GENG4411 Engineering Research Project Part 2
Final Report

Developing Humanoid Robotics: Human Pose Mapping, SLAM and Path Planning for Real World Applications

Joel Smith
23338559

School of Engineering, University of Western Australia

Supervisor: Professor Thomas Bräunl
School of Engineering, University of Western Australia

Word count: 6,971

School of Engineering
University of Western Australia

Submitted: October 13, 2025

Project Summary

The desire for dynamic and adaptable robots is increasing in modern day society where technology is progressing at a rapid rate. Previous humanoid robots have been extremely limited in their ability to traverse complex environments autonomously in real-world situations such as in the workplace. This project aimed to develop a humanoid robot capable of traversing dynamic environments by using a combination of human pose mapping and SLAM (Simultaneous Localisation and Mapping) to increase the performance of humanoid robots in real-world complex environments.

The project was structured around three components.

- **Human Pose Mapping**, this allows the robot to copy and record human movements using body-attached sensors.
- **SLAM Mapping**, this allows the humanoid robot to identify obstacles and build a map of its environment.
- **Path Planning Algorithms**, which allows the robot to safely navigate its dynamic environment to target destinations.

This research was programmed on a Unitree G1 robot using the ROS2 architecture. Human pose mapping was implemented using Sony's Mocopi body-mounted motion capture sensors integrated with ROS2. SLAM mapping utilised the robot's camera and lidar systems, with localisation achieved through a combination of Normal Distributions Transform (NDT) and Iterative Closest Point (ICP) algorithms. Path planning was implemented using a global A* path planner combined with a local Dynamic Window Approach (DWA) path planner to enable real time dynamic obstacle avoidance. Additionally, a web-based dashboard has been developed to run on the robot, providing completely remote viewing of live data and command execution capabilities.

Key achievements include successful self-collision detection combined with accurate tracking of human movements using Mocopi sensors, effective localisation through NDT-ICP algorithm integration, autonomous navigation using lidar SLAM, and robust path planning enabling real time dynamic obstacle avoidance. The web-based dashboard proved to be a highly effective central management system, allowing non-developers to control the robot whilst eliminating the need for external laptop connections to the robot.

Critical findings revealed that while Mocopi sensors effectively copy and record human actions, they are limited by amount of sensor output, as they are unable to detect fine motor actions such as hand closure due to their focus on major joints. Localisation performs effectively when provided with a quality base map and initial pose estimate. The A* and DWA combination provides fast, effective dynamic obstacle avoidance, enabling real time navigation. The web-based dashboard approach demonstrated significant advantages for robot control and management, providing accessibility and centralised operation.

Future work should focus on integrating additional sensors to capture fine motor movements to enable more complex movement monitoring, developing more robust initialisation procedures for localisation, fine-tuning path planning models to allow for the most robust planning possible, and expanding the web-based interface to include more advanced autonomous behaviours and multi-robot coordination capabilities.

List of Publications

H. Zhang, J. Inayat-Hussain, J. Smith, T. Ryan, T. Braunl. "A Comprehensive Control Architecture for Humanoid Robots: Integration of Locomotion, Manipulation, and Navigation Subsystems", Australasian Conference on Robotics and Automation, 2025, Perth, Australia.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction

Demand is rising for humanoid robots to perform in dynamic spaces as low cost humanoid robots are becoming more prevalent. This is important as humanoid robots can be integrated into human planned spaces without expensive infrastructure changes, giving advantage compared to traditional robotic systems. For humans, humanoid robots can be integrated into assistive tasks in society, i.e., hospitals, homes and schools, assisting people in need while dealing with difficult, changing environments. In industry, humanoid robots can perform dangerous or repetitive and task and conditions that render the workplace safer and more efficient.

Current humanoid robots have serious drawbacks when used independently to execute tasks in the real world. The ability to integrate intuitive human robot interaction, real time mapping, and dynamic obstacle avoidance, are major demands that are needed for actual world deployment. The implication of this work would be vast as without real world navigation abilities, humanoid robots couldn't be deployed safely without expensive preprogrammed solutions.

1.2 Background

Previous research in the field has been focused largely on navigation in static environments, such as Vedadi et al. (2024) [13] who employed a visual SLAM to guide a humanoid robot through an obstacle-free lab. While individual components such as SLAM algorithms and human pose estimation have been researched, little work integrates both under a single system. Current commercial humanoid robots operate by using preprogrammed routine, lacking adaptability required for real world application.

Advances in motion capture technology such as Sony's Mocopi sensors retailing at sub $450,USD, have made human pose mapping accessible for non-commercial applications. The Mocopi sensors consist of 6 wireless sensors attached to the body, capable of tracking joint positions on the body such as the head, wrists and ankles. The Mocopi sensors use their built in accelerometers to measure the 3D position of the joints in real time, which is then transmitted via Bluetooth to a mobile application. The application handles calibration and forwards data over WiFi to a connected computer, providing a low cost, portable alternative to traditional optical motion capture.

The availability of powerful humanoid platforms such as Unitree G1, with 29 degrees of freedom and onboard sensors, presents an opportunity for building and testing fully integrated autonomous navigation systems. The G1 is designed to integrate with the ROS2 architecture which is the de facto robotics development framework which offers complex, modular, real time capability needed for integration.

1.3 Project Objectives

The hypothesis guiding this project is that by combining human pose mapping with SLAM (Simultaneous Localisation and Mapping) and path planning algorithms, it could increase humanoid robots' ability to perform autonomously in real world environments. This project addressed this hypothesis through three objectives.

1.3.1 Objective 1: Human Pose Mapping

Develop accurate and repeatable human pose mapping that allows humans to demonstrate actions and the robots accurately mimic these movements. This was achieved using Mocopi sensors integrated with self collision detection to ensure that the movements are safe.

1.3.2 Objective 2: SLAM and Localisation

Implement SLAM mapping that allows the robot to map and detect live obstacles in its surroundings while determining its position within the environment. This was accomplished using the robot's inbuilt lidar and IMU combined with Normal Distributions Transform (NDT) and Iterative Closest Point (ICP) algorithms for localisation.

1.3.3 Objective 3: Dynamic Path Planning

Building upon SLAM, the implementation of path planning algorithms to generate efficient and safe paths for the robot to navigate its environment while avoiding static and dynamic obstacles is required. This was achieved through a global/local approach combining A* as a global planner with local Dynamic Window Approach (DWA) algorithm for obstacle avoidance.

These objectives advance the current state of the art by providing an integrated system that combines human pose demonstration with mapping and dynamic navigation on a humanoid platform. The successful achievement of these objectives could enable humanoid robots to be robust to human environments.

Achieving these objectives could be beneficial for stakeholders. For instance in healthcare, where McKinsey & Company (2020) [14] projects that automation might represent a $150 billion opportunity for improvement in healthcare. In industry, the ability to demonstrate new tasks through human pose mapping rather than time consuming reprogramming could reduce robot deployment time, which could improve operational efficiency.

2. Design Approach

This project is divided into three main areas human pose mapping, navigation, and system integration. Each area is further separated into a series of achievable components to efficiently meet the project's objectives using the Unitree G1 robot. The project has constraints imposed on it including the need for real time data processing, the computational limits of the G1, and safety requirements for operation with humans.

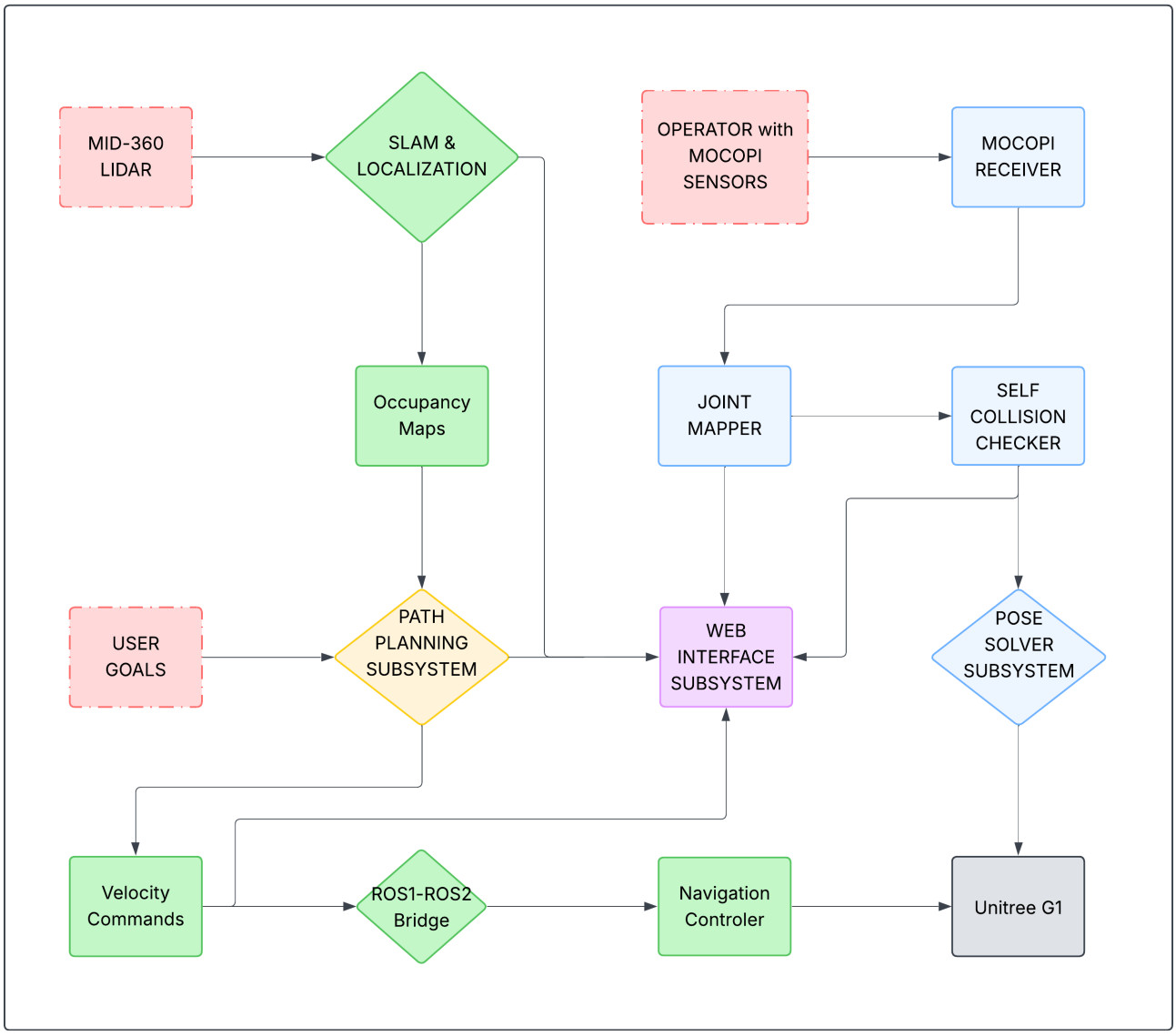


Figure 2.1: System architecture overview showing the main systems and how they interact with each other. Red: Input, Green: SLAM and Localisation, Yellow: Path Planning, Purple: Web Interface, Blue: Human Pose Mapping.

2.1 Design Constraints

2.1.1 Computational Constraints:

To maintain system isolation all code should run directly from the G1's onboard computer which limits processing power compared to running from a dedicated workstations.

2.1.2 Real time Performance:

To maintain natural interactions, human pose mapping should have sub 500ms latency, while robot mapping and path planning must operate at minimum 10Hz to ensure safe navigation avoiding live objects.

2.1.3 Physical Safety Constraints:

The humanoid robot operates in environments with humans, therefore the robot must incorporate self collision detection and emergency stop capabilities to be able to effectively protect itself and other people. Both velocity and acceleration are controlled to prevent damage.

2.2 Design Criteria and Success Metrics

2.2.1 Human Pose Mapping Criteria:

- *Accuracy:* Minimal joint position error compared to human.
- *Responsiveness:* Latency between operator movement to robot movement should be sub 500ms.
- *Safety:* Zero self collision events during operation.

Joint position accuracy will be verified by comparing reference poses and robot joint angles. Latency will be measured by comparing the time differences between human movement and robot movement using video footage. A latency goal of sub 500ms was chosen to preserve the sense of natural interaction between human and robot.

2.2.2 SLAM Criteria:

- *Mapping Accuracy:* Localisation error of less than 30cm
- *Robustness:* Navigation must remain reliable even in environments containing moving obstacles.
- *Real time Performance:* Path planning updates should occur at a minimum 10Hz.

Localisation accuracy can be measured by comparing the robot's reported position against its known position, with measurements taken across multiple test. The robustness will be assessed through trials involving moving obstacles. Performance is verified through ROS monitoring of the path planner's update frequency. The 30cm localisation threshold was selected based on the robot's physical footprint and typical doorway clearances in indoor environments, while the 10Hz update rate represents an acceptable update frequency for dynamic obstacle response.

2.2.3 System Integration Criteria:

- *Modularity:* Independent operation of pose mapping and navigation systems.
- *Usability:* The web interface must be able to be used and navigated by users that do not have a technical background and or without command line access.

2.3 Design Ideation and Alternative Approaches

2.3.1 Motion Capture Selection:

Sony's Mocopi motion capture sensors have been provided as part of the project as existing equipment. The research objective was to integrate this technology with the G1 robot and evaluate its performance for pose mapping applications. This constraint focused the project on the implementation challenge and performance rather than motion capture system comparison.

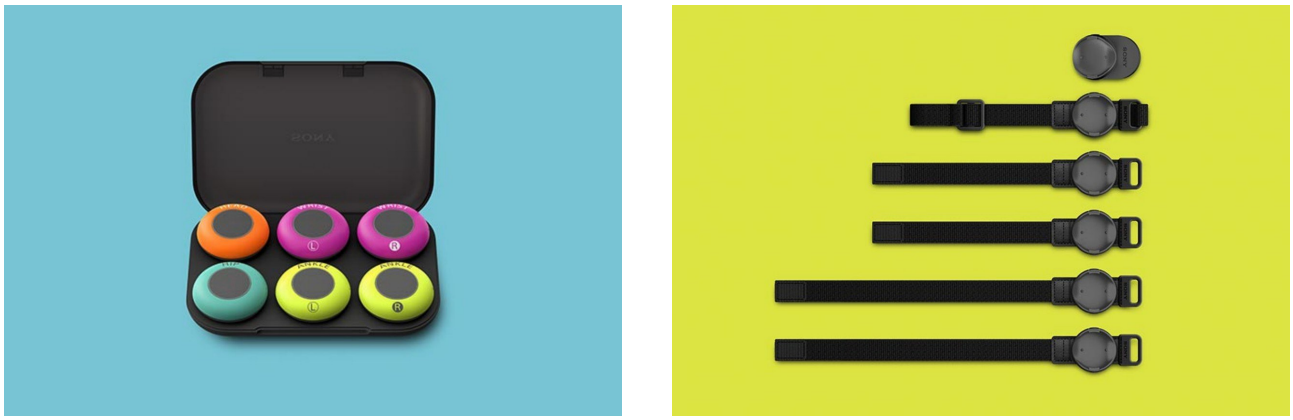


Figure 2.2: Sony Mocopi motion capture system components showing sensors and straps [4].

2.3.2 SLAM Algorithm Selection:

SLAM algorithms are well studied and have multiple implementations available such as ORB-SLAM3 or RTAB-Map [15] [16] which use a camera to map the environment, often called visual SLAM, or lidar based SLAM such as GMapping, Cartographer or SLAM Toolbox [17] [18] [19]. Visual SLAM approaches were considered due to the features available from camera data, however they were rejected due to their computational load and sensitivity to conditions. Lidar based SLAM was selected as the G1 platform has already been fitted with a MID-360 lidar, along with the strong consistent results, lidar based approaches have had in the past.

2.3.3 Localisation Approach:

Initially, scan based localisation offered by Google's Cartographer [20] was considered as it integrates into the G1. However, an existing implementation from the FAST_LIO_LOCALIZATION_HUMANOID package [1] combining the Normal Distributions Transform (NDT) and Iterative Closest Point (ICP) algorithms was identified that offered potential improvements in localisation accuracy and convergence speed for the MID-360's dense 3D point clouds.

2.3.4 Path Planning Architecture:

Recently researchers have been evaluating the use of AI either using supervised or deep reinforcement learning to try and improve on classical algorithms, however AI as a whole is quite heavy and slow compared to classical algorithms which may work against it in a situation such where compute power is limited. Several classical path planning approaches were evaluated. Dijkstra's algorithm provides optimal paths but explores the entire map which significantly increases runtime in large and rapidly changing maps. The rapidly exploring random trees algorithm has the opposite problem where it has fast planning but produces non-optimal paths requiring significant post processing. A* was selected for global planning due to its heuristic guided search which help provides optimal paths with reduced computational cost. For local planning, potential fields was considered but rejected due to local minima problems in cluttered environments where the robot would get stuck. The DWA planner was selected for its ability to consider robot dynamics while providing smooth, collision free paths in real time.

2.4 Design Tools and Software Architecture

2.4.1 ROS Framework:

The ROS framework has been used in the robotics development for nearly two decades, with ROS1 being released in 2007 and the newer ROS2 in 2020. Both provide stable, supported platforms to build robotic systems. ROS1 has a larger library to select from, while ROS2 provides improved performance and more modern integration. ROS2 was chosen to be the primary framework for this project as it has enhanced performance and good support. However, for some specific components of the project ROS1 was also used where important libraries or functions had not been implemented in ROS2 and could not be ported without significant effort.

Figure 2.3: Unitree G1 Education Edition humanoid robot. [5]

2.4.2 Development Environment:

The G1 humanoid has been built on Ubuntu 20.04 with ROS2 Foxy and ROS1 Noetic installed providing a stable development platform. Python and C++ were used for implementation, which grants the benefit of large libraries and fast runtime.

2.5 Relevant Standards and Design Codes

2.5.1 ISO 10218-1:2025:

This standard [21] is designed for generic safety requirements of industrial robots. It describes hazards associated with these robots and provides requirements for the elimination or reduction of risks. To comply with this standard, joint velocity and acceleration limits were set to a maximum of 50% of the robots ability to comply with force limitation requirements. Also in the 2025 revision, the authors incorporated safety requirements for industrial robots intended for use in collaborative applications such as human interaction, formerly the content of ISO/TS 15066 [22], where speed monitoring is required during interaction. This was met through the web interface which allows speed monitoring.

2.5.2 ANSI/RIA R15.08-1-2020:

This standard [23] defines safety requirements specifically for industrial mobile robots in workplace environments. This standard addresses the differing capabilities of autonomous mobile robots. Key requirements from this standard include, path replanning for obstacle avoidance must not introduce new hazards, where this project has implemented A* and DWA path planning algorithms such that they have been validated to ensure that the paths given maintain a safety margin of at least 0.3m. Stopping distance must be within a reasonable distance for the given robot. For the G1, maximum velocity was limited to 0.5m/s to ensure that the stopping distance would be within 0.5m if an emergency stop would be needed. The standard also requires implementation of an emergency stop function and protective stop functions. For this software based emergency stops were implemented both automatically if an action is too extreme in which the robot will enter a dampening mode and shut off, and also through the web interface which will kill all commands to the robot.

2.6 Implementation Architecture

2.6.1 Human Pose Mapping Architecture

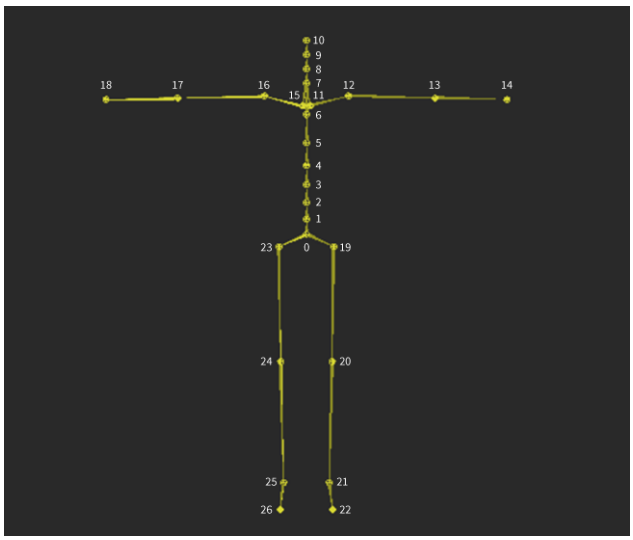
The human pose mapping system is built upon four primary components, the Mocopi interface, joint mapping, self collision detection, and G1 motor control.

The Mocopi interface receives UDP packets on port 12351 containing skeleton data for 27 body segments. A custom ROS2 node deserialises these packets and publishes equivalent transforms to the ROS2 tf2 tree at 100Hz polling rate. The implementation adapts an existing ROS1 package [24] with modifications for ROS2 compatibility.

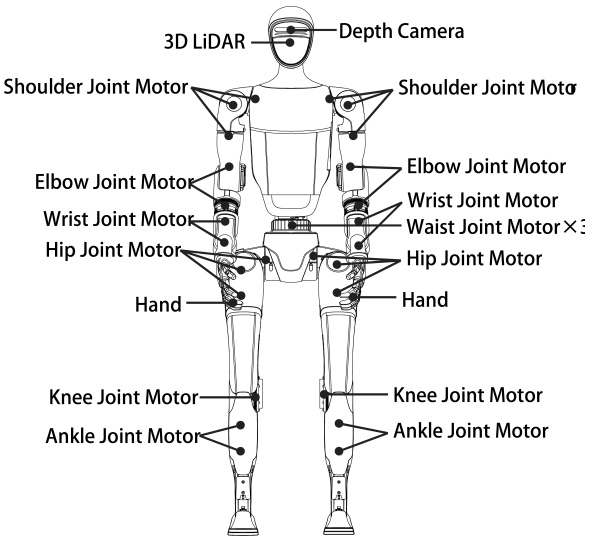
Joint mapping transforms the 27 segment skeleton to the G1's 29 joint structure through dictionary based mapping. Quaternion orientations undergo adjustments and are then broken down into roll, pitch and yaw. Then each of the Mocopi segments map to one or more robot joints with a designated component. Mapped angles are then published to `/joint_states`.

Collision detection employs the Flexible Collision Library [25] with robot meshes expanded 10cm for safety margins. The checker rejects unsafe poses and validated joint states publish to `/cleaned_joint_states`.

The G1 controller is based on Unitree's XR_Teleoperate [26] controller used for integration with the Apple Vision Pro [27] and has been adapted to use the Unitree SDK2 Python [28] to interface with the G1 via DDS protocol. Validated joint angles are converted to desired poses and passed to the kinematics solver which uses both Pinocchio [29] and CasADi [30] libraries for optimisation that minimises a cost function with translational error, rotational



(a) Mocopi 27 skeleton



(b) G1 29 degree-of-freedom joint configuration

Figure 2.4: Comparison of Mocopi skeleton [6] and Unitree G1 structure [7].

error, joint regularisation, and trajectory smoothing with weights of 50:1:0.02:0.1 respectively. Through the provided URDF file from Unitree [31] all motors are constrained within URDF joint limits. The solver typically converges within 50 iterations. A weighted moving average filter [0.4, 0.3, 0.2, 0.1] then smooths outputs. The G1 uses PD control for joint actuation and applies gains $K_p = 80$ N·m/rad, $K_d = 3$ N·m/(rad·s) for shoulder/elbow and $K_p = 40$ N·m/rad, $K_d = 1.5$ N·m/(rad·s) for wrists. A hardware deadman switch provides emergency stop capability. Detailed implementation specifications including packet deserialisation algorithms, coordinate transformation matrices, and collision detection parameters are provided in Appendix B.

2.6.2 SLAM and Localization Architecture

The SLAM and localisation system enables the G1 to build an accurate map of its surroundings by using its inbuilt MID-360 lidar to generate 3D point clouds. The implementation uses the FAST_LIO_LOCALIZATION_HUMANOID package [1], which integrates NDT and ICP algorithms for pose estimation and map alignment. This package was selected for its potential improvements convergence speed and accuracy on dense 3D point clouds.

A map was first recorded by walking the G1 around the area to generate a reference map of the main operating environment. During localisation, the system performs an initial rough alignment of the incoming lidar points and the reference map, providing a initial pose estimate. The NDT algorithm formulates the map as a collection of probability distributions to allow efficient point correspondence matching, and ICP subsequently refines the estimated transformation using iterative point matching. This hybrid approach could yield both strong convergence and accuracy.

FAST-LIO maintains a full 3D point cloud map as the base representation. However, to interface with the navigation stack, the 3D map is projected into a binary 2D occupancy grid with resolution of 0.05m set to ignore both the floor and roof, which serves as the global grid map for the A* planner. This preserves static obstacles while discarding excess noise. For real time navigation, live lidar scans are continuously processed into a smaller local map at 10Hz. This local map reflects dynamic obstacles that are not represented in the static SLAM map and is used by the DWA planner for collision avoidance.

2.6.3 Path Planning Architecture

The path planning system uses a global and local planner structure which allows for the use of A* advantages with DWA advantages to enable safe navigation through both static and dynamic obstacles. The A* global planner operates on the occupancy grid map generated by the SLAM and Localisation section, computing an optimal path from the robot's current position to the goal. This search has advantages over other searches such as Dijkstra's as it ensures an optimal solution without exploring the full space in most cases.

To join the global plan with local planner, the system implements a waypoint strategy that helps the robot to follow the global path closely. Rather than allowing the local planner unrestricted goal pursuit which could cause significant deviation from the optimal global path, intermediate waypoints are selected based on the robot's field of view. From the robots starting location, the algorithm projects forward to identify the next furthest visible point creating a intermediate goal, continuing this process until reaching the final goal. This visibility based segmentation forces the robot to follow along the globally optimal path with local obstacle avoidance capability, so that the DWA planner cannot take shortcuts that are locally promising but may cause the robot get stuck or be forced to backtrack.

The DWA local planner generates collision free paths to each waypoint by simulating robot motion over a prediction horizon. The planner samples velocities which have been set as follows maximum linear velocity 0.5 m/s, maximum angular velocity 1.5 rad/s, maximum acceleration 0.5 m/s^2, maximum angular acceleration 1.0 rad/s^2 and projects resulting paths over 2.0 seconds using 0.1 second time steps. Each candidate trajectory is evaluated using a cost function combining goal distance, velocity preference, and obstacle distance with weights of 1:0.1:1 respectively.

This allows real time response to dynamic obstacles. It balances a global optimal route with local dynamics so that the robot traces efficient routes but accommodates unexpected obstacles not included in the original map.

2.6.4 System Integration Architecture

The system integration provides a web based interface for monitoring and controlling the G1 robot. A lightweight Flask server [32] hosts a local web application running directly on the robot, ensuring access without external dependencies. The interface establishes a web socket connection for real time communication between the robot's ROS environment and its connected clients.

The web client uses ros3djs and roslibjs [3][2] libraries to visualise live ROS 1 and 2 topics, including lidar scans, robot pose, and navigation paths. Custom visualisations were implemented that display multiple robot states and information in one environment similar to RViz, allowing users to view SLAM data and localisation updates in real time.

The interface provides control endpoints to execute commands directly from the browser and enables full autonomy while maintaining a responsive, network accessible dashboard for supervision and testing. The website also incorporates an emergency stop button which immediately stops all operation on the robot.

2.7 Testing and Validation Procedures

A continuous testing method was used to measure the methods before physical deployment on the robot. Every component was initially tested in simulation by providing live sensor data as input and forwards outputs to the simulated robot instead of the real robot. RViz [33] also facilitated real time display of unit sensor data, robot state, and planned paths while

testing, allowing immediate feedback of system operation. After algorithms had been verified by simulation, testing moved to the physical robot. Beginning with testing of all emergency stops. Hardware testing progressed in three stages. First, controlled testing with permanent deadman switch control where outputs were compared against specified behaviour (5s> intervals). Second, semi-autonomous testing where short sequences (30s> intervals) were run with frequent pausing to compare behaviour with operators on stand-by at emergency stops. Finally, fully autonomous testing where the system ran full sequences (>1min intervals) under supervision with emergency controls available. This approach allowed the team to build confidence in the system gradually while keeping safety as the first priority.

2.8 Health and Safety Considerations

2.8.1 Development Safety:

All testing occurred in controlled environments with multiple personnel monitoring for intervention. Emergency stop systems were tested and verified before each operation session.

2.8.2 Operational Safety:

Maximum joint velocities were limited to prevent both damage to self, humans and items around. Comprehensive self collision detection prevents internal damage, while deadman switches stops motion upon controllers decision.

2.8.3 Software Safety:

All movement controllers implement deadman switches such that if anything would occur the controller could let go and all movement would stop instantly.

2.8.4 User Safety:

The web based interface includes operational status indicators and emergency stop buttons to prevent events.

3. Results and Discussion

3.1 Human Pose Mapping Results and Discussion

3.1.1 Overview:

This human pose mapping system is designed to be used to intuitively demonstrate actions and for the robot to repeat the action, reducing the expense and complexity of reprogramming. The system captures human motion data using Sony mocopi sensors. This data is decoded into a standard tf2 tree framework from ROS2. A dictionary based mapping algorithm then translates the Mocopi structure into the G1's structure through mapping joint angles. The resulting pose is first evaluated by collision detection if viable, the end effector position is sent to an inverse kinematics solver, which generates the low level commands transmitted to the robot via the Unitree SDK2 Python [28].

3.1.2 Experimental Setup

The system was evaluated through a series of trials where a human operator wearing Mocopi sensors performed various upper body poses and gestures and the robot attempted to copy them. Various poses were included in the test situations such as arms extended forward, overhead reaches and asymmetric positions.

Performance was assessed through visual comparison of the operator's pose and the robot's reproduced pose (Figure 3.1). Key evaluation criteria included accuracy, joint angle correspondence, system responsiveness, and self collision occurrence. Topic publishing frequencies were recorded to access system latency and command transmission rates (Table 3.1).

3.1.3 Results

The pose mapping system successfully tracked and copied human motions across a range of poses and gestures. Visual assessment confirmed that the robot accurately reproduced poses for the majority of tested configurations (Figure 3.1a-c).

The system maintained high transmission rates throughout testing, with the SDK interface publishing commands at 155Hz and cleaned joint states publishing at approximately 15Hz (Table 3.1). The standard deviation in joint state publishing, 0.48 seconds (Table 3.1) indicates some variation in the system which is likely due to differing computational efforts required for different poses.

Even with the high transmission rates, the robot still failed to reproduce movements in a sub 500ms response rate and measured responses between 1-2 seconds Table 3.2. Since the transmission rates are high, it is likely that a stack of movements are being stored and has to be worked through before the current movement is executed.

Table 3.1: Measured publish frequencies for ROS topics during pose mapping operation.

Topic	Mean (Hz)	Std Dev (s)
/arm_sdk	155.15	0.00317
/joint_states	32.87	0.00664
/joint_states_cleaned	15.06	0.48473

(a) (b) (c) (d) (e)

Figure 3.1: Different poses being mimicked by the human pose mapping system.

Table 3.2: System response time measurements between human motion input and robot execution.

Metric	**Response Time (s)**
Mean	1.58
Std Dev	0.49
Min	0.91
Max	2.28

No self collision events occurred throughout all testing scenarios. The collision detection system successfully prevented commands being sent to the robot that would cause the robot to collide with itself.

3.1.4 Discussion

The results show that the human pose mapping system can translate human motion into robot executable commands via the Mocopi sensors. The SDK publishing rate of 155Hz provides sufficient resolution for smooth motion reproduction. The 1-2 second response lag (Table 3.2) limits real time control ability. The difference between high transmission rates and slow response suggests that the command is buffering within the human pose mapping system, where motions accumulate faster than execution, causing the robot to replay historical rather than current poses.

The absence of self collisions shows that the collision detection strategy is effective and protects the robot from damaging itself. The 10cm safety margin has effectively prevented the robot from colliding with itself.

Some poses such as the pose in Figure 3.1d are difficult to reproduce due to the robot's inability to twist its torso. This is a fundamental problem with the robots structure that cannot be overcome. The system also struggled with poses near limits such as in the overhead reach where the shoulders could get stuck (Figure 3.1e). This is due to the controller approach which lacks the sophistication to handle movements from limits, resulting in problematic shoulder behavior during overhead reaches.

3.1.5 Limitations

Limitations the system's practical application. The absence of torso motion tracking limits pose reproduction accuracy for any motion involving body rotation or bending, restricting effective operation to poses where the human maintains an upright posture. However the main limitation is the response time of the robot (1-2 seconds Table 3.2) which increases difficultly of using the system in real world scenarios.

3.2 SLAM and Localisation Results and Discussion

3.2.1 Overview:

The system utilises the robot's in built lidar and the FAST_LIO_LOCALIZATION_HUMANOID package [1] to construct an environmental map. A reference map is built initially which is then used for comparison when localising. A hybrid NDT-ICP algorithm is used for a initial pose estimate and iteration which produces an accurate pose estimation. The resulting 3D map is then projected into a 2D occupancy grid for global path planning. While a separate frequently updated local map is used for real time obstacle avoidance. This process enables precise localisation and provides the dynamic obstacles for adaptive navigation.

3.2.2 Experimental Setup

The localisation system was evaluated through a series of tests in an indoor environment. A reference map was constructed and localisation was tested by placing the robot at known positions within the mapped area and measuring the system's ability to accurately determine its pose using the hybrid NDT-ICP approach.

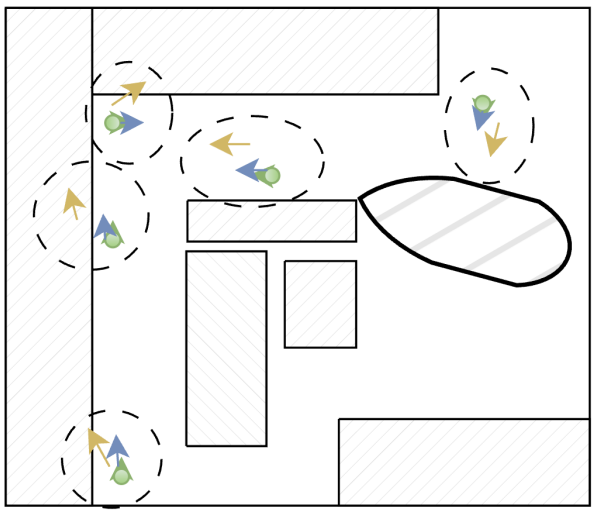


Figure 3.2: Example 5 samples of localisation data. Green: Actual Robot Position, Blue: NDT Initial Pose Estimate, Yellow: Final ICP after 30 seconds.

Table 3.3: Measured publish frequencies for ROS topics during navigation operation.

Topic	Mean (Hz)	Std Dev (s)
/local_occupancy_grid	9.99	0.00258
/cmd_vel	10.00	0.00033
/cloud_registered_1	10.00	0.00417

Localisation was assessed over five trials and were performed at various positions throughout the environment, with real positions measured using a measuring tape. For each test, the system first estimated an initial pose with the NDT algorithm, it is then refined using ICP over 30 seconds. Localisation was assessed by measuring the distance between the estimated pose and the known position. Measurements were recorded for both the initial estimate done by NDT and the final ICP improved pose. Topic publishing rates were monitored to verify real time operation capability (Table 3.3).

3.2.3 Results

The SLAM system successfully constructed detailed environmental maps during initial mapping phases, demonstrating effective sensor fusion and point cloud processing. The generated maps exhibited accuracy for static environmental features, providing a reference for future localisation operations (Figure 3.3).

Table 3.4: Localisation for NDT initial estimates and ICP refined poses after 30 seconds of refinement. Distance represents error.

Trial	NDT Initial Error (cm)	ICP Final Error (cm)
1	<15	126
3	<15	47
4	<15	30
2	<15	125
5	<30	39

The NDT-ICP localisation system demonstrated accurate initial estimation with all five trials achieved initial estimates within 30 cm of the actual robot position (Table 3.4). This rapid initial convergence shows NDT's effectiveness for localisation in environments with defining features.

The topic publishing frequencies (Table 3.3) confirmed that the system maintained consistent 10Hz operation for critical topics, demonstrating that the system is reliable and has real time performance ability.

3.2.4 Discussion

The results show a large difference between the performance of the NDT and ICP algorithms of the localisation system. The NDT algorithm's initial estimate demonstrates its effectiveness, making use of its probability distribution representation of the reference map for precise pose estimation without iteration.

The decline in estimation accuracy during ICP refinement is opposite to the algorithm's intent and suggests issues with the iterative matching process. ICP convergence depends on similar or matching points between the sensor and the reference map. The observed divergence suggests that changes have occurred to the reference map between recording of the map and the localisation trials. Rearrangement of the laboratory space, including furniture repositioning or the introduction of new obstacles, could create mismatches that could cause ICP to converge toward incorrect alignments.

The navigation topics were publishing at a rate of 10Hz which indicates that computational loads remained manageable and was not the cause of the pose divergence but rather by the behaviour of the algorithm in the presence environment discrepancies.

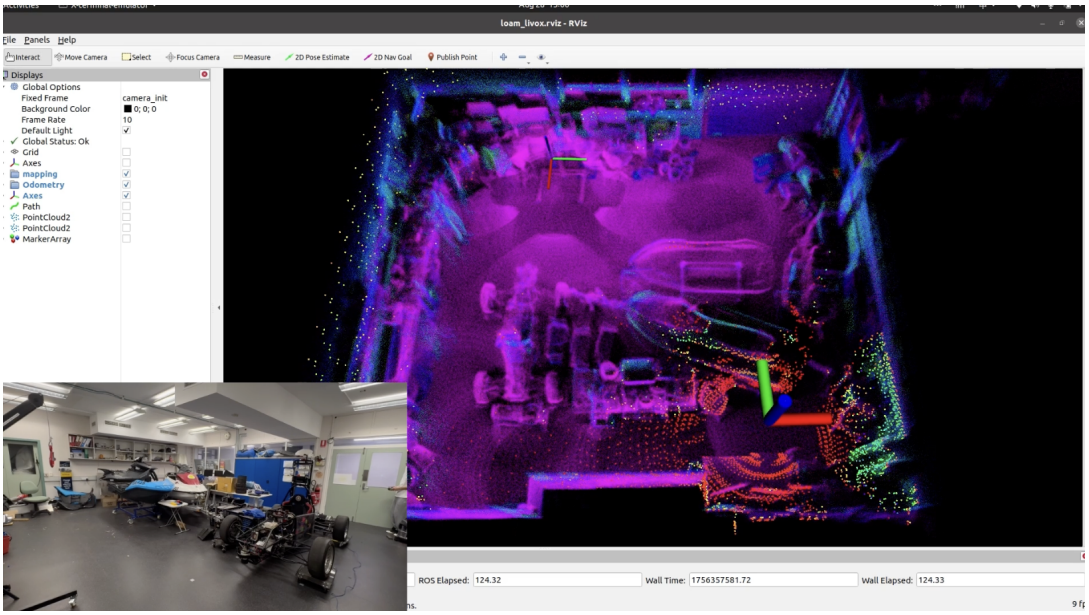


Figure 3.3: SLAM map built from FAST_LIO_LOCALIZATION_HUMANOID [1] on the Unitree G1, recorded and filmed by PHD student Hongtao Zhang[8]

3.2.5 Limitations

The largest limitation of the navigation system was the sensitivity to environmental changes between reference map generation and deployment. The ICP refinement process consistently diverged when faced with differences between reference map and live data, which shows that

the system lacks robustness for frequently updated environments. This severely constrains deployment, as maintaining perfectly static environments is unrealistic for the majority of scenarios. Future work should investigate adaptive localisation techniques that could integrate map updating mechanisms to maintain reference map accuracy.

3.3 Path Planning Results and Discussion

3.3.1 Overview:

This system uses a global A* algorithm with a local DWA algorithm. The global A* planner calculates an optimal path on the generated SLAM map, while a waypoint strategy ensures the robot adheres to this route. The local DWA planner then generates safe velocity commands by simulating paths that are limited by the robot's constraints. This structure allows efficient navigation towards a goal while dynamically avoiding obstacles.

3.3.2 Experimental Setup

The path planning system was evaluated through a series of navigation tests in an indoor environment with various obstacles. Multiple goal positions were manually set at various positions each requiring navigation through spaces with static and dynamic obstacles, narrow passages, and confined corners. Each trial began with the robot generating a global path using A* on the SLAM generated occupancy grid, followed by autonomous navigation using DWA for local trajectory planning.

Performance was assessed through observation of the navigation behaviour, with particular attention paid to path characteristics, behaviour, and failure positions. The robot's pose, generated paths, and velocity commands were recorded for post analysis. Scenarios targeted challenging and general scenarios that the robot would be expected to navigate.

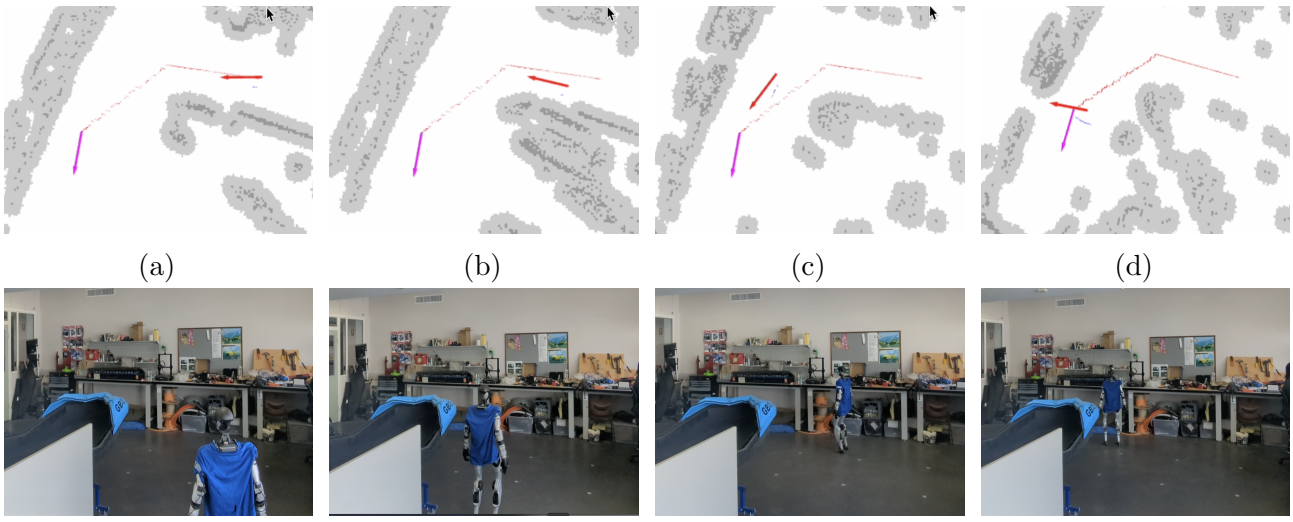


Figure 3.4: Comparison of navigation between web dashboard visualisation and real world execution. The sequence shows the robot navigating from its starting position to a designated goal.

3.3.3 Results

The path planning system successfully navigates to its desired goal in the majority of test scenarios. Showing its effective integration between the global and local planners to navigate its environment. The global A* planner generates optimal paths to the desired final goal. This distance optimal approach typically positioned the planned path 0.2–0.3 meters from detected objects. This maximises path efficiency while maintaining safety margins (Figure 3.4).

During execution, the local DWA planner exhibited velocity optimising behaviour. In open corridors and unobstructed spaces, the robot planned smooth paths approaching the configured maximum linear velocity of 0.5 m/s. However, when executing turning manoeuvres, the planner showed a preference for wide turning radius, often producing a radius of 1.0–2.0 meters radius even when the environment permitted tighter turns. This behaviour reflects DWA's cost function bias toward maintaining higher forward velocities, which in turn penalises sharp turns that require velocity reduction.

A significant implementation issue affected the robot when executing a path movement. Unitree's high level controller API would timeout after short motion segments, causing the robot to execute a stop start motion rather than smooth continuous movement. The robot would typically execute 2-3 steps before stopping, requiring the robot to wait for the API to return a timeout error (typically 5 seconds) before sending new commands, resulting in jerky navigation behaviour throughout all attempts.

Performance problems were observed in constrained environments. In tight corners or narrow passages with clearance below approximately 0.6 meters, the planner frequently failed to generate forward paths. In these scenarios, the robot exhibited failure mode of spinning. Several trials resulted in the robot getting stuck and being unable to move towards the goal unless manual intervention was provided.

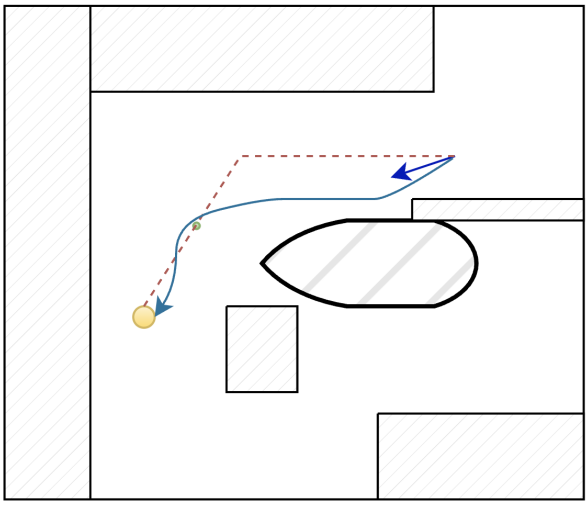


Figure 3.5: Diagram of an example path planning response. Red: Global Path, Blue: Local Path Followed, Green Arrow: Initial Position, Yellow Circle: Final Goal, Small Green Circle: Intermediate Goal.

3.3.4 Discussion

The results highlighted a trade off when using a split A*, DWA architecture between optimality and robustness. The global planners distance minimisation to optimise the path taken by the

local planner effectively exploits the known map structure to generate these efficient paths, consistent with A*'s objective. This optimisation positions the planned optimal path close to obstacles boundaries, reducing the available options of the local planner for obstacle avoidance and dynamic replanning. This design choice prioritises path efficiency over opportunity, which proved problematic in constrained environments where greater manoeuvrability was required.

The DWA's cost function successfully promotes efficient motion in open spaces by favouring paths that maintain high forward speeds towards the goal and by avoiding ones close to objects. This characteristic however often produces unnecessarily wide turning behaviours that reduces agility. The discrepancy between the global planner's tight corners and the local planner's wide arcs indicates a mismatch between the global and local planners objectives.

The corner behaviour represents a documented limitation of local planning algorithms operating without global context awareness [34]. When surrounded by obstacles on multiple sides, DWA's prediction horizon of 2 seconds prevents the finding of strategies that would enable escape. The observed spinning behaviour indicates the planner's attempt to reorient towards more promising heading angles, but without the capability to execute reverse motion, this strategy fails when all forward paths remain obstructed. Integration of recovery behaviours such as backward motion planning would likely improve performance in these failure scenarios.

3.3.5 Limitations

Several limitations have been identified during testing that constrain system performance. The global planner's path optimisation causes the planner to generate paths with insufficient clearance to obstacles, limiting the local planner's reactive capability. The DWA cost function's preference for velocity maximisation produces wide turning radius' that reduce agility. Most critically, the system lacks any recovery behaviour, making it capable of local minima in tight corners.

3.4 System Integration Results and Discussion

3.4.1 Overview:

This system uses integrated web based interface for monitoring and controlling the G1 robot, enabling remote supervision and interaction. Hosted directly on the robot via a lightweight Flask server [32], the system establishes a web socket link to the robot's ROS environment. The client-side application, built with roslibjs and ros3djs [2][3], visualises live data in a unified dashboard similar to RViz [33]. Beyond visualisation, the interface provides endpoints for command execution, and has safety features like an emergency stop.

3.4.2 Results:

The interface does provide a functional and responsive dashboard for robot operation. The performance metrics measured demonstrate the system's efficiency. The 3D visualisation client updated at a rate of 120Hz while topic data updated at a rate of 10-20Hz depending on type (Table 3.6). This effectively allows live scans and robot transforms to be streamed without significant lag. Hosting the server locally on the robot resulted in a negligible CPU load increase of approximately 0.21% (Table 3.5), confirming the lightweight nature of the architecture.

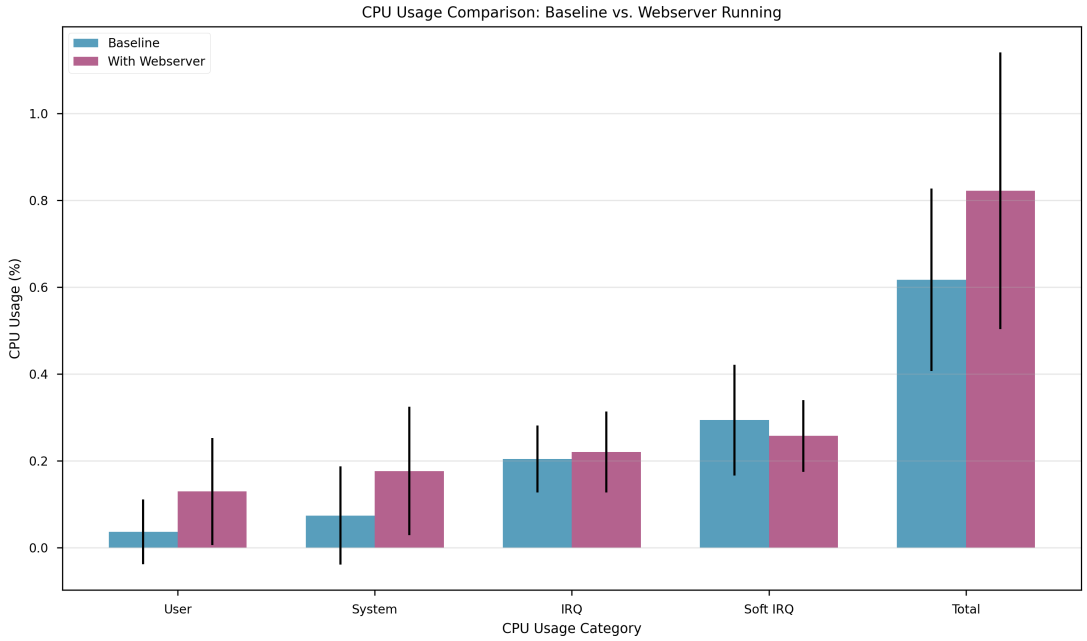


Figure 3.6: CPU usage comparison between baseline operation and when the web server is active.

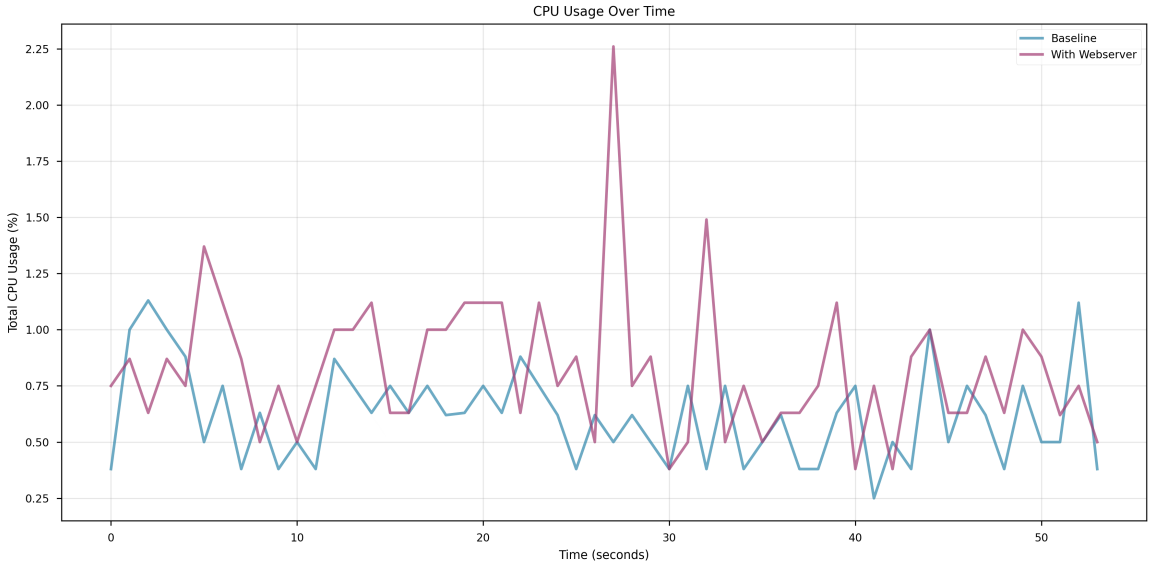


Figure 3.7: Total CPU usage over time comparing baseline operation (blue) and web server operation (purple).

Table 3.5: Comparison between CPU usage between baseline and web server operation.

Condition	Mean CPU (%)	Std Dev (%)	Max CPU (%)	Min CPU (%)
Baseline	0.62	0.21	1.13	0.25
With Web Server	0.82	0.32	2.26	0.38
Overhead	0.21	0.38	1.13	0.13

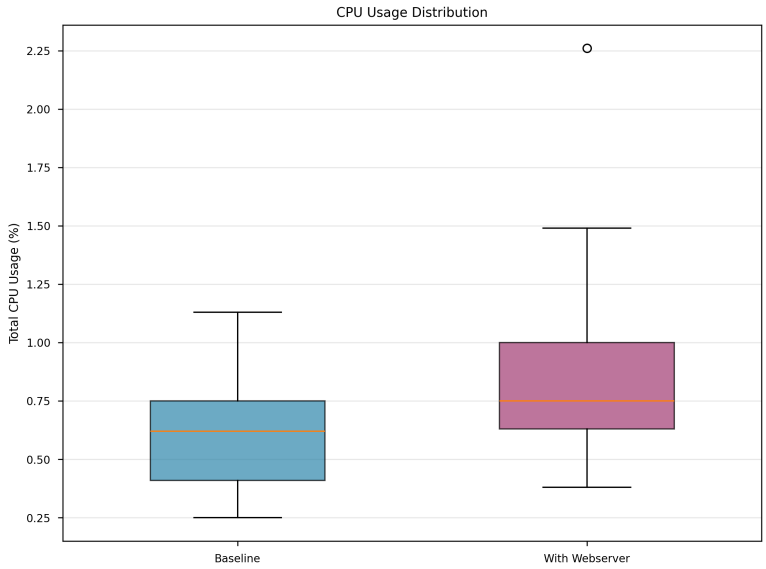


Figure 3.8: Box plot comparing the distribution of total CPU usage between baseline and web server operation.

Table 3.6: Measured publish frequencies for ROS topics during web server operation. Topics with larger standard deviations tended to have inactive portions during the test where frequency would drop to 0.

Topic	Mean (Hz)	Std Dev (Hz)	Max (Hz)
/local_occupancy_grid	19.8	3.27	22
/local_path	14.1	9.43	22
/cloud_registered_1	9.9	1.66	11
/cmd_vel	7.0	4.77	12

3.4.3 Discussion:

The web interface was effective in its objective of providing a usable and accessible dashboard and which has been confirmed by the results. Low latency and low CPU usage validate the choice of using a lightweight web server, which has been determined to be reliable for data transfer. A significant benefit of this approach is that it is independent, by running everything locally the system operates without external dependencies, which makes it well suited for deployment in the field. The integration of roslibjs [2] provides a bridge between the browser and ROS, demonstrating that complex robotics data can be effectively visualised and interacted with through a web client, increasing accessibility compared to desktop or terminal tools like RViz [33] especially for non technical users where it may be most useful.

3.4.4 Limitations:

A primary limitation for this implementation was the inability to integrate the Pinocchio library [29] which is used in the Human Pose Mapping implementation due to compilation failures on the robot's ARM architecture, for which no compatible compiled binaries existed. This prevented the implementation of planned human pose control features directly through the web interface. Future work could involve cross-compiling or containerising the application use a service such as Docker [35] to resolve this dependency. As the current visualisation stands, while functional lacks the extensive plugin ecosystem of other more complete visualisers such as RViz.

4. Conclusions and Future Work

This project successfully integrated multiple systems to create a autonomous navigation and teleoperation platform for the G1 robot. Four subsystems were developed and tested including human pose mapping, SLAM and localisation, path planning, and a web based interface for remote monitoring and control.

4.1 Human Pose Mapping

The human pose mapping system demonstrated effective translation of human motion into robot commands using Mocopi sensors, achieving high transmission rates with zero self collision incidents. However, the system had a response latency of 1.58 ± 0.49 seconds which limits real time ability. The lack of torso motion movement of the G1 also limits pose reproduction to upright poses only.

The issue of command buffering leading to a 1-2 second response lag needs investigation in each of the major components of the system including the SDK to help identify queue management problems or implement a command priority structure. The inverse kinematic solver requires improvement with more effective constraint avoidance to avoid shoulder joint problems during overhead reaches. Alternative motion capture systems with higher update rates could potentially reduce latency further.

4.2 SLAM and Localisation

The SLAM and localisation system successfully constructed environmental maps and achieved highly accurate initial pose estimation through the NDT algorithm. The ICP refinement process still leaves much to be desired as it tended to diverge from the known position. This counter intuitive result has been attributed to changes in environment between when the reference map was constructed and operating time.

The system could benefit from an adaptive localisation mechanisms which could handle changes between map recording and deployment. Implementation of convergence monitoring to stop diverging ICP iterations could help prevent the observed accuracy loss. Reference automatic map updating capabilities could also help maintain an accurate map over large periods. Investigation of alternative localisation approaches may provide increased robustness to changing environments.

4.3 Path Planning

Global A* path planning with local DWA path planning was successfully implemented into the navigation system allowing the robot to navigate successfully to designated goals in the majority of tests. The global planner generated optimal paths, while the local planner maintained smooth paths at velocities approaching 0.5m/s in open spaces. The local planner did suffer when encountering environments below 0.6 meters clearance which would cause the robot to get stuck. Additionally, a critical implementation issue with the Unitree high level API which caused a stop start motion patterns with 5 second timeout delays, preventing smooth continuous navigation.

The most significant enhancement is fixing of the high level API timeout issue, which may require low level control integration or alternative API implementations. Recovery behaviours

including backward motion planning and rotation-in-place strategies could improve robustness in constrained spaces. Implementation of replanning triggered by deviation from the global path could improve adaptability.

4.4 System Integration

The web based interface successfully provides a responsive remote monitoring and control capabilities with minimal computational load on the robot system. The system maintained 120Hz visualisation update rates while introducing only 0.21% CPU overhead. Hosting the web server on the robot eliminates external dependencies, making the system suited for in the field deployment. The integration of ros3djs and roslibjs libraries [3] [2] show that robotics visualisation can be effectively delivered through web browsers, this can significantly improve accessibility compared to desktop tools such as RViz [33].

Future work on system integration could include the integration of the Pinocchio library [29] which would allow direct human pose control through the dashboard. To do this it would require the ARM compilation issues to be fixed, potentially through cross compilation or containerisation. Extension of the visualisation to match capabilities of other systems would also enhance utility.

Overall, the project objectives were mostly achieved, the integrated system demonstrated functional autonomous navigation, intuitive teleoperation, and accessible remote monitoring capabilities. Several critical limitations however prevent immediate real world deployment such as the pose mapping response latency which hinders real time control, the localisation system lacks robustness to changes in environment, the navigation system needs additional recovery behaviours that allow better movement in constrained spaces, and the high level API timeout issue prevents smooth motion execution. These limitations represent implementation challenges rather than fundamental flaws, which provide clear direction for future development.

5. Bibliography

[1] Xu, Yihao and Li, Xin and Zhao, Zhongxia and Zhou, Gaohao and Li, Dongliang and An, Xiang and Tan, Huajie and Feng, Ziyong, "FAST_LIO_LOCALIZATION_HUMANOID," 2025. Accessed: October 2025.

[2] Robot Web Tools, "roslibjs: Javascript library for ros websocket communication," 2025. Accessed: October 2025.

[3] Robot Web Tools, "ros3djs: 3d visualization library for ros in the browser," 2025. Accessed: October 2025.

[4] Yanko Design, "Sony mocopi wearable sensors let you control avatars with your whole body," December 2022. Accessed: October 4, 2025.

[5] Unitree Robotics, "Unitree g1 education edition programmable humanoid robot for scientific–educational use," 2025. Accessed: October 6, 2025.

[6] Sony Corporation, "mocopi technical specifications," 2023. Accessed: October 2025.

[7] Unitree Robotics, *Unitree G1 Humanoid Robot User Manual*. Unitree Robotics Co., Ltd., 2024. Accessed: October 2025.

[8] H. Zhang, "Fast lio localization humanoid recording." 2025.

[9] B. Zhang, J. Moh, and H.-O. Lim, "Research on automatic navigation for a bipedal humanoid robot," *2021 IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2021.

[10] Y. Xiao, G. Zhang, J. Hu, Y. Zhou, H. Fan, and S. Zhang, "Indoor mapping and analysis based on optimized existing 2d slam algorithm," *2024 3rd International Conference on Robotics, Artificial Intelligence and Intelligent Control (RAIIC)*, 2024.

[11] I. Hassanzadeh, K. Madani, and M. A. Badamchizadeh, "Mobile robot path planning based on shuffled frog leaping optimization algorithm," *2010 IEEE International Conference on Automation Science and Engineering*, 2010.

[12] Sony Corporation, "mocopi: Motion capture system design gallery," 2023. Accessed: October 4, 2025.

[13] A. Vedadi, A. Yousefi-Koma, P. Yazdankhah, and A. Mozayyan, "Comparative evaluation of rgb-d slam methods for humanoid robot localization and mapping," *IEEE Transactions on Robotics*, 2024.

[14] M. . Company, "Making healthcare more affordable through scalable automation," September 2020. Accessed September 27, 2025.

[15] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multi-map slam," in *IEEE Transactions on Robotics*, vol. 37, pp. 1874–1890, 2021.

[16] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual slam library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[17] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," in *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

[18] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016.

[19] S. Macenski, F. Martín, R. White, and J. Ginés Clavero, "Slam toolbox: Slam for the dynamic world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5899–5906, 2020.

[20] Google, "Cartographer," 2023. Accessed: 2025-10-01.

[21] International Organization for Standardization, "ISO 10218-1:2025 Robots and robotic devices – Safety requirements for industrial robots – Part 1: Robots," 2025. Accessed October 1, 2025.

[22] International Organization for Standardization, "Robots and robotic devices — Collaborative robots," 2025. Accessed October 1, 2025.

[23] Robotic Industries Association, "ANSI/RIA R15.08-1-2020: Industrial Mobile Robots – Safety Requirements – Part 1: Requirements for the Industrial Mobile Robot," 2020. Accessed October 1, 2025.

[24] Hello World Lab, "mocopi_ros: Ros interface for sony mocopi motion capture system," 2023. Accessed: October 4, 2025.

[25] F. C. Library, "Flexible collision library (fcl)," 2025. Accessed: 2025-05-06.

[26] Unitree, "XR_Teleoperate," 2024. Accessed: October 2025.

[27] Apple Inc., "Apple Vision Pro," 2024. Accessed: October 2025.

[28] U. Robotics, "Unitree sdk2 python," 2025. Accessed: 2025-10-05.

[29] S. of Tasks, "Pinocchio," 2025. Accessed: 2025-10-05.

[30] Casadi, "Casadi," 2025. Accessed: 2025-10-05.

[31] U. Robotics, "Unitree g1 urdf," 2025. Accessed: 2025-10-06.

[32] Flask, "Flask," 2025. Accessed: October 2025.

[33] Dave Hershberger, David Gossow, Josh Faust, William Woodall, Robert Haschke, "Rviz," 2025. Accessed October 1, 2025.

[34] Fox, D. and Burgard, W. and Thrun, S., "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, 1997.

[35] Docker, "Docker," 2025. Accessed: October 2025.

[36] W. Song, X. Guo, F. Jiang, S. Yang, G. Jiang, and Y. Shi, "Teleoperation humanoid robot control system based on kinect sensor," *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2012.

[37] Y. Huang and S. Guo, "Path planning of mobile robots based on improved a* algorithm," *2022 Asia Conference on Advanced Robotics, Automation, and Control Engineering (ARACE)*, 2022.

[38] E.-J. Rolley-Parnell, D. Kanoulas, A. Laurenzi, B. Delhaisse, L. Rozo, D. G. Caldwell, and N. G. Tsagarakis, "Bi-manual articulated robot teleoperation using an external rgb-d range sensor," *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018.

[39] M. Çoban and G. Gelen, "Wireless teleoperation of an industrial robot by using myo arm band," *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 2018.

[40] Z. Cheng, B. Li, and B. Liu, "Research on path planning of mobile robot based on dynamic environment," *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2022.

Appendix A. Literature Review

Humanoid robotics has been rapidly advancing in recent years. However, the ability for humanoid robots to traverse complex real world environments is still limited. This literature review will focus on the current state of humanoid robotics, in the fields of human pose mapping, SLAM and path planning, to identify the current limitations of existing implementations. The focus of this project is to develop a humanoid robot that is capable enough to perform in dynamic environments, by combining human pose mapping and SLAM. Looking at the insights from Zhang et al. (2021) on visual SLAM [9], Song et al. (2012) on teleoperation of humanoid robots using Kinect sensors [36], and the improvements of the A* algorithm by Huang and Guo (2022) [37], this literature review argues for a system with both integrated human pose mapping and SLAM navigation to address limitations in previous implementations.

Current work in SLAM has been focused on improving the accuracy and efficiency of SLAM algorithms and alternative methods of SLAM. Zhang et al. (2021) [9] has proposed a visual method of SLAM. Zhang has proposed this as an alternative that potentially could have successful results with dynamic obstacles. Zhang's shows visual SLAM's potential for dynamic detection, showing via simulation the obstacle avoidance and path generation that visual SLAM is capable of when presented with an obstacle. However, Zhang's work struggled with localisation of the surrounding environment with fragmented walls in the map built but their algorithm seen below in Figure A.1.

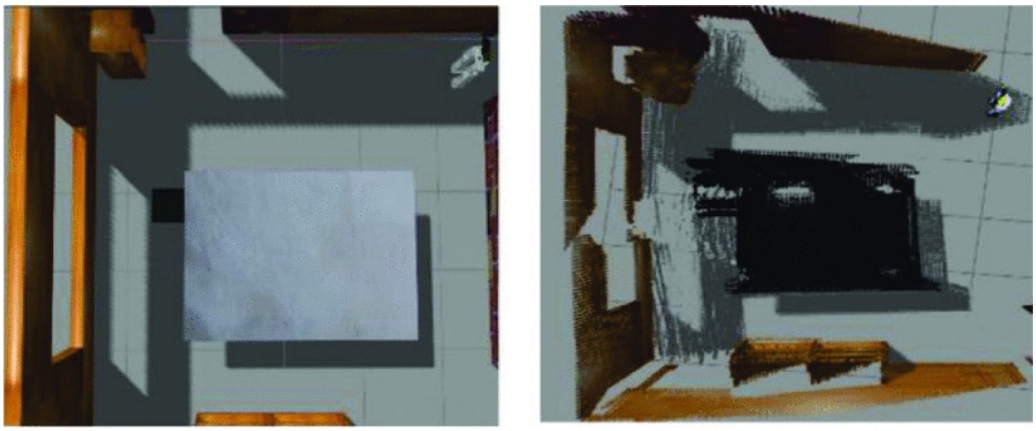


Figure A.1: Built map from Zhang's visual SLAM technique. **Source:** Zhang et al. (2021) [9].

Unlike Zhang's approach, Xiao et al. (2022) builds upon Google's Cartographer, a 2D lidar based SLAM library, optimising it for indoor environments by reducing computationally heavy tasks through key improvements [10]. As demonstrated by Figure A.2, their method shows promise in performance, Xiao has proposed that when IMU data is reliable, by disabling the computationally expensive CRM (Correlative Scan Matching), it frees resources for thresholds. Xiao continues, in the global scale by reducing the minimum score threshold to maintain map coherence, then by lowering the sampling rate of submap features and removing overlapping submaps it also reduces the computational cost of SLAM.

Xiao's approach's computational efficiency could allow for better real time performance for humanoid robots with limited onboard processing, however unlike Zhang's approach it cannot handle dynamic obstacles. For navigation in complex environments, integrating visual

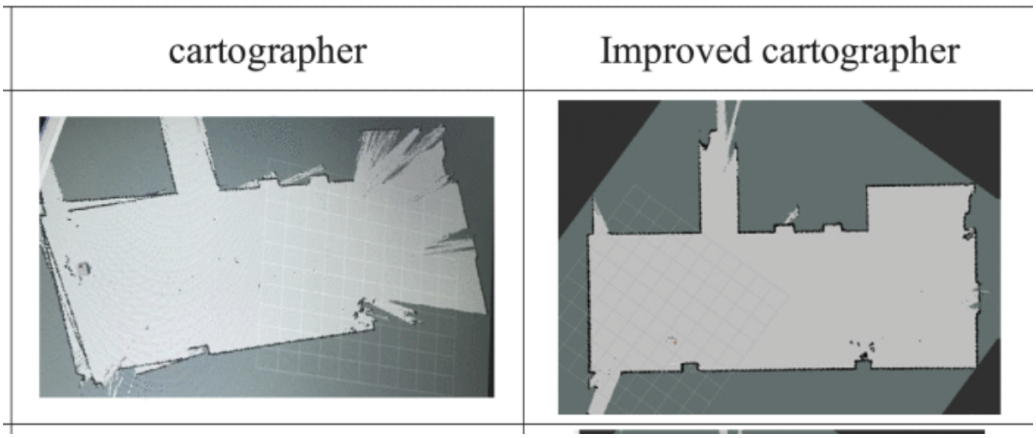


Figure A.2: Built map from Xiao's optimised 2D SLAM. **Source:** Adapted from Xiao et al. (2024) [10].

SLAM's dynamic object detection with lidar's precision mapping appears promising. This hybrid approach could address the localisation limitations observed in Zhang's work while maintaining its dynamic obstacle detection and avoidance through Xiao's optimisation techniques.

Various technologies have been developed for human teleoperation of manipulators, ranging from cost effective single sensor solutions to advanced multi sensors systems. Song et al. 2012 [36] proposed a Kinect based system which was designed on the idea of providing a low cost single sensor solution to teleoperation of manipulators. The Microsoft Kinect is an accessible sensor system with integrated depth sensor and skeleton tracking algorithms which Song leveraged providing a potential solution for affordable teleoperation. Song had a degree of success, successfully estimating human pose accurately and in real time using the Kinect, however, Song's work was limited to simulation only and did not implement a robot implementation. Building on top of this Rolley-Parnell et al., 2018 [38] used a robust and economical RGB-D sensor to achieve a physical implementation. Rolley-Parnell's method combined a 2D pose estimation from RGB images with a depth map. Unlike Song, Rolley-Parnell implemented real world demonstrations of their teleoperation, on their robot platform, CENTAURO. Alternatively to vision based systems, Coban and Gelen, 2018 [39] implemented a sensor based version using a Myo armband, leveraging its built in gyroscopic and electromyography sensors on a 6 degree of freedom industrial robot arm. Coban and Gelen's method used gestures and motions captured by the Myo armband to map onto the robotic arm to control it. While functionally viable, the proposed method lacked significant accuracy, causing the operators to consistently overshoot desired positions due sensor drift and EMG precision. This highlights key challenges in a wearable sensor based teleoperation.

Path planning research has long been looking for the balance between computational efficiency with solution accuracy. This is a key challenge with humanoid robots with their onboard computational limitation. Huang and Guo (2022) [37] proposed a variant of A* that claims to have addressed these computational limits. Huang and Guo have done this by introducing a state list which stores intermediate node computational cost for future use. Their improved A* has boasted a 99.5% performance enhancement over traditional A* on a raster map that is 20x20. While this reduction is impressive, the small sample map may not show its true performance in larger environments. While classical algorithms have been conventionally used in mobile robots, recently researchers have been expanding using the gains of AI and more complex algorithms into the world of path planning. Such as Hassanzadeh et al. (2010) [11] who has proposed a shuffling frog leaping algorithm, which diverges from traditional algorithms. Hassanzadeh's

algorithm uses a genetic algorithm, creating a population of "frogs" divided into multiple sets of cultures. Each of these cultures, are assigned to optimising local map regions, by adjusting positions based on maximising the value function. Hassanzadeh then provides an example in which the algorithm was tasked to navigate a map with six static obstacles, which it successfully created a path for seen below in Figure A.3 . However, this form of navigation requires many iterations to achieve a result. This shows its potential to have real time limitations when applied to real world environments. Cheng et at. (2022) [40] further looked into the capability of classical algorithms from a dynamic obstacle and laser SLAM perspective, comparing algorithms for global path planning and investigating the integration of the Dynamic Window Approach (DWA) and Bug2 to obtain autonomous function and real time autonomous obstacle avoidance capability. Cheng first compares the global path planning algorithms Dijkstra's and A*, from a laser SLAM map they found that the A* algorithm, due to its heuristic explored 75% less nodes than Dijkstra's and had a 91.4% reduction in computation time. This shows while Dijkstra's always calculates the best possible path, is too computationally expensive to use in real time control of humanoid robots, and by adapting its counter part, A* into humanoid robotics it should be able to run in real time. Cheng then goes on, once the global path has been calculated a local obstacle algorithm must be adapted such that the robot can adapt to dynamic obstacles. For this Cheng has suggested the use of DWA which uses the obstacles kinematics to avoid them. This algorithm falls apart in small spaces however, so Cheng has offered the improvement of the Bug2 algorithm which is a simple wall following algorithm in spaces that are deemed to small. This improvement while simple seems to be a quite robust solution as Bug2 is adaptable and can run in real time, according to Cheng's experiments, they also had success combining Bug2 into the DWA.

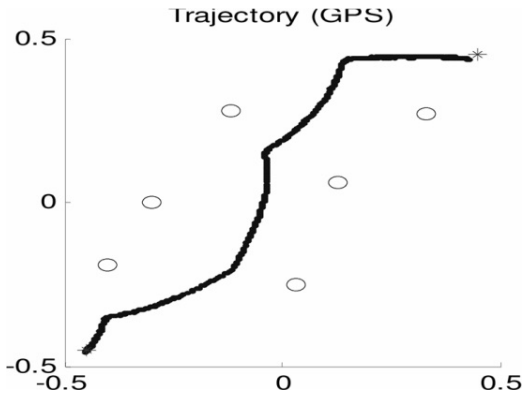


Figure A.3: Hassanzadeh's Frog Algorithm Path Plan. **Source:** Hassanzadeh et al. (2010) [11].

This review has examined critical research in regards to this project, covering the topics of SLAM, human pose mapping and path planning. Zhang et al. (2021) [9] proposed visual SLAM led promising research in the ability to detect and plan around dynamic obstacles, however had localisation problems, that Xiao et al's (2022)[10] laser SLAM solved but could not handle dynamic obstacles, potentially alluding to investigation into hybrid SLAM systems. Like SLAM, human pose mapping also highlighted the trade off between various sensor systems. Such as Song et al's (2012) [36] Kinect based simulations and Rolley-Parnell et al's (2022) [38] use of visual based system, which were both accurate and affordable but lacked the ability to be used in real world implementations due to the limitations of a camera requiring you to be in frame. This was addressed by Coban and Gelen's (2018) [39] use of a wearable sensor, the Myo arm band which contained multiple sensors to control the robot, but however lacked the precision and accuracy of visual systems. In path planning, strides have been made in the improvement of classical algorithms for new problems. Such as Huang and Guo's (2022) [37] proposed improvement to

A*, suggesting an improvement of 99.5% faster than the classical A*, however lacks validation in large maps. Hassanzadeh et al. (2010) [11] suggested an alternative approach, using a genetic algorithm for path generation, but fell short in handling dynamic obstacles due to its requirement on iterative calculations. Path planning in laser SLAM specifically was researched by Cheng et at. (2022) [40], who found that A* showed promise in global path planning, then used a combination of DWA and the Bug2 algorithm to bridge the gap between global path and a local path for dynamic obstacles. This research shows the need for further development in the field of humanoid robots to be able to effectively operate in dynamic environments.

Appendix B. Human Pose Mapping Implementation

B.0.1 Mocopi ROS2 Interface

The Mocopi motion capture system is designed to interface into Sony's dedicated phone app which handles the Bluetooth connection and calibration of the sensors. The app also allows the forwarding of the data to a designated port on an IP address through the User Datagram Protocol (UDP) packets, through this we can take advantage of that and pass the data from the mobile application to the robots PC.

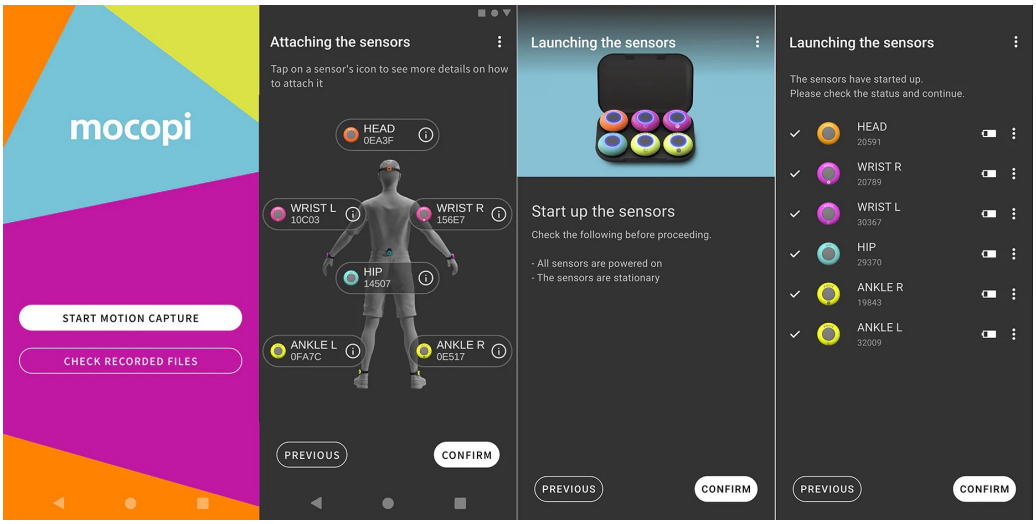


Figure B.1: Mocopi mobile application interface [12].

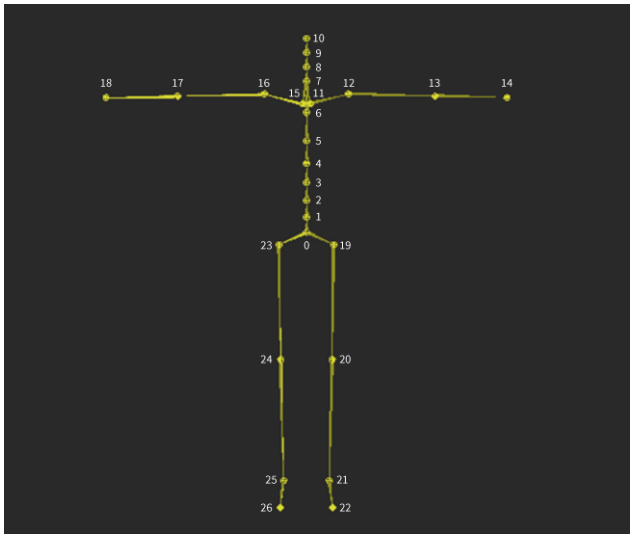
The UDP packets transmitted by the Mocopi application contain proprietary binary encoded skeletal tracking data where each section of the body is a designated 'bone'. A custom ROS2 node was developed to receive these transmissions on port 12351, deserialise the binary format, and publish the pose information to the ROS2 transform tree. The implementation was adapted from an existing ROS1 package by Hello World Lab [24] which was made to handle the Mocopi data to ROS1 but required modifications to be compatible ROS2. These changes includes the transition from rospy to rclpy, updated transform broadcasting using tf2_ros, and adherence to ROS2 lifecycle patterns.

The deserialisation process implements a recursive parser to handle the nested structure of Mocopi packets. Each packet contains a header section identifying packet type and frame data. The frame data provides a 'bone' identifier, parent reference, and its transform data which includes a translation (x, y, z) and rotation quaternion (x, y, z, w) for each of its 27 tracked body segments. Using this child-parent structure we can transform it into the ROS2 Transform Tree which uses a similar child-parent structure.

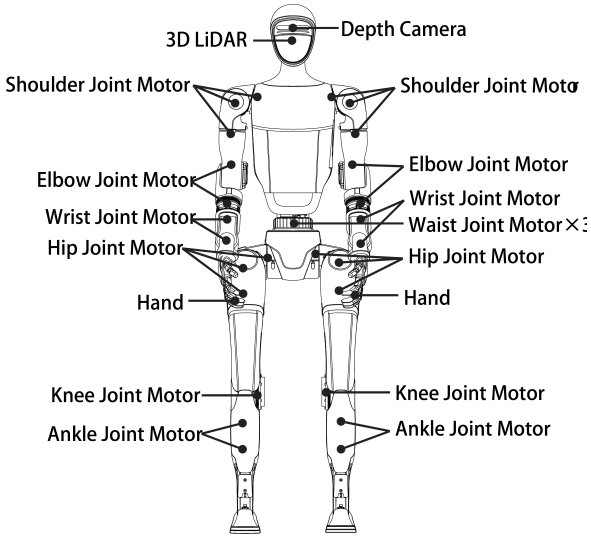
The receiver node operates asynchronously polling data at 100Hz, invoking a callback to poll the UDP socket with a 2048-byte buffer. This polling rate ensures that the transmission rate between the Mocopi System and the PC will not a be a bottleneck and will keep natural movement. Exception handling manages malformed packets or incomplete data, logging errors without disrupting continuous operation. For joint mapping and robot control, the published transform tree provides live human pose data suitable processing.

B.0.2 Joint Mapping Between Mocopi Skeleton and G1 Skeleton

The Mocopi skeleton and G1 structure are significantly different both dimensionally and topologically which requires careful joint mapping to correctly map the right data to the robot. The Mocopi system tracks 27 body segments representing a simplified human skeletal structure (Figure B.2a), while the G1 possesses 29 motors distributed across the torso (3 joints), arms (7 joints each), and legs (6 joints each) (Figure B.2b). This dimensional mismatch necessitates both direct joint mapping where appropriate and inference for joints not directly measurable by the Mocopi system.



(a) Mocopi 27-bone skeletal hierarchy



(b) G1 29 degree-of-freedom joint configuration

Figure B.2: Comparison of Mocopi skeletal model [6] and Unitree G1 kinematic structure [7], illustrating the topological differences requiring algorithmic joint mapping.

The mapping algorithm receives quaternion orientation data from the Mocopi transform tree and converts these to Euler angles for joint angle extraction. The base conversion applies a quaternion to Euler transformation which uses a XYZ rotation sequence, returning roll, pitch, and yaw from the quaternion.

The mapping algorithm implements a dictionary based correspondence between Mocopi skeleton segments and G1 URDF joints. Each segment from the Mocopi skeleton maps to one or more robot joints, with an associated component (roll, pitch, or yaw) designated, specifying which component controls that particular joint. This approach enables both one to one mappings for equivalent joints and one to many mappings where a single human body segment must control multiple robot degrees of freedom, such as the forearm which has multiple degrees of freedom and as such maps to multiple motors on the robot.

The processed joint angles are then published to the ROS2 `/joint_states` topic at 30Hz, providing mapped position commands for all joints. The joint state message includes the URDF joint name and position for each degree of freedom. This allows other controllers and tools to track the robot's current position.

B.0.3 Self-Collision Prevention

The joint positions must be validated before sending the commands to the robot to prevent self collision. This collision could occur between the robots linkages if the operator puts into a dangerous position which may result in hardware damage or unsafe operation. The Flexible Collision Library (FCL) [25] was integrated to provide real-time collision checking between the

Unitree G1's links. Collision meshes for each link were derived from the robot's URDF model, with simplification applied to reduce computational runtime while maintaining adequate safety margins.

To account for modeling uncertainties and to add an additional stopping distance or safety margin, each collision mesh was expanded by 10 cm using uniform scaling. This expansion ensures the collision checker triggers before actual contact occurs, allowing time for operator response or automatic safety interventions.

The collision checking operation was found to be capable of running at ∼270Hz, well beyond the 30Hz limiting factor off the mapped joint states. If a collision is detected by the algorithm, the operator is warned, the command is rejected and the last safe position is maintained. This ensures that when the robot gets within the safety margin it will not continue until another valid position is sent. Once a position is determined as safe, the joint angles are then published to another ROS2 topic `/cleaned_joint_states` which represents just the valid set of joint angles to prevent collision.

B.0.4 G1 Controller Interface

The final stage of the pose mapping is to interface with the G1's low level controller by combining inverse kinematics and hardware level joint position control. The controller implementation uses the Unitree SDK2 Python [28] bindings to communicate with the robot's control system via the Data Distribution Service (DDS) protocol, publishing low level commands to the robot.

The end effector positions are calculated from the cleaned joint states from the self collision checker and are processed through a numerical inverse kinematics solver based on the Pinocchio rigid body dynamics library [29] and CasADi optimisation framework [30]. The solver treats the joint positions as a optimisation problem, combining translational accuracy, rotational accuracy, joint angle regularisation, and trajectory smoothing. These objectives are weighted 50:1:0.02:0.1 respectively, prioritising accurate positioning while preferring smooth motions.

The optimisation handles both arms at 7 DOF each, with all other joints are handled by Unitree's high level controller which helps the robot stand and balance while the arms are being controlled. Joint limits described by the robot's URDF file constrain the solver to the limits of robot, and the solver typically converges within 50 iterations. A four point weighted moving average filter [0.4, 0.3, 0.2, 0.1] then smooths the resulting joint angles, reducing noise and jitter.

Computed joint positions are sent to the motor controllers using position control with proportional derivative gains. Shoulder and elbow joints use $K_p = 80$ N·m/rad and $K_d = 3$ N·m·s/rad, while wrist joints use reduced gains ($K_p = 40$ N·m/rad, $K_d = 1.5$ N·m·s/rad) to accommodate their lower torque ability.

A deadman switch has been added to the wireless controller which provides independent emergency stop functionality. The controller node monitors the deadman state at 100Hz. When released, joint commands are immediately discontinued and the robot maintains position through the motor controller's position hold mode.

Appendix C. Gantt Chart

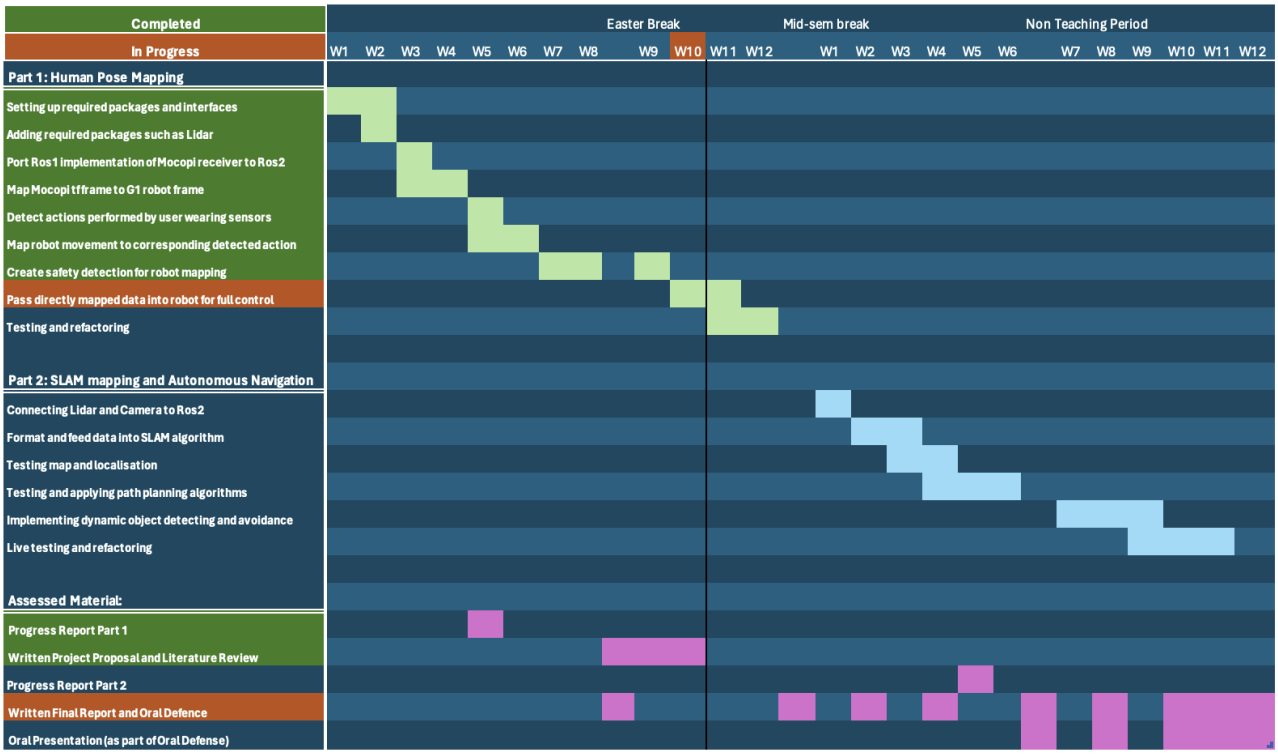


Figure C.1: Project Gantt Chart