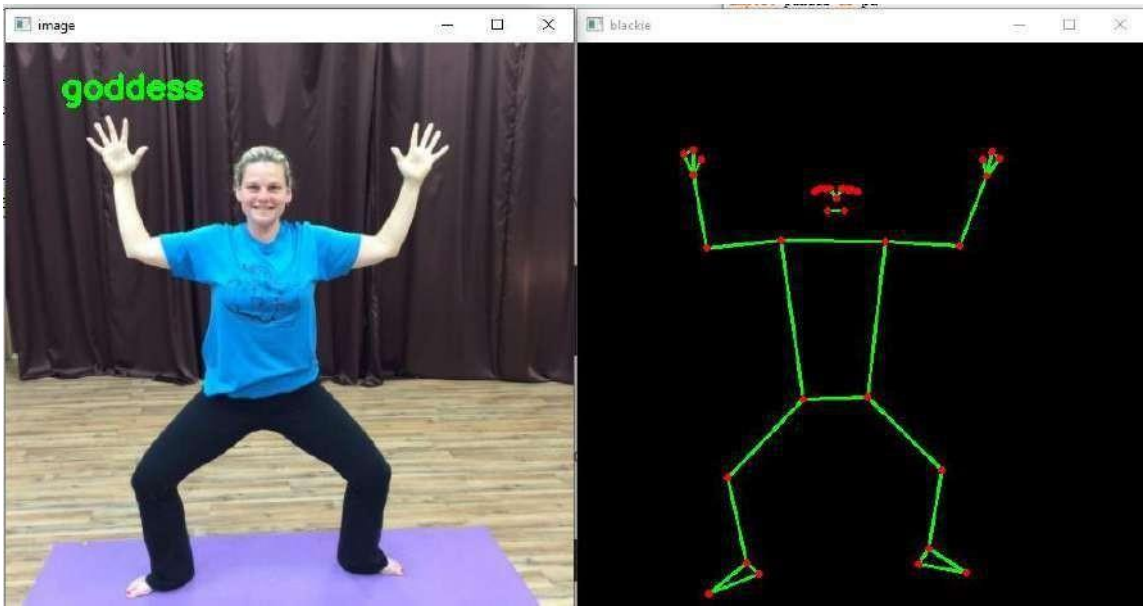


*Figure G2: Result of Models (2)*

# REFERENCES


[1] M. Taraiants, “Challenges of Human Pose Estimation in AI-Powered Fitness Apps”, InfoQ, 15 Oct 2020. {Online} Available: https://www.infoq.com/articles/human-pose-estimation-ai-powered-fitness-apps/. {Accessed 1 June 2022}.

[2] “Home.” Mediapipe, google Available: https://www.google.github.io/mediapipe/. {Accessed 23 April 2022}.

[3] A. Toshev and C. Szegedy, “DeepPose: Human Pose Estimation via Deep Neural Networks”, arXiv {cs. CV}, 2013. {Accessed June 2022}.

[4] A. Newell, K. Yang and Deng, “Stacked hourglass networks for Human pose estimation”, in Computer Vision-ECCV 2016, Cham: Springer International Publishing 2016, pp. 483-499. {Accessed April 2022}.

[5] S.-E Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional Pose Machines”, arXiv {cs. CV}, 2016. {Accessed June 2022}.

[6] X. Incubation,” Real-Time Human Pose Recognition in Parts from Single Depth Images”, Microsoft.com, {Online} Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartR. {Accessed May 2022}.

[7] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, “Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image,” *arXiv [cs.CV]*, 2016. {Accessed June 2022}.

[8] P. Zell, B. Wandt, and B. Rosenhahn, “Joint 3D human motion capture and physical analysis from monocular videos,” in 2017 IEEE Conference on Computer Vision and Pattern Workshops {CVPRW}, 2017.

{Accessed June 2022}.

[9] Bandi, Chaitanya. “Regression-Based 3D Hand Pose Estimation Using Heatmaps”. Sci Te Press, 2 March, 2022. Available: www.scitepress.org/Link.aspx?doi=10.5220/000897320636 0643.

{Accessed May 2022}

[10] Chen, S. and Yang, R., 2022, “Pose Trainer: Correcting Exercise Posture using Pose Estimation”. {Online} DeepAI Available: https://deepai.org/publication/pose-trainer-correcting-exercise-posture-using-pose-estimation.

[11] “On-device, Real-time Body Pose Tracking with Mediapipe BlazePose”, Google AI Blog. Available:

https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html.

[12] V. Bazarevsky, I. grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grungmann, “BlazePose: On device Real time Body Pose tracking”, arXiv: 2006.10204 [cs], June, 2020. {Online}. Available: https://arxiv.org/abs/2006.10204.

[13] A Bulat and G. Tzimiropoulos, “Human Pose Estimation via Convolution Part Heatmap Regression”, arXiv: 1609.01743 [cs], volume 9911, pp. 717-732, 2016, and doi: 10.1007/978-3-319-46478-7_44.

[14] A. Newell, K. Yang, and J. Deng, “Stacked Hourglass Networks for Human Pose Estimation”, arXiv: 1603.06937 [cs], July, 2016, {Online}. Available: https://arxiv.org/abs/1603.06937. {Accessed April 19, 2022}.

[15] "Regression Algorithms | 5 Regression Algorithms you should know", Analytics Vidhya May 26, 2021.

Available: https://www.analyticsvidhya.com/blog/2021/05/5-regression-algorthms-you-should-know-introductory-guide/#~:text=Linear%20Regression%20is%20an%20ML. {Accessed September 18, 2022}.

[16] A. Tavanaei, "Embedded Encoder-Decoder in Convolution Networks towards Explainable AI", arXiv: 2007.06712 [cs], June, 2020. {Online}.

Available: https://arxiv.org/abs/2007.06712. {Accessed April 18, 2022}.

[17] Oleic, "Deep learning for Humans", Oleic 01. November 21, 2018.

Available: https://medium.com/oleicverse/deep-learning-for-humans-89b3b2de0bc8 {Accessed August 2022}

# APPENDICES


*Figure A1: Bicep Curl Evaluation*


*Figure A2: Squat Evaluation*


*Figure A3: Push-up Evaluation*

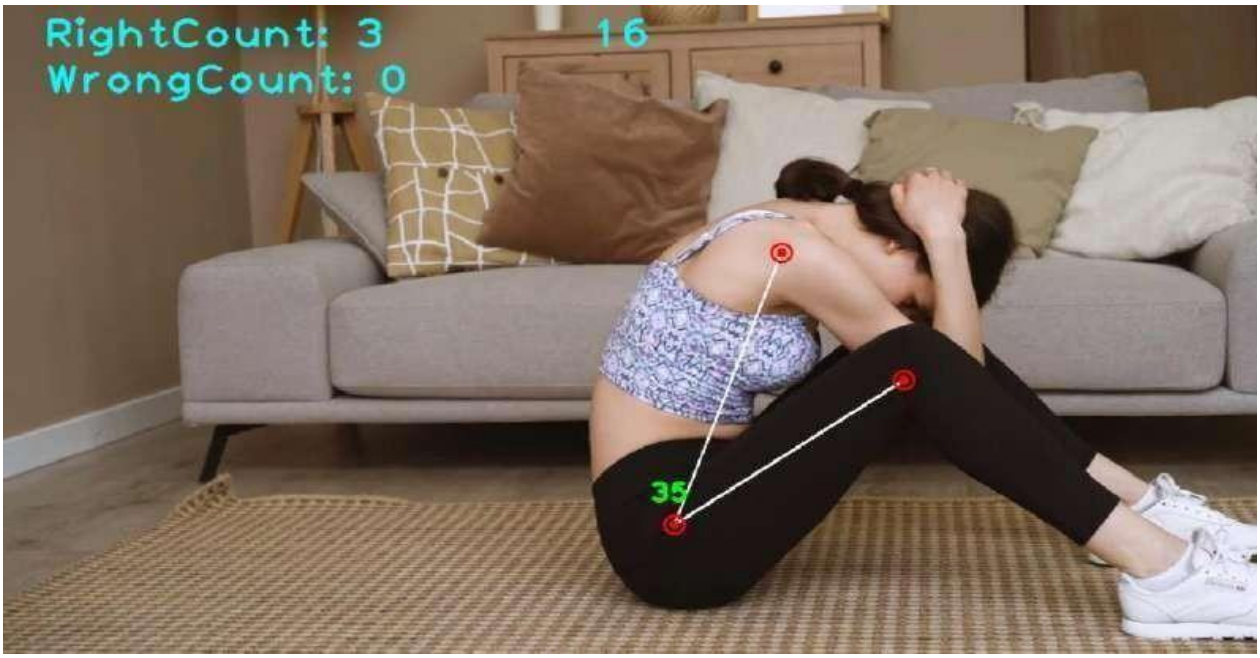


*Figure A4: Sit-up Evaluation*

```
if len(lmList) != 0:

    if lmList[14][3] < lmList[13][3]:
        right_elbow_angle = detector.findAngle(img, 12, 14, 16, 25, 25)
        right_elbow_percentage = np.interp(right_elbow_angle, (150,50), (0,100))
        right_hip_angle = detector.findAngle(img, 12, 24, 26, 25, 25)
        if right_elbow_percentage == 100:
            if right_hip_angle >= 175 and right_hip_angle <= 185:
                if dir_right_hand == 0:
                    right_count += 0.5
                    dir_right_hand = 1
            else:
                if dir_right_hand == 0:
                    wrong_count += 0.5
                    dir_right_hand = 1

        if right_elbow_percentage == 0 :
            if right_hip_angle >= 175 and right_hip_angle <= 185:
                if dir_right_hand == 1:
                    right_count += 0.5
                    dir_right_hand = 0
            else:
                if dir_right_hand == 1:
                    wrong_count += 0.5
                    dir_right_hand = 0
```

*Figure A5: Code for Bicep Curl Right Hand Perspective*

```
else:
    left_elbow_angle = detector.findAngle(img, 11, 13, 15, 25, 25)
    left_elbow_percentage = np.interp(left_elbow_angle, (210, 310), (0, 100))
    left_hip_angle = detector.findAngle(img, 11, 23, 25, 25, 25)
    if left_elbow_percentage == 100:
        if left_hip_angle >= 175 and left_hip_angle <= 185:
            if dir_left_hand == 1:
                right_count += 0.5
                dir_left_hand = 0
        else:
            if dir_left_hand == 1:
                wrong_count += 0.5
                dir_left_hand = 0

    if left_elbow_percentage == 0:
        if left_hip_angle >= 175 and left_hip_angle <= 185:
            if dir_left_hand == 0:
                right_count += 0.5
                dir_left_hand = 1
        else:
            if dir_left_hand == 0:
                wrong_count += 0.5
                dir_left_hand = 1
```

*Figure A6: Code for Bicep Curl Left Hand Perspective*