DeltaWalker: A Soft, Linearly Actuated Delta Quadruped Robot

Jennifer Yang

CMU-RI-TR-24-23

August 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Zeynep Temel, *chair*
Zachary Manchester
Zilin Si

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my parents and sister.

Abstract

Quadruped robots offer a versatile solution for navigating complex terrain, making them valuable for applications such as industrial automation or search and rescue. Although quadrupeds are more complex than bipeds, they are easier to balance and control and require fewer joints to actuate compared to hexapods. Traditional quadruped designs, however, often feature complex leg mechanisms that are difficult to scale and require many actuators.

We introduce the DeltaWalker, a novel quadruped robot that uses linearly actuated Delta robots as legs. Delta robots offer advantages such as precision and scalability, while still being able to move with three degrees of freedom. This design is inspired by DeltaHands, a 4-fingered hand robot with soft, 3D-printed linear Delta robots. We turn the DeltaHands upside down and adapt the design for locomotion. We explore various gait patterns, including manually designed and trajectory-optimized gaits, evaluated in both simulated and real-world environments. We also describe the system kinematics and investigate other capabilities, such as rotating in place. The findings from these evaluations demonstrate the potential of the DeltaWalker as a simpler, scalable, omnidirectional quadruped robot.

Acknowledgments

I would first like to thank my advisor, Prof. Zeynep Temel for her guidance and encouragement throughout my Masters. She has been an amazing mentor and someone I can truly rely on for support. I have grown as a robotics researcher, and I could not have done it without her guidance. I would also like to thank my labmates in the Zoom Lab, especially Zilin Si for her mentorship and friendship. I'm also incredibly grateful to Sarvesh Patil, Shashwat Singh, Sidney Nimako-Boateng, Sha Yi, and Chris Chang for their feedback and discussions. From Robotic Exploration Lab, I would like to thank Prof. Zachary Manchester and Arun Bishop for their guidance. I would also like to thank Alfred Chang and Jon Yeo for their help with this project.

Finally, I want to thank my family and friends for all of their support throughout my masters as well. I am especially grateful for my parents, Zhiqiang Yang and Zhen Yu, as well as my little sister, Sarah Yang. They have shaped me to be who I am today, and I would not be here without them. I am also incredibly grateful to Danny Marks for his endless encouragement. He has given me great advice and helped me stay motivated.

Funding

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-2024794.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Mobile robots are extensively studied as they can be used to autonomously navigate and interact with diverse environments. From industrial automation to space exploration to search and rescue, mobile robots are a versatile substitute for using humans in assorted complex conditions [4]. These robots can be divided into three primary categories: legged robots, wheeled robots, and tracked robots [28]. Although legged robots are more complex from a design and control perspective than the other categories of robots, they stand out as they are capable of navigating complex terrain [40]. Legged robots can be further subdivided into categories based on the number of legs the robot has. The primary categories are biped (2 legs), quadruped (4 legs), hexapod (6 legs), and octopod robots (8 legs). Although the quadrupeds are more complex robots than bipeds, they are easier to balance and control. Quadrupeds are also easier to control and simpler robots than hexapods or octopods as they have fewer joints to actuate [40].

Quadrupeds often feature complex leg designs [49]. However, these designs are difficult to scale and require many individual components to actuate. So how do we make the leg design of a quadruped simpler, cheaper, and easier to manufacture? We believe we achieve this by using Delta robots. Delta robots are fast, precise, and easily scalable [35]. They are capable of moving their end-effector with 3 translation degrees of freedom (DOF) and only require 3 rotary or linear actuators [36].

We present a novel quadruped robot, the DeltaWalker (Figure 1.1). This robot uses 4 soft, linearly-actuated Delta robots as legs arranged in a square pattern around the center of the robot. Our goal is to investigate if using linear Delta robots as legs is feasible for a quadruped robot. This was inspired by DeltaHands [39]— a 4-fingered hand robot that uses soft, 3D printed, linear Delta robots as each finger. We want to see if by turning the robot upside down, it can function as a legged robot. This would be a cheaper and simpler quadruped than many of the other quadrupeds currently available.

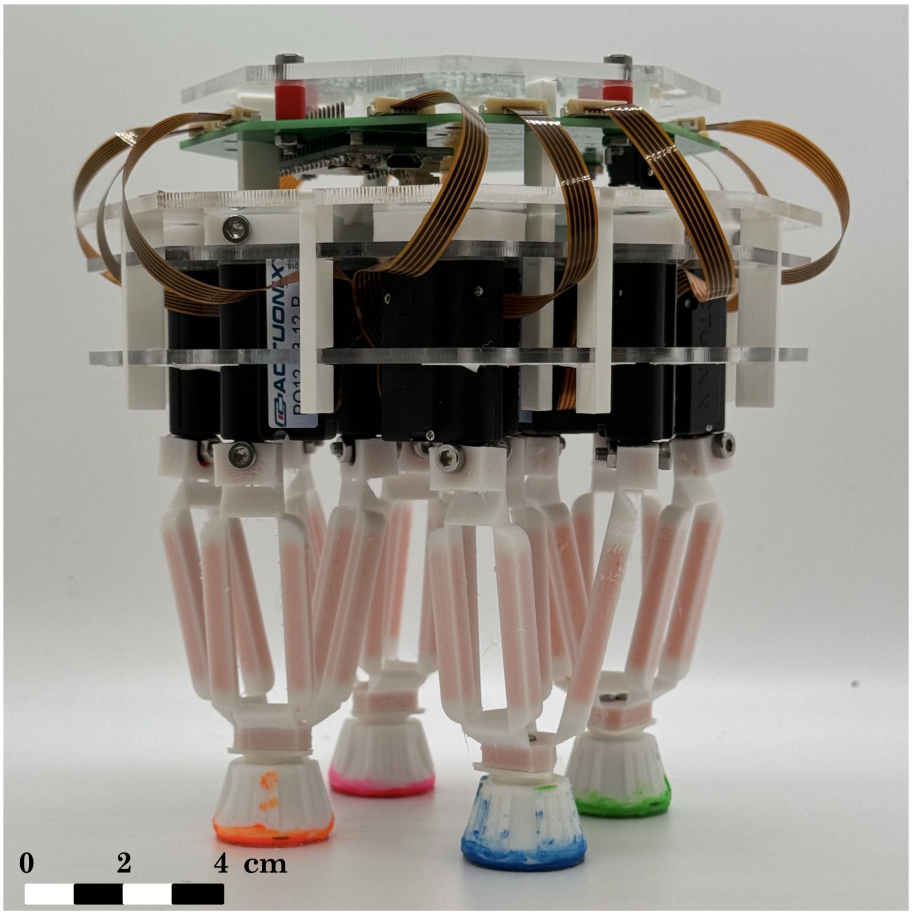

Figure 1.1: DeltaWalker: A novel quadruped robot that uses 4 soft, linearly-actuated Delta robots as legs.

1.2 Related Work

1.2.1 Quadruped Robots

Quadrupeds are an extensively researched field, with research dating back as early as the 1900s with a rudimentary walking mechanism [4]. Recent research greatly improved upon these initial designs and introduced a spectrum of robots (Figure 1.2).

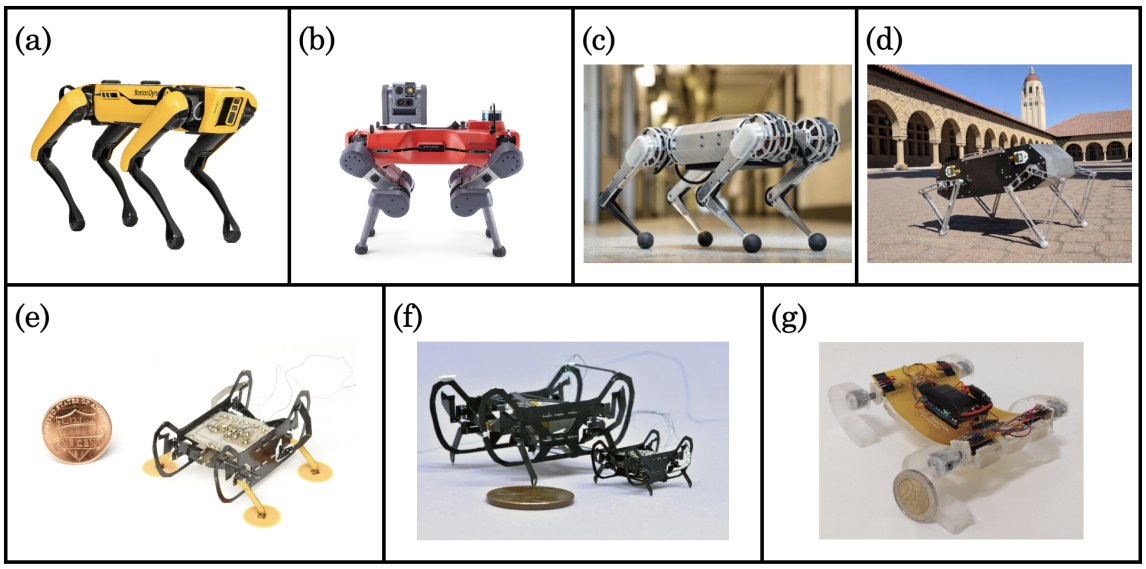

Figure 1.2: Assorted state of the art quadruped robots ranging from large to microscale. (a) Spot [12], (b) ANYmal [30], (c) MiniCheetah [9], (d) Doggo [23], (e) HAMR-E [6], (f) HAMR-JR [20], (g) S-Quad [21]

.

Large scale quadrupeds, such as Boston Dynamics Spot [1, 12, 13, 14, 15, 16] or ANYbotics ANYmal [3, 18, 30, 34] can use dynamic locomotion to traverse diverse terrains. Medium scale quadrupeds like MIT MiniCheetah [9, 22] and Stanford Doggo [23] are lighter and more agile. However, these large and medium scale robots are expensive and heavy. Furthermore, they often feature complex leg designs. The actuators or additional transmission components (eg: belts) can be spread throughout the leg. For example, Boston Dynamics Spot has two actuators located at the hip and one actuator located at the knee (Figure 1.3) [14]. MiniCheetah has all 3 actuators located at the hip, but there is a belt between the hip and the knee [22]. This can

make the actuators or transmission components more susceptible to damage.

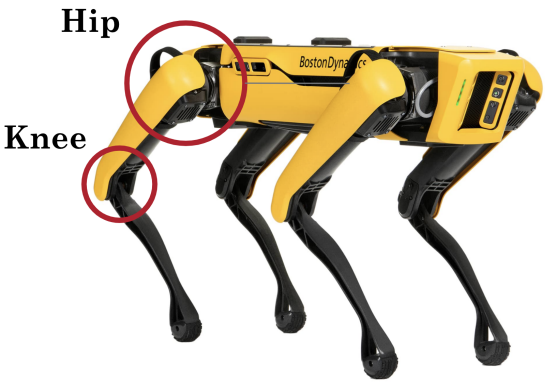

Figure 1.3: The Boston Dynamics Spot robot leg has two main joints— the hip and the knee [14].

There are also smaller scale quadrupeds such as Bilkent University's S-Quad [21]. This robot uses a soft leg design that allows it to traverse some varied terrain. There are also assorted micro-robots, such as Harvard's HAMR robots [6, 11, 20]. There are multiple versions of the robot, with various speed and traversal capabilities. The most relevant are HAMR-E [6, 11], for its development of the modified tripedal gait, and Harvard-JR [20] as the most recent version of the robot. These small and micro-robots are highly useful in narrow environments, such as small pipes or potentially the human body. They have low power requirements. They are also cheap and quick to manufacture, so they could potentially be used as disposable robots [45]. However, these smaller robots have limited payloads and cannot traverse as diverse environments, as they may not be able to climb obstacles or steps. Furthermore, they can be more susceptible to environmental factors such as wind or rocks. They may also require complex manufacturing processes, which may require expensive equipment. For example, HAMR-JR is manufactured with the PC-MEMS fabrication process and pop-up assembly techniques [20].

A summary of these robots can be found in Table 1.1. A velocity metric in Body Lengths (BL) per second is include to benchmark these robots relative to each other. Leg length values are estimated. For example, the MiniCheetah leg length is estimated as the sum of the two leg link lengths given. Values that could not be found or estimated are left blank. Descriptions of quadruped gaits can be found in

Section 1.2.2.

Robot	**Spot**	**ANYmal**	**Mini Cheetah**	**Doggo**	**HAMR-E**	**HAMR-JR**	**S-Quad**
Length (mm)	1100	930	380	420	45	22.5	150
Width (mm)	500	530	-	200	40	-	100
Height (mm)	610	890	300	320	20	-	40
Leg Length (mm)	-	250	400	160	-	-	-
Max Speed (mm/s)	1600	1300	2450	900	4.6	313	81
Max Speed (BL/s)	1.45	1.4	6.45	2.14	0.1	13.91	0.54
Mass (g)	32.7 × 1000	50 × 1000	9 ×1000	4.8 × 1000	1.48	0.32	69
DOF (Per Leg)	3	3	3	2	2	2	1
Actuators (Per Leg)	3	3	3	2	2	2	1
Leg Type	Rigid	Rigid	Rigid	Rigid	Rigid	Rigid	Soft
Gaits	Walk, Crawl, Amble, Jog, Hop	Walk, Trot	Trot, Trot-Run, Bound, Pronk	Walk, Trot, Bound, Pronk	Modified Tripedal	Trot, Pronk, Bound, Jump	Trot
Cost ($)	74,500	150,000	3,600+	3,000	-	-	-

Table 1.1: A comparison of the characteristics of different state-of-the-art quadruped robots.

1.2.2 Quadruped Gaits

Quadruped gaits are often inspired by existing mammals and contribute to the design of the robot. These gaits can be categorized with a hierarchy, as detailed in [48] and summarized here. The first category is periodic versus aperiodic gaits, referring to whether or not the gait follows a set pattern. The main aperiodic gait is the free gait, used for traversing complex terrain. For periodic gaits, there are further subcategories of continuous versus discontinuous gaits. The primary discontinuous gait is the crawl. For continuous gaits, there are further subcategories of static and dynamic gaits. The main static gait is walk. This gait is also known as a quasi-statically stable gait, as it can be stopped at any moment and remain stable. Amble, modified tripedal, trot, bound, pronk, jog, hop, and jump would all fall under dynamic gaits as they tend to be faster gaits. Basic descriptions of these gaits can be found in Table 1.2, and it is supplemented with Figure 1.4.

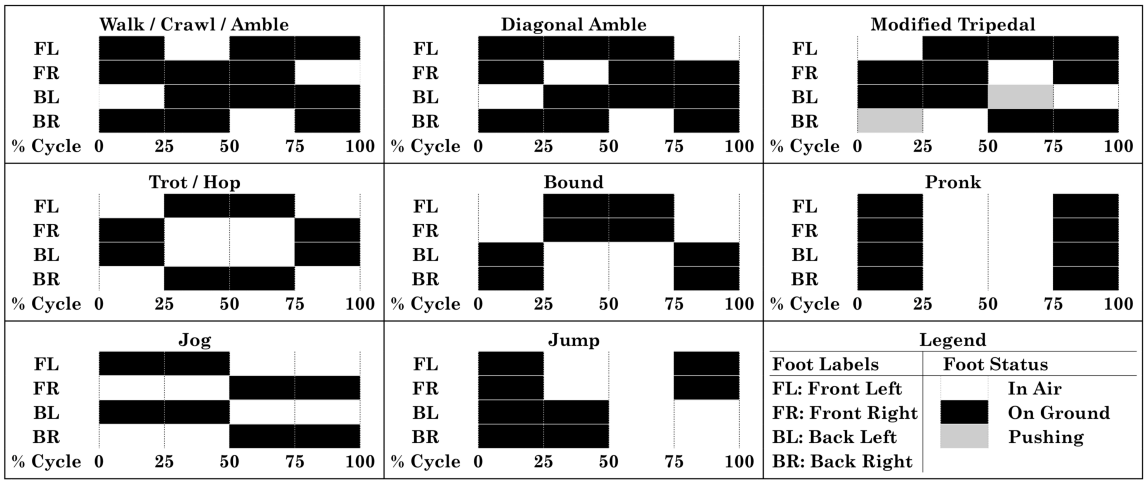

Figure 1.4: Diagram of common quadruped gaits and their timings.

Gait	Speed	Description
Walk	Slow	Back left leg moves, then front left, then back right, then front right
Crawl	Slow	Same as walk but with a lowered COM
Amble	Medium	Faster version of walk with 3 feet on the ground at all times
Diagonal Amble	Medium	Variation of amble where back left leg steps, then front right, then back right, then front left
Modified Tripedal	Medium	Back legs push front legs forward as the robot moves, adapted from a hexapod tripedal gait
Trot	Medium	Diagonal legs pairs alternate in contacting the ground
Bound	Medium	Front and back leg pairs alternate in contacting the ground
Pronk	Fast	All four legs leave the ground and return at the same time
Jog	Fast	Left and right leg pairs alternate in contacting the ground
Hop	Fast	Same as trot but switching pairs after 5 hops rather than each hop
Jump	Fast	Front legs depart the ground and then the rear legs, leaving the robot airborne temporarily
Free	Variable	Foot swing sequence is planned live based on the terrain

Table 1.2: Description and speeds of common quadruped gaits [11, 13, 15, 17, 29, 37, 48].

1.2.3 Trajectory Optimization

The development of control strategies for achieving stable quadruped locomotion has advanced significantly over the years. Due to the complex nature of these quadruped gaits and robots, many control approaches have been explored. Some methods are model-free reinforcement learning (RL), model predictive control (MPC), and trajectory optimization. Although each method has its respective advantages, we opt to use hybrid trajectory optimization.

RL algorithms do not require any explicit model of the system dynamics, and instead optimizes a reward function to tune a controller. As the controller is continuously executed, it is able to learn new behaviors and adapt to new situations in real time [46]. This concept has been used to train ANYmal in complex, unknown terrains [24].

MPC uses a predictive model to optimize control inputs over a finite time horizon, considering system dynamics and constraints [41]. This method also allows robots to navigate rough terrain, and has been tested on Unitree A1 [44] in the paper [31].

Trajectory optimization computes optimal trajectories for robot motion by minimizing a cost function subject to system dynamics and constraints [42]. Hybrid trajectory optimization is specifically applicable as it takes into account known contact mode sequences [25, 42]. This allows us to designate specific gait patterns, as outlined in Section 1.2.2 to use for generating trajectories.

1.2.4 Soft Linear Delta Robots

Delta robots were initially designed as a pick and place tool for industrial environments [36]. Standard Delta robots have a fixed based and moving platform that always remain parallel to each other and are connected with 3 kinematic chains. More recently, they are used as a method for changing the position of 3D printer extruders [47]. There are assorted variations of the Delta robot, one of which being the linear Delta. These robots use linear actuators rather than rotary actuators to control the movement of the end-effector platform.

The manufacturing techniques, applications, and capabilities of these robots have been further explored in research [26, 27, 32, 33, 39]. Specifically, recent research has developed a method for manufacturing soft Delta robots [26]. This method uses 3D

printing to print an adapted version of the Delta links with compliant materials and living hinges. This method has been utilized in assorted research focused on using multiple Delta robots to perform dexterous manipulation tasks [32, 33, 39].

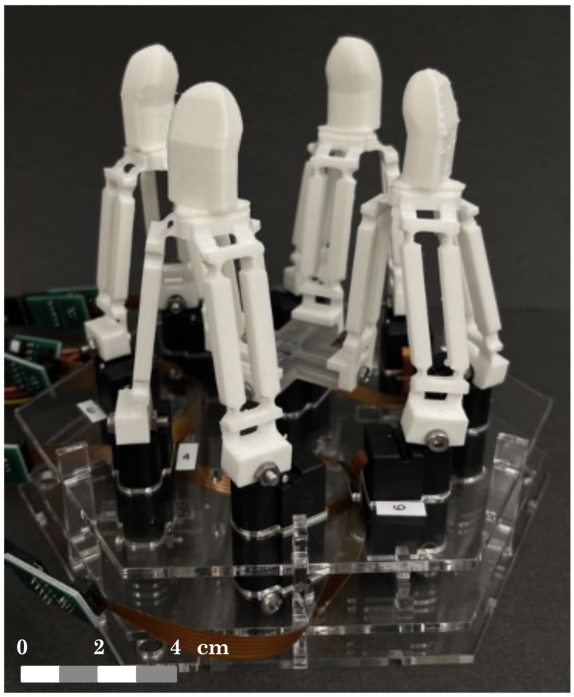

Figure 1.5: DeltaHand: A manipulation robot that uses 4 coupled, soft, linearly-actuated Delta robots as fingers [39].

DeltaHand [39], shown in Figure 1.5, is a manipulation robot and its design is the basis for the DeltaWalker. It is a hand robot that uses 4 soft, linear Delta robots as fingers. The design of this robot is modularized, making it easy to manufacture and modify. The links are 3D printed with soft TPU material and other housing components are 3D printed or laser cut with acrylic. It uses low cost off-the-shelf components, costing less than $800 total.

1.3 Contributions

With this thesis, we aim to demonstrate the capabilities of the DeltaWalker. The contributions of this document are laid out below:

- We modify the DeltaHands [39] design to be better suited towards locomotion rather than manipulation (Sections 2.1 and 2.2).
- We introduce the kinematics of the system (Section 2.3).
- We explore 4 different gait patterns (Section 3.2) through manually designed trajectories (Section 3.3) and trajectories generated through trajectory optimization (Section 3.4).
- We build a simulation environment for the system (Section 4.1).
- We check the feasibility of the generated trajectories in simulation (Section 4.2).
- We develop a framework for running real-world testing on the system (Section 5.1).
- We evaluate the performance of the generated trajectories in the real world (Sections 5.2 and 5.3).
- We perform additional studies such as rotating the robot in place to further understand the robot's capabilities (Section 6).

Chapter 2

DeltaWalker

2.1 Hardware

The DeltaWalker (Figure 1.1) weighs 570 grams and is approximately 15 cm x 15 cm x 18 cm. It uses 12 linear actuators to move 4 compliant Delta robots. Each Delta has 3 translational degrees of freedom, allowing the robot to move omnidirectionally. This eliminates the any need for the robot to have to be reoriented in order to change the direction of movement. The compact and lightweight design allows the robot to be useful in tighter spaces where a large robot would not be able to fit but a microrobot would not be effective at completing a task within. This robot is also low cost, with materials costing less than $800 in total.

2.1.1 Primary Components

Like the DeltaHand, the structure of the walker is modularized to make components easy to manufacture and replace [39]. The main components of the DeltaWalker are the legs, feet, shoes, frames, spacers, actuators, and PCB of the robot, shown in Figure 2.1.

We 3D print the legs, feet, and shoes, shown in Figure 2.2. The legs are each Delta robots with compliant links, printed with a combination of TPU and PLA. These links are a modified version of [26]. Each parallelogram arm of the Delta link is 5 cm long by 3 cm wide by 0.5 cm tall, with each beam of the arm being 0.5 cm wide

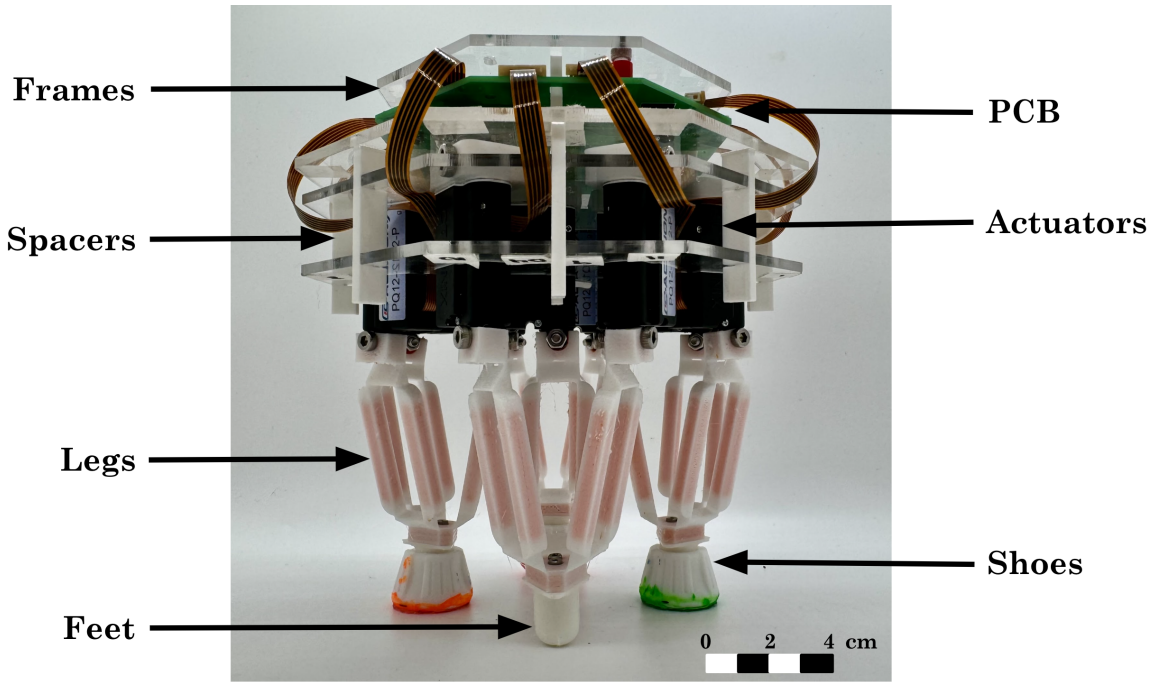

Figure 2.1: The components of the DeltaWalker consists of the laser cut acrylic frames, custom PCB, commercial actuators, and 3D printed spacers, legs, feet, and shoes. Dimensions are provided in the text.

as well. The links are primarily printed with TPU. However the beams of the links and mounting interface between the Deltas and actuators are filled with PLA cores. This is to prevent any deformation or twisting of the links, as the soft nature of the TPU makes it subject to easy deformation. This reduces how often they have to be replaced and minimizes inconsistencies in the walker weight distribution. The feet are printed with TPU and attached to the end of the links. The feet are screwed into the end-effector point of each Delta. They are rounded cylinders that are 1.2 cm in diameter and 1.5 cm tall. The surface that would contact the ground without shoes are 0.6 cm in diameter. However, this contact area is too small and causes the robot to tip easily. Furthermore, the material is too slippery causing the robot to slide. So, we supplement this with shoes, which are also printed out of TPU but have grip tape [2] added to the contact surface. The shoes have a diameter of 2.5 cm. They can be capped onto the end of the feet, resulting in a combined foot and shoe height of 1.8 cm. Alternative shoes can be printed and easily swapped.

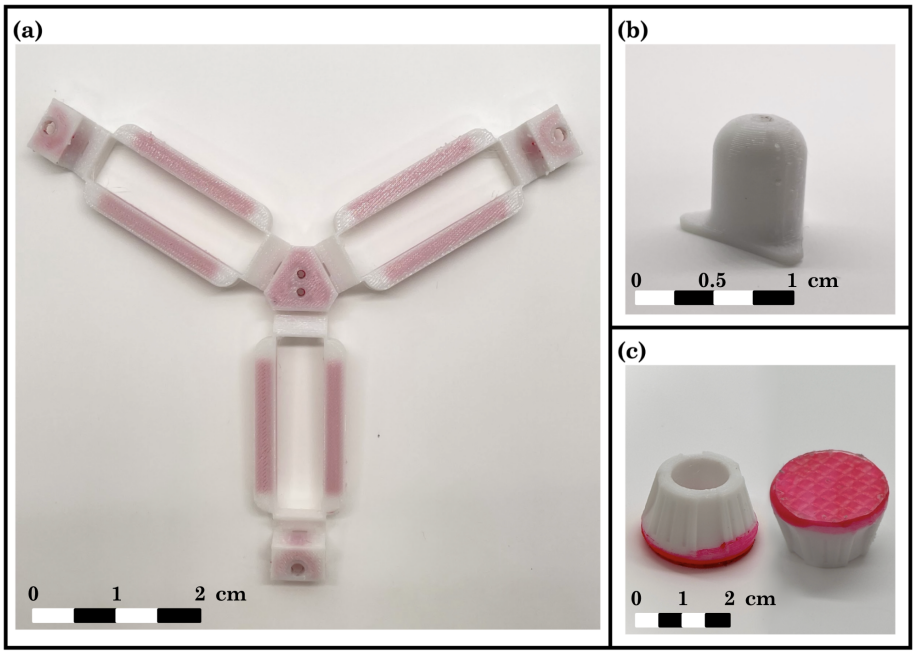

Figure 2.2: A single Delta robot is used to create a leg. The components are (a) the compliant link printed out of TPU and PLA, (b) the foot printed out of TPU, and (c) the shoe printed out of TPU with grip tape adhered to the base of the shoe. Dimensions are provided in the text.

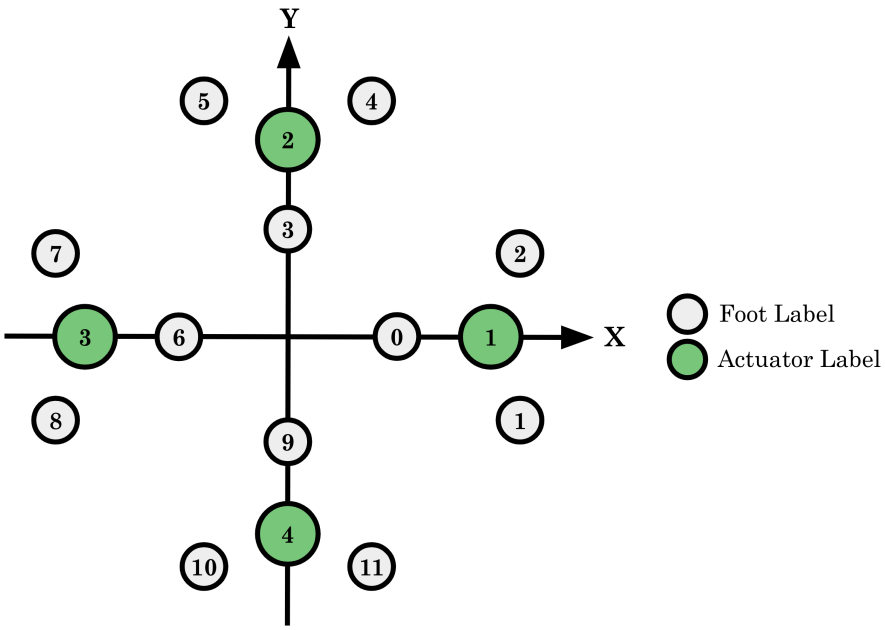

Figure 2.3: Diagram of each of the Delta foot labels and the actuators associated with that foot. Dimensions are provided in the text.

When the robot is viewed from above, the coordinate axes can be drawn as shown in Figure 2.3. The Deltas are arranged in a diamond pattern about the center of the robot, and each Delta center is located 4 cm from the center of the robot. They are labeled sequentially counterclockwise, with the Delta along the positive x-axis denoted as Delta 1. We further label each actuator used by the Deltas sequentially, starting with the actuator closest to the center followed by the actuators counterclockwise to it sequentially.

The Delta links are bolted to the linear actuators. We use the same 20 mm stroke length linear actuators (Actuonix PQ12-63-6-P [19]) as the DeltaHand because we wanted to perform our initial exploration with the same workspace (12.4 cm $\times$ 12.4 cm $\times$ 2.5 cm) [39]. These actuators are placed into the frames of the robot. We laser cut the frames of the robot and 3D printed supplemental spacers to hold the actuators and the PCB. The original DeltaHands used hexagon shaped frames, however this causes uneven weight distribution across the robot. Instead, these frames are cut into an octagon shape so that the weight distribution of the robot is equal. The electronics were changed from using Adafruit components to a custom PCB, making it more compact and lighter for the robot. This change was made by Zilin Si, and is a part of ongoing work. The new PCB is mounted on top of the robot. The frames and PCB are shown in Figure 2.4

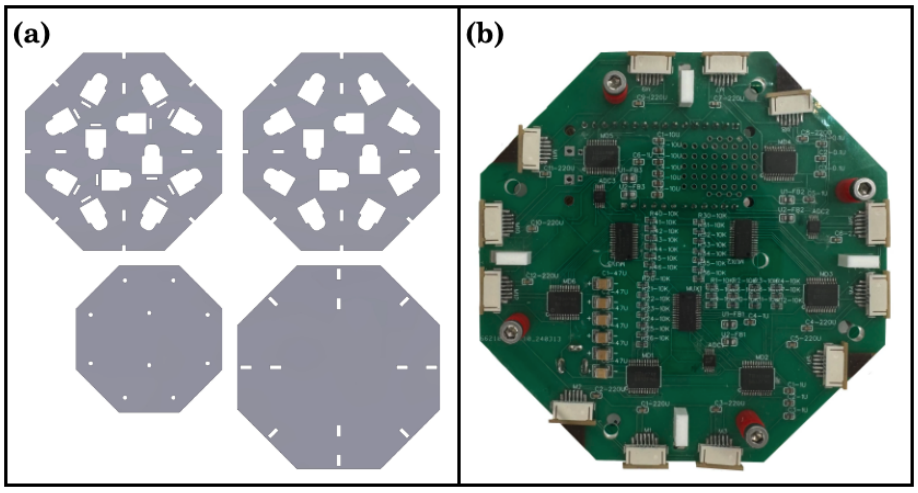

Figure 2.4: (a) CAD of the acrylic frames for the robot, (b) The custom PCB Board for the robot.

2.1.2 Variations

There are 3 variations of the robot (Figure 2.5). There are the 12 actuator, 9 actuator, and 5 actuator versions. Throughout the majority of this work, we primarily use the 12 actuator version of the robot. We only use the 5 and 9 actuator versions in Section 3.1.

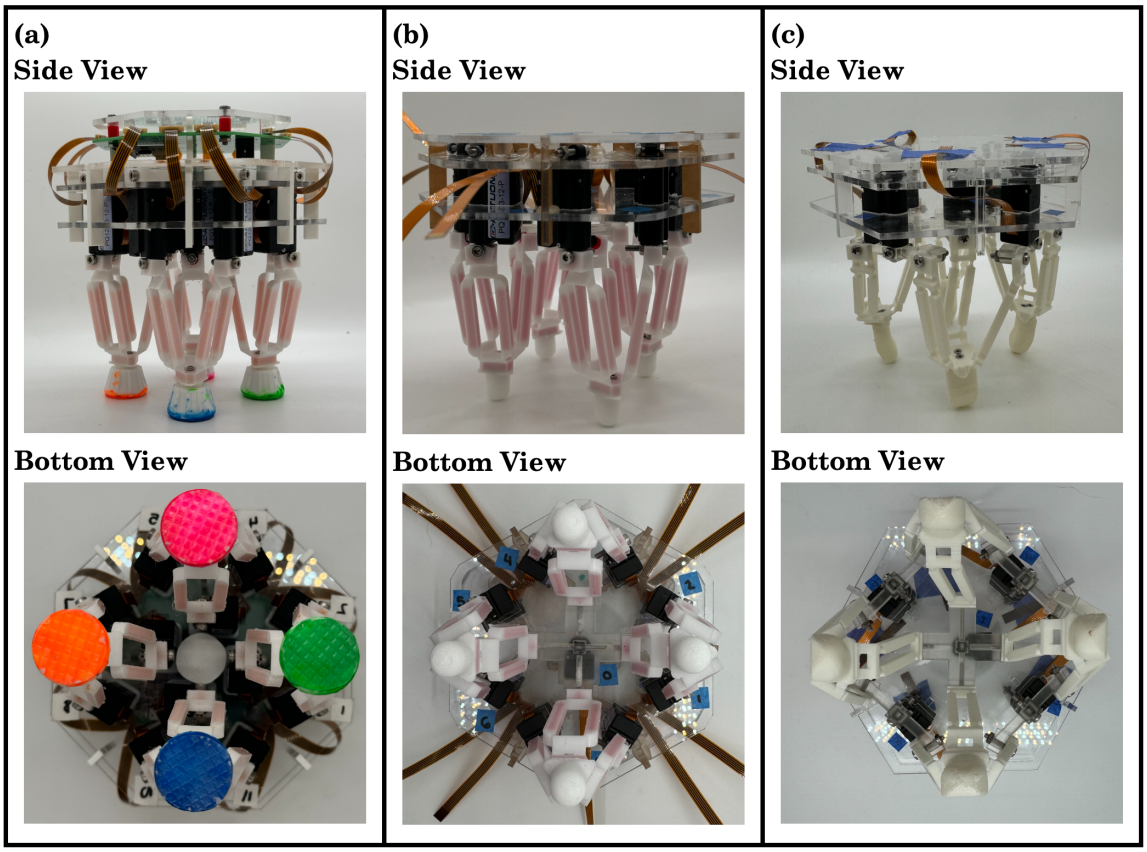

Figure 2.5: Side and bottom view of the (a) 12 actuator, (b) 9 actuator, and (c) 5 actuator versions of the robot.

The 12 actuator version has one actuator per link of a Delta robot. This allows for every foot to move independently in the workspace. The 9 actuator version couples the middle actuators of all of the Deltas onto one actuator. So moving the central actuator would move all of the Deltas, bringing them closer or further apart as the actuator moves in and out. The 5 actuator version additionally couples the outer actuators of adjacent Deltas. This results in the outer actuators moving two Deltas

at once. The 5 actuator version of the robot used the original DeltaHands design [39]. This means that it uses fingertips rather than feet and the Deltas are made entirely of TPU. The 9 and 12 actuator versions of the robot use the feet design design and dual-printed Deltas. The 12 actuator version of the robot also uses the shoes.

2.2 Software

To control the robot, we use existing control software of the DeltaHand [38] and of the DeltaArrays [7, 8, 33]. We generate trajectories of each foot's desired positions. We utilize the inverse kinematics of the Delta robot, detailed in Section 2.3 to calculate the desired motor positions, which are the waypoints of the trajectory. These waypoints are then passed from the computer to the robot via wired serial communication.

Upon receiving the trajectories, the robot uses the motor's maximum velocity and acceleration to calculate the total execution time for each motor to reach the desired waypoint. Since the distances each motor has to travel may vary, the total execution time varies as well. However, it is important to synchronize the motors such that they all arrive at their destinations at the same time so that the feet do not drag or push the robot in undesired directions. So the target velocities of each motor are calculated such that the maximum possible velocity of a motor is proportional to the ratio of the completion time for that motor to the longest completion time of all of the motors. The target velocities are converted into speed commands to the motor and used in combination with a closed loop PID controller that uses the desired and actual motor positions to command all the motors simultaneously every 10 ms. Once the motor position is within 1 mm of the desired position, the robot moves onto the next trajectory waypoint. To understand these calculations in more detail, please refer to [7].

2.3 Kinematics

Delta robot kinematics have closed-form unique solutions [43, 47]. These solutions are summarized in Section 2.3.1. This can be easily extended to finding the forward and inverse kinematics of the DeltaWalker for generating trajectories. We can take in motor positions and output end-effector positions for forward kinematics and the reverse for inverse kinematics. This will allow us to find the necessary positions of the robot— the four end-effector positions and the center of mass position— in the world frame.

2.3.1 Single Delta Kinematics

We know the kinematic solutions for a single Delta with respect to a single Delta's reference frame, as denoted by Figure 2.6. The primary components of a single Delta kinematics are the positions of the base platform, located at the origin of the coordinate axes; the position of the end-effector platform, located at a variable distance below the base platform; and the motor position values, restricted to be within the motor stroke length values.

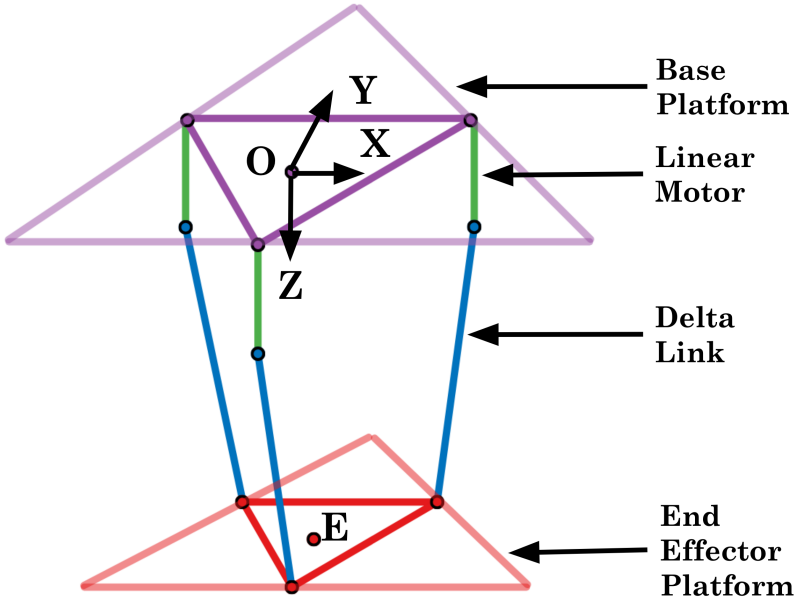

Figure 2.6: Kinematic reference frame for a single Delta robot.

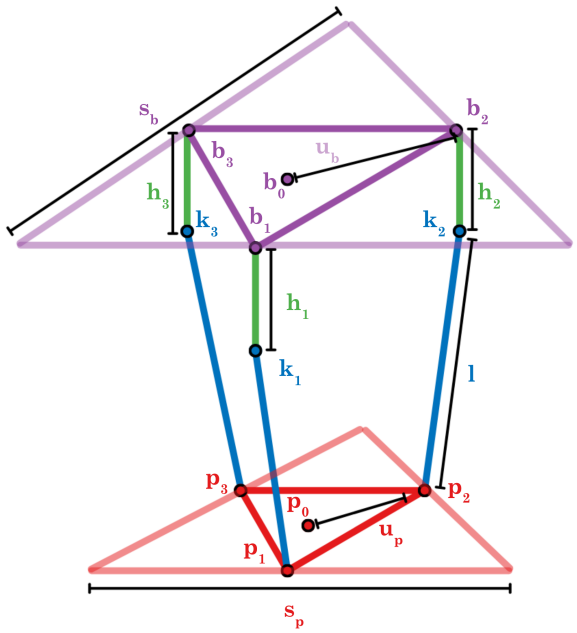

Figure 2.7: Variables used for calculating single Delta kinematics.

Variable(s)	**Descriptions**
b_0	Base platform origin position
b_1, b_2, b_3	Base vertices positions
k_1, k_2, k_3	Knee vertices positions
p_0	End-effector platform origin position
p_1, p_2, p_3	End-effector vertices positions
h_1, h_2, h_3	Motor actuation amounts
s_b	Side length of the base platform
u_b	Inradius of the base platform
s_p	Side length of the end-effector platform
u_p	Inradius of the end-effector platform
l	Delta link length (leg length)

Table 2.1: Variables and descriptions used for values of a single Delta platform.

We can define the points on Figure 2.7 with Table 2.1. Note that the base and end-effector platforms consist of large triangle with a smaller triangle inscribed such that the vertices of the inscribed triangle is at the midpoint of the outer triangle sides. For consistency, we will refer to the large triangle as the platform and the small triangle as the inscribed triangle for the corresponding platform.

The base vertices (b_1, b_2, b_3) correlate of each of the motor bases. The knee vertices (k_1, k_2, k_3) correlate to the connection between the motor and the Delta link. Finally, the end-effector vertices (p_1, p_2, p_3) correlate to where the link meets end-effector platform. All position values are in $\begin{bmatrix} x & y & z \end{bmatrix}^T$.

The relationship between the platform side lengths (s) and the platform inradius values (u) are as follows:

$$s = 2\sqrt{3}u \tag{2.1}$$

The positions of the base platform origin and vertices are fixed. The relationship between the base vertices and the base platform origin are as follows:

$$b_1 = b_0 + \begin{bmatrix} 0 & -u_b & 0 \end{bmatrix}^T \tag{2.2}$$

$$b_2 = b_0 + \begin{bmatrix} u_b \cos 30 & u_b \sin 30 & 0 \end{bmatrix}^T \tag{2.3}$$

$$b_3 = b_0 + \begin{bmatrix} -u_b \cos 30 & u_b \sin 30 & 0 \end{bmatrix}^T \tag{2.4}$$

Given our reference frame, we always assume $b_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$. The equations that relate the base vertices and base platform origin can similarly be applied to relate the end-effector vertices and the end-effector origin.

Inverse Kinematics

The objective of the inverse kinematics is to use a desired end-effector position to calculate motor actuation amounts. It is possible for this to yield no solution if the desired end-effector position is infeasible.

To calculate the inverse kinematics, we first calculate the squared XY distance of each end-effector vertices to the platform vertices. This can be done using the following general formula where $i = 1, 2, 3$:

$$d_i = (p_{i,x} - b_{i,x})^2 + (p_{i,y} - b_{i,y})^2 \tag{2.5}$$

Then we check if the end-effector center is a feasible distance away from the base platform by checking if $d_i \leq l^2$ for $i = 1, 2, 3$. If any i value results in the inequality being false, then the desired position is infeasible. If all 3 inequalities are true, then the motor actuation amounts can be calculated for $i = 1, 2, 3$ as follows:

$$h_i = -(p_{0,z} + \sqrt{l^2 - d_i^2}) \tag{2.6}$$

Forward Kinematics

The objective of the forward kinematics is to use desired motor actuation amounts to calculate the end-effector position. We can use the method of 3 intersecting spheres, as described in [43] and shown in Figure 2.8, to calculate the forward kinematics. It is possible for this to yield no solutions as well if the resulting position is infeasible.

We know the base vertices (b_1, b_2, b_3) positions are fixed, and we are given motor actuation amounts (h_1, h_2, h_3). This allow us to calculate the positions of the knee vertices (k_1, k_2, k_3) by adding the corresponding h value to the z component of each b. We can center the spheres at the knee vertices with a radius equivalent to the leg length l. We want these spheres to intersect at a single point, $p_0 = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^T$ to solve the forward kinematics problem. To do this, we shift the centers of the spheres inwards by the inradius of the platform triangle, u_p. This results in a triangle with the following inradius:

$$u_t = u_b - u_p \tag{2.7}$$

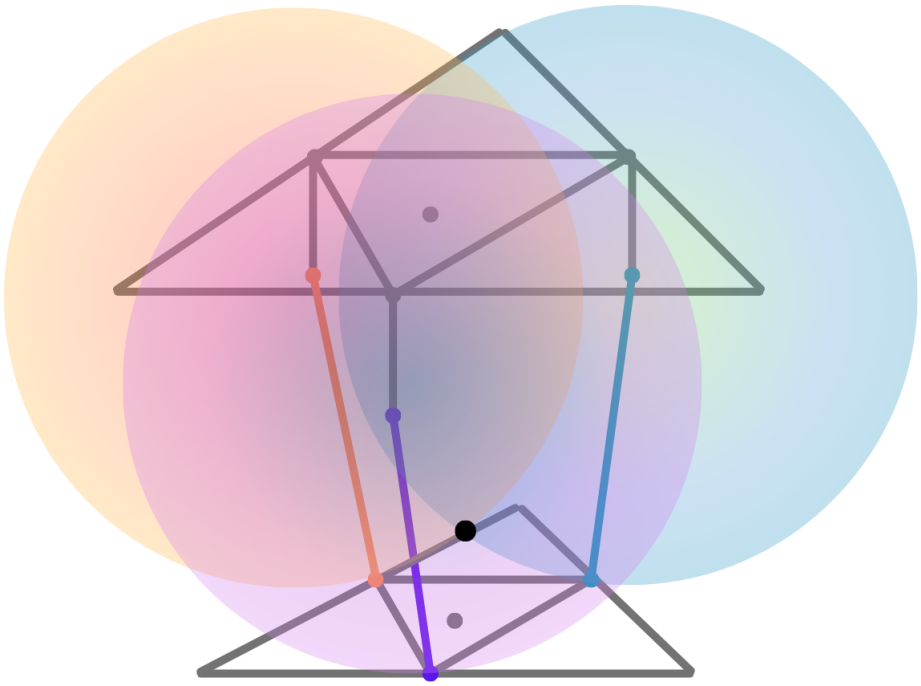

Figure 2.8: Example intersection of 3 spheres for a Delta actuator's forward kinematics.

The corner positions of this new triangle, centered at $t_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, can be denoted as t_1, t_2, t_3, are as follows:

$$t_1 = \begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}^T = \begin{bmatrix} 0 & -u_t & -h_1 \end{bmatrix}^T \tag{2.8}$$

$$t_2 = \begin{bmatrix} x_2 & y_2 & z_2 \end{bmatrix}^T = \begin{bmatrix} u_t \cos 30 & u_t \sin 30 & -h_2 \end{bmatrix}^T \tag{2.9}$$

$$t_3 = \begin{bmatrix} x_3 & y_3 & z_3 \end{bmatrix}^T = \begin{bmatrix} -u_t \cos 30 & u_t \sin 30 & -h_3 \end{bmatrix}^T \tag{2.10}$$

We use these values to solve a system of equations where the objective is to satisfy the requirement that the radius of the sphere is equivalent to the leg length. This is done with each t vertex, so we end up with the following system of equations:

$$l^2 = (x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2 \tag{2.11}$$

$$l^2 = (x_0 - x_2)^2 + (y_0 - y_2)^2 + (z_0 - z_2)^2 \tag{2.12}$$

$$l^2 = (x_0 - x_3)^2 + (y_0 - y_3)^2 + (z_0 - z_3)^2 \tag{2.13}$$

Keeping in mind that $x_1 = 0$, we can rearrange the system of equations as follows:

$$(w_1 - w_2)/2 = x_2 x_0 + (y_1 - y_2)y_0 + (z_1 - z_2)z_0 \tag{2.14}$$
$$(w_1 - w_3)/2 = x_3 x_0 + (y_1 - y_3)y_0 + (z_1 - z_3)z_0 \tag{2.15}$$
$$(w_2 - w_3)/2 = (x_2 - x_3)x_0 + (y_2 - y_3)y_0 + (z_2 - z_3)z_0 \tag{2.16}$$

Where the intermediate calculation value w_i for $i = 1, 2, 3$ is given by:

$$w_i = x_i^2 + y_i^2 + z_i^2$$

Further rearrangement yields the following:

$$x_0 = (a_1 z_0 + b_1)/d \tag{2.17}$$
$$y_0 - (a_2 z_9 + b_2)/d \tag{2.18}$$

Where the intermediate calculation values are:

$$\begin{aligned}
a_1 &= (z_2 - z_1)(y_3 - y_1) - (z_3 - z_1)(y_2 - y_1) \\
a_2 &= -(z_2 - z_1)x_3 + (z_3 - z_1)x_2 \\
b_1 &= \big((w_2 - w_1)(y_3 - y_1) - (w_3 - w_1)(y_2 - y_1)\big)/2 \\
b_2 &= \big((w_2 - w_1)x_3 - (w_3 - w_1)x_2)\big)/2 \\
d &= (y_2 - y_1)x_3 - (y_3 - y_1)x_2
\end{aligned}$$

Substituting in 2.17 and 2.18 to 2.11 results in the following:

$$a z_0^2 + b z_0 + c = 0 \tag{2.19}$$

Where the coefficients are:

$$\begin{aligned}
a &= a_1^2 + a_2^2 + d^2 \\
b &= 2(a_1 b_1 + a_2(b_2 - y_1 d) - z_1 d^2) \\
c &= (b_2 - y_1 d)(b_2 - y_1 d) + b_1^2 + d^2(z_1^2 - l^2)
\end{aligned}$$

These coefficients can be used to calculate the discriminant $D = b^2 - 4ac$ where if D is less than 0, then there are no real solutions so the desired forward kinematics point is unattainable. But if it is greater than 0, then z_0 can be solved for. This would yield the forward kinematics solution for the Delta.

2.3.2 Setup and Reference Frames

As mentioned in Section 2.1.1, we can consider the DeltaWalker as a series of four Delta robots arranged in a diamond pattern about the origin (Figure 2.3). We denote the center of mass of the robot (located towards the top of the robot) as the origin of the robot reference frame. These Deltas are labeled 1 through 4 starting from the Delta aligned with the positive x-axis and moving counterclockwise. Each Delta robot origin is placed a fixed distance away from the robot origin and along either the x- or the y-axes. They are rotated about the robot origin such that the Delta's y-axis will always point outwards along the axis that specific Delta is on, as shown in Figure 2.9 The locations of the base platform and the end-effector platforms are displaced in the x- and y- directions as well.

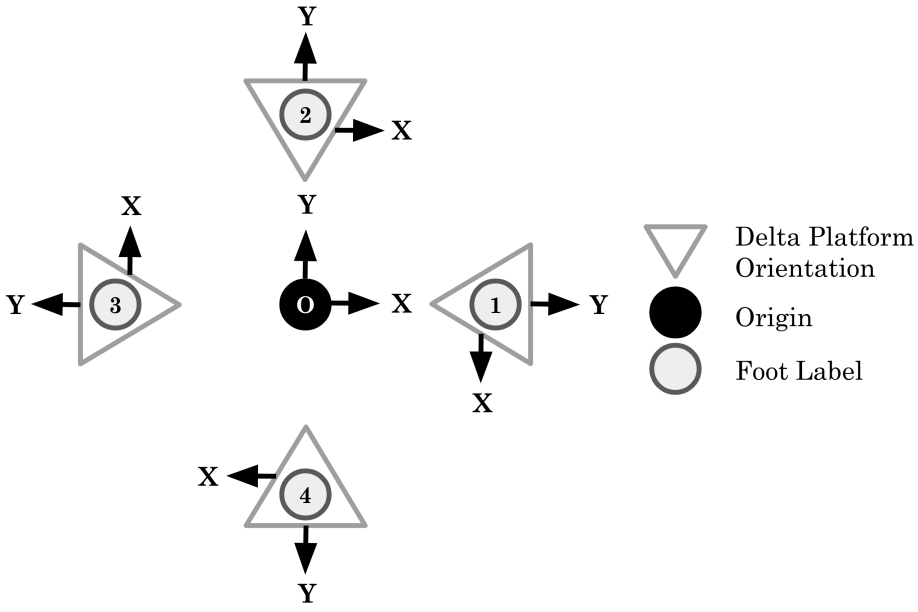

Figure 2.9: Kinematic platform rotations of each Delta robot with the origin as a reference.

However, the robot reference frame denotes the $z = 0$ plane as the COM of the robot whereas in the world frame, the $z = 0$ plane is denoted as the ground, as shown in Figure 2.10. So in the robot (and the Delta) reference frame, the end-effector positions are in the negative z-axis. Furthermore, as the robot takes additional steps, the position of the center of mass (COM) in the world frame changes. To translate the robot from the robot frame to the world frame, we need to account for these offsets. We need to subtract the most negative z-value from all of the positions of the robot and add the known COM position of the robot in the world frame to all of the x- and y-positions of the robot.

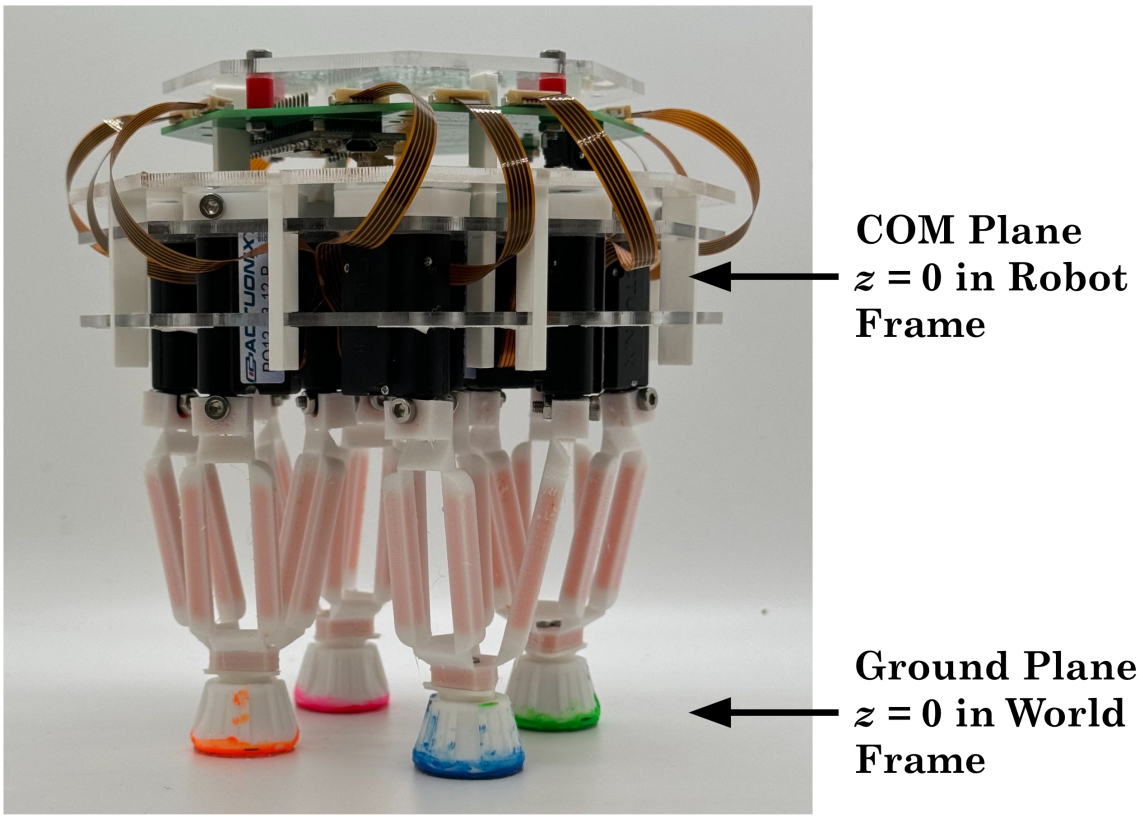

Figure 2.10: The references frames of the robot, where the robot frame $z = 0$ plane is on the COM plane and the world frame $z = 0$ plane is on the ground plane.

2.3.3 DeltaWalker Kinematics

The forward and inverse kinematics of the DeltaWalker can be calculated using the forward and inverse kinematics of a single Delta robot, as detailed in 2.3.1. Because each Delta robot on the DeltaWalker is rotated and displaced from a Delta located at the origin, we need to account for these transformations when performing forward and inverse kinematics on the Delta.

For inverse kinematics, we know the desired end-effector position in the world frame. We first subtract the position of the COM of the robot in the world frame from the positions of the COM and end-effectors of the robot. If the desired COM is known, then that is the value we use as the COM of the robot in the world frame. If it is not known, then the position of the COM in the world frame is the previous COM position. Then, for each Delta, we can convert the individual leg from the robot frame to the reference frame for a single Delta robot. Then, we can calculate the desired motor positions.

For forward kinematics, we must know the desired motor positions. Using this, we can calculate the updated position of each of the end-effectors in the individual Delta robot frames. Then we perform rotations and translations to convert these values in the robot frame. Finally, we can add the current COM of the robot to the positions of the COM and end-effectors in the robot frame to determine the position of the robot in the world frame.

Chapter 3

Gait Design

3.1 Preliminary Gait Exploration

We want to explore if coupling the links of the Deltas would be feasible for creating walking gaits because we want to use as few actuators as possible to allow the robot to be lighter. For this, we must first know if the robot can lift a single leg. Then, we can attempt to create walking gaits. To investigate, we use simple position commands to each of the actuators.

We start with the 5 actuator version of the robot. We attempt to lift one leg independently. However, this is not possible. The links are overly coupled so moving the center actuator causes all 4 Deltas to move simultaneously and moving an actuator causes 2 Deltas to move simultaneously. As a result, the robot legs slide rather than lift.

Since the 5 actuator version of the robot cannot be used for lifting a leg, we experiment with using the 9 actuator version of the robot. We are able to lift single legs independently. Furthermore, we are able to generate a basic step that could be repeated and shift the robot forwards. One challenge with using the 9 actuator version of the robot is the movements are not intuitive.

To better generate walking gaits, we experiment with the 12 actuator version of the robot. We are able to lift single feet since they are fully independent. We are able to generate a basic stepping pattern that could move the robot forwards.

Through this preliminary exploration, we learn that it is possible to create steps

and walking gaits with the 9 and 12 actuator versions of the robot but it is not possible with the 5 actuator version. However, to better explore various gaits, we need to utilize inverse kinematics and design gaits based on the desired end-effector positions of the robot and using inverse kinematics to solve for the corresponding motor positions. Furthermore, we realize that initial exploration in the real world is slow. So it is worthwhile to test trajectories in simulation first before testing them in the real world to explore more trajectories quickly.

3.2 Gaits and Robot Orientations

Since a Delta robot can move with 3 translational degrees of freedom, the robot should be omnidirectional. This means that the robot does not need to be restricted to any specific orientation. Thus, we use two different orientations when designing the robot gaits. The first orientation matches a standard quadruped, with two front feet and two rear feet. This is denoted as FB (front/back) orientation. This orientation is designed to move the robot diagonally between the x- and y-axes. The second orientation is set such that there is one front foot, two side feet, and one rear foot. This is denoted as FBS (front/back/side) orientation. This orientation is designed to move the robot along the x- or y- axes. The distinction between these orientations and their corresponding gaits are shown in Figure 3.1. The foot movement order varies per gait.

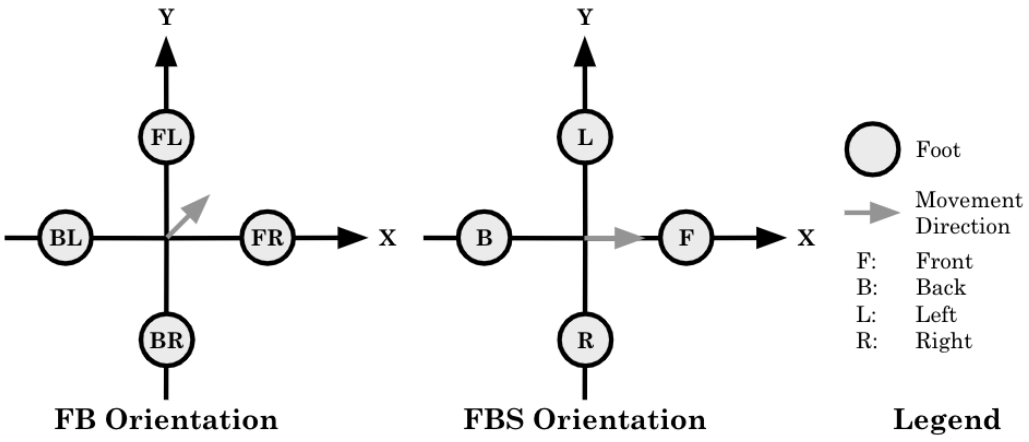

Figure 3.1: The FB and FBS orientations of the robot can be set up in the same way but with different foot labelings and movement directions.

The numbering of each foot remains consistent. However, due to the omnidirectional nature of the robot and the different orientations of the robot, we need to designate different Deltas as the front left, front right, front, etc. legs. This allows us to think of moving in any direction as moving forwards in that direction. These orientation and direction combinations are shown in Table 3.1.

	FB Orientation		**FBS Orientation**	
Label	**Quadrant**	**Order**	**Direction**	**Order**
1	1	1 → FR, 2 → FL, 3 → BL, 4 → BR	+ x	1 → F, 2 → L, 3 → B, 4 → R
2	2	1 → BR, 2 → FR, 3 → FL, 4 → BL	+ y	1 → R, 2 → F, 3 → L, 4 → B
3	3	1 → BL, 2 → BR, 3 → FR, 4 → FL	− x	1 → B 2 → R, 3 → F, 4 → L
4	4	1 → FL, 2 → BL, 3 → BR, 4 → FR	− y	1 → L, 2 → B, 3 → R, 4 → F

Table 3.1: Depending on the desired direction of movement, the designation of which foot is the front (F), back (B), left (L), right (R), or combination of front/back and left/right foot varies.

We use two methods for generating the gaits: manually designing gaits and trajectory optimized gaits. These methods are further outlined in Section 3.3 and Section 3.4 respectively. These two methods result in slightly different gait designs.

The FB gaits are Walk and Diagonal Amble, shown in Figures 3.2 and 3.4. The Walk gait is chosen because it is the most standard quadruped gait and it is quasi-statically stable. The Diagonal Amble gait, referred to as Amble for short, is another quasi-statically stable gait but with a different foot order. We choose this gait to experiment with different foot ordering.

It is worth noting that the foot order for the Amble gait is different for the manually designed trajectories and the trajectory optimization trajectories. The former starts with a front foot and the latter starts with a rear foot. We opt to switch to using the rear foot first for the primary trials of the trajectory optimization gaits to be consistent with literature [Y] and with the Walk trajectory starting foot.

The FBS gaits are S-Gait and C-Gait, both shown in Figures 3.3 and 3.4. The S-Gait is an adapted version of the Modified Tripedal gait presented by Harvard [11]. This gait, intended for FB orientation, is meant to use the rear legs to push the front legs forward. The manually designed version of this gait uses the same idea but adapted to this orientation. Then as we switch to trajectory optimization, the foot order remains the same but the weight shifts are solved for automatically rather than designed to specifically use the rear legs to push the front legs forwards.

The C-Gait is a slight variation of this gait. After some preliminary testing detailed in Appendix A, we are interested in seeing if the slight change in foot order would improve the performance of the gait in the FBS orientation. However, we did not pursue this trajectory as a manually designed option as it was created after the manual design experiments. The results of those experiments, outlined in Section 3.1, indicated that it was not worthwhile.

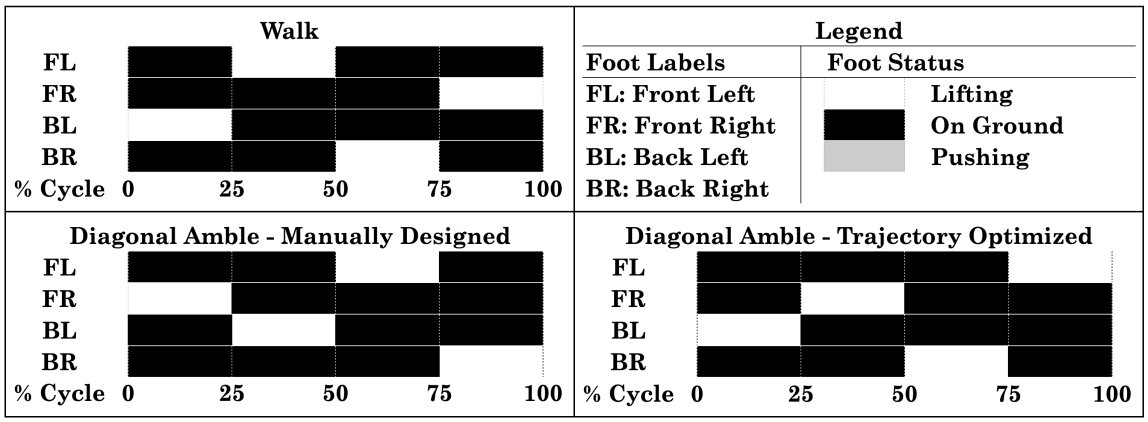

Figure 3.2: Diagrams of the timings for the FB orientation gaits: Walk and Diagonal Amble (Amble for short). For Amble, there are two variations of the foot order, one for the manually designed gait and one for the trajectory optimized gait.

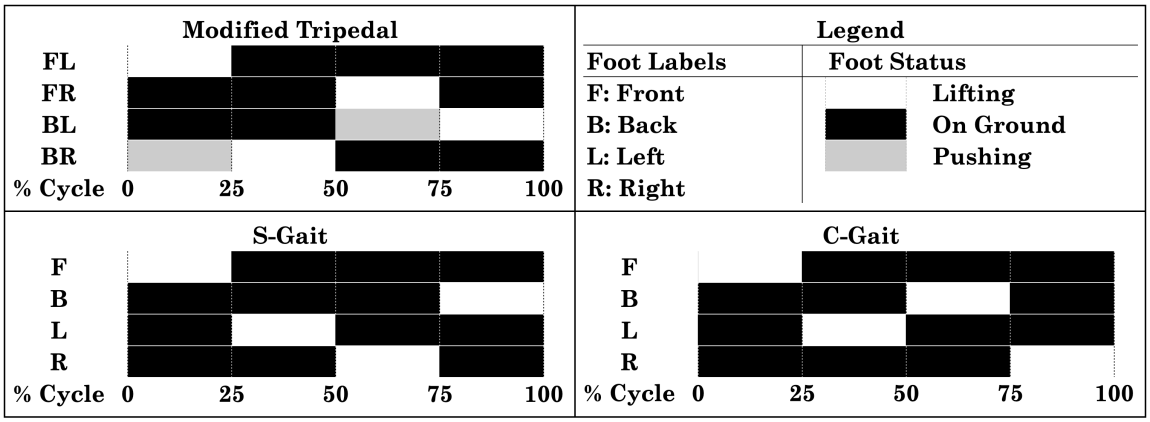

Figure 3.3: Diagrams of the timings for the FBS orientation gaits: S-Gait and C-Gait. The Modified Tripedal gait is also visualized as it was the inspiration for the FBS orientation gaits.

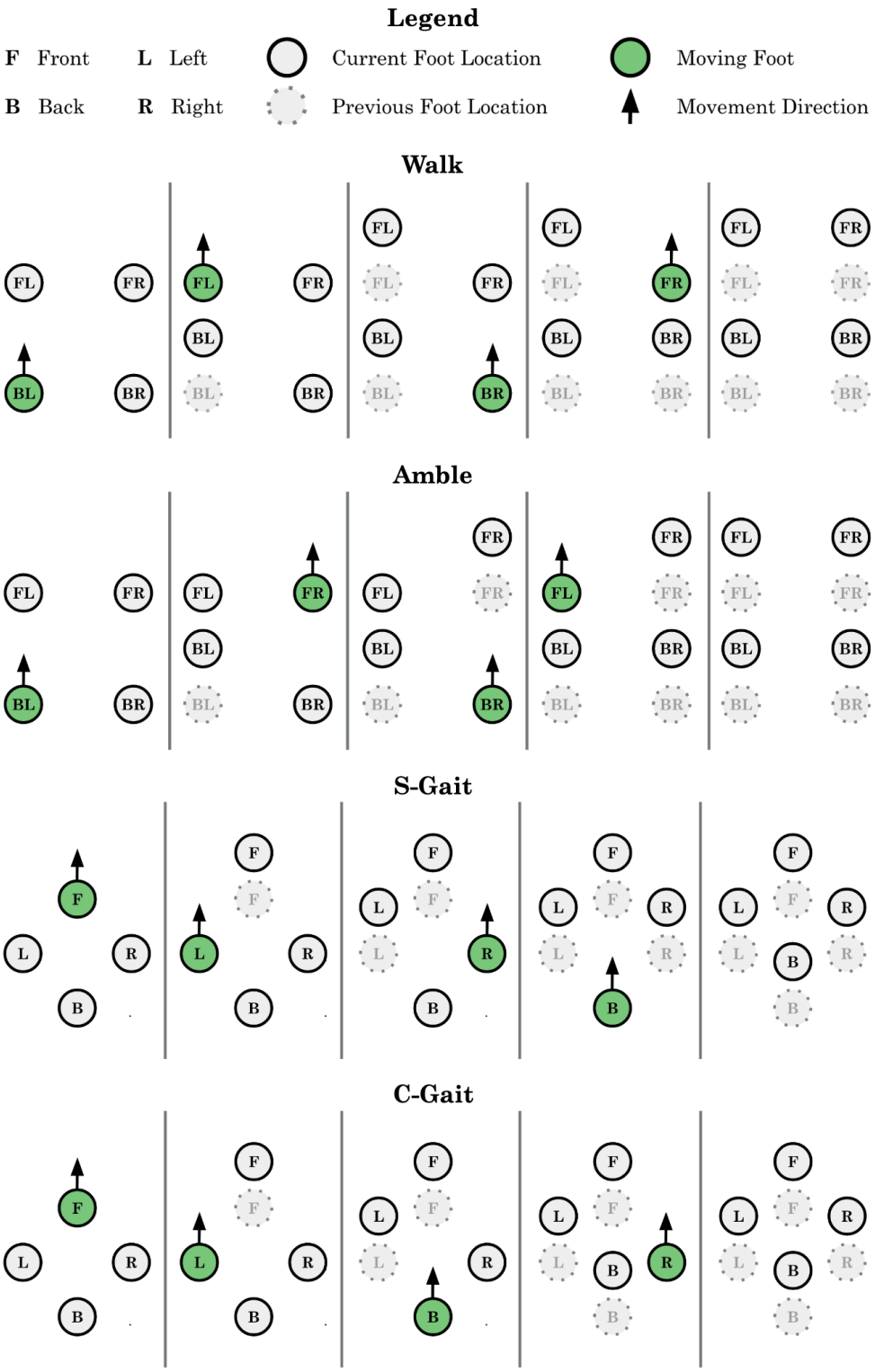

Figure 3.4: Foot stepping order diagram for all 4 gaits: Walk, Amble, S-Gait, and C-Gait.

3.3 Manually Designed Gaits

We first manually design trajectories to generally explore the capabilities of the robot and gain intuition of how different gait patterns affect the robot.

3.3.1 Method

To generate the manually designed trajectories, we first create a dictionary of all of the possible trajectory segments that might be used by each of the Deltas. These trajectory segments all have a set number of steps. They are defined by starting and ending displacements of the end-effector, where the displacements are relative to a Delta that has been fully extended in the world frame. For example, if all of the motors on a Delta were fully extended, the displacement of the end-effector would be $[0, 0, 0]$ m. However, if all of the motors of the Delta were fully retracted, the position of the end-effector would be $[0, 0, 0.02]$. So the position of the motor can be calculated with $0.02 - z$. If the end-effector was intended to be displaced 1 cm in the positive x-, y-, and z-directions according to the world coordinate axes, the position of the end-effector would be denoted as $[0.01, 0.01, 0]$ m. These displacement positions can be broken up into two components, the xy-component and the z-component. The xy-component has 3 main positions, as shown in Table 3.2. The z-component can be broken up into 4 main positions, as shown in Table 3.3. The resulting position can be referenced as combinations of the xy-position name and the position name. For example, $[0, 0, 0.015]$ m would be the home-base position.

It is worth noting that although the full extension of the motor would really result in a position of $z = 0$ cm, we notice that the robot tips easily when the end-effector is fully extended and therefore we elect to use less motor extension. Furthermore, the kinematics of a Delta robot limits the position of the end-effector such that $z = 0$ can only result in x- and y-values being 0 as well. The trajectory segments and their descriptions can be found in Table 3.4.

To account for the different directions of the trajectories, the desired step position in the x- and y-directions are passed into the trajectory generation. The orientation and direction combination is recognized, as per Table 3.1, and the Delta assigned to each foot is adjusted accordingly.

Name	XY-Position	Description
Home	(0, 0)	The center of the end-effector platform is aligned with the center of the base platform.
Step	(+x, +y)	The center of the end-effector platform is in the positive xy-direction relative to the center of the base platform, for stepping the robot forwards.
Lean	(−x, −y)	The center of the end-effector platform is in the negative xy-direction relative to the center of the base platform, for leaning the robot forwards.

Table 3.2: Descriptions of each of the xy-positions.

Name	Z-Position	Motor Position	Description
None	0.0195	0.0005	Z-position for a lifted foot, lower limit of z-positions.
Base	0.015	0.005	Default z-position when the robot is not taking a step.
Part	0.013	0.007	Z-position to push the robot slightly forwards.
Full	0.011	0.009	Z-position to push the robot more forwards, upper limit of z-positions.

Table 3.3: Descriptions of each of the z-positions.

We consider one iteration of the trajectory to be once every foot has moved one step. To create multiple iterations of these trajectories, we first initialize the robot to its home-base position and have it wait there for a moment. Then we append the first segment of the trajectory before appending the repeating section of the trajectory. Next, we either append the first segment of the next trajectory or we append the last segment of the current trajectory. This allows the trajectory to cycle continuously such that there is not a pause after each foot has taken a step.

When calculating motor positions, these displacements are first converted into the robot reference frame. Then the inverse kinematics can be calculated.

Name	Description	Label	Start	End
Push	Push the foot down to home base	A	Home-None	Home-Base
Lift	Lift the foot up to Home-None to prepare to take a step	A	Home-Base	Home-Base
		B	Lean-Part	
		C	Lean-Full	
Step	Step in the xy-direction while moving z-position down	A	Home-None	Step-Base
Zero	Reset to Home-Base position after taking a step	A	Step-Base	Home-Base
Wait	Wait at the current position	A	Home-Base	Home-Base
		B	Lean-Part	Lean-Part
		C	Lean-Full	Lean-Full
		D	Step-Base	Step-Base
Lean	Lean robot forward form Home-Base	A	Home-Base	Lean-Base
		B		Lean-Part
		C		Lean-Full

Table 3.4: Descriptions of each of the manually designed trajectory segments.

3.3.2 Trajectories

The manually designed trajectories are Walk, Amble, and S-Gait. We do not find it worthwhile to pursue a manually designed version of the C-Gait trajectory from our findings with the previous gaits. Each trajectory generates a step size of 1 cm total. For the Walk and Amble gaits, the displacements in the x- and y-directions are $1/\sqrt{2}$ cm. For the S-Gait, the displacement is 1 cm in the x- or the y-direction.

For the Walk and Amble gaits, we create 3 different versions of the gait. These versions vary by when the feet reset to the home-base positions. To explain the differences between the versions, the foot that has just stepped is denoted as Foot A and the foot that is going to take a step next is denoted as Foot B. They are additionally visualized in Figures 3.5 and 3.6.

In version 0, Foot A resets back to the home-base position immediately after it

has stepped to the step-base position. Foot B begins to lift while Foot A resets. In version 1, Foot A waits for Foot B to raise lift to the home-none position before it resets to the home-base position. This means that it waits for one trajectory segment before resetting. In version 2, Foot A waits for Foot B to to lift and then step down to its step-base position before resetting. This means that it waits for 2 trajectory segments before resetting. For version 1 and 2, Foot A waits in the step-base position while Foot B moves.

The S-Gait also has 3 versions. In version 0, the back leg extends fully to lean forwards as the front leg steps down. In version 1, the back leg extends partially to lean forwards as the front leg steps down. Finally, in version 2, the back leg extends fully and the side legs extend partially to lean forwards as the front leg steps down. These are further visualized in Figure 3.7.

These trajectories are summarized in Table 3.5.

Name	**Orientation**	**Version**	**Description**
Walk	FB	0	No delay prior to foot resetting
		1	1 segment delay prior to foot resetting
		2	2 segment delay prior to foot resetting
Amble	FB	0	No delay prior to foot resetting
		1	1 segment delay prior to foot resetting
		2	2 segment delay prior to foot resetting
S-Gait	FBS	0	Back leg extends fully
		1	Back leg extends partially
		2	Back leg extends fully, side legs extend partially

Table 3.5: Descriptions of each of the manually designed trajectories (Walk, Amble, S-Gait) and the variation between each version of each gait.

Walk V0												
FL	Push A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A	
FR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	
BL	Push A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	
BR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	
Segment	1	2	3	4	5	6	7	8	9	10	11	

Walk V1												
FL	Push A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Zero A	Wait A	Wait A	Wait A	Wait A
FR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Zero A
BL	Push A	Wait A	Lift A	Step A	Wait D	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A
BR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Zero A	Wait A	Wait A
Segment	1	2	3	4	5	6	7	8	9	10	11	12

Walk V2													
FL	Push A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Wait D	Zero A	Wait A	Wait A	Wait A	Wait A
FR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Wait D	Zero A
BL	Push A	Wait A	Lift A	Step A	Wait D	Wait D	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A
BR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Wait D	Zero A	Wait A	Wait A
Segment	1	2	3	4	5	6	7	8	9	10	11	12	13

Figure 3.5: Diagram of each segment of the manually designed Walk trajectory to show the difference between the versions.

Amble V0												
FL	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	
FR	Push A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	
BL	Push A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A	
BR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	
Segment	1	2	3	4	5	6	7	8	9	10	11	

Amble V1												
FL	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Zero A	Wait A	Wait A
FR	Push A	Wait A	Lift A	Step A	Wait D	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A
BL	Push A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Zero A	Wait A	Wait A	Wait A	Wait A
BR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Zero A
Segment	1	2	3	4	5	6	7	8	9	10	11	12

Amble V2													
FL	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Wait D	Zero A	Wait A	Wait A
FR	Push A	Wait A	Lift A	Step A	Wait D	Wait D	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A
BL	Push A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Wait D	Zero A	Wait A	Wait A	Wait A	Wait A
BR	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Wait D	Wait D	Zero A
Segment	1	2	3	4	5	6	7	8	9	10	11	12	13

Figure 3.6: Diagram of each segment of the manually designed Amble trajectory to show the difference between the versions.

S-Gait V0											
F	Push A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A
B	Push A	Wait A	Wait A	Wait A	Lean C	Wait C	Wait C	Wait C	Lift C	Step A	Zero A
L	Push A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A
R	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A
Segment	1	2	3	4	5	6	7	8	9	10	11
S-Gait V1											
F	Push A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A
B	Push A	Wait A	Wait A	Wait A	Lean B	Wait B	Wait B	Wait B	Lift C	Step A	Zero A
L	Push A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A
R	Push A	Wait A	Wait A	Wait A	Wait A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A
Segment	1	2	3	4	5	6	7	8	9	10	11
S-Gait V2											
F	Push A	Wait A	Lift A	Step A	Zero A	Wait A	Wait A	Wait A	Wait A	Wait A	Wait A
B	Push A	Wait A	Wait A	Wait A	Lean C	Wait C	Wait C	Wait C	Lift C	Step A	Zero A
L	Push A	Wait A	Wait A	Wait A	Lean B	Lift B	Step A	Zero A	Wait A	Wait A	Wait A
R	Push A	Wait A	Wait A	Wait A	Lean B	Wait B	Lift B	Step A	Zero A	Wait A	Wait A
Segment	1	2	3	4	5	6	7	8	9	10	11

Figure 3.7: Diagram of each segment of the manually designed S-Gait trajectory to show the difference between the versions.

3.4 Trajectory Optimized Gaits

One drawback to the manually designed trajectories is that we do not account for the position of the COM in those trajectories. This means that the robot may become unbalanced. We elect to use trajectory optimization to design gaits so that we can account for the position of the robot's COM. This allows us to ensure that the robot does not tip by enforcing that the robot COM stays within the support polygon of the robot (the polygon created by the feet in contact with the ground).

3.4.1 Method

We use the Interior Point OPTimizer solver (IPOPT) as it is suited for large-scale nonlinear optimization of continuous systems.

Setup

We represent the system with 5 point masses, one for the robot COM and one for each foot of the robot. The state uses the positions and velocities of each of the point masses in the x-, y-, and z-axes, resulting in a total of 12 degrees of freedom and 30 state variables. There are four control inputs which correspond to the internal force

of the system. This is the force on each foot from the COM. The system dynamics additionally accounts for the force due to gravity and the force of the contact with the ground on the feet. The state and control vectors are as follows:

$$x = \begin{bmatrix} p^{(b)} \\ p^{(1)} \\ p^{(2)} \\ p^{(3)} \\ p^{(4)} \\ v^{(b)} \\ v^{(1)} \\ v^{(2)} \\ v^{(3)} \\ v^{(4)} \end{bmatrix} \quad u = \begin{bmatrix} F^{(1)} \\ F^{(2)} \\ F^{(3)} \\ F^{(4)} \end{bmatrix} \tag{3.1}$$

where $p^{(i)}$ is the position of the body or foot i in each axis, $v^{(i)}$ is the velocity of the body or foot i in each axis, and $F^{(i)}$ is the force along leg i.

We formulated our system dynamics as a linear equation governing forces:

$$\begin{bmatrix} m_b I & & & & \\ & m_f I & & & \\ & & m_f I & & \\ & & & m_f I & \\ & & & & m_f I \end{bmatrix} \begin{bmatrix} \dot{v_b} \\ \dot{v_{f_1}} \\ \dot{v_{f_2}} \\ \dot{v_{f_3}} \\ \dot{v_{f_4}} \end{bmatrix} = \begin{bmatrix} -\ddot{m}_b g \\ -\ddot{m}_{f_1} g \\ -\ddot{m}_{f_2} g \\ -\ddot{m}_{f_3} g \\ -\ddot{m}_{f_4} g \end{bmatrix} + \begin{bmatrix} -I & -I & -I & -I \\ I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} \tag{3.2}$$

where m_b is the mass of the body, m_f is the mass of each foot, and λ_i is the contact force on foot i.

We experiment with two different cost functions initially and found that the second cost function (Equation 3.4) solves 2.5 times faster than the first cost function

(Equation 3.3). The first cost function uses a cost term for the distance between the current state and the goal state and a cost term for the internal forces. This encourages the solver to reach the goal state quickly and minimize the effort needed by the actuators. The second cost function omits the force constraint, and still produces the same trajectory. Thus, we move forward with using the second cost function.

$$J(x_{1:N}, u_{1:N-1}) = \sum_{i=1}^{N-1} \left[\frac{1}{2} ||x_i - x_g||_2^2 + \frac{1}{2} ||u_i||_2^2 \right] + \frac{1}{2} ||x_N - x_g||_2^2 \tag{3.3}$$

$$J(x_{1:N}) = \sum_{i=1}^{N} \frac{1}{2} ||x_i - x_g||_2^2 \tag{3.4}$$

To designate the foot ordering, we use a predefined set of knot points that determine which foot is lifted off the ground at which time. These points are stored in vectors M0, M1, M2, M3, and M4. M0 corresponds to when all four feet are on the ground, allowing the robot to shift its weight to remain within the support polygon. M1-M4 corresponds to one foot being off the ground to indicate taking a step.

Constraints

We use multiple constraints for the optimization problem. All of the constraints are linear, except for the body torque equality constraint (Equation 3.15). These are applied to the optimization problem as follows:

$$\min_{x_{1:N}} \quad J(x_{1:N})$$

Initial and goal state constraint

Initial and goal state constraints are used to designate the start and end states of the trajectory.

$$x_1 = x_{ic} \tag{3.5}$$

$$x_N = x_g \tag{3.6}$$

Dynamics constraint

The dynamics equality constraint is used to ensure that the system adheres to the model dynamics.

$$x_{k+1} = \text{dynamics}(x_k, u_k, \lambda_k) \quad \text{for } k \in [1, N-1] \tag{3.7}$$

Body height constraint

There is a primal inequality constraint on the z-height of the COM to prevent the body from falling to the ground or flying into the sky. The maximum range of motion for the COM is between 6.8 cm and 8.8 cm, but we specifically constrain the z-height of the body to be within 7.3 cm and 8.05 cm to ensure the robot can lift its legs off the ground.

$$0.053 \leq p_z^{(b)} \leq 0.0605 \qquad \text{for body z-height, } k \in [1, N] \tag{3.8}$$

Foot height constraint

There is a foot height inequality constraint to prevent the foot from falling below the ground.

$$p_z^{(i)} \geq 0 \quad \text{for i=1,2,3,4, } k \in [1, N] \tag{3.9}$$

Foot speed inequality constraint

There is a foot speed inequality constraint to prevent the robot from moving beyond the maximum speed of 15 mm/s of the linear actuators.

$$||v^{(i)}||_2 \leq 0.015 \quad \text{for i=1,2,3,4, } k \in [1, N] \tag{3.10}$$

Grounded foot velocity constraint

There is a velocity equality constraint that ensures that the feet that are touching the ground have zero velocity. This ensures that the solver does not assume frictionless contact. This assumption eliminates the need for a friction cone constraint. We assume that the contact forces from the motor are small in comparison to the body

mass, so the grounded feet should not slide.

$$||v^{(i)}||_2 = 0 \quad \text{for grounded foot}, k \in [1, N] \tag{3.11}$$

Grounded foot z-height constraint

To ensure that the foot never lifts more than one foot, we enforce a primal bound z-height constraint on the grounded feet to prevent them from lifting. Without this constraint, the solver creates trajectories where multiple feet are lifted.

$$p_z^{(i)} \leq 0.00025 \quad \text{for the grounded feet}, k \in [1, N] \tag{3.12}$$

Ground contact force constraint

We apply constraints such that the contact forces are nonnegative when the foot is in contact with the ground.

$$\lambda_z^{(i)} \geq 0 \quad \text{for i=1,2,3,4}, \; k \in [1, N-1] \tag{3.13}$$

Floating foot contact force constraint

We use a primal bound to ensure that the lifted feet no normal force from the ground.

$$\lambda_z^{(i)} = 0 \quad \text{for lifted feet } k \in [1, N-1] \tag{3.14}$$

Body torque equality constraint

To prevent the robot from tipping, we apply a net-zero torque constraint about the COM of the body to ensure that it remains within the support polygon and does not tip over.

$$\sum_{i=1}^{4} \lambda^{(i)} \times (p^{(i)} - p^{(b)}) = 0 \quad \text{for } k \in [1, N-1] \tag{3.15}$$

Foot position inequality constraints

To constrain the movement of the foot in the air, we create an inequality constraint based on the position of the actuator relative to the position of the foot. We model this constraint as a cylinder oriented vertically along the z-axis. This constraint is

based on the workspace of the robot's Deltas, which can be visualized as the space between two upward facing convex paraboloids, shown in Figure 3.8. $(p_z^{(a_i)} - 0.048)$ serves as an offset for the z-bounds of the cylinder. The cylinder is set up such that it must always intersect the ground plane so that it does not violate the other foot height constraints.

$$\|p_{x,y}^{(a_i)} - p_{x,y}^{(i)}\|_2 \leq 0.0135 \tag{3.16}$$

$$p_z^{(i)} \geq -0.0025 + (p_z^{(a_i)} - 0.048) \tag{3.17}$$

$$p_z^{(i)} \leq 0.0075 + (p_z^{(a_i)} - 0.048) \tag{3.18}$$

$$\text{for foot } i, \text{ for } k \in [1, N]$$

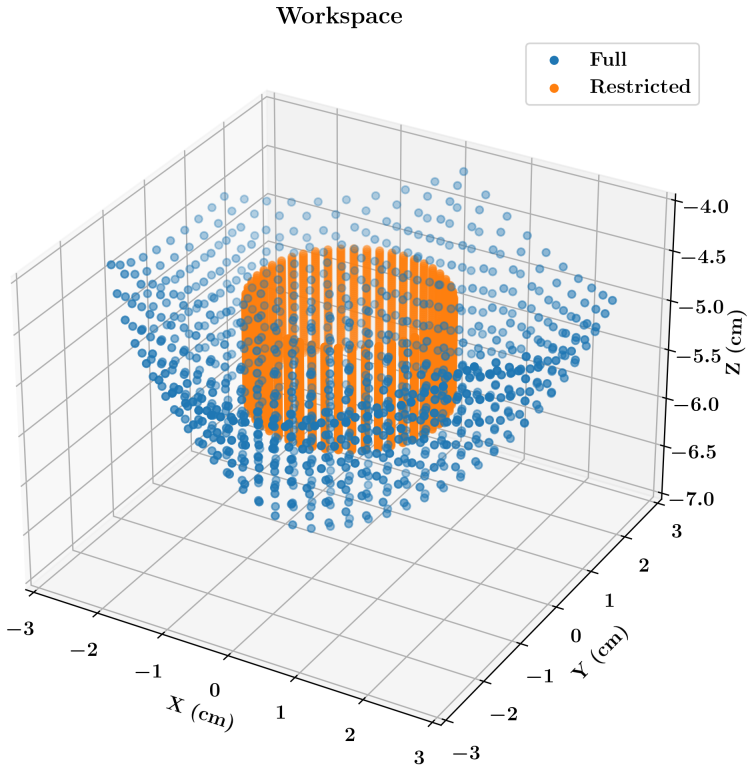

Figure 3.8: The full workspace of the Delta actuator compared to the restricted workspace used for the trajectory optimization problem constraint.

Results

The generated trajectories result in positions for the COM and each of the feet. These trajectories are generated in world coordinates, so they need to be converted into robot coordinates before using inverse kinematics to calculate the motor positions.

Each of the trajectories are generated to move in the positive x- and y-directions for FB orientation gaits and in the positive x-direction for the FBS orientation gaits. Since the robot is omnidirectional, we can rotate the trajectory 90^{o}, 180^{o}, and 270^{o} to create trajectories in the other orientation and direction combinations, as per Table 3.1. From there, the Delta assigned to each foot is adjusted accordingly before using the inverse kinematics to calculate the motor positions.

To initialize the robot prior to the trajectory starting, we prepend a short sequence that actuates the motors to the first trajectory position. To repeat multiple iterations of the trajectory, we append the motor positions multiple times. We consider one iteration of the trajectory to be once every foot has moved one step.

3.4.2 Trajectories

We generate trajectories for the Walk, Amble, S-Gait, and C-Gait sequences. For each of the trajectories, we attempt to move 1 cm, 1.5 cm, 2 cm, and 2.5 cm. Each trajectory generated has each foot move one step. This occurs over 40 knot points.

These trajectories use the foot order outlined in Section 3.2. However, as a part of the preliminary real-world testing, we experiment with different foot ordering for each trajectory. This is outlined in the Appendix (A).

Chapter 4

Simulation

We build a simulation of the robot in Pybullet [10]. It uses a simplified Unified Robotics Description Format (URDF) model of the robot and is set in the world frame. The objective of the simulation was to quickly verify that the generated trajectories are feasible. Ideally, it would also indicate that the movements are quasi-statically stable.

4.1 Setup

4.1.1 URDF Generation

We build simplified URDF models of the robot. To ensure that the simulation performs as expected, we first build a single actuator in the environment. Then, we build up to a single delta followed by multiple deltas. We build a model for the 12 actuator DeltaWalker (Figure 4.1), structured in floating and non floating environments. The floating environment is where the robot does not experience gravity. This is also used to confirm that the robot performs as expected. The non floating environment is where the robot does experience gravity, to mimic the real world. The position, size, and mass of all of the actuators, legs, feet, and shoes are as accurate to the real world as possible. The feet and shoes are combined into one element using an STL model of the two components combined. These models have friction to minimize slipping in the simulation environment. To simplify the

representation of the frames and PCB, they are combined into one single element, denoted as the base. This is centered at approximately the height of the center of mass of those components and the mass is accurate to the real world. Overall, this simplified model of the robot provides a sufficient estimation of what would occur in the real-world.

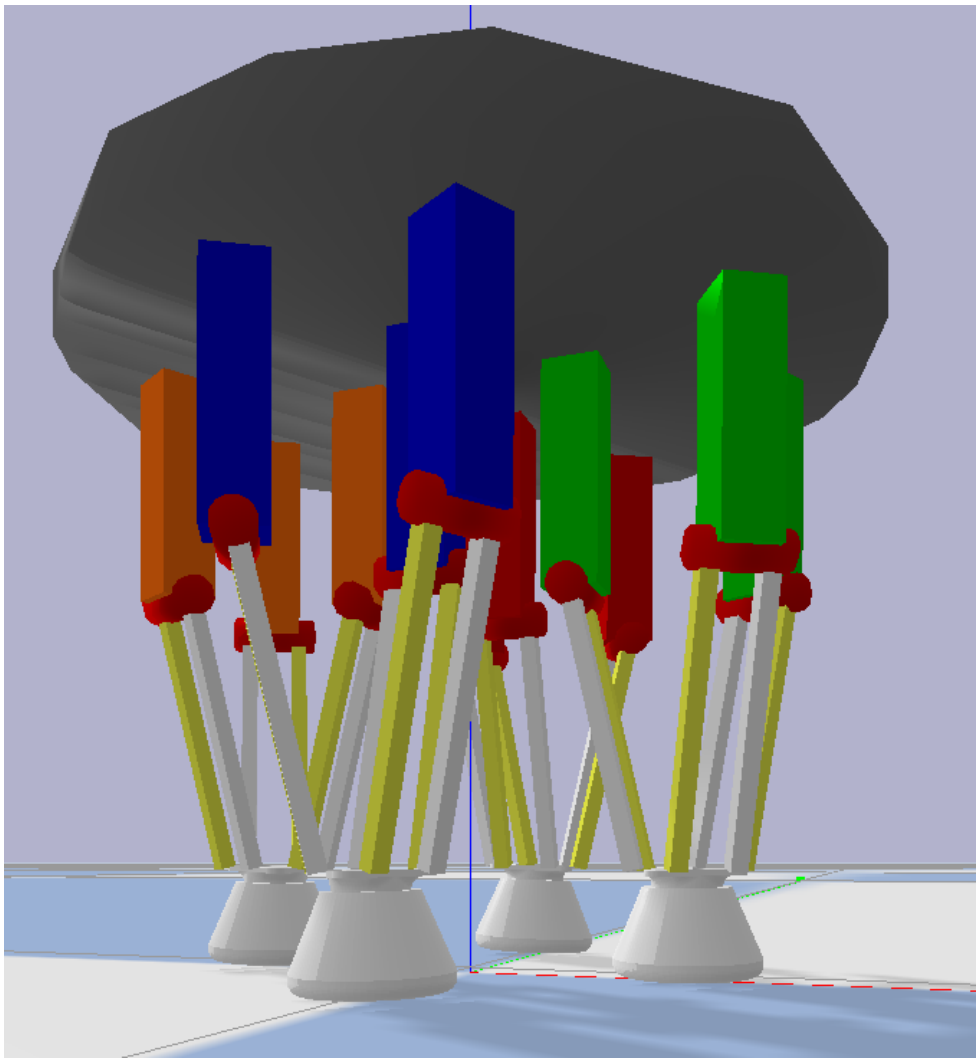

Figure 4.1: A URDF model of the DeltaWalker in the PyBullet simulation environment.

4.1.2 Simulation Environment

All simulations were initialized such that the robot was centered on the axes. Deltas 1 and 3 are aligned with the positive and negative x-axes respectively, and deltas 2 and 4 are aligned with the positive and negative y-axes respectively. The motors are also initialized to be fully retracted.

Using the generated trajectories from Sections 3.3 and 3.4 in conjunction with the concepts of inverse kinematics from Section 2.3.3, we are able to produce motor position commands at multiple timesteps. To simulate the motor position commands, we utilize Pybullet's built-in P controller. We input the desired motor positions and impose limits on the maximum force and velocity of the actuators to ensure realistic movement. Although the real robot employs a PID controller to command the motors, the built-in P controller sufficed for simulation purposes. Given that the trajectories should be quasi-statically stable and the robot moves slowly in the real world, we do not prioritize optimizing the simulated robot speed. We are able to track the world frame positions of the COM and the feet of the robot in simulation and compare them to the desired positions of the robot from the generated trajectory.

Each trajectory is run in one direction. The FB orientation trajectories are run with a positive displacement in the xy-direction. This meant that when looking at the robot from the top down view and aligned with the coordinate axes, the robot appeared to be moving diagonally along the line $y = x$ in quadrant 1. The FBS orientation trajectories are run with a positive displacement solely in the x-direction.

All of the trajectories are run for one iteration in simulation to qualitatively verify that the trajectory is feasible prior to testing it in the real world. We also calculate the mean square error between the desired and simulated positions to verify that the trajectory should be attainable.

4.2 Results

4.2.1 Manually Designed Gaits

The plots for all of the gaits and their versions are shown in Figures 4.2, 4.3, and 4.4. We find them to all be feasible. However, they may not necessarily be quasi-statically stable as they appear to tip. This is because these trajectories do not account for the position of the COM of the robot, so the COM may be outside of the support polygon. This would also explain why the mean square error values, shown in Table 4.1, are a bit high. The other cause for error could be simulation inaccuracies as the robot appears to slip, which should not happen in the real world. Nonetheless, these results indicate that the trajectories are worth testing in the real world.

Gait	Version	MSE (cm)
Walk	0	0.28
	1	0.33
	2	0.28
Amble	0	0.26
	1	0.23
	2	0.22
S-Gait	0	0.43
	1	0.26
	2	0.34

Table 4.1: Mean square error values for each of the simulated manually designed gaits with respect to the original trajectory of the gait.

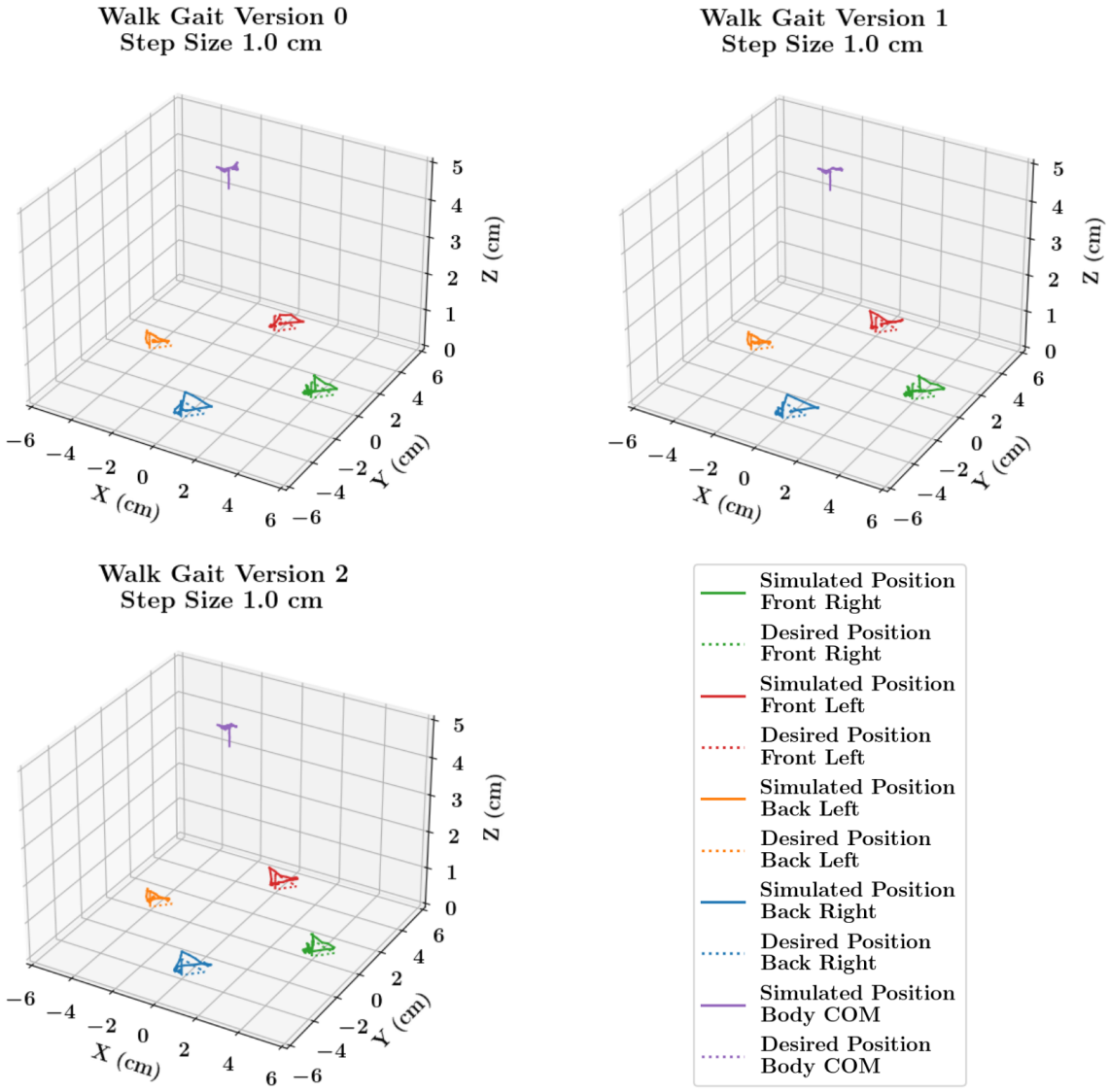

Figure 4.2: Simulation results for all versions of the manually designed Walk gait.

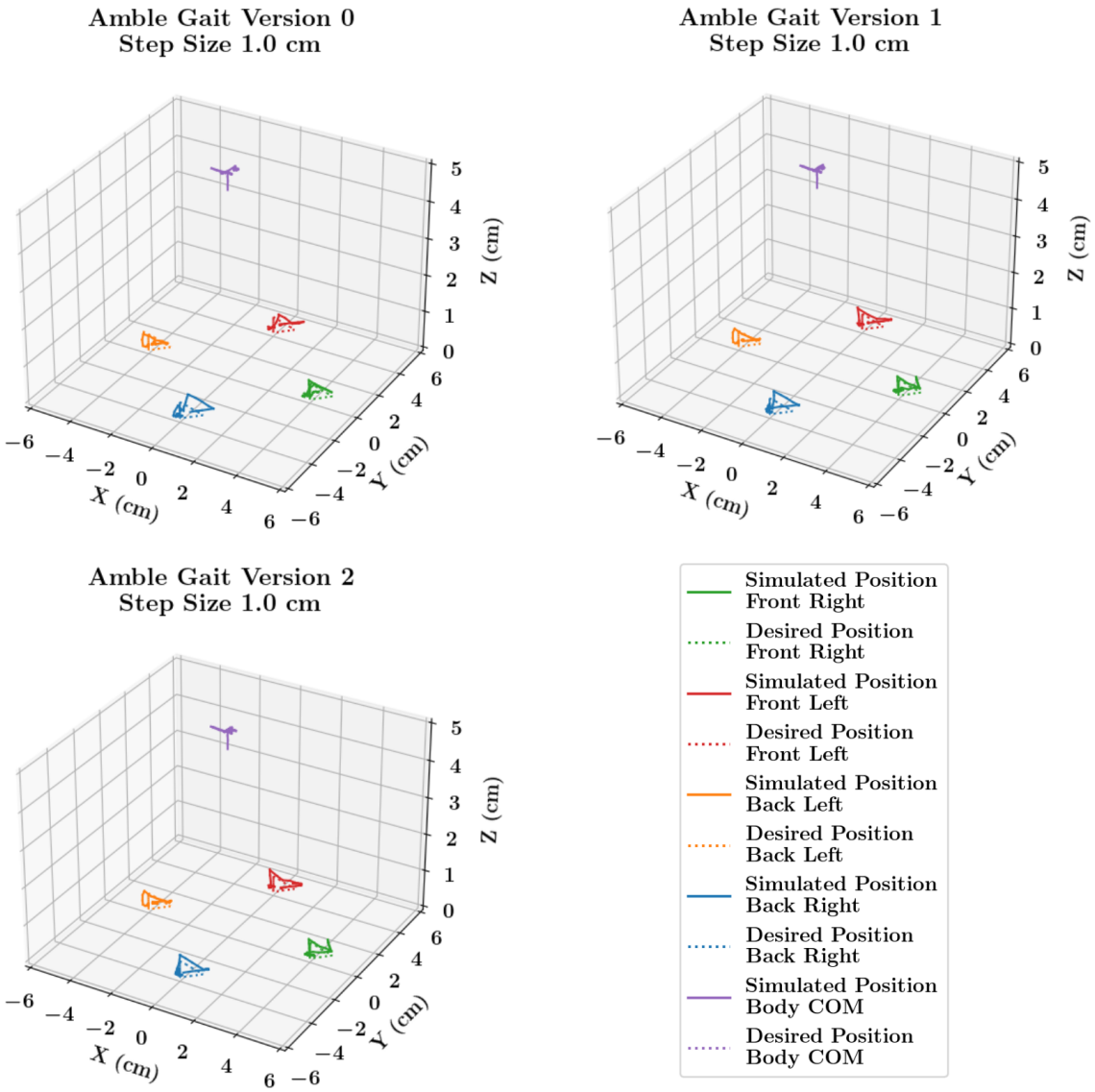

Figure 4.3: Simulation results for all versions of the manually designed Amble gait.

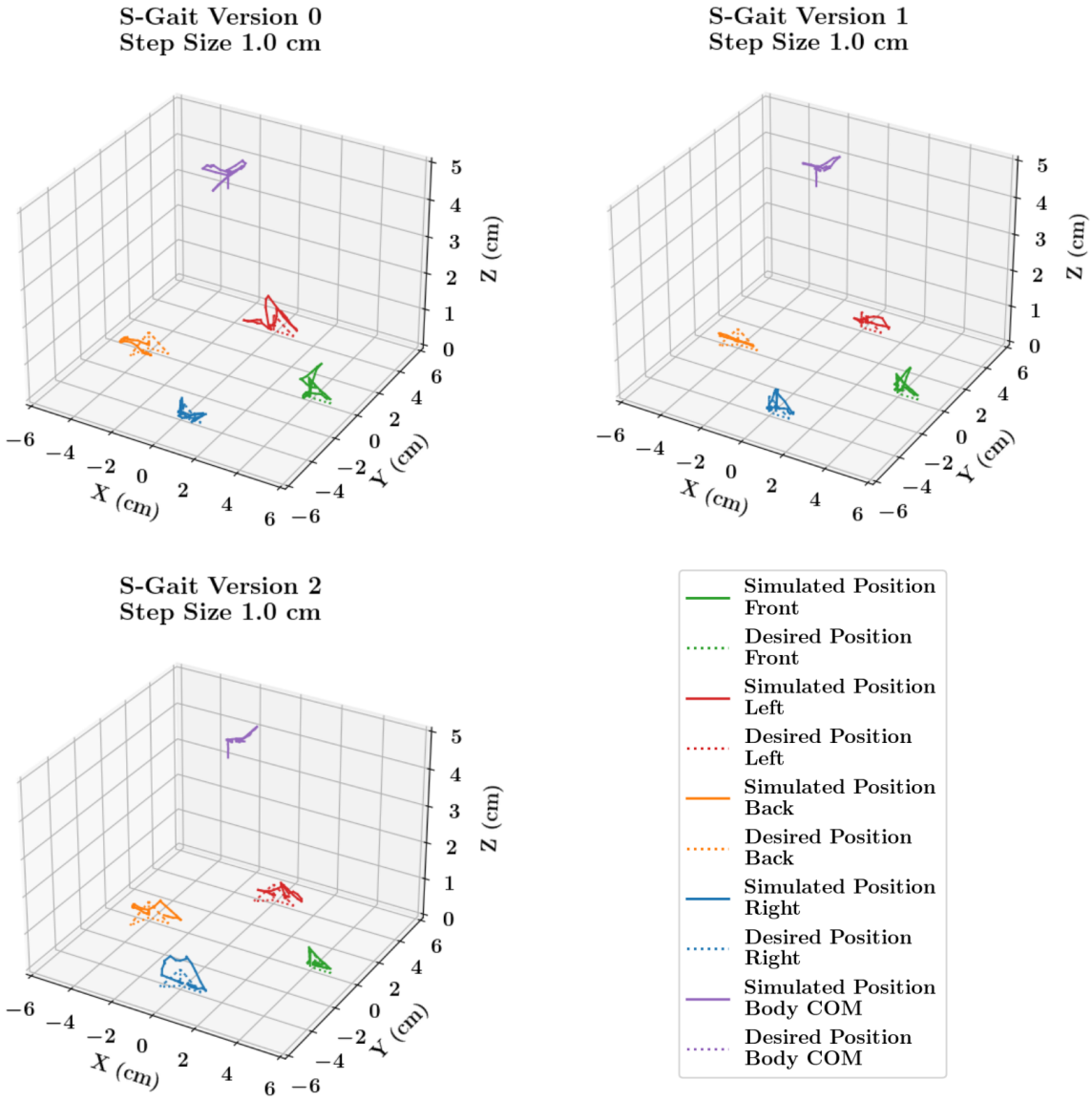

Figure 4.4: Simulation results for all versions of the manually designed S-Gait.

4.2.2 Trajectory Optimized Gaits

The plots for all of the gaits and their versions are shown in Figures 4.5, 4.6, 4.7, and 4.8. We find them to all be feasible. We also find that by accounting for the COM of the robot, the robot does not tip as much in simulation. We do notice at larger step sizes, especially around 2.5 cm, the robot tends to to be less accurate with more tipping and slipping. This is reflected in the mean square error values, shown in Table 4.2. However, these results overall indicate that the trajectories are worth testing in the real world.

Gait	Step Size (cm)	MSE (cm)
Walk	1.0	0.17
	1.5	0.16
	2.0	0.20
	2.5	0.64
Amble	1.0	0.21
	1.5	0.38
	2.0	0.59
	2.5	0.5
S-Gait	1.0	0.17
	1.5	0.34
	2.0	0.28
	2.5	0.6
C-Gait	1.0	0.18
	1.5	0.28
	2.0	0.46
	2.5	0.79

Table 4.2: Mean square error values for each of the simulated trajectory optimized gaits with respect to the original generated trajectory.

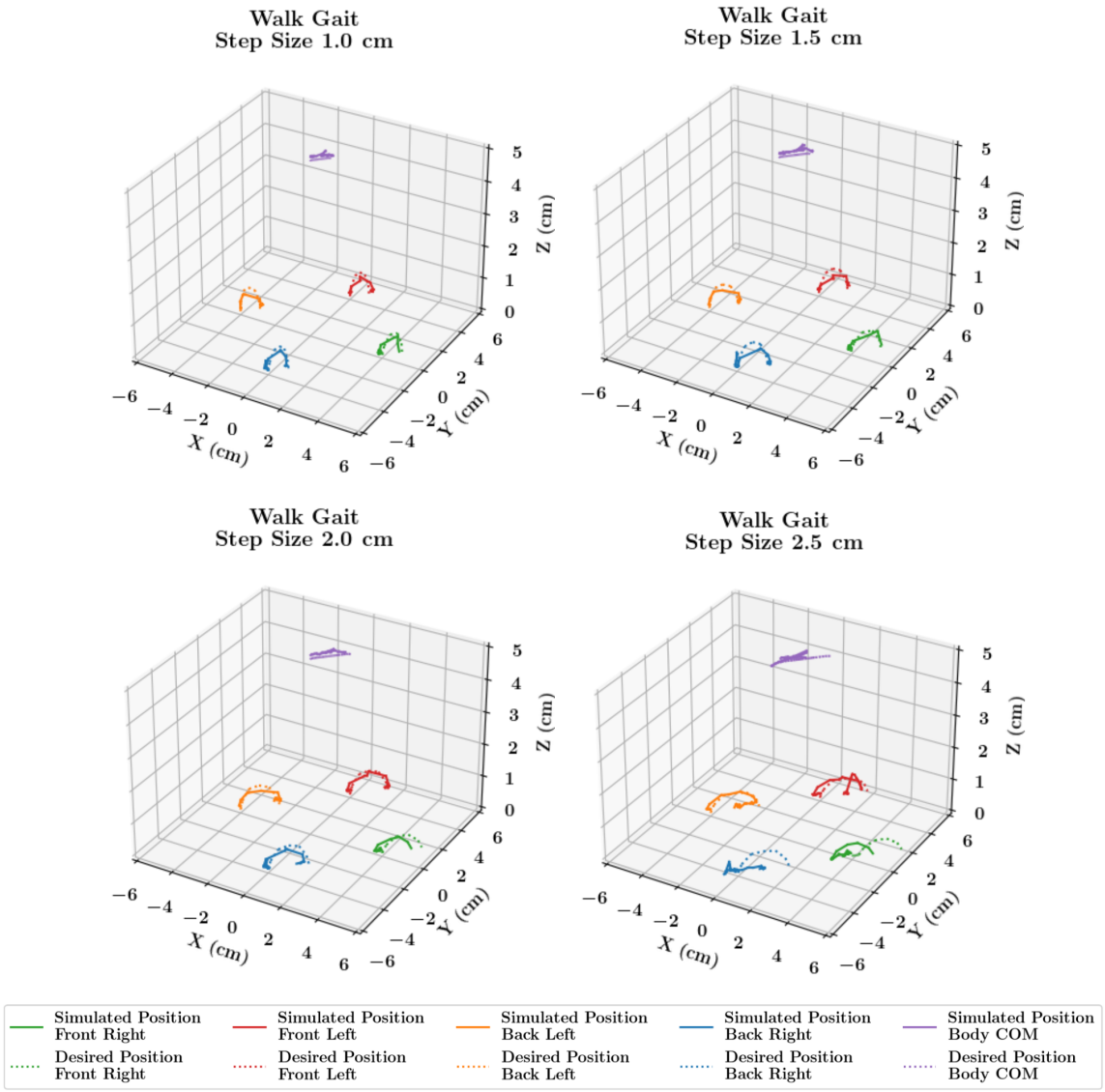

Figure 4.5: Simulation results for all versions of the trajectory optimized Walk gait.

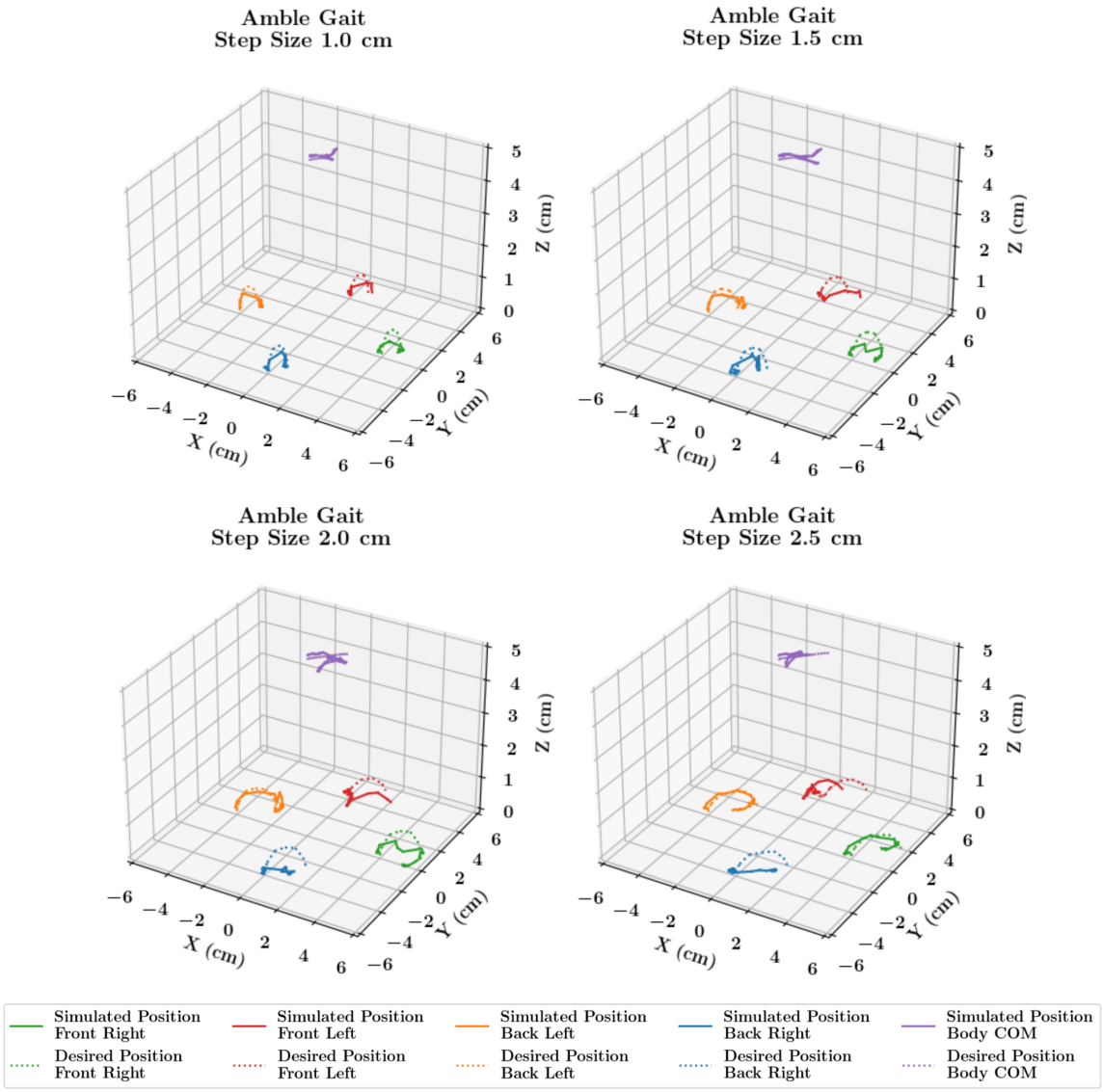

Figure 4.6: Simulation results for all versions of the trajectory optimized Amble gait.

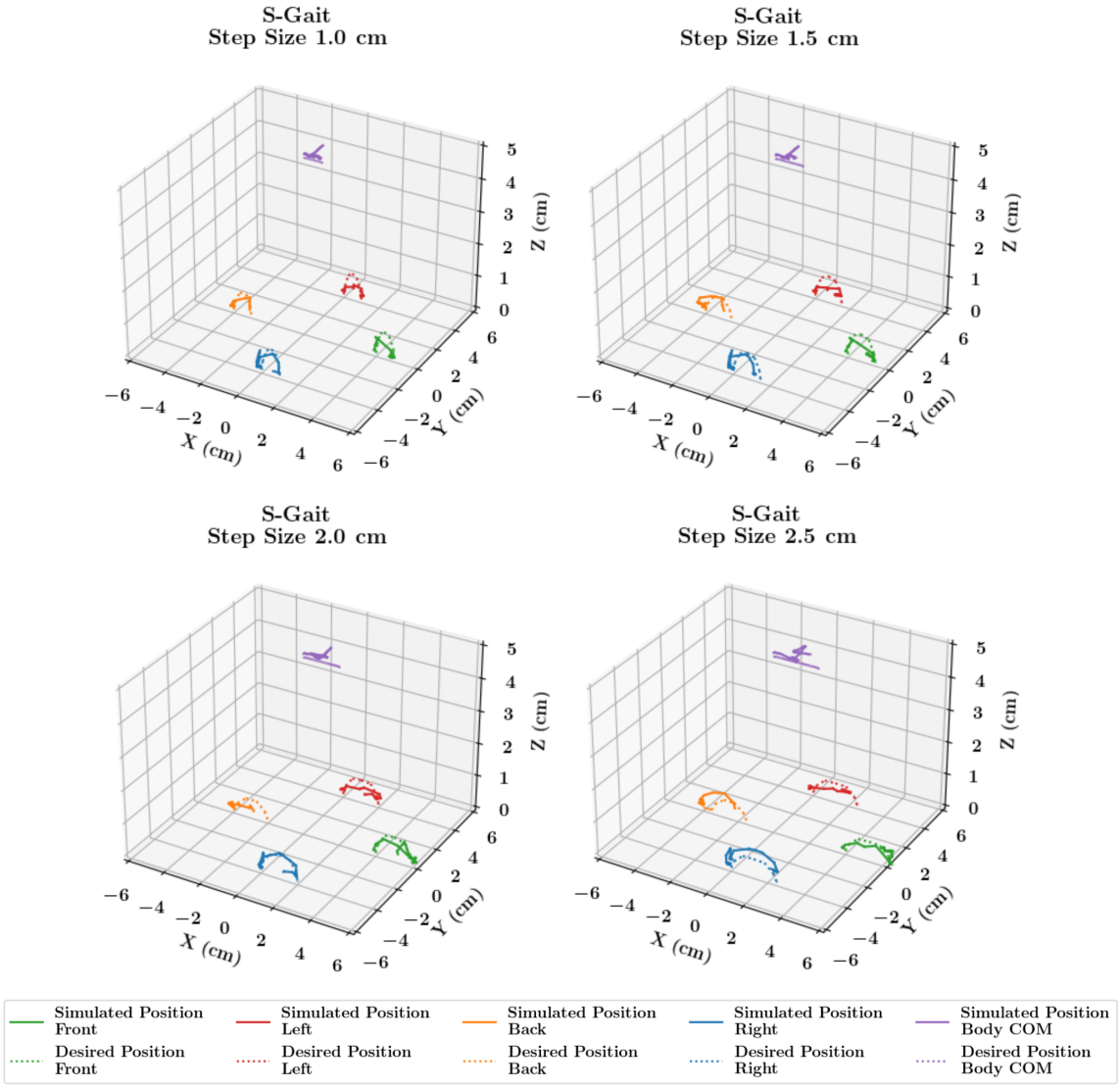

Figure 4.7: Simulation results for all versions of the trajectory optimized S-Gait.

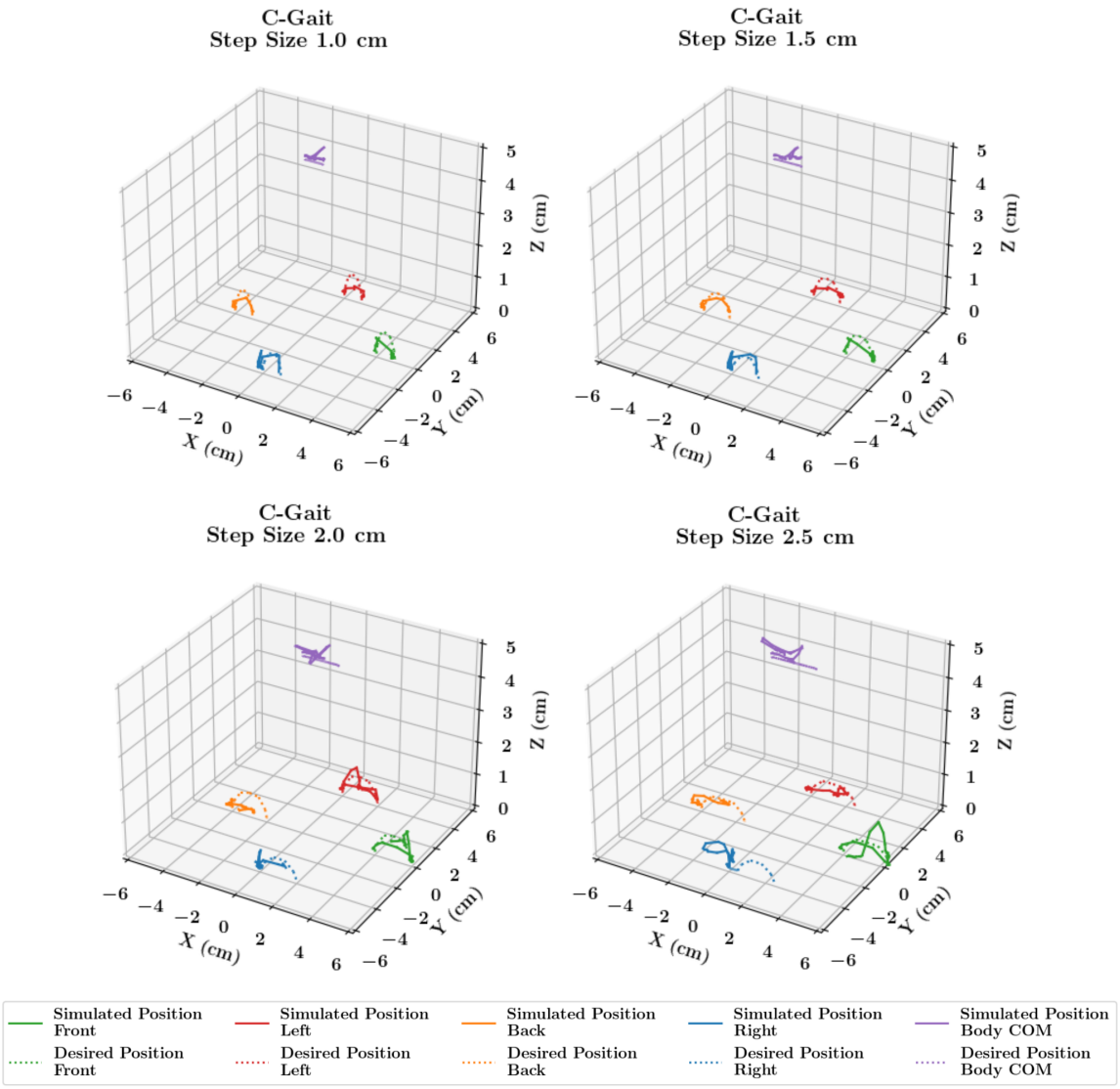

Figure 4.8: Simulation results for all versions of the trajectory optimized C-Gait.

Chapter 5

Experiments

After verifying that a trajectory is feasible in simulation, we move to testing the trajectory in the real world. The objective is to verify the trajectories are feasible in the real world as well.

5.1 Experimental Setup

To set up the experiments in the real-world, we build a rig out of 8020, laser cut acrylic boards, and 3D printed components, as seen in Figure 5.1. The bottom level of the rig houses a camera to record the bottom of the robot, which is then used for data analysis. The platform that the robot walks on has a 1 cm x 1 cm grid etched on it to make it easier to quantify how far the robot has moved and start the robot in a consistent position. The platform has two crossbars that are used as standoffs. The lower one— the wire support standoff— is used to rest the power and signal cable of the robot so that it minimizes how much the cable drags. The upper one— the cloth cover standoff— is used to hold up a black cloth that covers the whole setup. There are two light bars along the side of the rig a bit below the platform to illuminate the underside of the robot. An additional camera (not pictured) is set up on a tripod on one side of the robot to allow for a second view of the robot from the side. This camera is not used for data analysis.

The robot is set up such that the robot center is above the center of the platform. In both orientations, the robot is set up such that the trajectory is intended to be

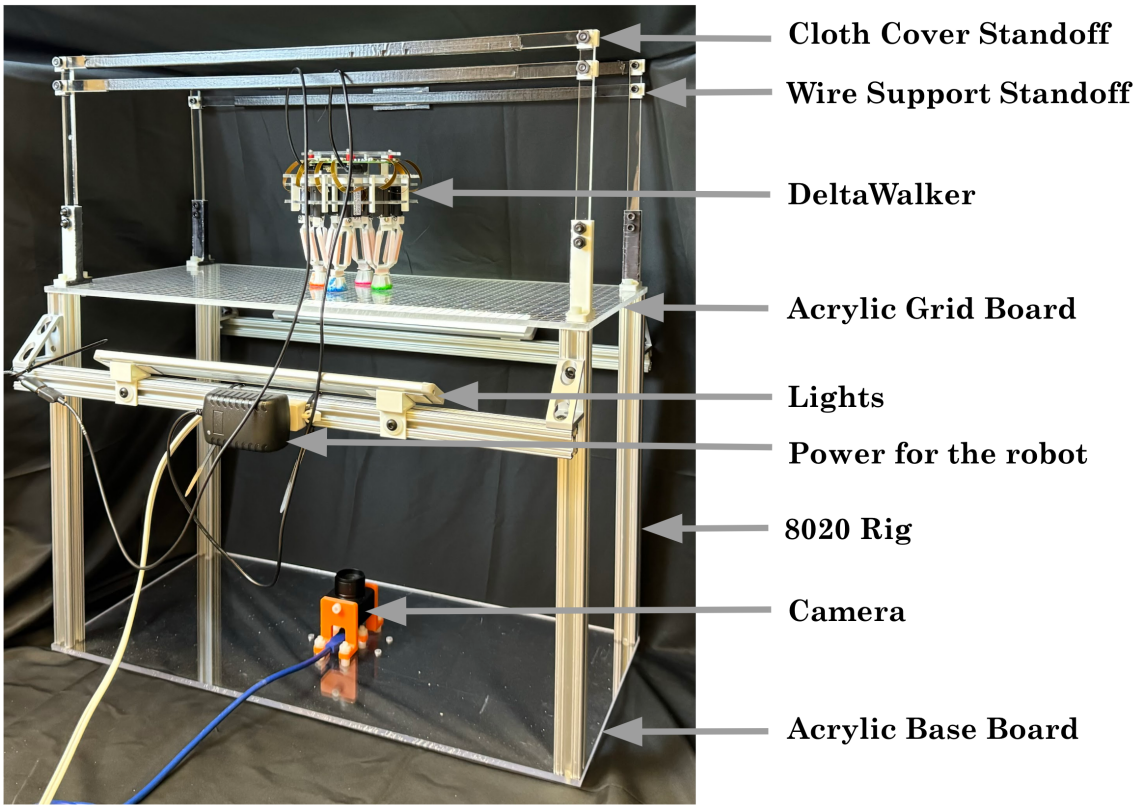

Figure 5.1: The test rig for running experiments on the robot. It is comprised of 8020 rails, laser cut acrylic boards, 3D printed mounting components, a camera, and lighting to illuminate the underside of the robot for data collection.

along the x- or y-axes, rather than at a diagonal to the axes. This is for easier analysis and data collection. This is to make data collection and analysis easier. For the FB orientation, this means that the robot is rotated 45^{o} clockwise about the center along the z-axis. For the FBS orientation, the robot feet are aligned with the axes. These are shown in Figure 5.2. The robot shoes are painted with different colors underneath the grip tape. The color designations for each foot are shown in Table 5.1. This is used to track the position of each foot in the bottom view recordings.

Foot	1	2	3	4
Color	Green	Red	Orange	Blue

Table 5.1: The feet are labelled and colored to allow for consistent color tracking of the robot.

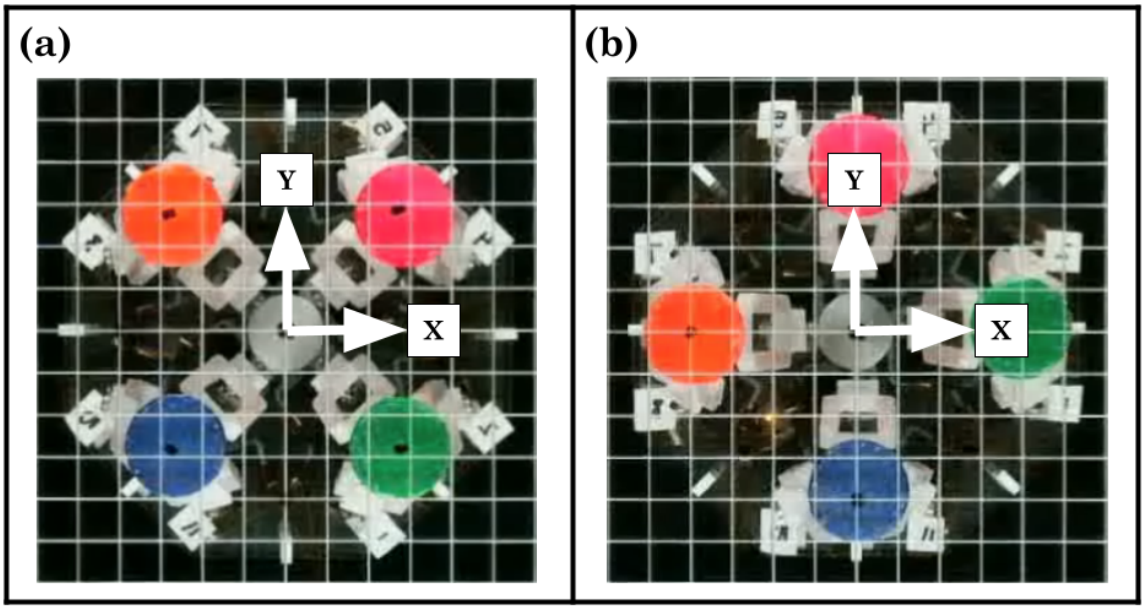

Figure 5.2: The two starting orientations of the robot for the testing: (a) FB Orientation, (b) FBS Orientation. Robot walks on an etched platform with a 1 cm x 1 cm grid.

Each trial of a trajectory is run for 3 iterations in the real world. This is to allow us to collect more step data and see how the robot would behave over time as it moves further. The specifics of each set of trials are detailed further in Sections 5.3.1 and 5.3.2.

5.2 Experimental Analysis

Each recording contains one trial of the trajectory being tested. The raw recording is saved at the correct real-time frame rate by calculating the frame rate at the end of each trial. These recordings are then modified to remove camera distortion, crop out extraneous parts of the image for quicker analysis, and remove any extra frames at the start or end of the recording. This includes trimming away the startup time of the robot as it moves to the initial position.

Next, the modified video is processed using OpenCV's [5] contour detection and color masking to track each of the foot positions. To minimize the number of false foot detections, there are bounds on the minimum and maximum contour area detected

by the algorithm. Once the contours are filtered by the contour area requirement, the area of the minimum enclosing circle of the contour is also restricted. That in combination with the color masks for each color ensures that the foot detection is as accurate as possible. The x- and y-position of the feet in pixels are recorded at each frame. At the end of the video processing, the data is saved into a comma-separated value (CSV) file. The original and processed video frames can be found in Figure 5.3.

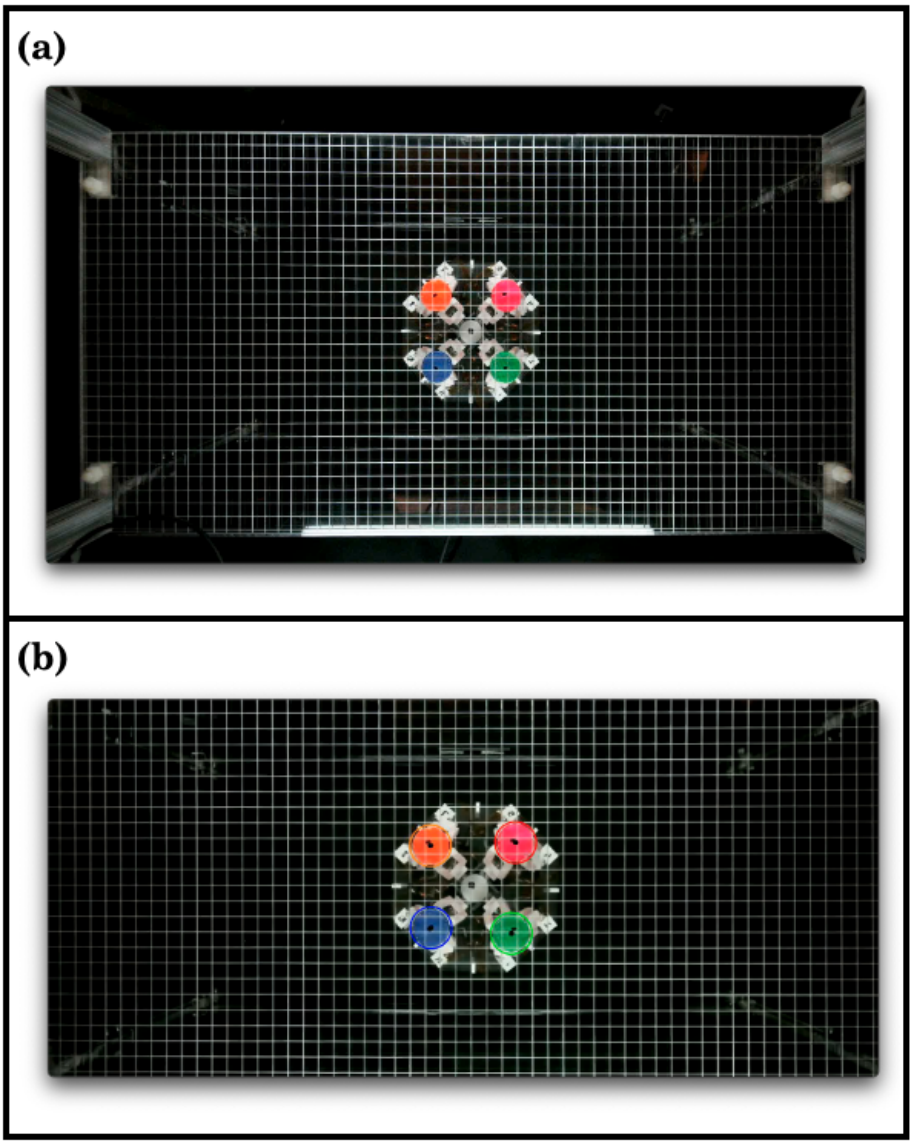

Figure 5.3: The (a) Raw frames versus the (b) Post-Processed Frames used for data collection.

There are occasionally frames where a foot is not detected, in which case the position of that foot is denoted as zero. There are also occasionally false positives. In those cases, we manually go through the areas where we see the false positives or negatives and we interpolate the nearby values to determine what that value should be. Because the robot is slow and the recordings are being made at approximately 13 frames per second, using interpolation for these values is fairly accurate. After the data is processed in pixels, the data is converted to centimeters and centered. The grids create an approximate conversion for the pixel to cm ratio and the location of the center of the grid can be determined in pixels. Once the data is converted into centimeter scale, it can be analyzed for how far the robot moves in total or per step.

For analyzing how far the robot moves in total, we first calculate the distance each foot moved and the angle that the foot moved with respect to the axis it moved along. We also find the percent error of the movement for that foot with respect to how far it was intended to move. We use the first and last recorded values of the robot position for this. Then we average the values across the four feet to get the movement of the robot in total for that trial. These values for the feet and the robot as a whole are then further averaged across all 3 trials of a trajectory to get an estimate for how far the robot can move in 3 iterations.

For analyzing how far the robot moves per step, the process is very similar. First we calculate the distance, angle, error of the foot movement for that step. For that foot, we average the step data. Then we average foot step data to get the average movement of the robot. Finally, we average those results across all 3 trials of a trajectory to get an estimate for how far the robot can move in 1 iteration. To make this easier, we create a user input-based interface. For each trial of each trajectory, the user selects the approximate point where the robot has completed the first and second iterations of the movement. These values and the first and last values of the data are used to calculate the movement for each iteration. We can also use these values and the duration it takes to move to calculate the approximate velocity of the robot. We only do step-wise analysis for the trajectory optimized gaits.

5.3 Results

5.3.1 Manually Designed Gaits

We initially run these experiments only in the x-direction. We run this qualitatively a few times and we run one trial of it while collecting data. We notice that almost of the trajectories move much further than anticipated, resulting in large errors. The only exception is version 2 of the Walk gait, which undershoots the desired point. We also notice that robot tends to move at an angle. These results are summarized in Table 5.2 and can be visualized in Figures 5.5, 5.7, and 5.9.

Gait	Version	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
Walk	0	3.0	4.0	2.6	32.8
	1	3.0	3.4	6.4	13.4
	2	3.0	2.7	19.7	-8.9
Amble	0	3.0	4.0	2.7	31.9
	1	3.0	3.6	5.7	19.3
	2	3.0	4.8	7.7	60.1
S-Gait	0	3.0	5.0	5.7	65.1
	1	3.0	5.4	11.9	80.1
	2	3.0	5.4	11.3	80.4

Table 5.2: Experimental results for all of the manually designed gaits where the goal was to move 3 cm total in the +x direction.

We further notice that the feet that should be stationary are slipping. As the robot attempts to lift one foot, it tends to fall on that foot instead (Figure 5.4). This causes the robot to be pushed by the foot that is supposed to be taking a step. Although the grip tape is intended to reduce slipping, the force of the robot being pushed overcame the friction force provided by the grip tape. We believe that the tipping is either a result of poor trajectory design or deformations in the 3D printed

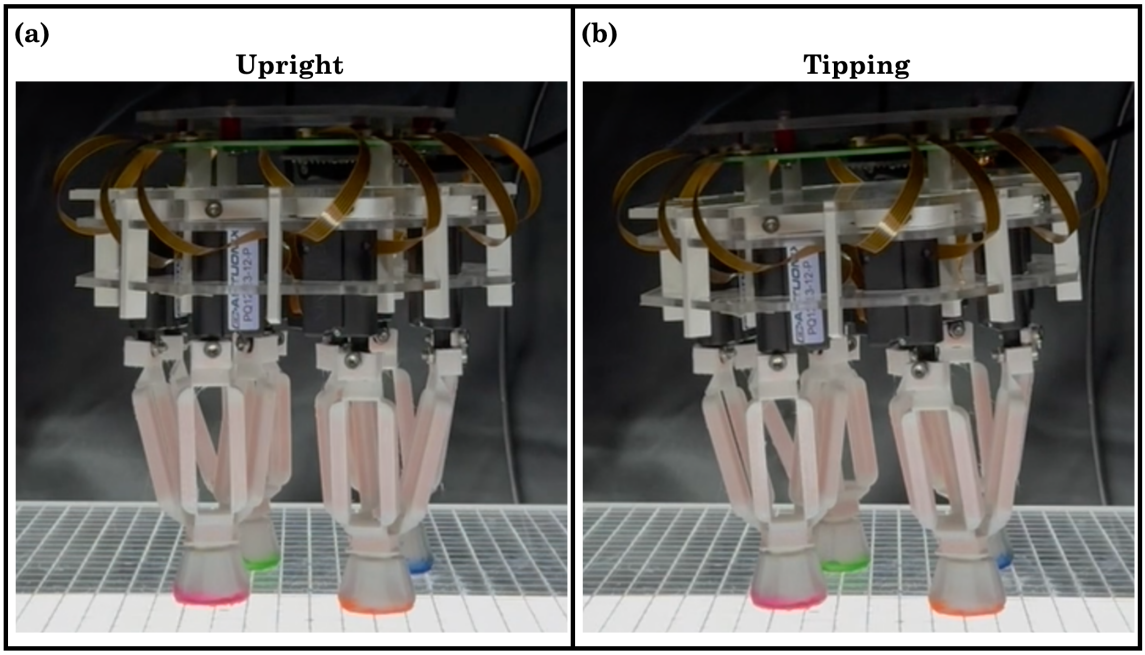

Figure 5.4: A comparison of how the robot should be (a) upright when taking a step versus the actual movement of the robot when taking a step: (b) the robot tilts.

delta links causing the robot to put uneven force on each foot even when the robot should theoretically be stable with the COM directly over the center of the robot.

We run the trajectories again in the y-direction only. These results are summarized in Table 5.3 and can be visualized in Figures 5.6, 5.8, and 5.10. The angles listed are relative to the x-axis. We find that the Walk trajectory undershoots slightly but moves diagonally rather than in the positive y-direction. For the Amble trajectory, the gait has a severe tendency to overshoot for versions 0 and 1 and overshoots slightly for version 2, but all gaits still move at an angle. Finally, for S-Gait the trajectories are the most straight but they also overshoot severely. We believe that these issues we are seeing can be partially attributed to the deformations in the deltas. However, we mainly believe it could be due to power and signal cable dragging the robot back a little, causing some of the gaits to undershoot. Finally, we still notice the feet are slipping.

These trajectories, although quasi-statically stable in simulation, are not quasi-statically stable in real life. Although this issue could partially be attributed to deformations in the delta links, as perhaps the foot opposite the one being lifted

Gait	Version	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
Walk	0	3.0	3.0	66.5	-1.1
	1	3.0	2.9	66.0	-5.0
	2	3.0	2.8	56.3	-5.7
Amble	0	3.0	1.9	76.4	-35.6
	1	3.0	2.5	68.5	-17.8
	2	3.0	3.2	65.7	5.3
S-Gait	0	3.0	4.6	85.8	52.3
	1	3.0	5.1	87.4	69.6
	2	3.0	4.6	87.4	54.5

Table 5.3: Experimental results for all of the manually designed gaits where the goal was to move 3 cm total in the +y direction.

is not quite in contact with the ground, this would still be an issue that could be resolved with a sufficient weight shift. The robot COM is likely at the edge of the support polygon and a slight shift would allow it to be stably within the support polygon. However, it is challenging to program these weight shifts into the trajectory as it takes a lot of iteration and constant testing.

Because we find that these trajectories move much further than anticipated, we do not plan to pursue running more trials or evaluating the sizes of each step. Furthermore, we do not plan to manually design or run versions of the adjacent sequential trajectory as we will likely see large errors in that too. Instead, we opt to switch to using trajectory optimization to account for the robot COM position so that the trajectories generated with these gaits would no longer tip.

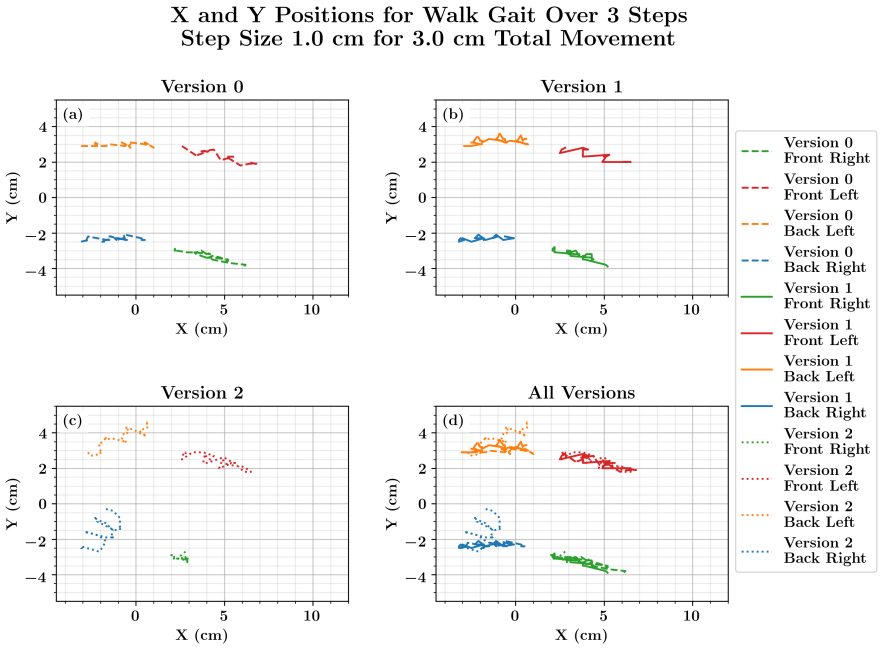

Figure 5.5: Experimental results for all versions of the Walk trajectory moving in the +x direction.

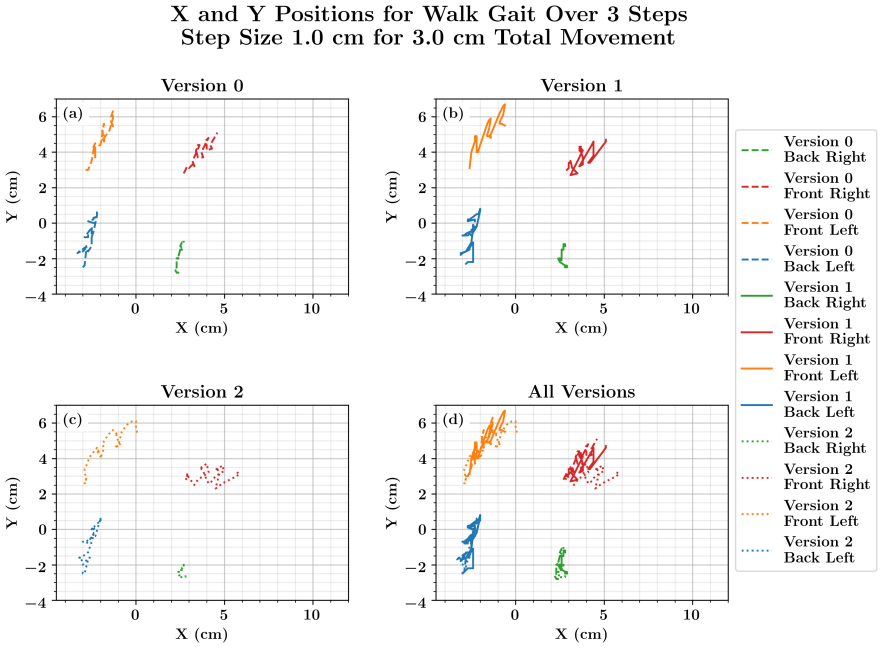

Figure 5.6: Experimental results for all versions of the Walk trajectory moving in the +y direction.

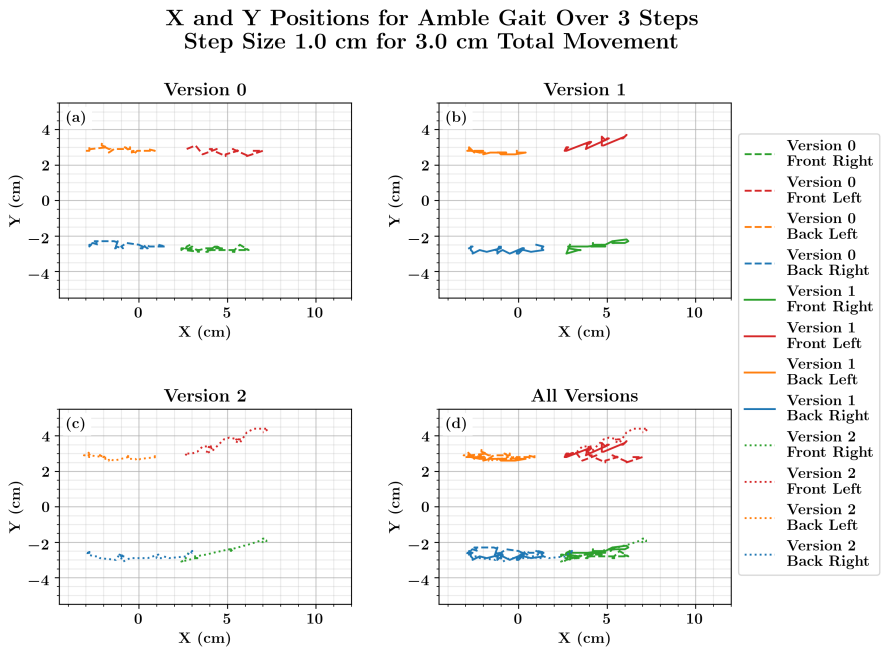

Figure 5.7: Experimental results for all versions of the Amble trajectory moving in the +x direction.

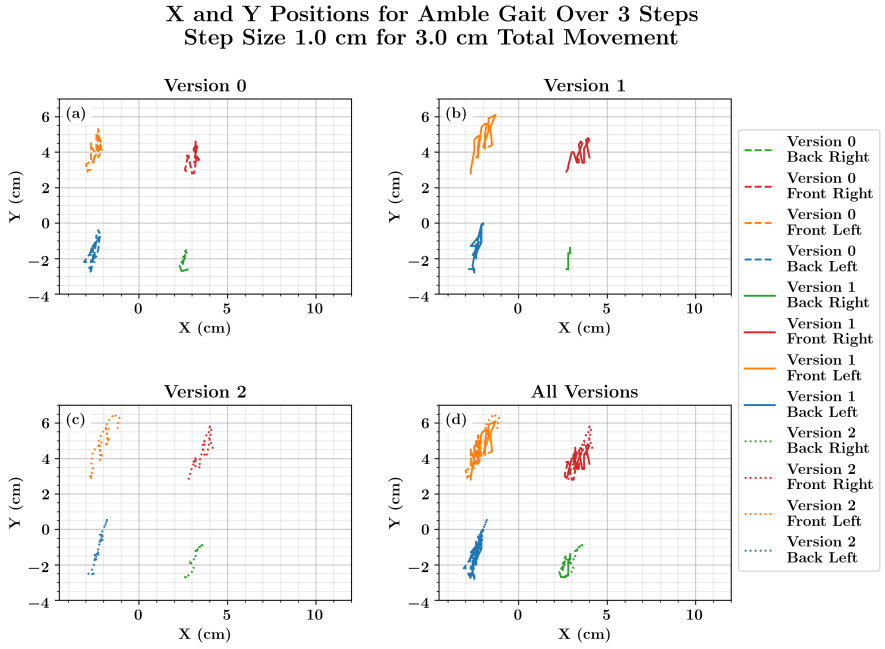

Figure 5.8: Experimental results for all versions of the Amble trajectory moving in the +y direction.

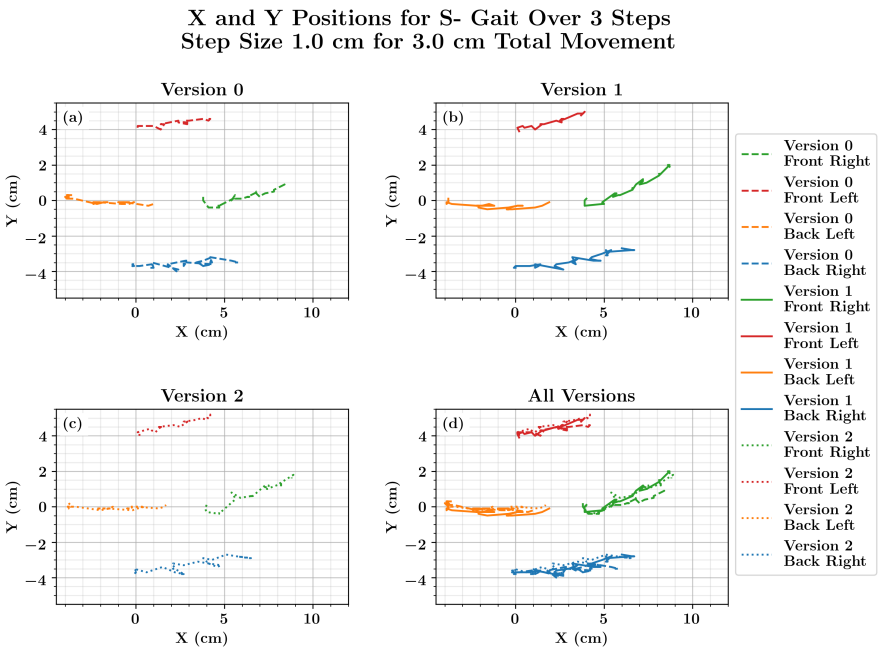

Figure 5.9: Experimental results for all versions of the S-Gait trajectory moving in the +x direction.

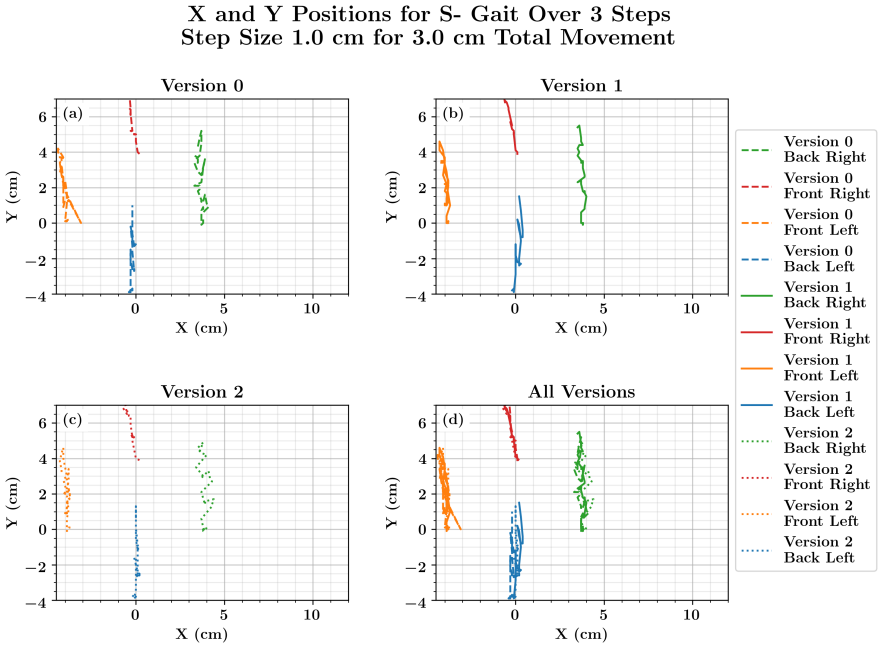

Figure 5.10: Experimental results for all versions of the S-Gait trajectory moving in the +y direction.

5.3.2 Trajectory Optimized Gaits

We run each trajectory for 3 iterations and 3 trials. These trials are only done in the x direction.

The Walk trajectory results can be found in Table 5.5 and Figure 5.11. We find that taking a step size of 1 cm and 1.5 cm the robot tends to overshoot the goal whereas at step size of 2 cm and 2.5 cm the robot tends to undershoot the goal. The ideal step size likely occurs between 1.5 cm and 2 cm. We also find that the robot tends to tilt at a slight angle. A step size greater than 2 cm is not quite feasible.

The Amble trajectory results can be found in Table 5.6 and Figure 5.12. We find that this trajectory undershoots at all of the step sizes and results in the robot at an angle. The 1.5 cm step size performs slightly better than the 1 cm step size but the trajectory performance worsens drastically at 2 cm and beyond.

The S-Gait trajectory results can be found in Table 5.7 and Figure 5.13. We find that it overshoots with a step size of 1 cm but overshoots with other step sizes. The error is minimal at a step size of 2 cm, indicating that it is likely the ideal step size. The robot also tends to tilt at a slight angle.

The C-Gait trajectory results can be found in Table 5.8 and Figure 5.14. The gait also tends to overshoot the goal point slightly at 1 cm step size but undershoots more at a larger step size. This indicates that the ideal step size is between 1 cm and 1.5 cm, however this trajectory performs worse than the C-Gait overall. The robot also has a tendency to tilt at an angle.

Overall, we also find that the robot moves rather slowly. The velocities based on the duration it takes for the robot to complete one iteration for each trajectory type are shown in Table 5.4. Of these trajectories, Walk and S-Gait are once again the best. The Amble and C-Gait move slower.

We find that the Walk trajectory is ideal for the FB orientation and the S-Gait trajectory is ideal for the FBS orientation, as they are capable of taking larger step sizes. The Amble trajectory does not work well for this robot, as it tends to undershoot and cause the robot to walk at an angle rather than in the intended direction. The C-Gait trajectory is decent however not as good as the S-Gait. It also has a tendency to undershoot. Both the Amble and C-Gait trajectories work better with smaller step sizes.

Gait	**Step Size**	**Time Per Step (s)**	**Velocity (mm/s)**	**Average Velocity (mm/s)**
Walk	1.0	30.2	0.4	0.475
	1.5	32.0	0.5	
	2.0	34.0	0.5	
	2.5	32.5	0.5	
Amble	1.0	32.3	0.3	0.375
	1.5	31.8	0.4	
	2.0	34.2	0.4	
	2.5	33.9	0.4	
S-Gait	1.0	30.2	0.4	0.525
	1.5	30.8	0.5	
	2.0	33.9	0.6	
	2.5	34.1	0.6	
C-Gait	1.0	29.8	0.3	0.4
	1.5	30.8	0.4	
	2.0	33.3	0.4	
	2.5	34.7	0.5	

Table 5.4: The average velocities of the robot, calculated based on the time needed to take one step with each foot of the robot.

Movement Type	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
	1.0	1.2	7.2	18.9
Individual	1.5	1.7	6.5	14.8
Step	2.0	1.8	4.1	-8.9
	2.5	1.7	3.3	-32.3
	3.0	3.7	8.5	22.0
Total	4.5	5.3	7.6	17.1
Movement	6.0	5.6	6.0	-6.2
	7.5	5.4	7.0	-28.2

Table 5.5: Experimental results for the Walk trajectory.

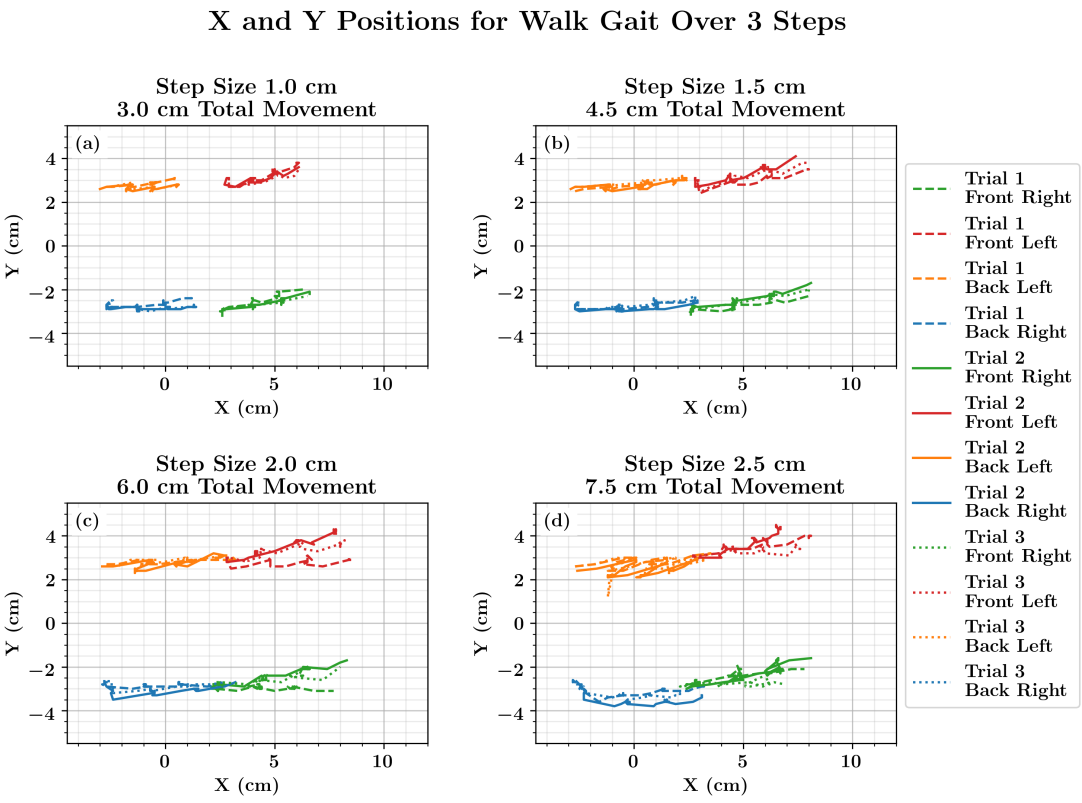

Figure 5.11: Experimental results for the Walk trajectory.

Movement Type	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
	1.0	0.9	-7.4	-12.4
Individual	1.5	1.3	-3.6	-11.4
Step	2.0	1.4	-10.7	-29.6
	2.5	1.5	-15.0	-40.9
	3.0	2.6	3.4	-12.5
Total	4.5	4.0	4.7	-10.8
Movement	6.0	4.2	6.7	-30.2
	7.5	4.2	13.6	-43.5

Table 5.6: Experimental results for the Amble trajectory.

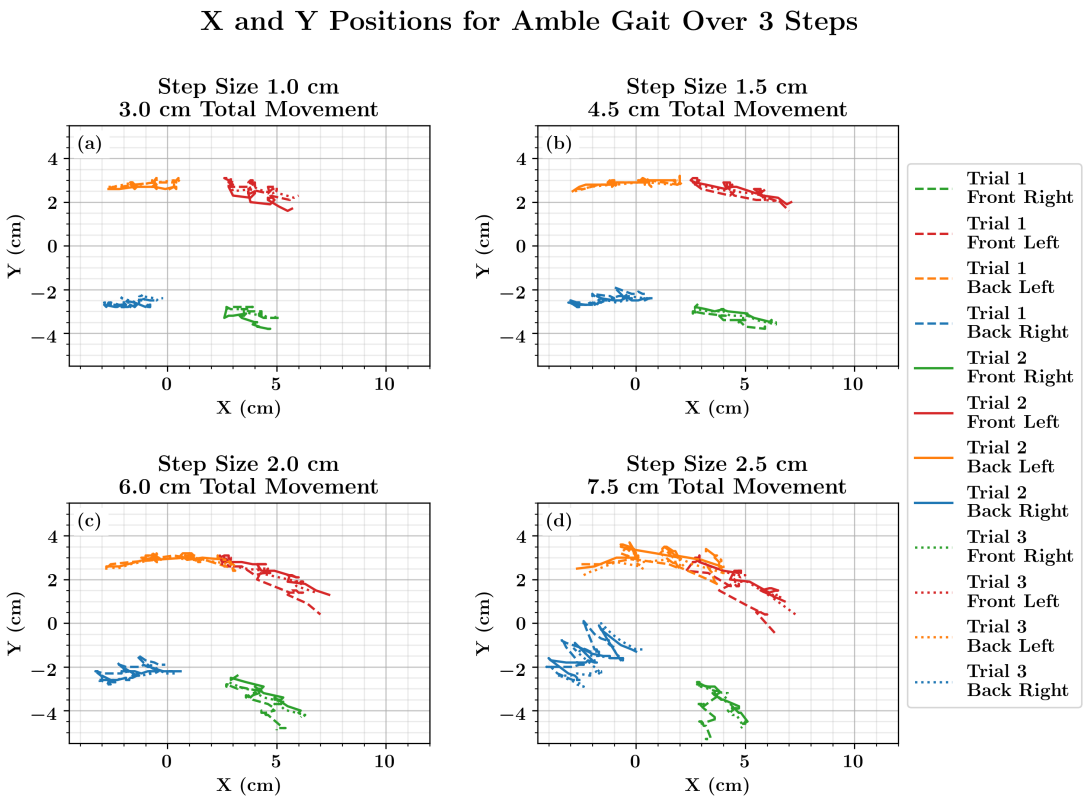

Figure 5.12: Experimental results for the Amble trajectory.

Movement Type	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
	1.0	1.1	6.9	9.5
Individual	1.5	1.4	4.4	-4.6
Step	2.0	1.9	1.8	-2.7
	2.5	2.2	-1.3	-12.5
	3.0	3.4	8.0	13.0
Total	4.5	4.4	5.8	-3.2
Movement	6.0	6.0	4.6	-0.3
	7.5	6.8	2.4	-9.1

Table 5.7: Experimental results for the S-Gait trajectory.

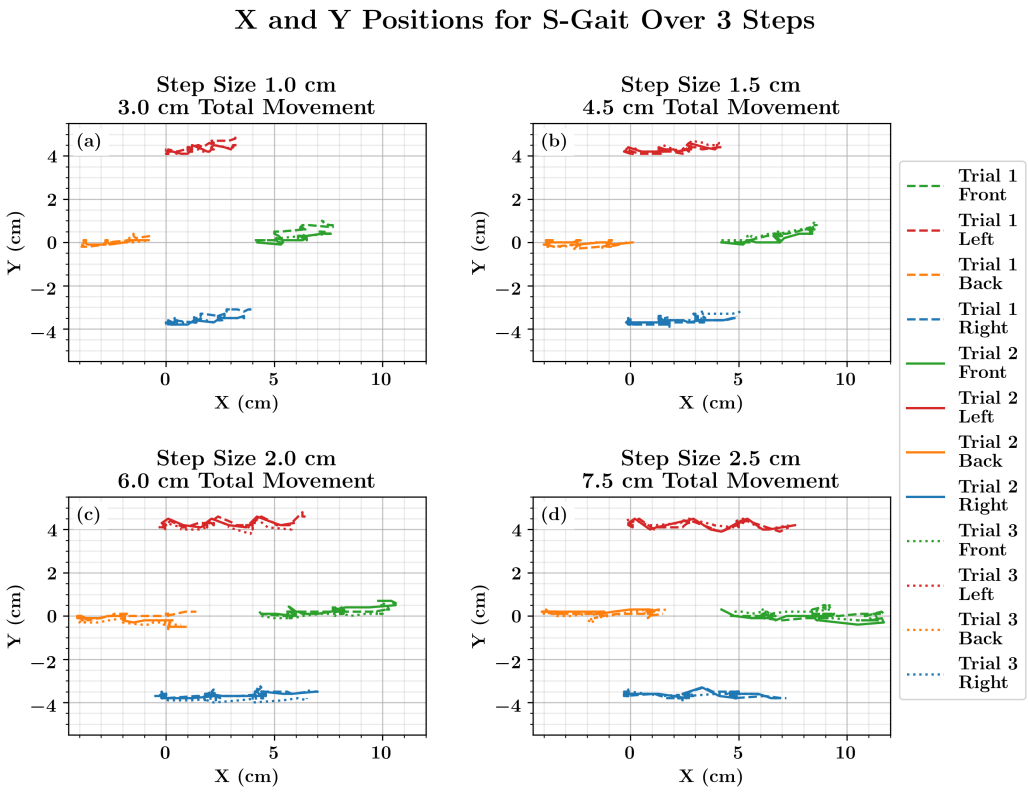

Figure 5.13: Experimental results for the S-Gait trajectory.

Movement Type	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
	1.0	1.0	4.2	2.6
Individual	1.5	1.3	5.7	-12.8
Step	2.0	1.4	6.5	-26.7
	2.5	1.7	-1.3	-32.7
	3.0	3.2	6.2	6.2
Total	4.5	4.0	8.6	-10.6
Movement	6.0	4.7	8.3	-22.3
	7.5	5.4	5.0	-27.8

Table 5.8: Experimental results for the C-Gait trajectory.

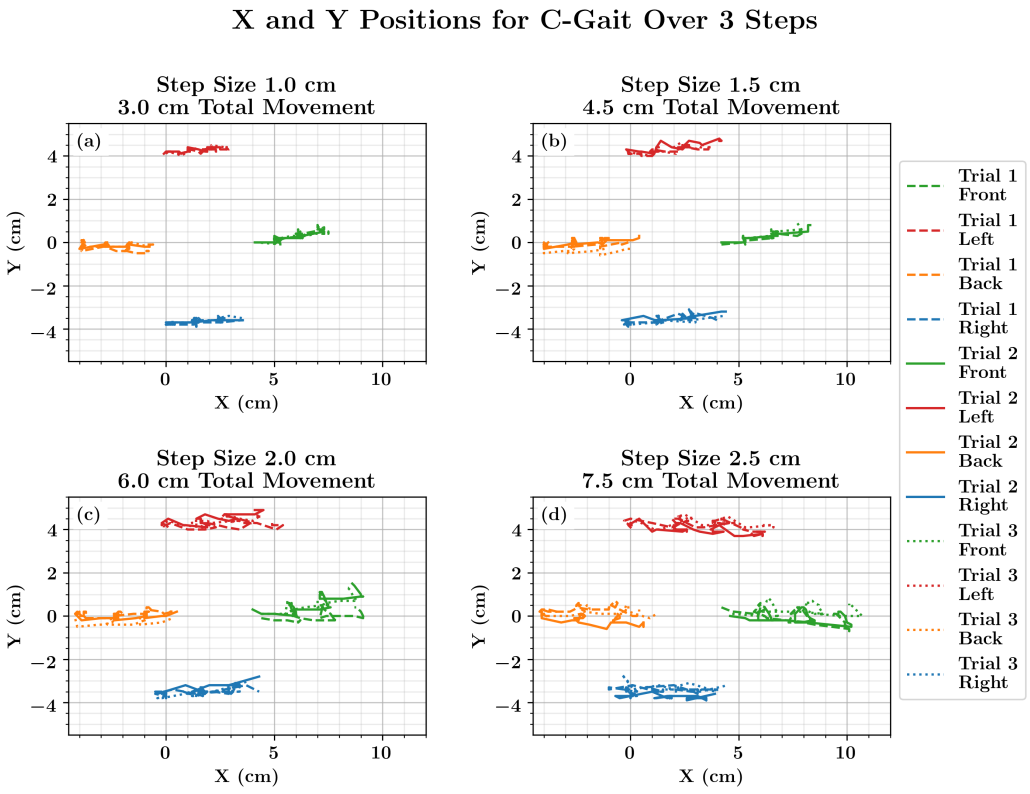

Figure 5.14: Experimental results for the C-Gait trajectory.

Chapter 6

Other Studies

6.1 Trajectory Optimization For 3 Steps

Since the Walk and S-Gait trajectories yielded the best results, we want to know how a trajectory that moved the robot 3 steps for a total of 6 cm would perform. We run three trials of each trajectory in the x-direction in the real world. These results can be found in Figures 6.1 and 6.2 and Table 6.1.

Gait	**Desired Distance (cm)**	**Actual Distance (cm)**	**Angle (deg)**	**MPE (%)**
Walk	6.0	6.3	7.2	4.3
S-Gait	6.0	5.7	7.9	-4.7

Table 6.1: Experimental results for using trajectory optimization to generate 3 steps of the Walk and S-Gaits.

We find that both gaits perform close to the desired values with small error. Walk tends to overshoot and S-Gait tends to undershoot. Both gaits still result in movement at a slight angle. In comparison to the results from Section 5.3.2, we find that the trajectories generated for 3 steps has a smaller absolute error but larger angle compared to using 3 iterations of a single step. Overall, this indicates that we can use either method for generating multiple steps for traversing a larger distance.

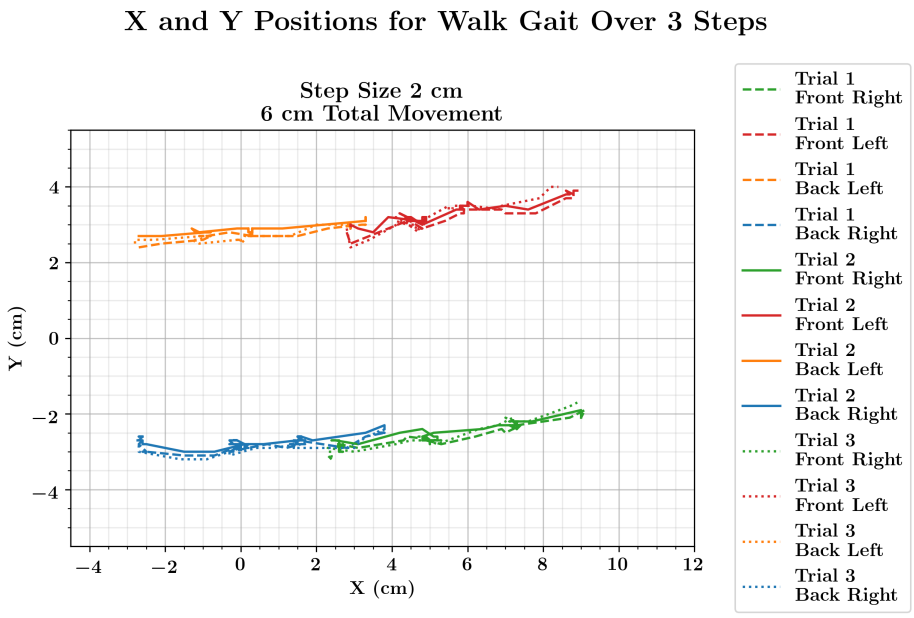

Figure 6.1: Experimental results for using trajectory optimization to generate 3 steps of the Walk Gait.

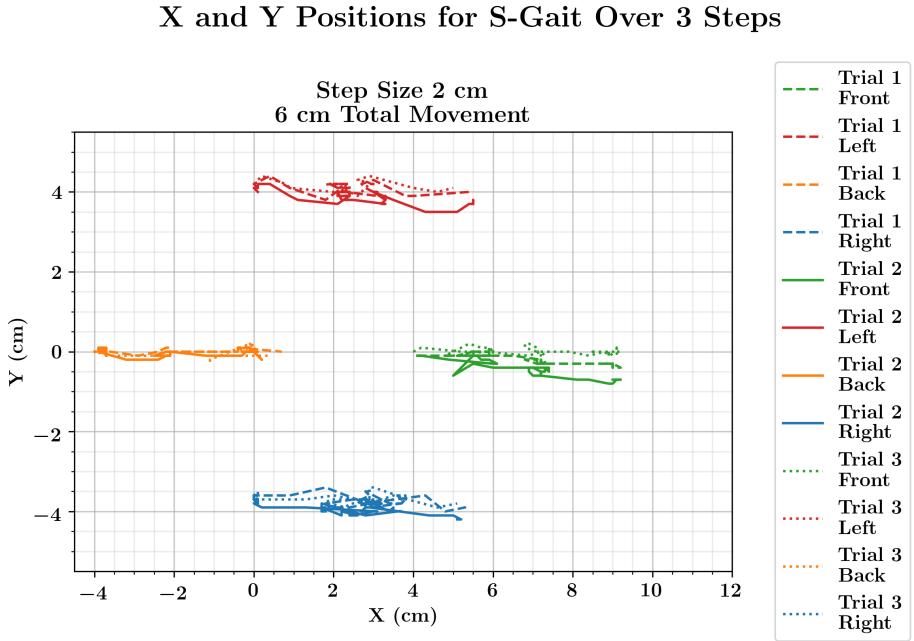

Figure 6.2: Experimental results for using trajectory optimization to generate 3 steps of the S-Gait.

6.2 Trajectory Optimization for Rotating in Place

We want to explore if the robot could feasibly rotate in place so that the robot could switch between FB and FBS orientations. To do this, we run the trajectory optimization algorithm for a rotation of 15^{o} about the z-axis. This was run on the robot for 3 iterations with the goal of turning approximately 45^{o}.

We can show qualitatively that it is feasible. The robot is able to move to approximately 45^{o}. The starting and ending points of the robot are shown in Figure 6.3.

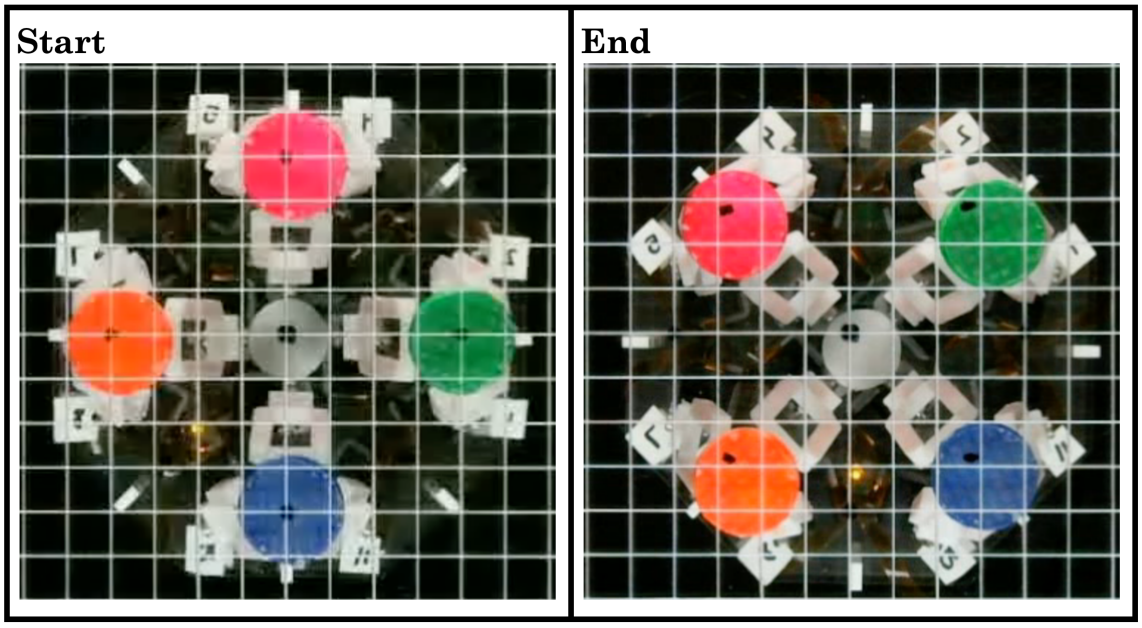

Figure 6.3: Start and end positions of performing a 45^{o} rotation by repeating a 15^{o} rotation 3 times.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Overall, we have introduced a novel quadruped robot that is capable of walking. We have created a simulation model for it as well as the physical robot and tested its capabilities. The kinematics of the robot, although complex, are known and can be utilized to implement trajectories generated through trajectory optimization onto the robot. The trajectory optimization framework can be used to create multiple different trajectories with different foot orders. It is faster than manual trajectory design and accounts for shifts in weight so that the robot can properly move.

We find that the robot is capable of walking in multiple orientations and can even turn in place. We find that the Walk gait is ideal in the FB orientation, with the ideal step size being between 1.5 cm and 2 cm. The S-Gait is ideal in the FBS orientation, with the ideal step size being approximately 2 cm. These Walk and S-Gait trajectory steps can be used as building blocks for the robot to follow larger paths. The Amble and C-Gaits are less ideal but can be utilized for taking smaller steps.

Code for the project can be found in the repository DeltaWalker under the ZoomLab GitHub (https://github.com/ZoomLabCMU/DeltaWalker).

7.2 Future Work

The robot still can be improved greatly with further exploration. We have outlined multiple avenues that we believe are worth pursuing.

7.2.1 Simulation

The simulation could be improved to be more accurate to the real world. The simulation as it stands attempts to mimic the real world as closely as possible. However, the base is simplified and it is difficult to represent the linear actuators as having a stationary base with only the shaft moving. So there are certain inaccuracies in the URDF model geometry. Furthermore, the simulation model friction and inertia could be modeled more accurately.

7.2.2 Robot Control

To improve the current configuration of the robot, we would like to improve the controller so the robot moves faster, smoother, and does not pause between each waypoint. We would like to implement closed loop control and an IMU to allow the robot to correct for any drifting that occurs.

7.2.3 Robot Design

We would also like to explore different design configurations. One change could be making the deltas more spread out or arranging them in alternative shapes, such as a rectangle rather than a square. We would also like to explore using different sizes of deltas to allow the robot to step further. Furthermore, switching to other actuators that allow for finer control and faster speeds would be worthwhile. Reducing the height of the center of the mass of the robot through changing the arrangement of the motors or the type of motors would likely improve the capabilities of the robot as well. It would be interesting to test different foot sizes to see if any changes would improve the stepping capabilities. Testing different grip tapes to find what helps minimize the slipping the most would be worthwhile as well. We would also like to

switch to being able to send trajectories to robot wirelessly and implement onboard power so that the robot can operate untethered.

We would also like to return to the 9 actuator version of the robot and explore the capabilities of that as well.

7.2.4 Gaits

We would like to improve the current gaits. We know that with the current configuration of the robot, the ideal step size is likely between 1.5 cm and 2 cm for a walk gait. However, we would like to run more experiments to confirm this theory. Furthermore, we are interested in seeing if the foot order matters. For example, the walk and amble gaits start with stepping a back foot first but we would like to explore in further depth what would happen if the front foot stepped first. And for the modified tripedal and adjacent sequence gaits, it would be interesting to see what would happen if the rear feet moved first. We would also like to explore new gaits. If we implement a better controller and faster motors, we could potentially create more dynamic gaits such as trotting or running. We could iterate upon the trajectory optimization setup through exploring how to improve the constraints.

7.2.5 Capabilities

We would like to explore other capabilities of the robot. We know that it can turn 15^{o} but we would like to see if it can turn more and how accurately it can turn these amounts. We would also like to see the robot operate on different terrain, such as sand, rock, or concrete. We are also curious about if the robot can climb a step and if so, how high of a step can it climb. We would also like to know the load capacity of the robot.

Appendix A

Appendix

Prior to running multiple trials and variations of each trajectory produced through trajectory optimization, we are curious if the foot order for a gait matters. For example, does it matter if we start with the front or the back feet? What about right versus left feet? To explore this, we use trajectory optimization to explore the different foot orders. We run one trial of each foot order where trajectory runs for 3 iterations, intending to step 1 cm each time. This means that the robot is supposed to move 3 cm in total. We run each of these trajectories in the positive x-direction.

For Walk, we find that the results are similar, as shown in Figure A.1 and Table A.1. We do notice that moving with a right foot prior to a left tends to cause the robot to move further. Furthermore, starting with the front feet appeared to move the robot more accurately than starting with the rear. However, these differences did not not seem large enough to be worth switching away from the classic back-leg first gait that many quadrupeds pursue. We decide to start with the back-left leg first.

The results for Amble can be found in Figure A.2 and Table A.2. This time there does not appear to be much of a difference between moving left and right foot first. Moving the back feet first also appears to work better. To be more consistent with the classic diagonal amble and the selected walk order, we elect to once again start with the back left leg.

For S-Gait, we find moving the right foot first results in overshooting whereas the left foot first results in undershooting. This is shown in Figure A.3 and Table A.3. Moving the front versus the back feet first does not appear to make a difference as

the left versus right patterns overshadow that. We choose to stay with the original modified tripedal foot order in that case. We opt to start with the front leg before moving to the left leg.

C-Gait originally started out as an alternative order for S-Gait. We were unsure if doing two side legs in a row would cause the robot to become unbalanced. However, we opt to make it a separate trajectory because it ultimately does follow a front/side/back/side pattern as opposed to a front/side/side/pattern. The results are shown in Figure A.4 and Table A.4. Moving with the left foot first results in undershooting and moving with the right foot first varies between undershooting and overshooting. The absolute errors are larger for starting with the front foot compared to starting with the back foot. However, to stay consistent with the S-Gait, we choose to start with the front leg and then move to the left leg.

Overall, we did not find any reason to believe that that front versus back and left versus right ordering mattered. However, we acknowledge that this requires more extensive testing with different step sizes and more trials to more thoroughly decide if this is true. Since this is just intended to be a preliminary test, it was more of a qualitative look with some quantitative data than a detailed series of tests. The objective was to see if there are any striking differences that would warrant further testing or specifically avoiding a certain order.

Version	Step Order	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
1	Back Left Front Left Back Right Front Right	3.0	3.3	14.4	9.7
2	Back Right Front Right Back Left Front Left	3.0	3.7	8.0	21.9
3	Front Left Back Left Front Right Back Right	3.0	2.9	6.1	-1.9
4	Front Right Back Right Front Left Back Left	3.0	3.4	3.9	14.7

Table A.1: Experimental results for the different versions and food orders of the Walk sequence.

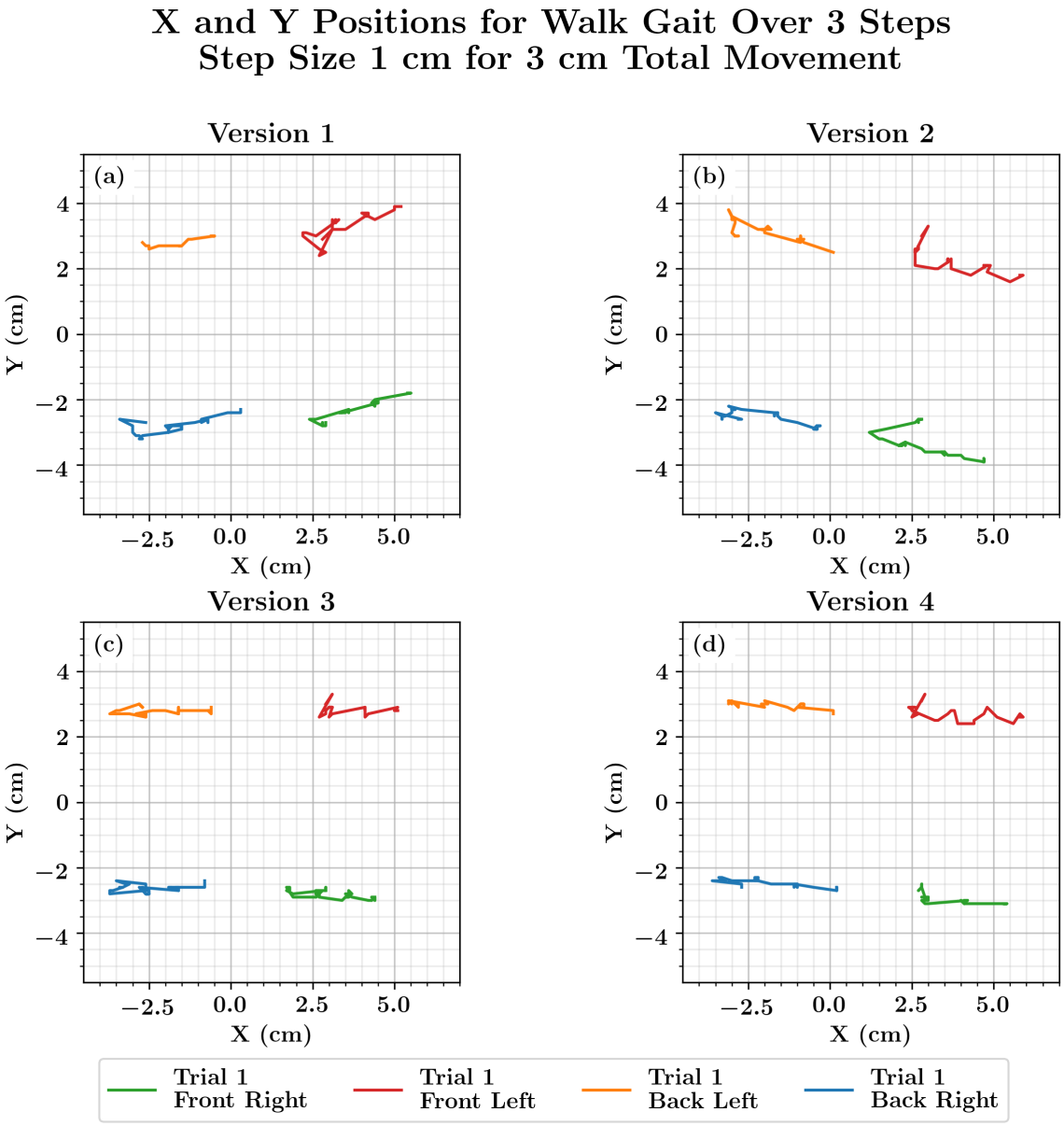

Figure A.1: Experimental results for the different versions and food orders of the Walk sequence.

Version	Step Order	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
1	Back Left Front Right Back Right Front Left	3.0	3.1	5.9	2.2
2	Back Right Front Left Back Left Front Right	3.0	3.2	5.5	7.7
3	Front Left Back Right Front Right Back Left	3.0	3.4	3.3	11.9
4	Front Right Back Left Front Left Back Right	3.0	3.3	6.9	9.6

Table A.2: Experimental results for the different versions and food orders of the Amble sequence.

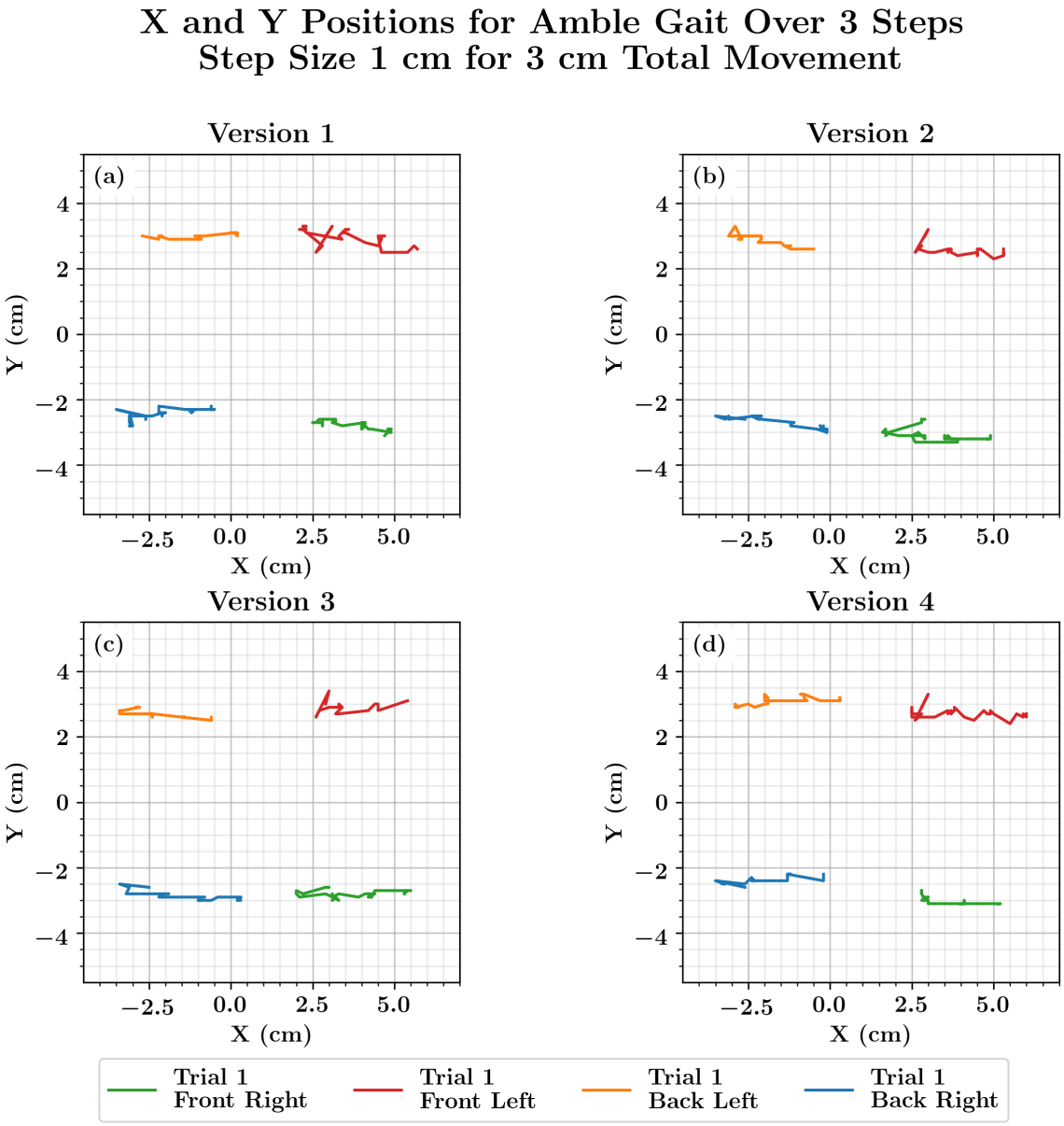

Figure A.2: Experimental results for the different versions and food orders of the Amble sequence.

Version	Step Order	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
1	Front Left Right Back	3.0	3.0	13.8	-1.1
2	Front Right Left Back	3.0	3.6	8.5	19.0
3	Back Left Right Front	3.0	2.8	10.0	-6.2
4	Back Right Left Front	3.0	3.2	11.8	5.0

Table A.3: Experimental results for the different versions and food orders of the S-Gait sequence.

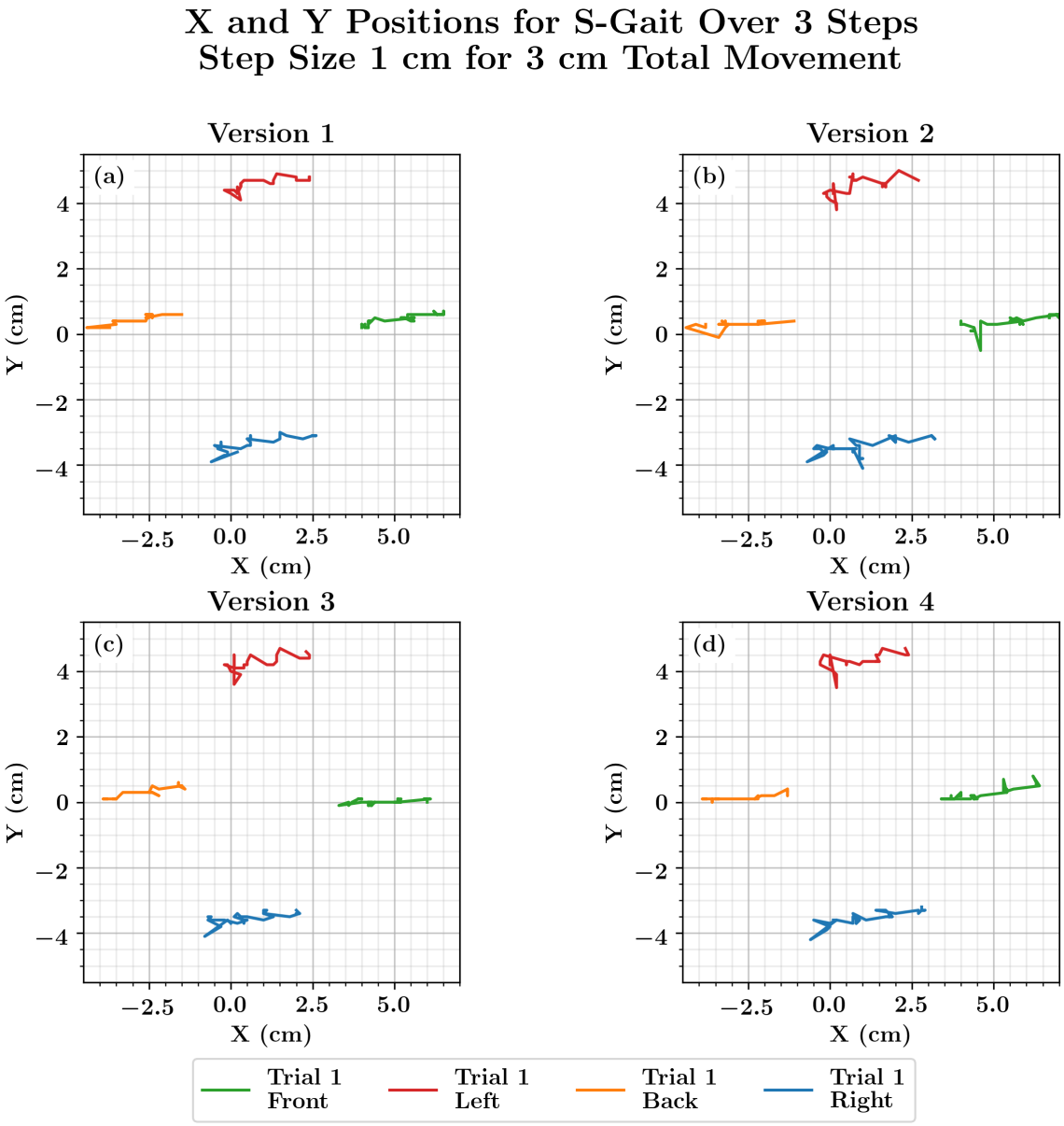

Figure A.3: Experimental results for the different versions and food orders of the S-Gait sequence.

Version	Step Order	Desired Distance (cm)	Actual Distance (cm)	Angle (deg)	MPE (%)
1	Front Left Back Right	3.0	2.8	10.0	-8.2
2	Front Right Back Left	3.0	3.8	8.4	26.4
3	Back Left Front Right	3.0	2.8	8.7	-6.5
4	Back Right Front Left	3.0	3.0	10.3	-0.3

Table A.4: Experimental results for the different versions and food orders of the C-Gait sequence.

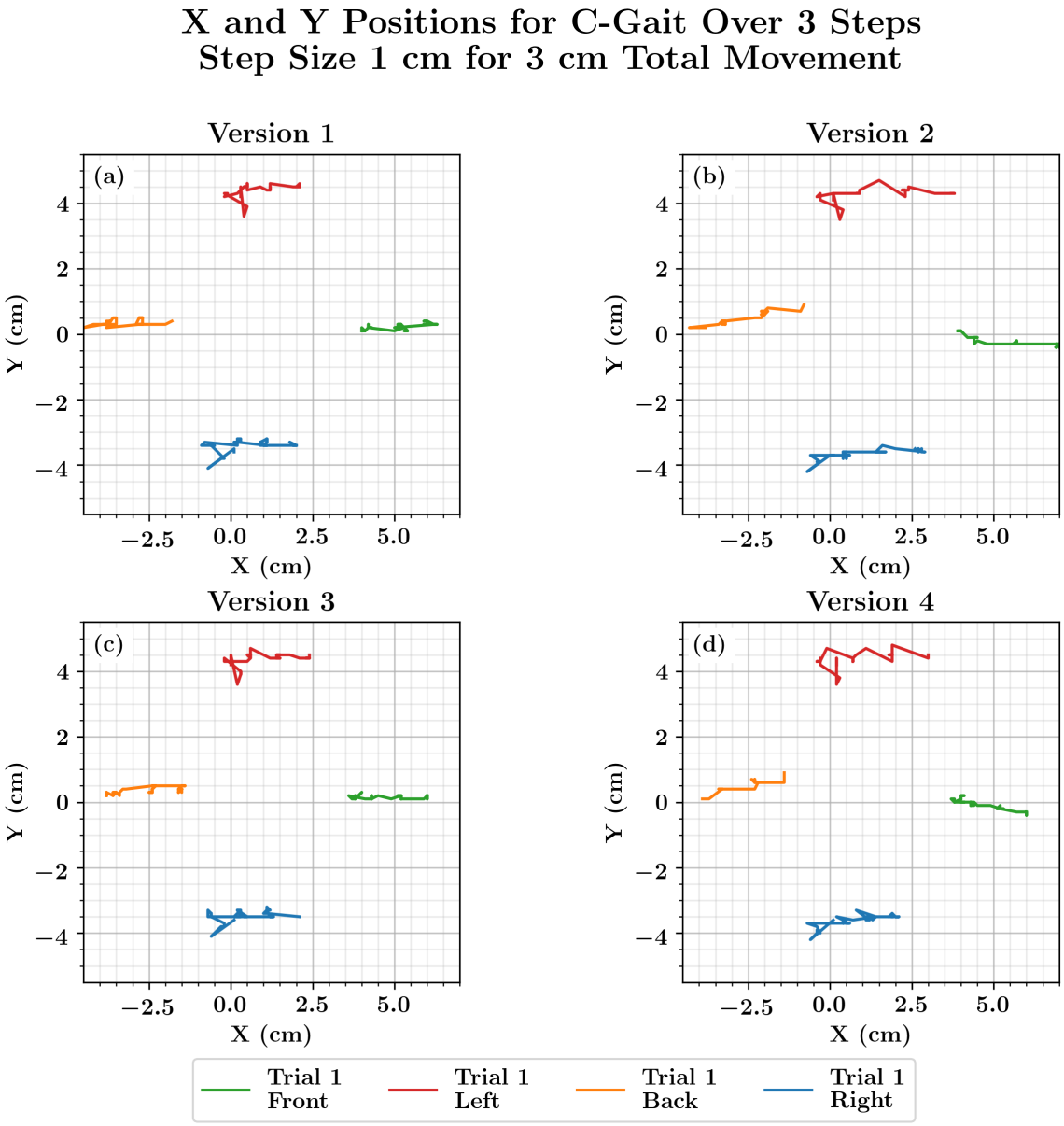

Figure A.4: Experimental results for the different versions and food orders of the C-Gait sequence.

Bibliography

[1] Evan Ackerman. Boston dynamics' spot robot dog now available for 74,500, 2020. 1.2.1

[2] Amazon. Cattongue grips non-abrasive anti slip grip tape, 2024. 2.1.1

[3] ANYbotics. Anymal technical specifications, 2022. 1.2.1

[4] Priyaranjan Biswal and Prases K Mohanty. Development of quadruped walking robots: A review. *Ain Shams Engineering Journal*, 12(2):2017–2031, 2021. 1.1, 1.2.1

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 5.2

[6] Lindsay Brownell. Robots with sticky feet can climb up, down, and all around, 2018. (document), 1.2, 1.2.1

[7] Chris Chang. Coordinated manipulation tasks using compliant delta robots. Jounral of 16-741 Mechanics of Manipulation, 2023. 2.2

[8] Chris Chang. Soft delta robots. GitHub Repository for Soft Deltas Robots, 2023. 2.2

[9] Jennifer Chu. Mini cheetah is the first four-legged robot to do a backflip, 2019. (document), 1.2, 1.2.1

[10] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 4

[11] Sébastien D De Rivaz, Benjamin Goldberg, Neel Doshi, Kaushik Jayaram, Jack Zhou, and Robert J Wood. Inverted and vertical climbing of a quadrupedal microrobot using electroadhesion. *Science Robotics*, 3(25):eaau3038, 2018. (document), 1.2.1, 1.2, 3.2

[12] Boston Dynamics. Boston dynamics expands global sales of spot® robot, 2020. (document), 1.2, 1.2.1

[13] Boston Dynamics. Spot operation: Demo gaits, 2020. (document), 1.2.1, 1.2

[14] Boston Dynamics. About spot, 2024. (document), 1.2.1, 1.3

[15] Boston Dynamics. Demo mode, 2024. (document), 1.2.1, 1.2

[16] Boston Dynamics. Spot - the agile mobile robot, 2024. 1.2.1

[17] Benjamin Goldberg, Neel Doshi, Kaushik Jayaram, and Robert J Wood. Gait studies for a quadrupedal microrobot reveal contrasting running templates in two frequency regimes. *Bioinspiration & biomimetics*, 12(4):046005, 2017. (document), 1.2

[18] Marco Hutter, Christian Gehring, Andreas Lauber, Fabian Gunther, Carmine Dario Bellicoso, Vassilios Tsounis, Péter Fankhauser, Remo Diethelm, Samuel Bachmann, Michael Blösch, et al. Anymal-toward legged robots for harsh environments. *Advanced Robotics*, 31(17):918–931, 2017. 1.2.1

[19] Actuonix Motion Devices Inc. Miniature linear motion series pq12, 2016. 2.1.1

[20] Kaushik Jayaram, Jennifer Shum, Samantha Castellanos, E. Farrell Helbling, and Robert J. Wood. Scaling down an insect-size microrobot, hamr-vi into hamr-jr. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10305–10311, 2020. (document), 1.2, 1.2.1

[21] Mert Ali İhsan Kalın, Cem Aygül, Altay Türkmen, Joanna Kwiczak-Yiğitbaşı, Bilge Baytekin, and Onur Özcan. Design, fabrication, and locomotion analysis of an untethered miniature soft quadruped, squad. *IEEE Robotics and Automation Letters*, 5(3):3854–3860, 2020. (document), 1.2, 1.2.1

[22] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301, 2019. 1.2.1

[23] Nathan Kau, Aaron Schultz, Natalie Ferrante, and Patrick Slade. Stanford doggo: An open-source, quasi-direct-drive quadruped. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6309–6315, 2019. (document), 1.2, 1.2.1

[24] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020. 1.2.3

[25] Zachary Manchester. Lecture notebooks 2023. Course Notes for CMU 16745, 2023. 1.2.3

[26] Pragna Mannam, Oliver Kroemer, and F Zeynep Temel. Characterization of compliant parallelogram links for 3d-printed delta manipulators. In *Experimental Robotics: The 17th International Symposium*, pages 75–84. Springer, 2021. 1.2.4, 2.1.1

[27] Hayley McClintock, Fatma Zeynep Temel, Neel Doshi, Je-sung Koh, and Robert J Wood. The millidelta: A high-bandwidth, high-precision, millimeter-scale delta

robot. *Science Robotics*, 3(14):eaar3018, 2018. 1.2.4

[28] Xiangrui Meng, Shuo Wang, Zhiqiang Cao, and Leijie Zhang. A review of quadruped robots and environment perception. In *2016 35th Chinese Control Conference (CCC)*, pages 6350–6356. IEEE, 2016. 1.1

[29] Anand Mishra. *Design, Simulation, Fabrication and Planning of Bio- Inspired Quadruped Robot [May 2014]*. PhD thesis, Cornell University, 05 2014. (document), 1.2

[30] Mike Oitzman. Anybotics introduces end-to-end robotic inspection solution, 2021. (document), 1.2, 1.2.1

[31] Abhishek Pandala, Randall T. Fawcett, Ugo Rosolia, Aaron D. Ames, and Kaveh Akbari Hamed. Robust predictive control for quadrupedal locomotion: Learning to close the gap between reduced- and full-order models. *IEEE Robotics and Automation Letters*, 7(3):6622–6629, 2022. 1.2.3

[32] Sarvesh Patil, Samuel C Alvares, Pragna Mannam, Oliver Kroemer, and F Zeynep Temel. Deltaz: An accessible compliant delta robot manipulator for research and education. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13213–13219. IEEE, 2022. 1.2.4

[33] Sarvesh Patil, Tony Tao, Tess Hellebrekers, Oliver Kroemer, and F Zeynep Temel. Linear delta arrays for compliant dexterous distributed manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10324–10330. IEEE, 2023. 1.2.4, 2.2

[34] Magdalena Petrova. Where four-legged robot dogs are finding work in a tight labor market, 2021. 1.2.1

[35] François Pierrot, Cl Reynaud, and Alain Fournier. Delta: a simple and efficient parallel robot. *Robotica*, 8(2):105–109, 1990. 1.1

[36] L Rey and R Clavel. The delta parallel robot. In *Parallel Kinematic Machines: Theoretical Aspects and Industrial Requirements*, pages 401–417. Springer, 1999. 1.1, 1.2.4

[37] Matteo Rovatti. A guide to quadrupeds' gaits - walk, amble, trot, pace, canter, gallop, 2023. (document), 1.2

[38] Zilin Si. Deltahands. GitHub Repository for DeltaHands, 2023. 2.2

[39] Zilin Si, Kevin Zhang, Oliver Kroemer, and F Zeynep Temel. Deltahands: A synergistic dexterous hand framework based on delta robots. *IEEE Robotics and Automation Letters*, 2024. (document), 1.1, 1.2.4, 1.5, 1.2.4, 1.3, 2.1.1, 2.1.1, 2.1.2

[40] Hamid Taheri and Nasser Mozayani. A study on quadruped mobile robots. *Mechanism and Machine Theory*, 190:105448, 2023. 1.1

[41] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012. 1.2.3

[42] Russ Tedrake. *Underactuated Robotics*. MIT, 2023. 1.2.3

[43] Hyper Triangle. Delta robot kinematics. Mirror of Trossen Robotics Forum, 2013. 2.3, 2.3.1

[44] Unitree. Unitree a1, 2024. 1.2.3

[45] Chao Wang, Hongzu Li, Zezhan Zhang, Peifeng Yu, Lihao Yang, Jiale Du, Yi Niu, and Jing Jiang. Review of bionic crawling micro-robots. *Journal of Intelligent & Robotic Systems*, 105(3):56, 2022. 1.2.1

[46] Patrick M Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. Optimization-based control for dynamic legged robots. *IEEE Transactions on Robotics*, 2023. 1.2.3

[47] Robert L. Williams, II. The delta parallel robot: Kinematics solutions, 2016. 1.2.4, 2.3

[48] Rui Wu Xu, Kai Chin Hsieh, Un Hou Chan, Hoi Un Cheang, Wei Kai Shi, and Chi Tin Hon. Analytical review on developing progress of the quadruped robot industry and gaits research. In *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*, pages 1–8. IEEE, 2022. (document), 1.2.2, 1.2

[49] Yuhai Zhong, Runxiao Wang, Huashan Feng, and Yasheng Chen. Analysis and research of quadruped robot's legs: A comprehensive review. *International Journal of Advanced Robotic Systems*, 16(3):1729881419844148, 2019. 1.1