

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

Members:

Kwok Ka Yan (3035705336)

Leung Cham Chung Terry (3035704435)

Tang Gillian (3035705130)

Chan Hiu Tung (3035705180)

Supervisors:

Dr. T. W. Chim

Mr. David Lee

Abstract

Living in an era filled with catastrophe, it is key to innovate technologies that could minimize the loss of not only financial capital, but most importantly, human lives. To advance search and rescue post natural disasters, where entrances of human beings could be risky and unsafe, quadruped robots with equipped with tolerance to rough terrains of crocs could be introduced as assistance for searching. Therefore, the project emphasizes on researching computer vision and machine learning technologies that could be equipped to quadruped robots. 3D scene reconstruction, autonomous navigation, dynamic object following and frontier exploration are the key elements to install to a quadruped robot. The research also includes the hardware composition using off-to-shelf materials that could provide accurate sensor data and execute the prediction results of the high-level algorithms. Robot Operating System (ROS) is the main platform to be researched to facilitate software and hardware integration. This leads to a modular architecture of the project development. To eliminate the technical barrier of robotics, the project includes on researching the development of a user interface. The current state of the robot could be viewed with a well-designed user interface without understanding the mechanical and algorithmic logics behind. Leveraging the modularity of ROS, the researched software product of the project could possibly be integrated to other robot models, that the barrier of robot acquirement could be reduced to make a greater impact in aiding the search and rescue of natural disasters.

Acknowledgment

Thank you Dr. T. W. Chim and Mr David Lee for supervising the research of the project.

Gratitude is also directed to the second examiner, Dr. C. K. Chui.

Acronyms

2D 2-dimensional

3D 3-dimensional

CHAMP Cheetah-inspired Hopping and Maneuverable Platform

DMP Digital Motion Processor

DoF Degree-of-freedom

ESDF Euclidean Signed Distance Function

IK Inverse Kinematics

IMU Inertial measurement unit

PID Proportion-Integral-Derivative

PWM Pulse Width Modulation

RGB Red, Green and Blue

RGB-D Red, Green, Blue and Depth

ROS Robot Operating System

SDF Signed Distance Function

SLAM Simultaneous Localization and Mapping

TSDF Truncated Signed Distance Function

UI User interface

URDF Unified Robot Description Format

UX User experience

Table of Contents

1	Introduction	1
1.1	<i>Background</i>	1
1.2	<i>Motivation</i>	2
1.3	<i>Objectives and Deliverables</i>	4
1.4	<i>Literature Review</i>	4
1.4.1	Firmware	4
1.4.1.1	Cheetah-inspired Hopping and Maneuverable Platform (CHAMP)	4
1.4.1.2	Inverse Kinematics	6
1.4.2	Robot Operating System (ROS)	7
1.4.2.1	Unified Robot Description Format (URDF)	9
1.4.3	Simultaneous Localization and Mapping (SLAM)	10
1.4.4	Web Interface	11
1.4.5	Previous Projects	12
1.5	<i>List of contributions</i>	13
2	Methodologies	14
2.1	<i>Design</i>	14
2.1.1	Hardware	14
2.1.1.1	Electronics Components	14
2.1.1.2	3D Modeling	16
2.1.2	UI/UX Design	16
2.1.2.1	Design Elements	16
2.1.2.1.1	Typeface	17
2.1.2.1.2	Color	18
2.1.2.1.3	Logo	19
2.1.2.2	Prototype Design	20
2.2	<i>Firmware</i>	23
2.2.1	Programming Framework	23
2.2.2	PWM Digital Servos	23

2.2.3	Inertial Measurement Unit (IMU)	24
2.2.4	ROS Serial	24
2.3	<i>Cheetah-inspired Hopping and Maneuverable Platform (CHAMP)</i>	24
2.3.1	Hardware Interface	24
2.3.2	URDF Model	25
2.3.3	Self-balancing algorithm	26
2.4	<i>Computer Vision and Machine Learning</i>	27
2.4.1	3D Scene Reconstruction	27
2.4.2	Autonomous Navigation	29
2.4.3	Dynamic Object Following	30
2.4.4	Frontier Exploration	31
2.5	<i>Object Detection</i>	32
2.6	<i>Web Application</i>	32
2.6.1	Appearance	32
2.6.1.1	Mobile Responsiveness	33
2.6.2	Architecture	34
2.6.2.1	React Framework	34
2.6.2.2	ROS communication	35
2.6.3	Features	35
2.6.3.1	Live Camera Feed	35
2.6.3.2	Virtual Simulation	35
2.6.3.3	User Authentication System	36
2.6.3.4	Push Notifications	37
2.7	<i>Collaboration Tools</i>	37
2.7.1	Git	37
2.7.2	Docker Container	38
2.8	<i>Development and deployment</i>	38
3	Experiment and Result	39
3.1	<i>Hardware</i>	39
3.1.1	Robot Construction	39
3.1.2	3D Modeling	40
3.1.3	PWM Servos	42

3.1.4	IMU	43
3.1.5	Stereo camera	43
3.1.6	Single-board Computer	44
3.1.7	Self-balancing algorithm	44
3.2	<i>Computer Vision and Machine Learning</i>	46
3.2.1	Visual SLAM and Frontier Exploration	46
3.2.2	Dynamic Object following	47
3.3	<i>Web Interface</i>	48
3.3.1	ROS Connection	48
3.3.2	Live Camera Feed	48
3.3.3	Virtual Simulation	49
3.3.4	User Authentication System	50
3.3.5	User Interface	51
3.3.5.1	Mobile Responsiveness	52
3.4	<i>Collaboration</i>	53
3.4.1	Git	53
3.4.2	Docker	54
4	Future Work	55
4.1	<i>Actuators</i>	55
4.2	<i>Sensor Fusion</i>	55
4.3	<i>Swarm Robots</i>	56
4.4	<i>User Interface</i>	56
5	Conclusion	58

1 Introduction

This section outlines the background of the project with the motivations that indicates the interest in research, main objectives and targets to be delivered and a literature review of relevant knowledge and projects.

1.1 Background

In the time of calamities, it is not unusual to be notified for another tragedy occurring around our living circle. Disasters including earthquakes, landslides and tornadoes typically cause huge damages to the affected areas, creating a scene of dilapidation filled by piles of crocs (Guardian News and Media, 2008). These natural disasters could create huge tangible loss to the impacted regions, which the search and rescue process with a high priority of saving lives usually creates extra burden to those areas suffering from limited resources. With the risk of building collapsing anytime amidst rescue and the destroy of infrastructure to the emission of toxic substances, it is preferable to use technologies to create automated rescue processes to alleviate the damage to the rescue teams and victims, where robots shall be introduced to assist and bring up its efficiency (Guardian News and Media, 2024; Plant, 2014).

Robots are typically biologically inspired, focusing on some key human features to execute physical actions by the computation of kinematics and calibration, serving the purpose of helping humans to complete tasks that could be dangerous or risky to human beings (Garcia et. al, 2007). There are various types of robots innovated during the journey of robot research in the past century, including underwater robots, walking robots and humanoid robots, which the choice of features usually depends on the goal the robot is set to achieve (Garcia et. al, 2007). With the maturity of robotics research in the past decades, the application of robots has become more extensive to serve more operations of humans, making people's daily lives more convenient (Garcia et. al, 2007).

Despite robots usually being inspired by human biological features, there are various kinds widely adopted by the research area, where quadruped robot is a prevalent kind with a long research history, tracing back to 1900s (Biswal & Mohanty, 2021). Quadruped robots, built with flexible 4 legs each having 3 DoFs, totaling 12 DoFs, along with customizable set of sensors for detecting the surroundings, could compute thorough analysis and execute useful actions by collecting environmental data from sensors (Garcia et. al, 2007; Biswal & Mohanty, 2021). Leveraging their flexibility in locomotion, quadruped robots are well utilized in rough surfaces with slopes or stairs, making omnidirectional movements to adapt to various types of environments, where wheeled robots may struggle with matching its stability (Garcia et. al, 2007). Quadruped robots having been extensively researched to enjoy mature technologies, are usually used to explore difficult environments without stable wired connection, having widely open spaces with rough terrains and slopes, for instance, natural rural areas or disaster regions (Biswal & Mohanty, 2021).

1.2 Motivation

Robots are generally becoming more prevalent in the recent decade, having a wider range of applications in day-to-day lives. Having a visual element to it, being able to explore the theories and technologies behind controlling a robot's movements, and the endless high-level technologies the interactive to the real-life environment the robot can compute, robotics is a mesmerizing topic to research on.

On top of that, having seen past incidents like the Sichuan and Fukushima earthquake, where the whole region have been heavily destroyed, with the addition of emission of fatal substances from the nuclear power plant, to the recent Hualien earthquake with over a thousand of aftershocks spanning over the month, creating continuous casualties and financial losses, it is crucial to introduce some automated solutions for search and rescue to reduce the human involvement in the process for safety (Otani et. al, 2012; Yoshida, et. al, 2014; Wong & Chung, 2024).

Quadruped robots, having 4 individual legs to be controlled, are known for its stability and flexibility in locomotion, which opens to more possibilities in the technologies it could apply and real-life application. It enables opportunities of learning mechanics, 3D modelling, machine learning, computer vision, web development and other valuable topics of the computer science field. Nonetheless, the 4 separate legs significantly increase its difficulty in modeling and calibration, having more components to cater and more variables to diminish the accuracy of positions compared to the software simulations.

Despite the rigidness and variety in movements they may offer, the state-of-the-art models of quadruped robots, for example, the MIT Cheetah, could be costly to acquire (Chu, 2019). Therefore, the option of building a self-made robot using easily accessible materials becomes a fascinating option, which enables the research on 3D modeling. Having known the availability of the HKU MakerLab quadruped robot model for upgrading via 3D printing, creating a customized robot with choices of sensors and compartments could be a cost-effective option in the resource-limited circumstances.

The disaster areas are usually damaged all in a sudden, leading to the previous record of environmental data may no longer apply that both humans and robots have no existing maps to follow. Having a quadruped dog with agile movements, it is resilient to the unknown terrains and excellent for exploring areas of without prior data of the actual environment. Therefore, a 3D reconstruction of the surroundings could be useful for the search and rescue team to follow to speed up their actions. Moreover, the discovery of living people may be delayed due to limited manpower, not maximizing the 'golden hours', first 72 hours of rescue for the highest survival rate (Montgomery, 2024). Thus, utilizing the robot dog to help detect any possible living human beings could be helpful for more targeted and efficient rescue. Acting as an assistant of the rescuer and not adding extra burden to him, allowing the robot to exploit its ability of human detection and environmental knowledge could enable the feature of it following its master (rescuer).

1.3 Objectives and Deliverables

The overall objective of the project is to create a machine learning software package with user interfacing, executed on a quadruped robot prototype to enhance the efficiency of search and rescue post natural disasters.

There are 3 key deliverables of the project, dividing the project into 3 main parts. The first is to design a quadruped robot with off-the-shelf components and inverse-kinematics capabilities. Balancing algorithm for stability should also be applied. Followed by it is to implement intelligent behaviors for a quadruped robot, which focuses on detecting an environment without prior knowledge, achieve frontier exploration and track specified objects. Finally, it is to create an interactive dashboard for high-level remote controls and real-time visualization of robot state, where user experienced should also be considered.

The completed project should demonstrated functional robot with quality, demonstrating the capabilities of the machine learning algorithms and have them reflected on the interactive dashboard.

1.4 Literature Review

In an effort to have a better insight of quadruped robot development and its current technological development, some technologies, theories and relevant projects have been reviewed.

1.4.1 Firmware

1.4.1.1 Cheetah-inspired Hopping and Maneuverable Platform (CHAMP)

CHAMP is an open-source framework for developing the quadruped robots and the corresponding control algorithm, based on a research project for highly dynamic locomotion of quadruped robot, model MIT Cheetah, from the Massachusetts Institute of Technology (Lee, 2013; chvmp, 2023). This framework utilizes stabilization tactics, ground

reaction to forces and modulation of gait patterns to enhance the locomotion of quadruped robots.

Quadruped robots, having 4 legs and 3 joints each, enjoys a total of 12 DoFs. However, this could be an obstacle for correctly controlling the movement of robot by commanding with high-level algorithm, as each joint is connected to its individual servo and motor. By imposing inverse-kinematics methods, the joint angles could be calculated to find the angles for desired movements. Nonetheless, the calibration for smooth locomotion requires extra algorithmic efforts (Otten, 2003).

Therefore, CHAMP framework could be an improvement algorithm. It first divides the legs into 2 phases, namely stance and swing, which are indicating the phases of the leg on the ground and off in the air, due to the difference in dynamics (Lee, 2013). To specify the changes in phases, 2 events are encoded to trigger any changes, LF and TD, which indicate the leg lifting off the ground (stance to swing) and the leg touching the ground (swing to stance) respectively (Lee, 2013).

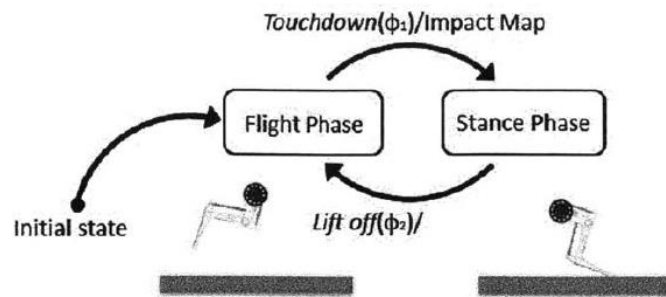


Figure 1.1: Finite state machine for leg dynamics (Lee, 2013)

Another significant element in CHAMP is the virtual compliance mechanism, which caters balance during the legs' stance phase to prevent from slipping (Lee, 2013). The mechanism is inspired by cheetahs' leg movement that it could be denote as the change of leg's equilibrium, thus the mechanism's goal is to maintain its leg equilibrium state against external forces or rough terrains during the phase of stance. The mechanism assigns a

reference leg (i.e. the front-right leg) to keep track of the velocity, gait pattern and the TD event change, ensuring that the state operations are as expected (Lee, 2013). Then, the gait patterns, converted to a series of swing and stance states, are assigned to the individual legs with synchronization. The TD event tracking is crucial due to the possibility of having external forces, that TD event trigger is the only way of moving the legs, and otherwise kept stationary to minimize the external impact (Lee, 2013).

After that, there will be leg trajectory compiled according to the phases the legs are in. A Bezier curve will be defined for legs in the swing phase aiming to give them sufficient grip to the ground and retraction rate, while attempting to save energy (Lee, 2013). A sinusoidal wave is utilized for better control the compliance force during the stance phase trajectory, maintaining the ground reaction force with the sinusoid's amplitude thus ensuring the correctness of the virtual compliance (Lee, 2013).

Eventually, having the leg trajectory information from every leg, the torque commands to the corresponding motors will be found by adopting the equations of motions and the actual geometry of the robot (Lee, 2013). The joints will be rotating to the calculated angles according to the feedback from sensors and forward kinematics (Lee, 2013).

1.4.1.2 Inverse Kinematics

Despite the CHAMP framework being responsible for obtaining the trajectories for the legs, there is a process required for translating the algorithmic commands into low-level motor controls. Consequently, an inverse kinematics algorithm could be found useful for achieving it, with the formal mathematical definition as the follows (Aristidou, et. al, 2018):

$$\theta = f^{-1}(s),$$

$s = (s_1, s_2, \dots, s_k)^T$ denotes the desired end effector positions and $\theta = (\theta_1, \theta_2, \dots, \theta_n)^T$ denotes the column vector for all DoFs (Aristidou, et. al, 2018). The optimal solution for θ is to achieve a smooth and stable locomotion (Aristidou, et. al, 2018).

One of the approaches would be Jacobian inverse methods, which defines a Jacobian matrix for representing the small changes in joint angles and its corresponding changes in end effector's orientations and positions (Aristidou, et. al, 2018). The information is used to decide the joint parameters for the ideal poses. Using the inverse of the Jacobian matrix, the end effector's orientation and positions could be derived, providing a local linear approximation (Aristidou, et. al, 2018). By iterating the algorithm, the joints movements leads to progressively moving closer towards the target destination (Aristidou, et. al, 2018). Nonetheless, when the matrix hits singular, there would be issues of inverting it and the approximation would gradually make impact towards the accuracy of calculations (Aristidou, et. al, 2018).

Another popular method would be the analytic solution. The θ is derived from the starting position, length of links and rotational constraints of the legs, forming a closed-form solution (Aristidou, et. al, 2018). Without the risk of experiencing matrix singularity, the accurate angles could be derived from trigonometric calculations, given that each leg has 3 DoFs (Aristidou, et. al, 2018).

1.4.2 Robot Operating System (ROS)

ROS is an operating system dedicated to robots, intended to provide the essential tools for seamless integration across different types of robots. It is an open-source library that is available for both commercial and non-commercial usage (Quigley et. al, 2009). Robotic application development could be done in a modular approach, annotating software modules as nodes, the smooth communication between nodes creates a useful software package (Quigley et. al, 2009). ROS acts a comprehensive middleware, which translates between user commands and the signals to hardware (Macenski et. al, 2022). It handles the firmware signals with hidden layers of API, that the client (developers) could simply use the interfaces provided with processed data for calculations (Macenski et. al, 2022). There are prebuild algorithms for commonly used for robotics, along with interfaces that translates firmware input to standardize format of data, the robotic software development could focus on creating useful tools to serve the application goal of the robot (Quigley et. al, 2009).

Having a modulated organization, the packages could also be easily applied to other application, which is especially suitable for using the software in different kinds of robots (Macenski et. al, 2022).

In a complex system with multiple nodes, message is a key form of communication. They are divided into different topics, which could be subscribed by other nodes to retrieve the processed information published to the relevant topics (Macenski et. al, 2022). Having a mature messaging system in ROS, it eases the complexity of interactions between modules.

In comparison to other frameworks and protocols commonly used for robotics, ROS is an all-in-one platform for catering the needs as a middleware, development platform and communication handler, with the virtue of modularity and reusability (Macenski et. al, 2022). An example of alternative framework would be Yet Another Robot Platform, which shares the benefit of modularity, yet mainly focuses on humanoid and legged robot specifically, that the software packages could not enjoy the seamless adaption to other robot types (Macenski et. al, 2022). Other instances include Open Robot Control Software and Lightweight Communications and Marshalling, emphasizing on message handling and real-time robotic control, lacking the holistic development experience handled on one single platform (Macenski et. al, 2022).

Following the locomotion framework and executing the calibrated joint angles, CHAMP package in ROS could be helpful for desirable end results. The package receives messages that contains commands of algorithmic results of the position, orientation and velocities. The joint angles will be calculated with inverse kinematics referencing to the current feet positions, to be sent to the robots physical joints. There is also the information for the odometry, which is essential for the estimation of the robot's current position and orientation compared to the beginning (chvmp, 2023). The information could be conducted based on various sensors installed in the robot, such as IMU, stereo camera and infrared sensors. To minimize the error during the inverse kinematics calculation process, a Kalman filter could be a useful algorithm for reducing the errors of uncertainty (chvmp, 2023).

The following is the list of useful parameters for gait configuration:

- Knee Orientation – The orientation for the knees, which knees could be bent leftwards, rightwards, inwards, and outwards.
- Max Linear Velocity X (meters/second) – Robot's maximum forward and reverse speed.
- Max Linear Velocity Y (meters/second) – Robot's maximum speed when moving sideways.
- Max Angular Velocity Z (radians/second) – Robot's maximum rotation speed.
- Stance Duration (seconds) – The designated time of each leg spending on the ground while walking.
- Leg Swing Height (meters) – Trajectory height during swing phase.
- Leg Stance Height (meters) – Trajectory depth during stance phase.
- Robot Walking Height (meters) - Distance from hip to the ground while walking.
- CoM X Translation (meters) – Translate reference point along x-axis, useful for compensating weight if the center of mass deviates from the center of robot (from front hip to rear hip).
- Odometry Scaler – Multiplier for the calculated velocities for dead reckoning, useful for compensating odometry errors on open-loop systems.

1.4.2.1 Unified Robot Description Format (URDF)

URDF is a standardized format for storing the hierarchical and dynamic information of physical robots. URDF models are used to state the hierarchy between the links and joints of the robot (Tola & Corke, 2024). The information of collision and inertial frames are stored. Between the links, there are joints used for defining the tree-like relationship between links, in addition to the DoF and rotations (Tola & Corke, 2024). The model is adopted in ROS when executing the CHAMP algorithms, defining the joint positions, limits and collision boxes for

the physical robots, assisting the generation of configurations and launch files (CHAMP, 2023).

There are specific requirements for defining a URDF model in CHAMP, which includes not allowing the rotational offsets of frames, restricting hip joint rotations in x-axis and lower leg joint rotation in y-axis, and keeping actuator meshes centered during rotations (CHAMP, 2023).

1.4.3 Simultaneous Localization and Mapping (SLAM)

SLAM is a widely researched algorithm in robotics, which represents the process of simultaneously estimating robot state and constructing map representing the explored surroundings, supporting multifarious advanced robotic tasks, such as path planning (Cadena, et. al, 2016). By localizing the estimation, the accumulated errors of sensor data (i.e. the IMU data) could be reduced by relacing dead-reckoning that would create noticeable drift shortly after launch (see figure 1.2) (Cadena, et. al, 2016). By detecting and identifying a landmark set, priori, SLAM could localize robot state and the processed map simultaneously (Cadena, et. al, 2016). Priors is independent from other types of sensor data, which could create more realistic more maps, not hindered by the accumulated errors (Cadena, et. al, 2016).

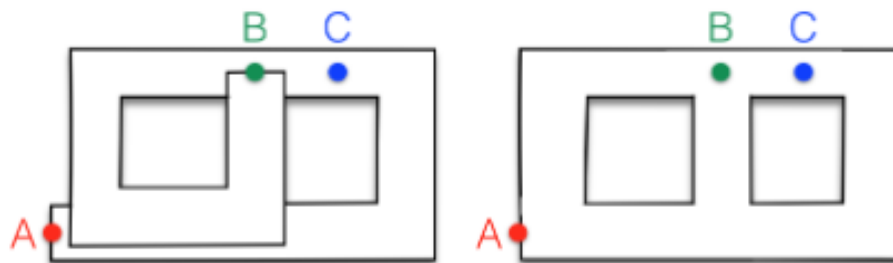


Figure 1.2: The comparison of maps constructed by odometry (left) and SLAM (right)

2 key components are demanded to properly execute SLAM, namely front-end and back-end. Depth data are passed to the front-end component, that the priori would be returned.

The back-end is responsible for extracting keyframes and estimating pose transformation in between (Garigipati, 2022).

Visual SLAM is one the most popular frameworks of SLAM. It adopts the algorithm by imagery data retrieved from sensors, which performance could not be guaranteed due to the varied quality of imagery sensors. Monocular visual SLAM is a case in point, receiving one source of image data. With the possibly inaccurate estimation of depth, limited information could be used for deriving priori. Nonetheless, a stereo visual SLAM could compare the synchronized image pairs to estimate disparities between matching key to conduct more accurate depth estimation. Having an extra layer of color information, RGB-D visual SLAM could derive an even more accurate map result.

1.4.4 Web Interface

Considering that not every user is equipped with technical skills to control the robot with command-line code or other coding-heavy measures, an interface easily accessible to users is necessary for maximizing the use cases of the robot. In spite of the locality and content preloading of mobile applications, as the first phase of development, launching a web application would be the most accessible to users across all platforms, with little restriction in implementation and maintenance (Holzer & Ondrus, 2012). After browsing a range of frameworks, for example, Vue.js and Angular.js, React.js is a suitable option to develop a web interface. Compared to Vue.js and Angular.js, React.js is an industry favor with the most starred and forked package on GitHub, while having the most node package managers developed around it, showing its dominance in web development (Saks, 2019). In React, components act as building blocks of the application as a while, allowing developers to create reusable components to be rendered multiple times by not only the current project, but also later projects with other requirements by parsing a couple of parameters for customization (Saks, 2019). The user interface could also be independent from the backend system, which could be done by parsing props from JavaScript, enhancing its flexibility and reusability for better legacy, aligning with the project objective of a creating high-level system applicable to multifarious types of robots (Saks, 2019). To create a coherent

appearance of the interface, Material UI, a UI component package for React.js developed by Google, could be an intuitive and efficient tool for creating React.js UI components (MUI Team, 2024).

Meanwhile, ROS is a favorable tool for web development. In ROS, the package `rosbridge` facilitates the communication between ROS and non-ROS programs, which creates web sockets for a list of data useful for computation or display of the outside world (ROS.org, 2024). Therefore, by subscribing to the ROS topics needed to be used on the web interface, messages are exported in readable format and data could be presented easily with on various user interface elements. On top of it, the URDF model could be retrieved from ROS messages in XML format, which could be interpreted by JavaScript code to make further user interface customizations (ROS.org, 2024).

1.4.5 Previous Projects

After the Fukushima earthquake, where the destruction of the nuclear power plant led to significant leakage of radiation, dangerous to human beings, there was a development of a rescue robot model, Quince, a wheeled robot designed to travel on rough terrain, aiming to capture the current view of the pos-disaster environment within the high radiation areas (Yoshida, et. al, 2014). However, not only does it have the limitation of walking around unexpected surfaces, but it does also not offer any high-level computational functions which lacks the speediness of offering analyses to prompt rescue (Yoshida, et. al, 2014).

For Computer Science students from the University of Hong Kong, there are several recent projects regarding quadruped robots that worth studying. One of which is the study of how a quadruped robot traverse on challenging terrains, which mainly focuses on the design of the hardware components of the robot to optimize the execution of simulated movements on the software side, and its integration of low-level algorithms to achieve best physical outcome (Chui, 2022). It uses C-based language for calibrating the motors by giving the desired amount of power to the joint elements (Chui, 2022). This could be useful for real-life application of complicated commands in the real-life environment, yet the complex analysis

of the environment is not designed. Another project aims to design a quadruped robot dog suitable for STEM education (Lee, 2022). It emphasizes on developing a self-balancing algorithm for the quadruped robot to stand and walk on sloped terrain (Lee, 2022). Nevertheless, it did not exploit the robot’s ability to travel around sloped areas to develop high-level system solutions for other applications. Another project with a similar topic, designing a quadruped robot for STEM education, focuses on using Arduino framework to assign commands to the robot dog’s physical limbs, hoping to improve its ability to walk on difficult terrains (Lau, 2023). Both projects develop their algorithms on Arduino framework. Unlike ROS, which offers tools for standardized signal processing, allowing the compiled software to be reapplied to other models of robots built with other sets of firmware, it is unfortunate that Arduino framework does not offer high-level GPU-accelerated machine learning features (ROS.org, 2024).

1.5 List of contributions

Task	Contributor(s)
Joint Calibration	James
Self-Balancing Algorithm	James
3D Modeling	James, Terry, Hilda, Gillian
3D Printing & Laser Cutting	James, Terry, Hilda, Gillian
Scene Reconstruction	Terry
Object Detection	Terry
Dynamic Object Following	Terry
UI/UX Design	Gillian
Web Interface Development	Hilda, Gillian

Table 1.1: List of contribution

2 Methodologies

2.1 Design

2.1.1 Hardware

The formation of the final design of the physical robot comprises a selected set of electronic components, 3D-printed materials, laser-cut acrylic boards and assembling parts (see figure 2.1).



Figure 2.1: final composition of the quadruped robot

2.1.1.1 Electronics Components

Despite the inheritance of the HKU CS Makerlab model of robot dog, changes have been made for the choice of electronic components due to the capacities and performances. The

following is the table of all the electronic components adopted in the final design of quadruped robot model (See Table 2.1):

Role	Model	Usage / Reason of Choice
Single-board Computer	Nvidia Orin Jetson Nano	To enjoy the built-in GPU for machine learning processes with better performance while having a light-weighted and compact design.
Microcontroller	ESP32	To control firmware including IMU, servo motors
Digital Servo	TD-8135MG	To serve the purpose of joint actuators
IMU, accelerometer and gyroscope	MPU-9250	To measure linear acceleration, angular velocity and orientation
Stereo and Infrared Camera	Intel Realsense D435I	To capture the color and depth information of the surroundings
Battery for microcontroller	7V Lithium-ion battery	To charge power for the microcontroller

Table 2.1: list of electronic components used in the robot

It is worth noting that to reduce the weight carried with the robot, the single-board computer is powered by cable via the USB port, usually an external power bank for better mobility.

During the process of construction of the robot model, there are multiple electronic components installed for performance testing, aiming to provide more accurate results to the analysis of joint movements. For instance, foot sensors, which are responsible for indicating the touch on and lift off the ground, are installed to provide another layer of information to calculating the inverse kinematics. However, the responsiveness may vary according to the reflection rate of different ground surfaces. Besides that, LiDAR was also installed for a 360-degree depth detection, aiding the accuracy on reconstructing scenes and avoiding obstacles. Nevertheless, the performance of the robot's movement declines attributed to the weight of the sensor.

2.1.1.2 3D Modeling

The design of the 3D robot dog inherits the design from the HKU CS Makerlab, having its outer 3D-printed and electronic compartments including the microcontroller, single board computer, actuators and sensors. The 3D-modeled body parts designs are stored in the form of STL files, leaving room for modifications of fine details. The components are 3D-printed to fit the installation of the original design, which comprises of an Orange Pi as the single-board computer, MG996R servos, which sizes and weights differ from the current set of firmware, plus the new addition of stereo cameras, thus requiring modifications of the 3D model.

To cater the heavier weights of the TD-8135MG servos, the model of the upper joints is modified to allow smoother movements supported by the more powerful servos. The modifications could be done by using the 3D modeling software, Blender, allowing edits of individual vertices.

2.1.2 UI/UX Design

For the user-facing web application, UI/UX design is the key of structuring the application well and present it to the users. User Interface (UI) is the visual representation of what users need to know about the software, which the design focuses on the appearance and provides easily understandable representations of complicated algorithmic processes (Bilousova, et. al, 2021). User experience design is the navigation flow to build the interactions between the users and the software, in consideration of the functionalities of the software (Bilousova, et. al, 2021). The aim of the UI/UX design is to leave the users satisfying first impressions by laying the key elements aligning with the branding style, being able to make interactions intuitively (Bilousova, et. al, 2021).

2.1.2.1 Design Elements

In order to offer an appealing branding recognition to the audience, a set of design elements is universally adopted in all visuals of the project.

2.1.2.1.1 Typeface

Firstly, the typeface choice is Jura and Roboto, which the title font is the former and the content font is the latter. Both being free for use and distribution Jura is under the license of SIL Open Font License and Roboto is under the license of Apache License 2.0, easing the future adaption to commercial project. Meanwhile, Jura and Roboto are both available on npm package `fontsource-variable`, easing import in React.js. Jura, a sans-serif font constructed geometrically, has uniformed clean lines and sharp angles, offering a futuristic and technical appeal, aligning with the branding of a robot dog. The design of the characters is formed uniquely, distinguishing it from other fonts, while providing a modern and simplistic look, along with multiple choice of font weights for design flexibility (see figure 2.2). Despite the simplicity of Jura, it could lack readability in larger pieces or smaller sizes of texts. Therefore, Roboto is chose to be the content font for any content details in the user interface. The sans-serif font offers clarity and simplicity, with its versatility to switch between font weights. The modern and technological appeal it gives aligns with the Jura and branding of a robotic project (see figure 2.3).

**A QUICK BROWN FOX
JUMPS OVER THE LAZY
DOG.**

a quick brown fox jumps over the lazy dog.

Figure 2.2: sample text of Jura (Google Fonts)

A QUICK BROWN FOX JUMPS OVER THE LAZY DOG

a quick brown fox jumps over the lazy dog

a quick brown fox jumps over the lazy dog

Figure 2.3: sample text of Roboto (Google Fonts)

2.1.2.1.2 Color

The color scheme is another determinant of the unity of branding style. In order to create a 4-point gradient as the main visual of the brand, 4 colors are chosen on a fixed brightness color wheel, namely #524FA0, #B62467, #F05A22, and #C9DA2A in hexadecimal code representation (see figure 2.4). A 4-point gradient dynamic aesthetics creates an appeal of diverse capabilities and transformation, which is one of the development objectives of the project. The 4 colors chosen spans widely on the color wheel, including relationships of complementary, analogous and contrast, resulting in vibrant but harmonious visuals to the audience.



Figure 2.4: Color selection for branding design in color wheel representation

2.1.2.1.3 Logo

To create a professional-looking brand, a logo could be essential for brand recognition. Utilizing the color scheme and font choices, the logo of the project is created. Attributed to the vibrant and eye-catching background, white is the choice of geometries and texts. The project name is kept simple on the logo. Taking the key physical features of the robot, stereo cameras as the eyes and 4 limbs as the legs, inferring an abstract representation of a dog, the project is named “Robot Dog”. In terms of geometrical elements, the front-view of the robot is abstracted to 3 rounded elements, representing the body and the 2 front limbs respectively (see figure 2.5). For a clearer first impression to the audience, the product name is included in the logo. However, due to the limited size, the title is eliminated on the website favicon and navigation bar logo (see figure 2.6). The logos and some relevant design materials are created via Adobe Illustrator, a vector graphic design software. This could ensure that the design materials will not become pixelated when enlarging in high-definition viewports.

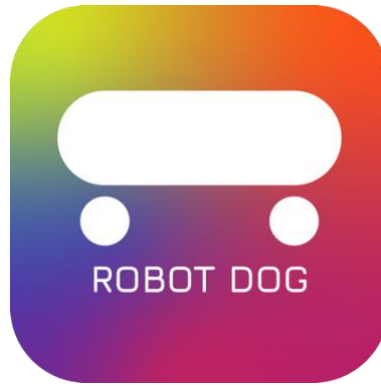


Figure 2.5: Logo of the project



Figure 2.6: Web interface logo

2.1.2.2 Prototype Design

As a user-facing product the interface prototype design inherits all design elements to align with its branding principles. To echo with the rounded design of the geometric elements of the logo, rounded elemental designs are preferred over the ones with sharp edges, enhancing its unity visually.

Initially, a mobile application was the choice of user interface due to the portability of mobile phones and the local features mobile applications could enjoy. Apart from that, there is an existing ROS mobile app (written in Android Studio) which provide some basic features of robot controlling, including joystick control and connection configuration (ROS-mobile,

2023). In an attempt to inherit the mobile app’s features, a prototype of mobile application was designed in the earlier stage using Figma (see figure 2.7).

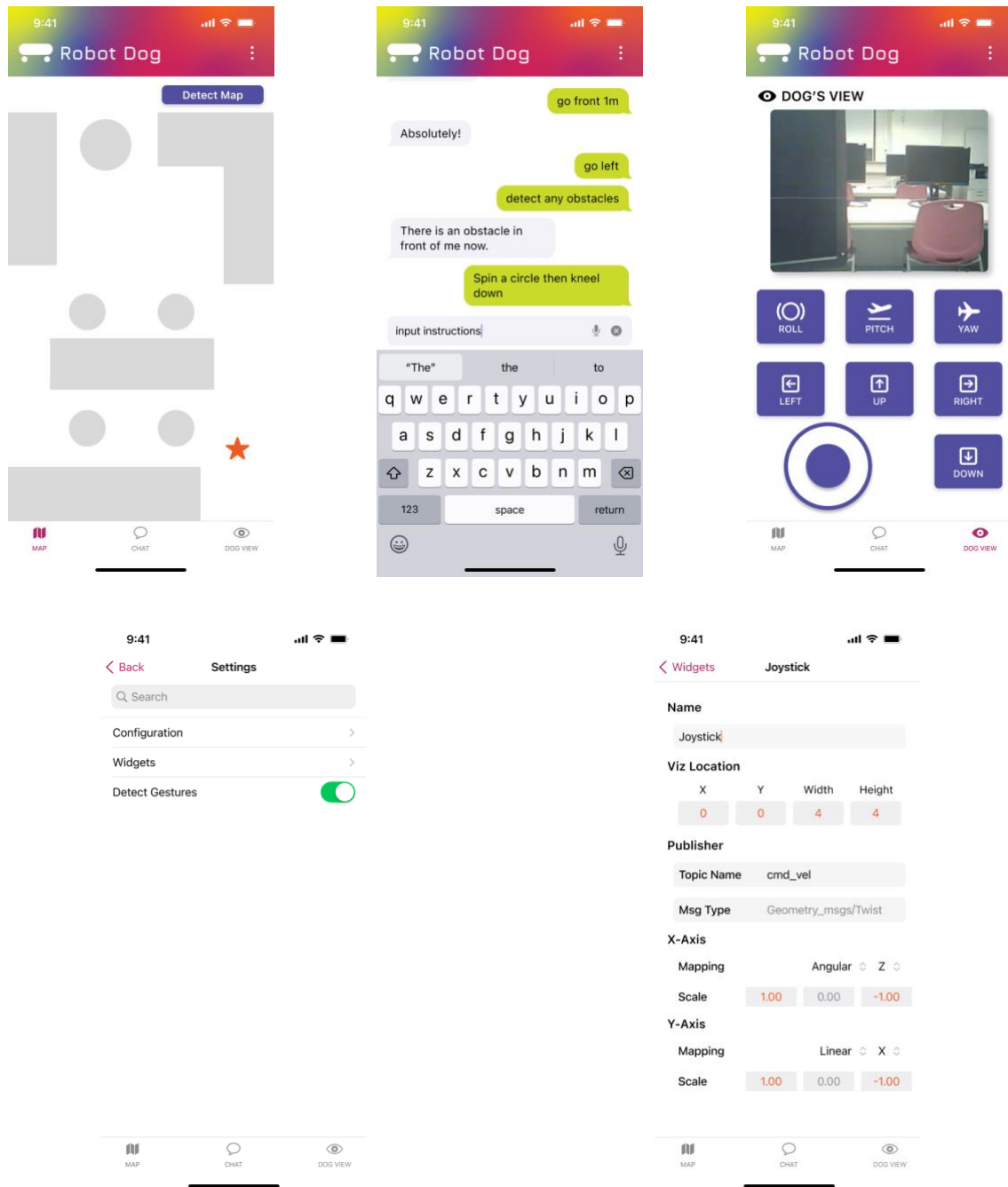


Figure 2.7: Initial prototype for user interface

As shown in Figure 2.7, features including joystick and natural language processing text-to-control were intended to be included in the user interface. However, after thorough

considerations, both features may not be a good fit of the project scope, which the former requires manual control of robot, not feasible during search rescue where every minute counts, and the latter serves more as an entertainment purpose, which could be replaced by simpler commands like buttons or even making the robot fully automated.

Having done literature review and thorough analysis of the system design, it is concluded that the user interface could focus on demonstrating the results of high-level machine learning algorithms to show the simulation of the robot's movements, the current view of the robot's stereo camera and notify users if any interesting objects are detected. In view of the development flexibility and offering a seamless virtual remodeling of the robot, a web interface was the final choice. Subsequently, a redesign of interface and navigation flow prototype was built with Figma (see figure 2.8).

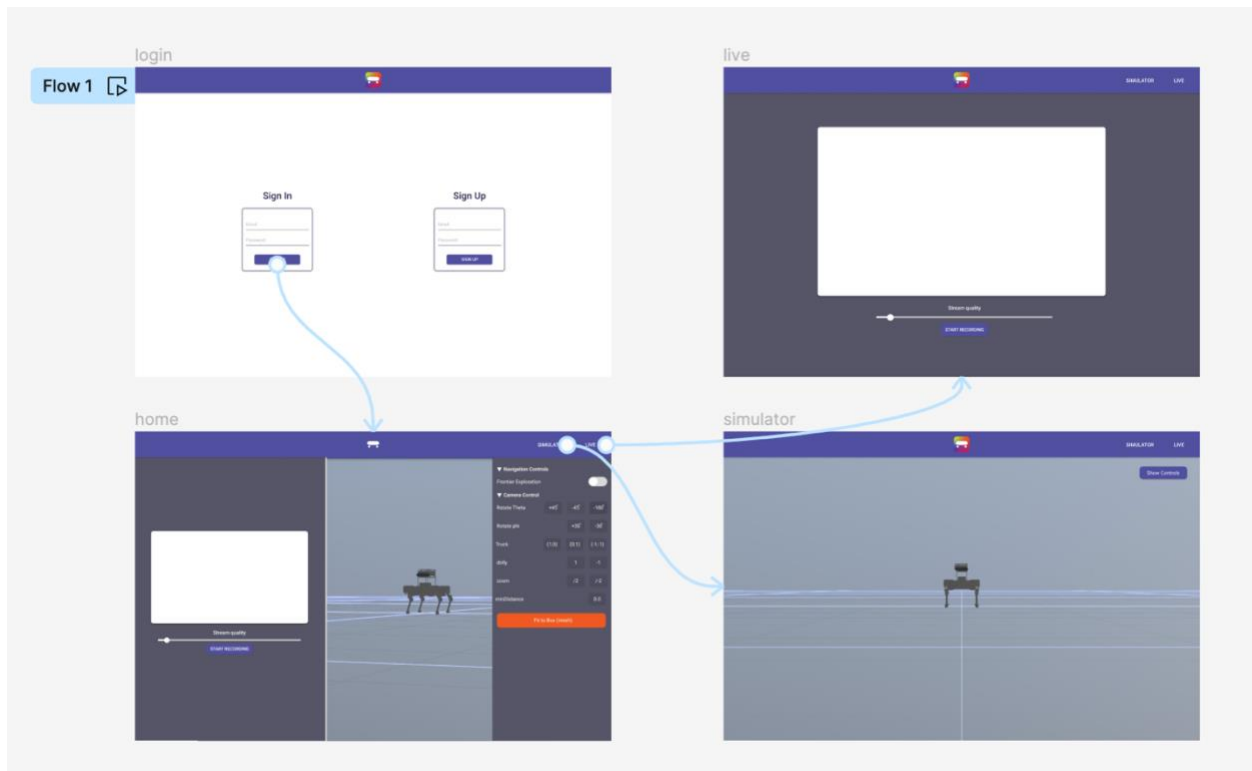


Figure 2.8: Web interface and flow prototype

To breakdown the prototype design shown in figure 11, the web interface comprises a login/sign up page, a home page as a web dashboard and pages for individual elements,

namely robot virtual simulator and live feed from the robot's stereo camera. The layout contains a top navigation bar and the main content section for easy traversal between pages. It is disabled on the login page to avoid unauthorized access to the pages. The main dashboard is divided into 2 sections in a horizontal layout, where the left consists of a live capture from the stereo camera of the dog, and the right contains a virtual simulation of the robot, along with an option for viewport control.

Considering intuitiveness of usage, some UX design tactics are included in the prototype design. A case in point is the easiness in navigation between pages, where a navigation bar is placed on top of the pages, allowing routes to other pages by one simple click. In the meantime, to simplify inputs and reduce errors, UI elements such as a slide bar for adjusting frame rate, a toggle to enable frontier exploration are contained and buttons for adjusting viewing angles of simulator.

2.2 Firmware

2.2.1 Programming Framework

Arduino framework alongside PlatformIO IDE offers interfaces of handling electronics, that several libraries could be found useful for catering the hardware model design. Adafruit PWM Servo Driver Library, MPU-9250 Digital Motion Processing (DMP) Arduino Library and Rosserial Arduino Library could be adopted to facilitate signals of the PWM driver chip installed on the ESP32 development kit, signals of IMU and DMP, and messages from and to ROS (Adafruit Industries, 2023; ROS Drivers, 2023; SparkFun Electronics, 2023).

2.2.2 PWM Digital Servos

By utilizing functions including `setPWMPFreq(freq)` and `setPWM(channel, on, off)` from the Adafruit PCA9685 PWM Servo Driver Library, desirable PWM signals could be translated and sent to the PWM digital servos. On top of the conversion between Champ returns and PWM signaling format, to reduce the impact of disturbance in real-life environment and physical

limitations of hardware, a series of calibration calculations needs to be done in the algorithm to optimize hardware performance.

2.2.3 Inertial Measurement Unit (IMU)

The IMU installed, MPU-9250 contains a digital motor processor, responsible for processing the accelerometer, gyroscope and magnetometer data to place into DMP register for the modified version SparkFun MPU-9250 DMP library to work with. Having pre-installed calculation and calibration functions in the DMP, accurate orientation information could be obtained.

2.2.4 ROS Serial

ROS serial is a package to serialize ROS message to be sent via multiple forms of connections, e.g. serial, Wi-Fi and Bluetooth, facilitating communications between the microcontroller and the single-board computer. Establishing a ROS serial node for the microcontroller, it acts a hub of message publishing and listening.

In the current architecture design, connections via serial Wi-Fi and Bluetooth are implemented, where configurations are done in the microcontroller with the NodeHandle interface and ArduinoHardware object. Supported by proper initialization of individual types of wired and wireless connections, e.g. SSID and password configuration for Wi-Fi, the ROS serial node could be accessed with ease (Espressif Systems).

2.3 Cheetah-inspired Hopping and Maneuverable Platform (CHAMP)

2.3.1 Hardware Interface

To facilitate control of hardware components connected to the ESP32 microcontroller, there is an adoption of `esp32_hw_controller` package from ROS. Joint state controller (`sensor_msgs/JointState`), Joint trajectory controller (`sensor_msgs/JointTrajectory`) and IMU sensor controller (`sensor_msgs/Imu`) are controllers used with their corresponding

message ROS topics. With ROS serial, the messages are forwarded and received by publish and subscription of topics by the ESP32 microcontroller. The joint state controller is referring to the trajectory commands due to the nature of servos being an internal closed loop. It is noted that there could be error of estimation of the odometry (chvmp, 2023).

2.3.2 URDF Model

The URDF model is designed in the 3D modeling tool, Blender, by stacking STL files of 3D models for individual link elements. Relationships between links and joints, offsets, rotation axes and limits are defined during the composition of the URDF model (see figure 2.9).

-5

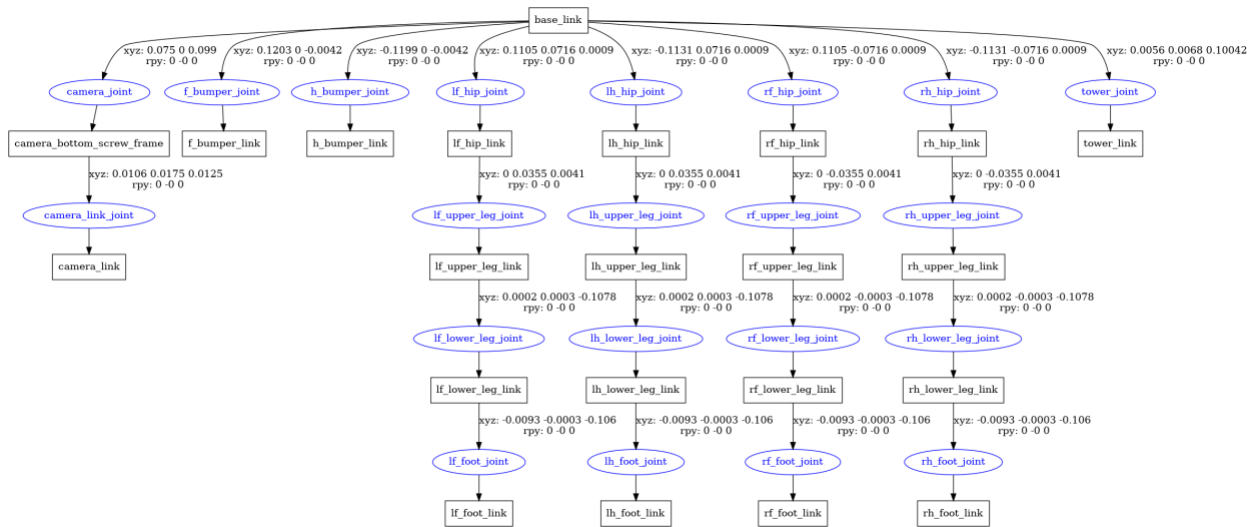


Figure 2.9: URDF model in tree structure

Base_link is the element indicating the origin of the robot, presumably locating in the center of the robot, with roll, pitch and yaw values set to 0. To configure the joint rotation limits, they are all referencing against the legs being fully stretched out to the ground according to their physical capabilities of mechanical design, blocking the inverse-kinematic engine to assign joints to physically impossible positions, causing stalls of servos to eventually break (see table 2.2).

Joints	Lower Limit	Upper Limit
Hip joints	-50°	50°
Upped leg joints	-60°	60°
Lower leg joints	-120°	-60°

Table 2.2: configuration for joint limits of URDF model

The configured URDF model is used for generating CHAMP configuration packages, subsequently used in calibration for hardware with gait configurations (see table 2.3).

Parameter	Value
Odometry scaler	0.9
Max linear velocity X	0.25m/s
Max linear velocity Y	0.25m/s
Max linear velocity Z	1.0m/s
Stance duration	0.20s
Leg swing height	0.05m
Leg stance height	0.24m
CoM X translation	-0.025m
Swing depth	0.00m

Table 2.3: CHAMP gait configuration parameters

2.3.3 Self-balancing algorithm

Due to various limitations of the hardware components, including limited torque of PWM servos, build quality of the 3D model and the absence of foot sensors to provide more accurate locations of legs, despite the hierarchical control algorithm provided by the CHAMP package, the balancing algorithm of may not execute as expected. Therefore, a self-

defined self-balancing algorithm could be implemented for the IMU. By having a balanced and leveled body, some sensors like the stereo camera may be able to capture more accurate orientation data when the robot is moving.

Referencing from a two-wheeled robot self-balancing algorithm, robots are assumed to be stable when the body is leveled. If the orientation quaternion retrieved from DMP is aligned with `base_link` element of the URDF model, the orientation value of the IMU is true to reality. Despite the hardware and external factors hampering the outcome, a PID Control algorithm could integrate more realistic orientation.

After implementing the aforementioned algorithm, the IMU orientation values could be used as controlled variables for the self-balancing algorithm. Considering that the rotation of z-axis is unrestricted, x and y axes (roll and pitch) are the emphases of the algorithm design.

With the import on `QuadrupedController::controlLoop_` in ROS package `champ_base`, the algorithm retrieves the errors of roll and pitch values calculated by the sensor-based orientation and position values. With PID control, roll and pitch commands could be updated and be passed inverse-kinematics functions and commanding the robot's movements. The last step is to tune parameters of PID control, which is done by trial-and-error.

2.4 Computer Vision and Machine Learning

2.4.1 3D Scene Reconstruction

Post disasters, the scene of affected areas could drastically change that in order to facilitate tasks including scene visualization, frontier exploration and autonomous navigation, reconstruction of the unknown scene could be an inevitable process. To minimize latency of the machine learning algorithm, a GPU-accelerated library, `Nvblox`, could aid the task by leveraging the GPU in the Nvidia Orin Jetson Nano.

The SLAM package is a tool for constructing a sparse representation of the environment, which the effectiveness was computing real-time localization and mapping tasks have been proven. Despite that, to facilitate navigation, including exploration of the unknown environment and dynamic object following, there is the requirement of an extra layer of information, density of obstacle (Cadena et. al, 2016). SDF, a function that calculates the orthogonal distance from a point to some given boundaries and returns whether the point is within or outside the boundaries. TSDF and ESDF, variations of SDF computed by Nvblox, which are the voxel array of distances to the nearest surfaces and the array of Euclidean distances with a truncation filter of a threshold, are widely researched in robotics and computer vision (Newcombe, 2011).

Nvblox could work with depth information retrieved from infrared cameras and 3D LiDAR, that it get depth and color input data from topics under sensor_msgs (see figure 2.10). Using the synchronized combination of infrared depth images and embedded IMU data, detection of visual landmarks could create an estimation of pose and odometry. Without sufficient visual landmarks, the odometry information will rely on the IMU data.

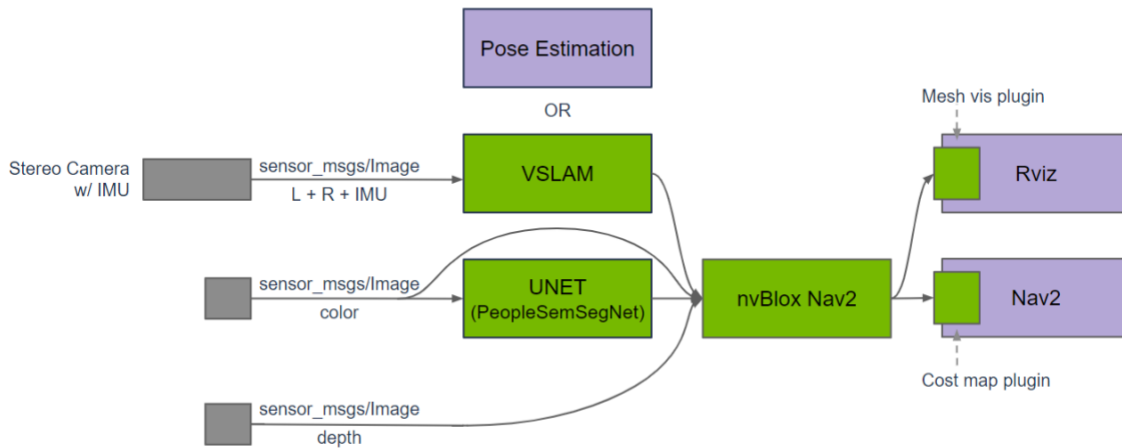


Figure 2.10: ROS node diagram of Nvblox

The RGB colored information retrieved could be useful for detecting any dynamic object in the environment, that the pixels could be isolated during the reconstruction of scene. Using

the RGB input images, there could neural network involved to create an image segmentation output for identification of dynamic objects, where used as a mask for scene reconstruction.

Having analyzed the pose information, depth and RGB image, they could be parsed as camera extrinsic parameters that the 2D images could be deprojected into a colored point cloud, which assists narrowing down voxels to be updated and prioritizing merge of TSDF by processes like update looping, raycasting or projection mapping (Oleynikova, et. al, 2017). Handled by Nvblox, there could be GPU-accelerated parallelization involved, giving better performances in comparison to other libraries like Voxelblox (Oleynikova, et. al, 2017).

Having acquired the TSDF of the scene, a 2D costmap could be generated by name slicing with predefined minimum and maximum obstacle height, aggregating the voxel layer values, subsequently indicating the occupancy of pixels in the explored scene. Meanwhile, with the aid of the efficient marching cube algorithm, a visualized mesh could be constructed from the TSDF. The algorithm iterates the process of determining a triangular pattern with 8 voxels for the entire TSDF array (Lorensen & Cline, 1998).

2.4.2 Autonomous Navigation

Autonomous navigation is the foundation to foster intelligent tasks of the robot, including dynamic object following and frontier exploration, which could be facilitated by Nav2 stack.

Nav2 stack is a modular behavior tree-based navigation stack, that behavior tree represents the navigation logics based on a tree-based execution model, assisting the interpretation from complicated navigation logics to readable format (Macenski, et. al, 2020; Colledanchise & Ögren, 2018). The behavior tree could also make structured safety and robustness analysis with state space description (Colledanchise & Ögren, 2018).

Nav2 is structured modularly, allowing switch of components with other compatible alternatives. For instance, with a tuning radius near 0, Savitzky-Golay smoother would be the module of choice to only reduce noise from the generated path (Macenski, et. al, 2020).

2.4.3 Dynamic Object Following

To ensure monitoring of the robot without adding extra weight to the rescuers during search and rescue, a dynamic object following feature to enable automatic following the footsteps of human could be found useful. Apart from that, leverage the modularity of system design, with simple modifications, the algorithm could be refined to tracking or following other targets.

To successfully execute the dynamic object following module, the algorithm has to identify human on screen with colored image, estimate his position in 2D coordinates, deproject it into the 3D space and eventually derive the goal pose with the behavior tree (Minaee, et. al, 2021).

The first step, which is to check the existence of a human, could be done by using pretrained models like PeopleSegNet ShuffleSeg model to create an image segmentation of the RGB image from the sensor input, labeling the pixels to human and background (non-human) (see figure 2.11)(Minaee, et. al, 2021).



Figure 2.11: example of human segmentation of RGB image (red as human and green as background)

Each strongly-connected component in frame is considered an individual object of human. To avoid confusion of having multiple humans in frame or objects created by misclassification, the largest object found by OpenCV library is treated as the target, the

centroid of which will be used as the targeted 2D coordinate to be deprojected in the 3D scene (Bradski & Kaehler, 2000). Matching the RGB image with the simultaneous infrared depth image, the depth of the coordinate could be obtained, facilitating a rescale to deproject to the 3D landscape.

By acquiring camera intrinsic and extrinsic information by subscription of topic camera_info and analysis of visual SLAM package, the goal pose could be derived and subsequently creating a behavior tree for the navigation towards the targeted human. The planning server would refresh its planned path with the constantly updated costmap, until the robot is a meter away from the target. The goal would also be updated by subscribing to the ROS topic, goal_update.

2.4.4 Frontier Exploration

Frontier exploration is a process to enhance the field mapping of 3D scene reconstruction for unexplored areas, which the package explore_lite could be adopted (see figure 2.12).

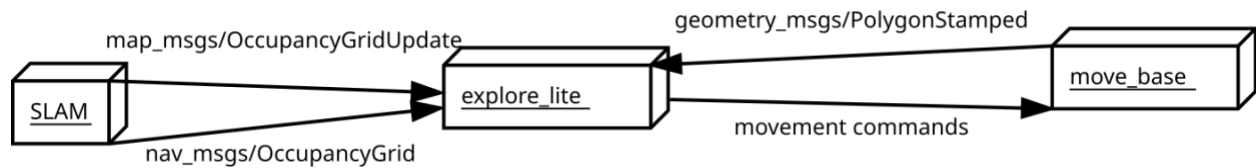


Figure 2.12: architecture for frontier exploration package explore_lite

Explore_lite is a package that traces the costmap of interest by retrieving information from message types nav_msgs/OccupancyGrid and map_msgs/OccupancyGridUpdate. A breadth-first search is conducted with the availability information from the costmap, returning an array of frontiers after a series of movement commands published to the navigation server (Hörner, 2016). By defining parameters like potential_scale, orientation_scale and min_frontier_size, the robot would use a greedy algorithm to traverse the designated unexplored areas to find all frontiers (see figure 2.13) (Hörner, 2016).

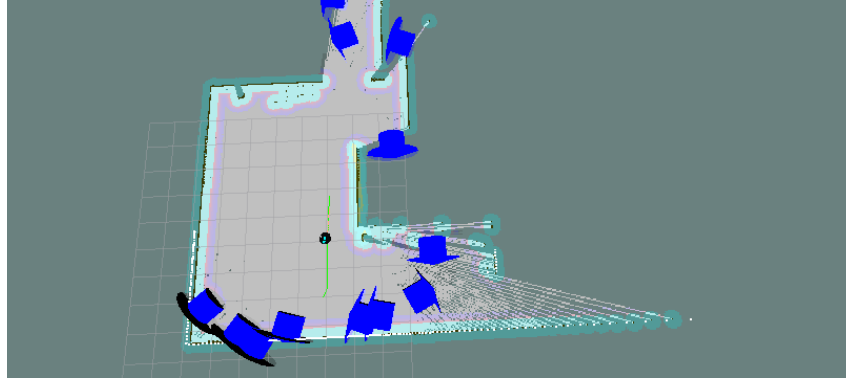


Figure 2.13: Example of frontier exploration (blue boxes as explored frontiers)

2.5 Object Detection

In order to facilitate the exploration of possible human lives, object detection algorithms could be utilized, which creates predictive bounding boxes of the identified objects from the list of identifiable objects (Honer, 2016). Detection model usually make efficient computation with the real-time YOLO detector, giving the detected objects classification labels (Honer, 2016). The objects of interest could be filtered out and whenever there is a new detection, a message could be published to relevant ROS topics.

2.6 Web Application

The web application is developed using the React.js framework with the assistance of relevant node.js packages. Individual interface elements are defined as standalone components, which are organized in a hierarchical approach, rendering from the bottom elements all the way to the top of the hierarchy (Saks, 2019). For any state changes involved, the `useEffect` method enables re-rendering of elements to ensure that the most updated information is being shown on screen (React team, 2024).

2.6.1 Appearance

Following the UI/UX design mentioned in section 2.1.3, the appearance of the UI elements and the navigation flow is developed based on the branding elements and prototype design.

To achieve a unified appearance with efficiency, Material UI package is adopted for interface element building. It enables a theme provider, allowing the customized definitions of color scheme, transitional effects and more, fostering a unified appearance (MUI team, 2024). Despite the wide usage of MUI, there are minor style definitions made by Cascading Stylesheets (CSS).

2.6.1.1 Mobile Responsiveness

Despite the sole development of a web application, the mobile users are also targeted, thus requiring some mobile responsive adaption in the user interface. To cater the vertical viewport of mobile devices, some originally horizontal layouts are switched to vertical (see figure 2.14 & 2.15).

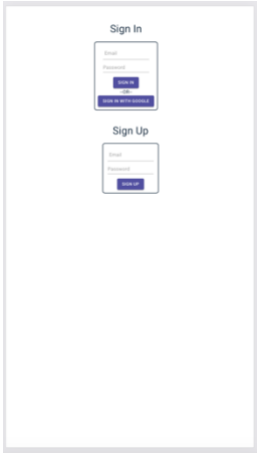
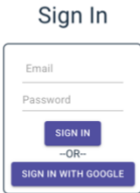


Figure 2.14: Horizontal login page layout on desktop

Figure 2.15: vertical login page layout on mobile devices

Meanwhile, the top navigation bar is also modified to a vertical layout, which the horizontal layout is replaced by a button-triggered drop-down menu, where the button is placed on the top-left corner of the screen (see figure 2.16 & 2.17).



Figure 2.16: Desktop view of navigation bar



Figure 2.17: Mobile view of navigation bar

2.6.2 Architecture

2.6.2.1 React Framework

On top of the hierarchy, there is the App.jsx file which is responsible for universal elements to be rendered on all web pages. For elements in the individual web pages, they are defined by separate jsx files stored in the components folder.

To implement navigation between pages, the react-router-dom package is applied to define the paths for each page (which is a react component). It creates an outlet element for the main rendering component, App.jsx, which renders the elements according to the url paths. However, since the navigation bar ought to be shown in all pages of the website except for the login page, it is separately included in App.jsx and show according to a Boolean flag.

To facilitate the changes of variable values of elements under the elemental hierarchy, not only could the value change be indicated by parsing parameters and returning values, but contexts could also be defined for the universal management of value changes. In the web application's architecture, subscription to ROS web sockets (detailed in section 2.4.2.2) is done by this means, that a context (RosContextProvider.jsx) is created, responsible for

connection to the ROS provider, allowing other React elements to directly use the context to subscribe the topics required, reducing the risk of establishing multiple connection to create asynchronous issues.

2.6.2.2 ROS communication

After establishing a communication channel on the robot as a provider via port 9090, the publish of ROS messages via the library `rosbridge_suite` could made to the non-ROS environment, where the web application could connect and subscribe with the assistance of the JavaScript package from ROS, `roslibjs` by sending JSON-based commands (ROS.org, 2024). The connection to the ROS provider in `RosContextProvider` utilizing `roslibjs` API enables demonstration of node and topic information, and the subscription to various topics. Having the `ros1_bridge` established, topics from both ROS1 and ROS2 could be listened by the web application. If a React component requires specific ROS information, it could import the `useContext` method to use the connection, then designate the topic name and message type and create a listener to receive the specified details of the robot's state.

2.6.3 Features

2.6.3.1 Live Camera Feed

To obtain the livestream of the stereo camera on the robot, the data could be retrieved from the with the assistance of the `web_video_server` package from ROS by publishing an imagery capture stream over HTTP. The stream render is done by feeding results of an HTTP request to a UI element. The slider responsible for adjusting the frame rate is modifying the update rate of the image feed, creating an illusion of it being a live video feed.

2.6.3.2 Virtual Simulation

To give user a visual understanding of the robot's current position and movements, virtual simulation is adopted in the web dashboard. To achieve that, there are 2 main parts of development.

The first part is retrieving the real-time link and joint information of the robot and information of the odometry, which could be done by simply using the defined ROS provider context for connection configuration and subscribing to the 'joint_states' and 'odom'. The former provides URDF data, showing the current positions of every joint, while the latter provides the estimation of distance and orientation compared to the robot's starting point.

The second part is the attempt to rebuild the visual appearance of the robot. Three.js is a node package that could create 3D visual space on React, allowing a simulated experience of traveling along with the robot (Three.js). Utilizing the node package URDFloader, by importing STL 3D visuals with the URDF model representing the relations of links and joints, coordinates of the joints could transform into a virtual robot (Johnson, 2023). To adapt to the axes orientation of the Three.js differing from ROS, rotation of -90° in x and z-axis have been made to base_link of the URDF model. Integrating with the information received from ROS in real time, the robot could virtually move around the virtual space.

Despite the option to drag to move x-axis and y-axis of the view, to obtain a better point-of-view of the robot's current position especially when it traverses around, a view control panel for the simulation space could be added. Leva, a graphical user interface that offers a responsive control panel, allowing expansion and collapse on individual section could be a convenient UI tool. With the predefined components from react-three-fiber from Three.js, capable of orbit control, virtual scene construction and lighting adjustments, the buttons in the Leva control panel could trigger different events, responsible for zooming, rotating, positioning to center and more angle adjustments, where frontier exploration could also be triggered by message publishing to ROS topic (Three.js).

2.6.3.3 User Authentication System

For better data security and privacy management, an account registration and login system, where done on the landing page of the web. This could prevent the unauthorized access to the information of the robots' current locations and the risk of leakage of confidential information captured during the exploration of the robot.

To facilitate the authentication process, a backend server and account database are required. To relief the strains of infrastructure setup and maintenance, Firebase Authentication, an account management platform offered by Google, could be suitable for the use case (Google). It handles the account authentication process for the system, while integrating well with Google's ecosystem to leverage the cloud storage and also providing simplistic user interface for management (Google). It also offers extra features, such as open authentication, allowing users to registering and signing in by connecting their accounts to their Google accounts, enhancing the seamlessness of user experience (Google).

To include Firebase Authentication to the system, it could be done by importing the firebase node package, providing configuration details with simple JavaScript code (Google).

2.6.3.4 Push Notifications

To demonstrate the results of object detection feature, a push notification system is set up for notifying users if there are any interesting objects detected by the system. A pop-up message is shown on screen to instantly grasp the attention if users, which the information is received from the listener of the ROS topic `detections_output`. This could be particularly useful during search and rescue that if new potential lives are found, the team of monitoring could be notified instantaneously (Nvidia, 2024).

2.7 Collaboration Tools

As a team of 4 with distinct operating systems involves (Windows, MacOS and Linux), an organized resource sharing system is demanded, where some of the collaboration tools are highlighted.

2.7.1 Git

Git is a version control system commonly used for team collaborative development, while GitHub is an online platform for git repository management, free for code contribution for

non-commercial use. By creating a repository on GitHub, teammates could all contribute code and resources with Internet access. It offers user-friendly and flexible interfaces such as desktop application and command line interface to allow a seamless contribution and code merging experience (Software Freedom Conservancy). To maintain the robustness and executability of code, developers could make changes locally, and commit to the repository later, preventing instant synchronization that code could not be successfully compiled for testing or deploying (Software Freedom Conservancy). To avert conflicts of code during instantaneous development, there is the option to create new branches to let users make changes based on a designated version of code, and merge or rebase after completion (Software Freedom Conservancy). These features could elevate the collaborative effectiveness of collaboration, especially on the project with a wide scope.

2.7.2 Docker Container

There is only one quadruped robot physically assembled, weighing over 10 kilograms. Considering the physical constraints, solutions are needed for developing in remote locations to facilitate testing of features. An image simulating the operation of ROS could be defined by a dockerfile. With docker compose files to run containers on local computers, simulation of ROS messaging system could be utilized for testing the connection via rosapi on the ROS interface without operating system compatibility issues.

2.8 Development and deployment

Since the ROS packages being adopted in the project are divided in multiple ROS distributions, there are multiple Docker images are built for launching the packages, which is listed as the follows:

- ROS1 Melodic – to install the CHAMP and rosbridge_suite packages.
- ROS1/ROS2 bridge – to build the ros1_bridge package with the additional feature of allowing customized message types.
- ROS2 Humble – to install the computer vision and Nav2 packages.

Docker compose files have been defined to launch the developer container for testing and debugging.

3 Experiment and Result

3.1 Hardware

3.1.1 Robot Construction

Despite experimenting the assemble robot, the key electronic components of the final version include a single-board computer with an attached GPU, a stereo camera and 7V 2-cell lithium battery (see figure 3.1).

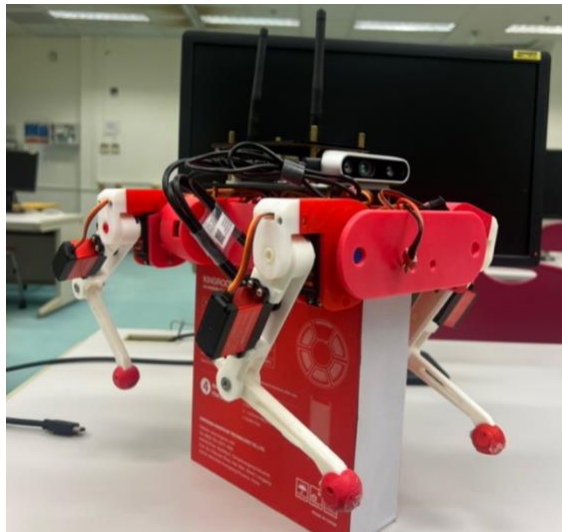


Figure 3.1: Final version of robot construction

There was a series of experiments with the construction of the robot. Assembling all purchased components, including the planar LiDAR and an external power bank to feed the single-board computer could be made possible, but the accumulated weight could not be borne by the 3D-printed limbs and power of the PWM servos. To minimize burden of the robot, the LiDAR sensor is removed, and the power bank is detached from the construction.

One of the previous versions of construction includes similar components, with the addition of a planar LiDAR and a power bank, but it was removed due to the weight and servo tolerance (see figure 3.2).



Figure 3.2: robot installed with a planar LiDAR and power bank

3.1.2 3D Modeling

As mentioned in section 3.1.1, there was a range of electronic components installed and tested during the development process, thus the 3D model inherited from the HKU CS MakerLab have to make adjustments to fit the new components.

To enhance the robustness of 3D model, there were multiple attempts for refining the individual models, stored in 3D model file extension, STL. In spite of novice of 3D modeling, the model creation was done smoothly. However, the performance of 3D printers in both HKU CS Makerlab and Innovation Wing vary, frustrates the remodeling of parts to find the perfect fit into the robot model. This also affects the rigidity of the overall construction, which could lead to larger vibrations during walks, affecting the accuracy of sensor data and the execution of software command to navigate to a destination.

For example, the upper shoulder joint is modified to become thicker to prevent wearing down during sideway movements due to the upgraded PWM digital servos for higher power (See figure 3.3, 3.4 and 3.5).

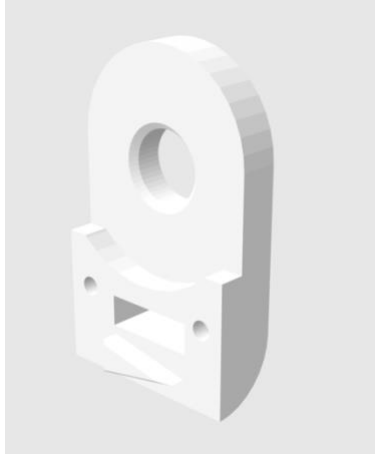


Figure 3.3: original design of upper shoulder joint inherited from HKU CS Makerlab

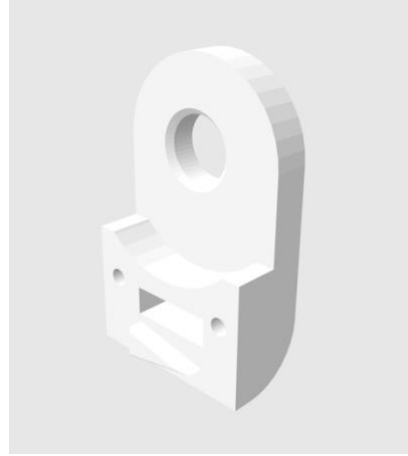


Figure 3.4: modified design of the upper shoulder joint



Figure 3.5: printing results of upper shoulder joints

To give another example, in order to include the larger-sized single-board computer, NVidia Orin Nano Jetson, there have been multiple attempts of replacing the top cover from HKU CS Makerlab's model (See Figure 3.6, 3.7 & 3.8). Due to the print size errors and time consumption of the HKU CS Makerlab's 3D printer, the design is formed by laser-cut acrylic boards cut in HKU InnoWing and Copper stands for lifting and securing the boards' positions (See Figure 9).

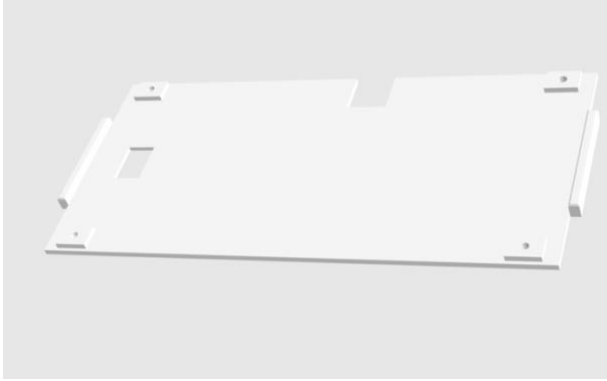


Figure 3.6: original model design of top cover from HKU CS Makerlab

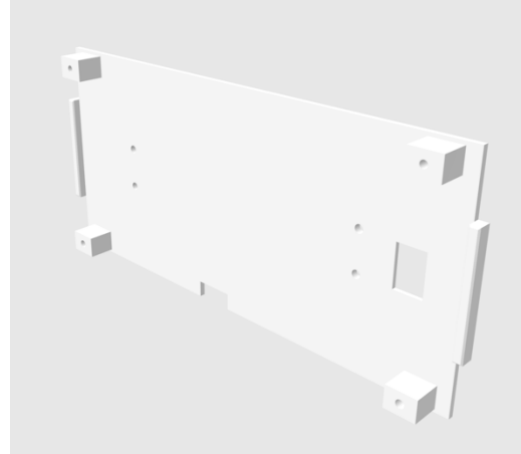


Figure 3.7: modified top cover model

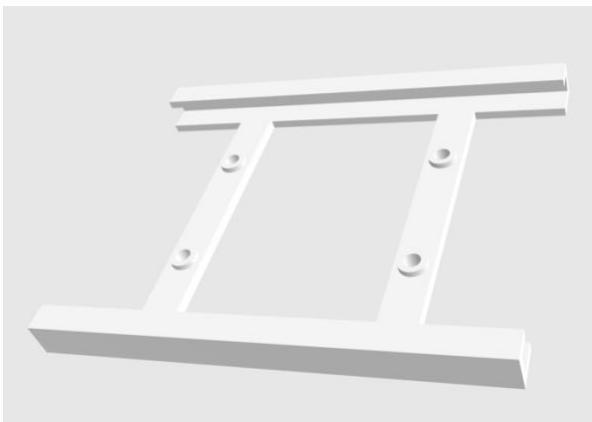


Figure 3.8: discarded version of 3D model of single-board computer holder



Figure 3.9: Final model design with laser-cut acrylic boards and copper stands

3.1.3 PWM Servos

With the more demanding sensors and computational power, new modeling parts are required to secure the positions of the PWM servos. One of the solutions was to create some laser-cut platforms. They move accurately to the angles commanded by software signals with the effort in calibrating the joints with the PCA9685 I2C PWM driver.

Nevertheless, the power and torque is limited that the weight support and rotation may not be sufficient, especially when the weight distribution was not thoroughly planned in the construction stage.

3.1.4 IMU

The accurate orientation of the robot could be computed by the DMP on the MPU-9250. However, due to the nature of the IMU and some physical limitations, there are encounters of drifts during locomotion. The current setup is limited to finding the local orientation relative to the robot, that inaccuracy may amplify by the accumulated error of orientation estimation. One of the possible solutions is to enable a magnetometer to find the absolute orientation.

3.1.5 Stereo camera

The stereo camera works as expected with environment of sufficient lights with simple lighting setup. It requires much calibration when more environmental variables are added, such as insufficient light and multiple sources of lighting (see figure 3.10). This could affect any computer algorithm that involves the analysis of the depth image.

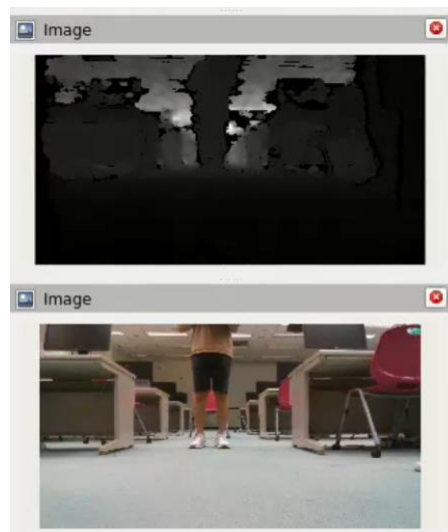


Figure 3.10: example of underexposed depth image

3.1.6 Single-board Computer

Even with the GPU installed, the CPU (6-core ARM) and main memory (shared 8GB) may not be sufficient for the nodes deployed on it using docker, including ROS1, ROS1-ROS2 Bridge, ROS2 and the web server. Consequently, swaps of memory spaces may occur that could weigh on the CPU, thus creating high latency. Fortunately, the computer vision and machine learning modules runs individually on the GPU, that offloading the inverse-kinematics engine and web server could show improvements in performance.

3.1.7 Self-balancing algorithm

In static positions, the robot could balance itself with leveled body in both sloped and discontinuous terrains with PID control (see figure 3.11, 3.12 & 3.13).

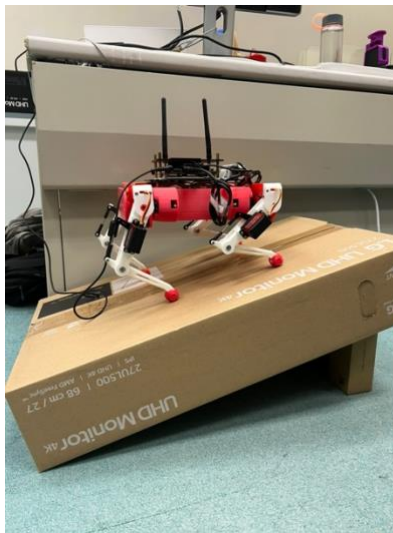


Figure 3.11: robot balancing on sloped surface

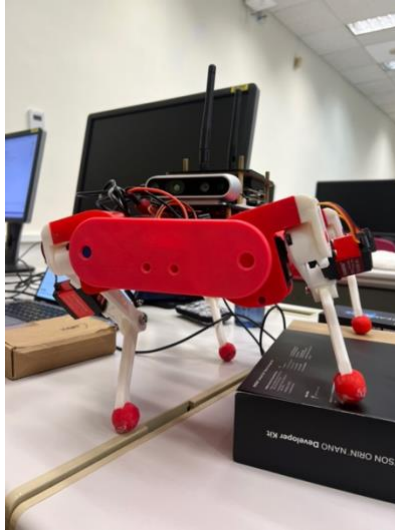


Figure 3.12: robot standing on a box (roll)

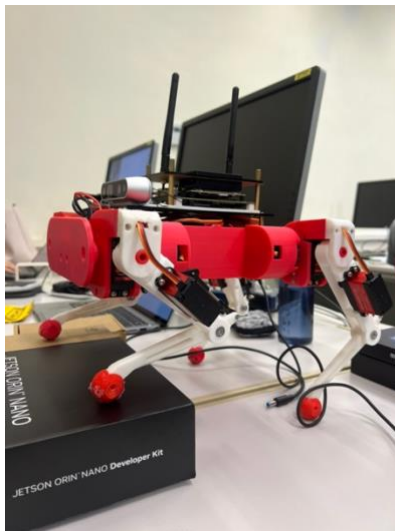


Figure 3.13: robot standing on a box (pitch)

In spite of the successful end result, there were multiple unsatisfying attempts. Without controlling the integration value of PID control, the body might not be able to level despite balancing on sloped surfaces (see figure 3.14).



Figure 3.14: robot balancing on sloped surface with unlevelled body

3.2 Computer Vision and Machine Learning

3.2.1 Visual SLAM and Frontier Exploration

The derivation of 2D costmap of the surrounding was satisfactory. Shown in figure 3.15, the green dots represent low probability of being occupied, while the red ones represent the high probability of the space being occupied, which is fairly true to reality compared to the RGB image. The grayscale plane underneath the dots is a representation of the feasibility of routing, which the darker it is, the less likely the robot routing system in going to direct it to. Z-axis calibration was done for a more accurate estimation due to the placement of the stereo camera (on top of the robot).

For frontier exploration, the robot was intended to traverse in the lighter areas of the grayscale plane. Due to sensor limitation, the orientation measurements drifts during the navigation along the planned route, leading to distorted costmap. One of the solutions would be clearin the costmap once in a while, but redetection is required in the approach, creating more distortions during the rotation.

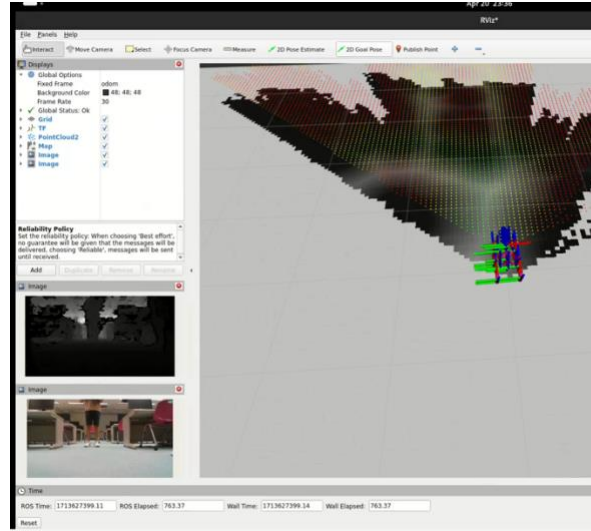


Figure 3.15: Result of visual SLAM and frontier exploration

3.2.2 Dynamic Object following

Human segmentation algorithm usually gives satisfying results, that the human is clearly identified, in red in figure 3.16. With sufficient calibrations, the depth image could be correctly shown. However, the estimation of the human in the 3D space could usually not be made correctly (the purple dot of the 3D map in figure 3.16).

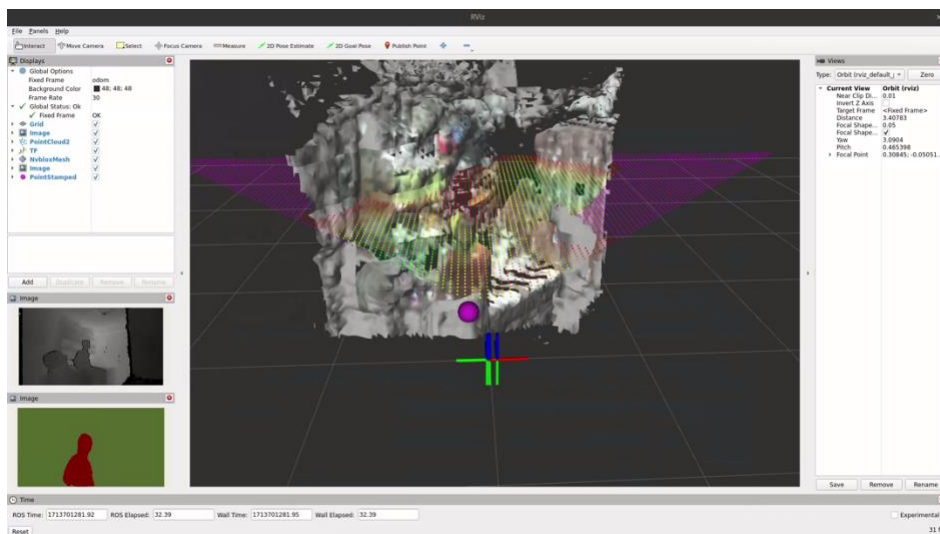


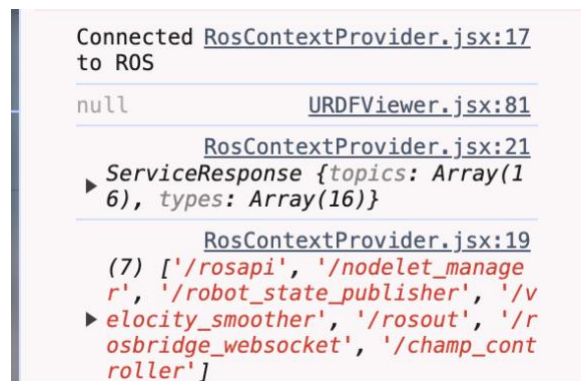
Figure 3.16 pose estimation result

The undesirable performance could be attributed synchronization issues, refresh rate and hardware limitation. The synchronization issue refers to the match of the segmented color image and the depth image. As they are obtained from separate ROS topics, the update rate may vary in the distributed ROS system. Therefore, the depth information may be misleading especially when the human moves to a different depth. The refresh rate of the algorithm could also make an impact when the target moves promptly. Moreover, when the robot is attempting to navigate towards the target, the hardware limitation causes a lot of vibrations, which affects the capture quality for the most updated imagery data.

3.3 Web Interface

3.3.1 ROS Connection

Assisted by rosapi, the connection to ROS has been successfully established, allowing retrieval of lists of nodes and topics (see figure 3.17). Topic subscriptions of individual elements have also been effective to display relevant data to be processed for UI rendering.



```
Connected RosContextProvider.jsx:17
to ROS
null URDFViewer.jsx:81
RosContextProvider.jsx:21
▶ ServiceResponse {topics: Array(16), types: Array(16)}
RosContextProvider.jsx:19
(7) ['/rosapi', '/nodelet_manager', '/robot_state_publisher', '/velocity_smoother', '/rosout', '/rosbridge_websocket', '/cham_controller']
```

Figure 3.17: Demonstration of successful connection to ROS provider and get the list of topics

3.3.2 Live Camera Feed

The imagery stream of the stereo camera of the robot is successfully pulled by HTTP requests in the format of mjpeg. The stream is displayed on the dashboard according to the

user-designated quality (a parameter of refresh rate input by the slider). Nonetheless, the delay and refresh rate could be affected by the network stability (see figure 3.18).

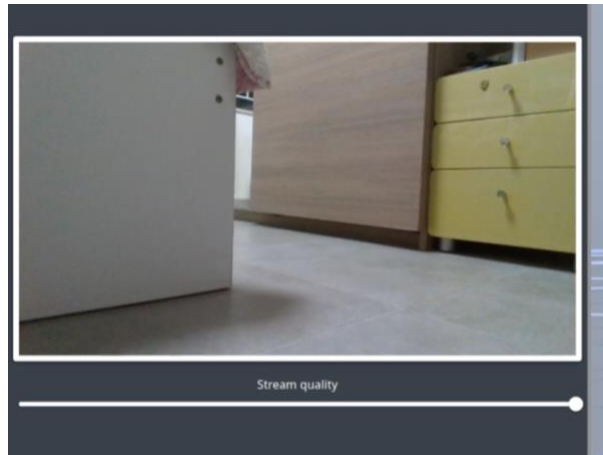


Figure 3.18: an example of live feed from the robot's stereo camera

3.3.3 Virtual Simulation

The virtual simulation of robot state is implemented on the web dashboard as expected, reflecting the movements of the robot joints and the traversal from odometry (see figure 3.19 & 3.20).



Figure 3.19: an example of moving robot virtual simulation of web dashboard

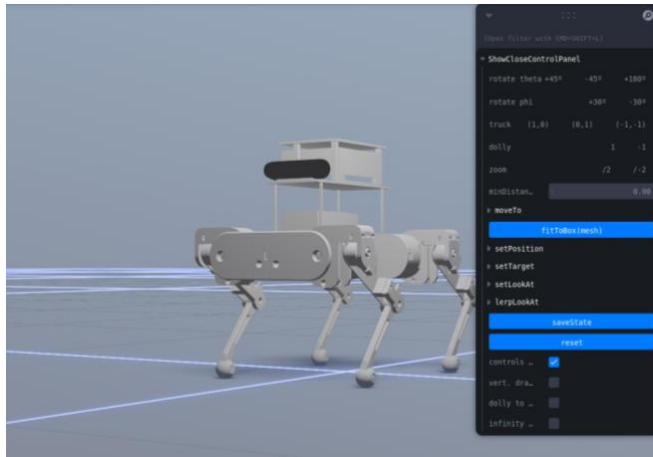


Figure 3.20: Demonstration of STL/URDF model loaded to simulator with view control panel

3.3.4 User Authentication System

The user authentication system has been successfully implemented to the web application. It could block unauthorized users from signing in (see figure 3.21). Registered users are successfully stored on Firebase Authentication (see figure 3.22).



Figure 3.21: example of unauthorized account login

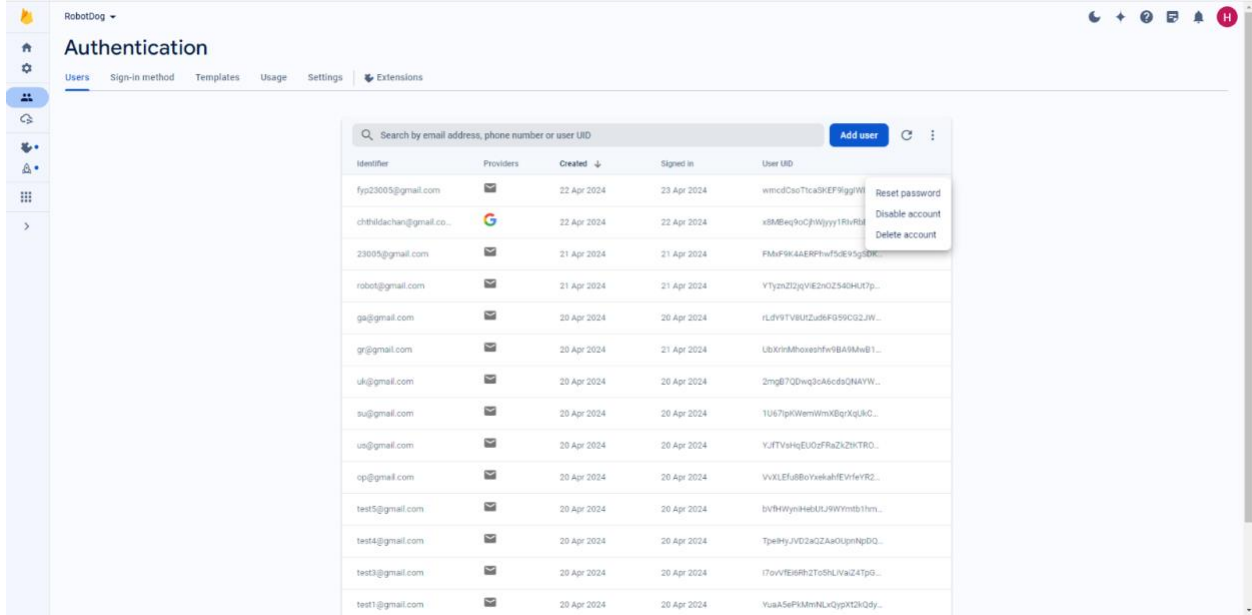


Figure 3.22: demonstration of registered email and Google accounts stored on Firebase (Google)

3.3.5 User Interface

The implementation of user interface design has mostly been successful by replicating the prototype design (see figure 3.23 & 3.24). It is worth noting that the UI design of the control panel does not align with the rest of the UI due to the restricted modifications that could be made to the Leva control panel element (see figure 3.25).

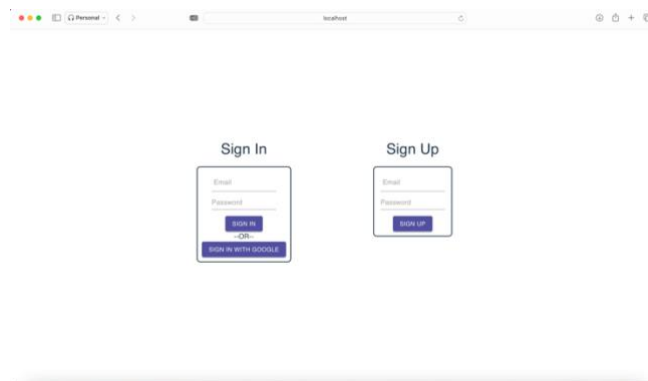


Figure 3.23: appearance of login page on web interface

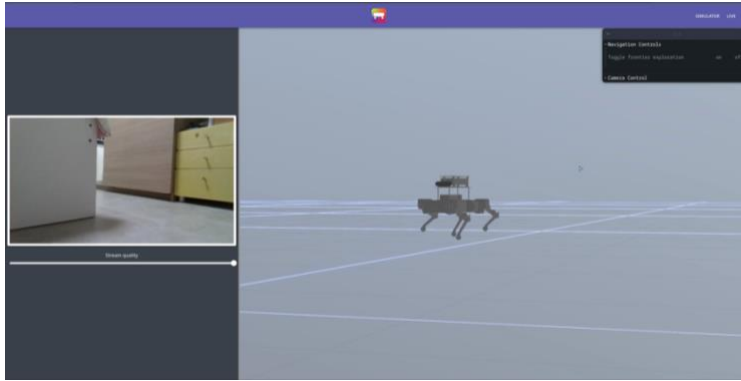


Figure 3.24: appearance of the main dashboard

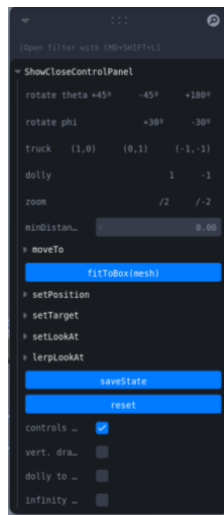


Figure 3.25: the appearance of web dashboard control panel component

3.3.5.1 Mobile Responsiveness

Mobile responsive features have been successfully implemented according to the UI/UX design (see figure 3.26 & 3.27). The top navigation bar, login page and main dashboard change to vertical layout responsively to optimize the portrait view of mobile devices.



Figure 3.26: Desktop view of navigation bar



Figure 3.27: Mobile view of navigation bar

3.4 Collaboration

It was an overall collaborative experience with assistance of the collaboration tools adopted in the project.

3.4.1 Git

Git and the corresponding GitHub repository were widely used to update new development code, having each teammate contributing to code of their responsibility. It was tricky to upload React.js code to the GitHub repository due to the large file size of node modules and the node dependency changes after pulling, that .gitignore file has to be carefully configured and install all dependencies before building. To avoid conflicted code development, branch development was a frequent practice, consequently demanding constant rebase of code. In spite of a smooth contribution experience, there were some conflicts encountered during the merge of contributions, which have to be resolved manually (see figure 3.28).

```
Auto-merging newviz/package-lock.json
CONFLICT (content): Merge conflict in newviz/package-lock.json
Auto-merging newviz/package.json
CONFLICT (content): Merge conflict in newviz/package.json
Auto-merging newviz/src/main.jsx
CONFLICT (content): Merge conflict in newviz/src/main.jsx
error: could not apply 4b06d73... record button
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 4b06d73... record button
~/D/C/robot-dog >>> git rebase --continue
```

Figure 3.28: example of code conflict during Git merge.

3.4.2 Docker

The simulation of ROS by the docker container composed successfully facilitated testing and debugging of the web application's interactions with ROS messages.

4 Future Work

4.1 Actuators

The adoption of PWM servos is due to the precision control it offers and limitation of mechanical knowledge of the team that the servo model could be inherited. However, servos are prone to stalling by its limited torque, abnormally pulling currents until breakage when the required angles commanded could not be met. To solve the issue, digital motors with higher torque could be considered. One of the alternatives would be a brushless motors, the Robomaster M2006 from DJI, that speed could be controlled by PWM or CAN signal, which integrates a motor encoder for locating the current motor position. With the information, the accuracy of PID control could be improved from positional error. To make the swap of actuators, there is a requirement of redesigning the motor housing for the robot.

4.2 Sensor Fusion

Despite the satisfying performance of DMP in finding orientation and the quality RGB-D information obtained from the stereo camera, there are limitations in data retrieval when relying on one type of sensor for 1 type of data. Therefore, a sensor-fusion approach could be implemented to improve the understanding of the surrounding and subsequently boosting the performance of machine learning algorithms. A case in point would be adding a LiDAR sensor to the model, which adopting prevalent point fusion and voxel fusion models like MV3D could offer a more comprehensive and accurate segmentation analysis and object tracking (Zhong et. al, 2021).

Another sensor that worth integrating would be the foot sensors, which would give the accurate feedback of foot positions to CHAMP. Nonetheless, there might be challenges in finding a suitable sensor, with the requirements of being small, sensitive and free from errors. For example, the accuracy of response of infrared sensor reactions varies from the surfaces the robot is on (Lomba, et. al, 1998). Another obstacle to overcome would be the

redesign of the leg 3D models to include the foot sensors. Taking advantage of the modularity of ROS, the high-level machine learning algorithms could be adapted to other hardware models of robots, which are equipped with more rigid structure and accurate sensors.

4.3 Swarm Robots

Considering the limited battery life and exploration speed of a robot, swarm robots could be introduced. Having a team of robots, robot models could be designed for shorter battery lifetime to have a lighter weight. The swarm of robots could be allocated to different areas of the site, via SLAM map merging and frontier exploration packages such as m-explore, the entire scene could be reconstructed in the centralized server (Hrnčíř, 2023). To maintain the user experience of the web dashboard, the virtual simulator should be upgraded to include multiple robots.

4.4 User Interface

Despite the display of key robotic status on the current web dashboard, there are flaws in the UI/UX design and the high-level algorithmic commands have not yet been included. To offer a cohesive and non-confusing interface for better user interactions, the control panel should be redesigned to using more precise indicators and unified color scheme. To stretch the capabilities of the application, commands including trigger of 3-D map reconstruction, object detection, and dynamic object following could be added on the control panel. The result of 3D map reconstruction and the 2D costmap could also be displayed on the virtual simulator with assistance of Three.js, allowing the hands-on rescuers to understand the current environment of the area and risk of difference areas, fostering prompt search, and rescue (Three.js). Moreover, to give users a more mobile experience, mobile applications for android and iOS devices could be developed by using framework like react, native offering preload contents, and utilizing native features like GPS for an extra layer of information.

In the emergency environment, constant checks of mobile devices may not be very feasible, taking advantage of the rosbriidge message topic subscription format, other forms of interfaces could also be considered, assisted by relevant ROS packages. For example, creating an application for smart wearable devices, giving notifications for new object detection, or using radio signals as transmission medium to Broadcast significant discoveries to communication devices like walkie-talkies.

5 Conclusion

Having researched on equipping computer vision and machine learning capabilities to a quadruped robot with a user interface to facilitate environment exploration, object identification and dynamic object following, the real-life execution of the software modules developed could be highly restricted by the hardware formation of the quadruped robot prototype. Nonetheless, the computer vision and machine learning algorithm results, especially scene reconstruction and object detection resulted fairly true-to-reality. The user interface is also implemented that could serve the purpose of providing simplistic user interactions with the software. It is believed that an improvement of hardware mechanics could show significant improve of the algorithms developed.

References

Adafruit Industries. (2023). Adafruit-PWM-Servo-Driver-Library. Retrieved April 24, 2024, from <https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library>

Aristidou, A., Lasenby, J., Chrysanthou, Y., & Shamir, A. (2018, September). Inverse kinematics techniques in computer graphics: A survey. In *Computer graphics forum* (Vol. 37, No. 6, pp. 35-58).

Bilousova, L. I., Gryzun, L. E., & Zhytienova, N. V. (2021). Fundamentals of UI/UX design as a component of the pre-service specialist's curriculum.

Biswal, P., & Mohanty, P. K. (2021). Development of quadruped walking robots: A review. *Ain Shams Engineering Journal*, 12(2), 2017-2031.

Bradski, G., & Kaehler, A. (2000). *Dr. Dobb's journal of software tools*. Dobb's Journal of Software Tools.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., ... & Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6), 1309-1332.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., ... & Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6), 1309-1332.

Chu, J. (2019, March 4). Mini cheetah is the first four-legged robot to do a backflip. MIT News | Massachusetts Institute of Technology. <https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304>

Chui, C. Y. (2022). END-TO-END DEVELOPMENT OF A ROBOTIC QUADRUPED FOR TRAVERSING ON CHALLENGING TERRAINS. The University of Hong Kong. <https://wp.cs.hku.hk/2021/fyp21046/>

chvmp. (2023). chvmp/champ. Retrieved April 24, 2024, from <https://github.com/chvmp/champ>

Colledanchise, M., & Ögren, P. (2018). Behavior trees in robotics and AI: An introduction. CRC Press.

E. Otten, "Inverse and forward dynamics: Models of multi-body systems," Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences, vol. 358, no. 1437, pp. 1493–1500, 2003.

Espressif Systems. (n.d.). Wi-Fi API - Arduino ESP32 latest documentation. Retrieved April 24, 2024, from <https://docs.espressif.com/projects/arduino-esp32/en/latest/api/wifi.html>

Garcia, E., Jimenez, M. A., De Santos, P. G., & Armada, M. (2007). The evolution of robotics research. IEEE Robotics & Automation Magazine, 14(1), 90-103.

Garigipati, B., Strokina, N., & Ghabcheloo, R. (2022, July). Evaluation and comparison of eight popular Lidar and Visual SLAM algorithms. In 2022 25th International Conference on Information Fusion (FUSION) (pp. 1-8). IEEE.

Google Fonts. (n.d.). Jura. Retrieved April 24, 2024, from <https://fonts.google.com/specimen/Jura>

Google Fonts. (n.d.). Roboto. Retrieved April 24, 2024, from <https://fonts.google.com/specimen/Roboto>

Google. (n.d.). Firebase Authentication Documentation. Retrieved April 24, 2024, from <https://firebase.google.com/docs/auth>

Guardian News and Media. (2008, August 14). Sichuan quake: China's earthquake reconstruction to cost \$150bn. The Guardian. <https://www.theguardian.com/world/2008/aug/15/chinaearthquake.china>

Guardian News and Media. (2024, April 3). "people were screaming": Hualien residents in shock after Taiwan earthquake. The Guardian. <https://www.theguardian.com/world/2024/apr/03/hualien-residents-in-shock-after-taiwan-earthquake>

H. Montgomery. (2024, January 5). Rescuers pull quake survivors from rubble in Japan as 72-hour 'golden period' closes. CNN World. <https://edition.cnn.com/2024/01/05/world/japan-wajima-earthquake-rescue-intl-hnk/index.html>

Holzer, A., & Ondrus, J. (2012). Mobile app development: Native or web?. In Proc. Workshop eBus.(WeB).

Hörner, J. (2016). Map-merging for multi-robot system.

Hrnčíř, J. (2023). m-explore. Retrieved April 24, 2024, from <https://github.com/hrnr/m-explore>

Johnson, G. K. (2023). urdf-loaders. Retrieved April 24, 2024, from <https://github.com/gkjohnson/urdf-loaders>

Lau, T. Y. (2023). 3D printed Robot Dog Walking on Terrain for STEM education. The University of Hong Kong. <https://wp.cs.hku.hk/2022/fyp22066/> .

Lee, J. (2013). Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control: Implementation on the MIT Cheetah robot (Doctoral dissertation, Massachusetts Institute of Technology).

Lee, C. K. D. (2022). 3D printed Robot Dog Walking on Terrain for STEM education. The University of Hong Kong. <https://wp.cs.hku.hk/2021/fyp21080/> .

Lomba, C. R., Valadas, R. T., & de Oliveira Duarte, A. M. (1998). Experimental characterisation and modelling of the reflection of infrared signals on indoor surfaces. *IEE Proceedings-Optoelectronics*, 145(3), 191-197.

Lorensen, W. E., & Cline, H. E. (1998). Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field* (pp. 347-353).

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66), eabm6074–eabm6074. <https://doi.org/10.1126/scirobotics.abm6074>

Macenski, S., Martín, F., White, R., & Clavero, J. G. (2020, October). The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2718-2725). IEEE.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7), 3523-3542.

MUI Team. (2024). MUI: The React component library you always wanted. Retrieved April 24, 2024, from <https://mui.com/>

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., ... & Fitzgibbon, A. (2011, October). Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality* (pp. 127-136). Ieee.

NVIDIA. (2024, February 8). isaac_ros_yolov8. Retrieved April 24, 2024, from https://nvidia-isaac-ros.github.io/repositories_and_packages/isaac_ros_object_detection/isaac_ros_yolov8/index.html#quickstart

Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., & Nieto, J. (2017, September). Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1366-1373). IEEE.

Otani, Y., Ando, T., Atobe, K., Haiden, A., Kao, S. Y., Saito, K., ... & Fukunaga, K. (2012). Comparison of two large earthquakes: the 2008 Sichuan Earthquake and the 2011 East Japan Earthquake. *The Keio journal of medicine*, 61(1), 35-39.

Plant, F. D. N. P. Improvements to the rescue robot Quince.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

React team. (2024). React. Retrieved April 24, 2024, from <https://react.dev/>

ROS Drivers. (2023). roserial. Retrieved April 24, 2024, from <https://github.com/ros-drivers/roserial>

ROS-Mobile. (2023). ROS-Mobile-Android. Retrieved April 24, 2024, from <https://github.com/ROS-Mobile/ROS-Mobile-Android>

ROS.org. (2024). ROS. <https://ros.org/>

Saks, E. (2019). JavaScript Frameworks: Angular vs React vs Vue.

Software Freedom Conservancy. (n.d.). Git - Documentation. Retrieved April 24, 2024, from <https://git-scm.com/doc>

SparkFun Electronics. (2023). SparkFun_MPU-9250-DMP_Arduino_Library. Retrieved April 24, 2024, from https://github.com/sparkfun/SparkFun_MPU-9250-DMP_Arduino_Library

Three.js. (n.d.). Three.js – JavaScript 3D library. Retrieved April 24, 2024, from <https://threejs.org/>

Tola, D., & Corke, P. (2024). Understanding URDF: A dataset and analysis. *IEEE Robotics and Automation Letters*.

Wong, H., & Chung, L. (2024, April 23). Big aftershocks rock Hualien county in Taiwan weeks after 7.3 earthquake. *South China Morning Post*.
<https://www.scmp.com/news/china/politics/article/3259979/big-aftershocks-rock-hualien-county-taiwan-weeks-after-72-earthquake>

Yoshida, T., Nagatani, K., Tadokoro, S., Nishimura, T., Koyanagi, E. (2014). Improvements to the Rescue Robot Quince Toward Future Indoor Surveillance Missions in the Fukushima Daiichi Nuclear Power Plant. In: Yoshida, K., Tadokoro, S. (eds) *Field and Service Robotics*. Springer Tracts in Advanced Robotics, vol 92. Springer, Berlin, Heidelberg.
https://doi.org/10.1007/978-3-642-40686-7_2

Zhong, H., Wang, H., Wu, Z., Zhang, C., Zheng, Y., & Tang, T. (2021). A survey of LiDAR and camera fusion enhancement. *Procedia Computer Science*, 183, 579-588.