ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Robot for Gym Exercise Guidance

Team #34

DALEI JIANG (daleij2@illinois.edu)
ZIFEI HAN (zifeih2@illinois.edu)
KUNLE LI (kunleli2@illinois.edu)
CHANG LIU(changl12@illinois.edu)

TA: Yiqun Niu
Sponsor: Prof. Gaoang Wang

May 21, 2023

Abstract

In the modern rapid life pace, doing exercise to burn off excess calories from daily intake has become a normal need for many people who wish to keep fitness. When doing sports, many people will choose to receive guidance in multiple ways to make sure they are doing correct movements. Among all those choices, hiring a gym coach is a quite common method.

However, most of the people who wants to get guidance cannot afford the cost of a gym coach. Besides, it can also be a hassle to schedule a time and place to meet the coach. Our team believe that one gym exercise guidance robot can offer more cheap and convenient service. It can be used in the gymnasiums or at home. You can use the robot to get feedback for your exercise at any time you want.

Our robot is equipped with human tracking function based on ROS, and the user can activate this function so that the robot will follow behind the user until the user reaches the location suitable for doing sports and turned off the tracking function.

We have designed user-friendly GUI, and the user can select the type of the exercise he or she wants to do. When the user is doing sports in front of the camera, the robot will do the counting for the periodic exercise type, such as squats and push-ups, and use yellow bar to point out any incorrect movements.

After completing doing exercise, the user can generate a report to show the exercise record. It will show you the report of your latest exercise. It will give you suggestions such as ”Hip angle is too large”, and show your postures not meet the standard.

Keywords: Human tracking, Pose evaluation, Graphical user interface

Contents

1 Introduction

1.1 Project Purpose

Doing exercise to keep fitness is a common need for most of people. However, non-standard movements cannot help the athletes to achieve the desired effects. Furthermore, improper movements in exercising may cause harm and damage to human bodies. According to a study published in the Journal of Strength and Conditioning Research, "improper weightlifting technique is one of the most common causes of injury in weightlifting". [7]

Obviously, one solution is to hire a professional gymnasium coach to offer the guidance. However, hiring an instructor will be too expansive and inconvenient for most of the exercisers. Asking for coach's help cost much money and need to make schedule every time. Besides, if you cannot afford a personal coach, you cannot get personal guidance based on the process of your recent exercise and improvements since the coach needs to work for many exercisers.

1.2 High Level Requirements

1. The robot should be able to track and follow the user's location within a certain distance (0.5m - 0.9m), using depth camera to get the environment information. The robot was able to keep its tracking object in the center of its field of vision.
2. The robot should be able to recognize body key points and skeleton binding, and can do the counting of the exercise for at least two types (squarts and push-ups).
3. The robot can check whether the person is doing correctly in the exercise based on the body angles and the exercise standard. The robot should be able to give comments, and store those evaluation results in the folder, and show those result in a report.
4. The user can use the robot to do the exercise evaluation function purely through the GUI platform without using keyboard to enter commands.

1.3 Block Diagram

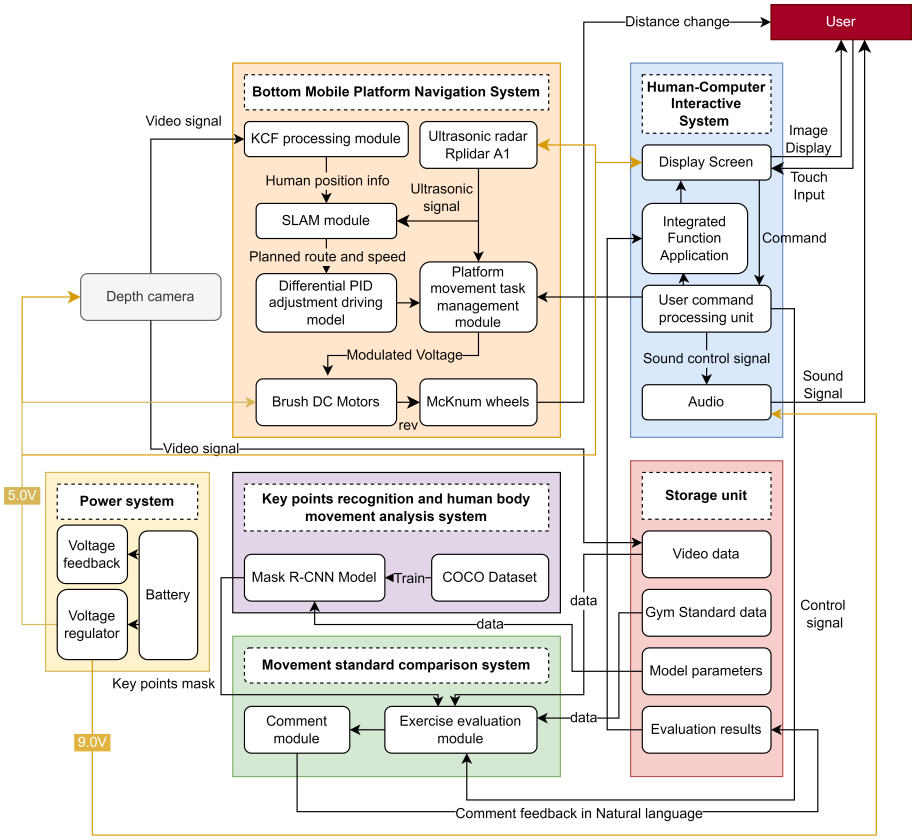


Figure 1: Block Diagram

2 Hardware setting

2.1 Hardware Structure

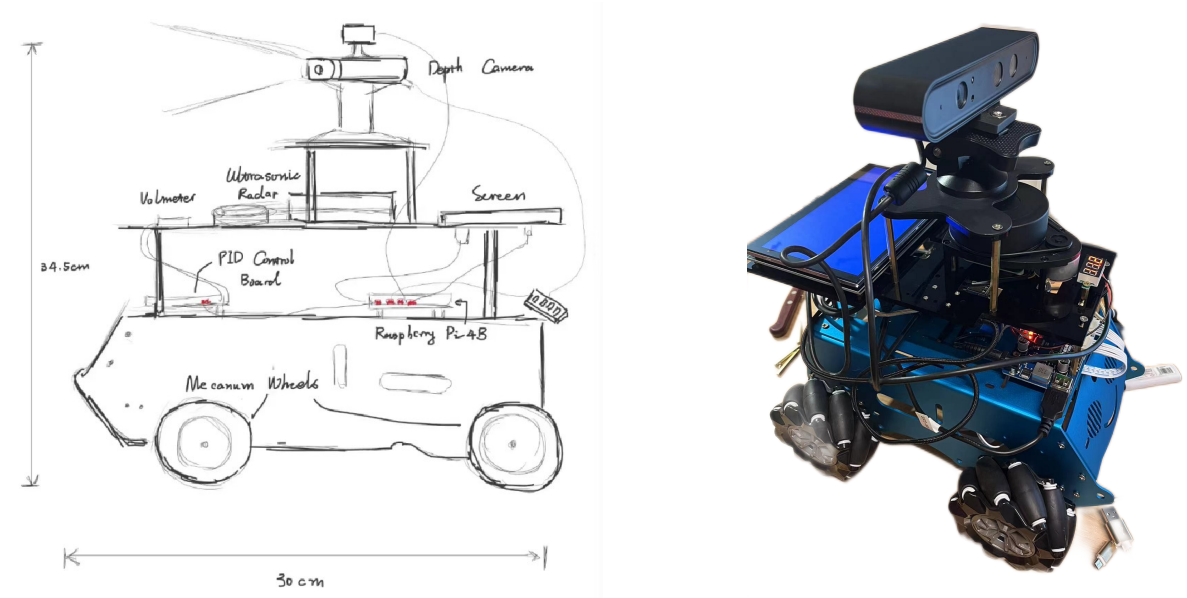
Our robot structure is shown below, and the hand-drawing draft shows all the components of the hardware system. The robot is equipped by two cameras, one is an HD camera which is use for movement recognition, and another one is an ROS depth camera used in human tracking.

The structure of the hardware has underwent many changes. Fig. 2 above shows our final version.

2.2 Hardware Components

2.2.1 Cameras

On the top of the robot, there are two cameras. The smaller one is a normal camera which returns 640*480 video signal. The bigger camera is a ”Astra Pro” ROS depth



(a) Hardware Structure (b) Photo

Figure 2: Overview of Robot

camera, which is able to return the depth image. The reason of using two cameras will be elaborated in the next section.

2.2.2 Raspberry Pi 4B

Two cameras will transport the signal to the main board. For the main board, we have chosen Raspberry Pi 4B based on the budget of our team. A board with high computational efficiency is important for our project, and Raspberry Pi 4B is the best choice we can make.

2.2.3 STM32F103RCT6 control board

After we get the location information of the human, the information need to be published and sent to the motors. We have used a PID control board (Model number STM32F103RCT6) which can control the angular velocity of the motors to control the wheels. At the same time, the PID control can do the work of stabilize the voltage on 5 volts, which is necessary for other components.

2.2.4 Motors, wheels and battery

Motors are selected to be DC brushed motors with 360 wire AB code. For the coding convenience, the wheels are selected to be Mecanum wheels. Battery is an 8400mA rechargeable lithium battery.

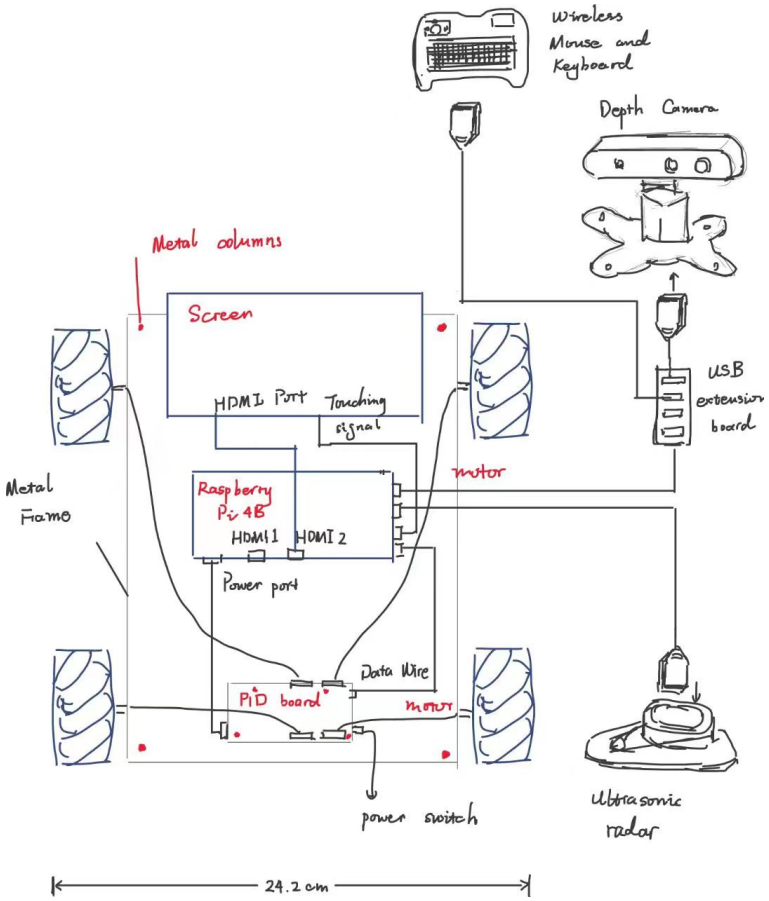


Figure 3: Schematics with components

2.2.5 Display screen

As we need to build GUI system, display screen is chosen to have touching signal transporting ability. The size of the display is 7 inches. Two wires are connected to the back of the display screen, and one is micro HDMI wire, and another one is the touching signal wire.

2.2.6 Wireless mouse and keyboard

For the convenience in building the system and further maintenance, we have add Bluetooth mouse and keyboard to the robot. A Bluetooth receiver is attached on the USB port.

2.2.7 Schematics with components

In Fig. 3, we have shown the schematics of the system.

3 Human tracking system

3.1 System introduction and requirements

When the user need to use the robot, we require the robot to track behind the user and follow the people until the user reach the location he/she wants. Thus, we need to design a human tracking function.

Our requirements of human tracking function includes:

1. The robot can follow the user who is walking with normal speed in straight line, and keep the distance with certain range.
2. The robot can keep the user in the center of the scenery. (From 1/4 to 3/4 of the screen horizontally).
3. The robot can avoid being influenced by the surrounding items it passes by, and keep following the desired target person.
4. The function can be launched and shut down easily.

3.2 Challenges

Firstly, our main board does not have quite high processing speed, which means that too complex algorithm cannot be adopted.

Secondly, our robot is only 34 cm tall, and it means that if the distance between robot and human is too small, only human's legs and feet will be included in robot's view, which will loss much features for recognizing human's position. However, if the tracking distance is large enough to include most of human body in eyes, more confounding items will also appears in robot's view.

Thirdly, the environment light condition is not determined, which means that we need the robot to be able to get rid of the dependence on color recognition, and also be able to work on different light condition.

3.3 Algorithm choices

After reading many papers and materials, we have chosen three kinds of algorithms and methods which is frequently used in object tracking.

3.3.1 Faster R-CNN

Faster R-CNN is a kind of classic neural network used in computer vision [9]. It can do the image segmentation and classification. Also, it can determine the region box

of the human-beings and get the location information. The region box generated by Faster R-CNN can help us to do the human tracking.

However, the disadvantage is that, it needs quite fast processing speed to catch up with the camera's rate. In Raspberry Pi, it costs 20 seconds to process one single image. This processing speed is not acceptable in tracking. This is the decisive flaw let us abandon this method.

3.3.2 Kernel Correlation Filter tracking algorithm

Kernel Correlation is also a quite important algorithm which is often used in object tracking. We choose this algorithm as the backup because this algorithm is in low-complexity and the requirements of hardware can be satisfied easily. When we are using this algorithm to track the object, we firstly need to determine the object we want to track, and then, the algorithm will extract the features from the box, and build the relation with filter models. Use the old features extracted to train the filters, and predict the possible location in new image, and add new features to the filter model [5].

This algorithm is abandoned because in this algorithm, we cannot get the distance information. The only information returned is the bounding area of the human. Without the distance, the robot is not able to know when to start and when to stop.

3.3.3 Central depth sampling method

Central depth sampling method is published on Github by Turtlebot, which is a great algorithm designed of ROS robot to do the tracking. This algorithm is quite simple but has great performance if the parameters are chosen properly.

The advantage of this algorithm is that, we can get both x, y, z-coordinate value of the predicted location. Then, we can adjust the robot's linear and angular velocity to realize the tracking. Finally, we choose this algorithm. The details will be introduced in the next section.

3.4 Principle of Central depth sampling method

3.4.1 Outline

Central depth sampling method is raised by ROS wiki officials as a suggested method to achieve the tracking on a ROS based robot [8]. The advantage of this method is that it only take use of the information transported from depth camera. Those data will be processed with basic mathematical calculation without the neuron network to get the

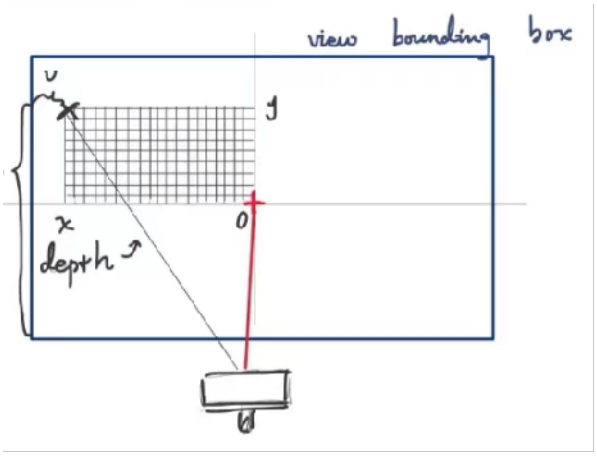


Figure 4: Trigonometric in scenery

target. Therefore, the processing speed is quite fast and we can set the rate higher to get better performance. The main board of our robot is Raspberry Pi 4B which does not have quite high processing speed. Central depth sampling method will be a quite good choice.

This algorithm has close relationship with the hardware parameters. It need to take use of the resolution and scenery angles of the camera. Before the image is imported, we need to get the trigonometric array of the robot view.

In Fig. 4, we show the scenery of the robot. Our depth image has a vertical field angle of $45.8°$, and a horizontal angle of $58.4°$. We need to generate a trigonometric array with the size $640 \times 480 \times 2$.

The scenery of the robot will be a tetragonal pyramid whose cross sections are rectangles. Any items in this tetragonal pyramid can be captured by the camera theoretically. When the depth data for one pixel is determined, and the pixel location is determined, then we can determine the rectangle pixel locates. Therefore, the x, y-coordinate values are in direct proportional to the depth of the location. Thus, we can build a 2-D metric containing the parameters for each proportion. In the robot, the image resolution is 640×480. For x and y coordinate, we need two set of the ratios. Thus, we need to build a $640 \times 480 \times 2$ matrix.

3.4.2 Ratio matrix

When building the ratio matrix, central depth sampling method makes use of parameters: X pixel amount, Y pixel amount, horizontal field angle, vertical field angle. With these information, we can get the relative location of the pixel in the image, as shown in Fig. 5.

Therefore, we can build the Ratio matrix:

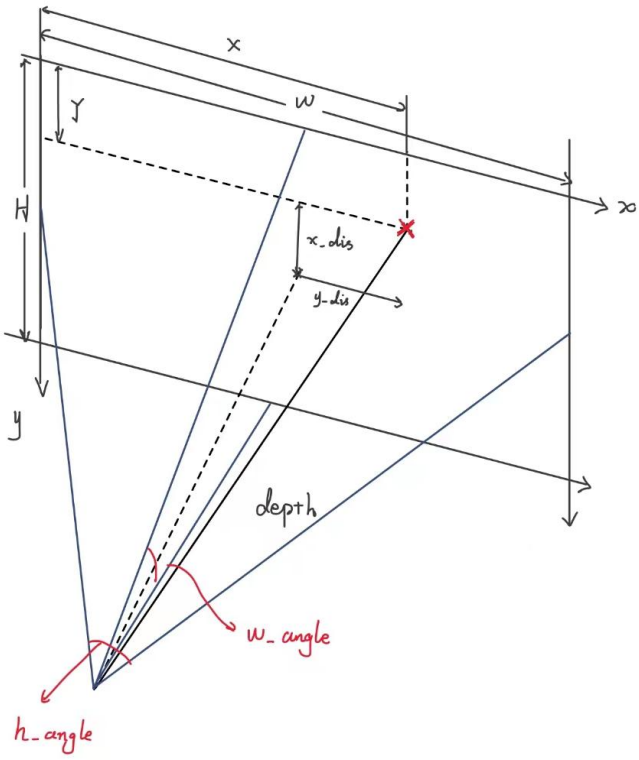


Figure 5: Ratio matrix

$$Ratio[X, Y, 0] = sin((X - W/2.0) * (W_{angle}/180/W * \pi) \quad (1)$$
$$Ratio[X, Y, 1] = sin((H/2.0 - Y) * (H_{angle}/180/H * \pi) \quad (2)$$
$$W_{angle} = 58.4° \quad (3)$$
$$H_{angle} = 45.8° \quad (4)$$
$$W = 640 \quad (5)$$
$$H = 480 \quad (6)$$

In this matrix, it uses some approximation to make the calculation easier. For example, the angle is assumed to be in direct proportional to the relative pixel location. That means the item scale is set to be much smaller than the distance. This makes the processing speed much faster, but does not influence the performance of the function.

However, we found that the scale item cannot be assumed to be much smaller than the distance. But this kind of the assumption does not change the performance of the function greatly.

To get better results, we have tested a more accurate ratio matrix:

$$Ratio2[X, Y, 0] = \frac{tan(0.5*H_{angle})*2*(X-\frac{W}{2}))}{\sqrt{(tan(0.5*H_{angle})^2*4*(X-\frac{W}{2})^2+\frac{tan(0.5*W_{angle})^2*4*(\frac{H}{2}-Y)^2*W^2}{H^2}+1}} \quad (7)$$
$$Ratio2[X, Y, 1] = \frac{tan(0.5*W_{angle})*2*(\frac{H}{2}-Y))}{\sqrt{\frac{tan(0.5*H_{angle})^2*4*(X-\frac{W}{2})^2*H^2}{W^2}+tan(0.5*W_{angle})^2*4*(\frac{H}{2}-Y)^2+1}} \quad (8)$$

This ratio matrix can get more accurate result of the x, y-coordinate value of the object. However, it costs more time. In tracking function, the processing time for one image is quite important. Therefore, after testing on the robot, we found that two matrices do not have great performance. Furthermore, the processing speed limits the performance of matrix 2 in complex environment. In this calculation, the precise location offset is not very meaningful, and an inaccurate offset with correct direction can achieve great performance. We decide to use matrix 1 finally with some approximation made to increase the calculation speed.

After getting the ratio matrix, we can get the physical coordinate value of the pixel (X,Y) by:

$$X_{dis} = Ratio[X, Y, 0] * depth[X, Y] \quad (9)$$
$$Y_{dis} = Ratio[X, Y, 1] * depth[X, Y] \quad (10)$$

After that, we can determine the location of the pixel $(X_{dis}, Y_{dis}, depth)$.

3.4.3 Interested box and distance range

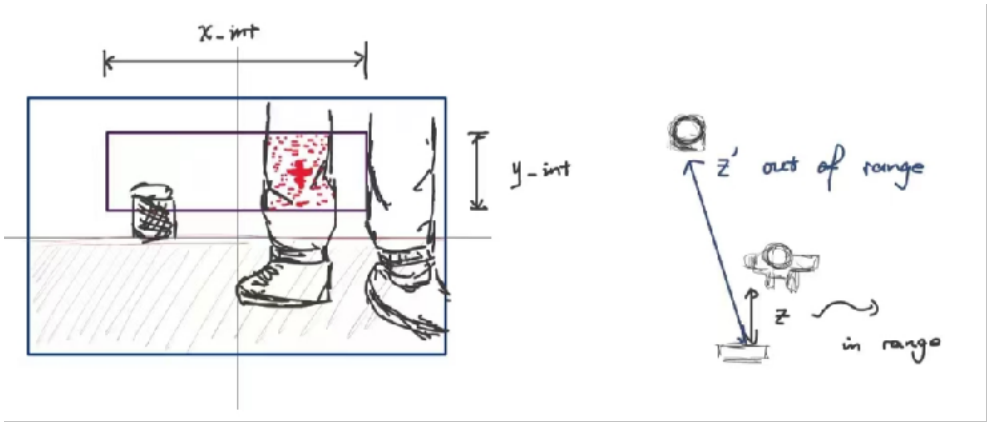


Figure 6: Interested box

After another location matrix of the pixels is determined with size $640 \times 480 \times 3$, we need to set an interested box on the image, as shown in Fig. 6. Only the pixels in the box will draw our attention. We need to determine a range of the distance we want to observe. If one pixel is in the interesting box, and its depth is in the set range, it will be added into the array "sampled points". After collecting all the sampled points, we will calculate the geometric center of those sampled points and get X_{target} and Y_{target}. Among are the points, we select the minimum depth and set it to be Z_{target}. Then, $(X_{target}, Y_{target}, Z_{target})$ will be the robot's target. The robot movement velocity will be:

$$\Omega_{angular} = -X_{target} * x_{scale} \quad (11)$$

$$V_{linear} = (Z_{target} - z_{goal}) * z_{scale} \quad (12)$$

3.4.4 Navigation publishing

After getting the linear and angular speed of the robot, we use the following commands to publish the velocity data on topic "geometry_msgs".

$$geometry\_msgs :: TwistPtrcmd(newgeometry\_msgs :: Twist());$$
$$cmd- > linear.x = (z - goal\_z\_) * z\_scale\_;$$
$$cmd- > angular.z = -x * x\_scale\_;$$
$$cmdpub\_.publish(cmd);$$

3.5 Improvement by range reduction

In this algorithm, we have check the sampling points in several images. One important failure is that when there are multiple objects locating in the range of the target distance. This case often happens when there is a close item behind the user with quite low distance.

To avoid adding multiple items into geometric center calculation, we have added a clustering module to avoid the influence of the object behind the target user. We set the closest distance to be Z. When the closest point is detected with the distance Z, the function will ignore the points with the distance larger than $Z + r$. Usually, the edge of the human will not be 20 centimeters far away from the front part. Therefore, we set "r" to be 0.2. In this way, our program can avoid the influence of the walls or other objects in the range.

After adding the range reduction, the robot is much less attracted to other objects.

3.6 Drawbacks and flaws

The GUI of tracking fail to be put in practical use. When we use the GUI on display to launch the tracking function, the display will be turned off when the control board is building the connection with Raspberry Pi. We have observed that the battery output voltage dropped greatly. The standard output voltage is 12.0 volts, and when camera, control board and display is working together, the output voltage will drop to 7.5 volts.

We have replaced the display, increased the battery voltage and replace the voltage stabilizer module. However, the problem cannot be solved. Finally, we have used a 24-inch display with independent power supply. This time, all the functions work

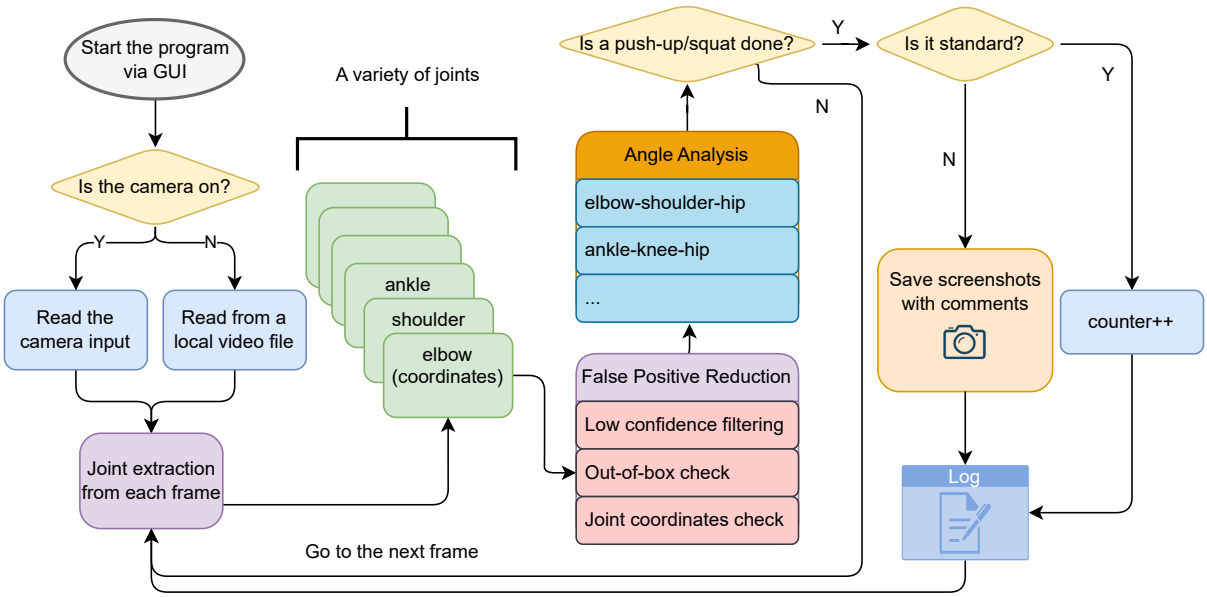


Figure 7: The overview of the pose evaluation pipeline

well. Nevertheless, the display should be connected will wire, and the robot cannot move in free.

4 Pose evaluation system

4.1 Overview

The design procedure for the Pose Evaluation module encompassed several stages, which were meticulously executed using the robust Mediapipe framework. Each stage played a vital role in accurately analyzing the movements of individuals performing push-ups or squats in the gym. This section provides a concise overview of the key stages involved in the design process, highlighting the extraction of joints, false positive reduction, angle analysis for identifying exercise start and end points, and the feedback generation. For reference, an overview of the designed pipeline is depicted in Figure 7. We have also open sourced the code for this part[1].

4.2 Components

The Pose Evaluation module is implemented in Python, utilizing the Mediapipe framework, and is mainly composed of the following components: the **counting mechanism** and the **feedback generation**. The implementation was thoughtfully organized in a modular manner, providing users with the flexibility to switch between different components seamlessly. For instance, users have the option to select either the camera or a

[1]https://github.com/unw9527/AI-Gym-Robot

```
for lndmrk in mp_pose.PoseLandmark:
    body_part_name = str(lndmrk).split(".")[1] # body_part_name is e.g. "NOSE"
    body_parts[body_part_name] = [
        landmarks[mp_pose.PoseLandmark[body_part_name].value].x,
        landmarks[mp_pose.PoseLandmark[body_part_name].value].y,
        landmarks[mp_pose.PoseLandmark[body_part_name].value].visibility
    ]
```

Figure 8: Coordinates and confidence of each joint collected using the Mediapipe framework

local video file as the input source, as well as the choice between the push-up counter or the squat counter. More importantly, the Pose Evaluation module can be completely detached from the robot, allowing users to use on any platforms including Windows, MacOS and Linux.

The Counting Mechanism The development of the counting mechanism involves meticulous implementation of Python scripts that leverage the capabilities of the Mediapipe framework. These scripts capitalize on the framework's existing components, such as the joint recognition model, which were customized to cater to the specific requirements of the counting mechanism.

Furthermore, the design of the scripts prioritized their easy integration with the robot, facilitating the robot's utilization of the counting mechanism to accurately tally the number of push-ups and squats performed by the user. This seamless integration enhances the overall functionality and usefulness of the robot as an exercise guide.

One of the primary challenges encountered in developing the counting mechanism was determining the correct execution of a push-up or squat. Existing methods often rely on simplistic patterns, such as monitoring the angles of the elbows and shoulders. However, these methods lack robustness when confronted with variations in users' movements.

To overcome this limitation, a more robust method was developed to accurately detect the start and end points of each push-up or squat. This method is based on the logical reasoning that the user's position should adhere to certain principles. For example, a person cannot do a squat while he is laying down. Additionally, the method incorporates a comprehensive analysis of various angles, evaluating whether they fall within a range predefined by extensive experiments. By extensively investigating into Mediapipe, we found that Mediapipe provides a convenient way to set limit on the variance in coordinates, as the coordinates of each joint, as well as a confidence value, are returned by the framework, as shown in Fig 8. Based on this information, the method

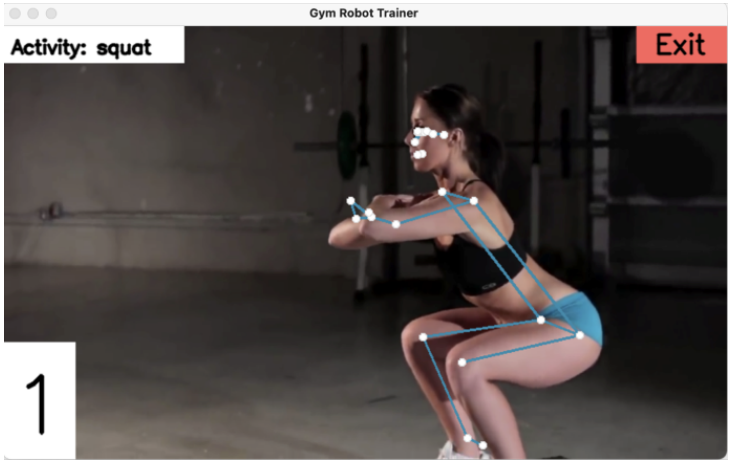


Figure 9: The visualization of the interface

determines the completion of a full push-up or squat.

By adopting this approach, the counting mechanism becomes more adept at accurately identifying the completion of each exercise repetition, accommodating variations in users' movements.

To enable the robot to better interact with the user, we also developed a visualization tool to help users see themselves during the exercise and the number of push-ups/squats recorded by the counter. The tool is implemented using the OpenCV library and shows the video feed from the camera with the counter overlaid on the video. Fig.9 is an example view of the interface where we used an online video[2] for the demonstration purpose.

This interface serves as a specification outlining the input/output requirements of the counting mechanism, ensuring compatibility and smooth communication with the other modules.

The Feedback Generation The feedback generation module is implemented using the logging library as well as the OpenCV library in Python. The logging library (i.e. logger) explicitly records the timestamp and the number of repetitions performed by the user, which can be used for further analysis. The logger also records which repetition is performed incorrectly, assisting in generating feedback for the user. Fig. 12 shows an example of the logger output.

The feedback generation module also utilizes the OpenCV library to capture the screenshot of the frame when the user performs an incorrect movement. The screenshot is then saved to the local disk, and the body part that is not in the correct position is

[2]https://www.youtube.com/watch?v=xqvCmoLULNY. Note that we can also use the camera to capture live video.

Figure 10: Illustration of the feedback collected from extensive experiments.

pointed out by using the OpenCV library to leave a comment on the screenshot.

In scenarios where the majority of the angles fall within the desired range while a few deviate from it, the system has been designed to capture a screenshot of the frame and emphasize the incorrect angles. We have conducted extensive experiments and the result can be found at Fig. 10

On the other hand, if the majority of angles detected are outside the predefined range, indicating a significant deviation from the expected exercise movements, the system identifies the movement as a random or unrelated action. In such cases, the system refrains from counting it as a valid exercise repetition to ensure accurate tracking and recording of the user's performance.

By employing these mechanisms, the system effectively distinguishes between correct exercise executions and random movements, promoting precision and reliability in counting the number of completed repetitions. This approach encourages users to maintain proper form and technique, facilitating effective exercise guidance and progress tracking.

5 GUI system and report generation

5.1 Introduction and requirements

As our system is required to be user friendly, we decided to integrate all funcitons into one GUI and also after the user finish exercising, the report could be displayed in a streightforward way. We designed a GUI where the user could enter and exit every function by pressing buttons, and the basic info of exercise could be displayed in a local webpage.

Our requirements of GUI and Feedback Webpage includes:

1. The buttons should be big, clear and user friendly.
2. The buttons should be responsive and run the correct program or direct to the correct page when pressed.
3. The report should be straightforward and includes the basic info of exercise.
4. The report should display examples of incorrect gesture if there is any.

5.2 Methodology

The Graphical User Interface (GUI) for the Exercise Counter application is implemented using the tkinter library in Python. The GUI consists of a main window with various components, including labels and buttons. When the GUI is initialized, the welcome message is displayed using a label. The start button is provided to initiate the exercise counting process. Upon clicking the start button, additional buttons for exercise counting and report generation are displayed, while the start button is hidden. The exercise counting buttons allow users to count squats and pushups by running specific programs using the run_program_1 and run_program_2 methods, respectively. The GUI also includes buttons to generate HTML reports, which are opened in the default web browser using the open_local_html method.

The GUI provides a user-friendly interface for interacting with the Exercise Counter functionalities. Users can easily navigate through the GUI by clicking the appropriate buttons to initiate exercises, generate reports, and track objects. The GUI structure and functionality can be extended to incorporate additional features as needed. Overall, the Exercise Counter GUI enhances the usability of the application and simplifies the exercise counting process for users.

The screenshots of the GUI are shown below.

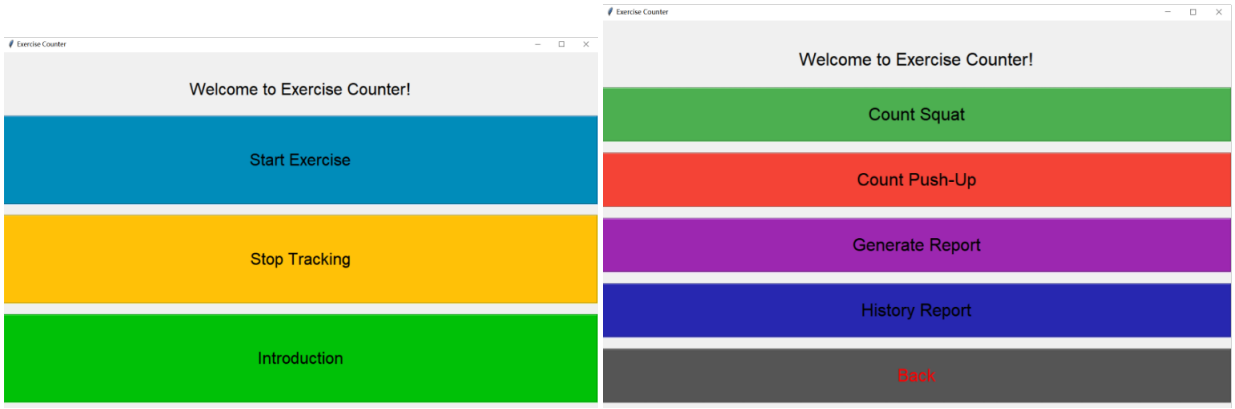


Figure 11: Graphical User Interface

The exercise report generation code extracts information from a log file generated during an exercise session and uses it to create an HTML report. The report includes exercise-specific details such as the exercise type, total counts, and time taken. A screenshot of the log file is shown below.

```
2023-05-18 16:07:09,991 INFO:Starting to do push-up
2023-05-18 16:07:10,023 INFO:Video source: push-up-example.mp4
2023-05-18 16:07:20,881 WARNING:Push-up 1: Your elbow angle is too large
2023-05-18 16:07:21,019 INFO:Push-up 1 done
2023-05-18 16:07:22,373 WARNING:Push-up 2: Your elbow angle is too large
2023-05-18 16:07:22,505 INFO:Push-up 2 done
2023-05-18 16:07:23,772 INFO:Push-up 3 done
2023-05-18 16:07:24,960 WARNING:Push-up 4: Your hip angle is too small
2023-05-18 16:07:25,008 WARNING:Push-up 4: Your hip angle is too small
2023-05-18 16:07:25,224 WARNING:Push-up 4: Your hip angle is too small
2023-05-18 16:07:49,736 INFO:Push-up 4 done
2023-05-18 16:08:10,330 WARNING:Push-up 5: Your elbow angle is too large
2023-05-18 16:08:10,463 INFO:Push-up 5 done
2023-05-18 16:08:12,165 WARNING:Push-up 6: Your elbow angle is too large
2023-05-18 16:08:12,214 WARNING:Push-up 6: Your elbow angle is too large
2023-05-18 16:08:12,347 INFO:Push-up 6 done
2023-05-18 16:08:14,197 WARNING:Push-up 7: Your elbow angle is too large
2023-05-18 16:08:14,246 WARNING:Push-up 7: Your elbow angle is too large
2023-05-18 16:08:14,337 INFO:Push-up 7 done
2023-05-18 16:08:15,188 INFO:Push-up 8 done
2023-05-18 16:08:15,394 INFO:Total number of repetitions: 8
2023-05-18 16:08:15,395 INFO:Used time: 0:01:05.351191
2023-05-18 16:08:15,395 INFO:Finish exercising. Thanks for using Gym Robot Trainer!
```

Figure 12: Log File

The report also displays screenshots of any improper gestures detected during the exercise. The code parses the log file, selects a random subset of images from a directory, and inserts them into the HTML template along with the extracted information. The generated report is then saved as an HTML file. Overall, the code combines log file parsing, HTML template population, and image selection to produce a comprehensive and visually appealing report summarizing the exercise session and highlighting any areas of improvement. A screenshot of the report webpage is shown below.

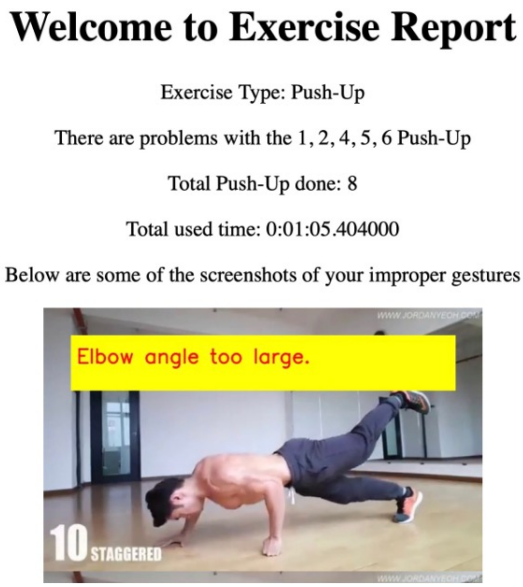


Figure 13: Exercise Report

5.3 Results and discussion

All the bottoms are responsive and friendly to amateur users and report generation is smooth and clear as well. However, there is plenty of room for us to improve, for example, we could add videos of guidance and GIF of proper movements to our report, and we could also create a mobile app so that the user could now view their exercise results on a mobile phone.

6 Verification

6.1 Tracking system

6.1.1 Verification procedure

1. When the human is walking in speed under 1m/s on the flat ground without any confounding items, check whether robot is able to focus on the human, and keep the tracking.
2. When the robot is doing the tracking, measure the distance between robot and human which should be kept in the range of 0.5m to 0.9m. When the robot finishes tracking, the distance should be 0.7m with error smaller than 0.1m.
3. The human changes the direction, the robot should be able to turn the head to keep the person in the center of its view, and -0.35m to 0.35m in scale.
4. Test the tracking function under different light conditions

5. When the robot is doing tracking, check if the scenery of the robot is displayed on the screen.

6.1.2 Verification results

1. The robot can do the tracking smoothly in flat ground and the environment without the obstacles.When people is walking under the speed of 1m/s, the robot can keep the tracking.
2. The distance can be kept in 0.5m and 0.9m for most of the time. The maximum distance is 0.94m and the lowest distance is 0.47m. The robot will only do the distance adjustment when the distance is out of range, so little overflow occurs in the peaks and ebbs, which is acceptable. When the robot reaches stable point, the distance is always in the range.
3. The robot's linear speed and angular speed allows the robot to keep the distance and make the user in the center of its view (-0.35m - 0.35m).
4. The ROS depth camera is equipped with thermographic camera. It can adapt different kinds of the light condition.
5. When the tracking function is running, the display can show the view of the robot.

6.2 GUI system

6.2.1 Verification procedure

We have invited five people who does not have any background information to use our robot. We need to determine have many of them can use all of our functions without extra guidance except for the instruction on the GUI.

6.2.2 Verification Results

Four people can understand the function of robot and take use of the movement recognition system and movement evaluation system. One people need some extra help to understand the logic of GUI.

6.3 Pose Evaluation System

6.3.1 Verification procedure

We have invited two different testers to do the squats and two testers to do the push-ups in different side views(left, right) and light conditions(normal, against the light).

The total number for one set of test is ten, and we will count the number of standard results, non-standard results and missing results to get the accuracy.

6.3.2 Verification Results

Test of Squats												
Conditions			**Results (S=standard, N=non-standard, M=missing)**									
Tester	***Side View***	***Light***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***Accuracy***
Dalei Jiang	Left	Normal	S	N	S	S	S	S	N	S	S	S	100%
Dalei Jiang	Right	Normal	S	S	N	S	S	S	S	S	M	S	90%
Zifei Han	Left	Normal	N	S	S	S	N	S	S	S	S	S	100%
Zifei Han	Right	Normal	S	S	S	S	N	S	N	S	S	N	100%
Zifei Han	Left	Against the Light	S	S	S	N	S	M	S	N	S	S	90%
Total Accuracy: $48/50 \times 100\% = 96\%$												

Table 1: Test of Squarts

Test of Push-ups												
Conditions			**Results (S=standard, N=non-standard, M=missing)**									
Tester	***Side View***	***Light***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***Accuracy***
Chang Liu	Left	Normal	N	S	N	S	S	S	S	N	S	M	90%
Chang Liu	Right	Normal	S	M	N	S	S	N	S	S	S	S	90%
Kunle Li	Left	Normal	S	S	S	N	N	S	S	S	S	N	100%
Kunle Li	Right	Normal	S	N	M	S	S	S	S	S	N	S	90%
Kunle Li	Left	Against the Light	S	S	N	S	S	N	S	N	S	S	100%
Total Accuracy: $47/50 \times 100\% = 94\%$												

Table 2: Test of Push-ups

7 Cost and schedule

7.1 Cost Analysis

7.1.1 Labor

Our fixed development costs are estimated to be ¥15/hour, 10 hours/week for everyone.

weekly cost = 15 RMB/hours $\times$ 10 hours/week $\times$ 8 weeks $\times$ 4 = 4800 RMB

7.1.2 Parts

Part	**Vendor**	**Cost (CNY)**	**Qty**	**Total(CNY)**
Depth Camera	Taobao	980.00	1	980.00
Raspberry Pi 4B Main-board	Taobao	855.00	1	855.00
Rplidar A1 Omnidirectional Ultrasonic Radar	Taobao	269.00	1	269.00
7 Inch touching display screen	Taobao	260.00	1	260.00
High-torque motor	Taobao	48.25	4	193.00
Metal chassis and Mecanum wheel	Taobao	98.00	1	98.00
Acrylic plate frame	Taobao	56.59	1	56.59
Differential PID motor speed regulating drive board	Taobao	150.00	1	150.00
GY-85 Nine-Axis Gyroscope	Taobao	70.51	1	70.51
Bluetooth speaker	Taobao	58.00	1	58.00
16G SD card and SD card reader	Taobao	34.90	1	34.90
12V lithium batery	Taobao	3.00	9	27.00
Data cables and adapting pieces	Taobao	10.00	1	10.00
Wired keyboard	Taobao	58.00	1	58.00
Total	—	—	—	3120.00

Table 3: Cost List

7.1.3 Total

Total cost estimate: 7920 RMB

7.2 Schedule

Week	Dalei Jiang	Zifei Han	Chang Liu	Kunle Li
3/20	Assemble the robot hardware devices and set up the OS and environment	Documentation reading about ROS and build (later update) lab notebook	Paper reading and environment setup	Environment Setup
3/27	Use the Rviz to set the navigation task management system in ROS	Explore and implement several algorithms on navigation in rospy	Try to launch the detection model on the laptop	Push-up count & motion evaluation
4/03	Based on OpenCV, apply the KCF algorithm on PC, and use the existing video to test	Documentation reading about KCF of recognition of human figure	Write the GUI and complete the related design of the scripts	Integrate the individual methods
4/10	Move the algorithm function to the robot, and connect the camera as input	Deploy the algorithms function onto the platform and debug	Deploy the algorithms function onto the platform and debug	Implement the GUI and adjust the adaption of the application
4/17	Set up easy-to-call scripts for target tracking function	Further debug and double check the interaction between software and hardware	Help Kunle combine GUI and Model	Implement the GUI and adjust the adaption of the application
4/24	Help Kunle migrate the system onto robot	Help Kunle migrate the system onto robot	Help Kunle migrate the system onto robot	Migrate the system onto robot
5/01	Test the function and debug	Test in expected usage situations and debug	Test the function and debug	Debug and enhance
5/08	Organize materials and write final report	Test in real applicable situations and debug	Organize materials and write final report	Exhaustive tests to improve stability

Table 4: Schedule

8 Ethical considerations

8.1 Privacy protection

All our processing and the data collection are done off-line, and we do not update any data to online server. The IEEE Code of Ethics indicated that engineers should ”protect the privacy and confidentiality of their clients or employers' information, including personal information.” [6] In ACM Code of Ethics, it mentions that ”The engineers should respect the privacy of users, collect as little user information as possible, and avoid information disclosure.” [1] On our project, none of the users' privacy will be collected, analyzed or published out side of the robot hardware.

8.2 Avoid discrimination

According to the IEEE Code of Ethics, We should not discriminate or treat users differently based on their race, age, gender or other factors. [6] In our program, our robot will only give comments on users' performance on doing exercise. We should avoid making comments on users' body shape or physical fitness levels.

8.3 Avoid physical and mental harm

According to the IEEE Code of Ethics, we need to avoid all forms of harm. Our robot will keep a distance from the user while moving to avoid collisions. [6] At the same time, we isolate all live equipment to prevent direct contact. Furthermore, we have paid attention to the feedback we offer for the user, to make the feedback positive and supportive.

9 Conclusion

9.1 Accomplishments

The gym robot system has been successfully developed and verified, demonstrating robust functionality and performance. The robot is capable of autonomously tracking a person within the gym, ensuring that the user can do exercise wherever wanted, without the hassle of moving the robot around. Once the person initiates their exercises by starting the exercise mode on the robot, the robot transitions to the pose evaluation module, which analyzes the individual's posture and generates a comprehensive feedback report for correction and improvement.

The robot is also capable of counting the number of repetitions performed by the user, providing a convenient and reliable tool for tracking exercise progress. The counter is

shown on the touch screen of the robot at run time, and the user can check the number in the report after he finishes exercising.

9.2 Uncertainties

Despite the successful development and evaluation of the gym-tracking robot system, there are certain uncertainties that should be acknowledged. These uncertainties stem from various factors and aspects of the project, and they should be considered for future improvements and advancements in the system.

- **Accuracy under different camera configurations**: The Pose Evaluation module exhibited reliable performance when analyzing standard movements. However, challenges arose when the user is facing various directions. This discrepancy raises uncertainties regarding the accuracy of the Pose Evaluation module under different camera configurations, distances, and angles. Further investigations and refinements are necessary to address this limitation and enhance the system's robustness.
- **Generalizability to diverse user populations**: The gym robot system has been evaluated using a limited set of users and exercise routines. Due to the limited availability of resources and time, the system's performance has not been evaluated across a broader range of users with varying body types, fitness levels, and exercise techniques. Future studies involving larger and more diverse user populations can provide valuable insights into the system's performance across a wider demographic.

Addressing these uncertainties through further research, and development will contribute to the refinement and advancement of the gym robot system.

9.3 Future work and alternatives

Firstly, the power supply problem should be solved. Currently, our robot can only use 24-inch display screen with individual power supply to make the tracking function GUI work. To achieve complete GUI operation of functions, we need to set another battery-display system, and modify current metal frame to fasten new battery.

Then, we can build more functions in movement recognition system, such as sit-ups and Bur-pees, and allows the robot to instruct more types of movement.

Besides, we can develop more peripheral applications, such as mobile phone APP to make the product more user-friendly.

References

[1] ACM, "Acm code of ethics," 2018. [Online]. Available: https://www.acm.org/code-of-ethics (visited on 03/10/2023).

[2] M. Andriluka, U. Iqbal, E. Insafutdinov, *et al.*, "Posetrack: A benchmark for human pose estimation and tracking," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5167–5176. DOI: 10.1109/CVPR.2018.00542.

[3] M. Chang, C. Zhang, Y. Chen, W. Chen, Y. Chen, and Y. Tsai, "Real-time multi-person squat detection and counting for group fitness videos," *IEEE Access*, vol. 8, pp. 15 248–15 259, 2020.

[4] Google, "Mediapipe: A framework for building pipelines to process media data," *Google AI Blog*, 2019. [Online]. Available: https://ai.googleblog.com/2019/03/mediapipe-framework-for-building.html.

[5] J. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[6] IEEE, "Ieee code of ethics," 2016. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/10/2023).

[7] J. W. Keogh, P. W. Winwood, and P. T. Nikolaidis, "Injury prevalence and severity in fitness athletes: Association with training experience," *Journal of Strength and Conditioning Research*, vol. 20, no. 4, pp. 855–860, 2006.

[8] W. D. Lee, *The turtlebot follower demo*, http://wiki.ros.org/turtlebot_follower/Tutorials/Demo/, Accessed March 31, 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[10] A. Vij, *Push-ups with python + mediapipe = open*, https://aryanvij02.medium.com/push-ups-with-python-mediapipe-open-a544bd9b4351, 2021.

[11] C. Wang, L. Xu, J. Liu, and D. Tao, "Human physical activity recognition by multi-modal deep learning method," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3656–3666, 2019.

Appendix A Requirements and Verification Table

Subsystem	Requirements	Verification (Y for success)
Human Tracking System	1. The robot can keep the user in the center of the scenery (occupying the center rectangle of half width and height).	1. Verify by activating another node to show the view of the robot on monitors and check if the user locates in the supposed region. (Y)
	2. The robot can follow the user who walks at a normal speed in a straight line, and keep the distance at 0.7 meters.	2. Verify by walking in a straight line in the view of the robot and measuring the distance between them. (Y)
	3. The robot can follow the user who turns left or right at a normal speed, and keep the distance at 0.7 meters.	3. Verify by first standing in front of the robot and then turning left or right, and measuring the distance between them. (Y)
	4. The robot can avoid being influenced by the surrounding items located at least 0.4 meters that it passes by.	4. Verify by walking by a few manually placed obstacles at around 0.4 meters and observing if the robot can still follow the user. (Y)
Pose Evaluation System	1. The robot can determine whether the user has done the supposed type of movement under certain modes.	1. Verify by doing the supposed type of movement in front of the camera and see if the system returns a count increase or a message for non-standard poses. (Y)
	2. The robot can count and show the number of standard movements, having an accuracy of 80% or higher.	2. Verify by doing 10 standard movements in a row for 5 times, and checking if the average counted number of movements reaches 8 or larger. (Y)
	3. The robot can reach an accuracy of 90% or higher in collecting standard or non-standard (but recognized) movements from all movements.	3. Verify by doing 10 movements in a row for 5 times, and checking if the average collected number of movements reaches 9 or larger. (Y)
GUI System and Report Generation	1. The buttons should be responsive and run the correct program or direct to the correct page when pressed.	1. Verify by clicking all buttons on the GUI for either tracking, exercising, or showing introductions and checking if they work as supposed. (Y)
	2. The generation of the log and report should be complete to record all movements.	2. Verify by checking in the report if the number of movements done equals the one shown in the report. (Y)
	3. The report should describe with joint angle conditions how each non-standard movement is judged as so.	3. Verify by checking the images in the report to see if the number of them equals the difference between standard movements and collected movements. (Y)