The University of Hong Kong

Faculty of Engineering

Department of Computer Science

COMP 4801 Final year Project 2023/24

Intelligent Robot Design and Implementation

Final Report

Group: FYP23005

Chan Hiu Tung (3035705180)

Supervisors:

Dr. T. W. Chim

Mr. David Lee

Second Examiner:

Dr. C. K. Chui

Content Page

Abstract 4
1 Introduction 5
1.1 Background 5
1.2 Motivation 5
2 Literature Review 7
2.1 Hierarchical Controller Algorithm (CHAMP) 7
2.2 Inverse Kinematics 8
2.3 Unified Robot Description Format (URDF) 9
2.4 ROS Control Framework 9
2.5 Web-based application 10
3 Methodology 15
3.1 Hardware 15
3.2 Firmware 16
3.2.1 Programming Framework 16
3.2.2 PWM digital servos 16
3.2.3 IMU 17
3.2.4 ROS Serial 17
3.3 CHAMP 18
3.3.1 Hardware Interface 18
3.3.2 URDF Model 18
3.3.3 Self-balancing algorithm 19
3.4 Development Platform 20
3.5 Computer Vision 22
3.5.1 3D Scene Reconstruction 22
3.5.2 Autonomous Navigation 23
3.5.3 Dynamic Object Following 23
3.5.4 Frontier Exploration 25
3.5.5 Object Detection 25
3.6 3D printing and models 26
4 Web-based application 28
4.1 Structure 29
4.2 Features 30
4.2.1 Live Streaming 30
4.2.2 URDF Web Viewer 30
4.2.3 User Authentication 31
4.2.4 Notification System: 31
4.3 Technologies Used 32

4.3.1 ReactJS 32
4.3.2 ROS Web Tools (roslibjs) 33
4.3.3 Three.js 33
4.3.4 Leva 33
4.3.5 Firebase 34
4.4 UI/UX Design 36
4.4.1 User-Friendly Design 36
4.4.2 Responsive Design 36
5 Experiments and Results 38
5.1 Hardware 38
5.2 Self-balancing algorithm 40
5.3 Computer Vision 42
6 Future Work 44
7 Conclusion 47
Bibliography 48

Abstract

Robotics has emerged as a popular topic in the field, especially during and after the COVID-19 pandemic, where robots are used in various businesses and industries to avoid face-to-face interaction. There exists a large number of robot commodities in the market, however, most of them are expensive. In this project, our team aims to develop an easily accessible and relatively affordable 3D-printed quadruped robot dog which is equipped with machine learning and computer vision technologies. The robot will be able to reconstruct the environment in 3D, detect objects and follow a target even if there is no pre-existing map available. To enhance user interaction and the usability of the product, a web-based application is also developed, providing users with the live streaming video captured by the robot's camera, and the 3D simulation of the robot dog for observing its movements.

The idea proposed in this project can be applied in various scenarios, such as rescue operations during natural disasters or accidents. With its small size and ability to navigate unknown environments, the robot dog could be deployed in places that are inaccessible to humans. It can assist in detecting and tracking any human objects and return the live streaming video and relevant data to the web-based application, facilitating the rescue teams in the rescue work and decision-making.

1 Introduction

1.1 Background

Robotics have always been a hot topic, and they have gained even more attention since the COVID-19 pandemic when industries and businesses such as restaurants, hotels, and customer services in shopping malls all adopted robots to avoid face-to-face interaction.

Currently, there are several robots for sale in the market, for instance, Boston Dynamic's Spot, Xiaomi's CyberDog etc. However, many of them are costly and the general public may not be able to buy a few of them to build a team of robot dogs.

Natural disasters and accidents are happening around the world, and it is a true tragedy when brave rescue team members get injured or even lose their lives while rescuing other victims. This motivates our team to think about some possible applications of robots in rescue work: is it possible to create a large team of robots helping humans to explore the unknown environment, notifying humans if any survivors are detected, and displaying the relevant data clearly?

1.2 Motivation

Our team is determined to develop a robot that is suitable for rescue operations. After natural disasters or accidents like fire, objects may fall, walls or floors may also be broken and many other things can block the road. Hence, we believe that developing a quadruped robot dog would be a good idea, as a quadruped robot has 4 legs and 12 degrees of freedom, which allows it to walk on surfaces that are not flat or even on rough terrain. Compared with robot cars which may only be operable on flat

surfaces without many obstacles, a quadruped robot seems much more suitable for the rescue job.

Considering the scenarios where houses are collapsed and there is not enough room for a human to get inside and rescue others, we decided to develop a robot dog that is smaller in size and can be deployed to areas that are too narrow and inaccessible to humans.

It is one of the most crucial things in rescue operations to detect and locate any survivor who is still stuck. Hence, cameras and sensors were planned to be installed on the robot dog, such that it is able to scan the surroundings and collect relevant data for further analysis. It is also our goal to develop computer vision functions that can follow a moving object and detect a specific object, as such functions would be immensely useful in identifying and tracking survivors.

Considering the rescue team members may have to deploy the robot in a distant and unknown place for searching survivors, the prebuilt map may not be available all the time. Therefore, we develop our robot in a way such that it can work without having any prior knowledge of the environment, furthermore, being able to help construct the map of the unknown place and avoid bumping into obstacles.
Finally, we believe that developing a web-based application which allows rescue team members to look at the live streaming video captured by the camera on the robot dog, as well as the movement of the robot would be useful.

2 Literature Review

2.1 Hierarchical Controller Algorithm (CHAMP)

CHAMP is an open-source development framework for building new quadrupedal robots and developing new control algorithms and is originally based on the MIT Cheetah robot [1].

2.1.1 Hybrid Dynamic Systems

CHAMP label the movements of the robot arm as the flight phase and the stance phase. The flight phase is when the machine is in the air, while the stance phase is when the machine is interacting with the ground. The dynamic changes whenever any leg touches down or lifts off the ground.

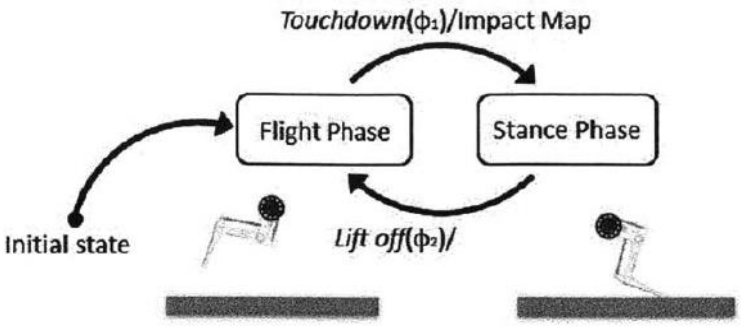


Figure 1 [1]

2.1.2 Virtual compliance

The virtual compliance mechanism of CHAMP prevents robot legs which are in the stance phase from slipping, as inspired by the equilibrium point hypothesis of how animals control the equilibrium point of their limb. Thus, the scholar claims that the leg force exerted on the ground "can be modulated through adjusting penetration depth of the virtual reference trajectory into the ground" and achieve compliance.

2.1.3 Control Framework

The control framework of CHAMP aims at tackling the three main challenges, which are the stabilization of the robot, control of ground reaction forces and modulation of the gait pattern. The paper proposed three ideas for tackling the challenges, which are implementing virtual compliance in robot legs to achieve self-stabilization, introducing penetration depth in foot-end trajectories to modulate ground reaction, and modulating the gait patterns with the velocity and sensory feedback.

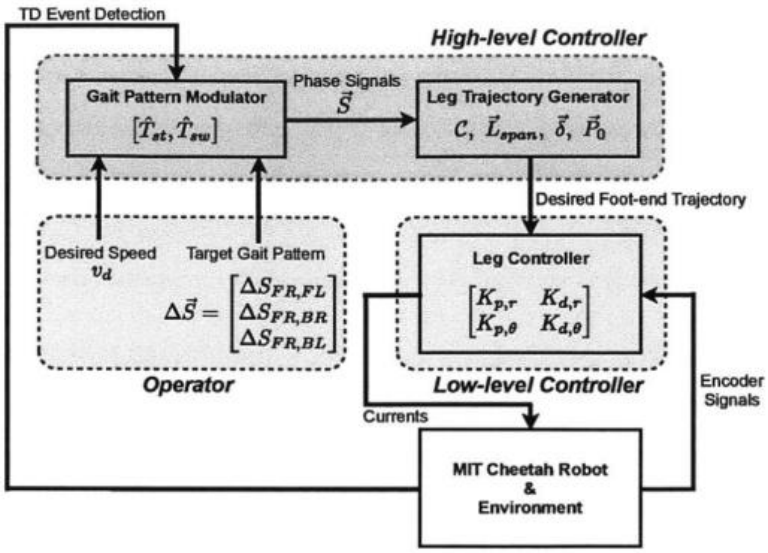


Figure 2 [1]

2.2 Inverse Kinematics

As mentioned above, the CHAMP calculates the desired food-end trajectories for the robot logs. To translate the commands into low-level motor controls, Inverse Kinematics (IK) is essential. The Inverse Kinematics algorithms can be written as $\theta = f^{-1}(x)$, where θ denotes the column vector giving the DoFs, and x refers to the vector of desired end effector positions [2]. Among all the θ, only the one that can return the most stable and the smoothest motion is needed.

The heuristic inverse kinematics algorithms can provide answers to an IK solution, which the Cyclic Coordinate Descent (CCD), and forward and backward reaching inverse kinematics (FABRIK) are examples of it. CCD aims at aligning each and every joint with its end effector and target. For FABRIK, the algorithm estimates and moves the joints to the target position iteratively.

2.3 Unified Robot Description Format (URDF)

Introduced by ROS developers, the Unified Robot Description Format is a format to describe the kinematics structure, dynamics parameters, and geometries of robots [3]. It can be used to describe a robot using a universal format and facilitate visualization and simulation of robots, as the robots can be imported and exported in this universal format easily.

In a URDF file, links and joints are recorded. A link refers to the rigid body that can be connected using joints. The name, inertial, visual and collision properties of a link are included. A joint is the connection between two links. The kinematics, dynamics and safety limits of a joint are also included.

In CHAMP, a URDF file can be loaded for the configuration of the robot legs. However, there are some assumptions and requirements on the URDF, as listed on the GitHub of CHAMP [4]. In this project, a URDF model is created using the STL models of the robot dog and is loaded in CHAMP.

2.4 ROS Control Framework

ROS control is a framework for controlling robots using ROS. Without the need for

additional codes, ros control can perform path planning for the robot arms with just some controller configuration files. The Hardware Abstraction Layer of the ROS control framework acts like a bridge to the real robots and simulated robots. For instance, with the robot commands from the joint command interfaces passed to the Hardware Abstraction Layer, it will write to the embedded controllers of the real robot by the write() function, which will then pass to the actuators and be responsible for the movement of the robot arm. Besides, hardware feedback or the states of the mechanism are also read by the read() function, which will then be passed to the joint state interface and enable the system to know the current states of the hardware, as shown in Figure 3 [5].

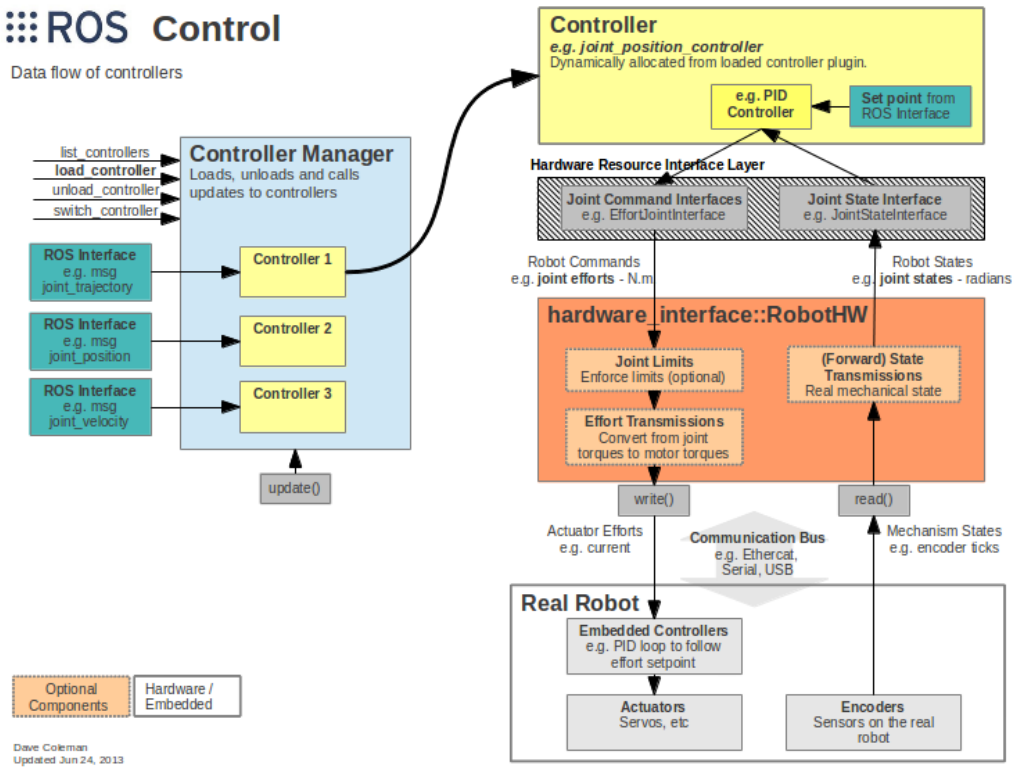


Figure 3[8]

2.5 Web-based application

There are some papers about web-based applications designed for robot control or 3D simulations, which provide us with great insights into the features and UI/UX design of our web-based application in this project. For example, in *Development of Web-based Application for Mobile Robot using IOT Platform*, the authors developed a web-based application for their mobile robot, which is a robot car with various sensors such as a gas sensor for measuring air quality. The team included a member login system, and mobile control which is similar to a joystick in their web-based application. The data collected by the sensors are also displayed in the form of graphs [6]. This paper provides us with some design ideas for our own web-based application, such as including a user authentication system and visualization of the collected data, which will be explained further in the parts discussing the features of our web-based application.

In the article *Web Based Robot Simulation Using VRML*, Rohrmeier discussed the application of virtual reality modellingq language (VRML) in web-based simulations and visualization of the 3D robot model in VRML. Rohrmeier concluded the advantages of VRML, such as the ease of using it as VRML can "run on any underlying platform" and only the web browser is required, but not any other hardware or software [7]. He also mentioned that the 3D interface improves the user experience and can make the program "international" as users can use the functions indicated by the signs and icons but not text in a certain language. Though this article was published in 2000 and VRML is slowly outdated, the article still provides us insights. First of all, it demonstrates the benefits of using web-based applications as the user interface of 3D models, which is the avoidance of specific hardware and software. Also, he also pointed out that for an easy-to-use 3D interface, users should be able to view the 3D model easily with the use of some buttons. This also inspired

us to add a control panel in our 3D simulation frame so that users can adjust the viewing angle even without prior knowledge (e.g. using the mouse to adjust the viewing position).

On the other hand, there are several websites showing 3D simulation which served as some good examples and ideas for us in this project. For example, the Eyes of Nasa (https://eyes.nasa.gov/) developed by NASA and the Jet Propulsion Laboratory of California Institute of Technology is a real-time 3D web-based application for users to view the 3D models of the solar systems and different planets. Users can zoom in, rotate the viewpoint and see the real-time movement of different planets and satellites. It even notifies the users about the current natural disaster or major events and allows users to click and explore more about that event, as shown in figure 4.

Figure 4 Eyes on the Earth

Another great example is the *Smithsonian 3D* developed by the Smithsonian museums. (https://3d.si.edu/) This website includes many 3D content of the exhibits in the Smithsonian museums in Washington D.C. Same as the Eyes of Nasa, the 3D

models on Smithsonian 3D can zoom in and adjust the viewing angle by rotating. As the website is mainly for educational purposes, it includes several functions intending to explain more about the exhibits. For example, as shown in Figure 5, on the right-hand side there is a separate section for the object details. For some of the exhibits, there are even some additional features that greatly improve the user experience and attractiveness, and in this case, further reinforcing the educational purpose. In Figure 6, users can click the buttons to turn on the annotation and see the name of the engines of the aeroplane, and even change the background/environment colour freely.

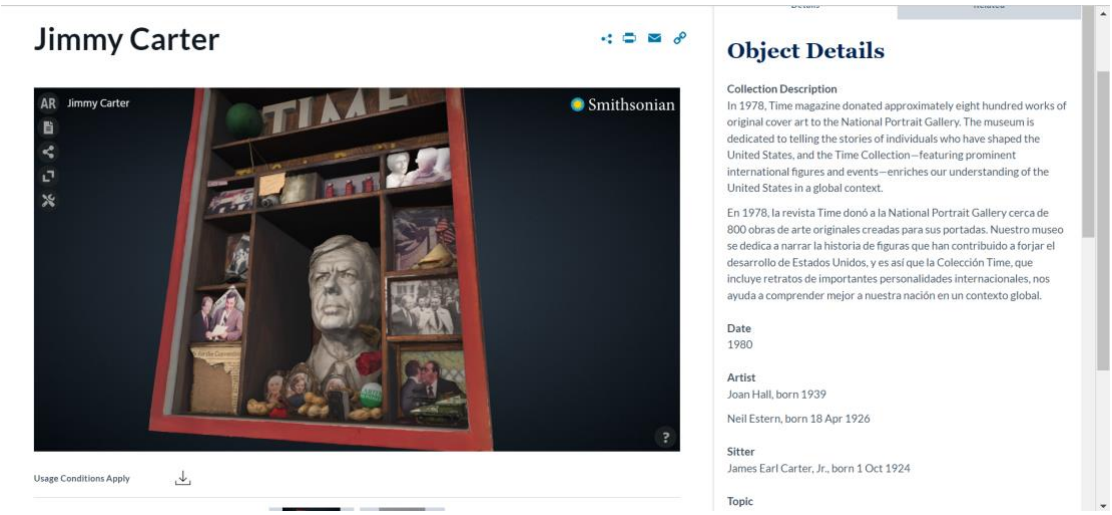


Figure 5

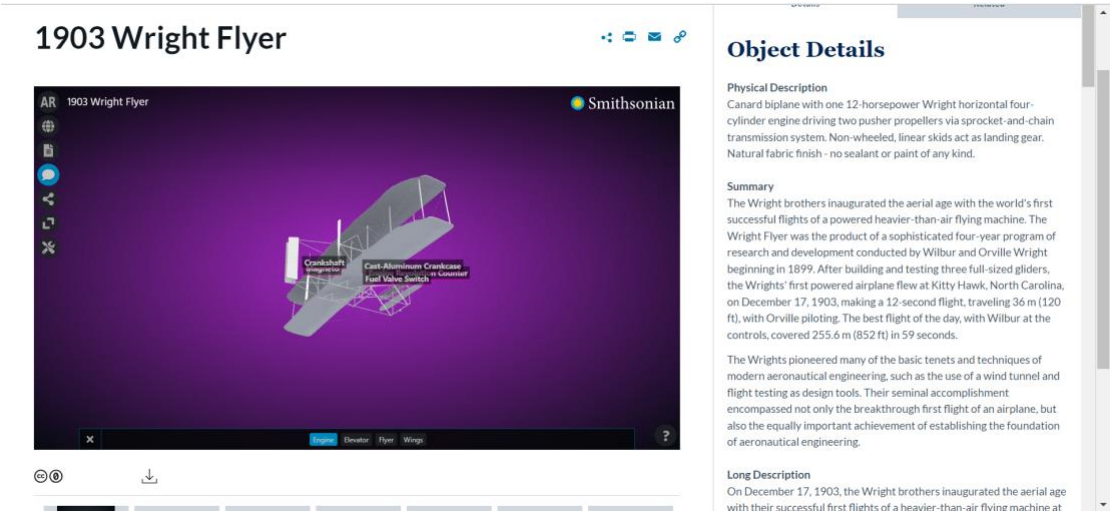


Figure 6

These two websites demonstrate interactive and attractive 3D simulations on the web platform. The features implemented in these websites, such as the ability to adjust the viewing angle, having a panel next to the simulation, the changing of background colours or the design of separating the whole web page into multiple sections so that users can view the 3D simulations and other information at the same time are being experimented, which will also be discussed in detail in following part discussing the features.

3 Methodology

3.1 Hardware

In this project, the following electronic components are utilized:

- ESP32 microcontroller is used and connected to ROS using the USB serial and coordinates components such as servo motors and IMU with ROS.
- MPU-9250, which is a 9-axis inertial measurement unit (IMU) that contains an accelerometer and a gyroscope. It supports functions such as measuring linear acceleration, angular rate and orientation etc. [9]
- Digital servos are used as the joint actuators, which receive and direct actions.
- Nvidia Nano Jetson is used as the single board computer (SBC) and is helpful in performing computer vision tasks such as image segmentation and object detection.
- Intel Realsense D435I camera, which is a stereo and IR camera with an IMU included. It can capture depth images and colour images, which is crucial in the computer vision part of this project, especially for the 3D reconstructions of the imaged scene.
- 7V Lithium-ion battery and external power bank to power the microcontroller and the SBC.

A Planar LiDAR (X4 ydlidar) was also added in addition to the above components, hoping to support perception and computer vision tasks. However, after trial, it was found that the LiDAR was too heavy and adding it would greatly affect the movement of the robot dog. As a result, our team decided to remove it and keep the stereo camera instead.

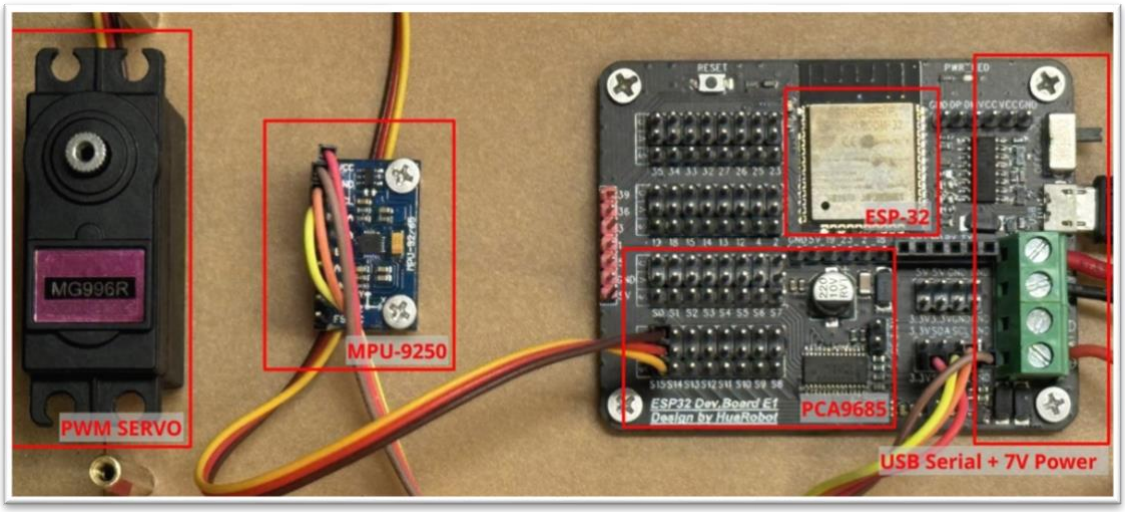


Figure 7

3.2 Firmware

3.2.1 Programming Framework

In this project, the Arduino framework with PlatformIO IDE is used. To control the electronic components and communicate with ROS, several libraries are being used. For instance, the Adafruit PWM Servo Driver Library for interfacing with the PCA9685 PWM driver chip, the Rosserial Arduino Library for transceiving ROS messages, and an extended version of the MPU-9250 Digital Motion Processing Arduino Library for interfacing with the MPU-9250 and DMP.

3.2.2 PWM digital servos

Using Adafruid PCA9685 PWM Servo Driver Library and functions setPWMFreq(freq) and setPWM(channel, on, off), we can set the PWM. Also, using the join_calibration package in CHAMP, we can turn the zero positions and orientation of the servo with regard to the URDF model and perform calibration.

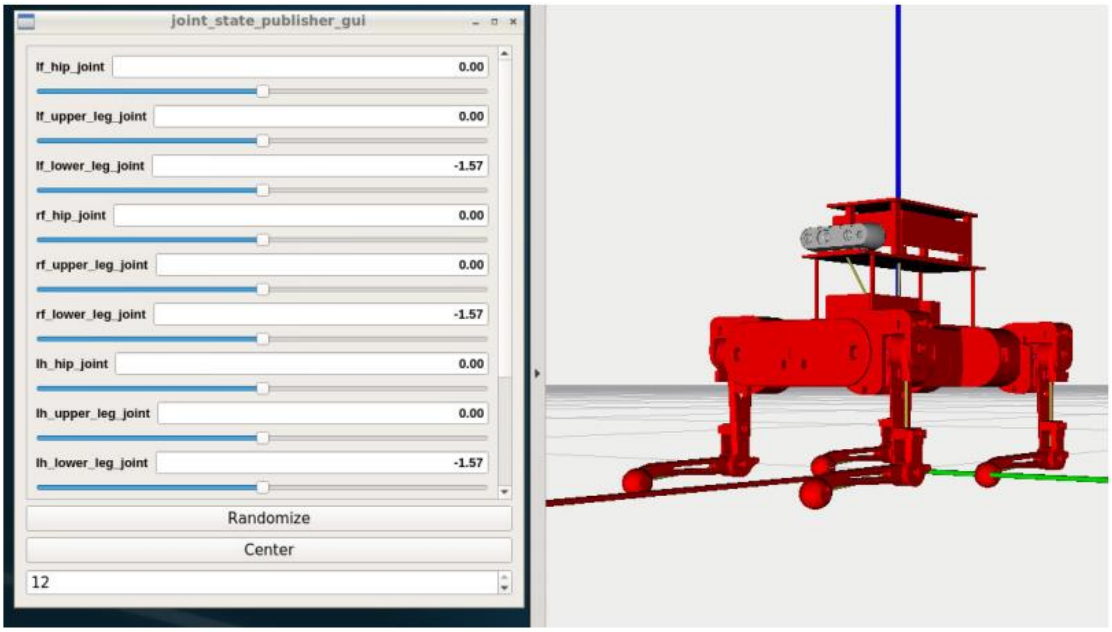


Figure 8

3.2.3 IMU

The MPU-9250 is the inertial measurement unit (IMU) that contains an accelerometer and a gyroscope. It supports various functions, such as the Low Power Quaternion and Gyroscope calibration used in this project. It also offers the data like angular velocity and raw linear acceleration.

3.2.4 ROS Serial

ROS serial is a protocol that packages ROS serialized messages and transmits them over Bluetooth, TCP or serial connection [10]. It handles the communication between ESP32 and the SBC, and a ROS node can be opened to publish or subscribe to topics.

A microcontroller is needed for defining the NodeHandle interface and implementing the ArduinoHardware if different communication channels are to be implemented. The NodeHandle interface can also be extended and support other communication

channels.

3.3 CHAMP

3.3.1 Hardware Interface

Esp32_hw_controller package is adopted and controllers such as sensor_msgs/JointState, sensor_msgs/JointTrajectory and sensor_msgs/Imu are used to communicate and control the respecitive hardware components. Besides, ROS serial is utilized in order to build the communication between ESP32 and the ROS network, via topics and nodes that are being published or subscribed to.

3.3.2 URDF Model

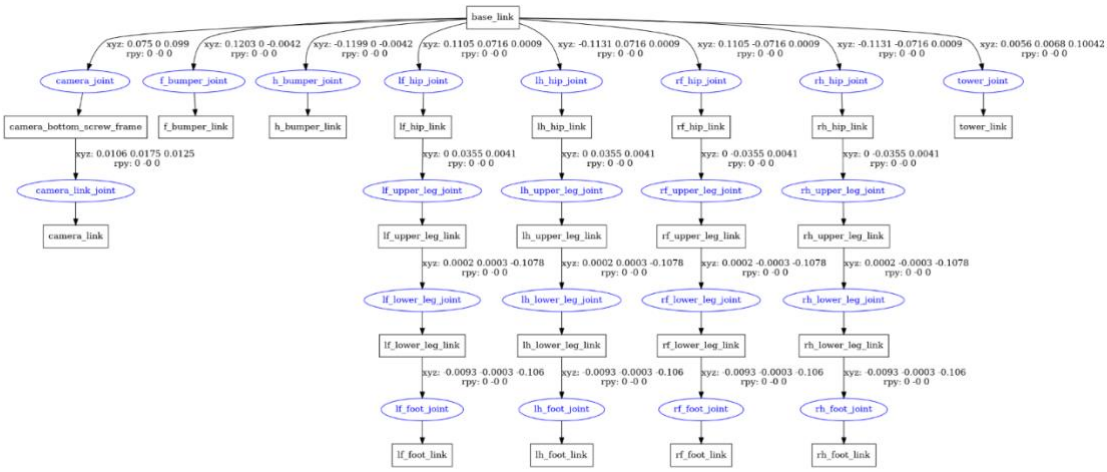


Figure 9

The URDF model is designed using the old STL files and 3D models. According to the physical outlook of the robot dog, the 3D models are grouped and stacked, and their joints are set to reflect the rotation axis and limits.

The origin of the robot is referred as the base_link in the Figure with all of iIts roll, pitch and yaw information equal to zero. To avoid the inverse kinematics engine from

moving the joints too fierecely, the joint rotation limits list the physical capabilities of that particular joint, and the limits will stop the IK engine from adjusting the joint if configured.

Joints	Lower Limit	Upper Limit
Hip Joints	-50 degree	50 degree
Upper Leg Joints	-60 degree	60 degree
Lower Leg Joints	-120 degree	-60 degree

3.3.3 Self-balancing algorithm

When a robot has a level body, it is most likely to be able to achieve stability and also stabilize the sensor installed on it. Hence, the self-balancing algorithm should find the required joint angle to achieve a level body.

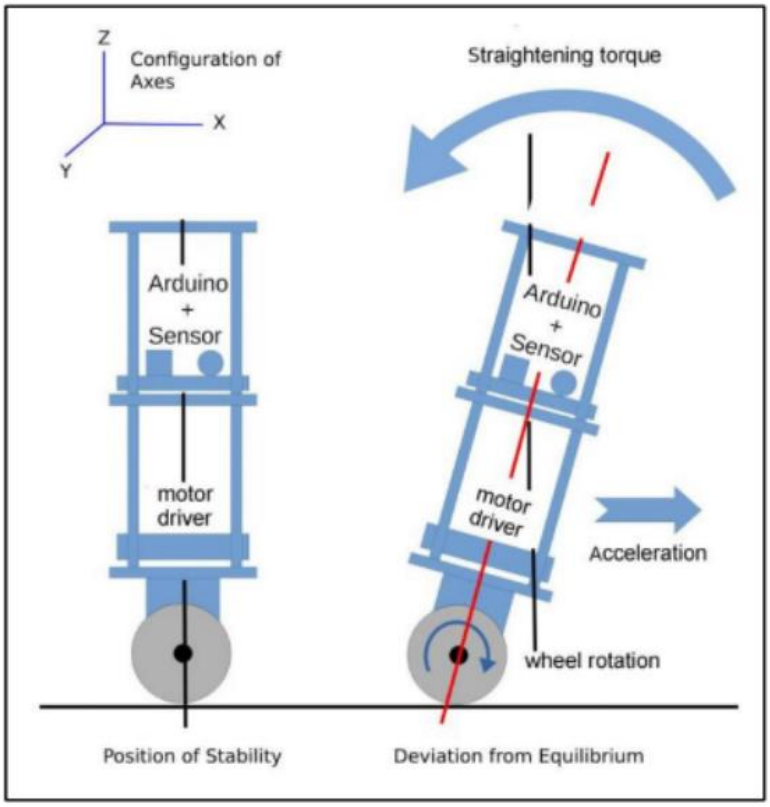


Figure 10 [11]

The mathematical expression of the PID (Proportion, Integral and Derivative) control is as follows:

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt}$$

,

Figure 11 [12]

Where $e(t) = v_desired(t) - v_actual(t)$, denoting the error between the desired and actual values in the function of time t.

Tuning P: faster response until the system starts to oscillate

Tuning D: damping the oscillation

Tuning I: removing the steady errors

3.4 Development Platform

Several virtual environments were created using Docker, which includes an environment for ROS1 (which installs CHAMP and other ROS1 packages), an environment for ROS2 (which installs compute vision and auto navigation packages), an environment for ROS1-to-ROS2 bridge for connecting the two, and an environment to host the webserver for the React web-based application.

There are several advantages of using Docker in our project. For instance, Docker enables a single set-up procedure that accommodates different development environments, such as having different hardware or operating systems. This is crucial for our team as our teammates are all using different operating systems, including

Windows, MAC and Linux. On the other hand, efficiency is improved as several containers can be opened at once using commands like docker-compose, and Docker images can be prebuilt and reused easily, which greatly reduces time and effort for development and deployment.

3.5 Computer Vision

3.5.1 3D Scene Reconstruction

In this project, we aim to achieve 3D scene reconstruction without using any prebuilt map data. Nvblox, a library that enables 3D reconstruction building from a realsense camera, is used in this project [13]. It calculates the Euclidean Signed Distance Field (ESDF) and Truncated Signed Distance Function (TSDF), which is useful for collision checking for robotic path-planning as well as representing the 3D environment. Nvblox inputs the following three things:

1. Pose from the infrared stereo camera and its embedded IMU data, which can be estimated by detecting the visual landmarks in the RGB image and depth image pair
2. RGB image with moving objects being isolated (after image segmentation)
3. Depth image from the infrared camera.

With all of these three pieces of information, a 2D image can be deprojected into a coloured point cloud. The sensor data will be integrated into a TSDF, which allows a 2D costmap to be generated (through slicing) and describes the occupancy status in the scene.

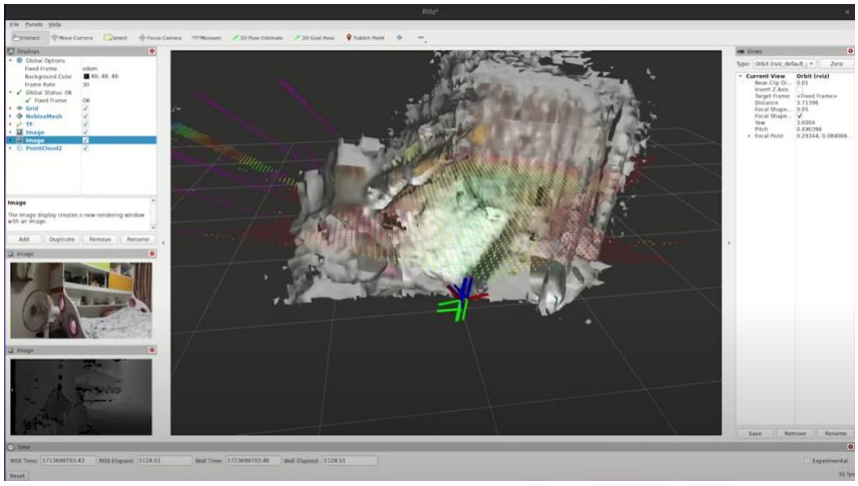


Figure 12

3.5.2 Autonomous Navigation

In our project, the Nav2 stack is used to enable autonomous navigation. It enables robots to navigate complex environments and provides the essential elements for building an autonomous system such as perception, planning, control, localization and visualization etc. [14].

3.5.3 Dynamic Object Following

The dynamic object following algorithm used in this project is as follows:

Step 1: Image segmentation – identifying the human object

Image segmentation partitions images into various segments or objects [15], such as identifying the background and the non-background objects. To identify and detect the "person" objects, PeopleSegNet ShuffleSeg model is used. As shown in Figure 13, the person object is identified and marked as red, while the background is in yellowish-green colour.

Figure 13

Step 2: Approximate the 2D coordinate of the human object

As mentioned in the previous step, the background and the human objects are identified and labelled separately. Thus, the connected components can be found with

ease. The largest connected component is to be picked and the “2D coordinate of the human object” is set to be the centroid of the largest connected component. For the depth information of the 2D coordinate, it can be found using the depth image.

Step 3: Deproject 2D coordinate to 3D world coordinate

With the 2D coordinate and the depth information, we can deproject the 2D point into the 3D space. For the intrinsic properties, they can be obtained by subscribing to the camera_info topic; for the extrinsic properties, they can be obtained from the pose information of the visual SLAM package. As the formula shown in Figure 14 demonstrates, with the 2D coordinate, intrinsic and extrinsic properties, the 3D world coordinate can be found intuitively.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Image Coordinates | Intrinsic properties (Optical Centre, scaling) | Extrinsic properties (Camera Rotation and translation) | 3D World Coordinates

Figure 14 [16]

Step 4: Update the goal pose

The 3D world coordinate will then be transformed into a pose, which is constantly updating by subscribing to the /goal_update topic.

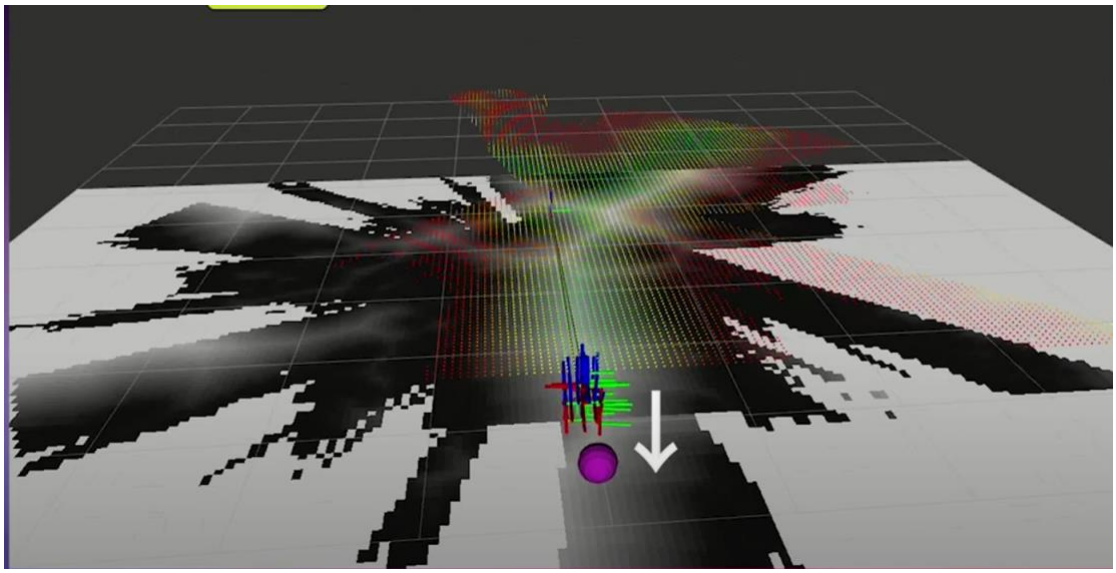


Figure 15

3.5.4 Frontier Exploration

The robot dog can explore the frontier and map the field with 3D scene reconstruction. In this project, explore_lite package is used. By subscribing to nav_msgs/OccupancyGrid and map_msgs/OccupancyGridUpdate, the costmap of interest is monitored. A breadth-first search will be implemented on the costmap, taking into account the occupancy status of each cell as well. The frontiers will then be found and explored.

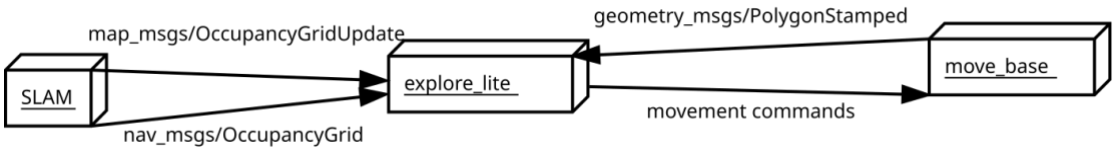


Figure 16.

3.5.5 Object Detection

In contrast to image segmentation which every pixel will be labelled, object detection only displays a bounding box if it detects the object (as shown in Figure 17), which also means that it requires way less computational power than image segmentation.

This makes it more appealing in scenarios when a large number of objects have to be identified. This can be applied in cases such as the rescue team would like to identify any specific hazardous objects on the site, as there may be a lot of them in an image.

On the other hand, some scholars also propose a convenient way of object detection in a video, which is the semi-automatic annotation, in which also a small amount of frame has to be manually annotated [17].

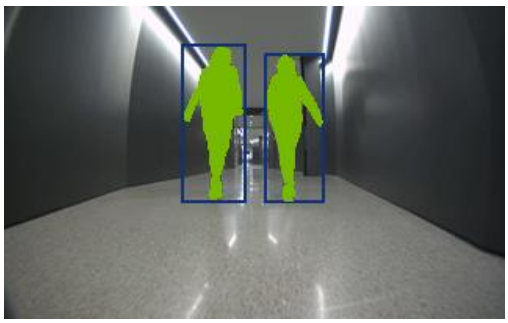


Figure 17 [18]

3.6 3D printing and models

The original 3D model of the robot dog is provided by the HKU CS Makerlab. To be more compatible with our hardware, some amendments have been made to the 3D model design.

In this project, we utilize softwares such as SketchUp and Blender to edit the 3D model, Cura for the slicing, and the 3D printer in the CS lab and the InnoWing for printing the components.

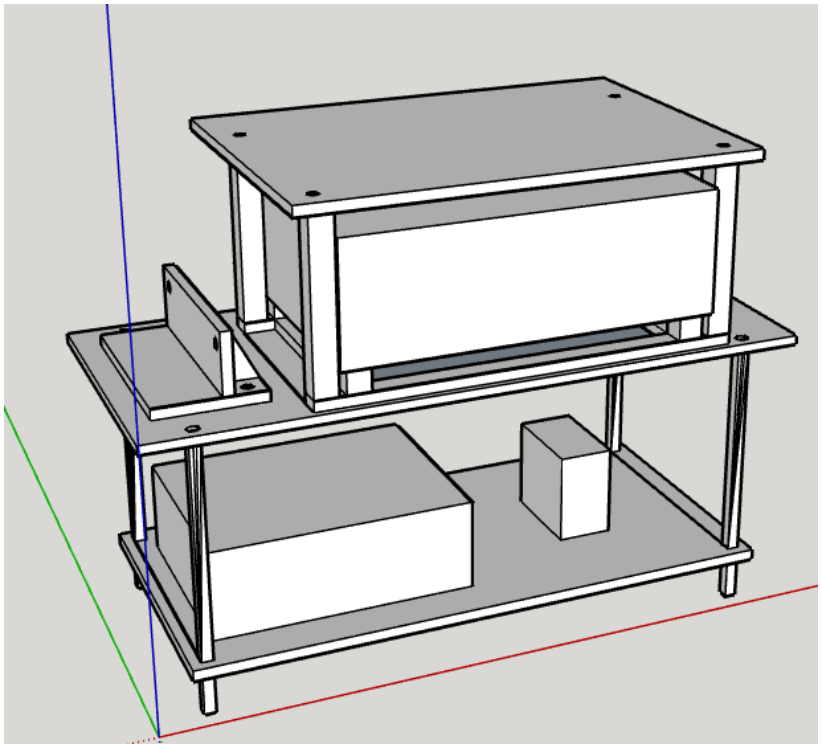


Figure 18: 3D model of the rack putting on the robot dog

Figure 19: the joints of the robot's shoulder, printed in InnoWing

4 Web-based application

To show the data from the robot dog and outputs from computer vision analysis, and allow users to interact and use the robot dog with ease, an easy-to-use web-based application is developed as the user interface.

The main objective of developing the web app was to cater to the needs of general users or non-technical users. While there exists a vast number of professional software or packages such as Rviz, a 3D robot visualizer for the ROS framework [19], or Blender, a software that performs 3D modelling and animation [20], it is important to improve user experience and the ease of use for general users. By developing an all-in-one web-based application, users are not required to install any additional software or packages on their devices or run multiple software at the same time just to see the live streaming video and the 3D simulation simultaneously.

All essential functions are integrated into this web-based application, for example, viewing the live streaming video captured by the stereo camera, looking at the real-time 3D simulation of the robot dog, incorporating a user authentication system to protect security and data privacy, and sending users alerts and notifications if the robot detects some target objects or unknown objects. To be more user-friendly and accommodate users using different devices, intuitive UI design and responsive web design are implemented.

In the following sub-sections, the web-based application's structure, key features, technologies employed, and UI/UX design will be discussed in detail.

4.1 Structure

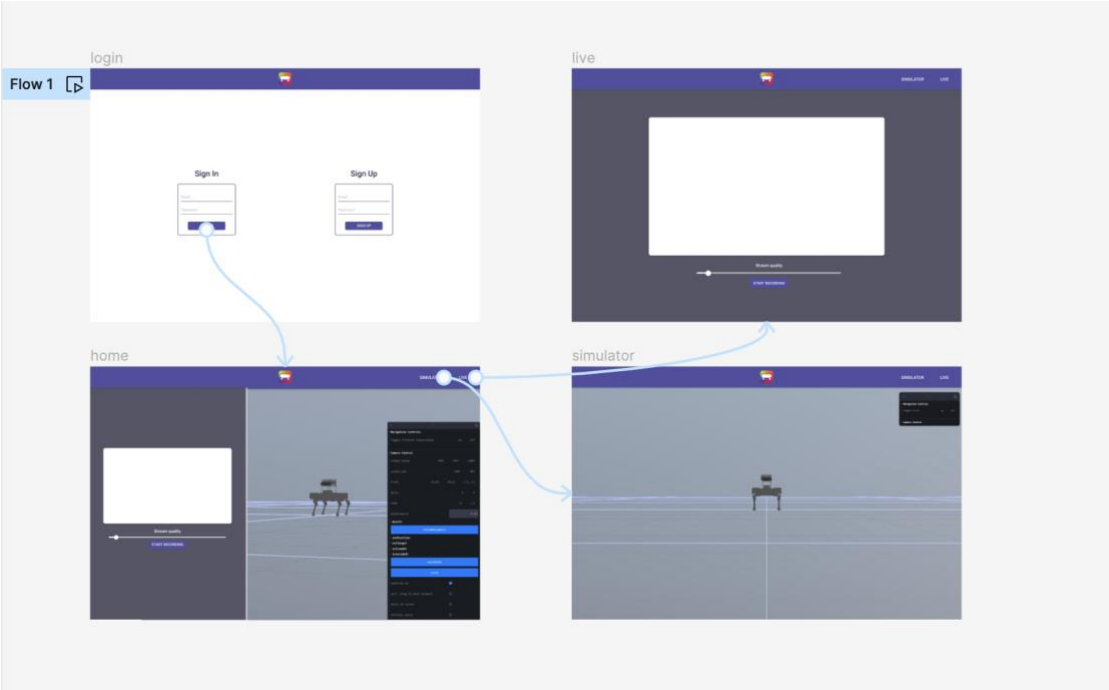


Figure 20

The web-based application is designed with a simple but clear structure which consists of four main web pages: the login page, the home page, the simulator page and the live page, as shown in Figure 20.

- The login page is the entry point of the application, where all users are required to sign in before accessing the other contents of the web-based application.
- After authentication, users will be redirected to the home page, which is a page for accessing and navigating all the features at the same time.
- The simulator page and the live page provide users with the real-time 3D simulation of the robot dog, and the live streaming video captured respectively. Opposed to the smaller frames on the home page, these two pages provide a full-screen frame for the simulation or the live video, which provides a more immersive experience for users to explore the captured image and motion of the

dog.

4.2 Features

After careful consideration of the potential application of our robot dog, such as participating in rescue work after natural disasters or accidents, we have implemented the following four features in our web-based application:

4.2.1 Live Streaming

In cases such as when there is a natural disaster and the rescue team deploys the robot dog in areas that are inaccessible to humans, it is crucial for the team to receive real-time visual information of the areas captured by the robot dog. Hence, in our web-based application, we allow users to view the live streaming video captured by the stereo camera of the robot. On the other hand, users can also adjust the stream quality of the video according to their network stability. In some use cases, users may also need to keep the video recording for documentation or further analyses, hence a button is added to the web-based application which enables users to capture the recording and save it to their local storage.

4.2.2 URDF Web Viewer

Users may also be interested in seeing the motion of the robot dog as a whole or observing the movement of some of its specific joints, especially when the robot dog is physically distant from the human users. Thus, we added a URDF web viewer to our web-based application, providing users with a 3D visualization of the robot dog's structure and movements, and the 3D structure will also be dynamically updated using the subscribed joint angle data from ROS. Users can view the 3D simulation in 360

degrees.

In addition to the simulation, a control panel is also added for adjusting the view angle on the screen using Leva (seeing section 4.3.4). Users can adjust their viewing angles by rotation or set the view position with a specific X, Y, Z position. This enables them to observe the movement of the robot dog's joints from different perspectives and have a more interactive and customized viewing experience.

4.2.3 User Authentication

Considering that the robot dog may be put in some private or restricted areas and capture images of confidential objects, it is vital to have a user authentication system to identify and control the user access rights for the live video and the 3D simulation. In our web-based application, a sign-up and sign-in system is implemented using Firebase Authentication (see section 4.3.5). Currently, two sign-in methods are supported, which are email/password and sign-in with Google accounts, providing users with a higher level of flexibility and convenience in the process.

The login page has been set to be the entry point such that users will first see it when visiting the website, and only after sign-in can they access the other features of the web-based application to ensure data security and privacy.

4.2.4 Notification System:

In the scenario where the robot dog is being deployed for a rescue task or on patrol, it is ideal if it not only identifies objects but also alerts human users about unknown or suspicious objects. Hence, in our web-based application, notifications will pop up in

the browser when new objects are detected.

4.3 Technologies Used

4.3.1 ReactJS

JavaScript and ReactJS are used in the development of the web-based application in our project. React is a front-end JavaScript library for based on components [21]. The reasons for choosing React as the framework for our web-based application are as follows:

- Component-based architecture

As described on the React official website, a React component represents a piece of the user interface with its own logic and appearance and each component is reusable and self-contained [21]. For example, we have two separate web pages dedicated to the live streaming video and the 3D simulation, as well as the home page that displays both of them simultaneously. Creating the components “live” and “simulation” and using the components on different web pages, it eliminates the need to replicate the code and greatly improves development efficiency.

- Virtual DOM

React offers a faster speed as it has a virtual DOM (VDOM). In many other web development frameworks that directly interact with the browser DOM, the entire DOM tree is manipulated on every page event. However, ReactJS has a virtual DOM (VDOM) which is stored in memory. Once a React element is updated, the changes are reflected and updated in VDOM first. After comparing the VDOM and the browser DOM, the newly updated objects could be identified and the browser DOM will then update only the new changes, instead of re-rendering the entire DOM [22].

4.3.1.1 Router in ReactJS

In React, a route is responsible for connecting a React component to be rendered and a URL path [23]. As demonstrated above in the structure section (see section 4.1), there are multiple pages in our web-based application. To link specific URL paths with the corresponding React components, routing is required. In our web-based application, we use the BrowserRouter, which enables client-side routing in a React app [23]. We successfully set the login page as the initial web page users see after entering the URL by setting the root URL ("/") route to the Login component, without lots of coding.

4.3.2 ROS Web Tools (roslibjs)

As the main function of our web-based application is to display the data from the robot dog, it is crucial that the web-based application is able to connect to ROS. roslibjs is a library which is for interacting with ROS from the browser [24]. Utilizing WebSockets as the middleware, robslibjs can connect with rosbridge, subscribe to ROS nodes and get data from them.

4.3.3 Three.js

Three.js is a 3D library that gets 3D content on a webpage and uses the WebGL API as its basis to draw 3D [25]. To load our 3D models in the format of .urdf, the URDF loader is used in this project [26].

4.3.4 Leva

Leva is a graphical user interface (GUI) that has a control panel-like look [27]. It is

movable on the web page and can be expanded or closed. By using a few folders and buttonGroups, we added a control panel that allows the users to adjust the parameters and angles for the viewing angle of the 3D simulation.

4.3.5 Firebase

Firebase is an app development platform by Google, which provides a lot of different products such as user authentication, database, extension etc., with fully managed backend infrastructure [28]. In this project, we leverage the Firebase Authentication for our user authentication system.

The advantages of using Firebase Authentication include eliminating the need and time to develop and manage backend servers or databases on our own. Besides, the user interface of Firebase Authentication is well designed and one can use it to manage the user accounts very easily, such as resetting the users' password, deleting or disabling their accounts etc. Additionally, Firebase Authentication provides a wide range of different sign-in providers or methods, such as email/password login, sign-in with Google/Yahoo/Facebook/Apple account etc., as shown in Figure 22. Web developers can freely add or remove login methods to cater for their target users.

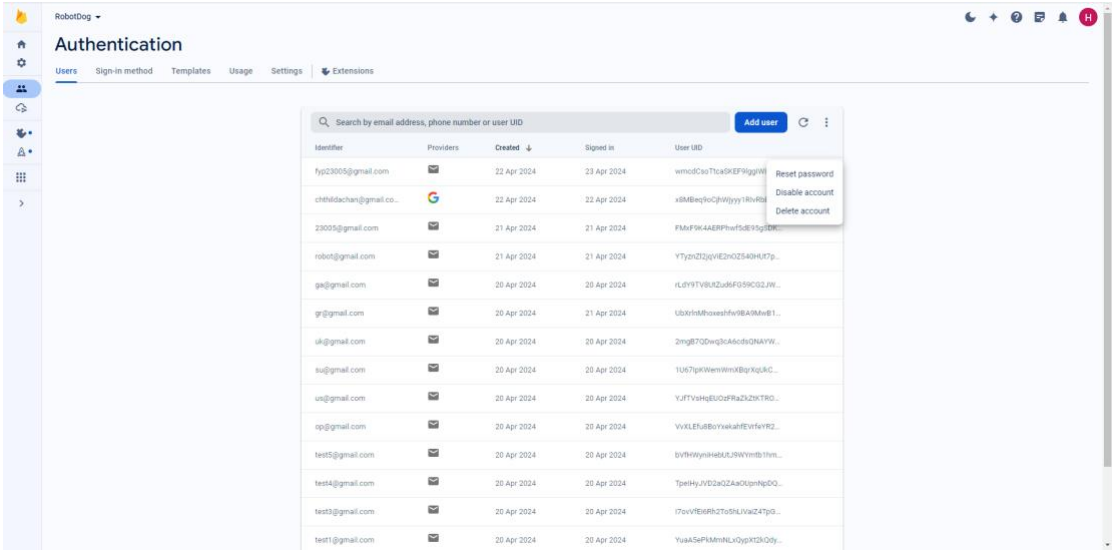


Figure 21

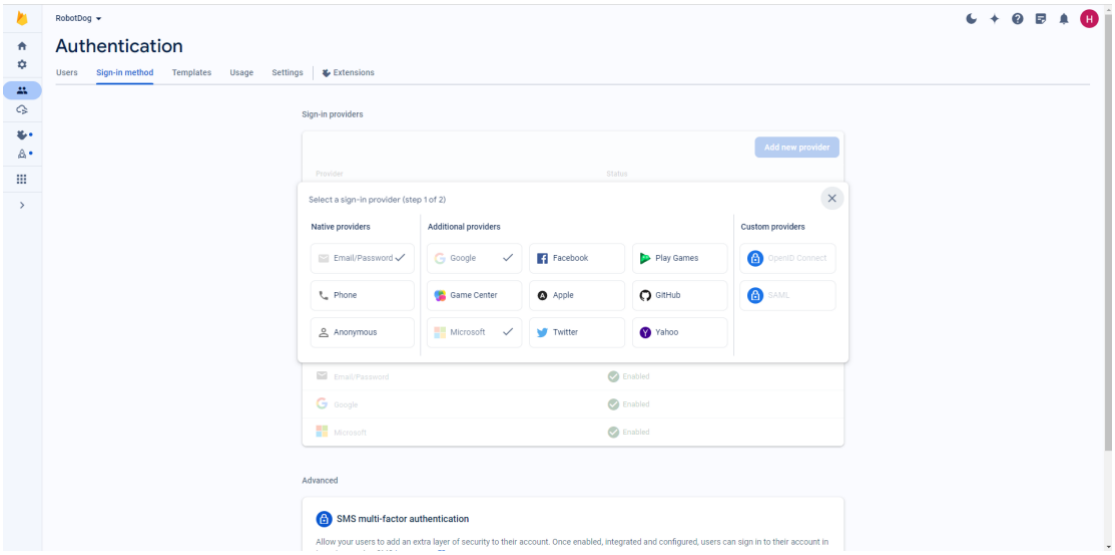


Figure 22

In our web-based application, we integrate Firebase Authentication in our project by using the web modular API provided by Firebase, in which we directly insert the code into the .jsx files of our signIn and signUp components.

For future work, one can even store user data in the databases in Firebase, such as Cloud Firestore. By saving the registered email or identifier in a collection, developers can save various user information and display them for the users,

providing a more personalized and customized experience to them.

4.4 UI/UX Design

4.4.1 User-Friendly Design

For convenient and intuitive navigation, several user-friendly elements are included in the UI design. For example, a top navigation bar is included on every page (excluding the login page), allowing users to quickly navigate and switch to different pages. Besides, on the Home page, a dragging function is implemented so that users can resize the layout of the live streaming video and the 3D simulation according to their preferences. To enhance user experience in observing the 3D simulation, a control panel that allows users to adjust the viewing point or viewing angle is also added.

To maintain a cohesive and appealing UI design, we utiltize the Material UI (MUI), a React component library that contains a variety of prebuilt UI components. Examples include using the createTheme function in MUI to set the theme colour and text colour in light and dark modes. It helps unify the colour scheme and provides visually appealing and professional-looking UI.

4.4.2 Responsive Design

Figure 23

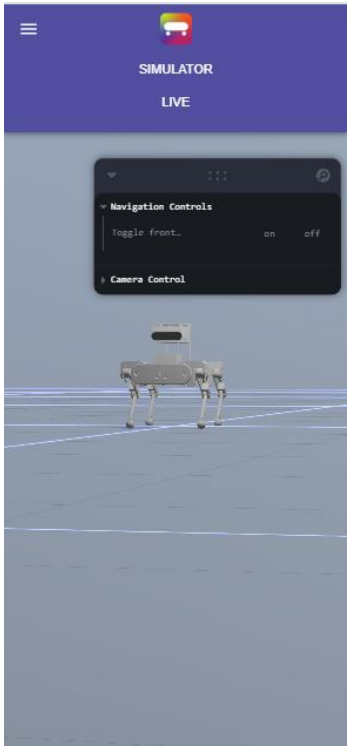


Figure 24

The web-based application may be viewed and visited by users using different devices. Take a rescue team as an example, team members may want to view the live video of the robot dog on their desktop computers in the office or with their mobile phone or tablet while on-site. To cater for the needs of more users and more devices, responsive web design is developed.

Without responsive web design, some functions may not be easily clickable on mobile devices, such as the width of the long navigation bar may be limited to the screen width, causing the two buttons to be compressed or be too tiny for clicking. To tackle this issue, we adopted a responsive web design. The navigation bar will be changed to a side menu and the size of the component (such as simulator and live video) will be adjusted according to screen width so that users can click the buttons and view the video or simulation with ease regardless of the device they are using.

5 Experiments and Results

5.1 Hardware

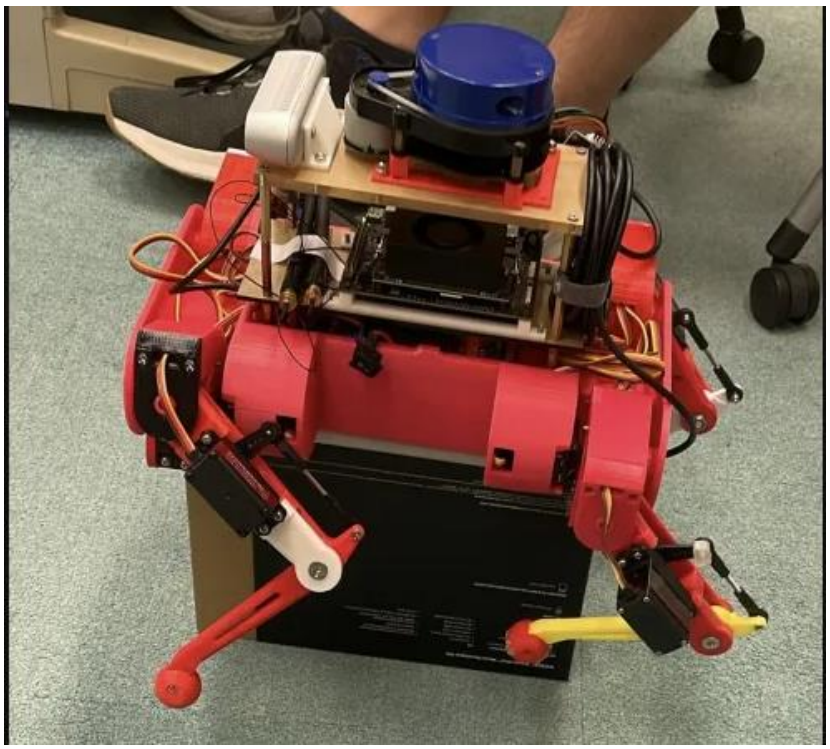


Figure 25

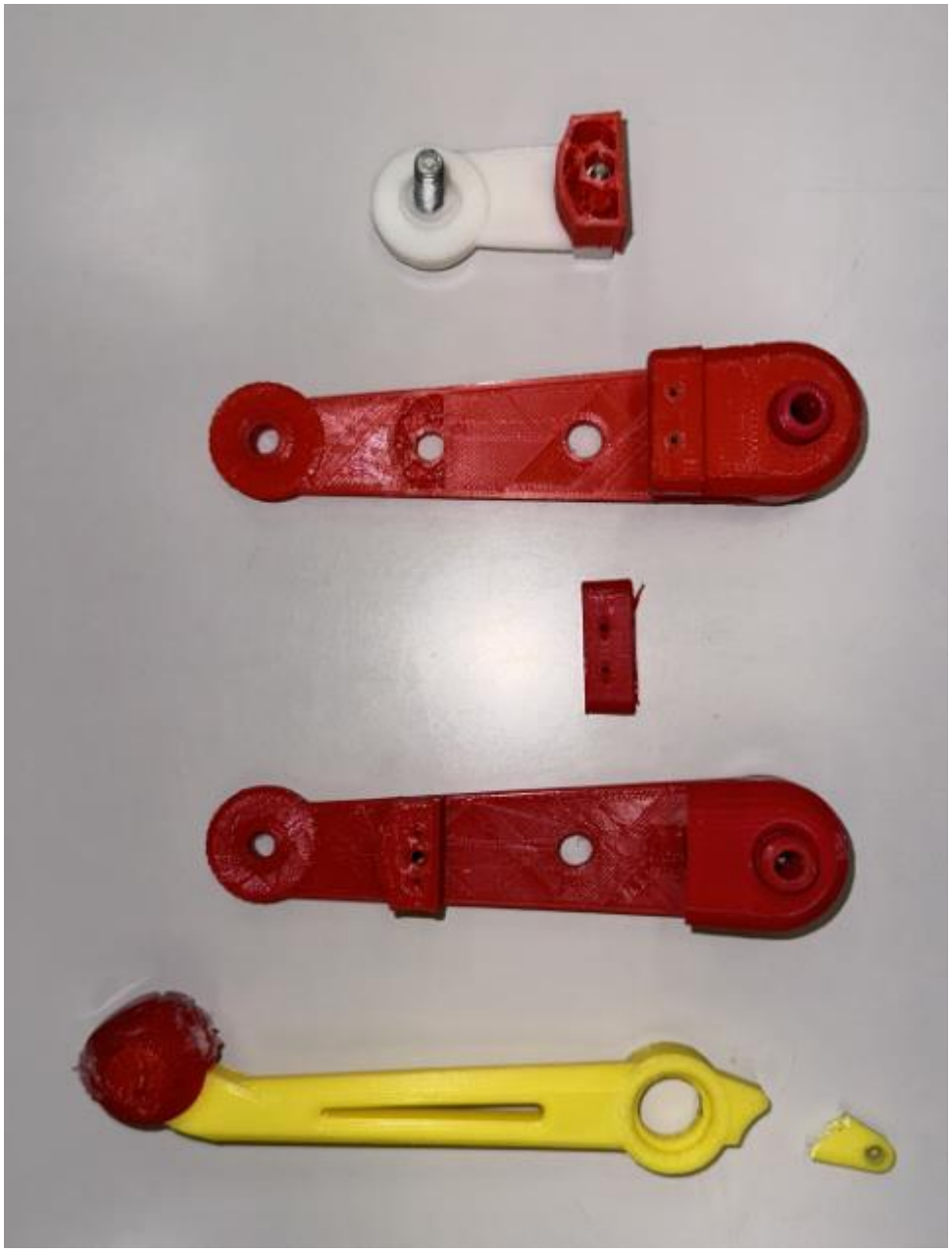


Figure 26

As the robot dog is 3D printed using PLA, the strength of it is way lower than the metal robot in the market. Wishing to have more details on the distance of a certain target from the robot dog, our team planned to use a Planar LiDAR at first.

However, the weight of the LiDAR, stereo camera, Jetson Nano, power bank and other electronic components are proved to be way top heavy for the plastic-made

robot dog. The set up shown in Figure 25 was equipped with the LiDAR, unfortunately the plastic component of the robot's leg broke immediately after a small walk as shown in Figure 26. After this experiment, we chose not to equip the LiDAR on the robot dog again, as the material and the structure of the dog probably cannot handle the additional weight of the LiDAR.

5.2 Self-balancing algorithm

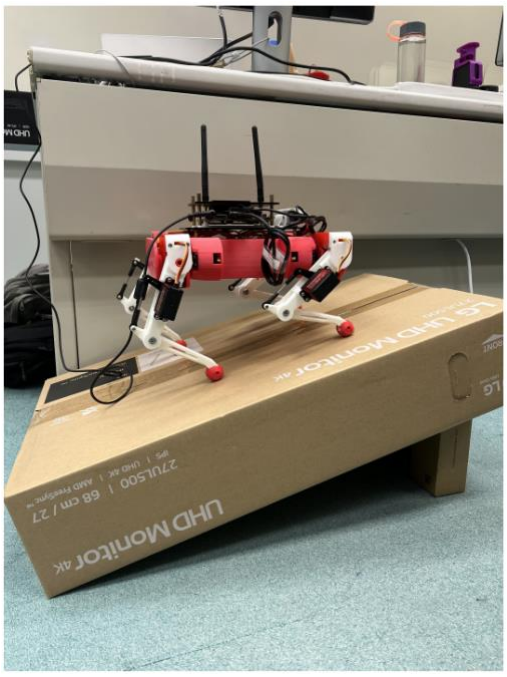
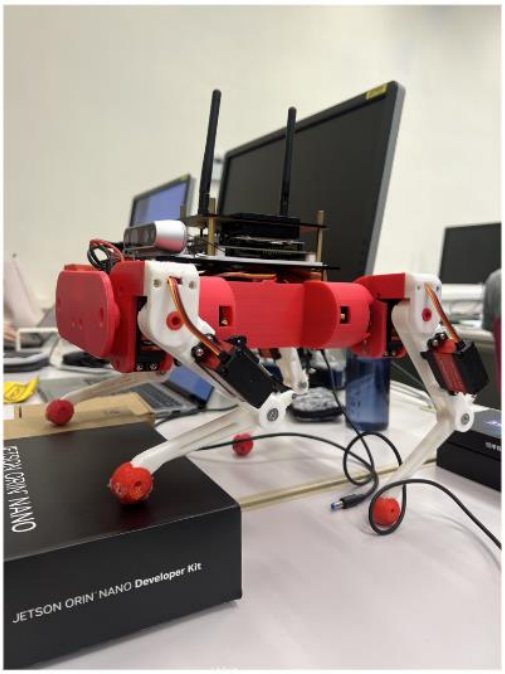


Figure 27 Figure 28

In Figure 27, where the I value of PID is equal to 0, the body of the robot dog is not very level. A possible reason contributing to this is the existence of the steady error. In Figure 28, where the I value of PID is a negative number, the body is much more level compared with the first one.



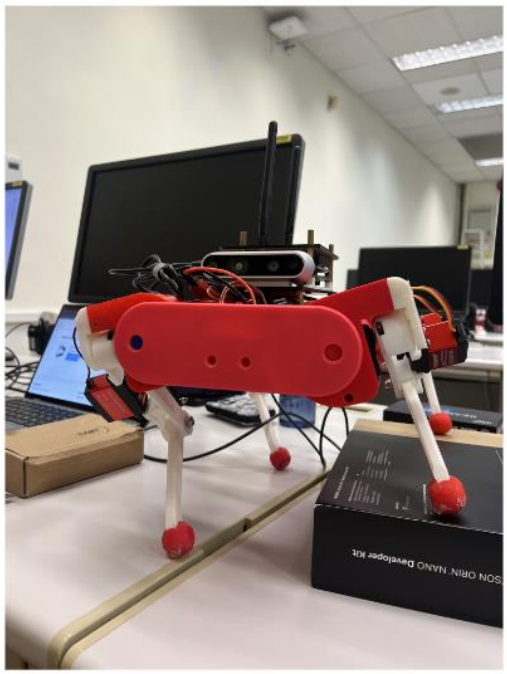


Figure 29 Figure 30

The above two images demonstrate how the discontinuous terrain affect the outcome of the self-balancing algorithm. In the first figure, where the dog was put on a constinous terrain, the self-balancing algorithm performs well and the body of the dog is level. However, in the second figure, where the dog's left legs were put on a higher position and its right legs were put on a lower position. As shown in the fugure, the body of the robot is tilteded and the two sides are not on the same height.

This can be explained by the absence of feedback indicatin the position of the robot's foot. The inverse kinematics engine is not sure if the robot is on a rough terrain or a box, hence the pose shown in the second figure seems less successful.

5.3 Computer Vision

While object following, the software will display a purple ball indicating the predicted position of the object/human. However, as shown in this Figure, the purple ball and the actual position is not consistent (in most of the cases the purple ball was indicating the previous position of the object). The reasons of the wrong predictition may be the followings:

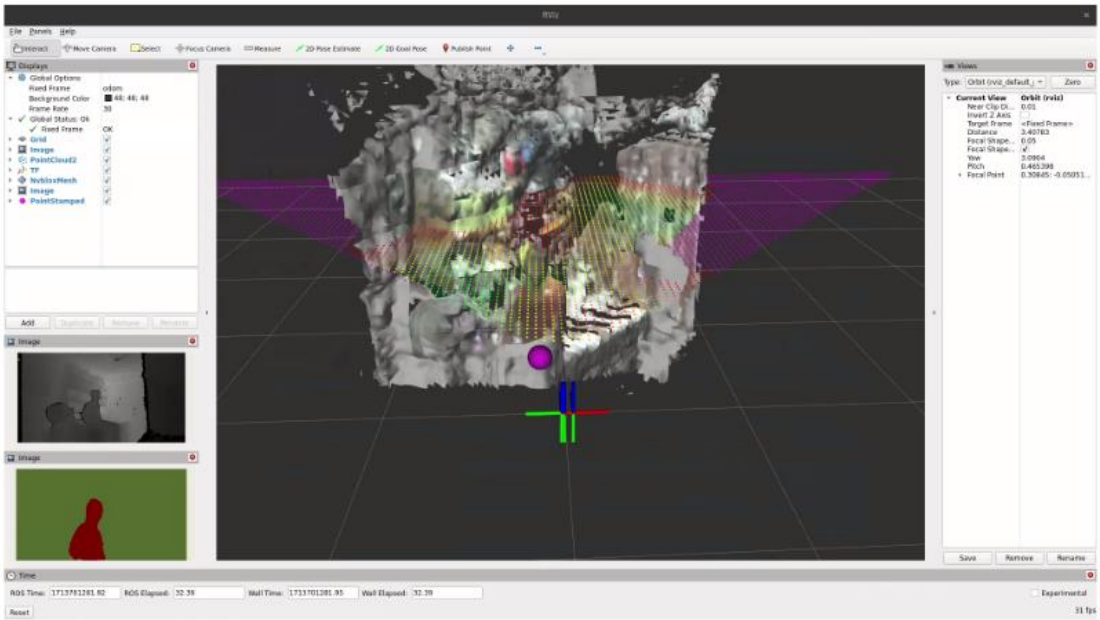


Figure 31

- Synchronization issue between the RGB image and the depth image

The two images may not be taken at the same time, as the two images are belongs to different ROS topics and having different camera settings. Additionally, image segmentation have to be performed on top of the two images, which require longer

processing time, and hence intensified the already-have time synchronization problem.

- Hardware invariance

As the robot is a legged machine, it experiences significant vibration when walking. Under strong movement and the vibration, the IMU may be drifted and hence the data collected may be inaccurate, which further affecting the accuracy of the camera extrinsic parameter matrix and the outcome of deprojection.

5.4 Web-based application

In the first term, our team tried out developinf a mobile app with Python and Kotlin, which displayed the video captured by the ESP32-CAM.

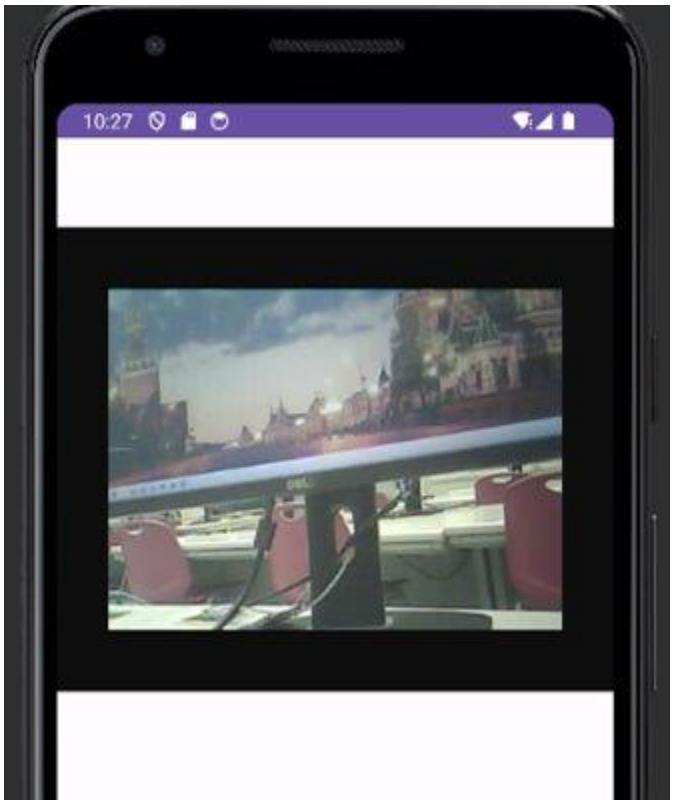


Figure 32

In that trial, only image, but not video can be obtained from ESP32-CAM.A Python

program was written to constantly get the image taken by the ESP32-CAM. Each image captured by the ESP32-CAM and the CameraWebServer has a similar path (http://***IPaddress***/capture?_cb=***UNIXtime***.jpeg). Hence, it is possible to start the live streaming by retrieving and displaying the image repeatedly.

After we adopted the stereo camera and planned not to include a joystick in the application, we believe that a web-based application would be better than a mobile app as it has a larger screen for user to view the captured video and the simulation. Hence, the current web-based application is created instead.

6 Future Work

Due to time limit and hardware limits, there are several things that are not implemented in our project. The following are some future work and improvements that could be done.

Odometry Sources

Currently, MPU-9250 is used and the DMP fins orientation well. The stereo camera, also provides RGB-D information. To further improve the project, sensor fusion which relies on multiple sensors like 3D LiDAR or magnetometer can be added.

Actuatiors – digital motos

In our project, we used the PWM servos as it is good for prevision control. However, stalling happens when the load is too much for the servos and making them overheat and burn out. In future projects, one can consider using DC motors with high torque

Foot sensors

It is believed that having foot sensors would be better as it can give feedback to CHAMP. However, the challenge is that it is very difficult to find suitable sensors and resign the 3D model due to the time limit of our project. Hence, we propose using other models in future work as the software is independent of the specific hardware.

Motion planning – deep learning

As mentioned above, in some applications there may be a large team of robot dogs. How the robot dogs move is crucial. As Quadrupeds have 12 degree of freefoms, motion planning is a high-dimensional problem. When there are rough terrain, foot configurations have to be carefully designed. For better motion, reinforcement and deep learning can be used.

Swarm robots

The idea of the quadruple robot dog can be applied to different scenario, in some cases it may be more powerful if it is a team of robot dogs instead of using just one or two of them. In cases that may require a map construction, a team of lighter robots can be used to detect the environemnet and a central machine can collect all the data. Some useful packages such as m-explore in ROS2 enable multi robot exploration of the environment and merging multiple maps from the robots [29].

Web applications

To cater for more mobile users, a native mobile application can be developed using React Native. Currently, there will be an alert pop up in the browser when some unknown object is detected. This can be adjusted according to the application and the hardware. For example, a rescue team, they may want to be alerted when there are

objects that look like humans or any objects that may be hazardous objects that should be avoided. This can be achieved by using a dataset that is trained for identifying those particular objects. If it is in a rescue team where rescue team members may not be able to hold a mobile phone, it is better to alert them of any new findings through audio if they are equipped with radio headphones etc. For higher precision and higher efficiency in rescue work, it will be better if the alert also notifies humans about the direction and the distance of the detected object.

7 Conclusion

To conclude, the objective of this project is to build an intelligent robot dog which is equipped with computer vision functions and can assist in rescue work. In this project, ROS, CHAMP, the IK algorithm are used in the building of the robot dog; several computer vision functions and a web-based application is also built in order to improve the usefulness and user interaction of the robot dog.

In future work or research, one can further perfect the computer vision functions of the project or develop a web-based or mobile application with more functions and is more user-friendly, for example, by considering what functions can be added to bring more benefits to the rescue team etc.

Bibliography

[1] J. Lee, “Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control : implementation on the MIT Cheetah robot,” *dspace.mit.edu*, 2013. http://hdl.handle.net/1721.1/85490

[2] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, “Inverse Kinematics Techniques in Computer Graphics: A Survey,” *Computer Graphics Forum*, vol. 37, no. 6, pp. 35–58, Nov. 2017, doi: https://doi.org/10.1111/cgf.13310.

[3] D. Tola and P. Corke, “Understanding URDF: A Dataset and Analysis,” *arXiv (Cornell University)*, Jan. 2023, doi: https://doi.org/10.48550/arxiv.2308.00514.

[4] “chvmp/champ_setup_assistant,” *GitHub*, Apr. 16, 2024. https://github.com/chvmp/champ_setup_assistant (accessed Apr. 26, 2024).

[5] S. Chitta *et al.*, “ros_control: A generic and simple control framework for ROS,” *The Journal of Open Source Software*, vol. 2, no. 20, p. 456, Dec. 2017, doi: https://doi.org/10.21105/joss.00456.

[6] Ravi Kant Jain, Baskar Joyti Saikia, Nitant Pilmo Rai, and Partha Pratim Ray, “Development of Web-based Application for Mobile Robot using IOT Platform,” Jul. 2020, doi: https://doi.org/10.1109/icccnt49239.2020.9225467.

[7] M. Rohrmeier, “Web based robot simulation using VRML,” *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, Nov. 2002, doi: https://doi.org/10.1109/wsc.2000.899135.

[8] “ros_control - ROS Wiki,” *Ros.org*, 2015. http://wiki.ros.org/ros_control

[9] “MPU-9250 Product Specification Revision 1.1 MPU-9250 Product Specification.” Available: https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

[10] “rosserial - ROS Wiki,” *wiki.ros.org*. http://wiki.ros.org/rosserial

[11] O. Halawa, A. Maged, A. Sabry, A. Gaber, A. Ramadan, and E. Tarik, “Mechatronics project: Two wheeled self-balancing robot,” May 2023, doi: https://doi.org/10.13140/RG.2.2.10832.48640.

[12] “PID Control,” *Autonomous Robots Lab*. https://www.autonomousrobotslab.com/pid-control.html

[13] “nvidia-isaac/nvblox,” *GitHub*, Apr. 26, 2024. https://github.com/nvidia-isaac/nvblox (accessed Apr. 26, 2024).

[14] “Nav2 — Navigation 2 1.0.0 documentation,” *navigation.ros.org*. https://navigation.ros.org/

[15] Shervin Minaee, Y. Boykov, Fatih Porikli, A. Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos, “Image Segmentation Using Deep Learning: A Survey,” *arXiv (Cornell University)*, Jan. 2020, doi: https://doi.org/10.48550/arxiv.2001.05566.

[16] “How to estimate intrinsic properties of a camera from data?,” *Stack Overflow*. https://stackoverflow.com/questions/63882698/how-to-estimate-intrinsic-properties-of-a-camera-from-data (accessed Apr. 26, 2024).

[17] Kutalmis Gokalp Ince, A. Koksal, Arda Fazla, and A. Aydin Alatan, “Semi-Automatic Annotation For Visual Object Tracking,” *arXiv (Cornell University)*, Oct. 2021, doi: https://doi.org/10.1109/iccvw54120.2021.00143.

[18] “NVIDIA-ISAAC-ROS/isaac_ros_object_detection,” *GitHub*, Apr. 23, 2024. https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_object_detection (accessed Apr. 26, 2024).

[19] “ros-visualization/rviz,” *GitHub*, Dec. 07, 2020. https://github.com/ros-visualization/rviz

[20] Blender Foundation, “Features — blender.org,” *blender.org*, 2019.

https://www.blender.org/features/

[21] “Quick Start,” *react.dev*. https://react.dev/learn

[22] S. Aggarwal, “Modern Web-Development using ReactJS,” *International Journal of Recent Research Aspects*, vol. 5, no. 1, pp. 133–137, 2018, Available: http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf

[23] R. Pm, “WEB APPLICATION FOR DIGITAL LIBRARY USING REACT/JAVA,” 1798. Accessed: Apr. 26, 2024. [Online]. Available: https://www.irjmets.com/uploadedfiles/paper/issue_3_march_2023/34510/final/fin_irjmets1679237956.pdf

[24] “roslibjs - ROS Wiki,” *wiki.ros.org*. http://wiki.ros.org/roslibjs (accessed Apr. 26, 2024).

[25] “three.js manual,” *threejs.org*. https://threejs.org/manual/#en/fundamentals

[26] G. Johnson, “gkjohnson/urdf-loaders,” *GitHub*, Apr. 26, 2024. https://github.com/gkjohnson/urdf-loaders (accessed Apr. 26, 2024).

[27] “pmndrs/leva,” *GitHub*, Apr. 25, 2024. https://github.com/pmndrs/leva?tab=readme-ov-file (accessed Apr. 26, 2024).

[28] Google, “Firebase,” *Firebase*, 2023. https://firebase.google.com/

[29] J. Hörner, “m-explore,” *GitHub*, Nov. 14, 2022. https://github.com/hrnr/m-explore