Projected Profile Humanoid Crawl Gait and Lidar Based Navigation Using GODZILA

THESIS

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

MASTER OF SCIENCE (Electrical Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Griswald Brooks

January 2017

Approved:

_____
Advisor

_____
Date

_____
Department Head Signature

_____
Date

University ID: N17150539
Net ID: gwb225

VITA

Dec., 1983	Born
Jun., 2001	Graduated from Inter-Lakes High School.
Sep., 2001	Enlisted into the US Navy to be trained as an Electronics Technician.
Dec., 2007	Seperated from US Navy with Honorable Discharge.
Jan., 2007	Entered the Lakes Region Community College as a Computer Technologies major.
May, 2010	Received AS Degree from the Lakes Region Community College.
Jan., 2010	Entered the Polytechnic Institute of NYU as a Computer Engineering major.
May, 2013	Received BS Degree cum laude from the Polytechnic Institute of NYU.
Jan., 2014	Entered the NYU Polytechnic School of Engineering as a Electrical Engineering major.
Jan., 2017	Received MS Degree from the NYU Tandon School of Engineering.

ACKNOWLEDGEMENTS

I would like to thank Professor Farshad Khorrami for the guidance and opportunities given to me over the years, and throughout the course of this thesis. Without question, had it not been for his motivation and persistence, this thesis would not have come to fruition.

I would also like to thank Dr. Krishnamurthy for his help understanding the more difficult concepts and providing inspiration when investigating new paths. Finally, thanks to my colleagues Brian Cairl and Agraj Jain for their help with implementation efforts and collaborations.

I would like to dedicate this thesis to my father Gerald W. Brooks and mother Judy Lawley. Throughout this process, they have always supported me and done whatever they could to ensure that I would succeed.

ABSTRACT

Projected Profile Humanoid Crawl Gait and Lidar Based Navigation Using GODZILA

by

Griswald Brooks

Advisor: Dr. Farshad Khorrami

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science (Electrical Engineering)

January 2017

This thesis details the development and implementation of the Projected Profile crawling gait and the GODZILA navigation algorithm using the Nao Humanoid robot by Aldebaran Robotics. Navigation is fundamental to the area of mobile robotics as the ability to traverse safely between locations enables the robot to perform other useful tasks such as surveillance, mapping, and transportation of objects. While planning paths around obstacles in the environment is essential, actuating the limbs of a walking platform to allow effective locomotion (known as gaiting) is paramount. The set of gaits a humanoid can perform directly impacts the traversable regions of the robot and can expand the navigable area. To this end, a crawling gait was developed as they allow the traversal across difficult terrain and beneath low hanging obstacles which would otherwise impede the travel of the robot.

The design and construction of the mobile platform used in the experiments is first detailed. This consisted of three major parts: the Nao H25 Humanoid Platform, a Hokuyo URG-04LX-UG01 Lidar, and a custom 3D printed mount to attach the Lidar to the Nao. The Lidar supplemented the Nao's sensor suite in order to provide detailed range information about environmental obstacles. This allowed the navigation system to operate with higher performance.

The theory behind both the navigation and crawling algorithms is discussed next. The navigation scheme used is the GODZILA path planning algorithm. Based on the Potential Field concept, it is a local navigation algorithm with trap escape strategy and straight-line path planner when the goal is within the line-of-sight. The Projected Profile crawling gait uses a multimodal kinematic approach to allow a humanoid robot to crawl under very low objects. The gait alternates between positioning the arms and legs while the torso is in contact with the ground and the arms and legs touching the ground while positioning the torso.

Lastly, the results from the navigation and crawling experiments are shown. During the navigation experiments, the Nao is placed in several environments with different obstacles to show the path planning effectiveness. The robot was commanded to a goal location indicated by a red cube that the Nao detected using its onboard camera. The Lidar was used to guide the robot around the obstacles. The crawling experiments showed that the Nao could crawl under obstacles as low as 8 inches. This is in contrast to the 23 inch standing height of the robot. The gait was also optimized to minimize the joint torque required to crawl.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

Introduction

1.1 Concept and Motivation

Humanoid robots are a very versatile mobile platform, allowing many different areas of research. In this thesis, the areas of navigation and crawling were explored. Mobile platforms allow for many interesting tasks. They can be used for security surveillance [23], create maps [42], or deliver objects from one place to another [41]. To allow for these applications, robots must be able to safely traverse through an environment. Figure 1 shows an example of this. Development of safe and robust navigation algorithms is important to achieving this end. To enable navigation on legged platforms, the robot must have an effective gaiting strategy. While walking is often explored, crawling can allow a robot to traverse highly irregular terrain as well as reach height constrained spaces. This broadens the set of observable areas as well as potential manipulation targets.

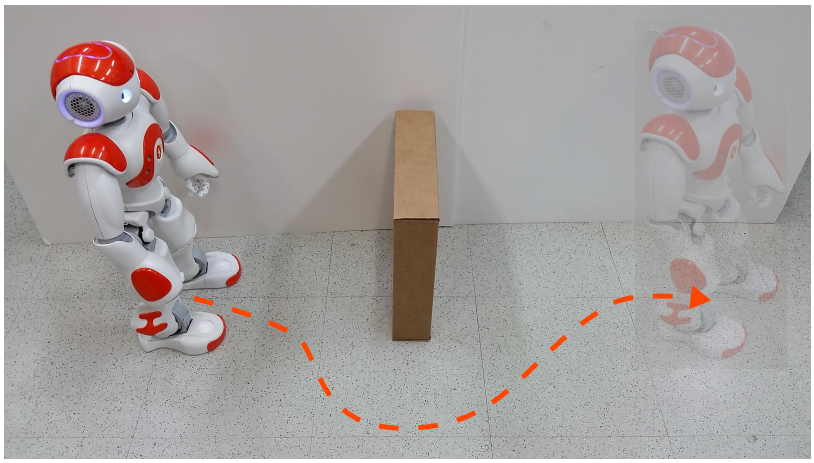

Figure 1: Example illustration of the Nao humanoid robot in an environment with an obstacle. To reach the left side of the area, it must plan a path through the environment.

1.2 Background

Humanoid robots offer a number of attractive features and interesting challenges. They range in size from the size of children [13, 14, 21, 43], to the size of an average adult [1, 9, 37, 45]. As a legged platform, it has a configuration that is adaptable to various terrains [20, 33, 46], while using a minimal number of limbs to accomplish locomotion. In addition to being a mobile base, humanoids are also manipulators. This opens up possibilities for interesting interactions with the environment [5, 25, 28]. Humanoid platforms are also a good choice for experimenting with robots that are expected to operate in human oriented environments as homes and offices are already configured to be traversable by humans. This is showcased by competitions such as the DARPA Robotics Challenge [2] and the RoboCup@Home Competitions [39] which both aim to further the field of robotics toward operating in human oriented environments. As well as being useful for navigation and manipulation, humanoid platforms are perhaps uniquely suited for studying interactions between robots and humans [22, 34]. They allow for increased control of an experiment and ensure a greater about of repeatability between experiments.

Moving a robot from one location to another in central to the problem of mobile robotics. Navigating a mobile robot is typically implemented using multiple approaches addressing different aspects of the problem. Search-based planners, such as A* search [12, 15, 32] and Rapidly Exploring Random Trees [27] are typically used to efficiently address global path planning, but often produce unsmooth paths. Planners which consider a continuous function space such as the Potential Field algorithm [6, 10, 17, 24] or Backstepping approaches [35, 36] produce smooth paths but can have problems with complicated environments. The navigation algorithm used in this thesis was based on a Potential Field approach [26], with which we have previous experience [3].

Humanoid robots have been used as a platform for exploring walking [7, 11, 44], which is important for the field of mobile human robots. Equally important is the ability to have multiple gaiting strategies available to traversing difficult terrain. Humanoid crawling has been explored before in [29, 30, 38] and provides an interesting opportunity to expand the set of feasible humanoid gaits [4].

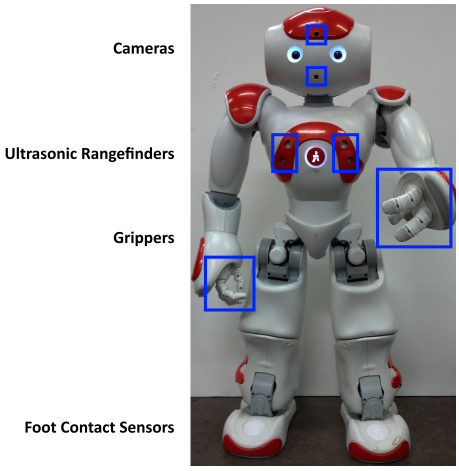

Figure 2: Coronal view of the Nao humanoid with a few pertinent features highlighted.

1.3 Platform Overview

The mobile platform used for these experiments was the Nao H25 Humanoid Platform from Aldebaran Robotics [40]. Figure 2 shows the Nao at the Control/Robotics Research Lab at NYU. It is a 25 degree-of-freedom humanoid with a rich set of sensors including cameras, sonars, joint angle sensors, and inertial measurement unit. It also features an onboard computer, Wifi, Ethernet, and USB 2.0. Combined with an extensive API, the Nao is a very capable platform, making it well suited to explore navigation and crawling. To supplement the sensing capabilities of the Nao, a Hokuyo URG-04LX-UG01 [18] scanning laser rangefinder (Lidar) was mounted to the robot. This allowed the Nao to have a high resolution survey of the local environment that the navigation algorithm used to avoid colliding with obstacles.

1.4 GODZILA Navigation

The navigation strategy implemented was based on the GODZILA planning algorithm [26]. It is a local navigation strategy which produces smooth paths. It also has a small memory footprint and is computationally cheap. Based on the Potential Field approach, it could serve as the foundation that other planning algorithms build upon. While a local planner, as is shown in Chapter III, it is capable of navigating the robot through parts of the environment which were not observable at the start of the path.

1.5 Projected Profile Crawl Gait

The crawling gait implemented was the Projected Profile gait [4]. It is a statically stable crawl with two phases. In the first phase, the robot is treated as a set of open chain manipulators, which allows the arms and feet to be positioned independently. In the second phase, the robot forms a closed chain, which is responsible for shifting the center-of-mass forward. By alternating these two phases, the robot is able to crawl forward. While implemented on a humanoid platform, this gaiting strategy is applicable to any platform which can be viewed in this manner, such as a quadrupedal robot.

1.6 Thesis Structure

Chapter II reviews in detail, the design considerations for the platform used, including the features of the Nao Humanoid Platform, Hokuyo Lidar, and the mounting hardware designed to join the two. Discussion of the navigation algorithm is presented in Chapter III while the results of the experiments are analyzed in Chapter V. Formulation of the crawling gait is detailed in Chapter IV and both simulations and experimental results of the crawl are shown in Chapter VI. Finally, the performance of the algorithms are reviewed, and future work is discussed in Chapter VII.

CHAPTER II

Nao Humanoid Platform

The platform used consisted of bringing together three major pieces. The Aldebaran Nao H25 was used for the mobile platform, the Hokuyo URG-04LX Lidar was the sensor used for obstacle detection, and a custom mounting system was built to mount the Lidar to the Nao.

The Nao H25 Humanoid is made by Aldebaran Robotics. It is a small humanoid platform which that can navigate a variety of indoor spaces but is safe for humans to work with without specialized safety equipment. It has a flexible API which allows operates at a range of abstractions. It allows for low-level command of individual joint angles when experimenting with gaiting and control algorithms or high-level command of navigation vectors when working with path planning. This makes the Nao an attractive mobile base for the topics covered in this thesis.

To augment the Nao's sensor suite, the Hokuyo URG-04LX Lidar was added to the platform. The URG allows for high resolution planar sensing of obstacles in the environment. This is very important for effective navigation in unstructured environments. The URG is also compact in size and light in weight making it appropriate for mobile robotics applications.

To mount the Lidar to the Nao, a custom mount had to be designed and constructed. The mount locates the Lidar around the torso of the Nao, which places it such that it does not interfere with the locomotion of the robot while navigating and minimizes the amount of motion seen by the Lidar due to gait oscillations.

Together, these three components comprised the platform used to perform experiments in this thesis. The Chapter III utilized all three of the major components while in Chapter IV only the Nao was used as the Lidar would have interfered with the crawl gait.

2.1 Hardware Overview

The hardware assembled for this platform can be divided into three major parts, the Nao H25 Humanoid Platform, a Hokuyo URG-04LX-UG01 Lidar, and a custom 3D printed assembly to mount the Lidar to the Nao. The Nao acted as the mobile base and provided onboard sensing and computation. Its large number of degrees of freedom and humanoid configuration were amenable to the exploration of crawling algorithms. The Nao is also equipped with two color cameras for viewing the environment and two sonars for measuring the range to objects. In the navigation experiment, the forward facing camera was used to estimate the pose to a goal object. Though the two sonars can be used for obstacle avoidance, previous experimentation showed they were insufficient for navigation. This is because in a number of scenarios the specular refections would cause confusing readings and the low angular resolution would prevent the robot from walking through traversable apertures. To supplement the Nao's sensor suite, the URG Lidar was mounted to the robot to provide accurate, high resolution range data for use with navigation. As the Nao does not have any external mounting points, a custom mount was designed and constructed to allow the URG to be mounted to the Nao.

2.2 Goal Object

Like many navigation algorithms, the one used in Chapter III requires the platform to have a goal location for which to traverse. For these experiments, the goal location was provided to the robot by means of a physical object that the robot could detect with its onboard camera. The Nao API provides a simple "red ball" tracker that can be used to track red objects via the onboard camera. It provides not only bearing information but also range by assuming the red object has a width of $0.06m$.

The goal was built as a $0.127m$ wide red cube. This was affixed to the top of a wooden dowel, mounted to a heavy wooden base to allow it to be easily placed. Figure 3 shows a picture of the cube. The dowel was approximately $0.6m$ tall to allow the robot to see the cube while minimizing the amount the head needs to pitch in order to keep the cube in view. Building the cube to be wider than the expected width allowed the target to be seen by the robot from longer distances. This longer range allowed for the construction of a larger arena as well as more robust tracking at shorter ranges. The goal was built as a cube rather than a ball because during testing the tracking algorithm did not seem to be affected by the change in geometry and a cube was easier to construct.

Figure 3: Red cube that the Nao tracked as the goal location.

2.3 Nao Hardware

The Nao Humanoid Platform is made by Aldebaran Robotics. Figure 4 shows a picture of the Nao at the Control/Robotics Research Lab of NYU Polytechnic School of Engineering. The robot weighs $5.4kg$ and has a maximum forward velocity over flat terrain of approximately $1\frac{m}{s}$. It has 25 degrees-of-freedom (DoF) embodied via a 2 DoF head, two 5 DoF legs, two 5 DoF arms, two 1 DoF hands, and a 1 DoF hip. Figure 6 shows a diagram locating each of the joints on the robot. Each joint has an absolute angular position encoder, while the feet each have four force sensitive resistors to detect ground contact. The head has two color cameras, one that faces forward and the other that faces downward to get a view of the terrain. The chest contains a 2-axis gyroscope and a 3-axis accelerometer for inertial measurement, and two sonar transmitter-receiver pairs for measuring the range to obstacles. Figure 5 shows a diagram locating the various robot sensors, including those not listed above. The sonars have a minimum range of $0.25m$ and a maximum range of $2.55m$. They have a resolution of $1cm$ and detect any obstacle within a 60° cone. The cameras have a $1.22Mp$ resolution, $30Hz$ frame rate, a horizontal field-of-view of 60.9°, and a vertical field-of-view of 47.6°. The robot is equipped with a 1.6 GHz Intel® Atom^{TM} CPU, 1 GB of RAM, and 2 GB of

Figure 4: Figure showing the Nao Humanoid Platform at the Control/Robotics Research Lab at NYU Polytechnic School of Engineering.

Flash memory. It also has Gigabit Ethernet, 802.11 b/g/n WiFi, and USB 2.0. Nao's operating system is called NAOqi OS which is a customized version of Gentoo Linux. Being a Linux distribution, the robot can be easily accessed using ssh, scp, and ftp. This allows familiar command line tools to be used to script behavior and manage services. The robot can be natively programmed using the provided C++ or Python API.

The Nao is a very feature rich platform suitable for a variety of autonomous applications. The following sections will review specific aspects of the robot relevant to Chapters V and VI.

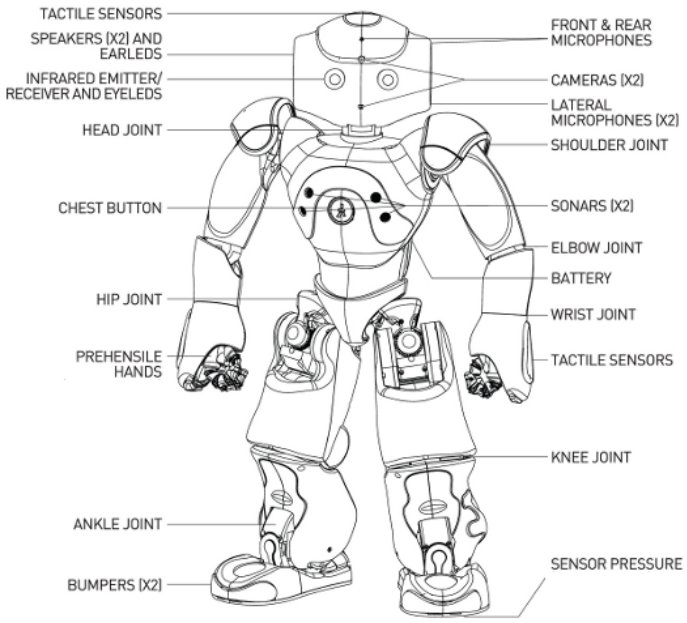

Figure 5: Figure locating the various features on the robot.

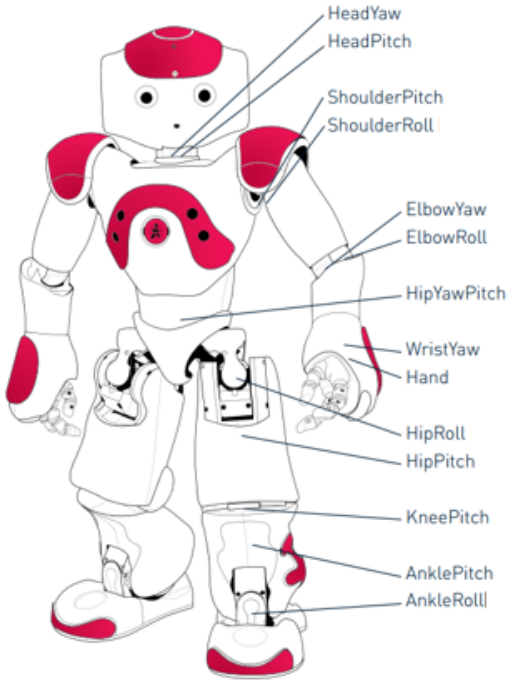

Figure 6: Figure indicating the various joint locations and their names.

2.3.1 Frame Definitions

The NAOqi API defines three frames that can be seen in Figure 7. They are FRAME_TORSO, FRAME_ROBOT, and FRAME_WORLD. The first two frames, FRAME_TORSO and FRAME_ROBOT, are rigidly attached to the robot, while FRAME_WORLD is an inertial frame initialized when the robot first starts.

FRAME_TORSO is a frame rigidly attached to the torso of the Nao. The positive Z-axis points up through the head of the robot, while the positive X-axis points forwards. It is useful for referencing different parts of the robot such and joint frames and manipulation targets relative to the robot. The frame origin along the X and Y directions is in the geometric center of the torso, while Figure 9 shows its Z location on the torso as the point that the neck and hip offsets are referenced with respect to.

FRAME_ROBOT is a frame whose origin is between the feet of the robot. It's positive Z-axis always points upwards and positive X-axis always points forwards. It is useful when specifying navigation targets, relative to the Nao's current pose.

FRAME_WORLD is a frame that is coincident with FRAME_ROBOT when the Nao is started, and remains static for the life of the run. It is useful for specifying navigation and manipulation targets in a global coordinate sense.

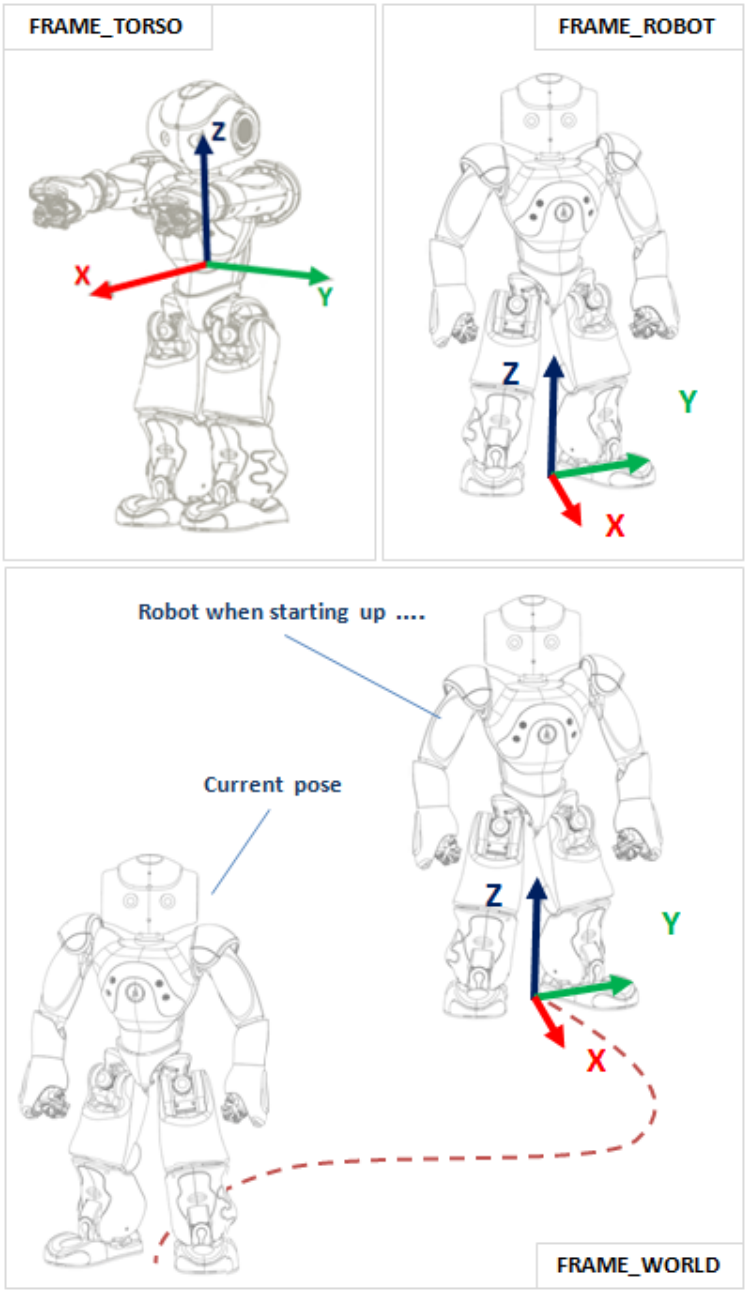

Figure 7: Nao frames.

Orientation targets for the Nao are commonly expressed using Tait-Bryan angles, more commonly known as roll, pitch, and yaw. As can be seen in Figure 8, yaw is expressed as rotation about the Z-axis, roll is about the X-axis, and pitch is about the Y-axis. When the robot is being referenced from the FRAME_TORSO and is initialized in the StandZero posture referenced in the proceeding section, the naming conventions of the joints become clear. All joints whose axis are parallel to the torso frame Y-axis are

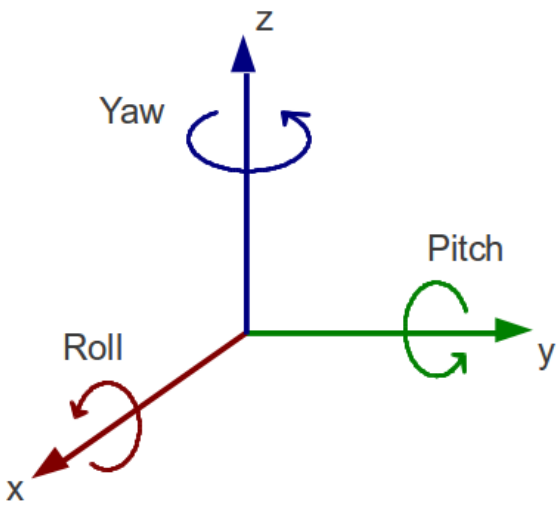

Figure 8: Definition of roll pitch and yaw.

postfixed with the word "Pitch", those parallel to the Z-axis are postfixed with the word "Yaw", and X-axis with "Roll". The only exception to this rule is the ganged pelvis joint axis. Each side of this degree of freedom rotates along a vector that is in the plane formed by the torso Z-Y axes and is therefore postfixed with the term "YawPitch".

2.3.2 Links and Joints

The Nao H25 is a bipedal platform with two legs, two arms, and an articulate head. Each arm is composed of 5 joints and 3 non-zero length links, as are the legs. The neck is composed of 2 joints. There are also three fingers per hand and two feet. Detailed measurements and specifications on the Nao H25 can be found on the Aldebaran Website [40]. We will review a few relevant characteristics of the robot below.

Links Figure 9 shows some of the large scale link lengths of the Nao H25. The hip and neck Z offsets are measured with respect to FRAME_TORSO, as are the shoulder and hip Y offsets. The thigh and tibia lengths are each roughly $100mm$. Their total is $202.8mm$. The neck and hip offsets sum to 211.5, making the torso approximately the same length as the legs. The humerus of the robot is nearly $100mm$ and the forearm to the wrist is half as long. The wrist to the hand is about half as long as the humerus, meaning in total, the arms are about as long and the legs.

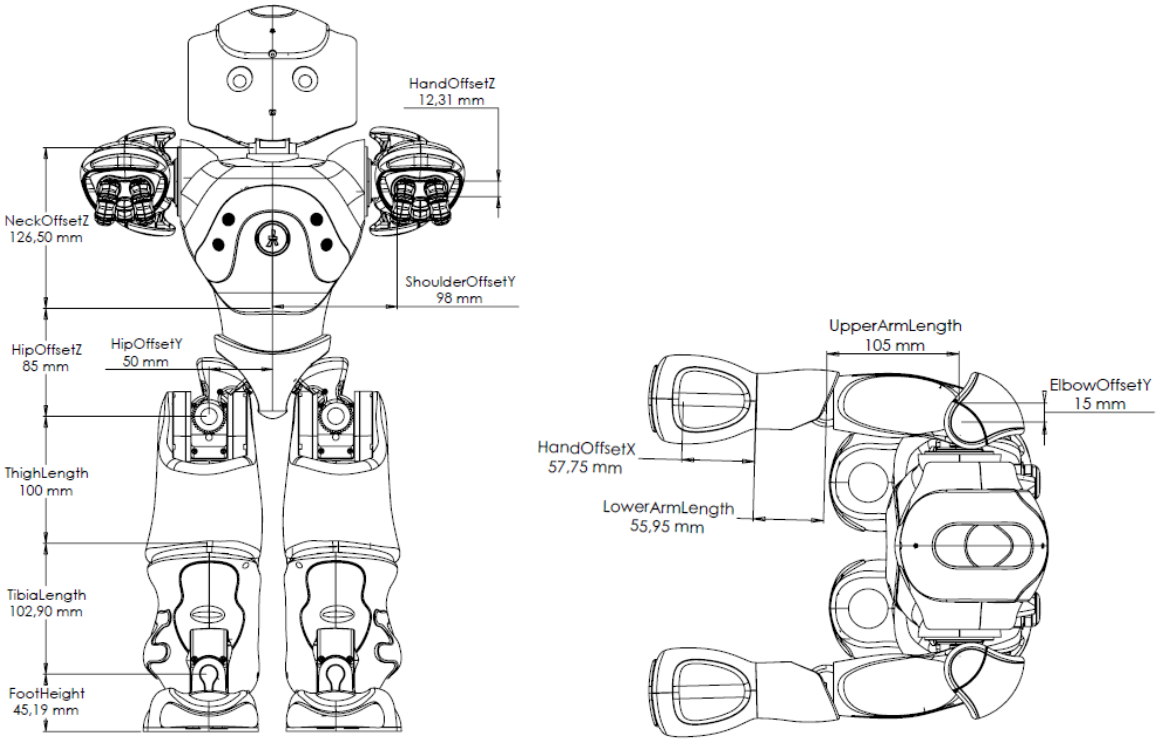

Figure 9: Nao link lengths.

Joints The Nao H25 has 25 degrees-of-freedom. Each of the two hands constitute one degree of freedom even though each hand has 8 rotary joints. This is because they are mechanically linked through a single actuator that pulls a cable to close the fingers. The hand opens again through relaxation of the cable which allows springs to extend the fingers. This means the Nao can hold its hand closed, but not open. The two pelvis joints are mechanically linked, resulting in one degree of freedom. A diagram of the pelvis can be seen in Figure 15. The result is the robot can spread or close both legs, but not an individual leg. This makes certain crawling techniques inaccessible to this body configuration. The 22 remaining joints are each independently actuatable and constitute the remaining degrees-of-freedom. Each leg and arm have 5 joints while the neck has two. As shown in Figure 10, one neck joint is for pitch and the second for yaw. Figures 11 and 12 detail the arms which have a shoulder joint for both pitch and roll, an elbow joint for both roll and yaw, and a wrist joint for yaw. Figures 13 and 14 detail the legs which have a hip joint for both pitch and roll, a knee joint for pitch, and an ankle joint for both pitch and roll. Table 1 shows the order in which each joint appears in their respective kinematic chains.

Figures 10 through 15, show the range of rotation for each joint. The joints are

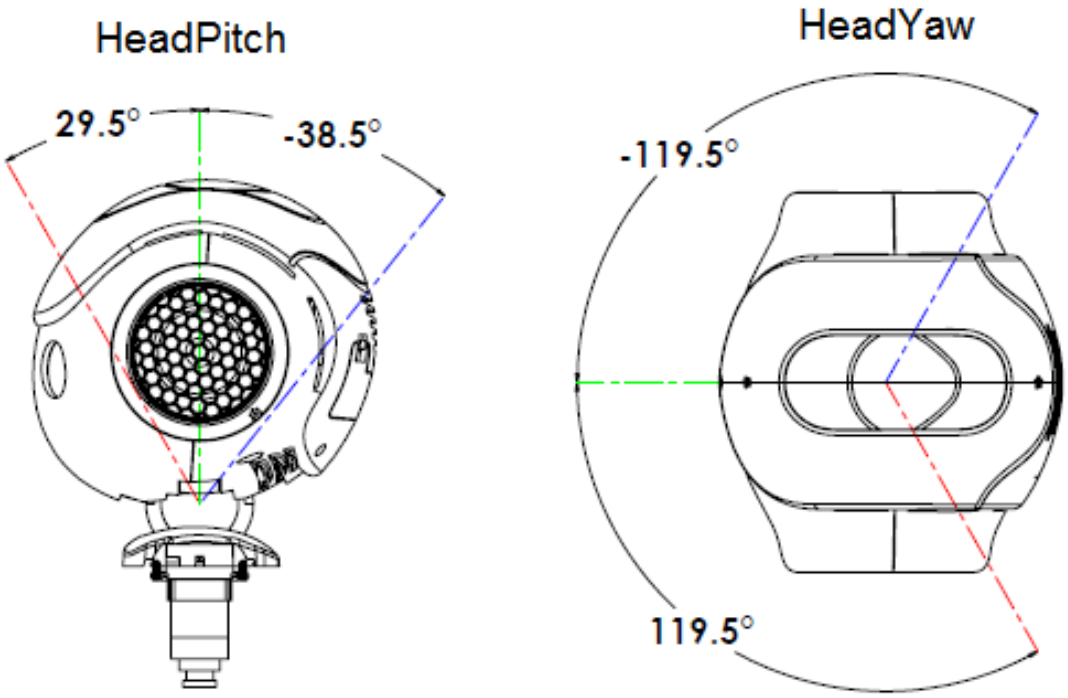

Figure 10: Diagram of the angular ranges of the neck joints in degrees.

shown in their positions representing zero angle. The green line is a reference representing the zero. The red line represents the maximum angle the joint can actuate to, while the blue line is the minimum angle.

Each joint contains a magnetic rotary encoder for measuring angular position. These encoders have a 0.1° precision. A motor current sensor for each joint can also be read to get a sense of how much torque each joint is producing, though in practice this reading is a poor approximation as it is influenced by other factors such as motor control loop methodology.

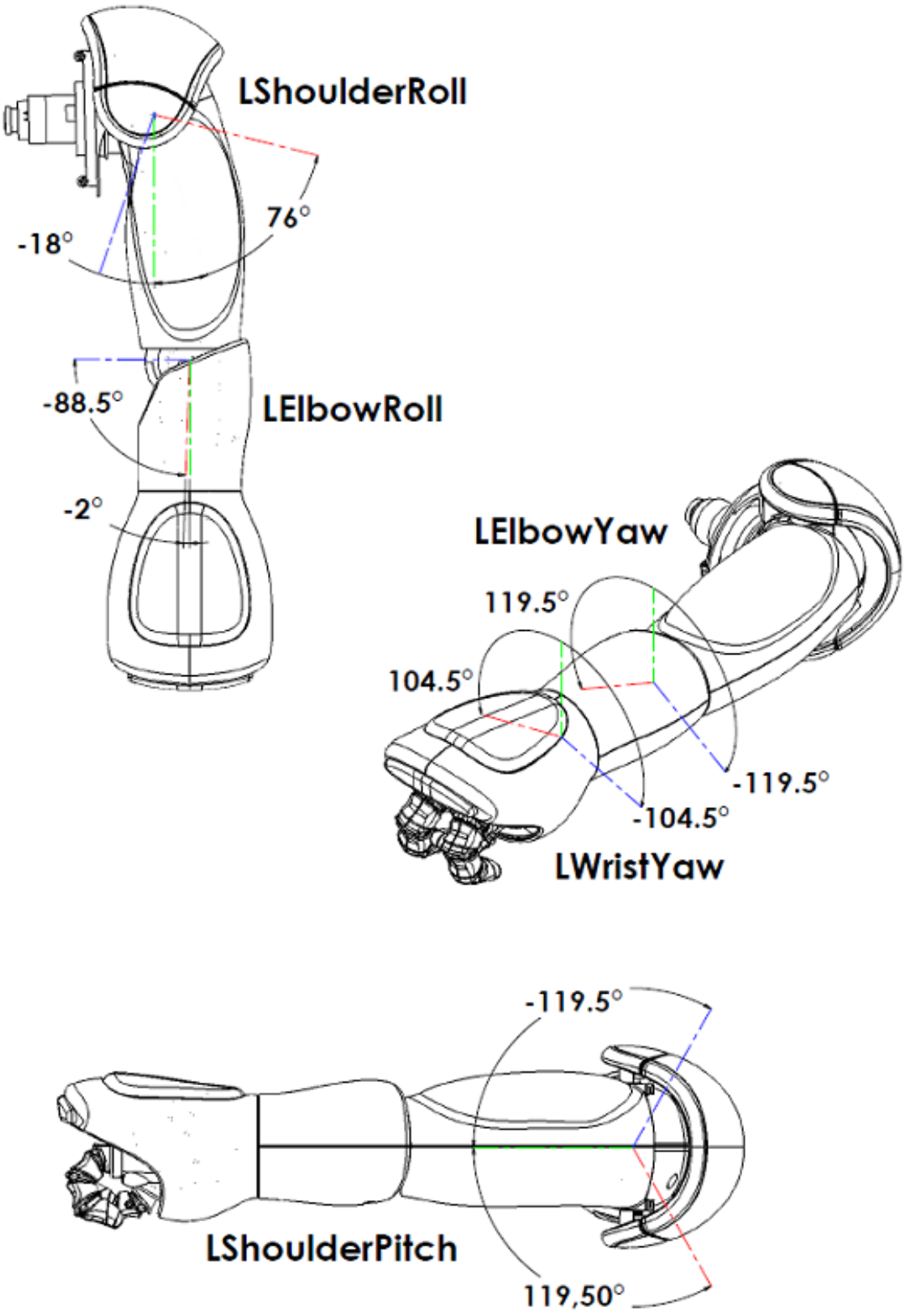

Figure 11: Diagram of the angular ranges of the left arm joints in degrees.

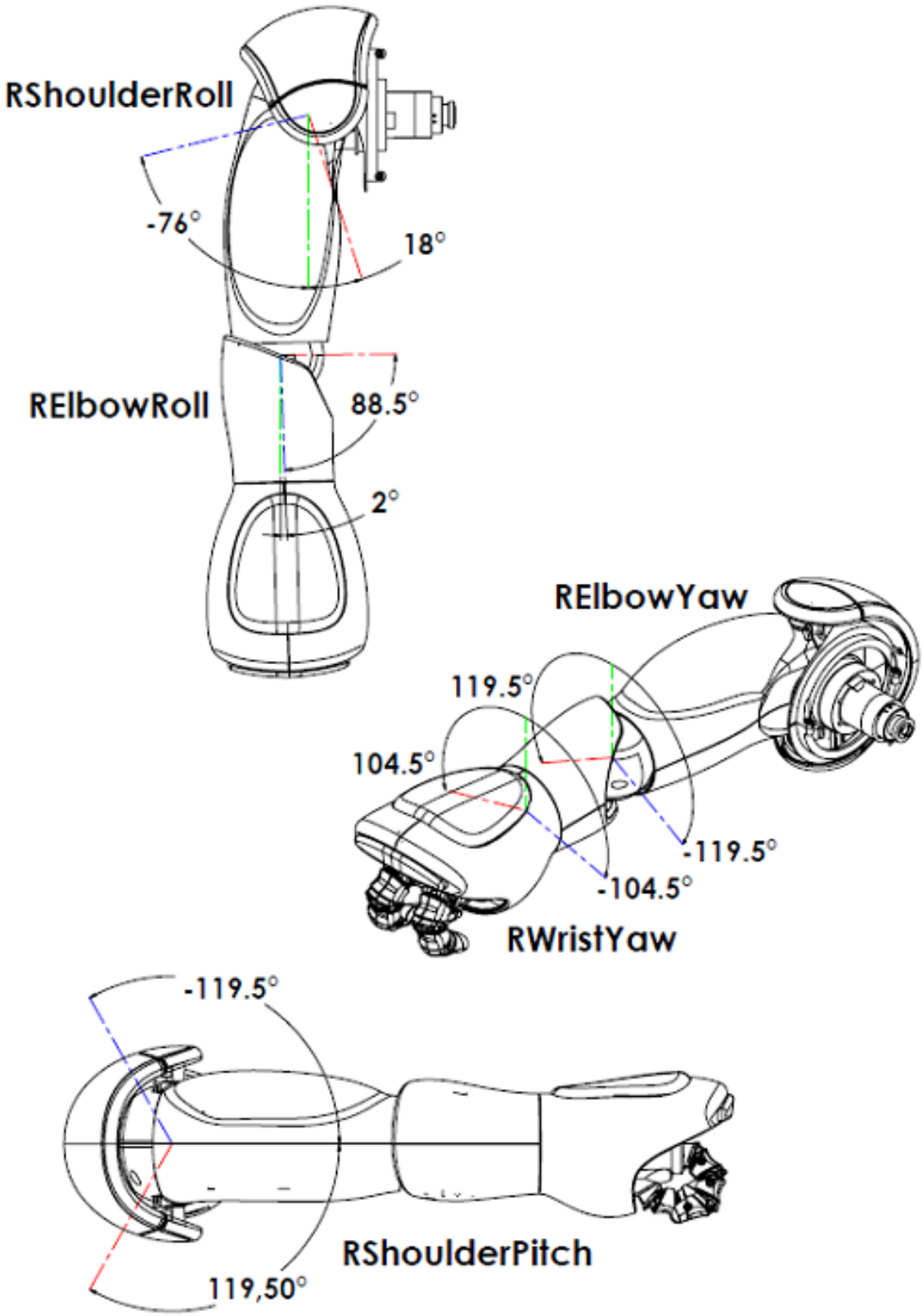

Figure 12: Diagram of the angular ranges of the right arm joints in degrees.

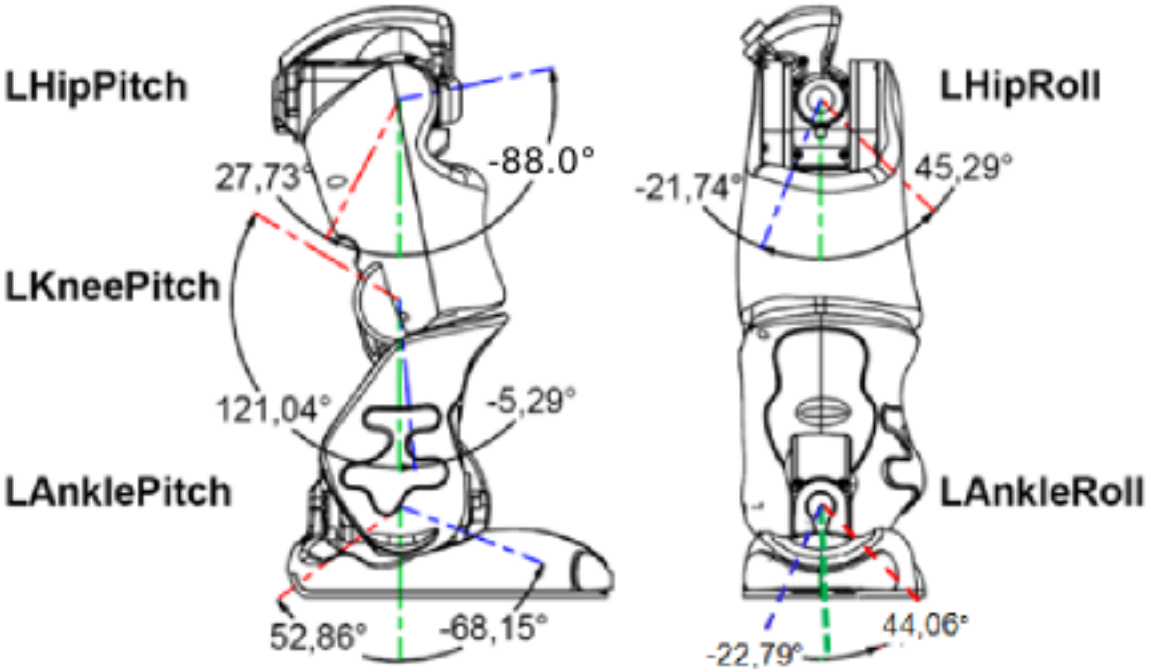

Figure 13: Diagram of the angular ranges of the left leg joints in degrees.

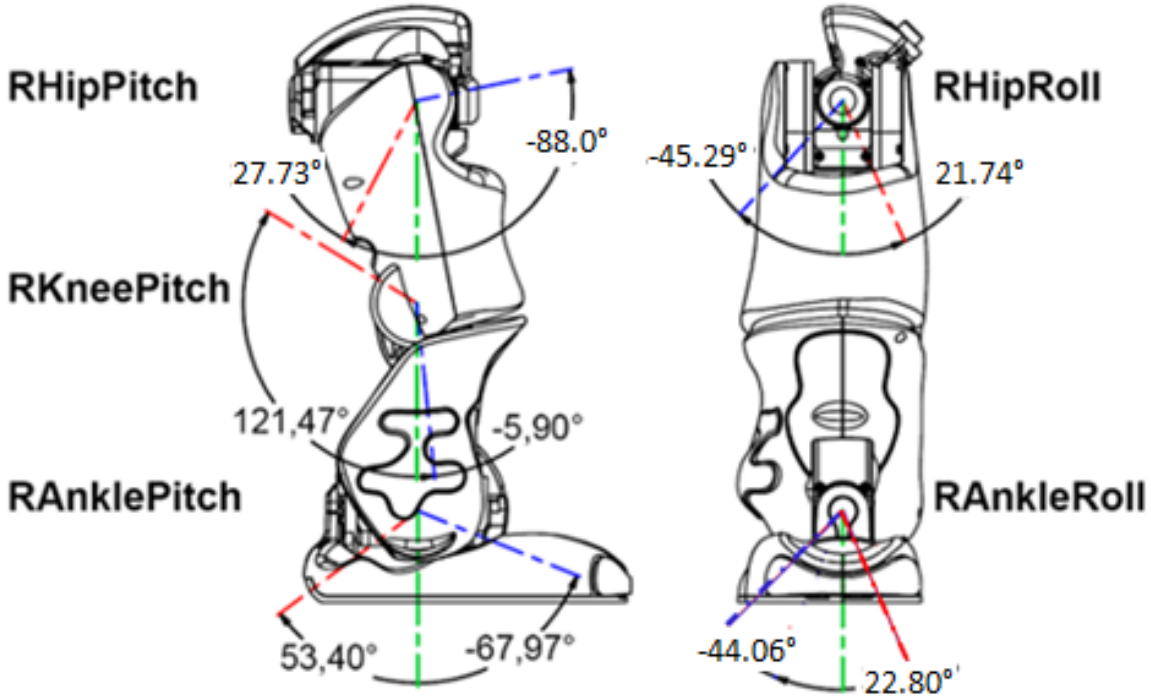

Figure 14: Diagram of the angular ranges of the right leg joints in degrees.

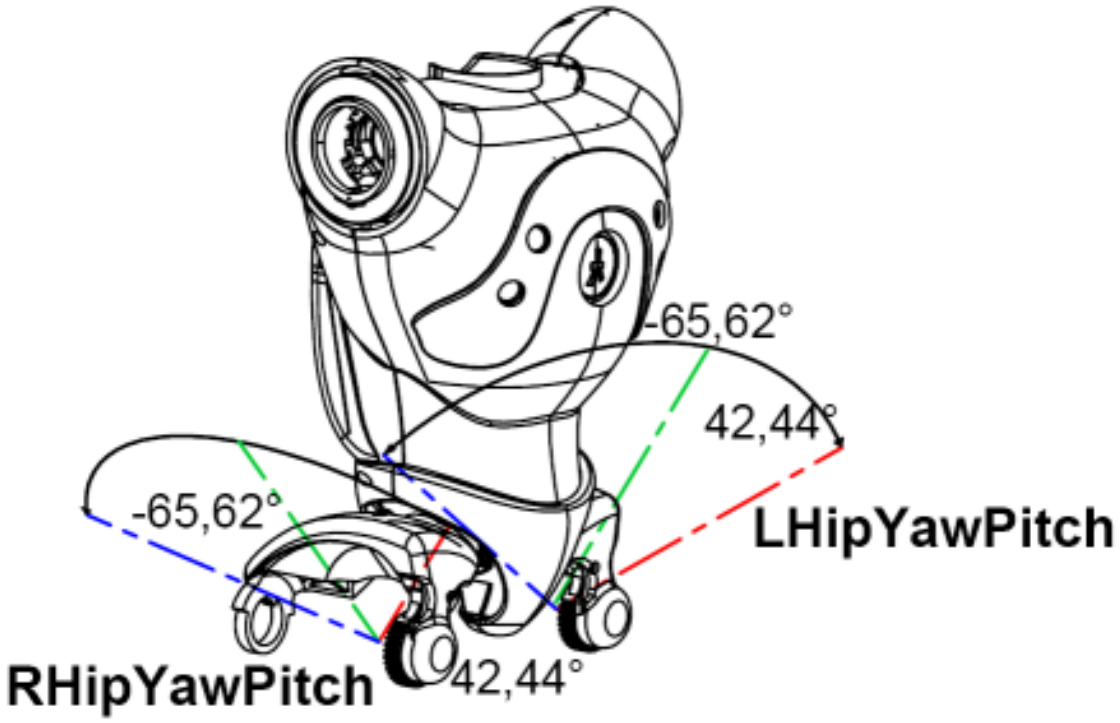

Figure 15: Diagram of the angular ranges of the pelvis joints in degrees. While there are two joints, they are mechanically linked such that when actuated, the joints are at the same angular position.

Chain	**Joints**
Head	HeadYaw → HeadPitch
Left Arm	LShoulderPitch → LShoulderRoll → LElbowYaw → LElbowRoll → LWristYaw → LHand
Right Arm	RShoulderPitch → RShoulderRoll → RElbowYaw → RElbowRoll → RWristYaw → RHand
Left Leg	LHipYawPitch → LHipRoll → LHipPitch → LKneePitch → LAnklePitch → LAngleRoll
Right Leg	RHipYawPitch → RHipRoll → RHipPitch → RKneePitch → RAnklePitch → RAngleRoll

Table 1: Nao H25 kinematic chains that stem from the torso. Each row lists the sequence of joints for that chain.

Joint Rotation Direction While each joint has a positive and negative angular limit, the direction of each joint is not relative to the previous joint frame, but rather the torso frame. This can be seen most clearly in Figure 16, which is a diagram showing the arms side-by-side, highlighting the roll joints. It shows that positive joint angle in the roll joints always corresponds to moving to the left side of the robot, if all other joints are in the zero position. Negative joint angle always moves roll joints to the right. This is in contrast to other conventions which might hold that positive roll angle might move the joints "outward" or "inward". As usual, joint angle frames are attached to joints and not the torso frame. This means that if the right arm's elbow yaw joint rotated by a positive 90° and all other joints are at zero angle, positive elbow roll would now correspond to the joint pitching upwards, as the elbow yaw joint precedes the roll joint in the kinematic chain. Figure 17 shows that when all joints are in their zero positions, the arms point forwards.

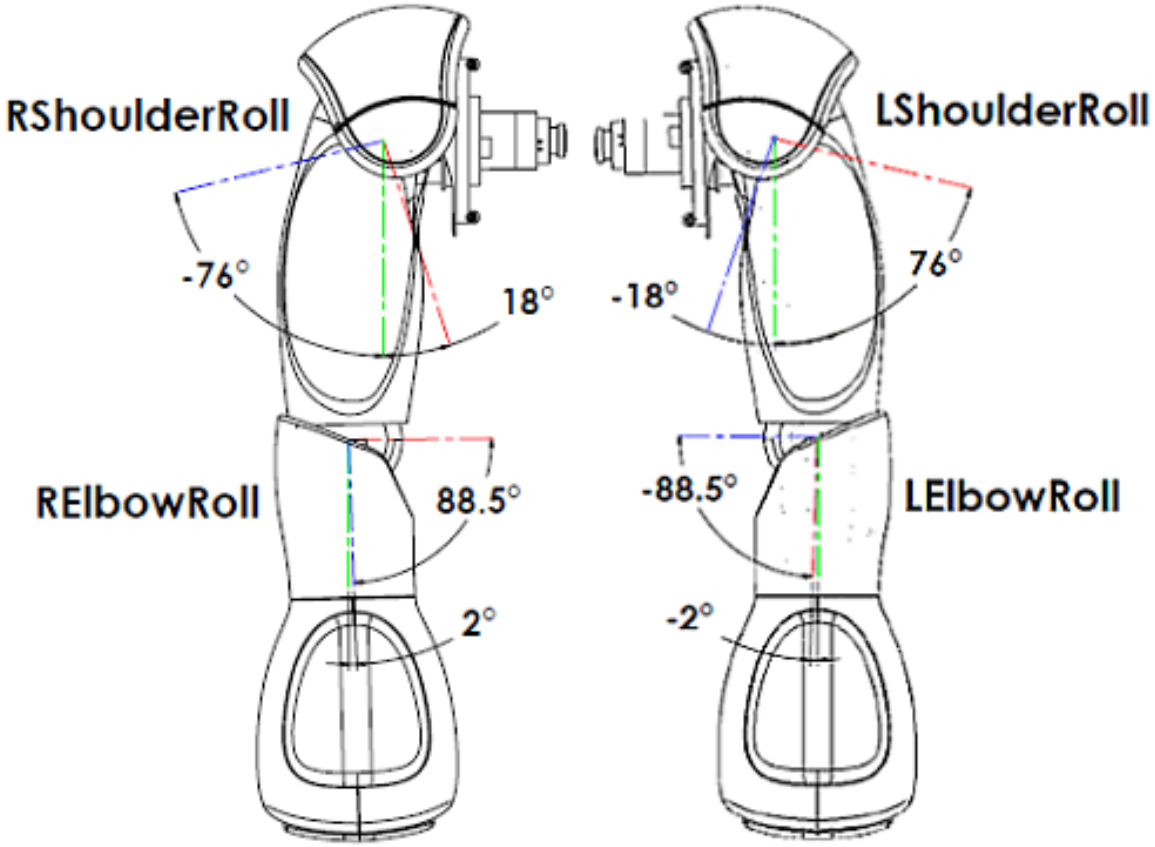

Figure 16: Figure showing arm joints and how they are a reflection.

2.3.3 Joint Torques

The actuators for each joint are sized according to their expected capacity requirements. The leg joints, for example, use motors with high torque as they are expected to counteract the forces seen due to walking. This is in contrast to the hand actuator motors which aren't required to have large grasp force as the Nao is not rated to move heavy objects. Table 2 lists the different motors with their rated torques used in each of the joints. Table 3 lists the gearboxes used in each actuator while Table 4 lists which motor and actuator used in each joint. These figures are taken from the Aldebaran Website [40].

2.3.4 Postures

The NAOqi API contains a module called ALRobotPosture which allows the robot to be commanded to any one of a number of predefined kinematic configurations, called "postures". These postures are shown in Figure 17. The robot can be commanded to transition to these postures in two different ways. Trivially, the first method is that the robot can command its joints to go directly to the joint angles corresponding to each posture. This is convenient in situations where robot safety is not a factor, such as tests or initializations. More interestingly, the second method is the robot can plan its transition from its current joint configuration to the commanded posture in a way that avoids self collision and prevents the robot from endangering itself, such as when the robot might fall over. This can involve the robot first transitioning to intermediate postures or executing maneuvers that ensure robot stability. One example is when the robot is standing and is commanded to the "Sit" posture. In this scenario, the Nao will place its right hand on the ground behind it in order to prevent itself from falling over as it moves its legs. Using this planning paradigm, it can be commanded to any posture from any joint configuration safely. This feature is particularly useful when planning a

Motor Type	1	2	3	4
Model	22NT82213P	17N88208E	16GT83210E	DCX16S01GBKL651
No Load Speed	$8,300rpm$	$8,400rpm$	$10,700rpm$	$12,700rpm$
Stall Torque	$68mNm$	$9.4mNm$	$14.3mNm$	$22.4mNm$
Nominal Torque	$16.1mNm$	$4.9mNm$	$6.2mNm$	$5.53mNm$

Table 2: Various parameters for each of the motors used in the joint actuators.

Motor Type	1	2	3	4
Ratio A	201.3	50.61	150.27	150.27
Ratio B	130.85	36.24	173.22	

Table 3: Speed reduction gear ratios used in the joint actuators. Each motor type has a two different gear ratios that can be used with it in the joint actuator, ratio A or B.

Chain	**Joint**	**Motor Type**	**Ratio**
Head			
	HeadYaw	3	A
	HeadPitch	3	B
Arms			
	ShoulderPitch	3	A
	ShoulderRoll	3	B
	ElbowYaw	3	A
	ElbowRoll	3	B
	WristYaw	2	A
Hands			
	Hand	2	B
Legs			
	HipYawPitch	1	A
	HipRoll	1	A
	HipPitch	1	B
	KneePitch	1	B
	AnklePitch	1	B
	AnkleRoll	1	A

Table 4: Actuator configuration for each joint consisting of motor type and gear reduction ratio.

full navigation sequence and is used in Chapter IV to transition the robot from walking to crawling.

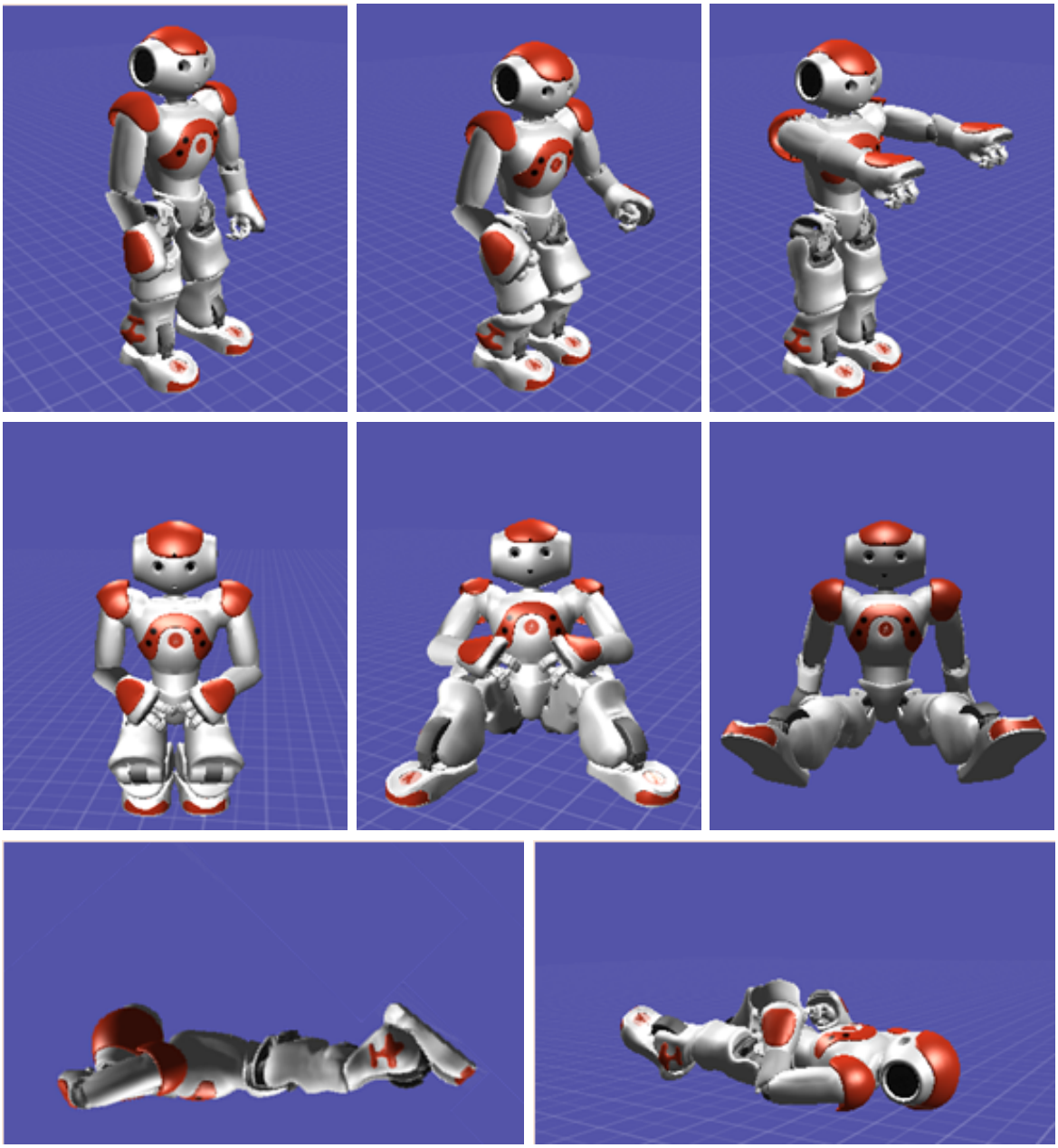

Figure 17: Posture options for the Nao H25. From the upper left pane to the lower right pane they are: Stand, StandInit, StandZero, Crouch, Sit, SitRelax, LyingBelly, LyingBack.

2.4 Lidar Hardware

The sensor used to detect environmental obstacles was a Hokuyo URG-04LX-UG01, which can be seen in Figure 18. The URG is a 2D scanning laser rangefinder, more commonly known as a Lidar. Lidar is a portmanteau for *Li*ght *D*etection *a*nd *R*anging and operates using principles similar to Radar and Sonar. 2D Lidars provide a number of discrete range measurements along a plane. This effectively gives a "top-down" or "blue-print" style view of an environment. This, when mounted to a robot, allows the detection of objects in one plane of the environment so that the robot can avoid or plan a path around them.

2D scanning Lidars such as the URG, sense the environment by emitting a laser beam and measuring the time it takes for the emitted photons to return to the sensor. This time is used to compute the range to the obstacle that the beam encountered. Commonly, photons are not actually timed, but rather the phase difference of the emitted light is used to estimate the time. Following this measurement, the laser is then mechanically rotated to another angular position to take another reading. The process is repeated through the entire scanning angle of the sensor, and the set of range-angle pairs is considered "one scan". The URG uses a spinning mirror to rotate the beam, while other sensors rotate the entire laser assembly. An example of the data returned from the URG can be seen in Figure 19. The URG was placed in a hall with a doorway to its left.

The URG is a good choice for these experiments due to its lightweight and small dimensions. This allows it to be easily mounted to the Nao Humanoid while respecting the payload capacity of the robot. It has a wide scan angle, high angular resolution, and a good range for this application. A high angular resolution allows the robot to detect narrow obstacles that might obstruct the robot's path, yet unlike sonar, provide detailed information about unobstructed areas the robot can traverse. While the URG is rated for indoor use only, this is sufficient for our application as the Nao is not often used in outdoor situations. Lastly, the sensor can be operated through a single USB port, sourcing power from and providing data to the robot. This means no additional hardware is necessary for the Nao to make use of the sensor, as a USB port is available in the back of the robot's head. Figure 20 shows a mechanical drawing of the sensor, while Table 5 lists some of the relevant specifications of the URG.

Figure 18: Photograph of the URG-04LX-UG01 Lidar. It is shown with a USB cable plugged in, which is the method of powering and retrieving data from the Lidar.

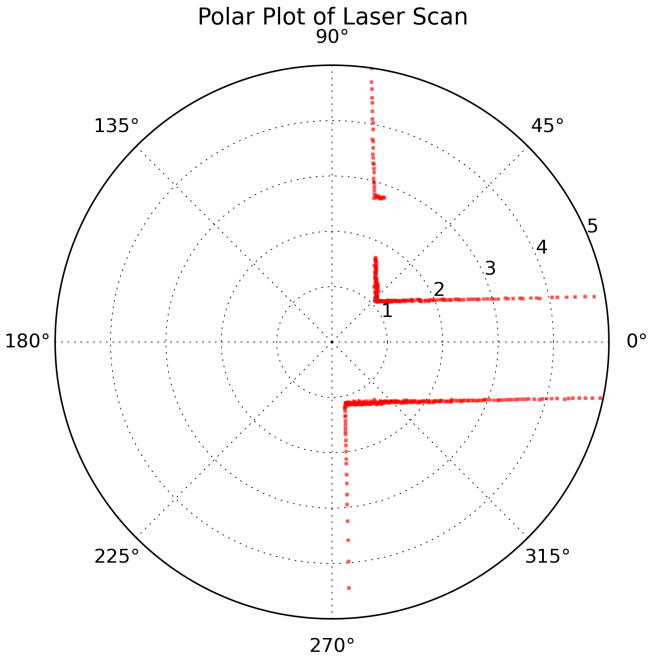

Figure 19: Scan from the URG. The sensor is in the center of the figure. The radius of the figure is limited to 5 meters, which is 1 meter greater than the maximum rated range of the sensor. While the unit will still provide data at this range, its accuracy is unrated. In this scan, the sensor is facing a hall, with a doorway to its left.

Properties	Condition	Min	Typ	Max	Units
Scanner					
View Angle			240		degrees
Angular Resolution			0.36		degrees
Range		20		4000	mm
Linear Resolution			1		mm
Accuracy	Distance: 20 mm to 1000 mm		± 30		mm
	Distance: 20 mm to 4000 mm		± 3		%
Mechanical					
Update Rate			100		ms/scan
Weight			160		g
Width			50		mm
Length			50		mm
Height			70		mm
Electrical					
Voltage Rating		4.75	5.0	5.25	V
Current Draw			500	800	mA

Table 5: This table contains various specifications of the Hokuyo URG-04LX-UG01. These values were taken from the datasheet of the Lidar, which can be found here [18].

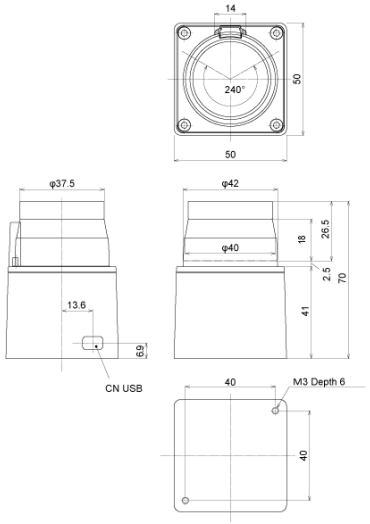

Figure 20: Mechanical drawing of the URG-04LX-UG01. The compact size of the URG makes it easy to mount to the Nao.

2.5 Lidar Mount

To attach the URG Lidar to the Nao, a custom Lidar mount was built. The mount needed to hold the Lidar such that obstacles in front of the robot could be sensed. The mount should also be lightweight, rigidly attached, and allow the robot to move its head to track the red cube. Finally, it should not inhibit the use of the body mounted sonars, in the event that the sonars are used in the future.

The Lidar mount was designed as a tray positioned at waist-height, in the front of the robot. As the Nao is not equipped with any mounting points, the mount was attached using straps. The rigid parts were manufactured out of PLA plastic using a 3D printer. The printed parts that would contact the robot were padded with foam to create a surface that would conform to the Nao's body and hold securely. Figure 21 shows a CAD model of the finished assembly.

The mount consisted of two major parts: the front subassembly and the back plate. The front subassembly held the Lidar to the Nao at waist height and the back plate provided a place to anchor the straps from the front. Both pieces sandwiched the chest of the Nao to create a tight jacket that the robot would carry. A picture of the assembly mounted to the robot can be seen in Figure 22, which shows the straps over the shoulder of the robot and around the waist.

Figure 23 shows the back, front, and side view of the CAD model of the Lidar mount with the URG, torso, and head of the Nao. The back view shows the back plate which is an anchor for the straps. The front view shows the front subassembly holding the URG. The side view shows how these pieces conform to the shape of the robot. One set of straps go from the front of the front subassembly to the back plate, while the second set go around the waist. When installed, they form a tight sandwich that can support the weight of the robot.

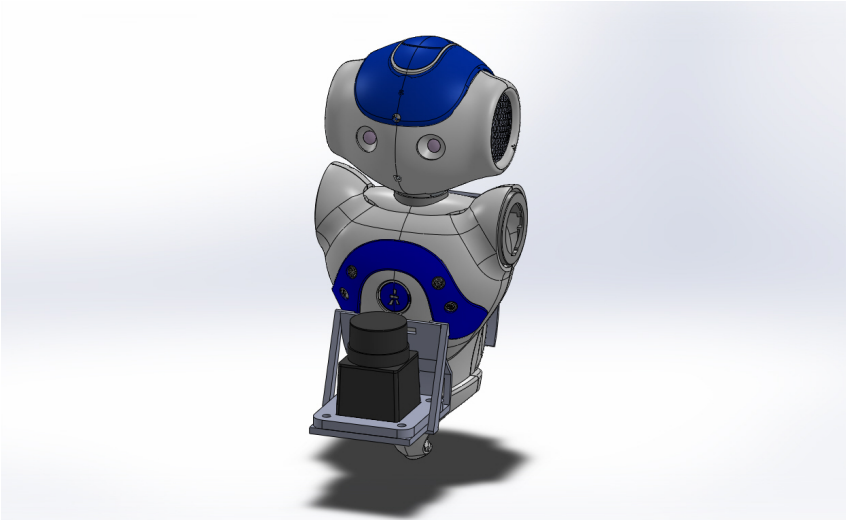

Figure 21: CAD model of Nao with Hokuyo URG-04LX-UG01 mounted to the waist of the robot using a custom mount.

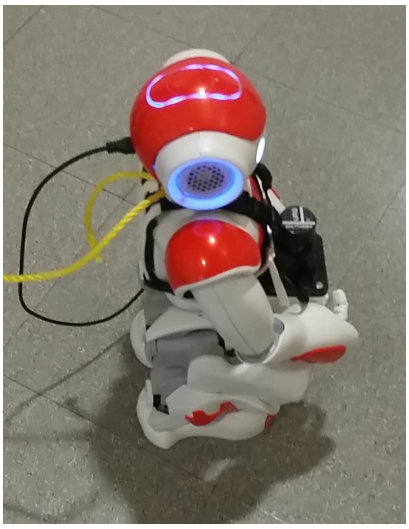

Figure 22: Photograph of the URG mounted to the Nao using the custom Lidar mount.

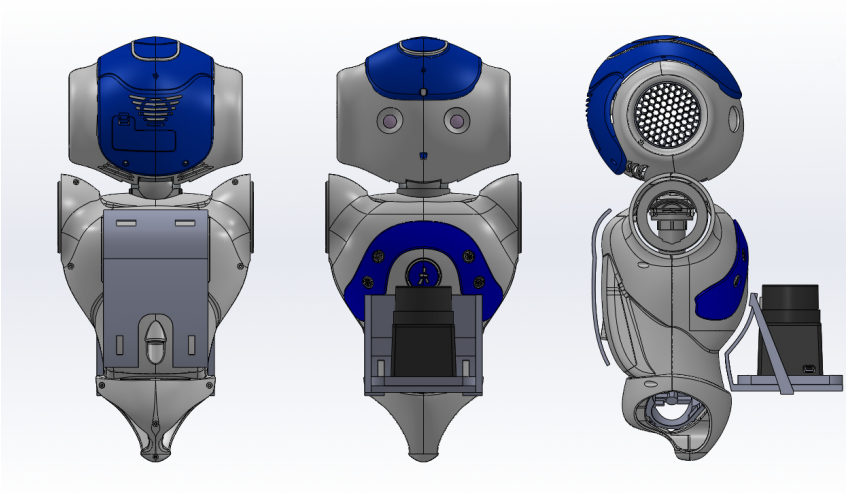

Figure 23: Back, front, and side views of the URG CAD model mounted to the Nao using the custom mount.

2.5.1 Front Subassembly

The front subassembly of the Lidar mount consists of three parts: a front plate, a base plate, and side supports. Figure 24 shows a CAD model of the assembled front subassembly with the URG installed.

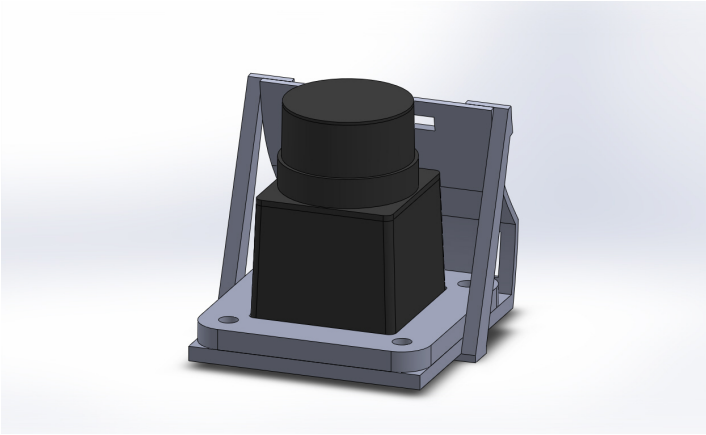

Figure 24: CAD model of the URG attached to the front subassembly of the custom Lidar mount.

The front plate interfaces with the surface of the robot and provides the holes for the straps. It also has a flat platform that acts like a tray which protrudes from the waist of the robot to carry the base plate. The base plate holds the URG and attaches

to the front plate platform. Two side supports act as gussets to reinforce the tray and to discourage bending. Figure 25 shows the back, front, and side views of the assembly. The back view shows the holes for the straps, while the side view shows the front plate curvature that conforms to the shape of the robot. The side supports can also be seen in the side view.

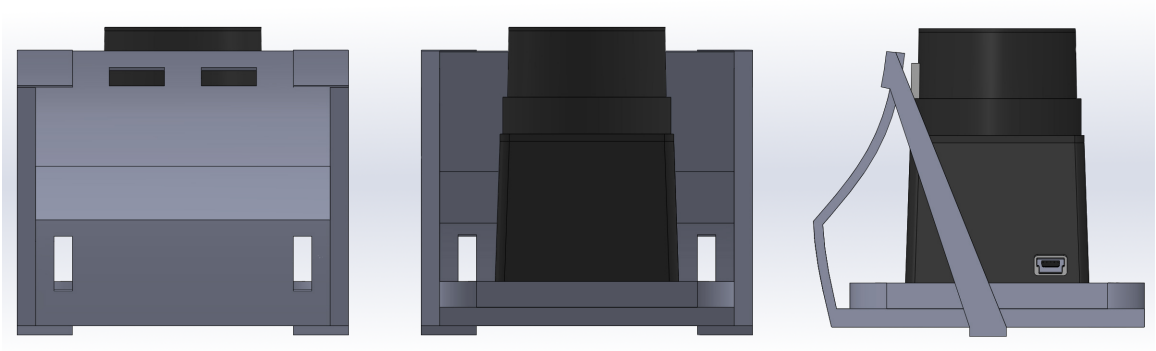

Figure 25: Back, front, and side views of the URG CAD model attached to the front subassembly of the custom mount.

Front Plate

The front plate has a shape that follows the form of the Nao. It is padded with foam that is glued to the plate and absorbs vibrations. Figure 26 shows a CAD model of the front plate. It has a flat platform or tray that projects perpendicular from the robot to hold the base plate. Two mounting holes on the tray secure the base plate to the tray using bolts. The front plate is secured to the robot via four rectangular holes which receive velcro straps that go to the back plate. The top two holes are for straps that route over the shoulder of the robot while the lower two attach waist straps. These straps are tightened to hold the Lidar mount to the robot.

Base Plate

The base plate is a flat piece which acts as an interface between the URG and front plate. The URG is attached to the base plate, which in turn is attached to the tray of the front plate. The URG is attached to this intermediate part rather than directly to the front plate to facilitate the mechanical interoperability of different sensors to the front plate without needing to produce different front plates for different sensors. Instead, different base plates are made for different sensors. For example, the lower cost RPLidar [19] or the longer range VLP-16 Puck 3D Lidar [31] are alternative Lidars

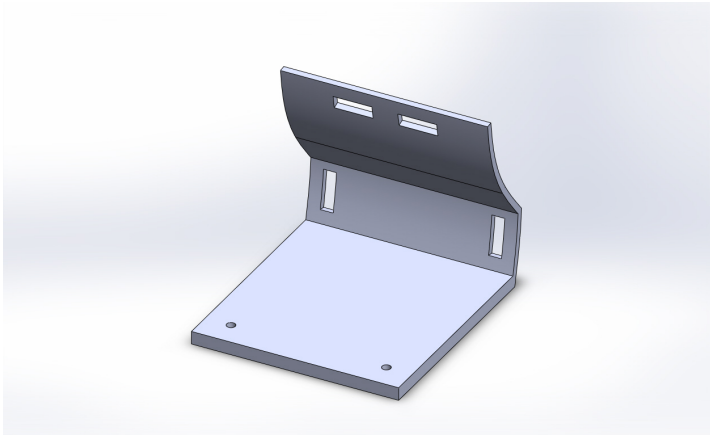

Figure 26: CAD model of the front plate of the custom Lidar mount. The base plate and side supports attach to this plate.

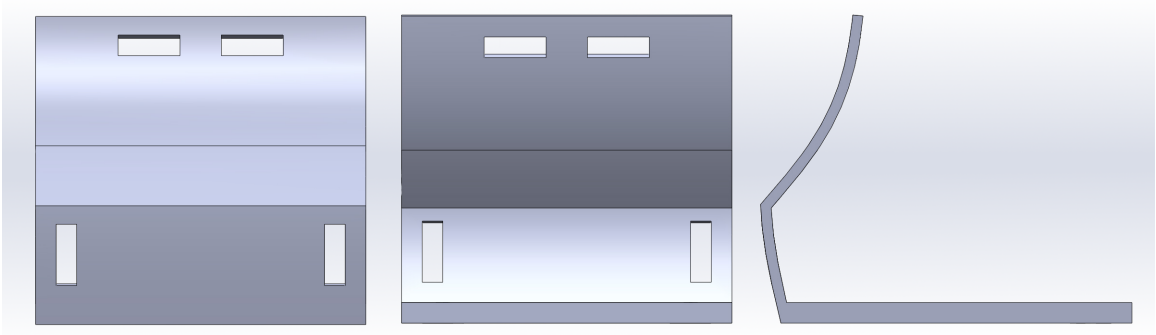

Figure 27: Back, front, and side views of the CAD model of the front plate.

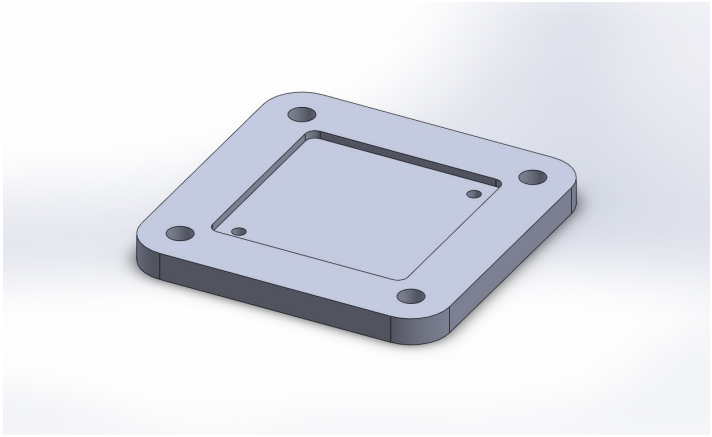

Figure 28: CAD model of the base plate of the custom Lidar mount. The base plate is part of the front subassembly. The URG attaches to this plate.

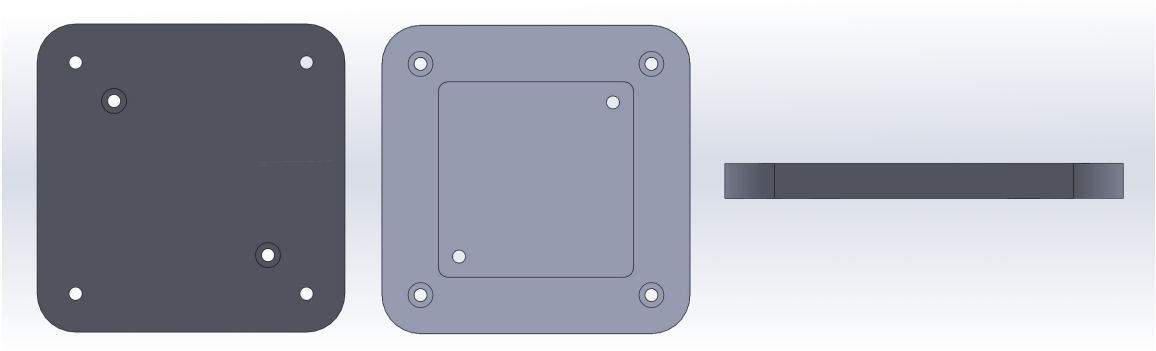

Figure 29: Back, front, and side views of the CAD model of the base plate.

that could be mounted to the Nao for experimentation but have a different mounting configuration than the URG. In this way, new base plates can be manufactured rather than front plates, allowing for a more modular design.

Figure 28 shows six mounting holes and a recessed area. Only two of the outer holes are used at a time and receive bolts that attach it to the front plate tray. The recessed area receives the URG and secures it to the plate using two bolts via the inner holes.

Side Supports

The side supports act as gussets to reduce the angular deflection that can occur with the cantilevered front plate tray. Each support has one flap that interfaces with the underside of the tray and another that sits against the Nao facing side of the front plate. The flaps are then bonded to the front plate using heat. They are positioned

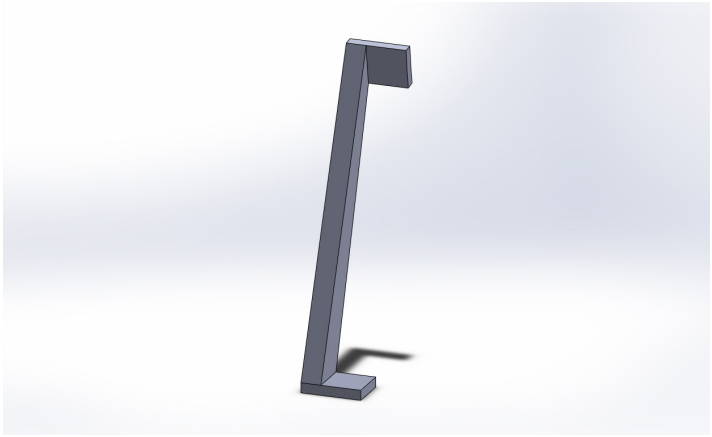

Figure 30: Figure showing a CAD model of the left side support of the custom Lidar mount. This support is bonded to the front plate to add rigidity to the front plate. The right side support is a mirror image of this part.

such that as the weight of the URG pushes down on the tray, the front-plate-to-support-flap interface will be under compression, and the bonded joint is not as strained. This reduces the possibility of the bonded joint being a failure mode.

Figure 31: Figure showing back, front and side views of the CAD model of the left side support.

2.5.2 Back Plate

The back plate is used to receive the straps from the front and provide another surface to squeeze the assembly onto the robot. This ensures a large surface area to transfer forces into the robot so no one area is stressed. It is also lined with foam padding to better conform to the shape of the Nao. Figure 32 shows the plate with the holes for the straps. The side view seen in Figure 33 shows how the shape conforms to the shape of the Nao. There is a large cutout in the bottom of the plate to allow the Nao to be plugged into external power while still wearing the assembly.

Figure 32: Figure showing a CAD model of the back plate of the custom Lidar mount.

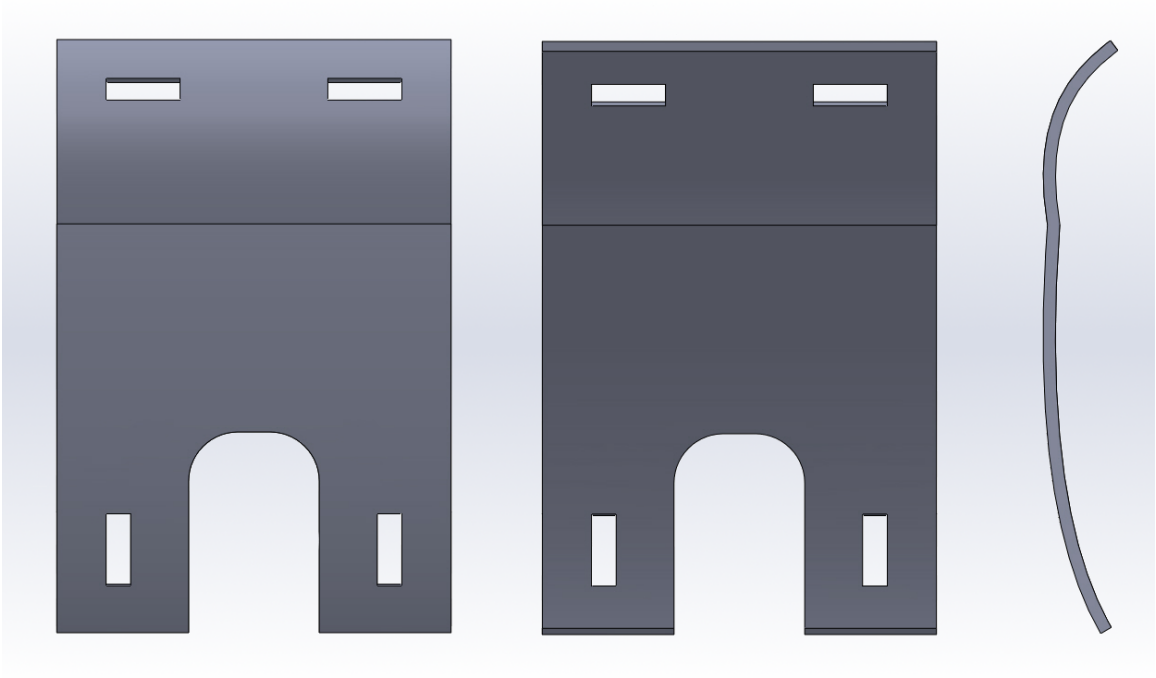

Figure 33: Figure showing back, front, and side views of the CAD model of the back plate of the custom Lidar mount.

CHAPTER III

Path Planning and Navigation

Navigation is the task of generating a series of commands that allow a robot to transit from its current pose to some goal pose. This task encompasses a number of subtasks such as path planning, environment sensing, and obstacle avoidance. As a robot transits through space from its current to goal pose, it necessarily visits a series of intermediate poses the choice of which depends on its kinematic constraints and the constraints of the environment it transits through. Collectively, this set of poses as a time sequence is known as a path. The spacial region in which the robot operates is known as the task space and all kinematically feasible configurations is known as the configuration space. Typically, the environment a robot traverses is not free space but also contains obstacles which constrain the paths which can be taken to the goal. While in some controlled environments the location of obstacles can be known *a priori* and supplied to the robot in the form of a map, often the robot will have the task of creating a map which describes the location of these obstacles. In some situations, "global" sensors can view the entire environment within which the robot will operate and can capture all of the constraints while in others (especially in the case of mobile robots) only "local" sensors mounted to the robot are available to detect environmental limitations to the robot's path as they are encountered. Even more challenging is the case where the environmental obstacles are not static but are moving adding a time varying element to the obstacle avoidance problem.

Algorithms that solve the navigation problem are sometimes described according to four major categories (though there are many other categories with which these algorithms could be described with). These categories are not mutually exclusive, and algorithms are commonly attributed with multiple classifications. The first two, known as global or local, speak to the amount of spatial data or time horizon considered when planning a solution. If only a short time horizon or occlusions in the immediate spatial region are accounted for, then the algorithm is considered local, otherwise it is thought of

as global. The other two deal with how the spaces or paths are represented, continuously or discretely. If the algorithm breaks up these spaces into finite resolution pieces, then it is discrete. Continuous algorithms choose some sort of parameterization to the spatial or environmental constraints using a model. Often algorithms are some mixture of all of these four categories.

The algorithm used for navigation in this thesis is the GODZILA navigation algorithm, which can be categorized as a local continuous algorithm based on the potential fields navigation strategy. While in many environments GODZILA solves the navigation problem, it can also be used in conjunction with global algorithms that can plan way points to the goal without having to consider the detailed local environment.

In the design on the obstacle avoidance algorithm for a robot, it is important to consider how different local sensors can affect the choice of navigation algorithm. The Nao humanoid platform used in this thesis is equipped with two sonar sensors for obstacle detection. Details on these are given in Chapter II. These sensors have a broad angular range and cannot provide information about where within this cone obstacles are located. This enables the robot to avoid colliding with objects directly in front of it but might cause the obstacle avoidance algorithm to preclude the discovery of possible paths (e.g., narrow corridors between obstacles). Conversely, scanning laser rangefinders (such as the one mounted to the Nao and described in Chapter II) have a very high degree of angular resolution and provide hundreds of range measurements. This allows for a more accurate description of environmental occlusions allowing more paths to be considered for traversal.

The notion of what objects in the environment occlude a path and which do not can be viewed as a function of the gait (or mode of locomotion) used by the robot. If the robot is restricted to a plane, such as the case in many ground vehicles, then an object of similar size to the vehicle could prevent traversal through that position. If the robot can also fly then that object may not present an impediment to proceeding towards the goal. While still not transiting through that exact position in 3D space, the 2D projection of the path onto the plane will appear to have moved through the obstacle. This can be though of in terms of a transformation of the obstacle set according to the different motion modalities, making a 2D planner still applicable to the problem. In Chapter IV, a crawling gait for the Nao robot is considered which allows it to go under occlusions that it would otherwise need to plan around.

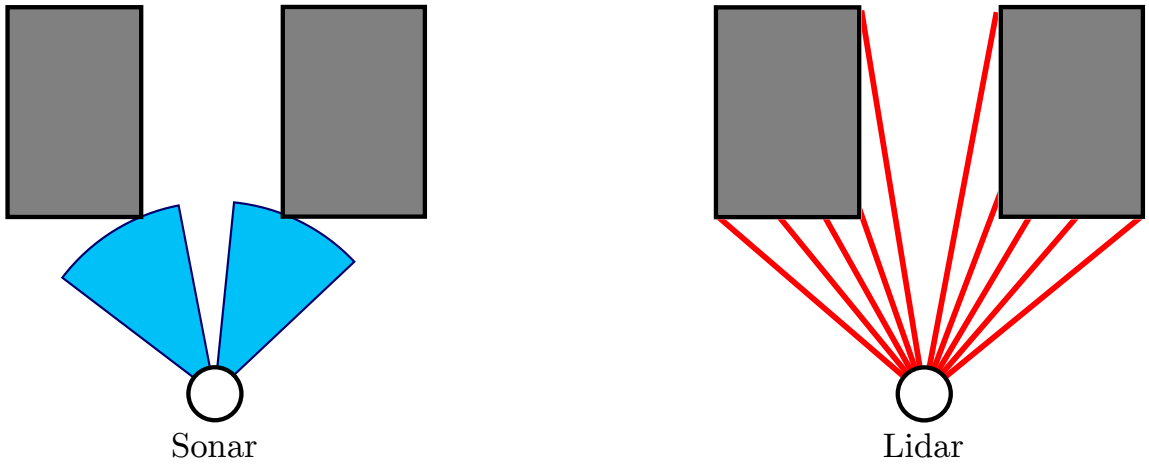

Figure 34: Sonar sensors are typically preferred when needing to detect the presence of an obstacle without needing precise position information. The location of a chair leg in the path of a robot would be detected by sonar while not knowing its position in the sonar cone. Laser rangefinders have very narrow beam width, typically requiring many of them to be effective at describing an environment. If many measurements are available at high angular resolution, then they can be very effective at describing the width of narrow apertures.

3.1 Algorithm Classifications

Here some of the different notions common to navigation algorithms are explored. These classifications are sometimes useful for the navigation engineer to think about as tools available to solve the navigation problem. While additional classifications and subclassifications of algorithms can be considered, the following description is intended as a brief overview of the essential concepts.

3.1.1 Global

One approach to the navigation problem is to consider the entire task and configuration space of a problem when generating a solution. For example, the task space of a robot arm might be the pose of the end effector and all of the occlusions within that space. The configuration space would be the set of all valid joint angles that the arm can attain. Using this, an algorithm can compute a time sequence of poses or movement commands that the robot should execute in order to bring it from its current pose to the final pose. While this approach is ideal in the sense of generating a solution to the original problem, it has several practical disadvantages. One requirement

Figure 35: In global approaches, an entire path is planned to the goal before commanding the robot. In local schemes, the robot is given commands calculated as a function of its position and the local environment and a complete path to the goal is typically precomputed.

of the algorithm is having a complete description of the task space. In many cases global knowledge of occlusions while planning is not available meaning that when new obstructions are observed, the algorithm needs to replan the path. The other commonly encountered problem with these algorithms is when the magnitude of the space to be planned through is so large that computing a solution cannot be done in real-time.

3.1.2 Local

A different approach to the navigation problem is generating commands based only on the current information available or a short history about the environment. In the robot arm example, there might be a sensor such as a camera that can detect objects that are obstructing the end effector from reaching the goal and instructs the arm to move around it. Such algorithms usually have the advantage of being quick to compute as compared to their global counterparts because they only have to consider a small subset of the task and configuration space. Their main disadvantage is that since they only use such a limited amount of information, they can become trapped in local optima that do not allow the robot to achieve the desired pose.

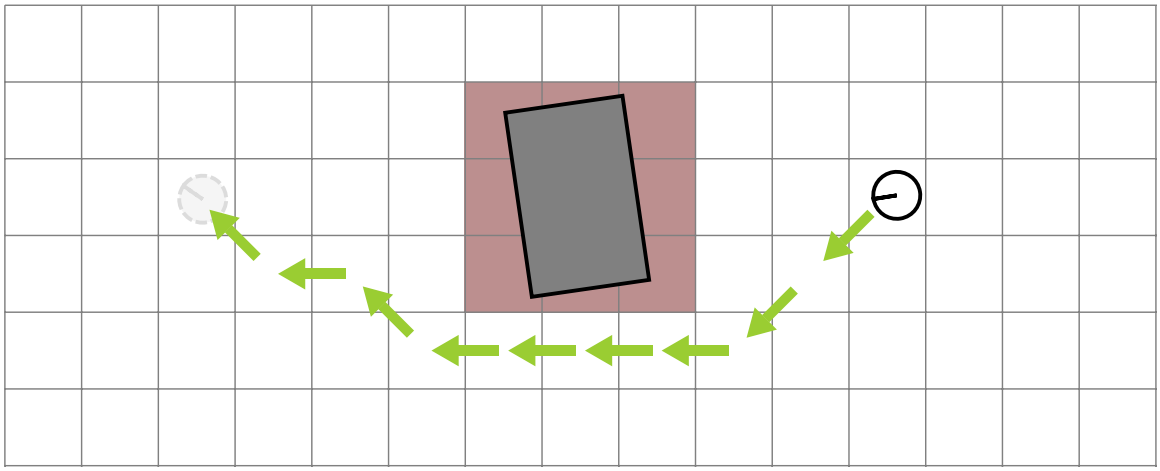

Figure 36: The environment here is being represented as occlusions in a discrete grid of locations. This allows the robot to plan through successive locations to the goal using graph-based techniques.

3.1.3 Discrete

Orthogonal to the ideas of local and global path planning which deal with the scope of the time and space being considered when generating navigation commands is how these spaces are represented. One way to represent the space is to break it up into a set of discrete locations and then plan through that discretized representation. The task space for example could be divided into uniform spatial regions which an algorithm can consider visiting when planning a path. Alternatively, as with the robot arm example, a finite set of movement commands can be considered and iterated through while generating a solution. Graph-based search algorithms such as Dijkstra's algorithm or A* are examples of discrete solvers. An advantage to this is that paths with complex shapes can be generated to accommodate difficult constraints. One point however to consider in this approach is how finely to resolve the task or configuration spaces. Coarser discretization can allow a solution to be computed rapidly but might miss more optimal solutions.

3.1.4 Continuous

Continuous spatial representation avoids the problem of having to choose a discretization resolution. The movement commands or paths are planned in a continuum allowing paths to take on intermediate values not available at a given discrete resolution. Continuous representation instead has a different problem of paths needing to take on

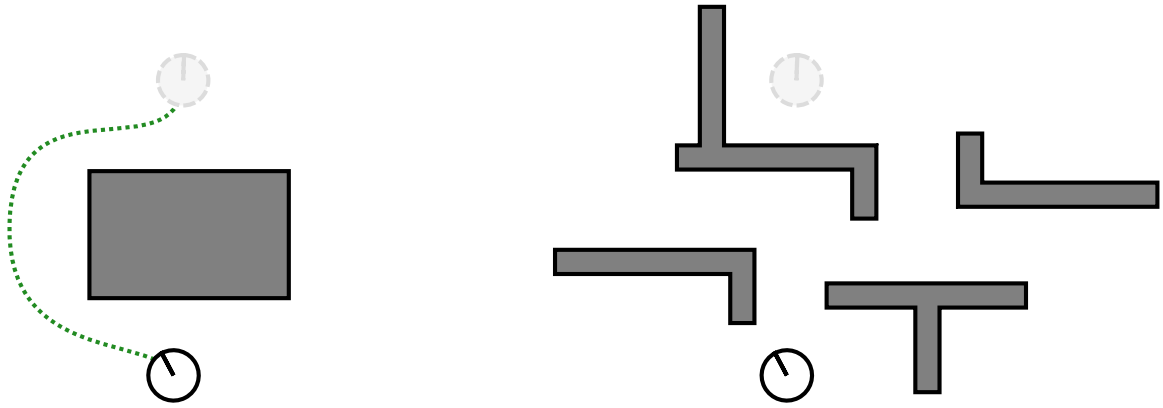

Figure 37: This figure shows two sample environments. On the left, the robot can follow a trajectory that takes a cubic form and reaches the goal location. On the right, a more complicated environment is shown where a single cubic cannot describe an appropriate path.

particular solution forms. A path through the task space might be represented as a cubic spline from the current pose to the goal pose. This path has access to the entire task space but may not be able to construct a path under complicated environments with many obstacles. Often instead of using a single cubic, designers will use a series of cubics, the termination of one being the starting position of the next, until the goal is reached. Alternatively, instead of restricting the path representation to a series of cubics, other trajectory forms can be utilized as a library of representations such as higher-order polynomials, trigonometric functions, etc. One example of such a planner is known as a "maneuver-based" planner.

3.1.5 Composite Approaches

In order to combine the strengths and combat the weaknesses, the above approaches are often combined to form a more robust navigation solution. Global plans can be generated as a series of intermediate goals or way points for local planners to navigate towards. These way points can be planned on a coarse discrete grid that is quick to compute while low dimensional continuous trajectories are plotted between them. Lattice planners are an example of such a composite algorithm.

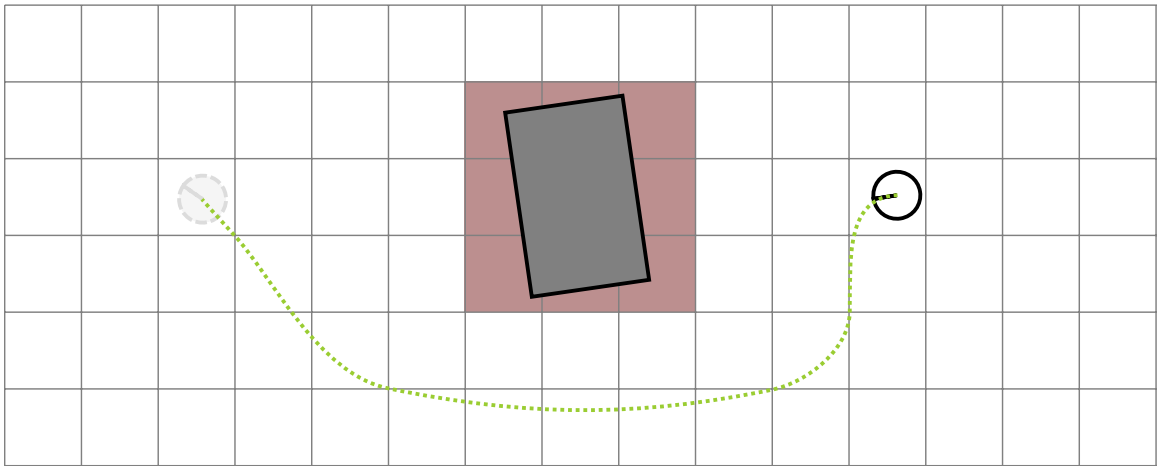

Figure 38: A lattice planner is an example of a composite discrete and continuous path planner. The planner first plans through a discrete grid and then designs trajectories that bring the robot from each point in the planned discrete grid to the next.

3.2 Potential Fields

Potential fields algorithms are continuous local planners. They work on the concept of modeling the robot as a sort of charged particle (like an electron) and obstacles in the environment are modeled as having the same polarity of charge as the robot thereby producing a repulsive field pushing the robot away from them. The goal location in is modeled as having an opposite charge to the robot and thus produces an attractive force, pulling the robot towards it. As the robot moves through the environment, it detects objects and generates motion commands based on its range and direction.

Given a vector x_r which describes the position of the robot in the task space and the positions of obstacles m_i in the set of all obstacles M, a force vector f_i can be generated for each obstacle that describes the magnitude and direction of the repulsive force applied to the robot by the obstacle. The force direction is in the opposite direction of the relative bearing of the obstacle to the robot with magnitude being inversely proportional to the distance of the obstacle from the robot. Equation (3.2) shows an example of force vector generation.

$$\|f_i\| = \frac{-c_i}{\|m_i - x_r\|^\alpha} \tag{3.1}$$

$$\angle f_i = \angle(m_i - x_r) \tag{3.2}$$

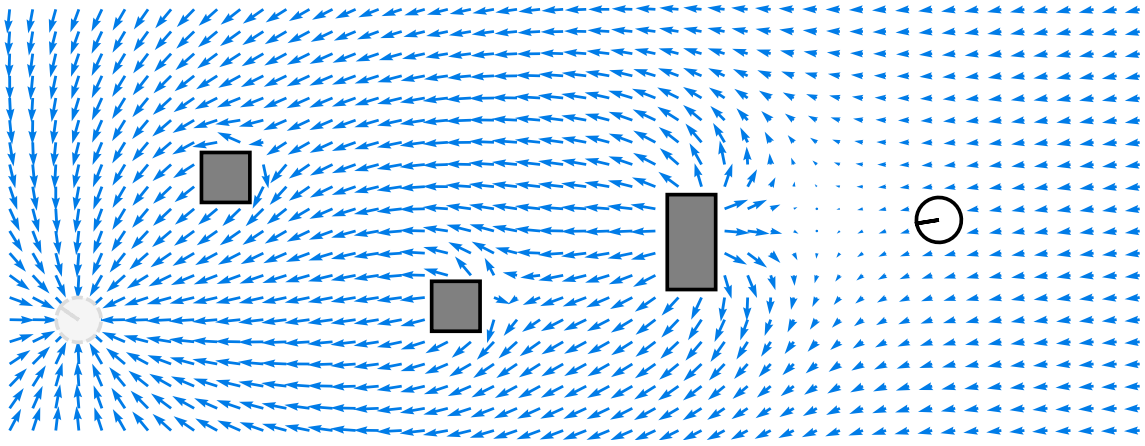

Figure 39: Visualization of potential fields idea. Objects in the environment emit a field that repels the robot and the goal emits an attractive field. This can be visualized as a 2D or 3D vector field.

$m_i - x_r$ is the location of the obstacle relative to the robot and c_i is a positive scalar used to tune force magnitude. In some implementations, c_i can be a function of the relative bearing, for example, decreasing the repulsion effect if the obstacle is not in the direction of the goal. The parameter α in general can be any positive scalar but is often picked to be 2.

For the goal seeking behavior, the equations are similar, with the direction of the force being in the direction of the goal, rather than away from it.

$$\|f_g\| = \frac{c_g}{\|x_g - x_r\|^\gamma} \tag{3.3}$$

$$\angle f_g = \angle(x_g - x_r) \tag{3.4}$$

$x_g - x_r$ in equation (3.4) is the relative location of the goal with respect to the robot and c_g is some positive tuning scalar. γ is some positive scalar, typically 2.

The sum of these forces is used to produce a force vector f_r used to command the robot away from obstacles and towards the goal location.

$$f_r = f_g + \sum_{i=1}^{\|M\|} f_i \tag{3.5}$$

where $\|M\|$ is the number of obstacles (modeled as a discrete set).

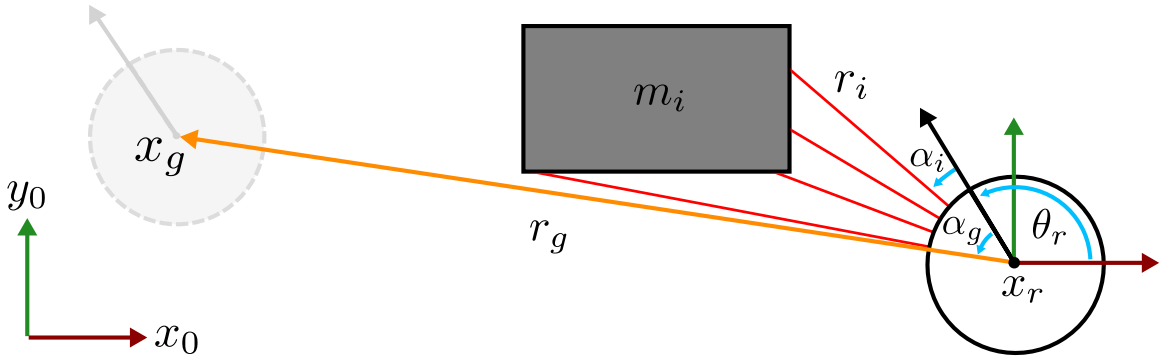

Figure 40: Illustration of various variables utilized in GODZILA. Range and bearing of the goal relative to the robot are required as well as being able to take range measurements to occlusions and knowing the relative bearings of these measurements.

3.3 GODZILA

GODZILA (Game-theoretic Optimal Deformable Zone with Inertia and Local Approach) [26] is a local continuous navigation algorithm based on the potential fields idea. In contrast to some potential fields formulations which work on the range and bearing of objects in the environment, GODZILA uses the sensor information about occlusion locations directly in order to formulate navigation commands. It is a memoryless algorithm and does not attempt to build a map of the environment making it very lightweight in terms of both computation and memory. It can be implemented on a variety of vehicles and is suited for the cases where a low computational power microcontroller is utilized. It has three components which are an optimization cost, a straight line planner, and a stochastic local minima escape strategy.

Figure 40 illustrates the key variables in the formulation.

Taking a 2D example of the application of the GODZILA algorithm, we have the following terms:

x_0 is the x-axis of the inertial frame.

y_0 is the y-axis of the inertial frame.

x_r is the (x,y) location of the robot in the inertial frame.

α'_r is the new heading the robot should be commanded to in the robot frame.

x_g is the (x,y) location of the goal in the inertial frame.

r_g is the range to the goal relative to the position of the robot.

α_g is the bearing to the goal location relative to the orientation of the robot.

z_i is the i^{th} sensor measurement made by the robot which includes range and bearing to an occlusion.

r_i is the i^{th} range measurement to an occlusion relative to the position of the robot.

α_i is the bearing angle to the i^{th} sensor measurement.

In this formulation, it is assumed that there is some mechanism to allow the robot to get an estimate of the relative range and bearing to the goal location r_g and α_g and that the robot has some sensors that can provide range and bearing to occlusions in the environment.

3.3.1 Optimization Formulation

The formulation of the GODZILA algorithm is in terms of an optimization cost. The goal of the algorithm is to bring the robot towards the goal while preventing it from colliding with obstacles. With this intuition the cost function has three components:

1. $J_1(\alpha'_r)$ which rewards goal seeking by penalizing directions other than those towards the goal.
2. $J_2(\alpha'_r)$ which penalizes directions towards occlusions.
3. $J_3(\alpha'_r)$ which dampens oscillations by penalizing changes in heading.

The sum of these terms produces the cost function to be minimized:

$$\min_{\alpha'_r} \sum_{i=1}^{3} J_i(\alpha'_r) \tag{3.6}$$

where α'_r is the heading direction the robot should travel in order to minimize the cost function. $J_1(\alpha'_r)$ takes on the form of:

$$J_1(\alpha'_r) = g_{11}(r_g) f_{11}(\|\alpha'_r - \alpha_g\|). \tag{3.7}$$

g_{11} is a class-$\mathcal{L}$ function meaning that it is continuous and monotonically non-increasing which maps $[0, \infty) \mapsto [0, \infty)$ and f_{11} is a class-$\mathcal{K}$ function meaning it is

monotonically non-decreasing which maps $[0, \infty) \mapsto [0, \infty)$. The intuition here is that f_{11} produces a higher cost in proportion to how much the direction command α'_r differs from the goal direction. g_{11} conversely reduces the cost if the range to the goal is large. The idea being that the farther from the goal the robot is, then being oriented towards the goal becomes less of a priority.

$J_2(\alpha'_r)$ takes on the form:

$$J_2(\alpha'_r) = J_{2_{\mathcal{I}_1}}(\alpha'_r) - J_{2_{\mathcal{I}_2}}(\alpha'_r) \tag{3.8}$$

$$J_{2_{\mathcal{I}_1}}(\alpha'_r) = \sum_{i \in \mathcal{I}_1} g_{21}(r_i) \Big[g_{22}(\|\alpha_g - \alpha_i\|) + g_{23}(\|\alpha_i\|) \Big] g_{24}(\|\alpha'_r - \alpha_i\|) \tag{3.9}$$

$$J_{2_{\mathcal{I}_2}}(\alpha'_r) = \sum_{i \in \mathcal{I}_2} f_{21}(r_i) g_{25}(\|\alpha_g - \alpha_i\|) g_{26}(\|\alpha'_r - \alpha_i\|) \tag{3.10}$$

Equation (3.8) has two components to it, separated by the membership of the sensor measurements z_i to one of two sets. The first is the set of all range measurements r_i whose value falls below some positive scalar r_c are put into a set $\mathcal{I}_1$. The second set is the remaining range measurements, which are necessarily greater than r_c, placed into set $\mathcal{I}_2$. This threshold distance r_c is picked such that if the occlusion is farther away than it, the robot does not need to be concerned with avoiding it. Equation (3.9) is concerned with the occlusions that need to be avoided, and has four class-$\mathcal{L}$ functions $g_{21}, g_{22}, g_{23}, g_{24}$ associated with it. The first three g_{21}, g_{22}, g_{23} act as gains that the control variable α'_r in function g_{24} must attenuate. g_{21} says that if the range r_i to the occlusion is small, then the cost of orienting in that direction is high. g_{22} amplifies this effect if the direction to the occlusion α_i is in the same direction of the goal α_g. g_{23} also amplifies g_{21} if the direction of the occlusion is similar to the heading direction of the robot. Equation (3.10) deals with the case where the occlusions can be approached. It has one class-$\mathcal{K}$ function f_{21} and two class-$\mathcal{L}$ functions g_{25} and g_{26}. f_{21} and g_{25} act as gains on the controlled function g_{26}. f_{21} promotes moving in the direction of ranges that are large and g_{25} promotes moving in the direction of occlusions that are in a similar direction to the goal direction.

The final term $J_3(\alpha'_r)$ is a class-$\mathcal{K}$ function that penalizes changes in direction.

$$J_3(\alpha'_r) = f_{31}(\|\alpha'_r\|) \tag{3.11}$$

This term is present to prevent high frequency oscillations and is analogous to giving the robot a physical inertia. All of the functions $f_{11}, f_{21}, f_{31}, g_{11}, g_{21}, g_{22}, g_{23}, g_{24}, g_{25}, g_{26}$

in the optimization are tunable by the designer but it was noted in [26] that in the case where the controlled functions $f_{11}, g_{24}, g_{26}, f_{31}$ are chosen to be quadratic, the optimization problem can be solved in closed form producing the solution seen in equation (3.12).

$$\alpha'_r = \frac{\alpha}{\|\alpha\|} \tag{3.12}$$

$$\alpha = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \tag{3.13}$$

The four terms in equation (3.13) correspond to terms arising from the four optimization terms $J_1(\alpha'_r), J_{2_{\mathcal{I}_1}}(\alpha'_r), J_{2_{\mathcal{I}_2}}(\alpha'_r), J_3(\alpha'_r)$. They are expanded in equations (3.14)-(3.17).

$$\alpha_1 = \overline{f}_{11} g_{11}(r_g)\alpha_g \tag{3.14}$$

$$\alpha_2 = -\overline{g}_{24} \sum_{i \in \mathcal{I}_1} g_{21}(r_i)\Big[g_{22}(\|\alpha_g - \alpha_i\|) + g_{23}(\|\alpha_i\|)\Big]\alpha_i \tag{3.15}$$

$$\alpha_3 = \overline{g}_{26} \sum_{i \in \mathcal{I}_2} f_{21}(r_i) g_{25}(\|\alpha_g - \alpha_i\|)\alpha_i \tag{3.16}$$

$$\alpha_4 = \overline{f}_{31}\alpha_r \tag{3.17}$$

The solutions to the equations reuse functions $f_{21}, g_{11}, g_{21}, g_{22}, g_{23}, g_{25}$ as they are not functions of α'_r and produce constant versions of $f_{11}, f_{31}, g_{24}, g_{26}$ as $\overline{f}_{11}, \overline{f}_{31}, \overline{g}_{24}, \overline{g}_{26}$ in the direction of their respective directions $\alpha_g, \alpha_i \forall i \in \mathcal{I}_1, \alpha_i \forall i \in \mathcal{I}_2, \alpha_r$ where $\alpha_r = [1, 0]^T$ representing the current heading of the robot in the vehicle frame.

The resultant α'_r is typically commanded as an angular rate $\dot{\alpha}'_r$ according to the angular velocity bandwidth of the vehicle.

Finally, the linear velocity is still to be commanded. A good choice for this function is of the form:

$$v_r = f_v(\min(R)) g_v(\dot{\alpha}'_r) \tag{3.18}$$

where R is the set of all range measurements, f_v is a class-$\mathcal{K}$ function and g_v is a class-$\mathcal{L}$ function. This reduces the speed of the vehicle when it is in close proximity to occlusions or if it is commanded to a high angular rate. Slower speeds near obstacles makes it less likely that the vehicle will collide with them. Slower speeds at times when the robot is commanded to high angular rates helps reduce the distance the robot travels in a previously commanded direction before completing a new turn command.

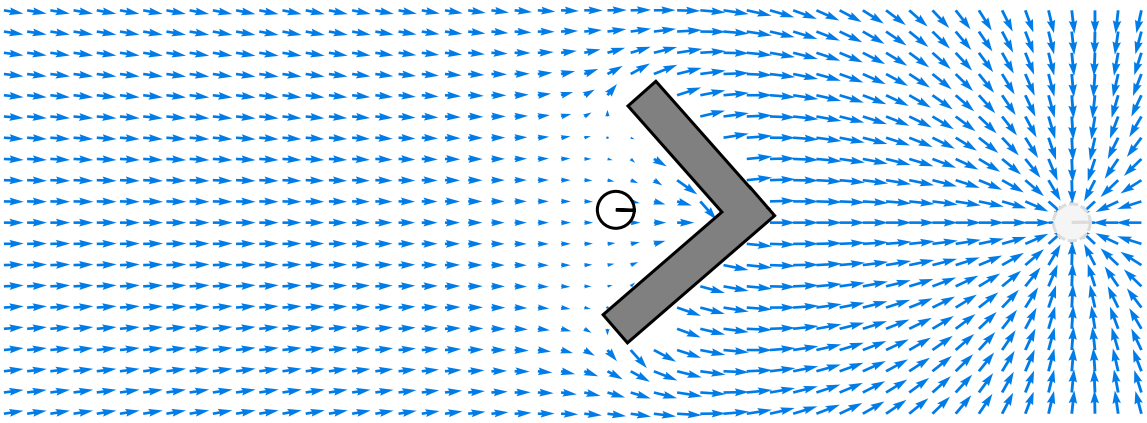

Figure 41: A degenerate case where the attractive force of the goal combines with the repulsive force of the wall producing a local optima where the robot gets trapped.

3.3.2 Straight Line Planner

If at some point during the navigation of the vehicle to the goal the straight line path from the robot to goal becomes unobstructed, it follows that the robot should proceed directly to the goal according to this path. This is sensible, but can bring the robot in close proximity to obstacles. It is therefore desirable to preserve the obstacle avoidance properties of the optimization procedure from Chapter 3.3.1. This procedure is especially useful in situations where the width of the aperture between occlusions to the goal approaches the width of the robot where the obstacle avoidance behavior might deter traversal to the goal. This can be achieved by prescribing intermediate goal locations according to:

$$\hat{x}_g(t) = (1 - \lambda(t - t_0))x_r + \lambda(t - t_0)x_g \tag{3.19}$$

where $\lambda(\tau)$ is a monotonically increasing function that maps $[0, T_f] \mapsto [0, 1]$ with T_f being some amount of time allowed for the vehicle to reach the goal location. The length of time the planner is active is T_f. If the robot has not reached the goal in this time but again has a straight line path to the goal the planner is allowed to restart.

3.3.3 Escape Strategy

While the inertial term provided by(3.11) is intended to reduce high frequency oscillations and backtracking that might trap the robot in small corners, it is not enough to account for degenerate cases such as those shown in Figure 41.

In such scenarios, an effective strategy to escape such traps is to randomly assign an alternate goal position x_{rand} that will allow the robot to escape the local minima and proceed to the goal. The distance to x_{rand} from the robot is typically selected to be smaller than the distance to the nearest occlusion $\min(R)$. After some fixed time, the goal position is set back to x_g. If the trap condition is detected multiple times then the next escape procedure is repeated that many times before being restored to x_g. One possible method to detect a trap condition is to integrate a sliding window of vehicle linear velocities v_r and if those values integrate to a position close to zero then is is likely that the robot has encountered a trap condition. It is shown in [26] that with all of these mechanisms the position of the robot x_r will converge to x_g in finite time with probability 1.

CHAPTER IV

Humanoid Low Profile Crawl Gait

Using the Nao platform, crawling gaits applicable to humanoid robots was explored. Not only is crawling a more stable gaiting strategy than walking, but it gives the robot access to areas of the environment that are inaccessible via walking alone. A low-profile laterally symmetric crawl gait is described, modeling the robot as a closed-chain manipulator with pseudo-static dynamics. This gait was parameterized on three joint variables and optimized using a cubic splines approach via a genetic algorithm.

As was described in Chapter III, the gaiting modality directly affects the navigation strategy. By selecting the appropriate gaiting strategy, the robot can modify its movement to achieve a commanded goal. This modified movement also modifies which environmental objects present as obstacles. Expanding the library of gaits legged robots have access to allows robots to be increasingly more capable and applicable to a wider range of scenarios.

4.1 Humanoid Crawling

Unlike walking and running which enjoy precise definitions, crawling seems to only have a subjective notion. [8] asserts that crawling is a statically stable walk. This definition is problematic as bipedal gaits can be statically stable but would not be classified as crawling. In addition to this, soldiers performing the high army crawl [16] can be seen to perform this motion very quickly which introduces a dynamic component to the crawl. The standard or baby crawl is described as using one's hands and knees to produce forward motion. In contrast to this, crawls such as the leopard, tiger, bear, and crab use hands and feet to produce forward motion and the low army crawl uses the hands to drag and one leg to push the body across the ground. Such diversity in crawling motion makes it difficult to differentiate a crawl from a statically stable quadrupedal gait that uses something other than the end effectors to interact with the environment.

Figure 42: The pane on the left shows the leg configuration of the Nao when the Hip Yaw-Pitch DoF is fully turned in. The right pane show the leg configuration when the Hip Yaw-Pitch is fully turned out. The legs are mechanically linked, making the amount of Hip Yaw-Pitch equal for each leg.

Despite this, the presented gait produces a motion that many would associate with humanoid crawling.

4.1.1 Nao Crawling Limitations

A primary limitation to the gaits that can be produced by the Nao is the limited number of degrees of freedom (DoF) of the platform in contrast to the large number of DoF present in humans. While the Nao has 25 DoF, the human body has 244 [47]. The human arm and leg each have 7 DoF while the Nao's arms and legs each have 5. Nao's hips have one more degree of freedom, called Hip Yaw-Pitch, which turn the legs together at an angle. Figure 42 illustrates this degree of freedom more clearly. The rest of the DoF of the Nao are in the hands and neck. Notably, Nao has no back joint. This prevents the gait designer from prescribing a twisting motion for use in the crawl gait. These limits in motion preclude the execution of any gaits that require lateral twisting or sagittal arching.

4.2 Projected Profile Humanoid Crawl Gait

The crawling gait presented in this thesis is based on viewing the humanoid form as a set of manipulators on the sagittal plane. Figure 43 illustrates this concept. When the robot is laying in the prone position, it necessarily makes contact with the ground. If we then view the robot from the side (looking at the sagittal plane) we can model it as a planar kinematic chain. If the chest and knees are making contact with the ground,

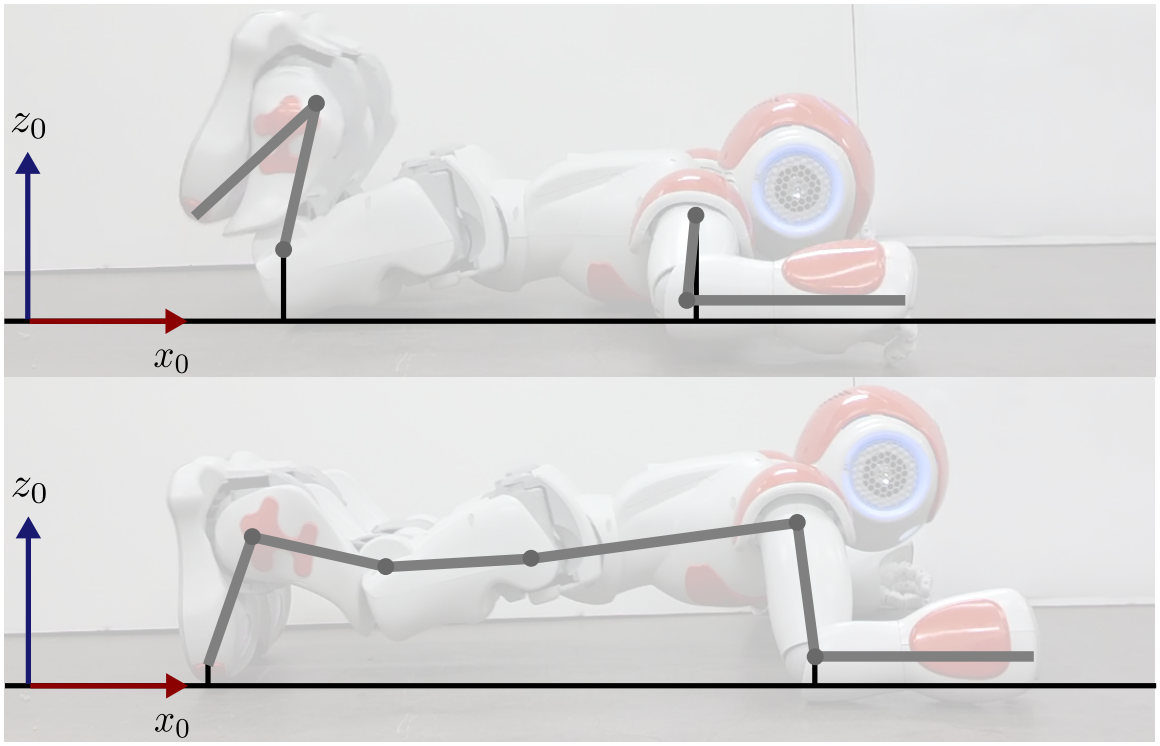

Figure 43: Illustration of Projected Profile concept. Both panes show the saggital view of the Nao with a schematic representation of the kinematics. In the top view, the robot appears as two open chain manipulators. In the bottom view, the robot appears as one closed chain manipulator.

then the arms from the shoulder joint to the hand, and the legs from the knees to the toes are free to move without affecting the rest of the body. This produces two open chain manipulators. In the case of the Nao, each has two degrees of freedom in the sagittal pane, allowing the hands and feet to move independently.

If the elbows and toes are placed on the ground, then the body from the toes to the elbow can be viewed as a closed chain manipulator. This allows the joints to work together to move the center of mass. Kinematically, these two phases share two common configurations. The first configuration is when the elbow is at full extension and the toes touching the knees. We will call this the "extension" configuration. This can be viewed as the robot reaching forward. The second is when the elbow is at full flexion and the ankles at extension. We will call this "compression". This can be viewed as the robot having pulled itself forward. These motions can then be combined to produce a full gait.

Using the Nao robot as an example, Figure 44 shows the full gaiting sequence. The

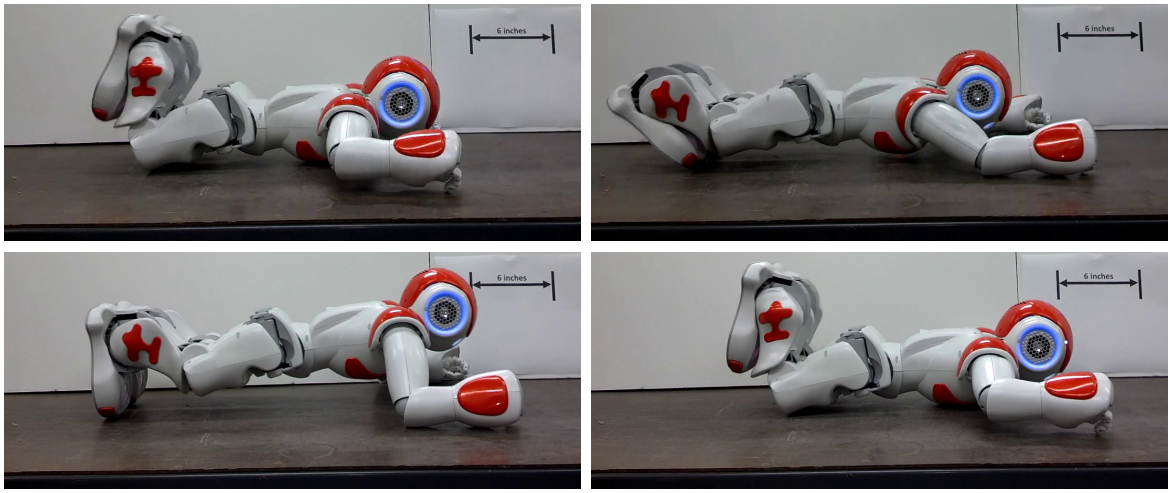

Figure 44: The above sequence shows the motion segments and robot postures in the crawl gait. The upper left pane shows the initial open chain configuration. The upper right pane moves the robot to the "extension" configuration. The lower left shows the "compression" configuration. Finally, the lower right shows the robot, having translated forward, once again in the open chain configuration. A 6-inch marker is shown in the background as a length scale reference.

robot initializes itself in the open chain configuration. From this, it can position its end effectors (the toes and elbows) into the first common configuration "extension". Next, the robot is in the closed chain configuration in which it can transport its center of mass forward until it reaches the "compression" configuration. Finally the robot is again in the open chain configuration and the cycle can start again.

The gait is laterally symmetric. If we actuate the joints at an appropriate rate, dynamic effects from the robot's motion do not become a significant factor. As detailed in Chapter VI, this gait can be performed on the Nao at a speed of 1 ft every 6 to 8 seconds. The wide surface area of the forearm provides a high coefficient of friction against slipping and the small surface area of the toes can act as a point of high pressure which can dig into soft surfaces such as carpets. The gait is statically stable in the sense that the robot's motion can be paused at any point and the robot will maintain that pose. The gait does not depend on the robot sliding along the surface nor does it highly depend on surface friction to pull the robot forward. The gait has a very low profile. The highest point on the robot during the gait (which is the top of the head) is about 8 inches off of the ground. In contrast, during walking, the Nao robot stands 23 inches tall.

Figure 45: Simplified kinematic model of the sagittal projection of the open chain configuration. The manipulator on the left represents the tibia-foot chain. The manipulator on the right represents the humerus-forearm chain. The origin of the knee and shoulder are represented as having a z-axis offset because the knee and chest of the robot have heights that raise their origins.

4.2.1 Open Chain Kinematics

The Projected Profile crawl gait has two kinematic configurations: open chain and closed chain. In the open chain configuration, the robot acts as two independent planar manipulators. Each manipulator has two degrees of freedom as can be seen in Figure45.

With the Nao facing downwards, the feet towards the origin and the head in the positive x direction, the forward kinematics for each manipulator are described by:

$$x = x_0 + l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2) \tag{4.1}$$

$$z = z_0 + l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2) \tag{4.2}$$

where x_0 and z_0 are the superior and posterior offsets with respect to the global frame, respectively. l_1 is the length of the humerus/tibia, l_2 is the length of the forearm/foot, θ_1 is the angle subtended by the shoulder/knee and the x-axis, and θ_2 is the angle subtended by the ankle/elbow and the x-axis of the humerus/tibia.

The solutions to the inverse kinematics problem can be seen as:

$$\theta_2 = \cos^{-1}\left(\frac{(x-x_0)^2 + (z-z_0)^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \tag{4.3}$$

$$\theta_1 = 2\tan^{-1}\left(\frac{-B \pm \sqrt{B^2 - 4AC}}{2A}\right) \tag{4.4}$$

$$A = (x-x_0) + (z-z_0) + l_1 + l_2(\cos(\theta_2) + \sin(\theta_2)) \tag{4.5}$$

$$B = -2(l_1 + l_2(\cos(\theta_2) - \sin(\theta_2)) \tag{4.6}$$

$$C = (x-x_0) + (z-z_0) - l_1 - l_2(\cos(\theta_2) + \sin(\theta_2)) \tag{4.7}$$

This solution derives from the standard inverse kinematics procedure of inverting the forward kinematics. θ_1 must be chosen such that no part of the robot tries to intersect with the floor. In practice, the two argument arc-tangent function $atan2$ is used instead of $\tan^{-1}$.

4.2.2 Closed Chain Kinematics

The closed chain configuration models the toes and elbows of the robot as being fixed to the ground. As with all closed chain kinematics, describing the forward kinematics requires solving an inverse kinematics equation. Modeling the robot in the same orientation as the open chain, with the toe at the origin and neglecting the thickness of the elbow, the forward kinematics of the closed chain are:

$$d_e = \sum_{i=1}^{5} l_i \cos(\sum_{j=1}^{i} \theta_j) \tag{4.8}$$

$$0 = \sum_{i=1}^{5} l_i \sin(\sum_{j=1}^{i} \theta_j) \tag{4.9}$$

$$\alpha = \sum_{i=1}^{5} \theta_i \tag{4.10}$$

where d_e is the prescribed distance of the elbow from the foot and α is the desired angle created by the x-axis of the humerus and the ground. θ_1 through θ_5 are the angles of the following joints with respect to their previous links when projected onto the sagittal plane: toe-to-ground, ankle, knee, hip, shoulder. In Figure 46, the red lines represent the links projected onto the sagittal plane. These equations are the standard planar manipulation equations, treating the foot as the base link with the elbow as the end effector. Equation (4.8) constrains the end effector to be a set distance from the foot and

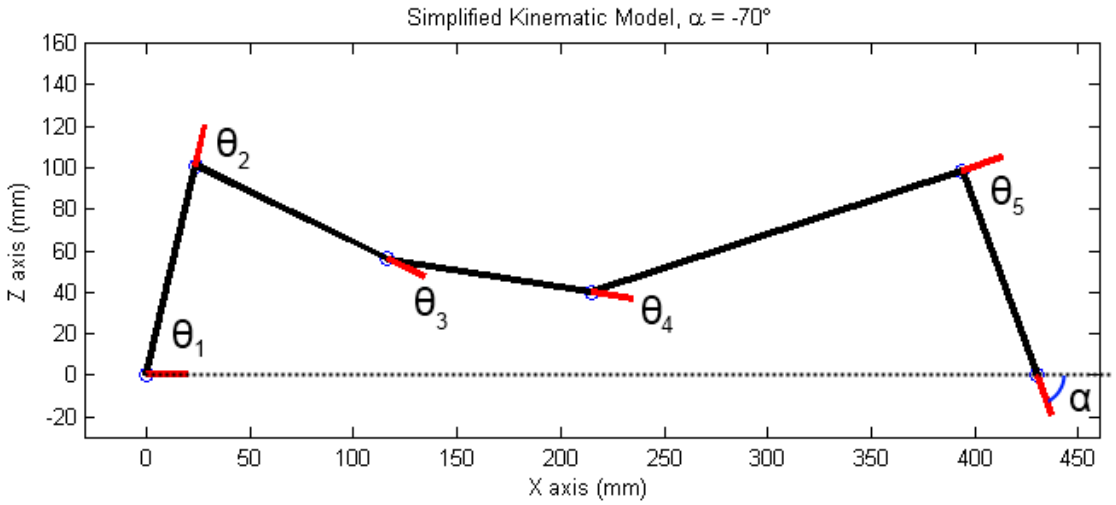

Figure 46: Simplified kinematic model of the sagittal projection of the closed chain configuration.

equation (4.9) constrains the end effector to be on the ground. Using equation (4.10), equations (4.8) and (4.9) can be rewritten as:

$$\sum_{i=1}^{4} l_i \cos(\sum_{j=1}^{i} \theta_j) = d_e - l_5 \cos(\alpha) \tag{4.11}$$

$$\sum_{i=1}^{4} l_i \sin(\sum_{j=1}^{i} \theta_j) = -l_5 \sin(\alpha). \tag{4.12}$$

If θ_3, θ_4, and α are prescribed angles, then equations (4.11) and (4.12) are two equations in the two unknowns θ_1 and θ_2. θ_3 and θ_4 can be constant or time-varying as the resultant configuration is a function of these two "free" variables. Taking the square of each of the equations (4.11) and (4.12), an equation in the single variable θ_2 is obtained as:

$$2l_1K_1\cos(\theta_2) + 2l_1K_2\sin(\theta_2) = [d_e - l_5\cos(\alpha)]^2 + [l_5\sin(\alpha)]^2 - l_1^2 - K_1^2 - K_2^2 \tag{4.13}$$

$$K_1 = l_2 + l_3\cos(\theta_3) + l_4\cos(\theta_3 + \theta_4) \tag{4.14}$$

$$K_2 = -l_3\sin(\theta_3) - l_4\sin(\theta_3 + \theta_4) \tag{4.15}$$

The solution to equation (4.13) is of similar form to that seen in equation (4.4). In general, there will be two solutions for θ_2, but one of them will cause the robot to collide

with the ground and must be guarded against. With θ_2 solved, equations (4.11) and (4.12) can be used to solve for θ_1:

$$\theta_1 = tan^{-1}\left(\frac{sin(\theta_1)}{cos(\theta_1)}\right) \tag{4.16}$$

$$cos(\theta_1) = \frac{K_3(d_e - l_5\cos(\alpha)) + l_5K_4\sin(\alpha)}{K_3^2 + K_4^2} \tag{4.17}$$

$$sin(\theta_1) = \frac{K_4(d_e - l_5\cos(\alpha)) - l_5K_3\sin(\alpha)}{K_3^2 + K_4^2} \tag{4.18}$$

$$K_3 = l_1 + K_1cos(\theta_2) + K_2sin(\theta_2) \tag{4.19}$$

$$K_4 = K_2cos(\theta_2) - K_1sin(\theta_2) \tag{4.20}$$

Lastly, θ_5 is solved using equation (4.10).

With these angles solved, the entire robot is parameterized on three angles $\theta_3, \theta_4, \alpha$. If θ_3 and θ_4 are fixed, then starting with the elbow at full extension, bringing the elbow to flexion moves the robot forward. This corresponds to α starting with a small negative angle and ending with a large negative angle. For the Nao, α is initialized at approximately $-30°$ and terminates at about $-90°$. The primary intuition about this procedure is that the closed chain is like a parallelogram that is used to shift the mass of the robot. Any robot (humanoid or not) that can be set into this configuration can use this framework in order to gait the robot.

4.2.3 Nao Kinematics

Once the projected profile time sequence of angles has been computed, it needs to be applied to the Nao. Figure 47 illustrates the sagittal view of the robot in the closed chain configuration. When Nao is set to this configuration, the ankle pitch, knee pitch, hip pitch, and shoulder pitch joints of the robot directly correspond to θ_2 through θ_5. θ_1 corresponds to the angle subtended by the robot's foot and the ground. Unlike the first five joint angles, angle α does not have a direct correspondence. The arm joint angles must be derived as a relationship between the angle α and additional arm positioning constraints.

The coordinate frame that all of the calculations will be done with respect to, will be embedded in the saggital plane n_p of the Nao. The x-axis is embedded in the saggital plane and in the direction of travel, parallel to the crawling surface. The z-axis is also embedded in the saggital place, perpendicular to the x-axis and pointing upwards. The y-axis is perpendicular to the saggital plane. Nominally, the axis of the shoulder pitch is

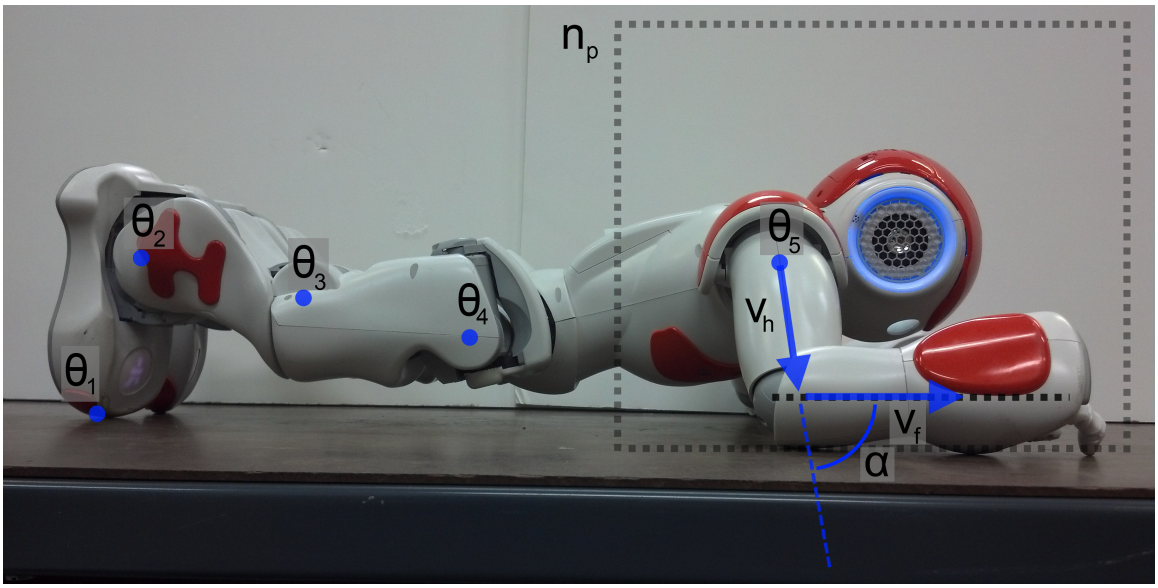

Figure 47: Sagittal view the Nao in the closed chain crawling configuration. The locations of the joints used in the projected profile calculation are shown as blue dots. n_p represents the sagittal plane of the robot. v_h is the humerus vector and v_f is the forearm vector.

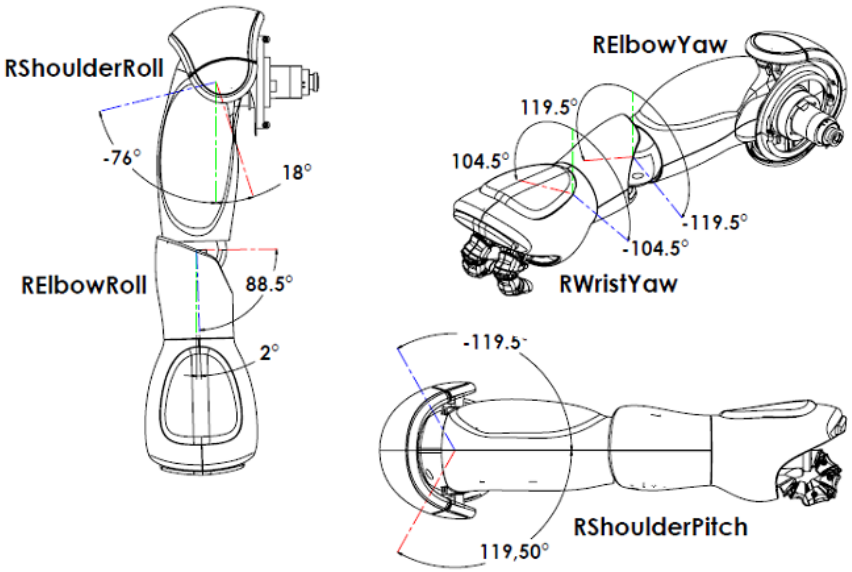

Figure 48: Diagram showing the arm joints of the Nao Humanoid Platform. The "R" preceeding each of the joint names indicates that these are the joints of the right arm. The left arm configuration is a mirror of the right arm configuration. Cite Aldebaran Nao documentation.

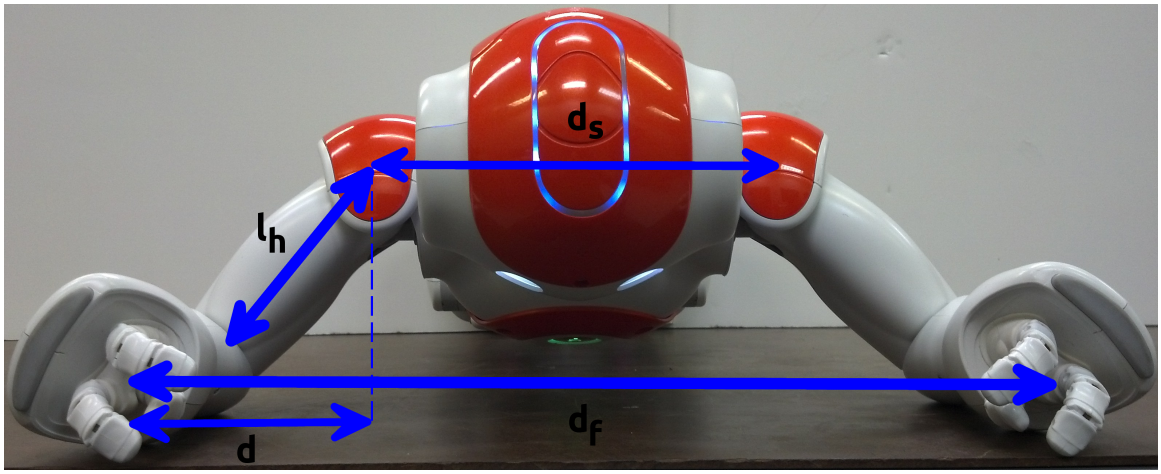

Figure 49: Diagram showing Nao arm positioning. l_h denotes the length of the humerus, d_s denotes shoulder-to-shoulder distance, d_f denotes forearm-to-forearm distance. d is the distance in the y axis that the elbow is from the saggital plane.

coincident with the y-axis. The origin of the coordinate system will be at the shoulder pitch joint.

As shown in Figure 48, the arm of the Nao has 5 degrees of freedom. The kinematic chain of the arm joints proceeds in the following order: shoulder pitch, shoulder roll, elbow yaw, elbow roll, wrist yaw. The angles for these joints will be denoted as θ_{sp}, θ_{sr}, θ_{ey}, θ_{er}, and θ_{wy} respectively. Due to the projected profile method being a saggital projection of the Nao's kinematic configuration, and the shoulder pitch joint being orthogonal to the saggital plane, $\theta_{sp} = \theta_5$ as stated previously. The wrist joint plays no role in the positioning of the arm in this context (as the hand orientation is considered not to be relevant), therefore, θ_{wy} is not a function of the projected profile angle set. The remaining joint angles θ_{sr}, θ_{ey}, and θ_{er} need to be solved for using the projected profile angle α and the following arm positioning constraints.

While the arms could be positioned such that θ_{sr} and θ_{ey} are set to a constant and θ_{er} corresponds to α, this configuration limits the lateral stability of the gait. To increase the lateral stability, the arms are positioned so the forearm-to-forearm distance d_f is greater than the shoulder-to-shoulder distance d_s. This increases the width of the support polygon that the center of mass can be projected onto. A diagram of these distances can be seen in Figure 49.

As the shoulder roll only proceeds the shoulder pitch, its value θ_{sr} can be computed as a function of d_s, d_f and the length of the humerus l_h. A scalar $d = \frac{d_f - d_s}{2}$ is the

distance in the y-axis that the elbow extends from the origin. From the geometry of the arm positioning, it can be seen that θ_{sr} can be computed as:

$$\theta_{sr} = sin^{-1}\left(\frac{d}{l_h}\right) \tag{4.21}$$

Intuitively, the elbow yaw and elbow roll are a function of the vector u_f, which represents the desired orientation of the forearm, the vector v_f, which is the orientation of the forearm when θ_{ey} and θ_{er} are zero, and the two vectors representing the axes of joint rotation about the two elbow joints, denoted v_{ey} and v_{er}. v_f is first rotated about v_{ey} by amount θ_{ey}, then about v_{er} by amount θ_{er}. This should bring v_f to be conincident with u_f. This can be expressed by equation (4.22).

$$u_f = R_{v_{er},\theta_{er}} R_{v_{ey},\theta_{ey}} v_f \tag{4.22}$$

When the joint angles are zero, $v_f = v_{ey}$. The desired orientation of the forearm is to be along the direction of travel to maximize the static friction in that direction. u_f is therefore coincident with the x-axis $u_f = [1, 0, 0]^T$.

As v_{ey} is coincident with the humerus, it can be found by computing the normalized humerus vector. An illustration of this vector can be seen in Figure 50. The elbow yaw axis is then defined by equation (4.23).

$$v_{ey} = [\tilde{l}_h \cos(\alpha), d, \tilde{l}_h \sin(\alpha)]^T / l_h \tag{4.23}$$

$\tilde{l}_h = \sqrt{l_h^2 - d^2}$ is the length of the humerus, projected onto the z-x plane.

By the kinematics of Nao's arm, v_{er} will always lie on the plane perpendicular to the vector v_{ey}. As θ_{ey} can be offset to make v_{er} any vector on that plane, v_{er} is chosen to be:

$$v_{er} = [-\sin(\alpha), 0, \cos(\alpha)]^T \tag{4.24}$$

which is perpendicular to v_{ey} by construction.

To find θ_{ey}, it is necessary to find the vector u_{er} to which v_{er} must be rotated, to allow v_f to be brought to u_f. u_{er} can be found by taking the cross product of v_f and u_f. Then, θ_{ey} is the inverse cosine of the dot product of v_{er} and u_{er} as they are all unit vectors.

$$u_{er} = v_f \times u_f \tag{4.25}$$

$$\theta_{ey} = \cos^{-1}(v_{er} \cdot u_{er}) \tag{4.26}$$

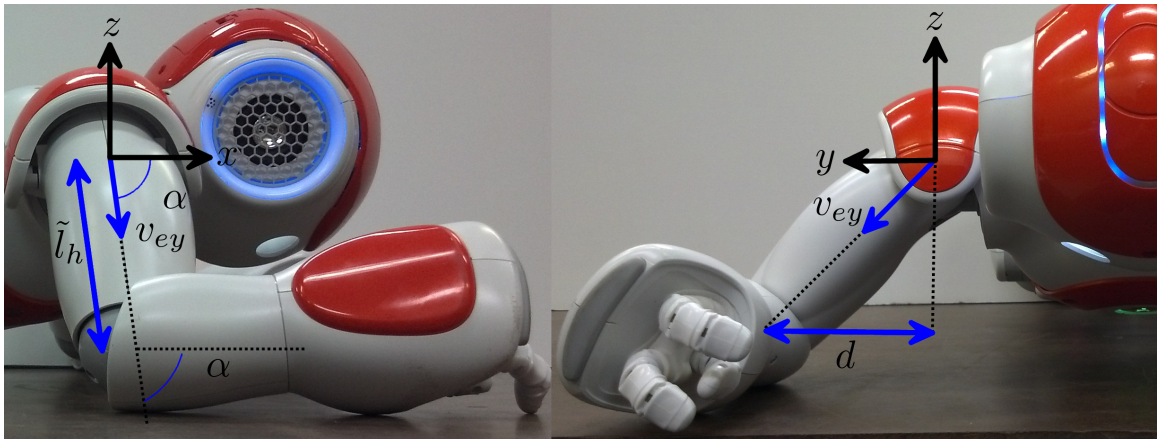

Figure 50: Diagram illustrating the vector v_{ey}. The left pane shows the vector in the z-x plane and its z and x components as functions of $\tilde{l}_h$ and α. α is defined as being the angle subtended by the humerus in the z-x plane and the ground plane, but as the ground plane is parallel to the x-axis in this derivation α can be used to describe v_{ey}. The right pane shows the vector in the z-y plane and its y component as d.

Finally, θ_{er} is the angle between the forearm vector v_f and the desired forearm vector u_f. This can again be found using the inverse cosine and dot product.

$$\theta_{er} = \cos^{-1}(v_f \cdot u_f) \tag{4.27}$$

4.3 Optimization

In the previous section, the Projected Profile crawl gait was parameterized on angle triplet $[\theta_3, \theta_4, \alpha]$. To achieve a crawling gait, θ_3 and θ_4 can be set to be constant and α linearly incremented from an initial angle to a final angle as a function of time. While this configuration successfully gaits the robot, it is heuristic. To improve the approach, the selection of a triplet of cubic splines $[\theta_3(t), \theta_4(t), \alpha(t)]$ is considered as an optimization problem. While many different quantities such as gait speed or the levelness of the back (for transportation of payloads) could be considered as optimization metrics, in this thesis the energy usage was minimized via joint torque measurements. The aim was to increase the amount of time the crawling gait could be performed and reduce the stress on the robot's joints.

4.3.1 Pseudo-static Model

In order to optimize the gait with respect to the joint torques of the robot, a dynamic model of the robot is required. As the Projected Profile crawl gait is conceived as a statically stable gait and performed at slow speeds, the dynamics due to the movement of the robot are not considered to be forces that the motor controllers must counteract. During any part of the gait the robot will not slide if the gaiting direction is orthogonal to gravity and if the robot were to relax its joints it would collapse. Considering this, the resultant joint torques can be seen to be a function of gravity. This pseudo-static model of the robot can then be used to produce a cost metric for the optimization procedure. While conceptually simple, analyzing the projected profile closed chain manipulator to produce the system dynamics is challenging. In lieu of this, the robot was simulated for different values of the angle triplet in the closed chain configuration to generate a table of torques. This table of torques is then interpolated to produce a model of joint torques as a function of the parameterized angles:

$$[\tau_2, \tau_2, \tau_4, \tau_5] = jointTorques(\theta_3, \theta_4, \alpha) \tag{4.28}$$

where $\tau_2, \tau_3, \tau_4, \tau_5$ are the torques of one of the ankles, knees, hips, and shoulders, respectively. In this case, only one of each of the joints is considered because the gait is laterally symmetric. This means the torque of the left joint should be the same of the right joint. τ_1 is not a product of the function as θ_1 is not an actuated joint.

The V-REP simulator by Coppelia Robotics was used to gather joint torque data. It uses the Open Dynamics Engine (ODE) as its dynamics solver and is distributed with a model of the Nao Humanoid Platform. The model of the Nao kinematically corresponds with the Nao V4 and has the same mass values for the links. The Nao V4 is the robot used in this thesis. It has an easy to use API that can interface with C++, Python, or MATLAB.

Figure 51 shows a screen capture of the Nao model in the V-REP simulator. The Nao was configured to be in the closed chain and set to different joints angles. The initial and final values of α were constrained due to the gaiting requirement of the elbow to start at extension and end at flexion. The ranges of θ_3 and θ_4 were defined according to what seemed like plausible knee and hip angles for the gait. Using these limits a discrete set of triplets were defined at a 2.5° resolution for each triplet parameter. These triplets were sent through the kinematics equations to produce the joint commands for the simulated Nao robot. Table 6 lists the parameters that describe the triplets tested. In

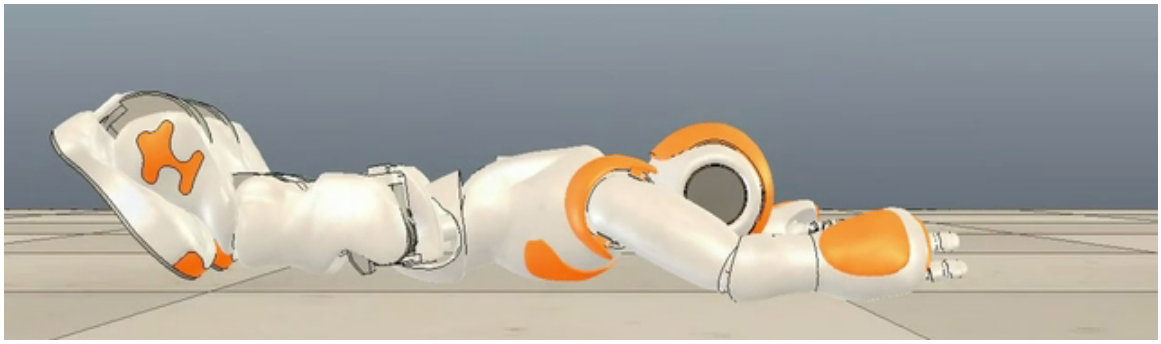

Figure 51: A sample screen capture of the V-REP simulation of the Nao robot in the closed chain configuration. The robot is set to different poses and then the joint torques are read after a short settle time.

Configuration Parameter	**Minimum Angle (degrees)**	**Maximum Angle (degrees)**
θ_3	-5	45
θ_4	15	-30
α	-30	-90

Table 6: Table of initial and final joint angles for each angle in the configuration triplet used to generate the set of angles used to configure the simulated Nao robot. The resultant set had 9,975 configurations with an angular resolution of 2.5°.

total, 9,975 different joint configurations were simulated. To allow for the effect of any dynamics generated by the change in configuration to settle, the torque values of each of the joints was recorded after a period of one second.

4.3.2 Cost Functions

To find the optimal parameters for the angle triplet splines $[\theta_3(t), \theta_4(t), \alpha(t)]$, a cost c_s has to be attributable to a given spline triplet as a function of the spline parameters. The cost function contains terms regarding the joint torques given by the pseudo-static model, as the primary goal of the optimization procedure is to reduce the overall usage of torque (and therefore energy) of the gait. Additionally, a number of indicator functions were introduced to the cost function in order to ensure kinematic constraints were not violated and to deter problems with the algorithm's implementation, which will be

discussed.

The cost function based on joint torques was computed as:

$$c_\tau(t) = \sum_{i=1}^{4} w_i \tau_i^2(t) \tag{4.29}$$

In this notation, $\tau_1, \tau_2, \tau_3, \tau_4$ are the torques of the ankle, knee, hip, and shoulder, respectively. These torques are taken from the pseudo-static model and, as stated, are a function of $\alpha(t), \theta_3(t)$, and $\theta_4(t)$. The weight vector is constant and equal to $w = [1, 1, 1, 5]^T$. w_4 has a higher value in order to reduce the use of Nao's shoulder, as its actuator can produce about 3 to 4 times less torque than each of the leg actuators.

The first set of indicator functions introduced deter violation of kinematic constraints. Equation (4.30) enforces joint limits in θ_3 and θ_4 while equation (4.31) enforces angular limits for α. Equation (4.32) prevents the Nao from being in a kinematic configuration where the hips are making contact with the crawling surface.

$$c_{\theta_i}(t) = \begin{cases} 1 & \text{if } \theta_i(t) > \theta_{i_{max}} \text{ or } \theta_i(t) < \theta_{i_{min}} \\ 0 & \text{otherwise} \end{cases} \tag{4.30}$$

$$c_\alpha(t) = \begin{cases} 1 & \text{if } \alpha(t) > \alpha_{max} \text{ or } \alpha(t) < \alpha_{min} \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

$$c_{z_{hip}}(t) = \begin{cases} 1 & \text{if } z_{hip}(t) < z_{hip_{thresh}} \\ 0 & \text{otherwise} \end{cases} \tag{4.32}$$

The function $z_{hip}(t)$ returns the height of the hips at time t using the the forward kinematics of the closed chain model of the gait. $z_{hip_{thresh}} \neq 0$, and is chosen to be the height of the hip joint when the robot is touching the crawling surface.

The next indicator function exists to deter a problem observed with the gaits being produced. Experimentally, it was seen that the optimizer would produce splines in which the parameter α would, for a large part of the gait, either stay nearly stationary or increase for some time, before decreasing by a large amount. To review, α is initialized at approximately $-30°$ and terminates at about $-90°$. This appeared as the robot lunging forward at the end of the gait. This can most readily intepreted as the robot incurring a large cost for a small time, which is perhaps a local optima. The indicator function shown in equation (4.33) was added to penalize splines in which $\alpha(t)$ was not monotonically decreasing, which allowed the optimizer to find a more efficient gait.

$$c_{\dot{\alpha}}(t) = \begin{cases} 1 & \text{if } \dot{\alpha}(t) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.33}$$

The indicator functions are then summed and multiplied by a large constant C_v so that the cost to violate any of these constraints is prohibitively large.

$$c_I(t) = C_v\Big(c_{\dot{\alpha}}(t) + c_{\alpha}(t) + c_{z_{hip}}(t) + \sum_{3}^{4} c_{\theta_i}(t)\Big) \tag{4.34}$$

The cost of the spline triplet is then given by integrating the sum of cost functions given by equations (4.29) and (4.34), and can be seen in equation (4.35).

$$c_s = \int_0^T c_\tau(t) + c_I(t)\, dt \tag{4.35}$$

where T is equal to one second, as it was seen that this was a reasonable time frame for this phase of the gait to be performed.

4.3.3 Optimization Procedure

The optimization procedure used to optimize the spline triplet $[\theta_3(t), \theta_4(t), \alpha(t)]$ was a basic genetic algorithm implemented using the genetic algorithm functions available in the MATLAB Global Optimization Toolbox.

Briefly, genetic algorithm optimization works by first creating a "population" of possible solutions, in this case the parameters for the spline triplet. Then the "fitness" of each solution is determined, here by evaluating the cost function for the solution. Finally, the solutions that are more fit are "selected" to for the "parent" set. These parents are then "bred" to form a new set of "child" solutions which will then be evaluated again. After some number of "generations", the most fit solution is selected as the optimal solution.

Genetic algorithms are well suited for problems like this as more classical optimization techniques would require the cost function to be analytical, which the joint torque table is not. As a result, because there is so much freedom in the cost function, different functions can be tried and evaluated to find a suitable result. Lastly, because the gait design problem presented in this thesis, was not required to run in real time on the Nao, using such a computationally heavy optimization technique did not impact gait performance. One pitfall to this technique is there is no guarantee that the global optima

will be found for a given cost function and therefore local optima are still possible to be returned.

Using the cost function presented in Section 4.3.2, the genetic algorithm was used to find the 12 triplet parameters (4 parameters for each cubic spline). Details on the results of this optimization can be found in Chapter 6.1.3.

CHAPTER V

Path Planning and Navigation Results

To test the efficacy of the GODZILA algorithm, the Nao humanoid platform was instrumented with a Hokuyo LIDAR and set in three different arenas to navigate to a goal. The Nao was used as the mobile base for the experiment while the Hokuyo LIDAR provided range and bearing information about arena obstacles. The NAOqi API provides functions for tracking a red object using its onboard cameras to estimate the range and bearing to the object. A red cube was designated as the goal location that the robot could track. A secondary camera was mounted to the ceiling of the room in which the arena was set up such that the entire arena and progress of the robot could be recorded. This video was only used to record the results of the experiment and at no time did the robot have access to this data during the experiment. The relative pose estimates from the Nao about the goal location were also recorded for analysis.

Details about the Nao and LIDAR can be found in Chapter II and details about the arena setup, data collection, and algorithm parameters can be found in Section 5.1. Data collected by the robot about the goal pose is presented in Section 5.2 and robot pose data collected by the global camera is presented in Section 5.3.

5.1 Experimental Setup

Using the NAOqi API discussed in Chapter II, the GODZILA algorithm was implemented on the Nao in C++. Different arenas were constructed and the robot was set at some start location and navigated to some radius from the goal location before stopping. Concurrently, the parameters of the algorithm were tuned such that the robot performed well in any of the tested arenas. This section discusses the platform hardware (Section 5.1.1), goal pose estimation (Section 5.1.2), and the tuned GODZILA parameters (Section 5.1.3). Finally, it overviews the experimental arenas (Section 5.1.4) and the global camera setup (Section 5.1.5) used to record each test.

5.1.1 Mobile Platform and LIDAR

The Nao humanoid platform was used as the mobile base for the experiment. The NAOqi API provides a bipedal gaiting function that allows the robot to walk over flat terrain. As the floor of the arena was approximately flat, this gait was applicable to this experiment and was not modified before use. To the robot's chest was attached a Hokuyo LIDAR via a 3D printed mount. The LIDAR had a measurement range between 0.02 and 5.6 meters with an angular resolution of 0.352° and a viewing angle of ±120°. Data from this sensor was used by the robot to detect obstacles in the environment. GODZILA used this data to generate the repulsive force vectors discussed in Chapter III. These repulsive forces allow the robot to avoid colliding with obstacles in the environment. An in depth discussion of the Nao, LIDAR, and LIDAR mount can be found in Chapter II. A shortcoming of this approach was that while the LIDAR and mount were physically compact and well within the load carrying capacity of the Nao platform, the dynamics generated by moving this additional mass during gaiting were not accounted for by the in-built walking algorithm as there was no method available to modify the dynamic model. As a result, the walking gait became marginally stable and at times, unstable. The marginal stability of the platform can be seen in the results presented in Section 5.3. In future work, these additional dynamics will need to be accounted for in order to increase the robustness of the platform, should it be used again for this or a similar experiment.

5.1.2 Goal Pose Estimation

The head of the Nao is equipped with two color cameras. The optical axis of the first camera is nearly collinear with the x-axis of the head, allowing the robot to see in front of it. The optical axis of the second camera is pitched down by approximately 40°, allowing the robot to see objects near it's feet. The NAOqi API provides functions to track a "red ball" using the images from the first camera. These functions actuate the head in an effort to keep the "red ball" in frame at all times. The functions also return the position of the "red ball" relative to the torso of the robot. This can be done because the API assumes the diameter of the ball to be $0.06m$, allowing the apparent diameter of the ball to estimate the actual diameter. With this as a tool, a red object was constructed to indicate to the robot the location of the navigation goal. The object was tracked during navigation and its estimated position was used by GODZILA to generate the attractive force vector discussed in Chapter III. The red object was built

as a cube with sides $0.127m$ long. The cube was mounted on a wooden dowel that was nearly as tall as the Nao which was in turn affixed to a heavy wooden base. The cube was mounted on the dowel to allow the robot to see it more easily as this allows the head to minimize the amount of pitching that must be done to keep the cube in view. The red object was a cube because during testing the "red ball" tracking algorithm did not seem to be affected by the change in geometry and the cube shape was easier to construct. The cube width was twice as wide as the expected diameter to allow the target to be seen by the robot from longer distances. This longer range allowed for the construction of a larger arena as well as more robust tracking at shorter ranges. However, the larger object results in an incorrect range estimate of the tracked object. While this estimation error is rectifiable through appropriate calibration, during these experiments the corrections were not performed. This would mean that the algorithm parameters would be different in a calibrated system and the presented data shows the robot stops roughly twice as far from the goal than as instructed.

5.1.3 GODZILA Parameters

The GODZILA algorithm uses a number of tunable parameters to generate vehicle navigation commands. Chapter 3.3.1 discusses the theoretical basis for these parameters. Each needs to be experimentally tuned in order to achieve optimal performance. They can be roughly divided into three categories which are discussed in this section. The first set is concerned with limits and thresholds governing navigation performance. The second influences the behavior of the robot while turning, while the third influences the forward motion of the robot. While testing the GODZILA implementation, these parameters were tuned according to the observed behavior during the testing runs. Table 7 shows the parameters used by the algorithm which produced the results presented in this chapter.

Navigation Thresholds

Seven parameters limit the navigation performance of the robot to be within certain bounds. Generally, these parameters exist to ensure the safety of the robot and environment but their inclusion affects the tuning of the other parameters as they introduce system non-linearities.

Category	**Parameter Name**	**Value**
Thresholds	Goal Stopping Radius	0.3 m
	Minimum Linear Velocity	-0.4 $\frac{m}{s}$
	Maximum Linear Velocity	0.4 $\frac{m}{s}$
	Minimum Angular Velocity	-0.2 $\frac{rad}{s}$
	Minimum Angular Velocity	0.2 $\frac{rad}{s}$
	Clearance Threshold	0.3 m
	Obstacle Threshold	3.0 m
Turning	Goal Attraction	100
	Obstacle Repulsion	20
	Obstacle Attraction	0
	Obstacle-Goal Bearing Ratio	1
	Vehicle Inertia Gain	15
Forward Motion	Velocity Gain	5
	Obstacle Repulsion	5
	Angular Rate Braking	3

Table 7: Table of GODZILA parameters and the values used during experimentation. Each parameter is divided into categories controlling thresholds, turning, and forward motion. While the thresholds have units, the turning and forward motion parameters consist of gains and ratios, which have no unit.

Goal Stopping Radius This is the maximum distance the robot can be from the goal point before GODZILA considers the robot to have reached the goal position. Practically, the robot will never estimate its range from the goal to be exactly zero, so to prevent the robot from oscillating around the goal pose, this parameter gives a stopping criteria for navigation.

Minimum Linear Velocity This is the minimum linear velocity of the vehicle in $\frac{m}{s}$, which is allowed to be negative. Minimums were decided we defined this way to allow simple configuration of non-symmetric velocity limits.

Maximum Linear Velocity This is the maximum linear velocity of the vehicle in $\frac{m}{s}$.

Minimum Angular Velocity This is the minimum angular velocity of the vehicle in $\frac{rad}{s}$, which is allowed to be negative.

Minimum Angular Velocity This is the maximum angular velocity of the vehicle in $\frac{rad}{s}$.

Clearance Threshold This is the minimum acceptable distance between the vehicle and an obstacle. It is not guaranteed that this threshold will never be violated, rather a very high cost is associated with traveling this close to obstacles. This distance is defined as the center of the robot to the center of an obstacle in meters. This center to center distance is because all objects are modeled as points.

Obstacle Threshold Obstacles that are farther from the robot than this distance are treated as attractive rather than repulsive. This allows the robot to approach obstacles in order to allow for some degree of environmental exploration and prevents the robot from simply settling in the middle of the largest expanse. This quantity is measured in meters.

Turning Parameters

Five parameters are used to shape the function used to generate turning commands for the robot. They affect the attraction and repulsion to the goal and obstacles as well as dampening oscillations.

Goal Attraction This gain influences the strength of the goal attraction force. Larger values increase attraction strength.

Obstacle Repulsion This gain influences the strength of the obstacle repulsion force for obstacles closer than the Obstacle Threshold distance. Larger values increase repulsion strength.

Obstacle Attraction This gain influences the strength of the obstacle attraction force for obstacles farther than the Obstacle Threshold distance. Larger values increase attraction strength. This predisposes the robot to point in the direction of far obstacles.

Obstacle-Goal Bearing Ratio This gain tunes the trade off between avoiding obstacles which are in the robot's current direction of travel versus avoiding objects which are in the direction of the goal. This parameters ranges from 1 to 0. Values closer to 1 amplify the avoidance of obstacles in the direction of travel. Values closer to 0 amplify the avoidance of obstacles in the direction of the goal.

Vehicle Inertia Gain This gain influences the strength of the vehicle's resistance to turning. Larger values mean more resistance to turning. This can be seen as increasing the inertia of the robot and will help dampen high frequency oscillations associated with intermittent obstacle detection and local minima conditions. The algorithm is constructed such that forward motion is reduced in the presence of large angular velocities so high frequency changes in desired heading angle slow the robot down.

Forward Motion Parameters

Three parameters are used to shape the function used to generate forward motion commands. The term forward motion is used as this function only commands the robot to move in the direction (or in the opposite direction) of its current heading. A different function would have to be formulated to take advantage of the planar holonomy of the Nao humanoid, which would likely have different parameters.

Velocity Gain This gain influences the "aggressiveness" of the forward velocity function. Larger values effectively increase the slope of this function, meaning the change in velocity will be greater for a given change in the other input parameters.

Obstacle Repulsion This gain influences the strength of repulsion force generated by the closest obstacle. Larger values increase repulsion strength. The strategy here is that the robot should slow down if there are obstacles close to it. Only the closest obstacle is used to modify forward velocity.

Angular Rate Braking This gain influences the amount by which high turning rates reduce forward velocity. Larger values reduce forward velocity to a greater extent. In general, when the robot is trying to turn quickly, it is likely that heading forward in that direction is no longer desired.

5.1.4 Experimental Arena

Three arenas were constructed to test the efficacy of the GODZILA algorithm. In all three, the goal location is observable by the robot at all times as the robot had no other method to know goal location, nor is there any provision in the GODZILA algorithm for doing searches or explicit explorations. The first arena was simply an open area in which the robot could walk towards the goal. This trivial environment acts as a control in which the robot is always expected to reach the goal. The second arena contains an obstacle which divides the arena into two parts such that the only means to reach the goal is through a narrow opening. The objective is to demonstrate the narrowest opening that can be traversed using this method. Finally, an arena was constructed which had a large obstacle that would always have a large local influence on the navigation. The robot must overcome the large repulsive forces and walk through a hallway to the goal. In common to the three arenas was a perimeter wall which acted to contain the environment and prevent the presence of unplanned obstacles from influencing test results. The arena perimeter was approximately $11.75m$ long by $5.5m$ wide. As an aside, none of the perimeter walls are depicted in any of the figures seen in this chapter. This is simply an aesthetic choice and are present in each experiment.

Arena Construction

The arenas were constructed using large cuboidal objects and flat walls. Aside from the ease of construction, this is motivated by the choice of sensor used on the robot for obstacle avoidance. The cuboids and walls have the same cross section when viewed from above. As the LIDAR used only detects objects in a plane, obstacles which have overhangs or voids at different levels are effectively invisible to the robot and pose a

Figure 52: This figure shows the open arena. The Nao can bee seen on the left at the starting location while the goal cube can be seen to the right.

collision risk. This is not a shortcoming of the algorithm but rather the sensor, therefore, using obstacles that avoid this risk is appropriate.

Open Arena

The open arena contained no obstacles and only the arena perimeter walls and the goal object were present to affect the navigable path. Figure 52 shows a picture of the arena as seen by the overhead camera.

Narrow Opening Arena

This arena contains an obstacle that divides the area into two equal parts, with the robot and goal being in opposing partitions. The only path between the two rooms is a narrow opening that is $73cm$ wide. This width is approximately 2.6 times the shoulder-to-shoulder width of the robot. Figure 53 shows the arena with the narrow opening. The objective was to demonstrate the narrowest opening that the robot could travel through with this method. It was found than openings narrower than this caused the robot to approach other obstacles too closely when the parameters were tuned to allow the robot to traverse this aperture. At times, with the parameters tuned to allow for tighter openings, the robot would collide with corners it might otherwise have avoided. To allow for the navigation of narrower openings that are still physically traversable, one strategy is to have a global path planner that could place intermediate goals to "pull" the robot through. As GODZILA is designed to be a lightweight local algorithm, such

Figure 53: This figure shows the arena with the narrow opening between partitions. The opening is approximately 2.6 times the width of the robot and represents the practical limit to the traversable apertures using this approach. The Nao can bee seen on the left at the starting location while the goal cube can be seen to the right.

provisions are inappropriate for addition to the approach as they tend to require maps. The different algorithm classes and approaches are reviewed in Chapter III.

Large Obstacle Arena

This arena contained the largest single obstacle possible while still allowing the arena to be traversable. It consists of a corridor constructed to be the minimum width a hallway can be with this approach. The corridor also has two 90° turns. The objective was to have the robot navigate in an area where the repulsion of obstacles was a strong influence during the entire path. It shows that while the robot is constantly being repelled, it can use the goal and far obstacle attraction to find a path. Figure 54 shows the arena.

Figure 54: This figure shows the arena containing a large obstacle. The result is a hallway that is the narrowest constructible and still have this approach be viable. This results in the robot being under strong repulsion for the length of the path. The Nao can bee seen on the left at the starting location while the goal cube can be seen to the right.

5.1.5 Robot Pose Estimation

In order to track the robot's path for later analysis, a GoPro camera was mounted on the ceiling above the arena that could record the starting and goal location in the same frame. This meant that the camera could remain fixed throughout the experiment, simplifying later analysis. As presented in Section 5.3, this data was processed using the OpenCV library in Python to track the Nao through the video and display the path of the robot for each experiment.

5.2 Goal Pose Tracking

During navigation, the robot used its camera to track the goal location relative to the robot as discussed in Section 5.1.2. Figure 55 shows the perceived relative goal range and bearing during the open arena experiment while Figures 56 and 57 show the results from the narrow opening and large obstacle experiments. All three datasets show the goal range data following an approximate asymptotic decrease over time. This is expected as in each of the arenas the robot gets closer to the goal in each step. While these datasets demonstrate this occurrence, in general there are some arenas where

the range could be constant for a time or even increase slightly. In the presence of a long obstacle that was radially constant from the goal, the range would constant. Alternatively, if there was a short wall that was in the path of the goal, the range could increase as the robot avoids it.

The goal bearing angle in the open arena converges to a small value during the majority of the experiment while in the narrow opening and large obstacle experiments, the bearing can be seen as oscillating around some slowly decreasing function. In all of the experiments, the robot is initially oriented facing the goal to ensure strong goal tracking. In the open arena, this means the robot is already nearly optimally oriented to approach the goal, and only small corrections to heading are necessary to reach the goal. In the narrow opening and large obstacle arenas, the robot must first turn away from the goal to avoid obstacles before the robot can start to face the goal again. Figures 59 and 60 give additional insight into this process. The oscillations are likely due to a combination of corrections to the navigation process and to large head movement due to gaiting oscillations. Section 5.3 gives more insights to these gaiting oscillations, due to the added LIDAR mass discussed in Section 5.1.1.

In Figures 55, 56, and 57, it is also shown that the final perceived goal range is close to the $0.3m$ prescribed by the Goal Stopping Radius from Table 7. Section 5.3 shows that the robot stopped at a radius closer to $0.8m$, a mismatch generated by the lack of calibration mentioned in Section 5.1.2.

Additionally, a future application of this data is to localize the robot during the experiment. This localization data could be used to inform other path planning algorithms or trap detection schemes to improve navigation in more complex environments.

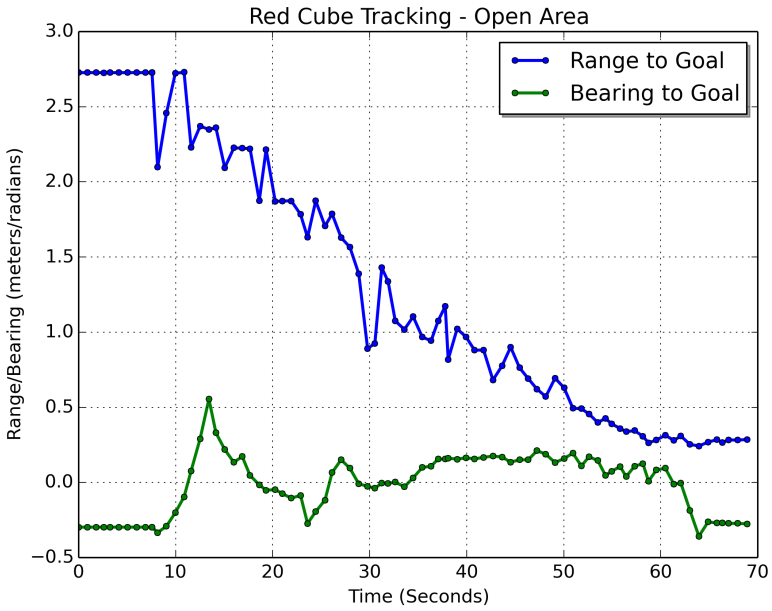

Figure 55: This figure plots the perceived range and bearing to the goal relative to the robot during the open arena experiment. The range to the goal decreases until the Goal Stopping Radius is reached.

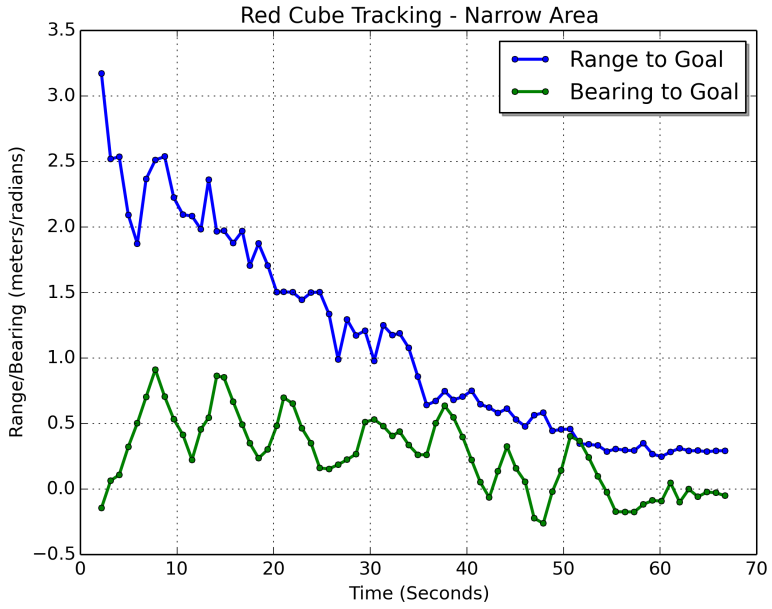

Figure 56: This figure plots the perceived range and bearing to the goal relative to the robot during the narrow opening arena experiment. The range to the goal decreases until the Goal Stopping Radius is reached.

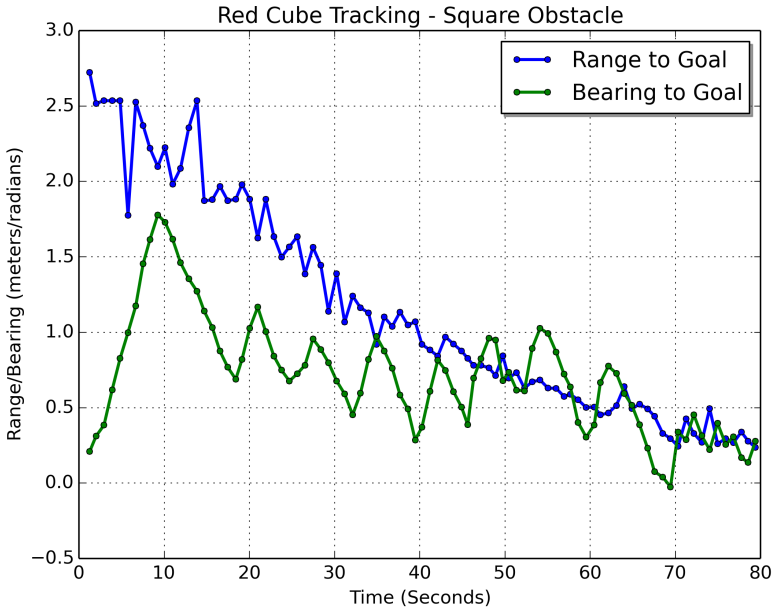

Figure 57: This figure plots the perceived range and bearing to the goal relative to the robot during the large obstacle arena experiment. The range to the goal decreases until the Goal Stopping Radius is reached.

5.3 Robot Pose Tracking

In order to record the pose of the robot during the experiments, a GoPro camera was mounted above the arena. This global camera could see the entire arena during the experiment, simplifying path analysis. The resultant videos show the robot traversing from the start location to the goal location while avoiding obstacles. The video was then analyzed to extract the approximate location of the robot in each frame and a path was produced from these samples.

5.3.1 Observed Path

As recorded by the global camera during the experiments, Figures 58, 59, and 60 show the robot as in traverses the arenas. The best fit path, explained in Section 5.3.2, is overlaid over each frame to better demonstrate the progression of the robot through the arena. In each of the experiments, the robot can be seen navigating from its starting position, to some position near the goal cube, while avoiding collision with obstacles. The robot does not approach the obstacles too closely, nor does it wander in the open areas. An issue that can be seen in each of the experiments, is the camera parallax between the head and feet of the robot as well as the goal cube and stand. This effect

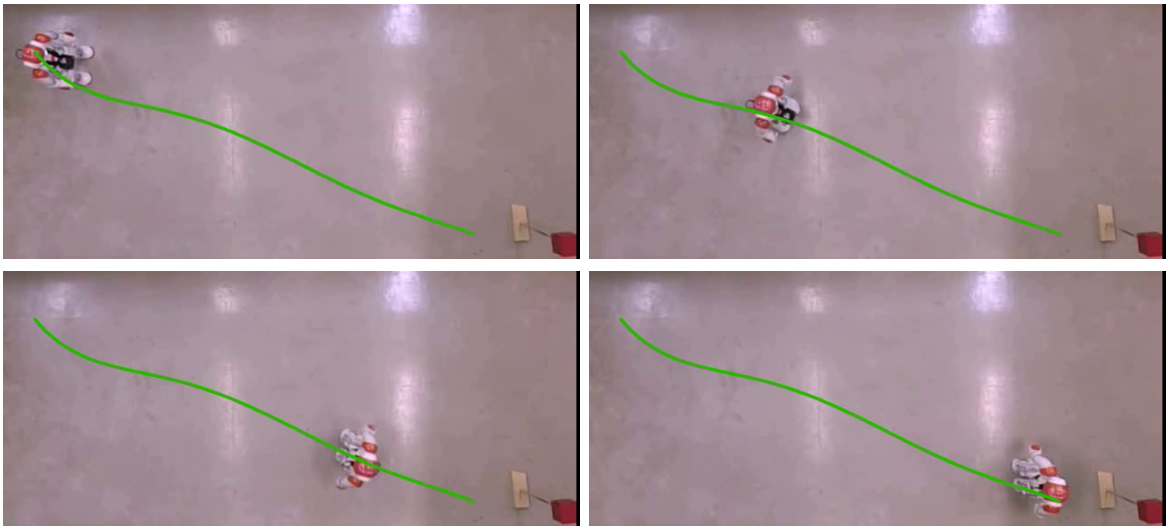

Figure 58: These images show the robot as it progresses through the open arena. The path of the robot is overlaid onto each of the images. The robot moves along a straight path to the goal.

exaggerates the farther an object is from the center of the frame. As the best fit path is a result of tracking the orange part of the Nao during the experiment, the path is distorted. This is especially apparent in Figure 60, where the path towards the bottom of the image seems to be farther from the obstacles than one might expect.

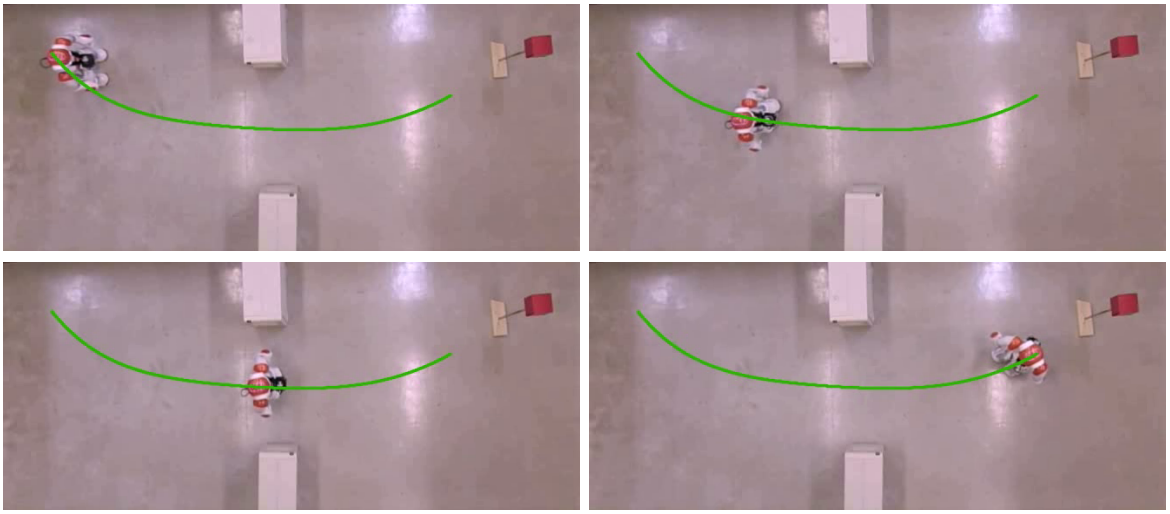

Figure 59: These images show the robot as it progresses through the narrow opening arena. The path of the robot is overlaid onto each of the images. The robot moves towards the middle of the narrow opening and then towards the goal.

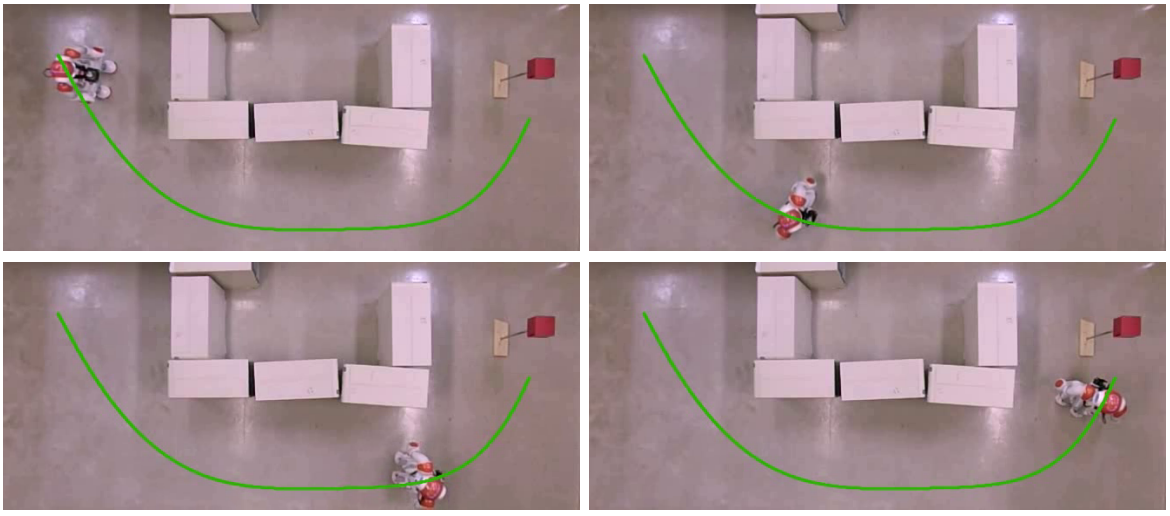

Figure 60: These images show the robot as it progresses through the large obstacle arena. The path of the robot is overlaid onto each of the images. The robot moves down from its starting location, along the corridor, and to the goal. Camera parallax distorts the path and makes it seem that the robot walks farther away from the obstacles during the middle of the run. The upper-right and lower-left images show that this is not the case.

5.3.2 Tracking Analysis

Using the global camera data from each of the experiments, the robot was tracked through each video to produce an approximation to the navigated path. The OpenCV library was used to process the images in Python. Briefly, the procedure used to produce the path from each video was:

1. Extract orange pixels from frame into a new image.
2. Filter that image using a closing kernel to reduce noise.
3. Find centroids of remaining pixel "blobs".
4. Eliminate blobs with small areas and join nearby blobs.
5. Process every video frame to produce array of centroids.
6. Group centroids using a density-based clustering algorithm.
7. Manually select the group corresponding to the robot path.
8. Fit a polynomial to the centroid clusters as the robot path estimate using least-squares.

A fifth-order polynomial was used to approximate the path of the robot because it was the lowest order polynomial fitting that produced a good result. The results from this procedure can be seen in Figures 61, 62, and 63 for the open arena, narrow opening arena, and large obstacle arena, respectively. The centroid samples and best-fit path are overlaid onto one another to show how the path approximates the path of the robot from the samples. In these plots, the samples clearly illustrate the oscillations in the walking gait. While every bipedal gait will produce some oscillations in the center-of-mass motion, the gait instabilities are amplified by the addition of the LIDAR mass as discussed in Section 5.1.1. This effect is especially evident in Figure 62 towards the middle of the plot. This corresponds to the area near the left side of the narrow opening. While it is not known why the magnitude of the oscillations at this point are so large, it is possible that they are not due to the nearby obstacle but due to an irregularity in the arena floor in this region. This is theorized because the large obstacle arena contained apertures which were the same width as the narrow opening. Despite this, Figure 63 does not show any oscillations approaching the magnitude of those seen in Figure 62.

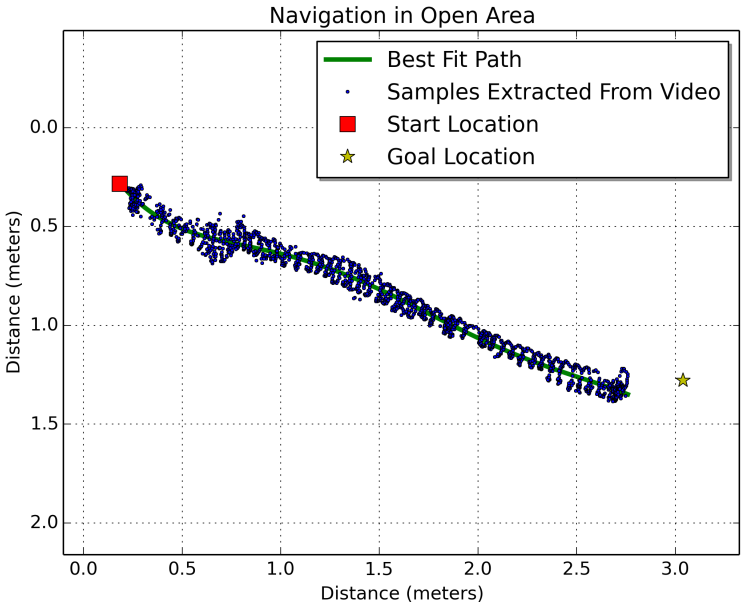

Figure 61: This plot shows the centroid samples extracted from the global camera video from the open arena experiment. A fifth order polynomial has been fit to the samples and overlaid onto the plot. The robot starting point is shown as a red square on the left and the goal point is shown as a yellow star on the right. The sample centroids clearly show the wobble in the gait. The robot stopped approximately $0.8m$ from the goal.

Lastly, as mentioned in Section 5.1.2, despite the Goal Stopping Radius from Table 7 being $0.3m$, the robot stops approximately $0.8m$ from the goal. This is close to double the parameter value, which is what was predicted as the cube was about twice as big as the "red ball" tracker expected it to be. The apparent size of the cube when approached from one of the corners, seen with greatest effect during the narrow opening experiment, would have also contributed to the robot stopping farther from the goal than intended.

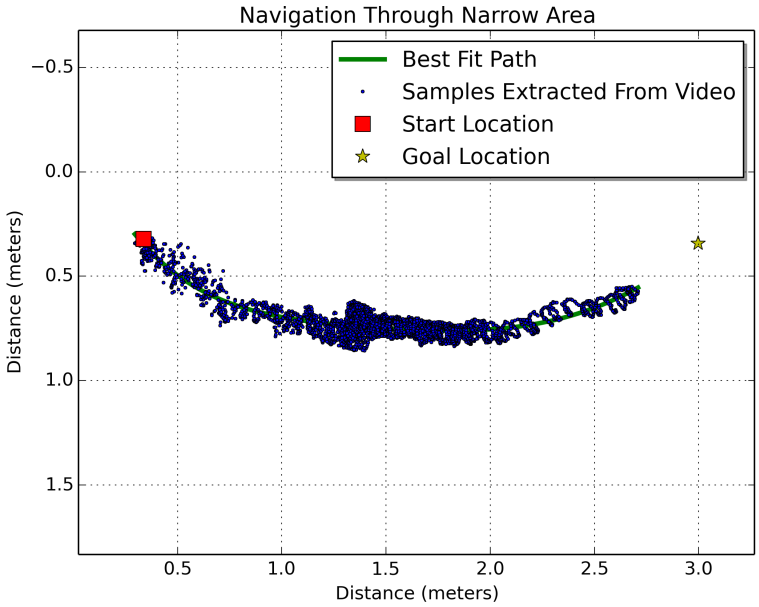

Figure 62: This plot shows the centroid samples and best-fit path from the narrow opening arena experiment. The oscillations shown by the samples can be seen to be the largest in the $(1.4, 0.75)$ region of the plot.

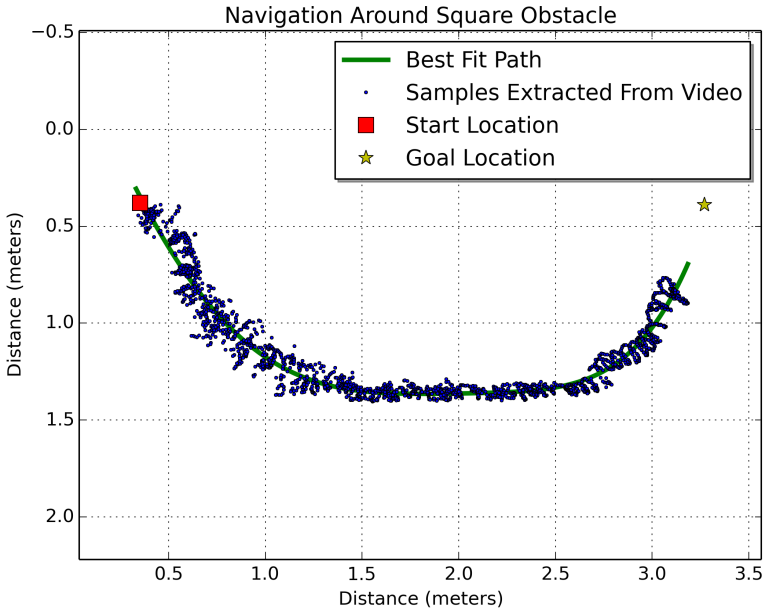

Figure 63: This plot shows the centroid samples and best-fit path from the large obstacle arena experiment. The oscillations reduce along the path towards the bottom of the plot, but this is more of a limitation of the image processing procedure and when viewing the video, it is clear the oscillations are of similar magnitude to those seen in the rest of the experiment.

CHAPTER VI

Humanoid Low Profile Crawl Gait Results

To test the efficacy of the Projected Profile crawling gait, a gait sequence was generated using MATLAB and simulated using the V-REP simulator by Coppelia Robotics. The crawling sequence was generated using the nominal crawling parameters. The Nao humanoid platform was then programmed using the NAOqi API in C++ to use the generated sequence to execute a crawling action. The robot was initialized to its crawling pose and placed on the ground where it proceeded to crawl under a vertically constrained table. During this experiment, the joint angles and joint motor currents were recorded for later analysis. To demonstrate a context under which the gait would be used, the Nao was programmed to walk to a fiducial marker, position itself into the crawling position, crawl under the vertically constrained table, and then return to a sitting position. NAOqi provides functions for the Nao to walk, position itself into a set of poses from any initial pose, and track a "red ball". The fiducial marker was a red circle on a piece of paper. The robot crawled under the table by executing a finite number of crawling sequences, before moving to a sitting position. A video of this experiment was recorded for later analysis.

Following this, the optimized crawling parameters were used to generate a new gait sequence in MATLAB. The optimized and nominal gait sequences were tested using the V-REP simulator and the simulated joint torques were recorded. The V-REP simulator provides a MATLAB API to record these torques. While the optimized gait sequence was formulated using the pseduo-static assumption, the gait was tested at different speeds to compare the increase in efficiency against the nominal gait for varying degrees of dynamic loading.

Details about the crawling environments, simulation, and crawling parameters can be found in Section 6.1. Data collected about the nominal crawling experiment is presented in Section 6.2, and the optimized crawling experiment in Section 6.3.

6.1 Experimental Setup

The crawl gaits were tested on the Nao humanoid in the V-REP simulator using MATLAB, and by having the actual robot crawl under a vertically constrained table using the NAOqi API in C++. The nominal parameters were tested both in simulation and on the actual robot, while the optimal parameters were only tested in simulation. MATLAB has tools to use genetic algorithms to optimize systems, which were used here to generate the optimal parameter splines.

6.1.1 Mobile Platform

The Nao humanoid was used to test the Projected Profile algorithm. Details about the Nao are discussed in Chapter II. It makes a convenient platform to test crawling algorithms as NAOqi provides an extensive API to control a range of parameters from individual joint angles to full body positioning. Importantly, the kinematic configuration of the robot is amenable to the crawling paradigm and its relatively small size allows environments to be easily constructed for testing.

6.1.2 Crawling Environments

Prior to any testing on the actual robot, the V-REP Simulator by Coppelia Robotics was used to verify that the algorithms functioned correctly and exhibited the desired behaviors. The V-REP simulator is a good choice for this application for a variety of reasons. It uses the mature and well known Open Dynamics Engine (ODE) as its physics simulator, supports multiple operating systems such as Windows and Linux, and provides an API for use with many languages such as C++, Python, and MATLAB. What's more, it provides a model of the Nao humanoid that can be easily commanded. Figure 64 shows an example of the Nao in V-REP set at the initial crawling position. The V-REP API also simulates various sensors, and allows the torque at each of the Nao's joints to be accessed. As mentioned in Chapter IV, this data was used to generate the gait-parameter-triplet-to-joint-torque mapping for the genetic optimizer. The simulated joint torques were also accessed while running the optimized crawling gait at different speeds to compare its improvement over the nominal gait. Results from those experiments are presented in Section 6.3.2.

While traversing rough terrain or over small obstacles is one application of a crawl gait, these experiments were instead designed to show that the robot could access areas with demanding height constraints. For this, a small table was used whose sides were

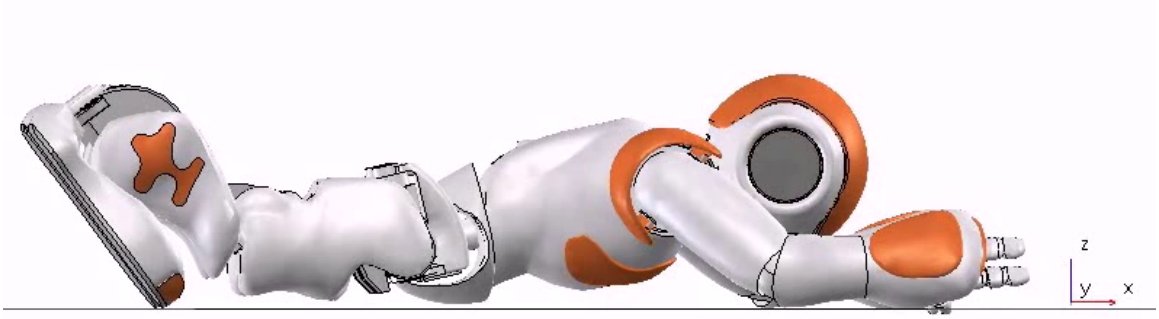

Figure 64: This figure shows the Nao humanoid V-REP simulation. The robot is positioned at the initial configuration of the close chain phase of the crawl gait.

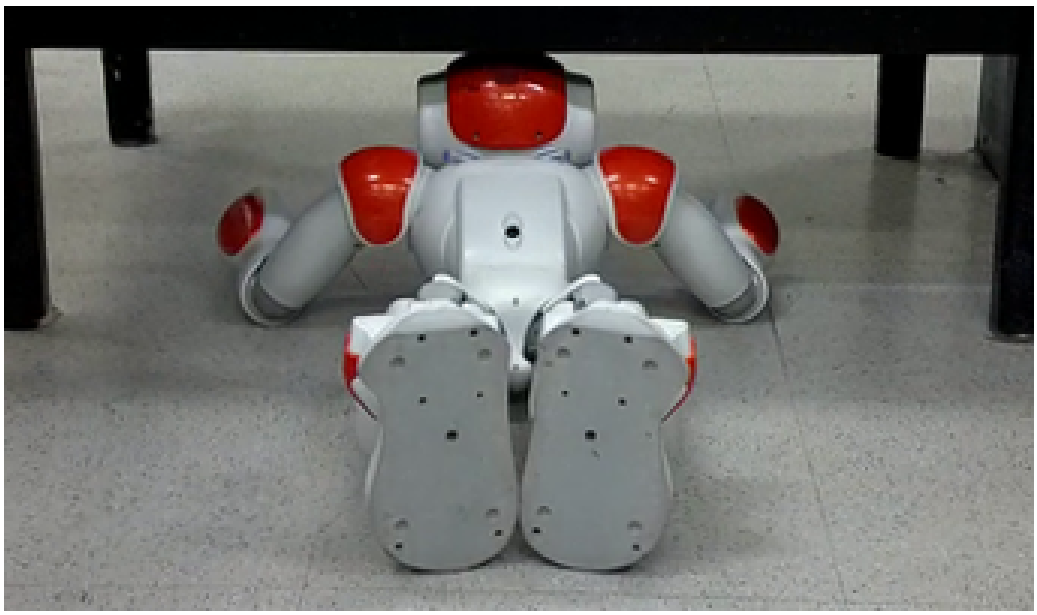

Figure 65: This figure shows the Nao as it crawls under the table with the low panels. The head of the robot nearly touches the panel, representing the lowest practical traversable height constraint.

blocked with panels down to a minimum of about 200 mm off of the ground. Figure 65 shows the Nao crawling under this table, with its head almost touching the bottom of the panel. This represents the practical height constraint that the Nao can satisfy with this gait.

6.1.3 Gait Parameters

As discussed in Chapter 4.2.2, the closed chain phase of the Projected Profile crawl gait can be parameterized on three angles, $[\theta_3, \theta_4, \alpha]$. These three variables are referred to as the gait parameters, or alternatively as the angle triplet. The nominal angle triplet can be seen in Table 8. It holds the θ_3 and θ_4 joints constant, while increasing the angle α linearly as a function of time.

Results using these nominal gait parameters can be seen in Section 6.2.

In order to optimize the gait using these parameters, the procedure outlined in Chapter 4.3 was used, modeling the gait parameters as cubic splines. The Genetic Algorithm in MATLAB's Global Optimization Toolbox was used in conjunction with the V-REP torque table detailed in Chapter 4.3.1 to generate the optimal spline parameters. As the genetic algorithm cannot guarantee finding the global optima for any arbitrary function, the optimization was executed several times and the best spline parameters were used for the experiment. Each optimization used 50 to 80 generations to converge on results, but as can be seen in Figure 66, after about 10 generations the optimization had already reached minima and was simply exploring nearby states for possible improvements.

Table 9 shows the spine parameters resulting from the optimization procedure. Each gait parameter is now a cubic polynomial which is a function of time $[\theta_3(t), \theta_4(t), \alpha(t)]$. The coefficients from the table are used in the cubic spline $c_3t^3 + c_2t^2 + c_1t + c_0$. The gait parameter splines are constrained by the starting and ending angles for each pa-

Gait Parameter	**Value**
θ_3	$16.5°$
θ_4	$27.5°$
α	$-30°$ to $-90°$

Table 8: Table of gait parameters for the nominal crawl gait. θ_3 and θ_4 are held constant, while α is linearly decreased in proportion to time.

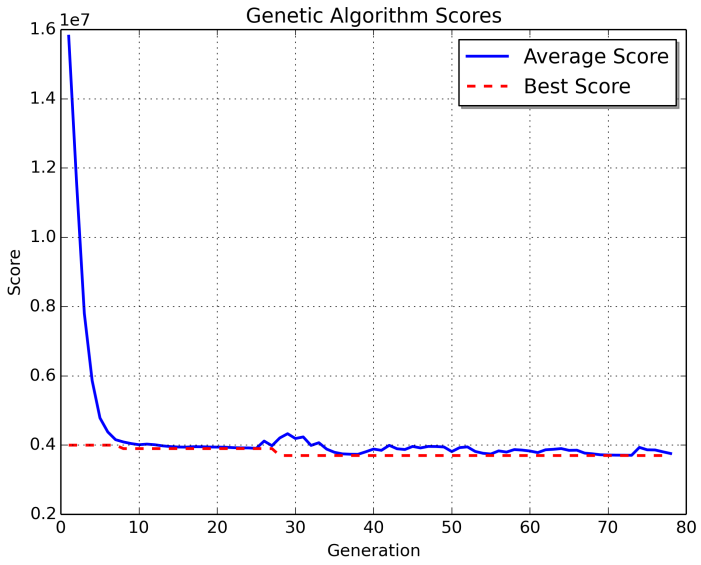

Figure 66: This figure shows one of the gait parameter optimization trials. As the genetic optimization procedure uses multiple children as detailed in Chapter 4.3.3, the plot shows the average score for the children and the score of the best child as the generations progress.

rameter. When $t = 0$, the splines must produce the starting angles for the triplet, $[\theta_3(0), \theta_4(0), \alpha(0)] = [0.28798, 0.47997, 0.52360]$. At the final time $t = t_f$, the splines must produce $[\theta_3(t_f), \theta_4(t_f), \alpha(t_f)] = [0.28798, 0.47997, -1.5710]$. The initial and final angle triplets are in radians.

Figure 67 shows the splines graphically, with the nominal gait parameters overlaid for comparison. While the optimized parameter trajectory for α is similar to its nominal trajectory, the trajectories for θ_3 and θ_4 dip significantly, with θ_4 having the most drastic deviation from the nominal.

Gait Parameter	c_3	c_2	c_1	c_0
θ_3	0.2365	0.0893	-0.3267	0.28798
θ_4	1.8796	-0.1365	-1.7434	0.47997
α	-0.2134	1.1570	-1.9898	0.52360

Table 9: Table of gait parameter coefficients for the optimal crawl gait. The c_i coefficients are used in the cubic spline $c_3t^3 + c_2t^2 + c_1t + c_0$ to vary the gait parameters as a function of time. The c_0 coefficients are simply the initial starting angles for each parameter.

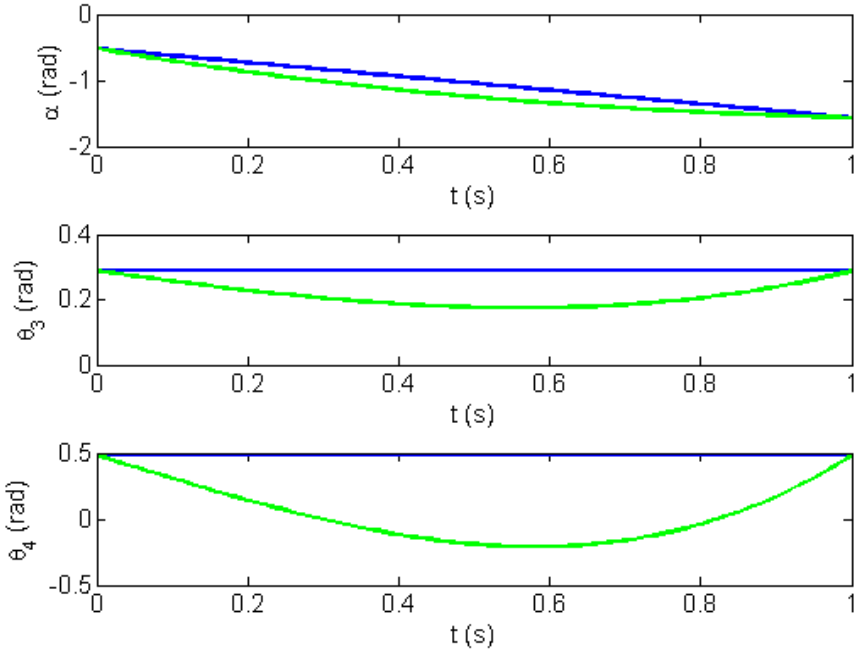

Figure 67: This figure shows the optimal gait parameter trajectories overlaid onto the nominal trajectories. The blue lines represent the nominal trajectories while the green lines show the optimal.

6.2 Nominal Crawl Gait Data

To test the Projected Profile gait using the nominal parameters reviewed in Section 6.1.3, the gait sequence was first generated in MATLAB and then simulated in V-REP. Following this, the gait was tested on the Nao by setting it to crawl under a vertically constrained table in two different experiments. In the first, the robot was set to the initial crawling pose and placed on the floor in front of the table. The robot then executed the crawling gait. In the second experiment the robot was programmed to recognize a fiducial marker, represented by a red circle mounted to the table, and walk to it. The robot then moved to a prone posture and crawled under the table. This procedure is more throughly examined in Section 6.1. Joint angle and joint motor current data was collected during the table experiments and is presented in this section. These table experiments demonstrate the efficacy of the gait to locomote the robot and the low profile nature of the gait to allow access to vertically constrained spaces.

6.2.1 Simulations

Figure 68 shows samples of the closed chain phase of the Projected Profile gait using the nominal parameters. It shows a simplified kinematic model representing a projection of the robot onto the saggital plane. The frames show the model starting in the initial pose, and by linearly increasing the α gait parameter and holding the θ_3 and θ_4 parameters constant, the model shifts forward until α has reached its terminal value of $-90°$. This places the model at the final closed chain pose. It can be seen that the highest point of this gait occurs when the ankle joint is at about $z = 100\ mm$. This model of course does not include the limb thicknesses nor the head of the Nao, as it only models joint centers. This model was created using MATLAB and is used to view the results of the gait sequence generation.

Once the gait sequence is generated, it is tested using the V-REP simulation of the Nao humanoid. Figure 69 shows the simulated Nao executing the closed chain phase of the nominal crawl gait. As with the simplified kinematic model, the α gait parameter is linearly increased from $-30°$ to $-90°$, which moves the robot forward. As detailed in Chapter 4.2.3, the gait parameters and other constraints are used to position the arms of the robot. Unlike the ankle pitch, hip pitch, and shoulder pitch joints which have angles that directly correspond to joint angles in the simplified kinematic model, the shoulder roll, elbow yaw, and elbow roll joints do not. The head in this simulation

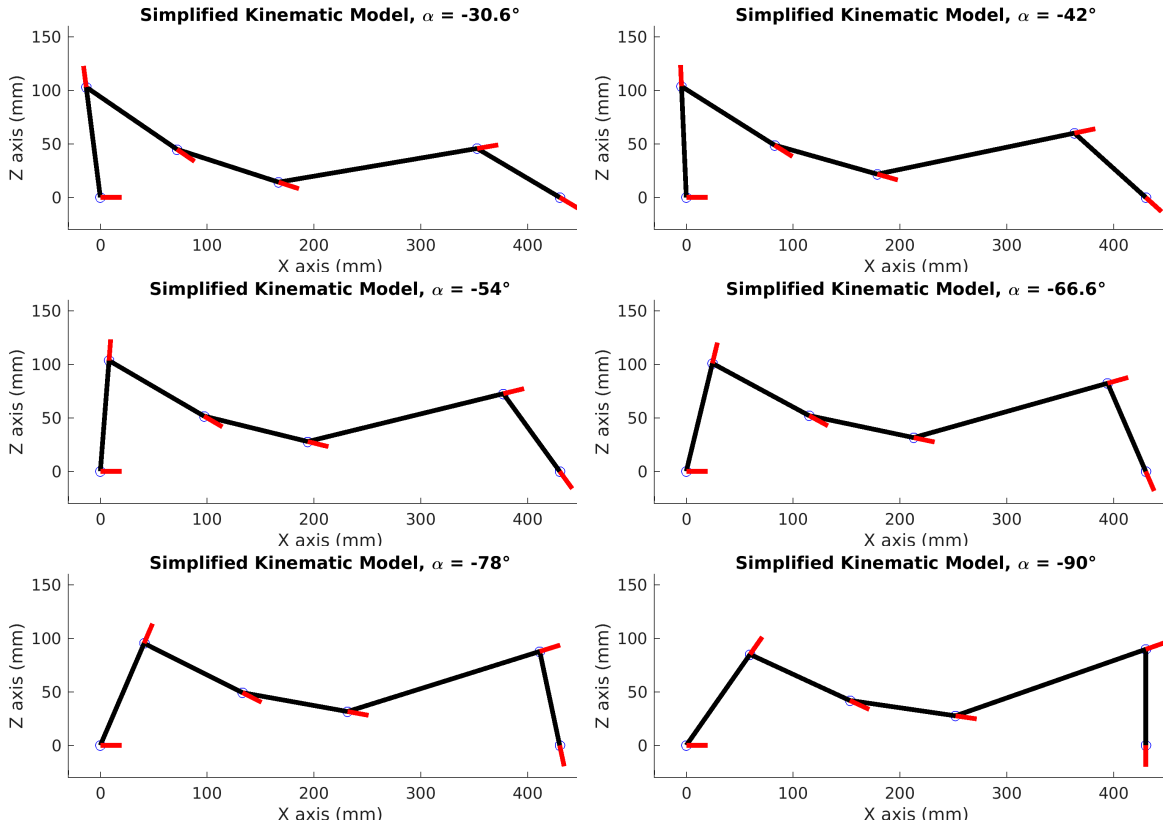

Figure 68: This figure shows a simplified kinematic model of the Nao as it executes the Projected Profile gait using nominal parameters. The model starts with $\alpha = -30°$ and terminates when $\alpha = -90°$.

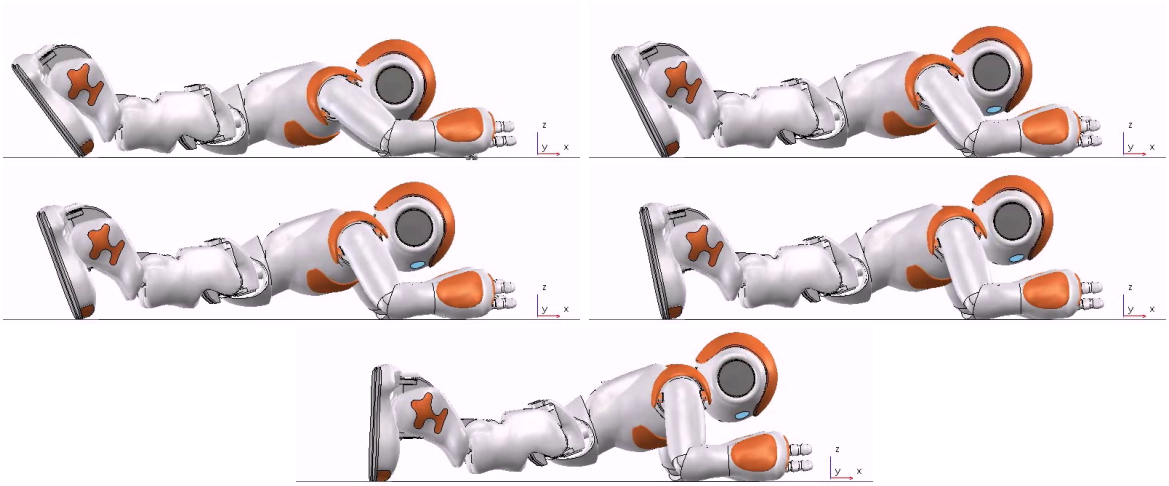

Figure 69: This figure shows the simulated Nao executing the closed chain phase of the gait using the nominal parameters.

can be seen to be the highest part of the robot throughout the majority of the gait, increasing the minimum value of the allowable vertical constraint.

6.2.2 Vertically Constrained Table

Following simulation, the Projected Profile crawl gait using the nominal parameters was tested on the Nao. Figure 70 shows the initial test of the robot crawling under the vertically constrained table. The Nao is set to the initial crawling pose on the floor outside of the table. The robot then executes the crawl gait for several sequences until the robot is under the table. For this experiment, the robot executed 9 sequences in 27 seconds. The robot traveled about 1 body length or 610 mm, equating to a velocity of 22.6 $\frac{mm}{s}$.

Figure 71 and 72 show the second experiment performed with the vertically constrained table. The robot has detected and walked to the red marker affixed to the table. It then transitions to a prone posture and begins the crawling sequence. After having crawled to the other side, the robot transitions to a sitting posture. The ability to transition from posture to posture is provided through the NAOqi API. For this experiment, the robot executed 25 sequences in 58 seconds. The robot traveled about 2.75 body lengths or 1,676.4 mm, equating to a velocity of 28.9 $\frac{mm}{s}$.

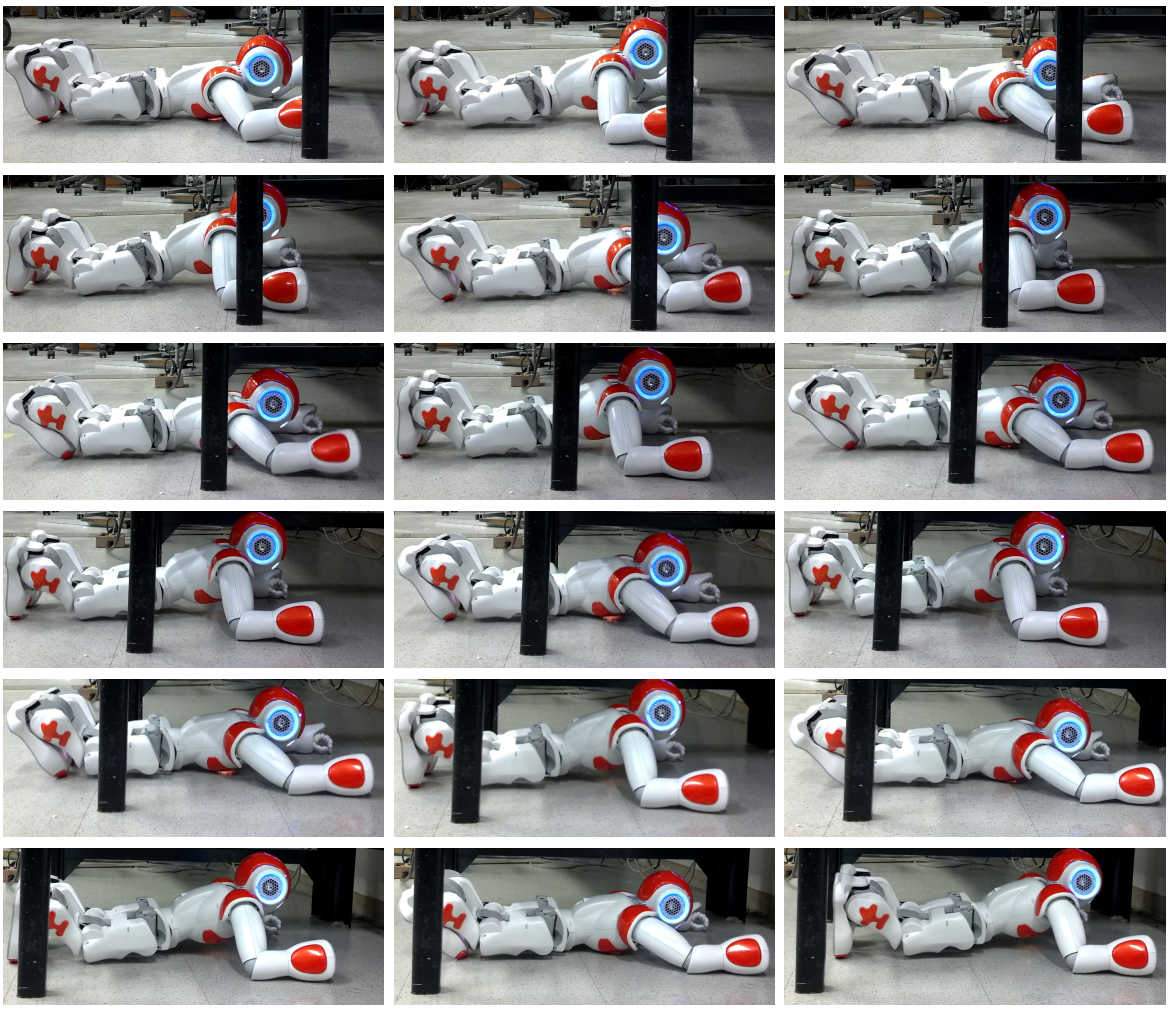

Figure 70: This figure shows the first experiment of the Nao crawling under the vertically constrained table.

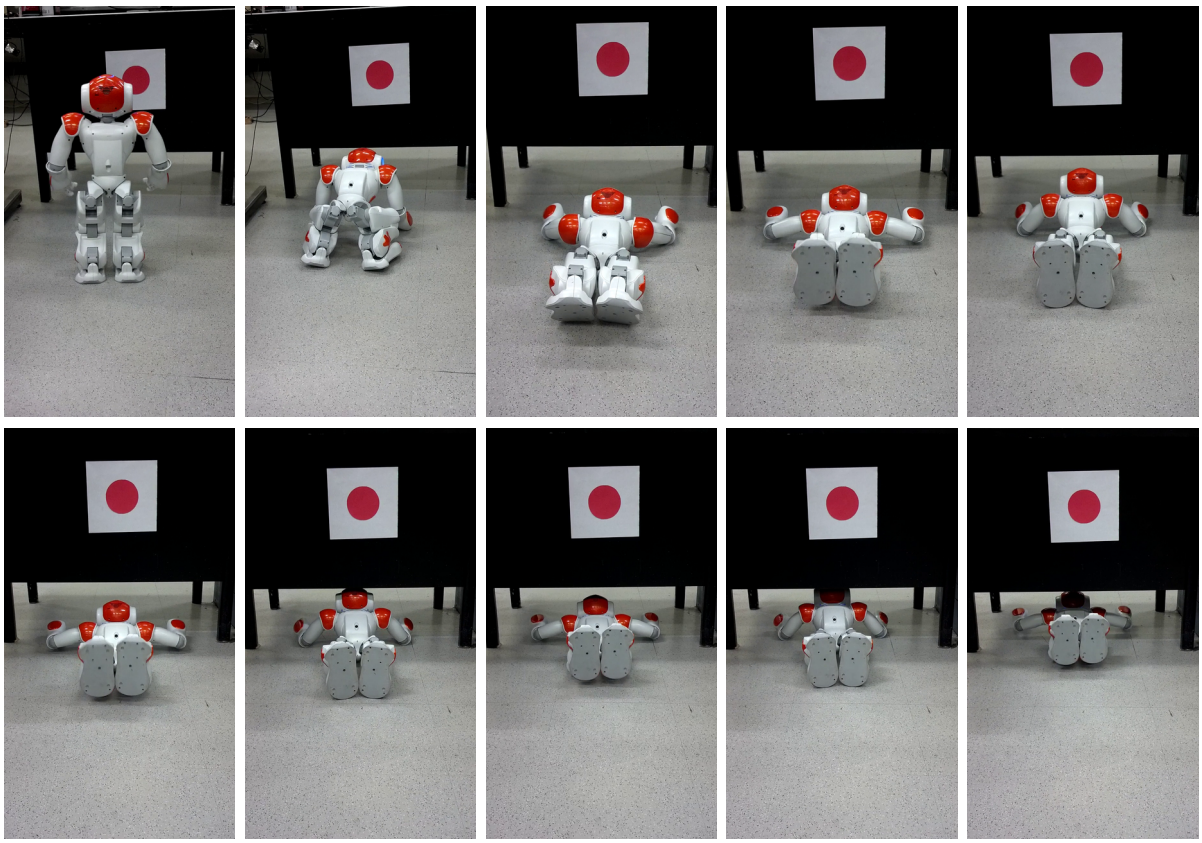

Figure 71: This figure shows the approach portion of the second vertically constrained table experiment. A red circle is used as a marker for the direction in which the robot is commanded to move. When the robot approaches below a specified distance threshold from the red circle, the crouch-down and crawl gait sequence is initiated.

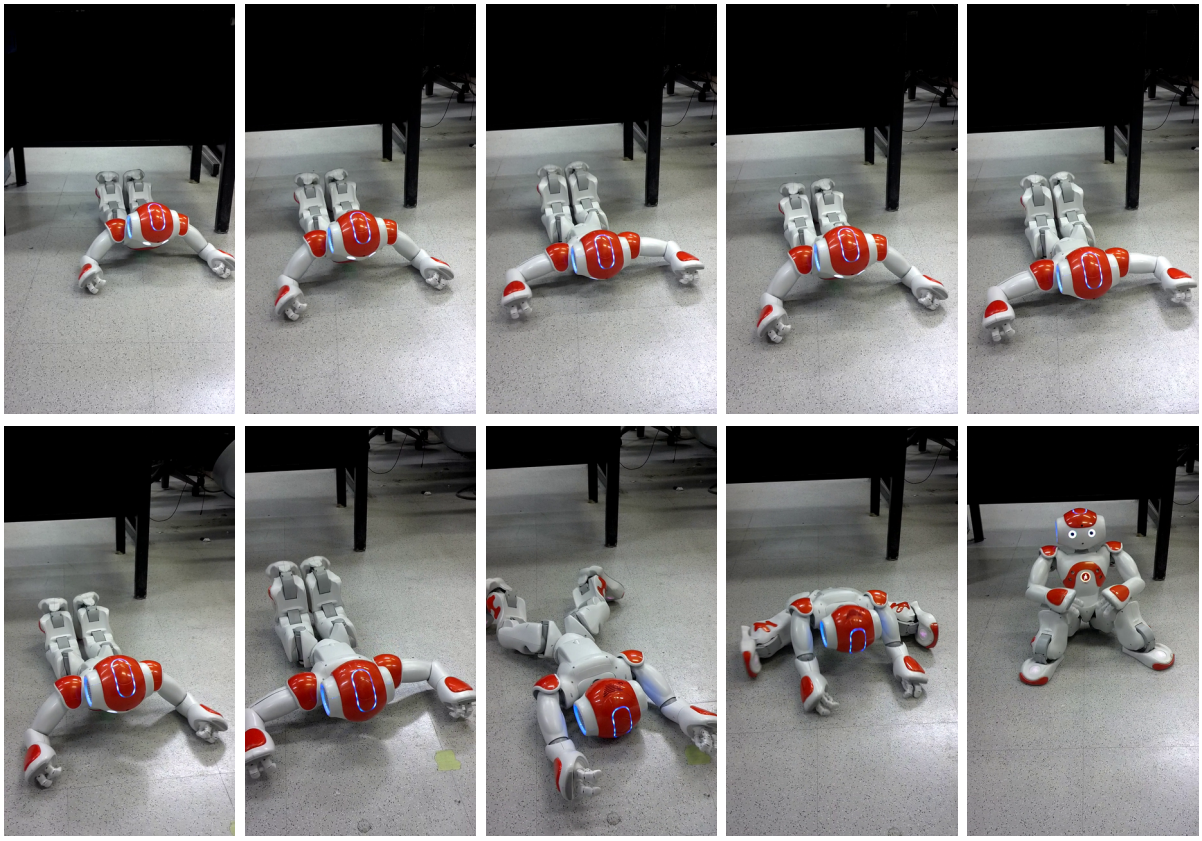

Figure 72: This figure shows the recovery portion of the second vertically constrained table experiment. After the Nao has executed a set number of crawl sequences, the robot transitions to a sitting posture.

6.2.3 Joint Data

The NAOqi API provides functions for recording joint angles and motor currents. During the nominal crawl gait experiments, these values were recorded and Figures 73, 74, and 75 plot the results. Figure 73 plots the joint angles for the arms and legs of the robot during the execution of multiple gait cycles. As expected, the plots show a clear periodicity to the gait. As the gait is laterally symmetric, the joint angles for the left and right side of the robot are similar, with the exception of the arm joints below the shoulder pitch. The joint angle curves for the shoulder roll, elbow yaw, and elbow roll shown in Figure 73 have a mirror symmetry rather than being identical. This is due to the robot frame and joint frame definitions from the NAOqi API. Figure 16 shows that the joint frame definitions are such that increasing the shoulder roll angle of each arm rotates each arm to the left. The elbow yaw and roll joints have a similar behavior. Therefore, to command the arm to the required positions, the right arm shoulder roll, elbow yaw, and elbow roll angles are the opposite of the left arm versions. In addition, the hip-yaw pitch, hip roll, and ankle roll are not symmetric here as they were set to small constants that do not actively participate in the Projected Profile gait.

Figure 74 shows the full joint angle sequence as the robot transitions from standing to crawling. The Nao starts in a standing posture and then transitions to a crouch posture. From there, it transitions to the initial crawling pose and begins to crawl. It executes five crawl sequences to simulate crawling under an object. Only five sequences were performed in order to make the plot of the joint angles easier to present. The robot then transitions back to the crouching posture. Each of the posture transitions was executed using the NAOqi ALRobotPosture API. Figure 17 shows the different postures used in these transitions.

While the Nao platform is equipped with encoders that can directly measure joint angle, it is not equipped with joint torque sensors. The robot is equipped with sensors that measure joint motor current which can be used to estimate joint torque. The motor currents were recorded for several crawl sequences and are presented in Figure 75. While the plots do show a periodicity to the motor current draws, it is difficult to observe other useful information. Though there is a relationship between motor current and joint torque, in general, current sensors give a poor estimate of torque. One limiting factor is that each joint uses some form of motor control to bring the joint to a desired angle. This controller will obfuscate the torque-current relationship as it draws power

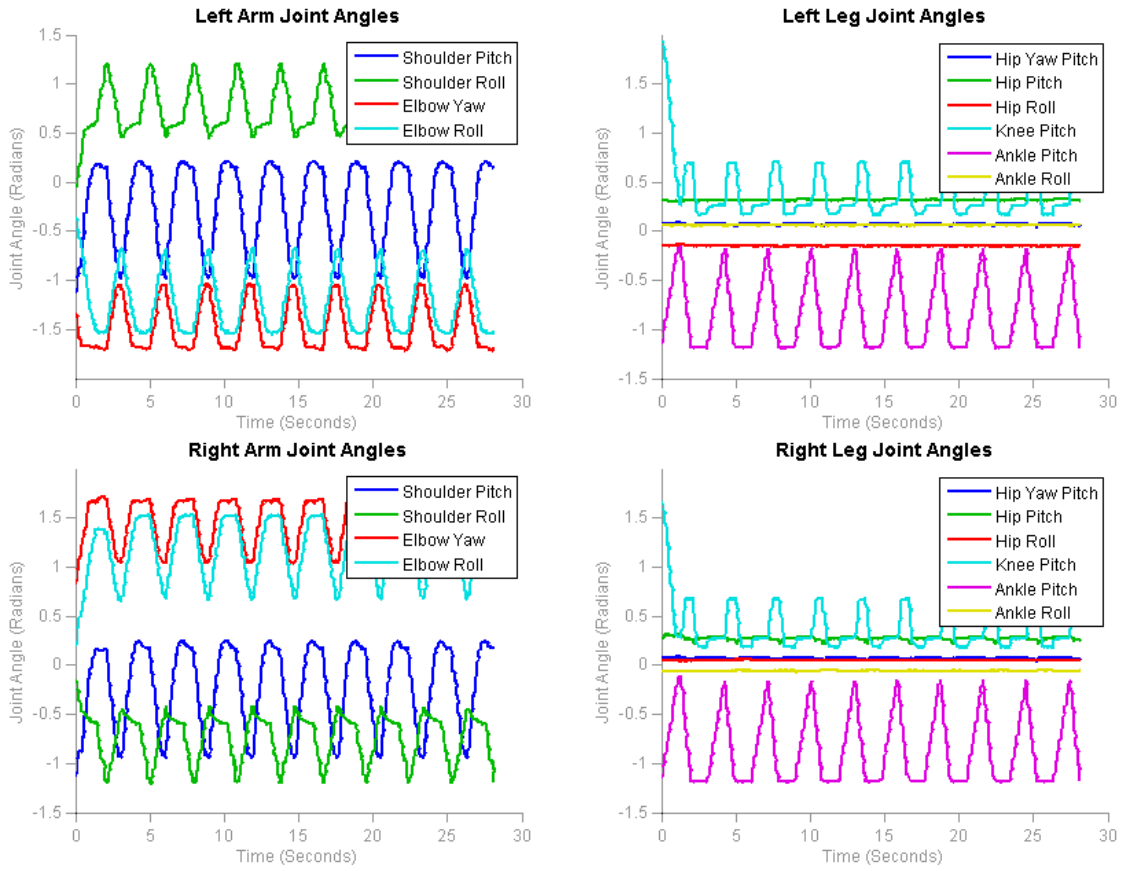

Figure 73: This figure shows the measured joint angles during multiple iterations of the periodic crawling gait. The angles for the left and right side can be seen to be identical, except for those of the shoulder roll, elbow yaw, and elbow roll. These have a mirror symmetry, due to the joint frame definitions.

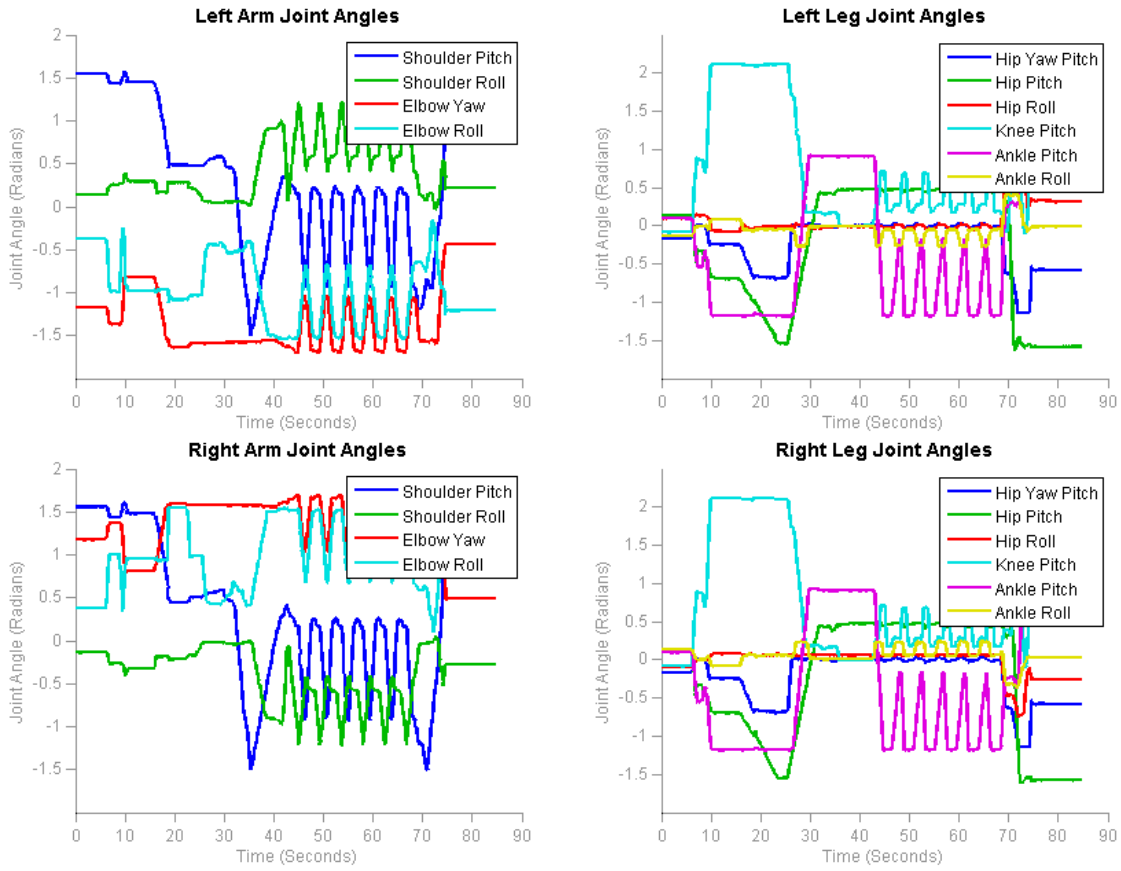

Figure 74: Measured joint angles for a sequence of transitioning from standing to crouch to crawling, crawling under a table, and then returning to crouch.

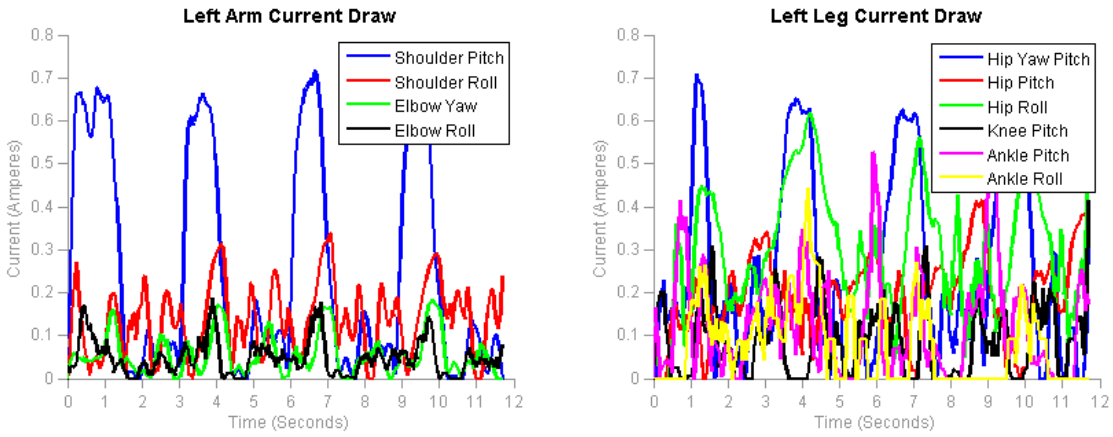

Figure 75: Measured motor current draws during multiple iterations of the periodic crawling gait.

to position the joint. Therefore, the joint current plots are of limited use.

6.3 Optimized Crawl Gait Data

To test the Projected Profile gait using the optimal parameters reviewed in Section 6.1.3, the gait sequence was generated in MATLAB and simulated in V-REP. The closed chain phase of the gait was the only portion that was optimized, so this phase of the gait was simulated and analyzed. As reviewed in Section 6.1.2, the V-REP simulator provides joint torque simulation and sensing, which can be recorded to evaluate the increase in gait efficiency using these parameters.

6.3.1 Simulations

Figure 76 shows samples of the simplified kinematic model, as it executes the closed chain phase of the optimized Projected Profile gait. As with the nominal crawl gait, the angle α transitions from $-30°$ to $-90°$, though not linearly, as reviewed in Section 6.1.3. While it is difficult to see the difference in the optimized α trajectory from the nominal version, the trajectory of θ_3 and θ_4 are quite different from those seen in Figure 68. As opposed to the nominal gait which appears to hold the midsection of the model low to the ground, the optimized gait arches the midsection upwards as the gait progresses, before being brought down. This resembles the tendency for humans to arch their back when attempting to support their weight in similar positions.

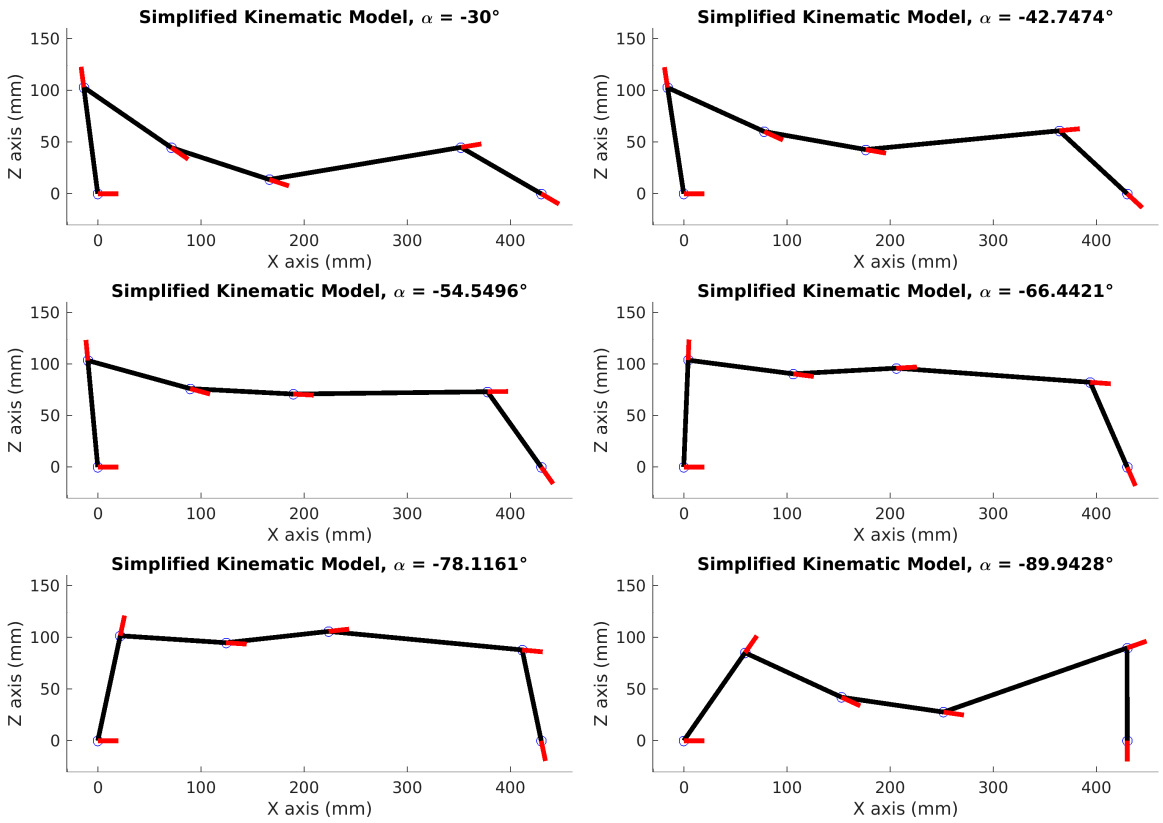

Figure 76: This figure shows the simplified kinematic model executing the close chain phase of the optimized gait.

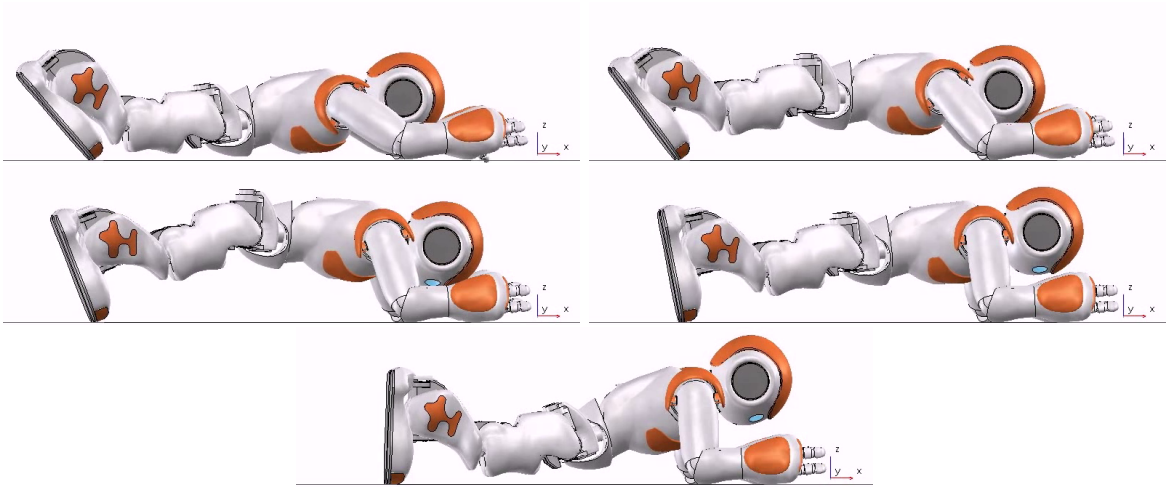

Figure 77: This figure shows the simulated Nao executing the closed chain phase of the optimized gait.

The V-REP simulation of the Nao executing the optimized gait can be seen in Figure 77. As with the simplified kinematic model, the Nao can be seen to be using its hips and knees to arch its back during the gait sequence.

6.3.2 Joint Torque Data

In order to analyze the efficiency increase to the gait using the optimal gait parameters, the torques from the simulated Nao were recorded during nominal and optimal closed chain phase gait executions. The simulated torques are easier to interpret than the motor currents presented in Section 6.2.3 as they directly represent the torques the Nao would experience during a crawl. The ankle pitch, knee pitch, hip pitch, and shoulder pitch torques of the simulated Nao were recorded, corresponding to the Projected Profile joints $[\theta_2, \theta_3, \theta_4, \theta_5]$. These were the joints which were used in the optimization procedure and were therefore recorded for analysis. To test the efficacy of the pseudo-static assumption, the crawl gaits were executed at different speeds so that the effects of transient torques would be present to varying degrees. As discussed in Chapter 4.3.1, the pseudo-static model assumes that gravity is the dominant force in the system and the transients do not affect the system significantly. Ten speeds were tested for each gait. The duration of each closed chain gait phase varied from one second to ten seconds, in one second increments. The transient torques will have a stronger influence on the joint torques at the higher speeds.

Figures 78 and 79 show joint torque plots from the nominal and optimal gaits,

respectively. Each panel shows the torques for each of the four joints, over the prescribed gait duration. Only a subset of the experiments are shown as the differences between panels diminishes as the duration increases. This is likely due to a shift from the dominant component of the system being the transient torques to the pseudo-statics.

Figures 80 and 81 show the torque curves for each joint as a function of time. The panels overlay the torques onto each other which shows their similarity during the initial portion of the gait, before fanning out in the later portion. This similarity is demonstrated more clearly in Figures 82 and 83, which highlight the transients. For the nominal gait, the joint torque curves for each joint from about $t = 0.0$ seconds to $t = 0.25$ seconds are similar regardless of the gait duration. The optimal gait has a similar time interval at around $t = 0.0$ seconds to $t = 0.175$ seconds. After these points, as seen in Figures 80 and 81, the curves tend to diverge. This gait-duration-invariant-with-respect-to-time portion suggests that the system transients are the dominant forces in this region. For the shorter duration gaits, this means the transients dominate as much as 20% of the gait cycle. But as the gait cycle duration is increased, the transients contribute to a diminishing percentage of the gait cycle, which appears as the left most portion of the panels in Figures 78 and 79 being compressed. The differences between the 7 and 10 second gaits for example, are less pronounced than between the 1 and 3 second gaits.

This similarity is more clearly demonstrated in Figures 84 and 85. These panels plot joint torques for each joint as a function of gait cycle percentage, rather than time. Within each of the panels, the joint torques from 20% to 100% can be seen to be similar between durations. This gait-duration-invariant-with-respect-to-cycle-percentage portion suggests that the pseudo-statics are the dominant forces in this region. This is because joint torques are now only a function of position and not the time it takes to get to that position.

When comparing the nominal gait torque curves to the optimal ones, as in Figure 86, a number of effects can be seen. For short duration gaits where the transient dynamics are dominant, a pattern is less obvious though the amplitude of the torque transients for joints θ_3 and θ_4 have been reduced. In the Nao, these are the knee pitch and hip pitch joints. In the longer duration gaits, the optimization appears to have the effect of compressing the transients to the left and reaching the affine region of the torque curve more quickly. This means while the torque curves do transit through these higher magnitude regions, they spend less time there and therefore accrue a lower torque cost.

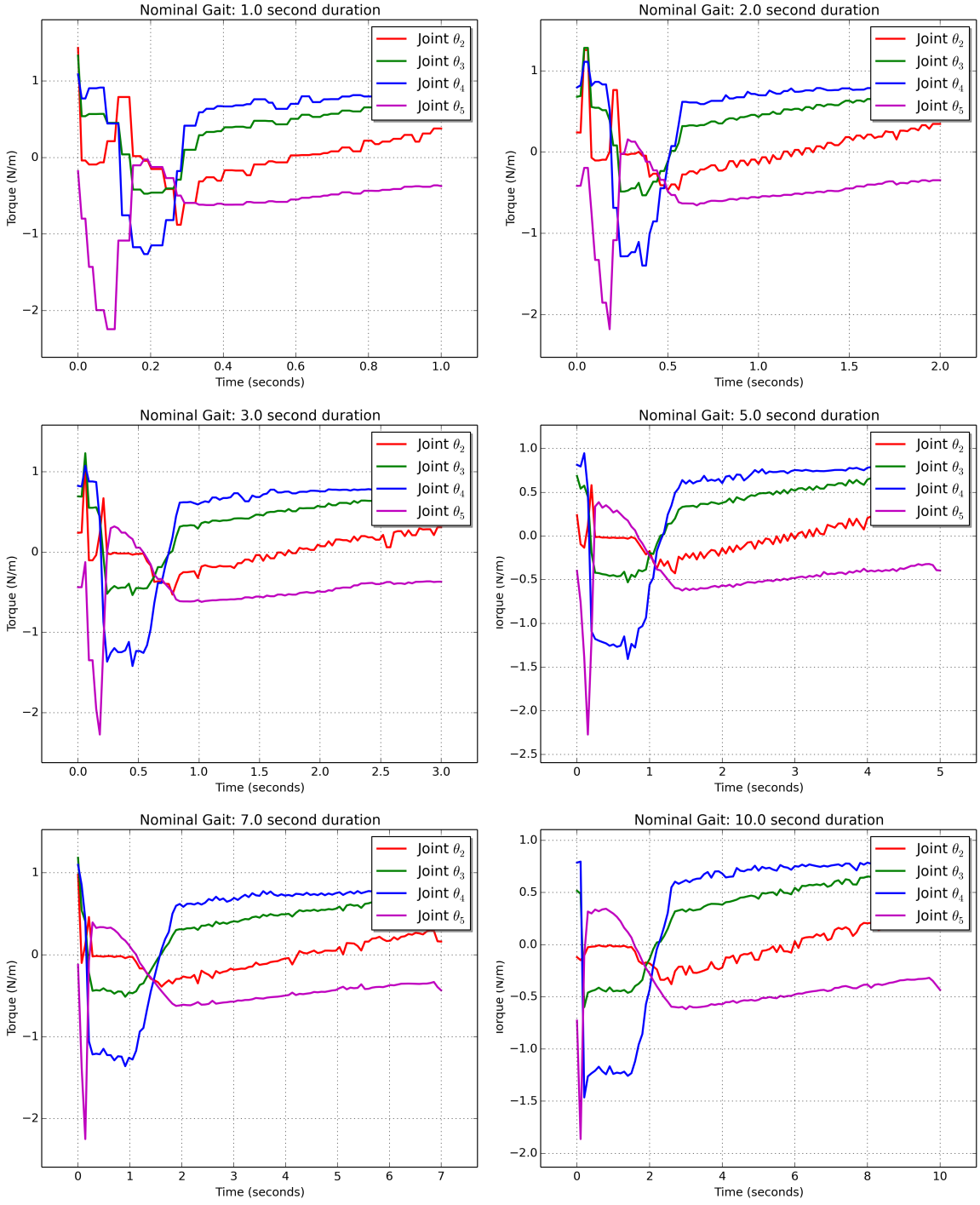

Figure 78: This figure shows the nominal joint torque curves for six durations. As the duration increased, the graphs show less variation.

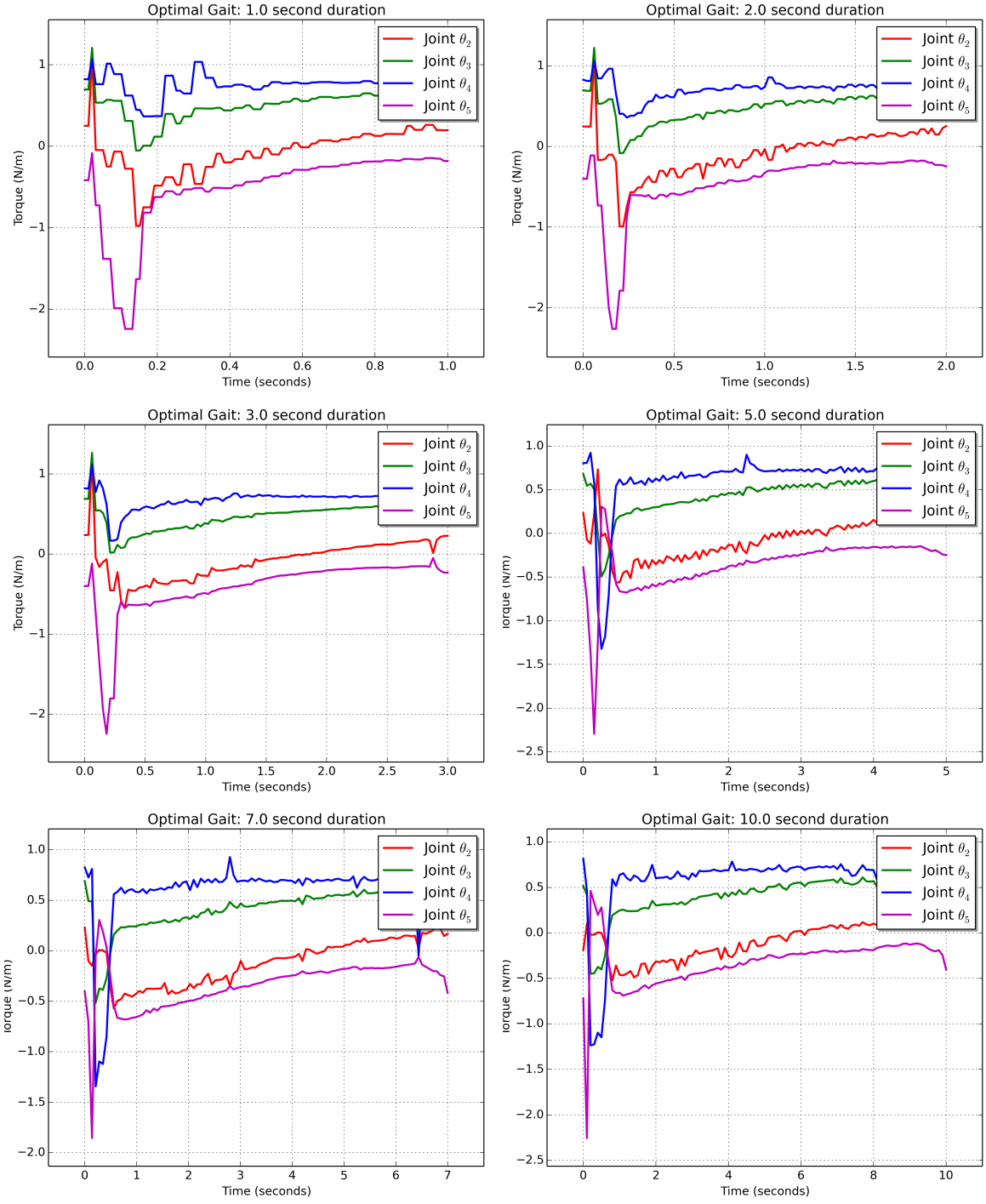

Figure 79: This figure shows the optimal joint torque curves for six durations. As the duration increased, the graphs show less variation.

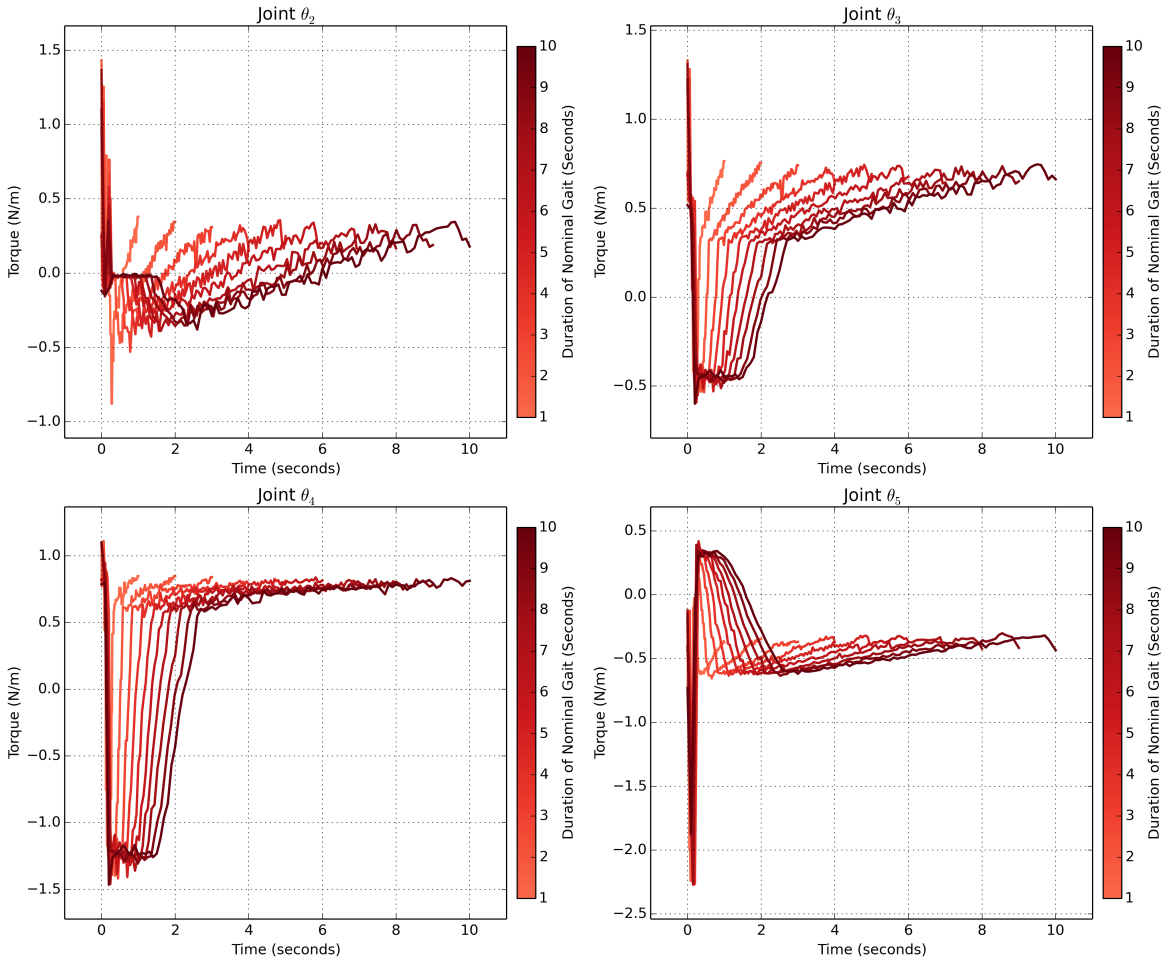

Figure 80: This figure shows the nominal joint torque curves as a function of time. Each panel shows the curves for one joint. The curves can be seen to be similar up to a certain time, at which point they diverge. Figure 82 shows a magnified view of the similar portions of the curves.

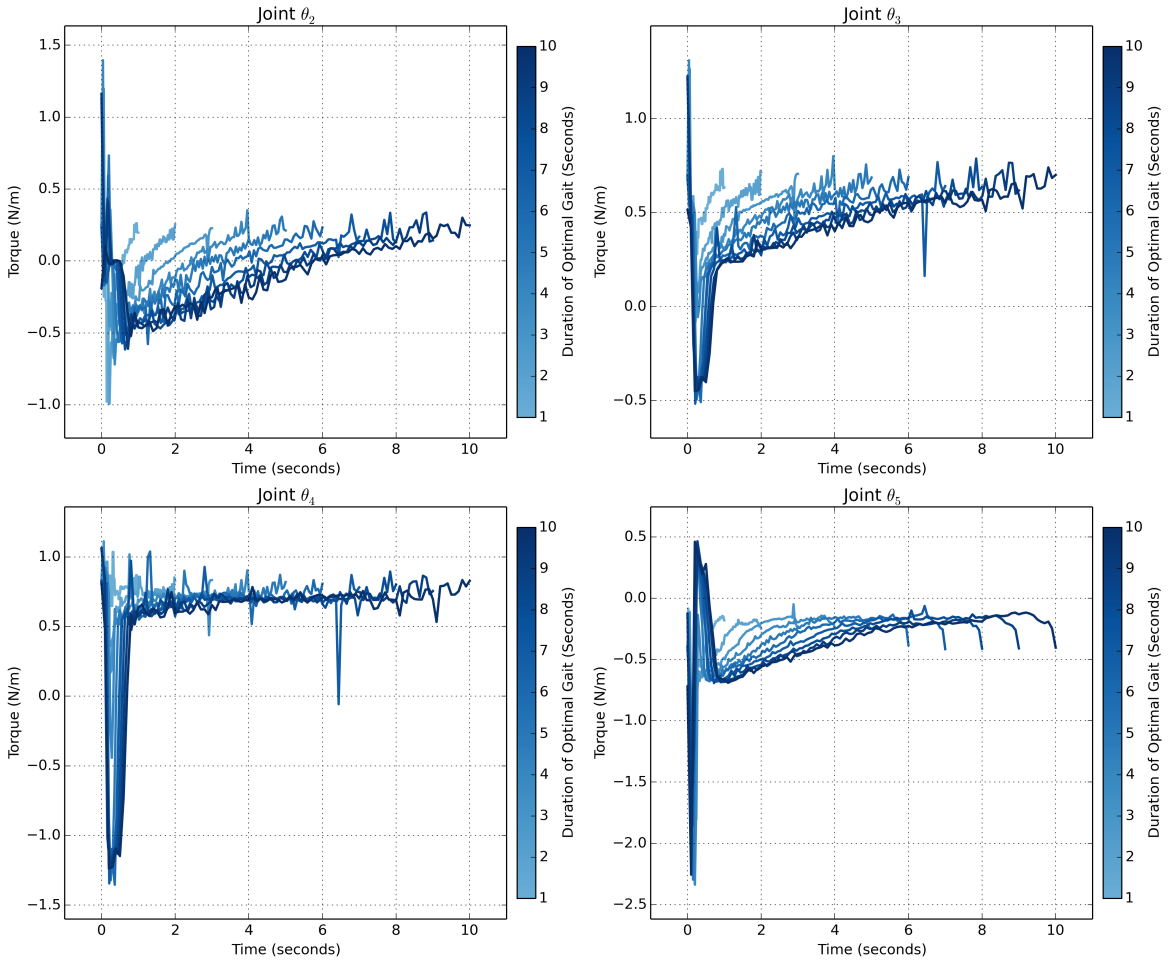

Figure 81: This figure shows the optimal joint torque curves as a function of time. Each panel shows the curves for one joint. The curves can be seen to be similar up to a certain time, at which point they diverge. Figure 83 shows a magnified view of the similar portions of the curves.

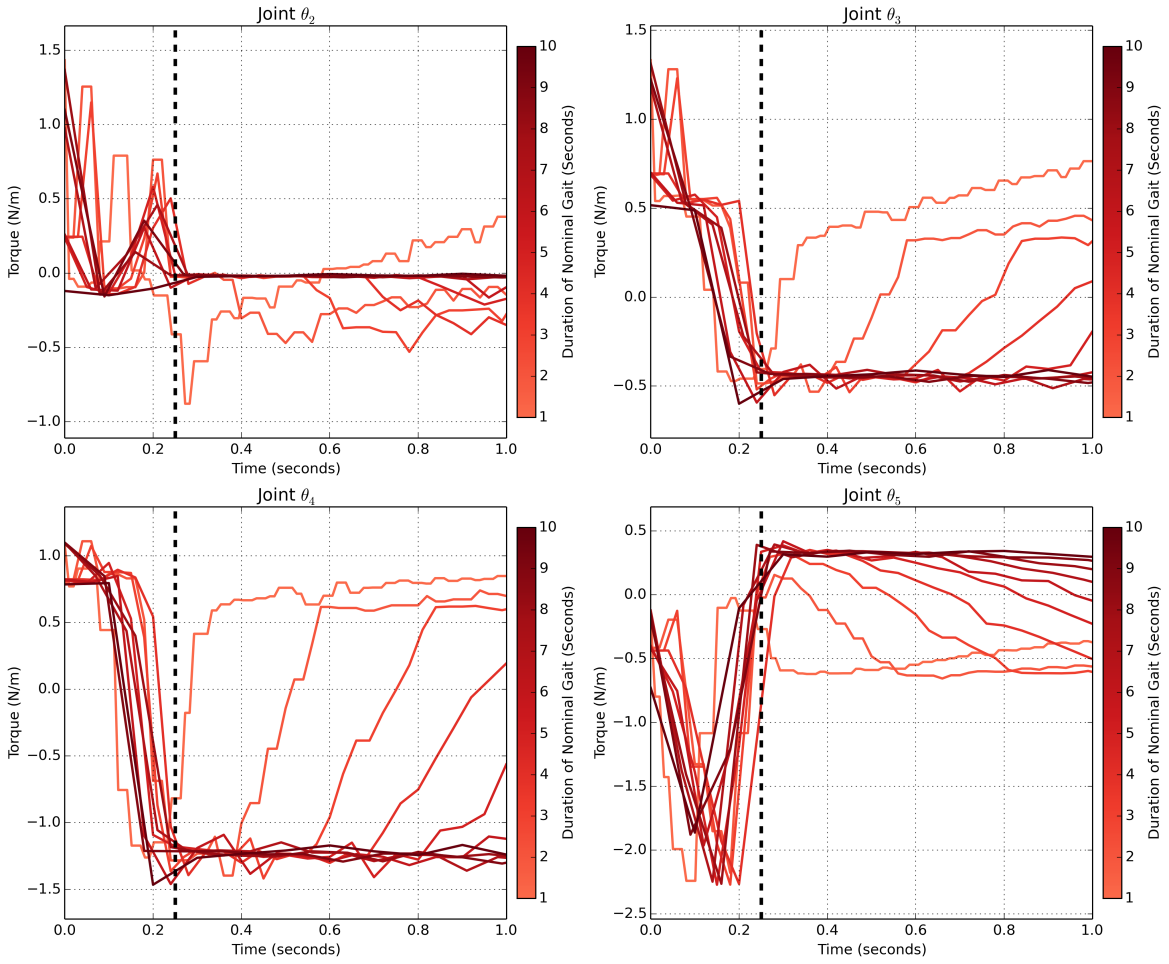

Figure 82: This figure shows a magnified view of the nominal joint torque curves as a function of time. Each panel shows the curves for one joint. The curves are overlaid to show their similarities when viewed with respect to time. The dashed black line signifies the time at which the curves diverge.

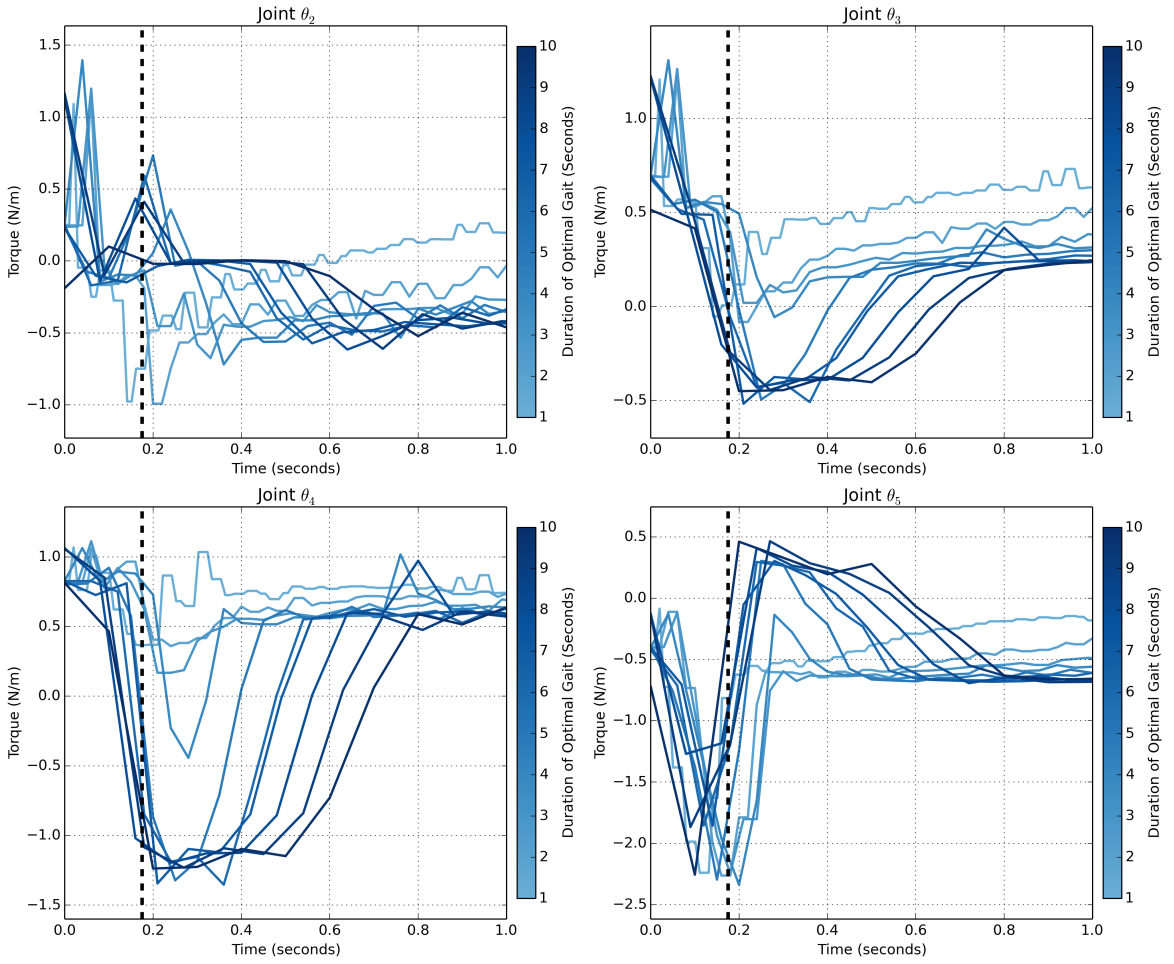

Figure 83: This figure shows a magnified view of the optimal joint torque curves as a function of time. Each panel shows the curves for one joint. The curves are overlaid to show their similarities when viewed with respect to time. The dashed black line signifies the time at which the curves diverge.

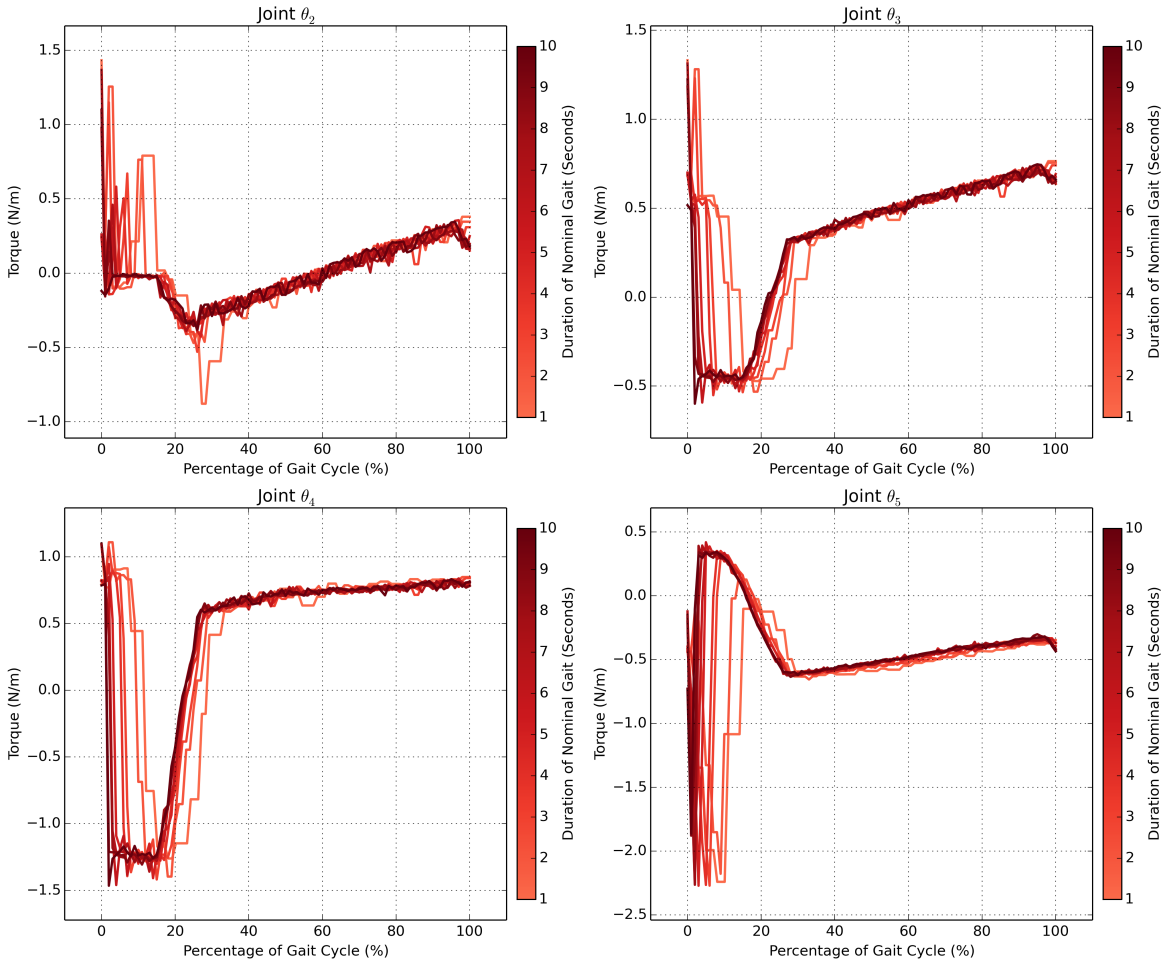

Figure 84: This figure shows the nominal joint torque curves as a function of gait cycle percentage. Each panel shows the curves for one joint. The curves are overlaid to show their similarities when viewed with respect to cycle percentage.

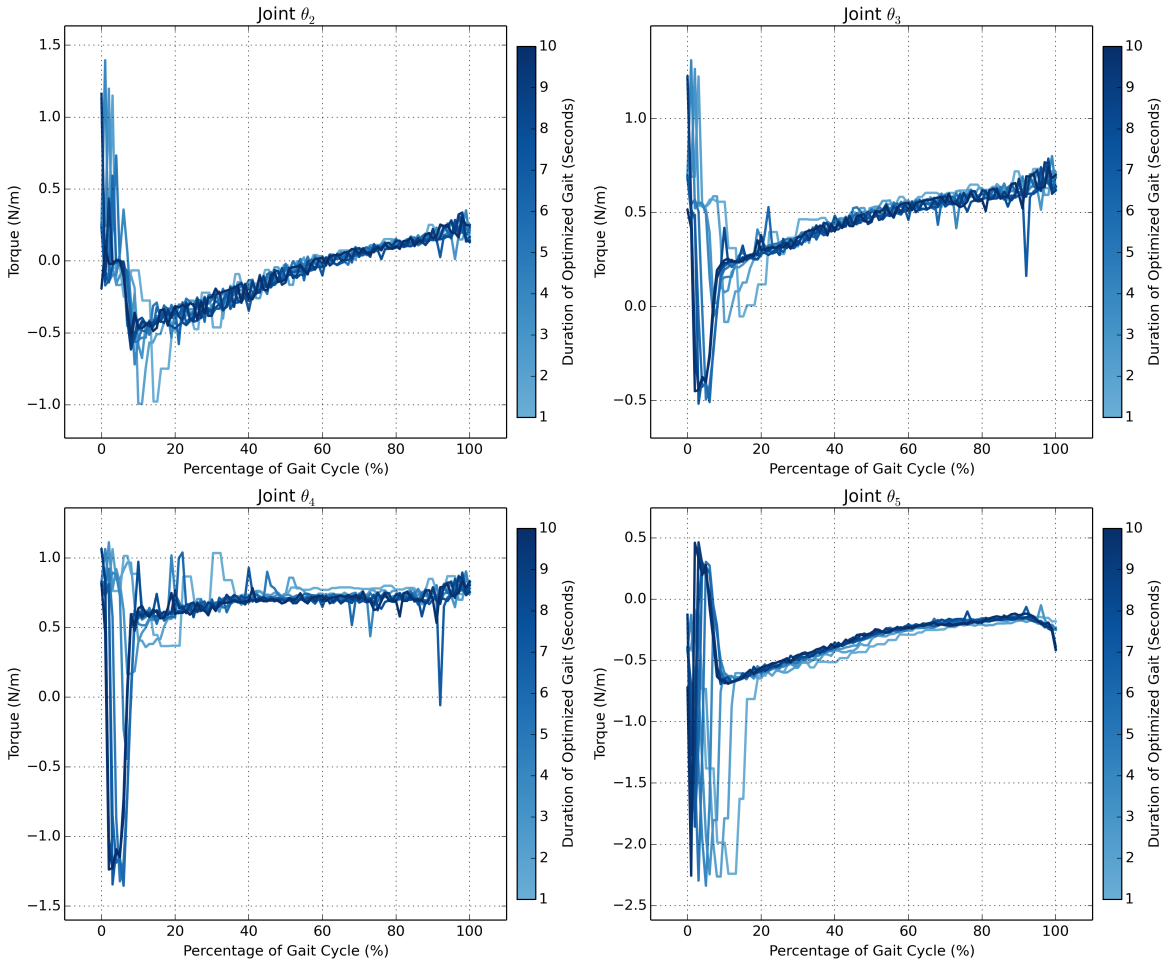

Figure 85: This figure shows the optimal joint torque curves as a function of gait cycle percentage. Each panel shows the curves for one joint. The curves are overlaid to show their similarities when viewed with respect to cycle percentage.

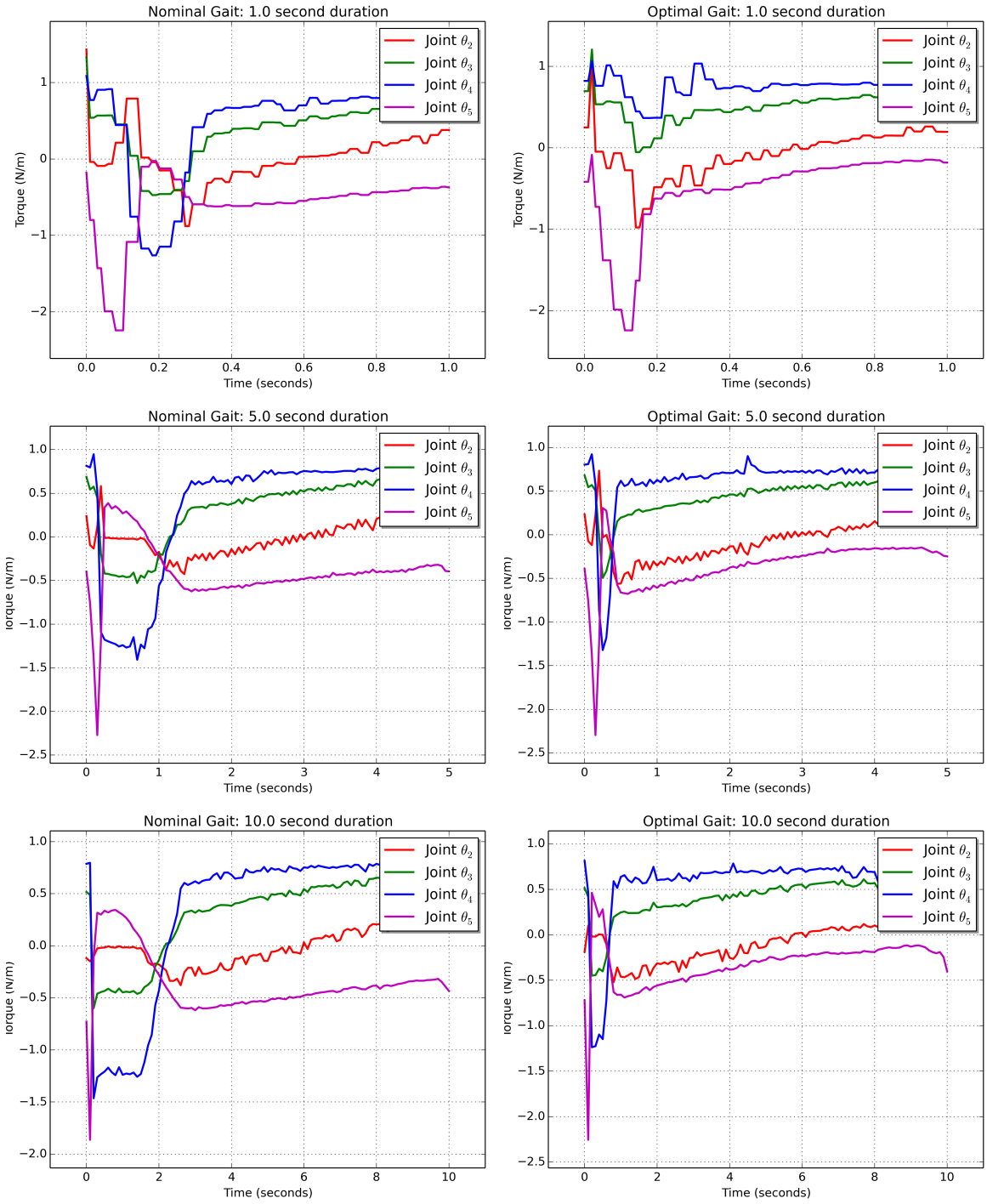

Figure 86: This figure compares the joint torque curves of the nominal and optimal gaits for three gait durations. The optimal curves extend the affine portion to the left, reducing the transient response.

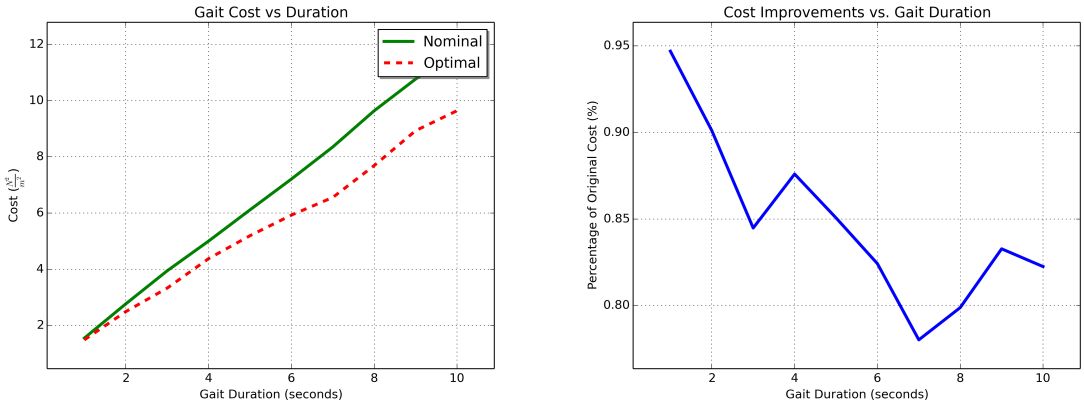

Figure 87: This figure shows cost comparisons between the optimal and nominal gaits as a function of gait duration. The left panel shows this in terms of absolute cost and the right panel shows this in terms of percentage. The right panel percentage is the ratio $\frac{C_o}{C_n}$ for each duration, where C_o is the optimal gait cost, and C_n is the nominal gait cost.

Examining the torque costs of the two gaits as a function of duration in Figure 87 supports the idea that as the effects of the transients diminishes and the pseudo-statics become more dominant, the efficacy of the optimization increases. This is expected as the optimization was formulated with this assumption. It should be noted that the costs presented in Figure 87 are not weighted according to the weight vector presented in Chapter 4.3.2. The figure shows the integral of the costs with the weight vector $w = [1, 1, 1, 1]^T$ in order to show the simple torque cost to the robot. The absolute and percentage difference in costs for the short durations is small, though the optimal gait always outperforms the nominal gait. As the duration increases, the optimal gait costs up to 21.5% less than the nominal gait. Therefore, while the optimal gait does not always outperform the nominal gait by a significant amount, it is always preferable to the nominal gait.

CHAPTER VII

Conclusion

In this thesis, navigation and gaiting which are fundamental issues surrounding mobile humanoid robotics were explored. Chapter II reviewed the platform, which detailed the major components and the design decisions behind the selections. The navigation algorithm was detailed in Chapter IIIand the experimental results were discussed in Chapter V. The algorithm is a lightweight stateless local planner, which performs well when coupled with sensors that can give environmental obstacle data, such as Lidar. The gaiting methodology allowed a humanoid with a large number of degrees-of-freedom to produce an optimized crawl gait using only three parameters. Chapter IV explained the method, while Chapter VI showed the results of the implementation. Here, the main ideas and issues will be briefly discussed, and ideas for improvement proposed.

7.1 Platform

The platform used consisted of a Nao H25 Humanoid, Hokuyo URG-04LX-UG01 Lidar, and a 3D printed mount to join the two. With regards to the navigation experiment, the Nao provided platform locomotion, onboard processing, and goal localization via the camera in the head. The Lidar provided range data about obstacles in the environment. While the range was limited to 5 meters, this was more than adequate as obstacle avoidance is a local schema. What is more problematic is that the Lidar only provides a planar view of the environment, and obstacles above or below this plane are not detectable. This means that the robot could inadvertently collide with them and damage itself or the environment.

For the crawling experiment, the Nao was used without the Lidar and was used only for locomotion. It was a satisfactory humanoid configuration for static crawling experiments, though it was limited to a certain set of crawling gaits. What was most limiting is that the joint motors had insufficient torque to use only the arms or legs to pull/push the robot forward requiring arm-leg cooperation to perform a crawl. Further-

more, the gearboxes in each joint were not designed to take such loads, and frequently broke, requiring the robot to be sent in for maintenance.

7.2 Navigation

Navigation is a core issue for mobile robots. Without the ability to navigate safely in an environment, the robot can damage both itself and its surroundings. Navigation requires a layered approach, with global path planning typically utilizing different schemas to local navigation and obstacle avoidance. GODZILA is a local approach, based on the potential field idea. With the Lidar providing rich environmental data, GODZILA was able to guide the robot around different obstacle configurations. It was also computationally fast and required a small memory footprint, as it does not build a map. While it was able to successfully navigate in the environments shown, potential field approaches have some short-comings. Narrow corridors can present a problem as tuning the algorithm to make them traversable, typically also brings the robot too close to obstacles in other scenarios. Escaping traps is also an issue. Theoretically, this issue can again be overcome by appropriate tuning, but it will cause the performance to suffer in other areas. Ultimately, both of these scenarios require more detailed analysis of the environment to resolve, recover, or avoid these problems and bias parameters or produce intermediate goal points. This is why the navigation problem is typically approached using many algorithms, layered together.

7.3 Crawl Gait

Different modes of locomotion allow the robot to navigate through a larger set of environments. Walking gaits are important to humanoid locomotion, but are not appropriate in all scenarios. Crawling is more stable than walking, and allows a humanoid to traverse beneath low overhangs and small apertures. This extends the navigable area of and environment. Using the Projected Profile approach, the Nao was able to crawl under objects hanging 8 inches above the ground. When standing, the Nao is 23 inches tall. In addition, the crawl gait is able to parameterize a 25 degree-of-freedom system using only 3 parameters. By modifying these parameters, the robot can devise an energy efficient gait, or control the pose of the back throughout the gait. One issue with the current implementation, is that there are no parameters available to control the direction of travel. Therefore, the robot can only crawl forward in an open loop fashion, unable to correct its trajectory if it starts to veer off course. Such a low profile,

also prevents the Nao from using the Lidar to sense the environment. This is because the mount designed for the Lidar, puts it at the waist of the robot which will interfere with the gait.

7.4 Future Work

The presented work provides a foundation to expand the capabilities for mobile humanoid robots. Using this, several improvements are planned to both improve the experimental platform and the algorithms used to be more effective. The addition of the Lidar to the platform greatly expands the obstacle avoidance capabilities of the robot, as well as allowing the use of commonly available Lidar based simultaneous localization and mapping algorithms. An issue with adding such a device is positioning it on the Nao humanoid. Placing the Lidar at hip level is a good choice for navigation as it is one of the parts of the robot that moves the least, but makes locomotion modalities such as crawling problematic. A better choice might be mounting it on the head, as it will not interfere with crawling, and can be oriented using the neck. This could also allow the robot to choose the scan plane during locomotion and might allow obstacle detecting during crawling. The major obstacle to this path is the lack of walking gait parameters to be tuned using the NAOqi API. The effect is, due to the walking gait not knowing about the additional mass and change in inertia, the gait is destabilized. When used with the current Lidar mount, this caused the robot to fall over during navigation. When experimenting with mounting the Lidar on the head, the problem was magnified and, in general, prevented the robot from walking any significant distance. Two options are available to overcome this challenge. This first is to devise a new walking gait, where these unmodeled parameters can be accounted for. A second approach is to command the arms of the robot to counteract the instability. This is potentially an easier approach than devising a new walking gait. It could also produce a new line of research into not only improving walking gaits, but reducing the occurrence of humanoids falling over when encountering unanticipated hazards. An additional platform enhancement would be to use a 3D Lidar. This would allow the robot to detect obstacles in the entire collision space of the robot and make navigation safer. Finally, as the Nao is equipped with cameras in the head, as vision-based approach to obstacle detection is also possible. As the robot is not equipped with a stereo pair, stereo vision techniques are infeasible. Despite this, a set of related algorithms called structure-from-motion are compatible with monocular-camera configurations.

While the benefits of the GODZILA technique have been presented, extensions are needed to form a robust navigation solution in complicated environments. Memoryless algorithms are unable to handle all trap conditions, and a system to detect and recover or avoid these scenarios is necessary. Some form of map is needed so the scenario can be analyzed. A local sliding window map that the robot can localize itself within, could be used to detect if the Nao hasn't moved from a region within a reasonable time and form a new plan or mark the trap as a virtual obstacle. A global map would inform the robot if it hasn't made significant progress toward the goal or facilitate the planning of a path that avoids many traps. Each of these approaches comes at the cost of increased memory footprint and computational complexity.

The current Projected Profile scheme does not reason about the robot turning. This will have to be addressed in order to make it more robust and widely applicable as the ability to correct the robot's path is a necessary feature. The presented implementation of the Projected Profile algorithm was optimized to minimize static joint torques. An extension to this is to optimize for dynamic torques by having the optimization routine choose a parameter set and test the resultant gait with the simulator. Additionally, instead of having a single parameter set utilized by the robot, an entire library of parameters could be produced for use by the robot. This would allow the robot to choose which parameters to use based on other gaiting criteria such as speed, head stability, or trajectory controllability. IMU's have become a common addition to mobile robots and is a standard feature on the Nao humanoids. The IMU on the Nao, could be used to inform the pose of the torso and infer if the robot is on a sloping terrain. The robot could then adjust its posture during gaiting to more aggressively crawl up a hill, or be more cautious when crawling into a pit.

There are many areas using the current platform that can be improved to produce a more robust set of solutions for use with humanoid robots. We look forward to exploring these in the future and are excited for the possibilities in the field.

REFERENCES

[1] National Aeronautics and Space Administration. *Robonaut2*, 2016. `https://robonaut.jsc.nasa.gov/R2` [Accessed: 12/08/2016].

[2] Defense Advanced Research Planning Agency. *DRC Finals*, 2016. `http://archive.darpa.mil/roboticschallenge/` [Accessed: 12/08/2016].

[3] G. Brooks, P. Krishnamurthy, and F. Khorrami. Humanoid robot navigation and obstacle avoidance in unknown environments. In *Proceedings of the Asian Control Conference*, pages 1–6, (Istanbul, Turkey), June 2013.

[4] G. Brooks, P. Krishnamurthy, and F. Khorrami. Low-profile crawling for humanoid motion in tight spaces. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5930–5935, (Hamburg, Germany), Sept. 2015.

[5] David Coleman, Ioan A Şucan, Mark Moll, Kei Okada, and Nikolaus Correll. Experience-based planning with sparse roadmap spanners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 900–905, (Seattle, WA), May 2015.

[6] F. Arambula Coso and M.A. Padilla Castaeda. Autonomous robot navigation using adaptive potential fields. *Mathematical and Computer Modelling*, 40(910):1141 – 1156, 2004.

[7] R. Deits and R. Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 279–286, (Madrid, Spain), Nov. 2014.

[8] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, New York, NY, USA, 2000.

[9] Boston Dynamics. *Atlas - The Agile Anthropomorphic Robot*, 2016. `http://www.bostondynamics.com/robot_Atlas.html` [Accessed: 12/08/2016].

[10] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, Mar. 1997.

[11] Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32(6), 2013.

[12] Juan Pablo Gonzalez and Maxim Likhachev. Search-based planning with provable suboptimality bounds for continuous state spaces. In *Proceedings of the Fourth Annual Symposium on Combinatorial Search*, (Barcelona, Spain), July 2011.

[13] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. Mechatronic design of nao humanoid. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 769–774, (Kobe, Japan), May 2009.

[14] I. Ha, Y. Tamura, H. Asama, J. Han, and D. W. Hong. Development of open humanoid platform darwin-op. In *SICE Annual Conference 2011*, pages 2178–2181, (Tokyo, Japan), Sept. 2011.

[15] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968.

[16] Headquarters, Department of the Army, Washington, DC. *The Warrior Ethos and Soldier Combat Skills, FM 3-21.75*, 2008.

[17] N. Hogan. Impedance control: An approach to manipulation. In *Proceedings of the American Control Conference*, pages 304–313, (San Diego, CA), June 1984.

[18] Hokuyo Automatic Co., Ltd., Osaka HU Building, 2-2-5 Tokiwamachi, Chuo-Ku, Osaka, 540-0028 Japan. *Scanning Laser Range Finder URG-04LX-UG01 (Simple-URG) Specifications*, 2009.

[19] Tony Huang. *RPLIDAR - Slamtec - Leading Service Robot Localization and Navigation Solution Provider*, 2016. `http://www.slamtec.com/en/lidar` [Accessed: 9/19/2016].

[20] Fumiya Iida and Russ Tedrake. Minimalistic control of biped walking in rough terrain. *Autonomous Robots*, 28(3):355–368, 2010.

[21] T. Ishida, Y. Kuroki, and J. Yamaguchi. Mechanical system of a small biped entertainment robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1129–1134 vol.2, (Las Vegas, NV), Oct. 2003.

[22] Luthffi Idzhar Ismail, Syamimi Shamsudin, Hanafiah Yussof, Fazah Akhtar Hanapiah, and Nur Ismarrubie Zahari. *Robot-based Intervention Program for autistic children with Humanoid Robot NAO: Initial response in stereotyped behavior*, volume 41, pages 1441–1447. Elsevier Ltd, 2012.

[23] Inc. Knightscope. *Knightscope*, 2016. `http://knightscope.com/` [Accessed: 12/09/2016].

[24] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1398–1404 vol.2, (Sacramento, CA), Apr. 1991.

[25] M. C. Koval, J. E. King, N. S. Pollard, and S. S. Srinivasa. Robust trajectory selection for rearrangement planning as a multi-armed bandit problem. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2678–2685, (Hamburg, Germany), Sept. 2015.

[26] P. Krishnamurthy and F. Khorrami. Godzila: A low-resource algorithm for path planning in unknown environments. *Journal of Intelligent and Robotic Systems*, 48(3):357–373, 2007.

[27] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[28] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[29] Cai Li, R. Lowe, B. Duran, and T. Ziemke. Humanoids that crawl: Comparing gait performance of icub and nao using a cpg architecture. In *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering*, volume 4, pages 577–582, (Shanghai, China), June 2011.

[30] Cai Li, Robert Lowe, and Tom Ziemke. Crawling posture learning in humanoid robots using a natural-actor-critic cpg architecture, 2013.

[31] Velodyne Lidar. *VLP-16*, 2016. `http://www.slamtec.com/en/lidar` [Accessed: 9/19/2016].

[32] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *Proceedings of the Conference on Advances in Neural Information Processing Systemsi (NIPS)*, Vancouver, Canada, June 2003. MIT Press.

[33] C. Lutz, F. Atmanspacher, A. Hornung, and M. Bennewitz. Nao walking down a ramp autonomously. In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5169–5170, (Vila Moura, Portugal), Oct. 2012.

[34] Lilia Moshkina, Sunghyun Park, Ronald C. Arkin, Jamee K. Lee, and HyunRyong Jung. Tame: Time-varying affective response for humanoid robots. *International Journal of Social Robotics*, 3(3):207–221, 2011.

[35] J. J. Park and B. Kuipers. A smooth control law for graceful motion of differential wheeled mobile robots in 2d environment. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4896–4902, (Shanghai, China), May 2011.

[36] J. J. Park and B. Kuipers. Feedback motion planning via non-holonomic rrt* for mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4035–4040, (Hamburg, Germany), Sept. 2015.

[37] Nicolaus A. Radford, Philip Strawser, Kimberly Hambuchen, Joshua S. Mehling, William K. Verdeyen, A. Stuart Donnan, James Holley, Jairo Sanchez, Vienny Nguyen, Lyndon Bridgwater, Reginald Berka, Robert Ambrose, Mason Myles Markee, N. J. Fraser-Chanpong, Christopher McQuin, John D. Yamokoski, Stephen Hart, Raymond Guo, Adam Parsons, Brian Wightman, Paul Dinh, Barrett Ames, Charles Blakely, Courtney Edmondson, Brett Sommers, Rochelle Rea, Chad Tobler, Heather Bibby, Brice Howard, Lei Niu, Andrew Lee, Michael Conover, Lily Truong, Ryan Reed, David Chesney, Robert Platt, Gwendolyn Johnson, Chien-Liang Fok,

Nicholas Paine, Luis Sentis, Eric Cousineau, Ryan Sinnet, Jordan Lack, Matthew Powell, Benjamin Morris, Aaron Ames, and Jide Akinyode. Valkyrie: Nasa's first bipedal humanoid robot. *Journal of Field Robotics*, 32(3):397–419, 2015.

[38] Ludovic Righetti and Auke Jan Ijspeert. Design methodologies for central pattern generators: an application to crawling humanoids. In *Proceedings of Robotics: Science and Systems*, number BIOROB-CONF-2006-004, pages 191–198, (Philadelphia, PA), 2006.

[39] RoboCup. *RoboCup@Home*, 2016. `http://www.robocupathome.org/` [Accessed: 12/09/2016].

[40] Aldebaran Robotics. *NAO H25*, 2016. `http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html` [Accessed: 11/07/2016].

[41] Savioke. *Savioke*, 2016. `http://www.savioke.com/` [Accessed: 12/09/2016].

[42] Sebastian Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[43] Nikolaos G. Tsagarakis, Giorgio Metta, Giulio Sandini, David Vernon, Ricardo Beira, Francesco Becchi, Ludovic Righetti, José Santos-Victor, Auke Jan Ijspeert, Maria Chiara Carrozza, and Darwin G. Caldwell. icub: the design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics*, 21:1151–1175, 2007.

[44] M. Vukobratovic and D. Juricic. Contribution to the synthesis of biped gait. *IEEE Transactions on Biomedical Engineering*, BME-16(1):1–6, Jan 1969.

[45] Honda Worldwide. *ASIMO by Honda*, 2016. `http://asimo.honda.com` [Accessed: 12/08/2016].

[46] Jia-chi Wu and Zoran Popović. Terrain-adaptive bipedal locomotion control. *ACM Trans. Graph.*, 29(4):72:1–72:10, July 2010.

[47] V. Zatsiorsky and B. Prilutsky. *Biomechanics of Skeletal Muscles*. Human Kinetics 10%.