BiQu

A Major Qualifying Project (MQP) Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degree of Bachelor of Science in

Robotic Engineering,
Computer Science

By:

Ethan Chandler
Akshay Jaitly
Lehong Wang
Puen Xu
Yifu Yuan
Tao Zou

Project Advisors:

Agheli Hajiabadi, Mohammad Mahdi
Jing Xiao

Date: Oct 2023

Abstract

In recent years, legged robots, particularly quadrupeds, have garnered significant interest from both academia and industry. This heightened attention is primarily attributed to their capacity to traverse various terrains, robustness in challenging environmental conditions, and their broad spectrum of practical applications. Extensive efforts have been dedicated to developing contact-based control strategies for quadrupedal robots. However, there lacks a system for solving contact planning in real-time. Our objective is to conceive and implement a heuristic-based search algorithm capable of effectively determining the optimal contact sequence for robots possessing a variable number of limbs. To validate our proposed system, we plan to conduct physical experiments using a quadruped robot equipped with a custom-designed robotic arm developed by our team.

Acknowledgements

(left intentionally blank)

Contents

1 Introduction **1**

1.1 Project Objectives . . . 1

1.1.1 Task 1: Stable dynamic locomotion given contacts . . . 1

1.1.2 Task 2: Find optimal contact sequences . . . 2

1.1.3 Task 3: Build and control a robust robot arm . . . 2

1.1.4 Task 4: Perceive, segment and represent the environment . . . 2

1.1.5 Task 5: Navigate indoor environments . . . 3

1.2 Background . . . 3

1.3 Related Work . . . 4

1.3.1 Dynamic Models . . . 4

1.3.2 Contact Sequence . . . 4

1.4 Background . . . 5

2 Progress **10**

2.1 Control . . . 10

2.1.1 Control Hierarchy . . . 10

2.1.2 OCS2 . . . 10

2.1.3 Alternative MPC . . . 11

2.2 Perception . . . 11

2.2.1 Sensor . . . 11

2.2.2 Elevation Mapping . . . 11

2.3 Arm Design . . . 12

2.3.1 First Design . . . 12

2.3.2 Second Design . . . 13

2.3.3 Final Design . . . 14

2.4 Arm Implementation . . . 16

2.4.1 First Iteration . . . 16

2.4.2 Second Iteration . . . 17

2.4.3 Final iteration . . . 18

2.4.4 Final result . . . 20

2.4.5 Looking ahead . . . 21

3 Experiments **21**

3.1 The Simulation Environment . . . 21

3.2 Implementing Controller and Perception . . . 22

3.3 The D435i Depth Camera . . . 23

4 Gantt Chart **24**

5 Conclusion **25**

Appendices **26**

A Appendix A Title **26**

References **27**

List of Tables

List of Figures

1 The single rigid body model as seen by the mathematical program. A body with mass m and inertia I is influenced by gravity F_g and contact forces F_{1-4}. The solid blue spheres P_{1-4} are the foot contact locations, and the transparent bubbles centered at the nominal foot position represent conservative kinematic constraints. . . . 7

2 Control Hierarchy . . . 10

3 Integration of Go1 and D435i in Gazebo . . . 11

4 Elevation mapping work in real-time with Go1 controller . . . 12

5 First CAD design of the arm . . . 13

6 First CAD design of the gearbox . . . 13

7 Second CAD design of the arm . . . 13

8 Second CAD design of the gearbox . . . 13

9 PID controller with the base link and the second link . . . 14

10 Final CAD design of the arm . . . 15

11 Exploded view of the CAD . . . 15

12 Final CAD design of the gear . . . 15

13 Manufactured gear . . . 15

14 PCB Board for the robot arm with custom gearbox . . . 16

15 Commercial Gearbox for our robot arm . . . 16

16 Circuit housing . . . 17

17 Communication architecture of the arm and robot dog . . . 17

18 Gripper (Third Link) closed . . . 18

19 Gripper (Third Link) closed . . . 18

20 Result of the second itertion . . . 18

21 List of options for the last iteration . . . 19

22 Comparison matrix for the last iteration . . . 19

23 Original design from 2nd iteration . . . 19

24 First option of the final iteration . . . 19

25 Second option of the final iteration . . . 20

26 Fifth option of the final iteration . . . 20

27 Finalized CAD design . . . 20

28 Arm URDF 21
29 Different terrains within Gazebo. 1)Flat ground with differently sized obstacles. 2)Stairs. 3)Flat ground with walls. 4)An elevated Plane. 22
30 Robot with added camera module in simulation environment. Picture in the background shows the results of terrain reconstruction, and picture in the foreground shows the terrain setup in Gazebo. 22
31 Robot perception in simulation. Colored polygons are the step-able regions, and the colored curves are the planned foot motion. 23
32 Error in terrain reconstruction and motion planning resulting from low update rate of the simulation. 23
33 Result of SLAM after carrying the camera up some stairs. The stairs are reconstructed into colored point cloud, and the path of the camera are shown as a dotted blue line. 24
34 Result of perception algorithm while scanning the same stair with the camera. Picture in the foreground is the camera view, and picture in the background shows the reconstructed terrain with the step-able areas outlined in colors. 24

1 Introduction

This project started as a research project by Ethan and Akshay, which aims to develop a software stack for solving and smoothly merging different motion primitives for legged robots. They were mostly working in simulation because the hardware, a self-built Solo-12 robot, is still under construction. [1] Later, our team decided to purchase a commercially available quadruped robot from Unitree, which is one of the most popular robots for quadruped research. Based on this more stable platform, we can focus more on developing the software for our quadruped.

Our project have a central theme, which is to utilize the mobility of a quadruped robot to perform search and rescue in the emergency of a indoor fire. To achieve that goal, we focus our project on developing software and hardware add-on for a commercially available quadruped robot, which we decided to be the Go1 Edu model robot from Unitree. (Go1) There are three main challenges for navigating in a indoor fire scene. First, the environment in a fire scene is mostly unstructured with obstacles and elevations. Existing control and motion planning algorithm that comes with the Go1 is unable to handle such terrain. Second, it is not uncommon such environment to contain elevations in the terrain that is physically impossible for the robot to overcome. Third, a valid map of the environment usually don't exist, and need to constructed on the fly. The robot also need to automatically decide where to explore during this task.

1.1 Project Objectives

Enhance current software and hardware components to develop a quadrupedal robot featuring an integrated robotic arm, which can potentially function as an auxiliary limb. The envisioned quadruped should demonstrate the capacity to solve for and execute versatile locomotion patterns utilizing a subset of its limbs. Example motions can include: climbing onto high platforms, opening doors, and standing on its hind legs. Furthermore, software enhancements will empower the robot to engage in real-time planning and navigate complex, dynamic environments.

As such, we have five main tasks :

1.1.1 Task 1: Stable dynamic locomotion given contacts

A quadruped is a switched system; the dynamics switch when different limbs are in contact with the ground. Because of this, traditional optimal control approaches don't work as well. The switched dynamics introduce discrete decision variables to the optimization problem, which makes it a lot harder. This is usually

solved by assuming a fixed sequence of modes, defining a gait. Our first task is to implement this, so that we can control a robot after giving it an arbitrary fixed contact sequence.

The subsequent sub-tasks are as follows.

- There must be a hierarchy of controllers to carry out planned trajectories.
- A simulation environment must be set up to test the controllers.
- The MPC must work on flat ground
- The MPC must be able to utilize terrain information
- The MPC must be able to solve given an arbitrary number of limbs

1.1.2 Task 2: Find optimal contact sequences

Design a heuristic-based algorithm to guide the search process of optimal contact sequences that are more efficient and enables online motion planning.

The specific constraints of this task are that

1.1.3 Task 3: Build and control a robust robot arm

Design and build a robot arm for an Unitree Go1 robot dog that needs to be strong enough to act as an auxiliary limb. The quadruped robot needs to exploit the robotic arm as a fifth limb to achieve tasks that the Go1 robot could not with only 4 legs.

1.1.4 Task 4: Perceive, segment and represent the environment

In order to achieve perceptive locomotion in rough terrain, quadruped's capability to perceive and understand its surrounding environment is crucial. It requires the robot to capture complex terrain information in real-time during dynamic locomotion and segment the raw terrain information to get convex safe regions, which will eventually be transformed into constraints for the planner to take in and achieve precise foot placement.

1.1.5 Task 5: Navigate indoor environments

Using onboard depth camera and state estimation, we would like to implement visual slam algorithm to enable the construction of indoor maps and navigate through the possibly clustered environment with complex terrain.

1.2 Background

Trajectory synthesis presents a fundamental challenge in robotics, particularly in the complex domain of legged robots. The main objective is to determine a sequence of control inputs that guide a robot from an initial to a desired state while navigating through a constraint-rich state space. The intricacy of this task for legged robots is inherent; the optimized controls must not only facilitate the robot's movement but also ensure kinodynamic feasibility and adhere to the contact constraints, which are a function of discrete contact events. A model with sufficient fidelity is imperative for effective trajectory synthesis. The model representation must account for the contact wrench cone (CWC) constraint, the underactuated floating base, long swing-leg flight times, and contact velocity/acceleration constraints. These restrictions motivate the need to automatically generate physically feasible trajectories for dynamic motions with an approach that can be generalized to varying robot morphologies and mode sequences.

Historically, the robotics community has grappled with this challenge through diverse motion planning paradigms, including potential fields, probabilistic roadmaps, and randomized tree techniques. However, recent advancements in computational power and mathematical programming have ushered in a new era of trajectory optimization methods [2]. These methods exhibit remarkable versatility, capable of accommodating constraints and cost functions of diverse forms within a unified framework. This flexibility enables the design of trajectories that prioritize speed, smoothness, or control efficiency across a broad range of robotic systems.

The crux of trajectory optimization methods lies in their resolution of the variational problem of optimal control. Two primary strategies have emerged. Indirect approaches first derive the Pontryagin conditions of optimality and subsequently numerically solve the resulting boundary-value problem. While such approaches excel in optimizing the cost function, they often demand accurate initial guesses, posing a challenge in practical implementations. On the other hand, direct approaches discretize the optimal control problem from the outset, addressing the resulting discrete problem using nonlinear programming (NLP) methods. Despite potentially yielding slightly suboptimal trajectories, direct approaches exhibit broader basins of attraction, rendering them preferable for solving robotics problems.

Within the realm of direct approaches, collocation methods have garnered substantial attention due to their efficiency and applicability across a spectrum of problems. Collocation methods use spline interpolators which cast the problem into implicit integrator form, offering an alternative to the explicit integrators commonly used in sequential shooting methods. The overarching efficacy and versatility of collocation methods form the focal point of this paper, specifically within the context of our work.

1.3 Related Work

1.3.1 Dynamic Models

To formulate the optimal control problem, we need to select an appropriate dynamic model to represent the system, as it will influence the complexity and fidelity of the solution. [3] proposes an online ZMP-based optimization scheme that generates trajectories between contact polygons, but is incapable of generating full flight phases. [4, 5] use a linearized SRB model to enable full flight phases in a compact quadratic program. Unfortunately, the former is restricted to negligible roll and pitch angles of the base, while Ding et al., make use of a variation-based linearization in SO(3)—at the cost of decreased accuracy over long horizons due to the first-order approximation. [6–8], make use of the full nonlinear SRB, but the former two opt for Euler angle representations, which introduces singularities in the jacobian. Nguyen et al. utilize the full nonlinear SO(3) representation but the approach necessitates a second TO layer with full rigid-body dynamics to account for the leg dynamics.

Higher fidelity models include the centroidal momentum dynamics [9] used in approaches such as [10–13], with full kinematics. The parameterization of the joint variables allows for a crisp description of kinematic constraints while still keeping the problem size small compared to the full rigid-body dynamics [14]. [15] finds a convex relaxation to the momentum dynamics which yields a single QCQP—however, a formal proof regarding the tightness of the relaxation is not presented. These centroidal momentum approaches motivate the work presented in this paper.

1.3.2 Contact Sequence

One of the most important aspects of legged robots is planning the discrete contact sequence. Implicit contact models such as [16] attempt to surpass the combinatorial nature of the problem by applying a smooth relaxation to the contact constraints, allowing end-effector force to be applied continuously to aid the gradient-based solvers' convergence, smoothly diminishing with the distance to the ground. However,

applying such an approach to general trajectory optimization problems remains difficult, because contact decisions are made implicitly as a highly non-linear function of the robot pose which is stiff to integrate. On the other hand, explicit contact models [17, 18] embed the contact variables in the dynamics, allowing the solver to reason about contact decisions directly.

Mixed integer approaches such as [19, 20] plan the contact sequence by segmenting the obstacle-free space into convex regions, which can then be optimized using MICP. Mixed integer formulations are attractive because they directly tackle the binary nature of contact decisions. Unfortunately, mixed-integer programs are computationally heavy and typically require binary constraint relaxations for convergence, making them largely unsuitable for real-time applications—despite their impressive offline performance in works such as [21] for the DARPA challenge.

In [22], the authors introduce an approach based on the complementary constraints of unilateral contact. These complementary constraints are defined by the condition between the contact force and the distance to contact (i.e., contact force can only exist if the distance to contact is zero, and, likewise, if the distance to contact is nonzero, the contact force must be zero). This technique is further refined in [23, 24]. Unfortunately, these approaches do not satisfy the Linear Independence Constraint Qualification (LICQ), which most off-the-shelf nonlinear program solvers assume. [25] takes advantage of the intuition that every leg will alternate between stance and swing phases, and as such, the contact sequence optimization can be turned into an entirely continuous problem by merely optimizing the timings of each leg's phase independently. This yields an elegant formulation mathematically identical to the LCP approaches but without the need for a specialized solver.

Recently, holistic approaches that solve the discrete contact sequence and continuous trajectory optimization problem simultaneously [26,27] have aroused attention. Most notably, [28] proposes an approach that combines trajectory optimization with an informed graph search coupled with sampling-based planning. Our paper introduces an alternative switched system solver that is suitable for the bilevel optimization introduced in [28].

1.4 Background

Optimal control is the study of finding control trajectories for a dynamical system while satisfying certain constraints. In general, there are two ways to approach this problem: Indirect and direct methods. One school of thought is to generate a feedback control law that provides a control input to guide the state towards some final state for all states within a (potentially global) basin of attraction. This class of approach

is called indirect methods and solves the conditions of optimality in continuous space.

Direct methods transcribe the continuous-time trajectory optimization problem into a nonlinear program with discrete decision variables that are solvable by a computer. The way in which a problem is transcribed is critical to its convergence time and solution optimality. Sequential transcription methods rely on forward simulation, or, 'shooting', and use an explicit integration scheme. Simultaneous direct methods use function approximation by casting the state trajectory as a polynomial, which yields implicit integrators. Such approaches include direct collocation and pseudo-spectral methods, which will be the focus of this work.

In general, indirect methods yield a closed-form solution valid for a certain region of state-space but are more difficult to compute. On the other hand, direct methods are an open-loop solution that is only valid for a certain starting state but are typically much easier to compute.

It is worth noting that control and trajectory planning are the two ends of a continuous spectrum. Robot control is traditionally used in a real-time, feedback-governed fashion, while trajectory planning is often a slower, open-loop procedure. It can be readily observed that there is no clear boundary between the two. For example, if we apply a control law to a system and simulate it forward in time, we are essentially drawing an open-loop plan in the state space for the system to follow. On the other hand, if we can compute open-loop trajectories fast enough, we can apply the first control input in the sequence, update the state, and then recompute the trajectory in an iterative fashion to converge towards the goal state. If these iterative trajectories can be generated at a rate given by the inverse of the Lipschitz constant, it can be used as a closed-loop solution to the optimal control problem.

Approaches using the full SRB nonlinear dynamics in SO(3) are scarce due to the overparameterization of the rotation matrix (9 decision variables but only 3 degrees of freedom in orientation), and the computational expense of computing the orientation error dynamics due to the matrix exponential and logarithm maps. Furthermore, the matrix logarithm suffers from numerical instability as the orientation error approaches 0, and tends to have a divergent Jacobian, which tends to cause algorithmic differentiation algorithms such as those found in Casadi [29] to struggle. Regardless, the SO(3) representation is highly desirable for acrobatic motions because it avoids the singularity encountered in Euler angle representations and the unwinding issue in quaternions.

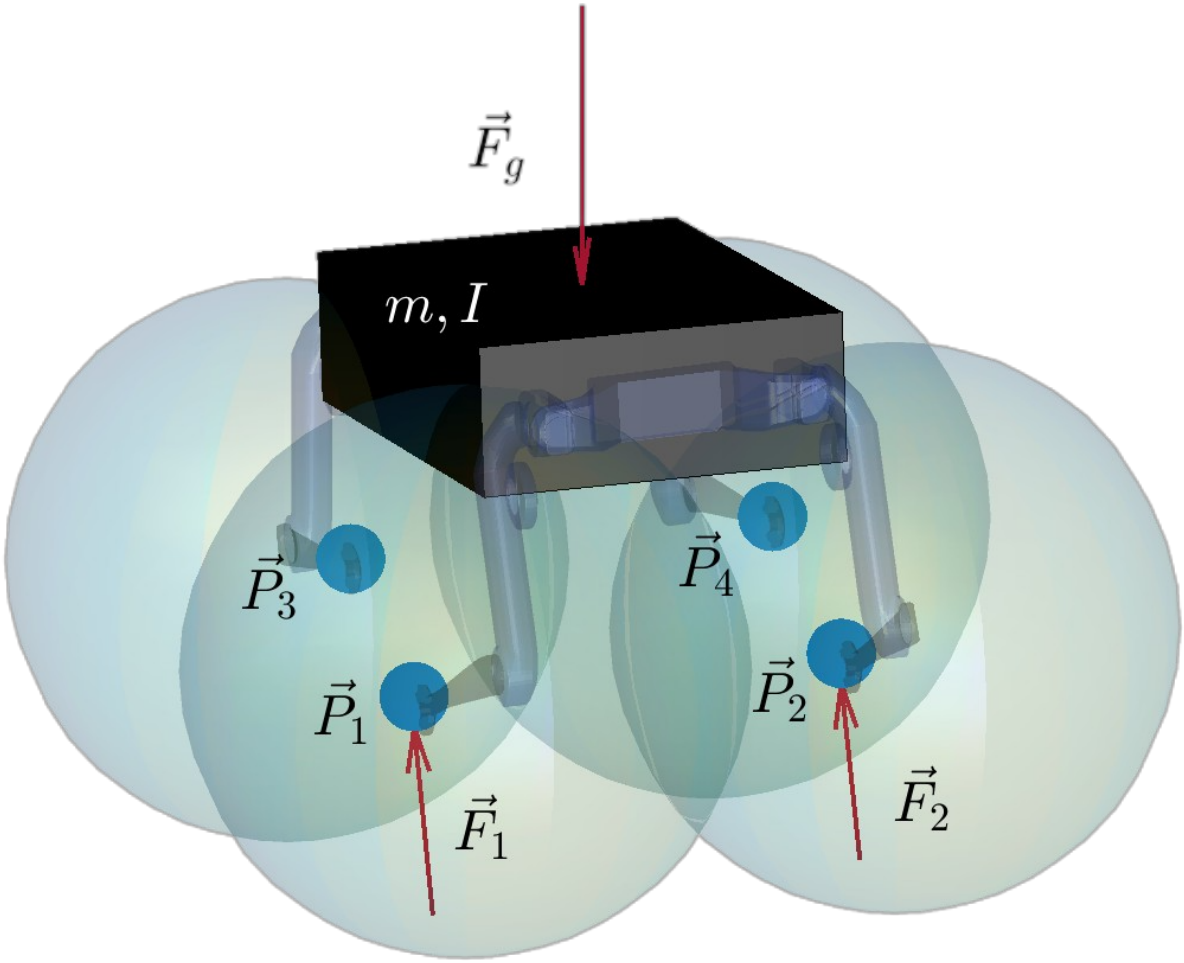

Figure 1: The single rigid body model as seen by the mathematical program. A body with mass m and inertia I is influenced by gravity F_g and contact forces F_{1-4}. The solid blue spheres P_{1-4} are the foot contact locations, and the transparent bubbles centered at the nominal foot position represent conservative kinematic constraints.

We derive the single rigid-body's equation of motion as:

$$\ddot{\mathbf{p}} = \sum_{s=1}^{n_s} \left(\frac{\mathbf{f}_s}{m} \right) - \mathbf{g} \tag{1}$$

$$\mathbf{I}_{\mathcal{B}} \dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times \mathbf{I}_{\mathcal{B}} \boldsymbol{\Omega} = \mathbf{R}^{-1} \sum_{s=1}^{n_s} \mathbf{f_s} \times \left(\mathbf{p} - \mathbf{p}_f^s \right) \tag{2}$$

$$\dot{\mathbf{R}} = \mathbf{R} \hat{\boldsymbol{\Omega}} \tag{3}$$

where n_s is the number of feet, m is the robot's mass, $\boldsymbol{\Omega} \in \mathbb{R}^3$ is angular velocity expressed in the body frame, $\mathbf{R}$ is a 3x3 rotation matrix of the body frame, $\mathbf{g}$ is the gravitational acceleration vector, $\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}} \in \mathbb{R}^3$

is the CoM position, velocity, and acceleration of the body in the world frame; $\mathbf{f}_s \in \mathbb{R}^3$ is the s^{th} force in the world frame. The hat symbol denotes the transformation $\mathbb{R}^3 \to so(3)$ which converts any vector in $\mathbb{R}^3$ to the space of skew-symmetric matrices. For the sake of notation, we define the SRB's state as $x := \left[\mathbf{p}; \dot{\mathbf{p}}; \boldsymbol{\Omega}; \dot{\boldsymbol{\Omega}}\right]$. The trajectory optimization is then formulated as follows:

Written in Bolza form, the general optimal control problem is given by

$$\begin{aligned} \underset{\mathbf{x},\mathbf{u}}{\text{minimize}} \quad & M(\mathbf{x}(t_0), \mathbf{x}(t_f)) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))dt \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \\ & \mathbf{c}(\mathbf{x}(t), \mathbf{u}(t)) \leq \mathbf{0} \\ & \mathbf{b}(\mathbf{x}(t_0), \mathbf{x}(t_f)) = \mathbf{0} \end{aligned} \tag{4}$$

We transcribe the trajectory optimization problem into a nonlinear program using function approximation. Specifically, we simultaneously optimize over the state and controls by approximating each segment between knot points as n^{th}-degree Lagrangian polynomials, with the constraints and dynamics enforced at the k Legendre-Gauss-Radau collocation points. For a detailed summary of pseudospectral collocation and direct collocation in general, the reader is referred to [30, 31], and the following tutorial paper [32].

Temporarily ignoring inequality constraints (these can be added via a logarithmic barrier function to the objective using Lagrangian multipliers), the Hamilton-Jacobi-Bellman equation seeks to produce a value function $V(x,t)$ which represents the cost incurred from starting in state x at time t_0 and driving the system optimally until time t_f

$$\begin{aligned} & \frac{\partial V(x,t)}{\partial t} + \min_u \left(\frac{\partial V(x,t)}{\partial x} \cdot f(x,u) + L(x,u) \right) = 0 \\ & \text{s.t. } V(x,t_f) = M(x) \end{aligned} \tag{5}$$

The presented approach makes use of Casadi [29] for its algorithmic differentiation tools so that we can provide gradient information to the Ipopt [33] solver.

For a constrained nonlinear optimization problem

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g_i(x) - b_i \geq 0 \quad i = 1, ..., k \\ & g_i(x) - b_i = 0 \quad i = k+1, ..., m \end{aligned} \tag{6}$$

The KKT conditions consist of the following elements for optimal primal (x) and dual (λ) variables.

$$\begin{aligned} & g_i(x^*) - b_i \text{ is feasible (Primal Feasibility)} \\ & \nabla f(x^*) - \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) = 0 \text{ (No Feasible Descent)} \\ & \lambda_i^*(g_i(x^*) - b_i) = 0 \text{ (Complimentary)} \\ & \lambda_i^* \geq 0 \text{ (Dual Feasibility)} \end{aligned} \tag{7}$$

Sequential Quadratic Programming iteratively linearizes the original problem at the operating point, producing a subproblem of the form

$$\begin{aligned} & \min \frac{1}{2} x^T Q x + c^T x \\ & \text{s.t. } Ax - b \leq 0 \\ & \text{s.t. } Ex - d = 0 \end{aligned} \tag{8}$$

which is easily solvable with matrix algebra. If the new minimum is better and feasible, we update the optimal solution and repeat until a tolerance is met. SQP methods tend to explore the boundaries of the constraint manifolds, in contrast to IPM approaches which explore the interior first (since the solution is pushed away from the constraint boundaries via a logarithmic barrier function)

2 Progress

2.1 Control

2.1.1 Control Hierarchy

In order to control the real robot, we must have some form of control heirarchy that takes in high level plans, and outputs joint torques. A very common structure is to layer a Model Predictive Controller (MPC), that solves an optimal control problem to find motion across a receding horizon, on top of a Whole Body Controller (WBC). The WBC runs at a higher rate, and attempts to follow the trajectories planned by the MPC.

Luckily, this control hierarchy has been implemented with a very mature open source library, legged_control.

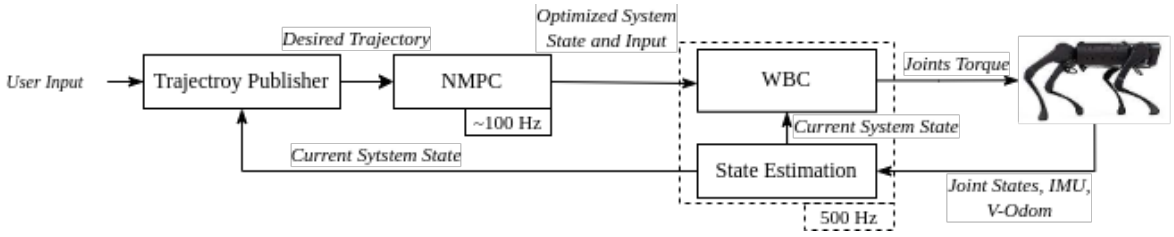

Figure 2: Control Hierarchy

On changing and implementing this library, we were able to command the quadruped Go1 to walk on flat ground in Gazebo.

2.1.2 OCS2

The MPC controller is where the bulk of our work going forward is going to be focused. Once implemented, the other controller in the hierarchy can abstracted away. As we have stated, the MPC needs to be able to solve a highly nonlinear dynamic stability problem given different modes.

OCS2 (Optimal Control for Switched Systems) is a framework for setting up and solving these problems. The work on Versatile Multicontact Planning uses the same framework. We have managed to implement it on flat ground with the legged_control library.

Now, we must be able to solve the problem on odd terrains. Our current work is focused on understanding the assumptions made about, and improving, the usage of terrain information.

2.1.3 Alternative MPC

This is for Ethan to talk about the Alternative solution he has created, and talking about benchmark testing it to evaluate it as a viable alternative

2.2 Perception

2.2.1 Sensor

Unlike the usual wheel-based robot that navigates in a 2D ground plane, a quadruped robot needs elevation information of its surrounding environment in order to execute precise foot placement over complex terrain. For such purposes, a 3D mapping of the target terrain is required.

We picked the Realsense camera D453i from Intel and planned to mount it at the front of the Go1 to generate the point cloud map. By testing, D435i generates the point cloud map with great quality that meets our needs at a reasonable price. Also, with the built-in IMU module inside the D435i, we are able to build a perception platform separately from the Go1. In such way, our perception module can be developed and tested independently, which allows it to be easily implemented on any robot platform with few calibrations.

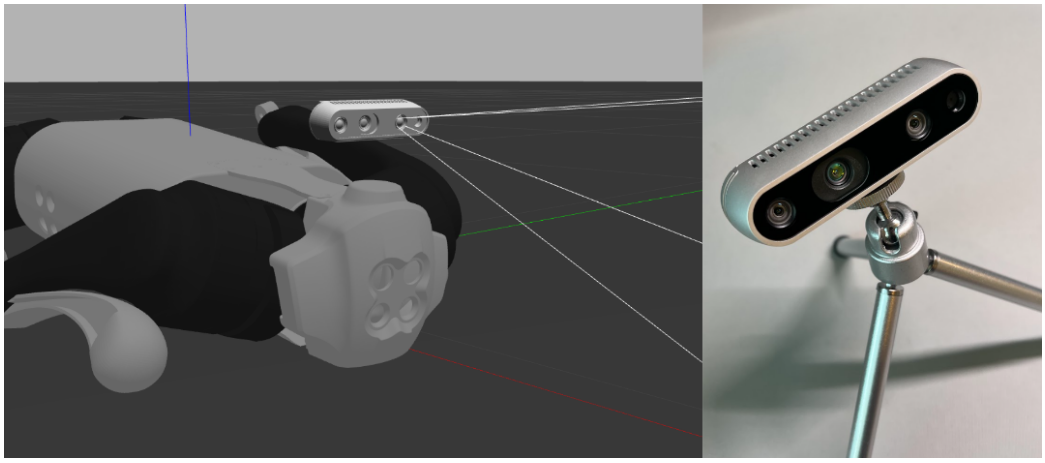

Figure 3: Integration of Go1 and D435i in Gazebo

2.2.2 Elevation Mapping

Robot-Centric Elevation Mapping is a ROS package developed by ANYbotics for rough terrain navigation. [34] [35] It utilized the universal grid map library which is designed for mobile robotic mapping to

store data such as elevation, variance, color, friction coefficient, foothold quality, surface normal, traversability, etc. Normal approaches focus on obtaining a globally consistent map, but such method heavily relies on accurate absolute localization of the system. Robot-Centric Elevation Mapping solved this problem by using relative localization based on proprioceptive sensing such as legged odometry. With such tool, we are relieved from solving the problem of map drifting and our work will be focusing on segmenting and defining convex safe regions for our planner to generate optimal foot placement over rough terrain.

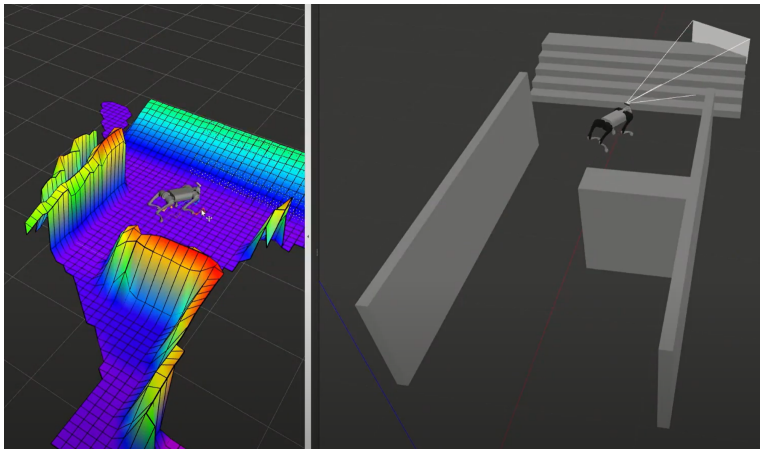

Figure 4: Elevation mapping work in real-time with Go1 controller

2.3 Arm Design

2.3.1 First Design

In A-term, in our first iteration of design, our goal was to attach a fire extinguisher on top of the Go1 robot, and design a robotic arm to manipulate the nozzle of the extinguisher to aim at fire. To tackle the problem, we designed a 3 DoF articulated robot manipulator (Figure 4) to achieve large workspace and great maneuverability. To actuate the robot arm, we were given 3 Pololu 37D motors with 1:50 gear boxes from our advisor Professor Agheli. Through our experiments, we found out that the maximum torque of the motor is 2.1 N-m, despite that it would require 7.42 N-m to enable the robot arm to reach any position within workspace. To amplify the torque of the motors, we designed a 1:4 gear box with 2 layers of spur gear sets, contributing to a 1:2 speed reduction for each layer. However, the gear box turned out to have a large diameter and a considerable width. Since we are hiding the actuators inside the joints, this would make arm joints large, which leads to high moment of inertia of the arm and negatively impact the aesthetic of the robot manipulator. Thus, we would need to make the gear boxes smaller in size in our next iteration of design to make them better fit in the arm joints.

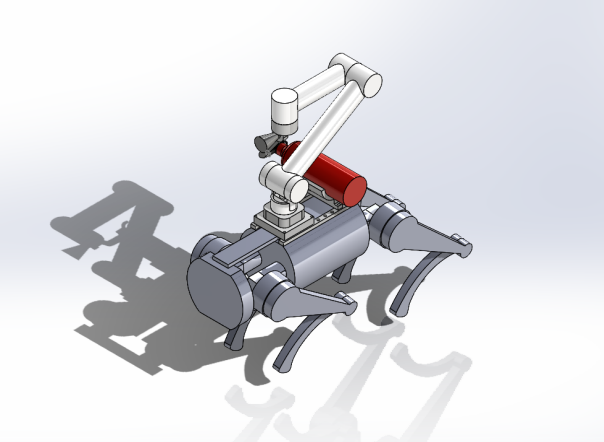

Figure 5: First CAD design of the arm

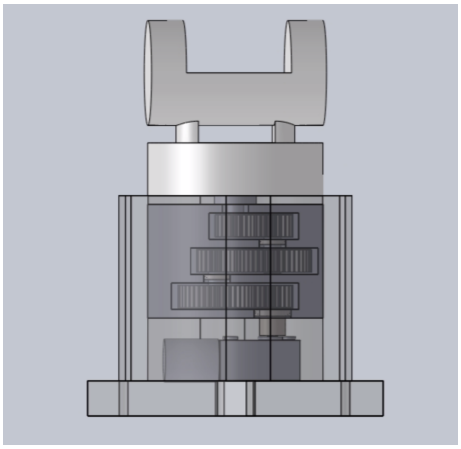

Figure 6: First CAD design of the gearbox

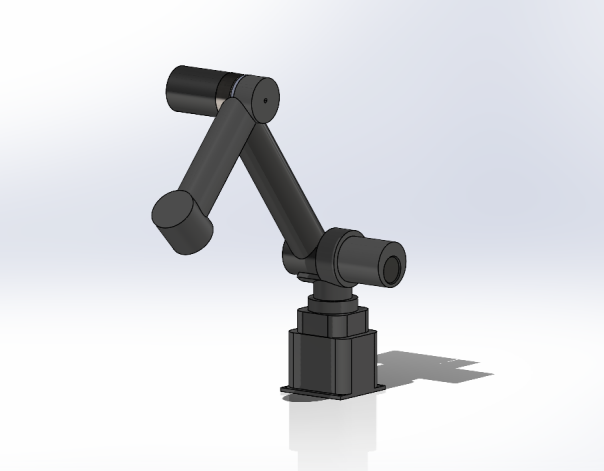

Figure 7: Second CAD design of the arm

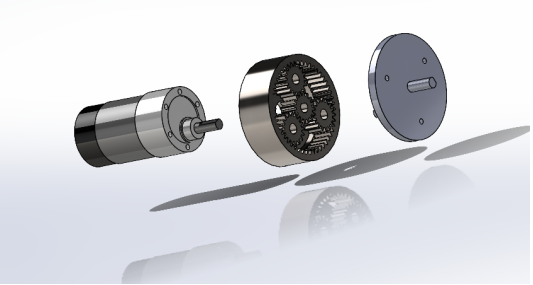

Figure 8: Second CAD design of the gearbox

2.3.2 Second Design

In the second round of design, we kept majority of the previous arm design (Figure 6) and redesigned the gear box to be compact. To achieve a similar 1:4 gear ratio despite decreased size, we designed a planetary gear set (Figure 7) that consist of a ring gear (stationary gear), a sun gear (driving gear), 3 planet gears (driven gears), and a carrier attached to the planet gears that served as the output shaft. Since the ring gear was hold stationary, the gear ratio could be calculated by dividing the tooth number of the sun gear by the sum of the sun gear tooth and ring gear tooth. Through calculation, the gear ratio turned out to be 1:4.125, which is greater than the previous gear ratio despite that the size decreased from 30mm Length x 60mm Diameter to 10L x 50D.

We were satisfied with the improvement in the gear box, and planned to 3D print the arm parts

and the gear box as well as conduct tests on them once they were manufactured.

Soon after the base link and the second link was 3D printed, we assembled them together along with the motor, which we wired with a L298N motor controller. Next, we implemented a PID controller for the motor to smoothly reach any position from -π to π upon a given serial input. In our vision, once we got the Go1 robot, we would program it to open a USB port and send a serial message that consist of 3 joint angles, and, shortly after that, the robot arm would receive the message and react accordingly.

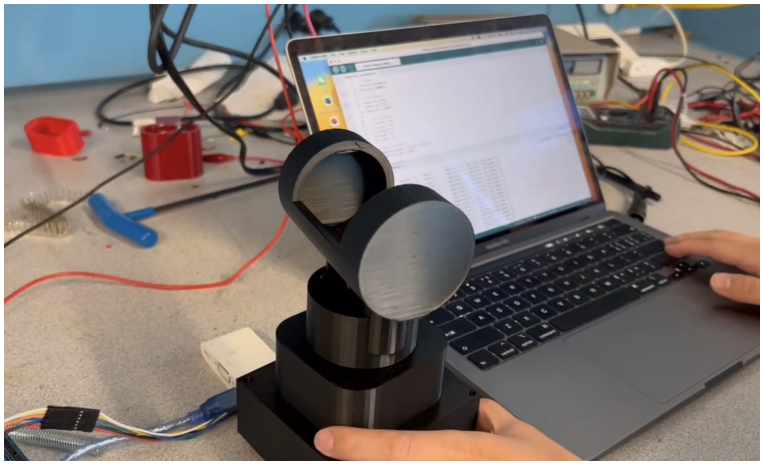

Figure 9: PID controller with the base link and the second link

One thing, however, when we were about to manufacture the rest of the arm parts, our teammates working on the control side of the project had a breakthrough, finding an open-source control framework which we were originally attempting to develop on our own. As a result, we would need to update our ultimate project goal. Through discussion, we decided to use the robotic arm as a fifth limb to achieve tasks that the Go1 robot could not with only 4 legs. Accordingly, we needed to redesign the robotic arm so that the end effector would generate a force which matches the one that a Go1 leg could produce at the tip. To achieve such a task, we would like to use a stronger motor, design a compact gear box with higher speed reduction, and make the robot arm structurally sound.

2.3.3 Final Design

In our final design, the dimensional data of the robotic arm did not change. The design of the base became more reasonable to achieve the goals of saving space and increasing stability. However, we updated the exterior design and internal design of links and joints to match the new motors and gearboxes (Figure 9 & 10). In order to achieve the goal of the robot arm supporting unitree go1, we added a buffer device between link1 and link2 (springs are placed in the left and right tubes to act like a hydraulic rod) to reduce

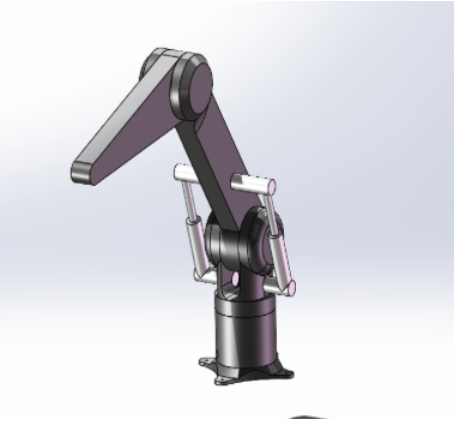

Figure 10: Final CAD design of the arm

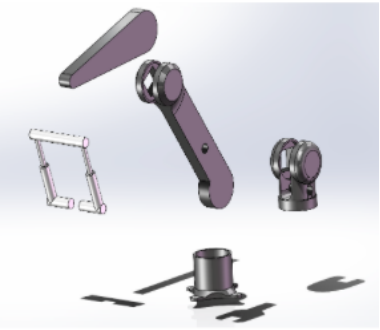

Figure 11: Exploded view of the CAD

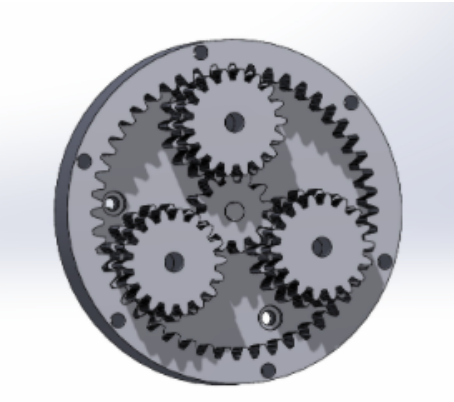

Figure 12: Final CAD design of the gear

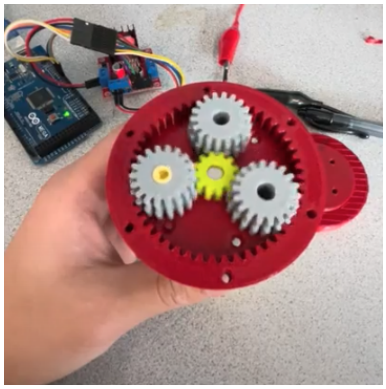

Figure 13: Manufactured gear

the pressure on the motor and gearbox and also reduces the torque requirements. Since our initial plan is to have the robotic arm function as the "fifth leg" of our unitree robot, the design of the end effector is relatively simple. If new features are developed in the future, we will continue to update the end effector.

After the design of the arm was concluded, we moved on to the gearbox design. Through force analysis, we estimated that the maximum torque requirement on the robot manipulator was 32 N-m, which was similar to the one on the Go1 leg. We would need motors with high performance and gearboxes to amplify the torque. Through some research, we decided to go for NEMA17 stepper motors with a stall torque of 0.59 N-cm and develop gearboxes with 1:80 speed reduction. At 400 RPM, the motor, attached to the 1:80 gearbox, would provide 32 N-m with a rotational speed of 5 RPM. This satisfied the torque requirement of the arm while spinning at a reasonable speed. Furthermore, by using a stepper motor, it would become easier to control. For the gearbox, we found out that compound planetary gear set would be the best practice since it produce a significant speed reduction while maintaining a compact form. The methodology behind a compound planetary gearbox was such that there are two layers of planetary gear sets,

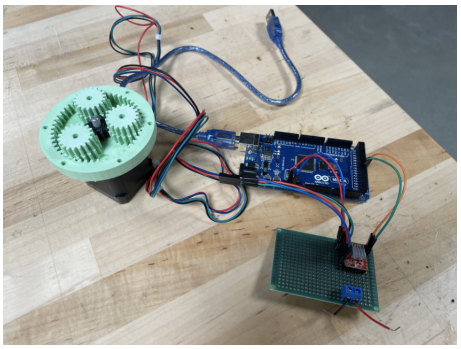

Figure 14: PCB Board for the robot arm with custom gearbox

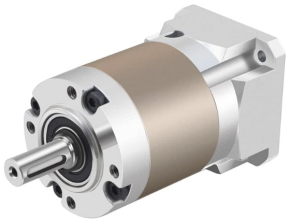

Figure 15: Commercial Gearbox for our robot arm

with a 48-tooth ring gear on the first layer and a 51-tooth ring gear on the second; whenever the first layer rotate 48 cycles, the second layer would make 51 revolutions in the opposite direction, leading to actually $51 - 48 = 3$ rotations. Thus, the gear ratio between the first and second layer was 3:48, which came down to 1:16. Also, since the first layer of planetary gear set had 1:5 speed reduction itself, the total gear ratio turned out to be 1:80. The CAD design of the updated gearbox could be found in Figure 11 and the physical 3D printed one was shown in Figure 12. Our next step was to 3D print the arm parts and machine the gears with metal to add integrity and durability to the gearbox.

2.4 Arm Implementation

2.4.1 First Iteration

Starting B-term, we began implementing our robot arm design. The very first thing that we did was to design a PCB board that served as the motor controller board and to develop Arduino code to drive the motors. We found out that, with our custom gearbox, the output layer cannot be driven by the motor due to slipping. We tried to adjust the motor speed to find out if this was caused by inadequate torque. By gradually reducing the speed down to 100 RPM, the gearbox output layer started spinning. Therefore, it suggested that it cannot provide enough torque to spin the gearbox output at a satisfying speed unless that the speed was decreased to 100 RPM (1.25 RPM after 1:80 gear reduction), which did not meet our needs.

Therefore, we decided to purchase for a commercial gearbox designed for NEMA 17 motors instead of using our custom gearbox. Accordingly, we needed to redesign our robot arm to fit the commercial gearbox.

2.4.2 Second Iteration

We redesigned the arm to reflect our change in the gearbox. In addition, we designed a motor controller box that contains 2 rechargeable 12V batteries (motor power supply), PCB motor controller board (can control up to 3 motors), and the Arduino microcontroller. Along with the controller board, we developed an Arduino code library to control to robot arm to move to any given configuration upon a given speed (within speed constraints). The position and velocity input could be either serial signals or ROS messages. In the near future, to connect the robot arm with the Unitree Go1 robot, we would need to ssh into one of the Jetson nanos embedded inside the Go1 robot via USB cable, and send a ROS message to the Arduino controller board.

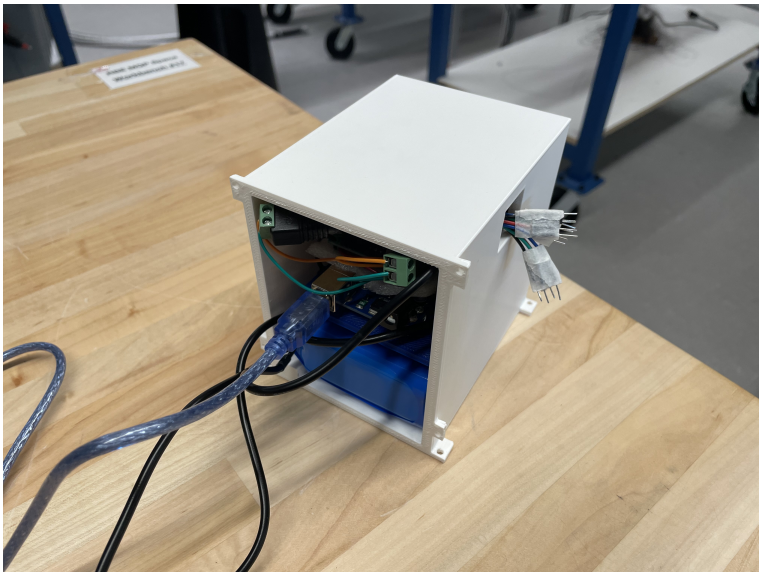

Figure 16: Circuit housing

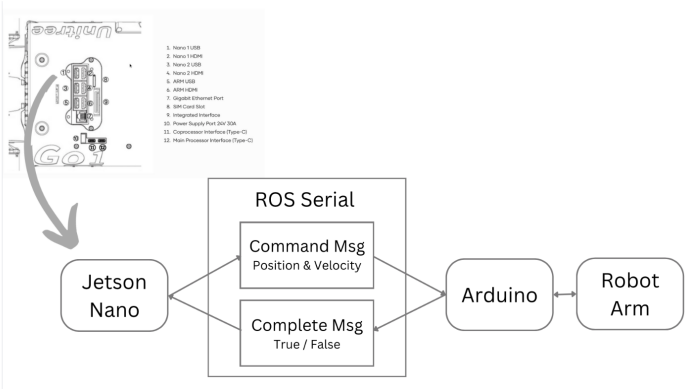

Figure 17: Communication architecture of the arm and robot dog

In addition, we designed the last link so that it was a leg to contact with the ground but, in the

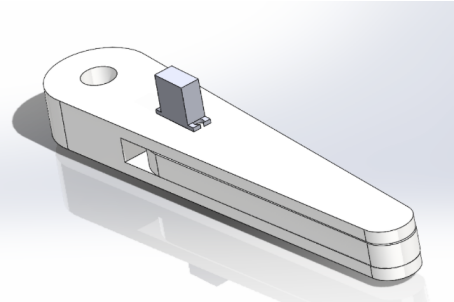

Figure 18: Gripper (Third Link) closed

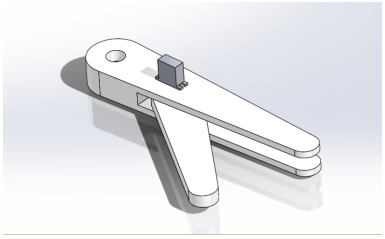

Figure 19: Gripper (Third Link) closed

same time, has the ability to extend to function as a gripper. We used a small-sized servo motor to control the open and close of the robotic gripper. Last but not least, we updated the code library so that the gripper would automatically open and close before and after the motion of the arm.

We 3D-printed the arm parts and assembled the links and joints, and demonstrated during one of the meetings to show the robot arm moved to given configuration and the corresponding motion of the gripper. However, the weight of the arm, 4.1kg, exceeded the desired value (One half of the maximum payload of Unitree Go1, 6kg). Therefore, we needed to redesign the arm to cut down the weight, while maintaining high torque, maneuverability, and large work space.

Figure 20: Result of the second itertion

2.4.3 Final iteration

To reduce the arm weight to be within the one half of the robot payload, we proposed several solutions. We categorized the options based on how the robot arm would contact external surface, and how

to transmit power to lift the last link.

Options	Contact point with the surface	Last joint transmission
1	Third Joint	Motor directly on last joint
2		Four bar
3	End effector	Motor directly on last joint
4		Four bar
5	Use pulley for the last joint and remove base joint	

Figure 21: List of options for the last iteration

Then, in one of the weekly meetings, we presented a comparison matrix of the different options and discussed within the team to choose which design we should proceed with.

	Contact Point	Transmission Method	Weight	Pros	Cons
1	Third Joint	Motor on joint	2.257kg	Easier to design	Large inertia of the link, less payload, limited workspace
2		Four bar	2.410kg	Mechanically sound	Different dynamics, hard to control, limited workspace
3	End Effector	Motor on joint (What we prev. did)	3.057kg	Easier to design	Large inertia of the link, less payload, heavier
4		Four bar	3.21kg	Mechanically sound	Different dynamics, hard to control, heavier
5	Pulley , No base joint		2.725kg	Fixed mass distribution issue	Different dynamics, a bit heavy

Figure 22: Comparison matrix for the last iteration

In the end, we chose the third option to proceed with. In this revision, we removed the 1:100 gear ratio at the base joint since we did not necessarily need high torque in the z-axis. In addition, we revised the gearbox housing so that the motor was not fully-contained to further remove some weight of the PLA parts.

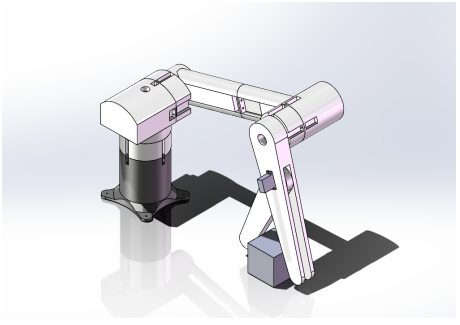

Figure 23: Original design from 2nd iteration

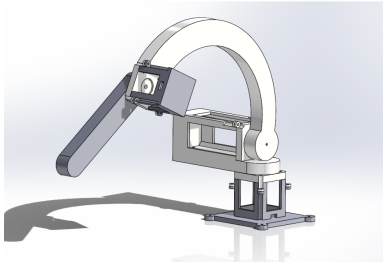

Figure 24: First option of the final iteration

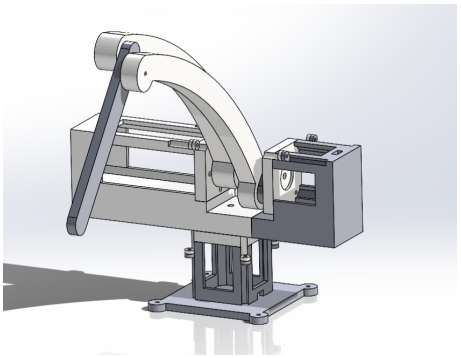

Figure 25: Second option of the final iteration

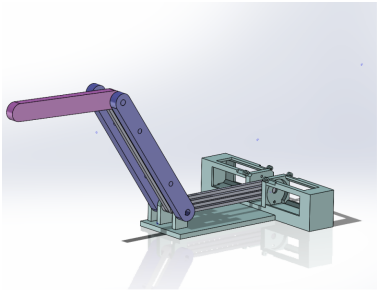

Figure 26: Fifth option of the final iteration

2.4.4 Final result

In the end of the term, we presented our final product. Firstly, we revised the CAD design to reflect the changes of removing first joint gearbox and redesigning motor housing.

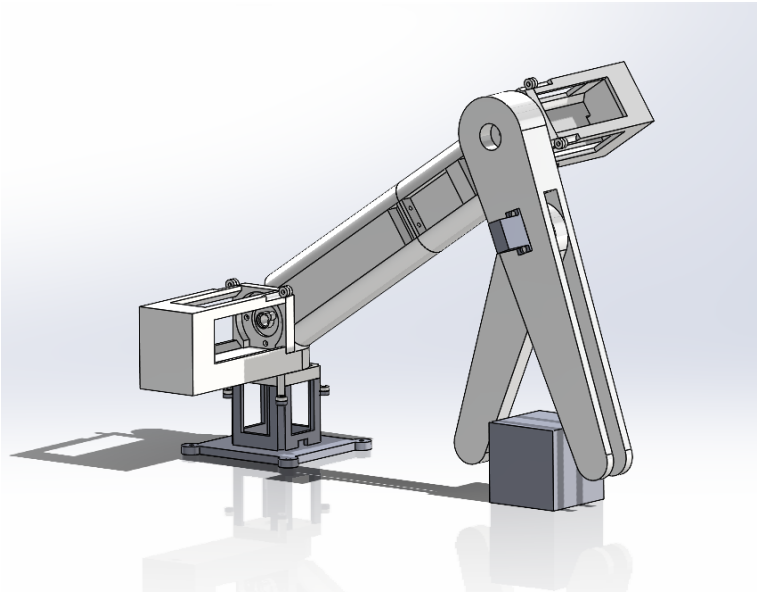

Figure 27: Finalized CAD design

In addition, we created a URDF file and a dynamics model using MATLAB Robotics System Toolbox. Last but not least, we documented our progress in a single Git repository which contained the Arduino code, MATLAB Kinematics Model, MATLAB Dynamics Model, SolidWorks CAD and STL files, and URDF files along with a README file for future teams to continue our work.

One thing, however, during our project presentation, the motor at the base joint was not able to spin. We suspected that it was a dead battery. However, by connecting it to a power supply, the motor did

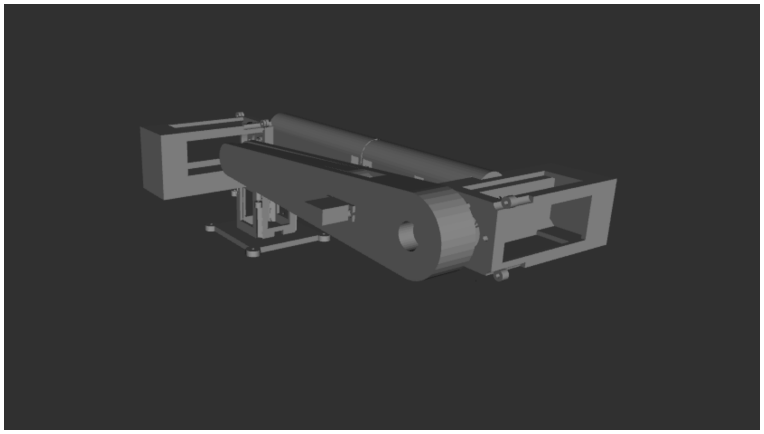

Figure 28: Arm URDF

make sound but did not spin, even without payload. We found that that there was a loose connection in the PCB board that we might need to redo the soldering work to make sure of good connection. We would fix this issue during the break, and we will be able to present a fully-working arm by the start of C-term.

2.4.5 Looking ahead

In the C-term, now that we have the physical robot arm, we plan to work with the trajectory optimization subteam to develop trajectory planning for the arm to accomplish complex tasks that we wish to demo on the project presentation day the following April.

3 Experiments

This project is aimed to implement control algorithm on a physical Unitree Go1 robot, however, due to administrative problems, we don't have access to the physical robot for the first two terms of the project. So for the first half of the project, we mainly focus on testing controllers in simulation, experimenting with depth camera, and building the robotic arm add-on.

3.1 The Simulation Environment

Our simulation is based on ROS Neotic and Gazebo. At the start of the project, we experimented with other environments like ROS2 Humble, Docker environments, Pybullet, etc., but ultimately, we found that ROS Neotic and Gazebo is the most popular combination, thus it has a larger community with more technical support.

We also discovered two git repositories [36] [37] by Qiayuan Liao that implemented and improved the non-linear MPC controllers developed by researchers from ETH Zurich. [38] [39] According to him, these are the best open-source frameworks for quadruped control available, and could be applied to various custom robots. His demos shows very convincing results when applied to the Unitree A1 robot, so we decided to start by reproducing his results in simulation.

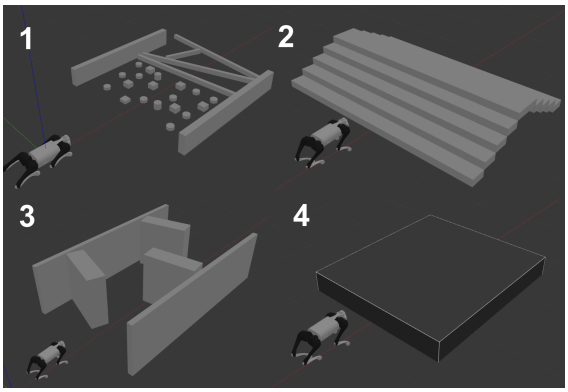

Figure 29: Different terrains within Gazebo. 1)Flat ground with differently sized obstacles. 2)Stairs. 3)Flat ground with walls. 4)An elevated Plane.

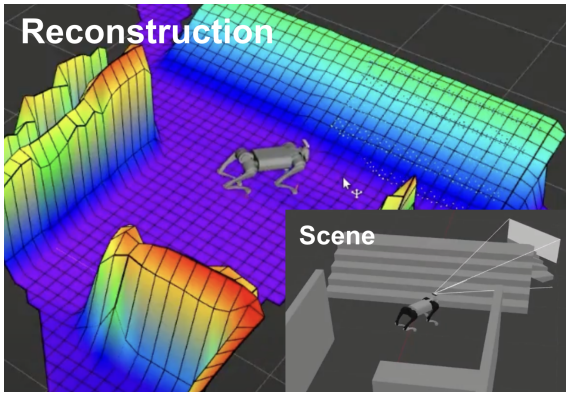

Figure 30: Robot with added camera module in simulation environment. Picture in the background shows the results of terrain reconstruction, and picture in the foreground shows the terrain setup in Gazebo.

3.2 Implementing Controller and Perception

We started with the controller part, and after some effort, we successfully got an A1 robot to move in simulation. Since we are going to work with a Go1 robot, we studied the framework and managed to load that into the simulation. We also learned how to setup terrain in gazebo environment and tested the controller on terrains with different types obstacles. (Fig. 29) The robot could walk unaffected through particles that are 10 centimeter in diameter, but failed to climb up stairs. This result suggested that to navigate through complex uneven terrain, active perception is needed.

Since the Unitree Go1 robot are equipped with stereo depth camera, we think is is a valid assumption to add a depth camera model to the robot in simulation. We choose the depth camera module from RealSense [40], which has a Gazebo plugin that enables simulation of the camera. After setting up the camera, it publishes depth information as a point cloud, as shown in Fig. 30. This point cloud message is then processed by the elevation mapping algorithm [35] mentioned above to reproduce the terrain, which then goes through a segmentation process to retrieve the step-able surfaces for robot footstep planning.

While working on this step, we discovered that Qiayuan just released his results implementing this

process on a Unitree robot. So we switched to his framework and was able to get the robot to walk with active vision in simulation. More specifically, the robot is able to get depth image from the camera, reconstruct it into a terrain, parse out the step-able regions, plan footsteps, and execute those plans in simulation. This enables the robot to climb up an elevated platform that was previously impossible for the robot without vision. (Fig. 31) However, due to limited computation power, our perception algorithm is only updating at around 8 Hz, which is much lower that the suggested update rate of 20 Hz. This results in mismatch between robot's estimated pose and perception, resulting in wrong motion planning which greatly affects the stability and mobility of the robot. (Fig. 32) As of the end of B term, we haven't found a good solution to this problem, but we envision that we wouldn't have the update rate problem when working on the physical robot.

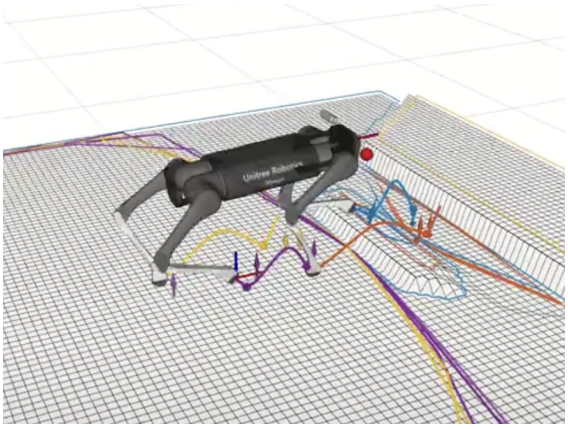

Figure 31: Robot perception in simulation. Colored polygons are the step-able regions, and the colored curves are the planned foot motion.

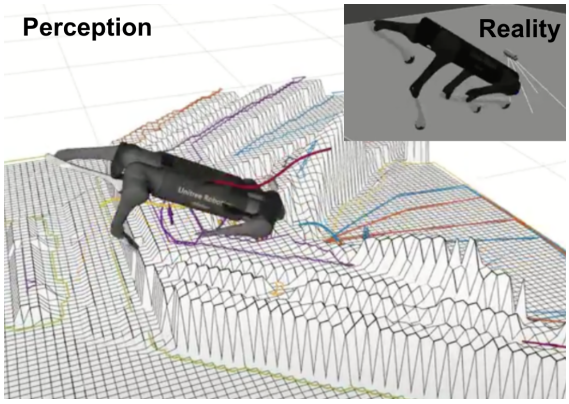

Figure 32: Error in terrain reconstruction and motion planning resulting from low update rate of the simulation.

3.3 The D435i Depth Camera

Though research, we found that RealSense depth camera is widely used for quadruped robot presentation. Since we didn't have access to the physical robot for the first half of our project, we decided to purchase a RealSense D435i camera to test out our perception algorithms. This camera is can capture depth information as well images, and is equipped with a built-in IMU module. This enables us to simulate the perception of the real robot by replacing robot pose estimation with estimation from the IMU. Out of the box, the camera is able to output colored 3D point cloud, and with the help of SDK provided by the manufacturer, it is able to perform SLAM tasks on itself. (Fig. 33) After some work, we managed run our perception algorithm real-time on the camera, and we were able to analyze the scene and parse out the step-able regions for foot step planning. (Fig. 34)

Since the physical robot is equipped with stereo depth camera and IMU, we envision that the algorithm can be easily moved to the physical robot, and the camera could be mounted onto the robot arm add-on as an "eye-in-hand".

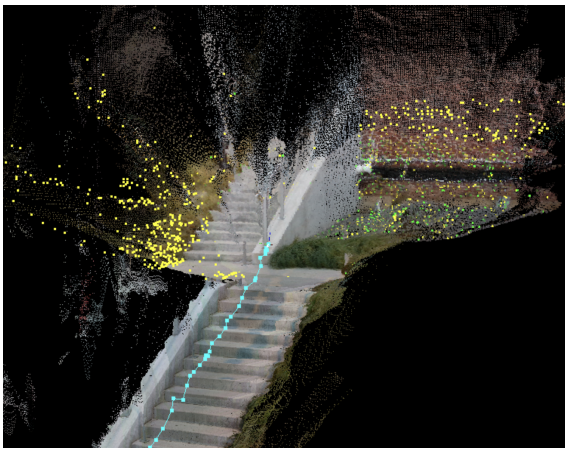

Figure 33: Result of SLAM after carrying the camera up some stairs. The stairs are reconstructed into colored point cloud, and the path of the camera are shown as a dotted blue line.

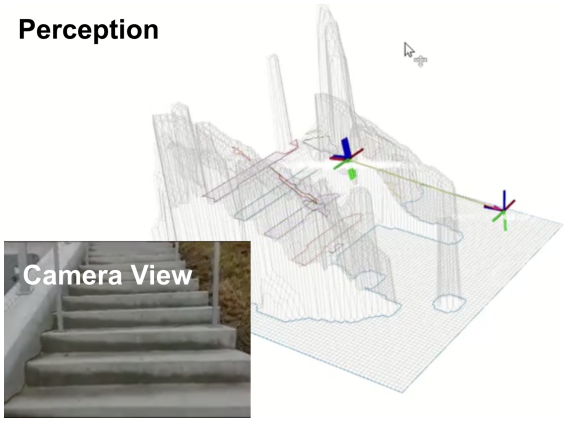

Figure 34: Result of perception algorithm while scanning the same stair with the camera. Picture in the foreground is the camera view, and picture in the background shows the reconstructed terrain with the step-able areas outlined in colors.

4 Gantt Chart

Link to the gantt chart

5 Conclusion

(left intentionally blank)

Appendices

A Appendix A Title

(left intentionally blank)

References

[1] Y. L. L. Hushmand Esmaeili and A. Puttur, "Bimodal quadruped robot," tech. rep., 2022.

[2] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, pp. 193–207, 3 1998.

[3] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," pp. 3359–3365, IEEE, 9 2017.

[4] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," 9 2019.

[5] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, pp. 1154–1171, 8 2021.

[6] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1560–1567, 7 2018.

[7] M. Chignoli and S. Kim, "Online trajectory optimization for dynamic aerial motions of a quadruped robot," pp. 7693–7699, IEEE, 5 2021.

[8] C. Nguyen and Q. Nguyen, "Contact-timing and trajectory optimization for 3d jumping on quadruped robots," pp. 11994–11999, IEEE, 10 2022.

[9] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, pp. 161–176, 10 2013.

[10] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," pp. 93–100, IEEE, 5 2017.

[11] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, pp. 4688–4695, 7 2021.

[12] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter, "A collision-free mpc for whole-body dynamic locomotion and manipulation," pp. 4686–4693, IEEE, 5 2022.

[13] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," pp. 577–584, IEEE, 11 2017.

[14] R. Featherstone, *Rigid Body Dynamic Algorithms*. Springer, 2nd ed., 2007.

[15] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of momentum dynamics for multi-contact motion generation," 7 2016.

[16] E. Todorov, "A convex, smooth and invertible contact model for trajectory optimization," pp. 1071–1076, IEEE, 5 2011.

[17] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, pp. 1–8, 8 2012.

[18] M. Neunert, M. Stauble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1458–1465, 7 2018.

[19] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," pp. 279–286, IEEE, 11 2014.

[20] A. Ibanez, P. Bidaud, and V. Padois, "Emergence of humanoid walking behaviors from mixed-integer model predictive control," pp. 4014–4021, IEEE, 9 2014.

[21] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, pp. 429–455, 3 2016.

[22] M. Anitescu and F. A. Potra, "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, pp. 231–247, 1997.

[23] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, pp. 69–81, 1 2014.

[24] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," pp. 295–302, IEEE, 11 2014.

[25] A. W. Winkler and M. S. I. Me, "Optimization-based motion planning for legged robots," 6 2018.

[26] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3316–3323, 10 2018.

[27] S. Tonneau, A. D. Prete, J. Pettre, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, pp. 586–601, 6 2018.

[28] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation," *Science Robotics*, vol. 8, 8 2023.

[29] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 3 2019.

[30] Q. Gong, W. Kang, N. S. Bedrossian, F. Fahroo, P. Sekhavat, and K. Bollino, "Pseudospectral optimal control for military and industrial applications," pp. 4128–4142, IEEE, 2007.

[31] J. D. Eide, W. W. Hager, and A. V. Rao, "Modified radau collocation method for solving optimal control problems with nonsmooth solutions part i: Lavrentiev phenomenon and the search space," pp. 1644–1650, IEEE, 12 2018.

[32] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, pp. 849–904, 1 2017.

[33] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 3 2006.

[34] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

[35] E. Zuish, "Robot-centric elevation mapping." [Online]. Available: https://github.com/ANYbotics/elevation_mapping.

[36] Q. Liao and Y. Shang, "Legged control." [Online]. Available: https://github.com/qiayuanl/legged_control.

[37] Q. Liao, "Legged perceptive." [Online]. Available: https://github.com/qiayuanl/legged_perceptive.

[38] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," 2021.

[39] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model predictive control," 2022.

[40] I. RealSense, "Realsense ros." [Online]. Available: http://wiki.ros.org/RealSense.