March 2022

Humanoid Robot Motion Control for Ramps and Stairs

Tommy Truong
University of South Florida

Humanoid Robot Motion Control for Ramps and Stairs

by

Tommy Truong

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Alfredo Weitzenfeld, Ph.D.
Les Piegl, Ph.D.
Yu Sun, Ph.D.

Date of Approval:
March 8, 2022

Keywords: Inverse Kinematics, Gait Planning, Computer Vision, Humanoid Robot

Dedication

I dedicate this to myself as an indicator of what I can do and be the first of many in my family to obtain a Master degree. I also dedicate this to the professors at the computer science department who show me that a master's degree was an option I could do

Acknowledgments

I would like to acknowledge Professor Weitzenfeld for providing guidance and insight about the Darwin-Op. I would like to acknowledge Carlo Canezo for his help during the initial phase of this project. I would like to acknowledge my commitee for their support of this project as well.

Table of Contents

List of Tables iii

List of Figures iv

Abstract v

Chapter 1: Introduction 1
- 1.1 Introduction and Rationale 1
- 1.2 Problem Statement 1
- 1.3 Problem Solution 2
- 1.4 Thesis Outline 3

Chapter 2: Literature Review 4
- 2.1 Object Detection 4
  - 2.1.1 Color Detection 4
  - 2.1.2 Neural Networks 5
  - 2.1.3 Inertial Measuring Unit 5
- 2.2 Bipedal Balancing 6
  - 2.2.1 Dynamic and Static Systems 6
  - 2.2.2 Generating Gaits 6
- 2.3 Remarks 7

Chapter 3: Darwin-Op Model and Kinematics 8
- 3.1 Darwin-Op Model 8
  - 3.1.1 Hardware Specification 9
- 3.2 Darwin-Op Measurements 10
  - 3.2.1 Joint Angle Limits 12
- 3.3 Darwin-Op Kinematics 14
  - 3.3.1 DH Parameters 14
    - 3.3.1.1 Head DH Parameters 15
    - 3.3.1.2 Arm DH Parameters 15
    - 3.3.1.3 Leg DH Parameters 16
  - 3.3.2 Forward Kinematics 16
  - 3.3.3 Inverse Kinematics 20
    - 3.3.3.1 Solution for θ_{13} 21
    - 3.3.3.2 Solution for θ_{7} 21
    - 3.3.3.3 Solution for θ_{11} 22
    - 3.3.3.4 Solution for θ_{15} 22

3.3.3.5 Solution for θ_9 . . . 23
3.3.3.6 Solution for θ_{17} . . . 23

Chapter 4: Proposed Methodology . . . 24
4.1 Assumptions . . . 24
4.2 Vision Control . . . 24
4.3 Gait Planning . . . 25
4.3.1 Zero Moment Point . . . 25
4.3.2 Cart-Table Model . . . 27
4.3.3 Trajectory Planning . . . 29

Chapter 5: Simulation . . . 30
5.1 Darwin-Op Model . . . 30
5.2 Environments . . . 31
5.3 Programming . . . 31
5.3.1 Control System Flowchart . . . 32
5.3.2 Controller Code . . . 34
5.3.3 Color Detection . . . 34
5.3.4 Gait Planning . . . 35

Chapter 6: Experiment and Results . . . 38
6.1 Experiment . . . 38
6.1.1 Ramp Experiment . . . 38
6.1.2 Stair Experiment . . . 39
6.2 Results . . . 39
6.2.1 Ramp Results . . . 39
6.2.2 Stair Results . . . 40
6.3 Remarks . . . 40

Chapter 7: Conclusion and Future Work . . . 42
7.1 Conclusion . . . 42
7.2 Future Work . . . 42
7.2.1 Walking Refinement . . . 42
7.2.2 Object Detection . . . 43
7.2.3 Ladder Climbing . . . 43

References . . . 44

Appendix A: Nomenclature . . . 46

Appendix B: Copyright Permissions . . . 47

List of Tables

Table 3.1 Head Lengths 11

Table 3.2 Torso Lengths 11

Table 3.3 Arm Lengths 11

Table 3.4 Leg Lengths 11

Table 3.5 Foot Lengths 12

Table 3.6 Head Joint Limits 13

Table 3.7 Right Arm Joint Limits 13

Table 3.8 Left Arm Joint Limits 13

Table 3.9 Right Leg Joint Limits 13

Table 3.10 Left Leg Joint Limits 14

Table 3.11 Head DH Parameters 15

Table 3.12 Arm DH Parameters 15

Table 3.13 Leg DH Parameters 16

Table 6.1 Ramp Experiment 40

Table 6.2 Stairs Experiment 40

List of Figures

Figure 3.1 Darwin OP Model 8

Figure 3.2 Darwin-Op Lengths Definitions 10

Figure 3.3 Darwin-Op Motor Positions 12

Figure 3.4 Head Coordinate Frames 15

Figure 3.5 Arm Coordinate Frames 16

Figure 3.6 Leg Coordinate Frames 17

Figure 4.1 Coronal View of the Cart-Table Model 27

Figure 4.2 Sagittal View of the Cart-Table Model 28

Figure 4.3 Stair Trajectory Planning 29

Figure 5.1 Simulation of Darwin-Op 31

Figure 5.2 Stair Environment 32

Figure 5.3 Ramp Environment 32

Figure 5.4 Control System Flowchart 33

Figure 5.5 Color Detection Algorithm 34

Figure 5.6 Sample Image Used for Color Parsing 35

Figure 5.7 Ladder Parsed Out of Image 35

Figure 5.8 Color Detection Code Snippet 36

Figure 5.9 Trajectory Planning Diagram 36

Figure 5.10 Fallen Function 37

Abstract

Humanoid robot research and development have been an ongoing effort since the 1900s and can be broken down to two problems. A mechanical problem, getting a humanoid robot to move human-like or a software problem, getting a humanoid robot to behave human-like. These problems of moving and behaving human-like can be often solved using control theory as research advances. For the premise of this research, we explore how to balance and walk on non-flat terrain for the humanoid robot Darwin-Op. Since the focus was on the control theory, the vision control to detect the non-flat terrain was a side objective. The vision control was implemented by detecting if a non-flat terrain such as a ramp or stairs had a specific color. Once the object had been identify, some computer vision techniques can be applied to get the distance and orientation from the object. While the vision control isn't as robust, it was enough to guide the robot to the obstacle. For the control theory aspect of the research, the robot was able to walk up stairs and ramps by using zero moment point trajectory planning with a cart-table model for the center of mass. After running some experiment, it was concluded that smooth surfaces such as ramp work the best compared to stairs. With the edges on a stairs, there were problems getting a proper step landing for step heights higher than the robots ankle. But it shows that autonomous movement on non-flat surfaces is viable.

Chapter 1: Introduction

1.1 Introduction and Rationale

The very first humanoid robot was built in the 1400s by Leonardo Da Vinci; The robot was an armored knight that could do simple arm movement and open its jaws. It wasn't until the 1900s that humanoid robots started to rapidly develop. With the technologies introduced during the industrial age, humanoid robots were able to take full advantage of computers to mimic the actions of humans. Such in 1939, with the creation of the humanoid Elektro, it could move using wheels, play recorded speech, and move its head and arms. The next big humanoid was in 1973, Wabot-1 was a full- scale humanoid robot that was able to walk on two legs and communicate with a person. Since then, there have been many breakthroughs in humanoid designs and control theory.

Breakthrough in humanoid control theory allows for humanoid movement to look more human-like and fluid. Such control theory that makes modern humanoid walk fluid is gait planning. Gait planning is essentially a cyclic motion pattern to produce locomotion in legged robots [9]. To make humanoids more human-like, they must be able to walk under all sorts of conditions and terrains such as ramps or stairs. While the challenge of having a bipedal robot walk on ramps and stairs has been solved, the rationale of this thesis is to implement such capabilities for the humanoid robot Darwin-Op.

1.2 Problem Statement

The ability to balance and walk on uneven terrain is something we as humans do intuitively, and sometimes on command. For example, we don't actively think about going up

or down a set of stairs, we can instinctual move our legs to the next point it needs to be. But when someone uses the phrase "Watch your step", then we actively think about how we move up or down by seeing where our next step will be, or if sight is impaired, then we slowly move to feel out the next stepping point.

The challenge with humanoid robots is that they don't have the native ability to walk a set of stairs without being instructed to do so. When teaching a robot how to walk, we teach it on a flat surface to make it easier, but now that scope has to be expanded to non-flat surfaces. The problem statement can be broken down into two categories:

1. How to recognize an uneven terrain

2. How to walk on non-level terrains

While the first category is a very open-ended problem, there are many methods in doing so such as color base recognition or object-based recognition. The main focal point of this thesis will be on the 2nd category and how it gets implemented and solved for walking on surfaces such as ramps, and stairs.

1.3 Problem Solution

The proposed solution revolves around a simple obstacle detection with a robust gait manager to handle known non-level surfaces such as ramps and stairs. The result of the obstacle detection is fed into the controller to know when and where the obstacle is at. Once the obstacle has been found, the robot will use the default walk mechanism to get there and switch to the implemented gait manager to walk up the obstacle.

For the proposed implemented gait manager, the gait manager will use zero moment point as a way to keep the robot stable while in locomotion. Zero Moment Point (ZMP) is the point on the ground at which the total horizontal inertia and gravity forces equal 0 and is a subset of the inverted pendulum problem [1]; An inverted pendulum is a pendulum that has its center of mass above its pivot point. Using a cart-table model, we can assume

the upper-body of a robot to be a "cart" located at a robot Center of Mass (CoM)[10]. CoM would describe the movement of the cart while ZMP would describe the position of the supporting foot on the ground. These two concepts can be combined to make a gait state of stand, left leg swing, finish left leg swing, right leg swing, finish right leg swing, and then back to walk ready pose.

1.4 Thesis Outline

The thesis is structured as follows: Chapter 2 discusses related work and literature reviews for the proposed solution. Chapter 3 goes into detail about the design of the Darwin-Op and how to formulate the Darwin-Op kinematics. Chapter 4 talks about the simulation used for the thesis and the control mechanism for moving the robot. Chapter 5 discusses the testing and result of the thesis while Chapter 6 highlight any resulting thought and conclusion about the thesis.

Chapter 2: Literature Review

This chapter reviews the research that supports the topic of the thesis. The literature review will discuss all the different techniques for self-balancing during locomotion, detecting obstacles, and path planning for locomotion.

2.1 Object Detection

Object detection is a computer vision technique to identify what and where an object is in an image or video. Object detection is leveraged in many fields of robotics such as humanoid-robot, self-driving cars, and medical robots. While there are many implementations to tackling the problem of object detection, this thesis will only cover 3 different techniques. Section 2.1.1 covers a rudimentary approach while section 2.1.2 covers a more modern approach to the problem. Section 2.1.3 covers a non-conventional approach that doesn't leverage computer vision but rather a physical approach.

2.1.1 Color Detection

The simplest approach to finding an object in an image is to assign the object a unique color so whenever an image sees that color then it is assumed to be the object. While the simplest, it is not robust as other objects could potentially share the same color. In paper [8], this color detection technique was leveraged to track a red ball and follow it. To further enhance the color detection technique, a red ball was decided to be found if there was a blob red or essentially a large gathering of red pixels. From there, the red ball location was found by using trigonometry between the camera and the ball.

2.1.2 Neural Networks

The modern approach to finding an object is with the use of artificial intelligence (AI). Region-based Conventional Neural Networks or R-CNN is one of the more common AI techniques for object-recognition tasks. R-CNN is a group of machine learning models that takes an input image and produces a set of bounding boxes as output where each bounding box contains an object and category of the object. Eventually, with enough training, an R-CNN can tell what the object is and where it is. In paper [5], the autonomous vehicle tracks its target using a Yolo-v3 model. The Yolo-v3 is another type of R-CNN and a one-stage detection model which is known for its speed and accuracy. By using a known data set to train the model, the vehicle was able to track the target and move towards it while avoiding obstacles.

2.1.3 Inertial Measuring Unit

An unconventional approach to finding an object is with an inertial measuring unit (IMU). In paper [11], it discusses the idea of localization of a person in an indoor environment through the use of a foot-mounted IMU. Through the combination of a Pedestrian Dead Reckoning (PDR) algorithm and an Extended Kalman Filter (EKF). The use of the PDR allows to make the step detection more reliable with the EKF providing error estimations at the IMU's update rate which was 100 Hz.

A person could be localized on a ramp by observing two parameters: pitch angle (ψ) of the IMU and rise (δz) between two consecutive steps since a ramp can be generally described as a leveled terrain by its slope. So using these concepts, a ramp can be detected using the described function:

$$\begin{cases} 1 & \psi(k) \cdot \delta z(k) < Th \\ 0 & Otherwise \end{cases}$$

where k is the index of the step detected and Th is a threshold value. In the paper, the threshold selected was 9 degrees · cm meaning any inclinations larger than 2 degrees would be seen as a ramp. The only caveat to this type of detection is that a robot would only know when it's on a ramp and not where it is ahead of time.

2.2 Bipedal Balancing

2.2.1 Dynamic and Static Systems

For a legged robot system, the gaits are determined dynamically or statically depending on the speed at which the system has to go. For slow-moving systems, a static system can be used but in the case where calculations need to be made on the spot or offline a dynamic system is used. In paper [3], a comparison was done with different variations of gaits and their effects on gait length and swings. It was found that the stability of the system could not reach stability depending on execution time rather than the length of the gaits.

2.2.2 Generating Gaits

The most important thing about gaits is how the trajectories are generated, in the case of legged robots, paper [9] discusses 3 methods for generating motions. One method is making a reactive system that is good for the desired motion but not so good for intuition to do so. The other method is to plan the individual motions for the legs and feet of a robot, it ensures the stability of the overall motion but involves complex constraints such as surface contact and conservation of momentum. The last and most popular method is to specify the gaits that a robot may use, essentially planning for more broad gaits that are more applicable to different uses and can be switched relatively easily.

2.3 Remarks

The literature review presents a very limited scope and understanding of how object detection and bipedal balancing work. The mentioned papers were selected for review with how they help support this thesis specifically. Some of the provided reviews cover the technique that was needed to develop the proposed and future solution of this thesis.

Chapter 3: Darwin-Op Model and Kinematics

This chapter goes into the detail of the humanoid robot the thesis is based on, more so specifically, it covers all the essential information needed for gait planning. In section 3.1, it goes over the specification of the robots and all the define lengths needed for section 3.3. Section 3.3 goes into how the inverse and forward kinematics is derived for the robot.

3.1 Darwin-Op Model

Figure 3.1: Darwin OP Model

The Darwin-Op as shown in fig 3.1 is a miniature-humanoid robot developed by Robotis in collaboration with Virginia Tech, Purdue University, and the University of Pennsylvania; Darwin-Op is short for Dynamic Anthropomorphic Robot with Intelligence - Open Platform. A newly packaged Darwin-Op2 comes with the following:

- Fully Assemble Robot
- Charging Cable and Power Supply

- Quick Start Manual
- Training Tools
- Spare parts

The reason for choosing this robot is because it is available at the lab and in the simulation software which will be discussed in Chapter 5. The robot is fully programmable and is suited for navigation, SLAM, AI, and human-robot interaction development.

3.1.1 Hardware Specification

The following are the hardware specification of the robot:

- Built-in PC:
  - Intel Atom N2600 @1.6 GHz dual core
  - 4GB DDR3 Ram
  - 32GB mSATA
- Management controller (CM-740): ARM CortexM3 STM32F103RE 72 MHz
- 20 actuator modules (Dynamixel MX-28T):
  - 6 DOF leg x2
  - 3 DOF arm x2
  - 2 DOF neck
- 1Mbps high-speed Dynamixel bus
- 1800mAh LIPO Battery
- 3-axis gyro
- 3-axis accelerometer

- button x3
- detection microphone x2

3.2 Darwin-Op Measurements

This section defines the various lengths needed for the Denavit-Hartenberg (DH) parameters discussed in Section 3.3; The DH parameter is a parameter convention used to assign reference frames of a robot manipulator. The following tables will reference Fig 3.2 for the length definitions [15]. [1]

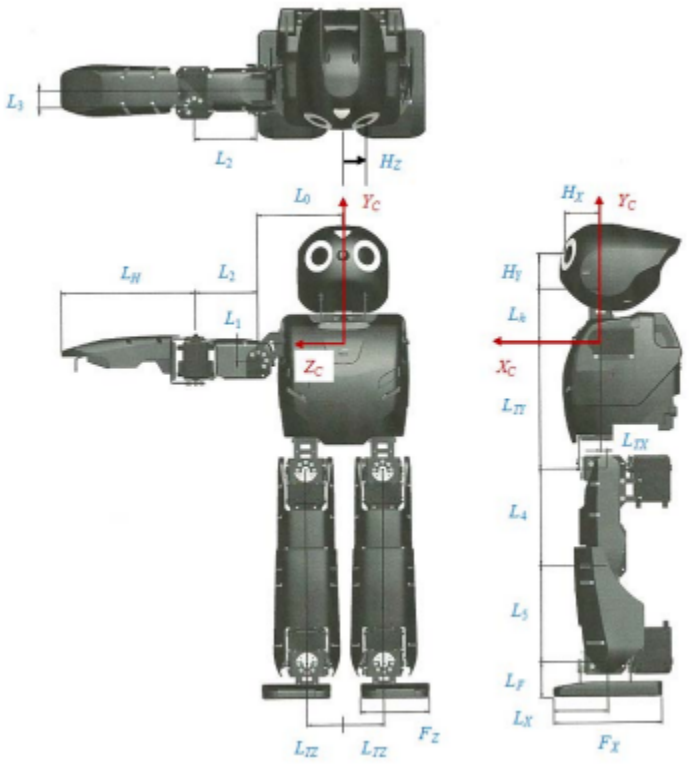


Figure 3.2: Darwin-Op Lengths Definitions

[1] Section 3.2 and 3.3 was published in IDETC/CIE 2012. Permission is included in Appendix B

Table 3.1: Head Lengths

Length	Value (mm)
H_X	32.2
H_Y	34.4
H_Z	22.5

Table 3.2: Torso Lengths

Length	Value (mm)
L_h	50.5
L_0	82.0
L_{TX}	5.0
L_{TY}	122.2
L_{TZ}	37.0

Table 3.3: Arm Lengths

Length	Value (mm)
L_1	16.0
L_2	60.0
L_3	16.0
L_H	129.0

Table 3.4: Leg Lengths

Length	Value (mm)
L_4	93.0
L_5	93.0
L_F	33.5

Table 3.5: Foot Lengths

Length	Value (mm)
F_X	104.0
F_Y	15.0 (10.0 front)
F_Z	66.0
L_X	52.0
L_Z	23.0

3.2.1 Joint Angle Limits

The following tables represent the joint angle limits for the DH parameters and can be found referencing the position of the motors found on Fig 3.3 [15].

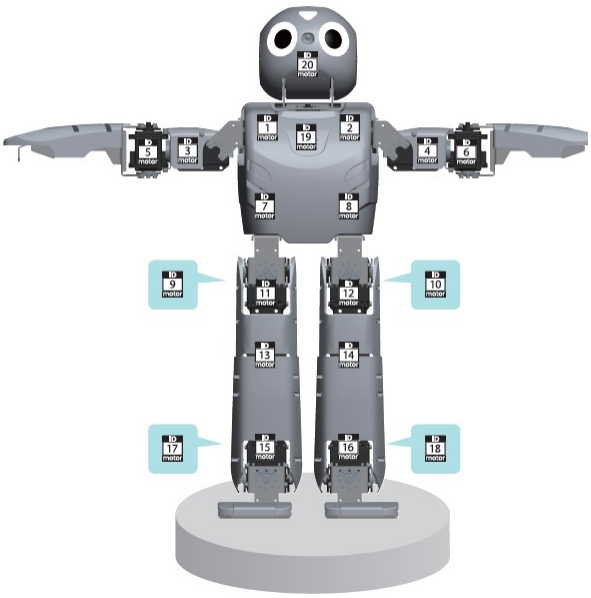


Figure 3.3: Darwin-Op Motor Positions

Table 3.6: Head Joint Limits

Joint i	Joint name	Axis	Θ_{iMIN}	Θ_{iMAX}
19	Head Pan	Z_1	-150	150
20	Head Tilt	Z_2	-60	30

Table 3.7: Right Arm Joint Limits

Joint i	Joint name	Axis	Θ_{iMIN}	Θ_{iMAX}
1	Shoulder Pitch	Z_1	-250	250
3	Shoulder Roll	Z_2	-100	100
5	Elbow	Z_3	0	160

Table 3.8: Left Arm Joint Limits

Joint i	Joint name	Axis	Θ_{iMIN}	Θ_{iMAX}
2	Shoulder Pitch	Z_1	-250	250
4	Shoulder Roll	Z_2	-100	100
6	Elbow	Z_3	0	-160

Table 3.9: Right Leg Joint Limits

Joint i	Joint name	Axis	Θ_{iMIN}	Θ_{iMAX}
7	Hip Yaw	Z_1	-150	45
11	Hip Roll	Z_2	0	60
9	Hip Pitch	Z_3	-100	30
13	Knee	Z_4	0	130
17	Ankle Pitch	Z_5	-60	60
15	Ankle Roll	Z_6	-30	60

Table 3.10: Left Leg Joint Limits

Joint i	Joint name	Axis	Θ_{iMIN}	Θ_{iMAX}
8	Hip Yaw	Z_1	-45	150
12	Hip Roll	Z_2	-60	0
10	Hip Pitch	Z_3	-30	100
14	Knee	Z_4	-130	0
18	Ankle Pitch	Z_5	-60	60
16	Ankle Roll	Z_6	-30	60

3.3 Darwin-Op Kinematics

This section establishes the kinematics used for gait planning in Chapter 4. Section 3.3.1 will cover all three serial links that are movable (head, arm, leg), but Section 3.3.2 and 3.3.3 will only cover the kinematics involved for the legs.

3.3.1 DH Parameters

The Cartesian reference frame for all the DH definitions is with respect to the x and z-axis in a zero-angle pose. The DH parameters also reference the measurement length definitions and joint angle limits established from section 3.2

For the DH parameters shown below, all the angles will be in degrees, and the following are the explanations of the column names:

- α_{i-1} is the angle between Z_i and Z_{i+1} along X_i
- a_{i-1} is the distance between Z_i and Z_{i+1} along X_i
- d_i is the distance between X_i and X_{i+1} along Z_i
- θ_i is the angle between X_i and X_{i+1} along Z_i
- i is the joint connecting link $i-1$ to link i

3.3.1.1 Head DH Parameters

The head only has 2 DOF as seen on Fig 3.4, therefore the DH parameters can be formed such as shown on Table 3.11

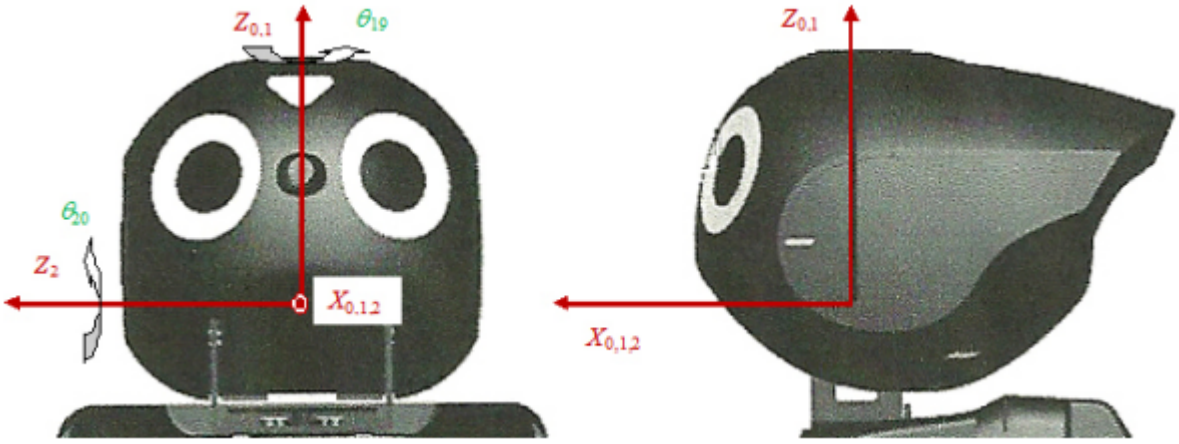


Figure 3.4: Head Coordinate Frames

Table 3.11: Head DH Parameters

i	α_{i-1}	a_{i-1}	d_i	Θ_i
1	0	0	0	θ_{19}
2	90	0	0	θ_{20}

3.3.1.2 Arm DH Parameters

The arm only has 3 DOF as seen on Fig 3.5, therefore the DH parameters can be formed such as shown on Table 3.12

Table 3.12: Arm DH Parameters

i	α_{i-1}	a_{i-1}	d_i	Θ_i
1	0	0	0	θ_1
2	-90	0	0	θ_3 - 90
3	90	L_2	0	θ_5

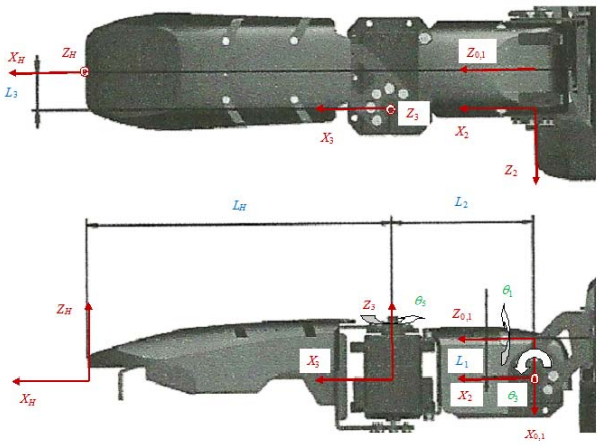


Figure 3.5: Arm Coordinate Frames

3.3.1.3 Leg DH Parameters

The leg only has 6 DOF as seen on Fig 3.4, therefore the DH parameters can be formed such as shown on Table 3.13

Table 3.13: Leg DH Parameters

i	α_{i-1}	a_{i-1}	d_i	Θ_i
1	0	0	0	θ_7
2	90	0	0	$\theta_{11} + 90$
3	90	0	0	θ_9
4	0	L_4	0	θ_{13}
5	0	L_5	0	θ_{17}
6	-90	0	0	θ_{15}

3.3.2 Forward Kinematics

Forward kinematic is essentially solving the problem of given a set of angles, what is the position of the end-effector, in this case the end position of the leg. Forward kinematics is solve by assigning a 4x4 homogeneous transformation matrix to a frame such that:

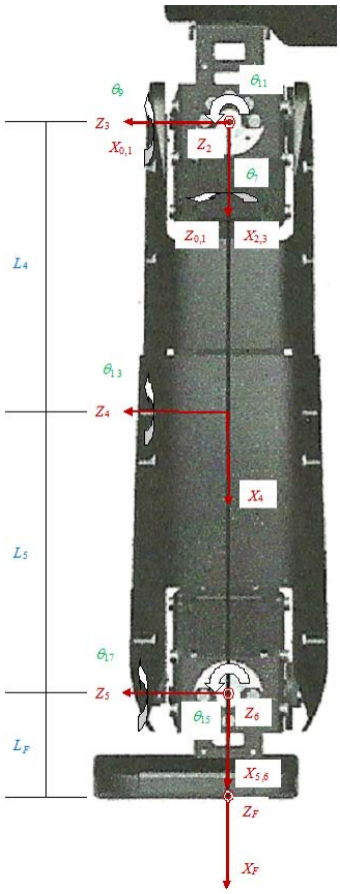


Figure 3.6: Leg Coordinate Frames

$$\left[T_i^{i-1}\right] = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_i - 1 \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & -c\alpha_{i-1} & d_i c\alpha_i - 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where c = cosine and s = sine. The upper left 3x3 matrix describes the rotation matrix and the upper right 3x1 describes the position matrix. The values of θ_i, α_i, and d_i are obtained directly from the DH table. So by following the legs DH table on Table 3.13 we can then

obtain the 6 transformation matrix:

$$\left[T_1^0\right] = \begin{bmatrix} c_7 & -s_7 & 0 & 0 \\ s_7 & c_7 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$\left[T_2^1\right] = \begin{bmatrix} -s_1 1 & -c_1 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ c_1 1 & -s_1 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$\left[T_3^2\right] = \begin{bmatrix} c_9 & -s_9 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_9 & c_9 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$\left[T_4^3\right] = \begin{bmatrix} c_{13} & -s_{13} & 0 & L_4 \\ s_{13} & c_{13} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$\left[T_5^4\right] = \begin{bmatrix} c_{17} & -s_{17} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_9 & c_9 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$\left[T_6^5\right] = \begin{bmatrix} c_{15} & -s_{15} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_{15} & -c_{15} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

therefore the end effector can be describe as so:

$$\left[T_6^0\right] = \left[T_1^0\right]\left[T_2^1\right]\left[T_3^2\right]\left[T_4^3\right]\left[T_5^4\right]\left[T_6^5\right] \tag{3.8}$$

$$\left[T_6^0\right] = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

where:

$$m_{11} = (s_7 s(\theta_9 + \theta_{13} + \theta_{17}) - c_7 s_{11} c(\theta_9 + \theta_{13} + \theta_{17}))c_{15} - c_7 c_{11} s_{15} \tag{3.10}$$

$$m_{21} = (-c_7 s(\theta_9 + \theta_{13} + \theta_{17}) - s_7 s_{11} c(\theta_9 + \theta_{13} + \theta_{17}))c_{15} - s_7 c_{11} s_{15} \tag{3.11}$$

$$m_{31} = -s_{11} s_{15} + c_{11} c_{15} c(\theta_9 + \theta_{13} + \theta_{17}) \tag{3.12}$$

$$m_{12} = -(s_7 s(\theta_9 + \theta_{13} + \theta_{17}) - c_7 s_{11} c(\theta_9 + \theta_{13} + \theta_{17}))c_{15} - c_7 c_{11} s_{15} \tag{3.13}$$

$$m_{22} = -(-c_7 s(\theta_9 + \theta_{13} + \theta_{17}) - s_7 s_{11} c(\theta_9 + \theta_{13} + \theta_{17})) - s_7 c_{11} c_{15} \tag{3.14}$$

$$m_{32} = -s_{11} c_{15} - c_{11} s_{15} c(\theta_9 + \theta_{13} + \theta_{17}) \tag{3.15}$$

$$m_{13} = s_7 c(\theta_9 + \theta_{13} + \theta_{17}) + c_7 s_{11} s(\theta_9 + \theta_{13} + \theta_{17}) \tag{3.16}$$

$$m_{23} = -c_7 c(\theta_9 + \theta_{13} + \theta_{17}) + s_7 s_{11} s(\theta_9 + \theta_{13} + \theta_{17}) \tag{3.17}$$

$$m_{33} = -cs(\theta_9 + \theta_{13} + \theta_{17}) \tag{3.18}$$

$$m_{14} = (L_4 s_9 + L_5 s(\theta_9 + \theta_{13}))s_7 - (L_4 c_9 + L_5 c(\theta_9 + \theta_{13}))c_7 s_{11} \tag{3.19}$$

$$m_{24} = -(L_4 s_9 + L_5 s(\theta_9 + \theta_{13}))c_7 - (L_4 c_9 + L_5 c(\theta_9 + \theta_{13}))s_7 s_{11} \tag{3.20}$$

$$m_{34} = (L_4 c_9 + L_5 c(\theta_9 + \theta_{13}))c_{11} \tag{3.21}$$

3.3.3 Inverse Kinematics

Inverse kinematics solves the set of the angles needed to achieve a given end-effector position. Inverse kinematics is more complicated than forward kinematics because there are multiple non-trivial solutions. In the case of the leg, the inverse kinematics can be solved by first solving for the angle of the knee or θ_{13} [13].

Before solving for θ_{13}, some equations must be established. Given equation (3.8), then

$$\begin{bmatrix} T_1^0 \end{bmatrix}^{-1} \begin{bmatrix} T_6^0 \end{bmatrix} = \begin{bmatrix} T_2^1 \end{bmatrix} \begin{bmatrix} T_3^2 \end{bmatrix} \begin{bmatrix} T_4^3 \end{bmatrix} \begin{bmatrix} T_5^4 \end{bmatrix} \begin{bmatrix} T_6^5 \end{bmatrix} \tag{3.22}$$

$$\begin{bmatrix} T_2^1 \end{bmatrix} \begin{bmatrix} T_1^0 \end{bmatrix}^{-1} \begin{bmatrix} T_6^0 \end{bmatrix} = \begin{bmatrix} T_3^2 \end{bmatrix} \begin{bmatrix} T_4^3 \end{bmatrix} \begin{bmatrix} T_5^4 \end{bmatrix} \begin{bmatrix} T_6^5 \end{bmatrix} \tag{3.23}$$

3.3.3.1 Solution for θ_{13}

To solve θ_{13}, we assume that the knee can be bent forward or backward meaning θ_{13} can be positive or negative. We also assume that joints 3, 4, and 5 form a spatial triangle together based on the positions of the joints. This means that θ_{13} can be solved using the law of cosines [13]:

$$c^2 = a^2 + b^2 - 2ab \cdot cos(C) \tag{3.24}$$

where a and b describes the length of joint 4 and 5, and c^2 is the distance between joint 5 and joint 3 at any given time. Then we can plug into the equation (3.26) to get:

$$c^2 = L_4^2 + L_5^2 - 2L_4L_5 \cdot cos(\pi - \theta_{13}) \tag{3.25}$$

which can be rearrange to solve for θ_{13} such as so:

$$\theta_{13} = \pm \arccos(\frac{c^2 - L_4^2 + L_5^2}{2L_4L_5}) \tag{3.26}$$

3.3.3.2 Solution for θ_7

θ_7 can be solved using equation (3.23) by making these assertions:

$$M_3 = \begin{bmatrix} T_2^1 \end{bmatrix} \begin{bmatrix} T_1^0 \end{bmatrix}^{-1} \begin{bmatrix} T_6^0 \end{bmatrix} \tag{3.27}$$

$$N_3 = \begin{bmatrix} T_3^2 \end{bmatrix} \begin{bmatrix} T_4^3 \end{bmatrix} \begin{bmatrix} T_5^4 \end{bmatrix} \begin{bmatrix} T_6^5 \end{bmatrix} \tag{3.28}$$

then we can derived two possible solutions for θ_7 by doing this:

$$M_3(2,3) = N_3(2,3) \tag{3.29}$$

$$M_3(2,4) = N_3(2,4) \tag{3.30}$$

3.3.3.3 Solution for θ_{11}

Similarly for solving θ_{11}, we can make the assertion from equation (3.22) that:

$$M_2 = \left[T_1^0\right]^{-1} \left[T_6^0\right] \tag{3.31}$$

$$N_2 = \left[T_2^1\right] \left[T_3^2\right] \left[T_4^3\right] \left[T_5^4\right] \left[T_6^5\right] \tag{3.32}$$

therefore by setting up this system of equations then

$$M_2(1,4) = N_2(1,4) \tag{3.33}$$

$$M_3(2,2) = N_3(2,2) \tag{3.34}$$

and because $L_5 \cos\theta_9 + \theta_{13} + L_4 \cos\theta_9$ can be positive or negative, then there are two possible solutions for θ_{11}

3.3.3.4 Solution for θ_{15}

θ_{15} can be solved by solving the two equations:

$$M_3(2,1) = N_3(2,1) \tag{3.35}$$

$$M_3(2,2) = N_3(2,2) \tag{3.36}$$

3.3.3.5 Solution for θ_9

Since θ_7 and θ_{11} were previously solved, then $M_3(1,4)$ and $M_3(3,4)$ are constants leaving θ_9 possible to solve.

3.3.3.6 Solution for θ_{17}

θ_{17} can be solved by solving the two equations:

$$M_4(1,3) = N_4(1,3) \tag{3.37}$$

$$M_4(2,3) = N_3(2,3) \tag{3.38}$$

While there are 8 possible different combinations of angles to obtain, because of the joint angle limits of the Darwin-Op, we can limit the 8 different solutions more precisely. For example, in θ_{13}, since the range only goes from 0 to 130 degrees, then it removes 50 degrees out of the 180 degrees possible in the arccos domain.

Chapter 4: Proposed Methodology

As stated in Section 1.2, the problem of autonomous balance locomotion can be broken into recognizing areas that are non-level and how to walk through them. This chapter goes into detail about the proposed methodology and some basic assumptions made for the research. Section 2.1 discusses how non-level terrains can be recognized using basic computer vision techniques. Section 4.3 discusses how to go over non-level terrains using zero moment point (ZMP) trajectory planning on a cart-table model.

4.1 Assumptions

Since the research work is all simulated, some general things are assumed:

- Noise level: 0-10%
- All objects/floor are non-slip meaning no slippage
- No wind meaning no air resistance
- No outside objects will be interfering with the robot during simulation
- The environment is known to the robot beforehand
- objects are known beforehand
- One object per environment

4.2 Vision Control

Training a neural network model usually takes a long time because the longer the training then the better the accuracy of the model. So due to the time constraint of training

a model, the better method was to detect a known object with colors since it was fast and easy to implement. To organize the objects, the breakdown of the objects are as such:

1. Ramp : Gray
2. Stairs : Red
3. Goal: Green
4. Start: Yellow

Once an object has been detected, some basic trigonometry is applied to calculated the distance between the robot and the object. While there are some more efficiency improvements made, the improvements will be discussed in Section 5.3.

4.3 Gait Planning

The ability to walk on non flat surfaces is directly tied to how stable our gaits are. For the gait planning, the problem can be simplify by using the cart-table model and ZMP [10]. Cart-table model describes the movement of the robot or the "cart". ZMP describes the position of the supporting foot on the ground to be balanced.

4.3.1 Zero Moment Point

Zero moment point is a common method to determined if a biped robot is stable. ZMP is the point on the ground at which the total horizontal inertia and gravity forces equal 0 [14]. This process starts in (eq) 4.1 where the superscript gi is the gravity plus inertia forces, m is the robot's mass, g is gravity, and a_G is the acceleration of the center of mass.

$$F^{gi} = mg - ma_G \tag{4.1}$$

To achieve ZMP, the moment around a point would be parallel to the normal vector n or directed along n as shown in (eq) 4.4; The moment around a point can be describe in (eq)

4.2. For future references, the point used for ZMP will be noted as Z as shown in (eq) 4.3.

$$M_p = PG \times mg - PG \times ma_G - H \tag{4.2}$$

$$M_z^{gi} = ZG \times mg - ZG \times ma_G - H \tag{4.3}$$

$$M_z^{gi} \times n = 0 \tag{4.4}$$

Then by using the Newton-Euler equation for the global motion of the biped robot (eq) 4.5, we can set that to 0 to get what is shown (eq) 4.7 which can be rewritten as (eq) 4.9.

$$F^c + mg = ma_G \tag{4.5}$$

$$M_p^c + PG \times mg = H + PG \times ma_G \tag{4.6}$$

$$F^c + mg - ma_G = 0 \tag{4.7}$$

$$M_p^c + PG \times mg - H - PG \times ma_G = 0 \tag{4.8}$$

$$F^c + F^{gi} = 0 \tag{4.9}$$

$$M_p^c + M_p^{gi} = 0 \tag{4.10}$$

What the previous equations tells us are that the biped is dynamically balanced if the contact forces (O) and the gravity inertia forces are strictly opposite as shown in (eq) 4.11.

$$OZ = \frac{n \times M_O^{gi}}{F^{gi} \cdot n} \tag{4.11}$$

4.3.2 Cart-Table Model

The cart-table model assumes the upper-body of the humanoid robot to be the cart located on the robot center of mass (CoM) where the mass is the total mass of the robot. The table portion of the table is the supporting foot of the cart. The cart-table model assumes two sets of cart-table, one for the movement in the coronal plane while the other is used in the sagittal plane; Coronal plane divides the body in half form front to back while sagittal plane divides the body in half from left to right.

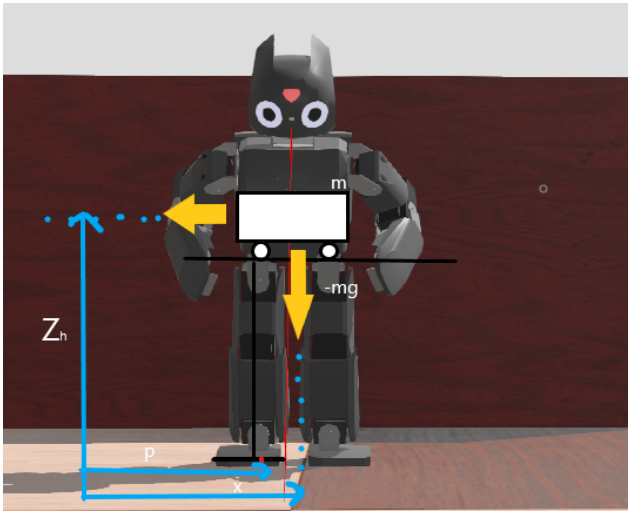


Figure 4.1: Coronal View of the Cart-Table Model

The cart-table model is represented in Figure 4.1 and 4.2. Align with the robot's coronal plane, the position of the cart is define to be at x and Z_h. The red point represents the center of pressure (CoP) as gravity g and cart acceleration creates a moment T_p; T_p is the amount of torque around P which can be found in (eq) 4.12; x represents the coronal plane while y represents the sagittal plane.

$$T_x = Mg(x - P_x) - M\ddot{x}Z_h \tag{4.12}$$

$$T_y = Mg(y - P_y) - M\ddot{y}Z_h \tag{4.13}$$

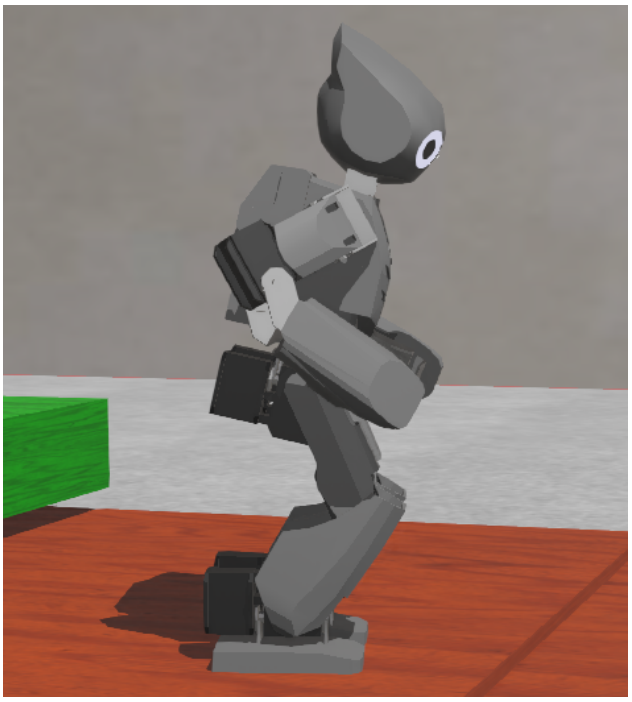


Figure 4.2: Sagittal View of the Cart-Table Model

For the robot to be considered balance, the ZMP and CoP are identical meaning $T_p = 0$, therefore by setting the left hand side of (eq) 4.12 to 0 then the position of the ZMP can be solved as seen in (eq) 4.16. The same concept is applied to the sagittal plane to obtain (eq) 4.17.

$$0 = Mg(x - P_x) - M\ddot{x}Z_h \tag{4.14}$$

$$M\ddot{x}Z_h = Mg(x - P_x) \tag{4.15}$$

$$P_x = x - \frac{Z_h}{g}\ddot{x} \tag{4.16}$$

$$P_y = y - \frac{Z_h}{g}\ddot{y} \tag{4.17}$$

With the position of the ZMP established, the positioning of the foot during walking must be planned and defined so the ZMP trajectory can be derived based on those constraints. The angular trajectory is found using the inverse kinematics discussed in Chapter 3.

4.3.3 Trajectory Planning

Trajectory planning for the robot can be done by path planning where the foot has to go. This also solves how the gait motion needs to be defined for the robot to walk properly. This is visualized in Figure 4.3, the left-hand side shows the path of the foot while the right-hand side shows the locomotion of the gait. Since the stairs are uniform in step height and length, we can define the gait cycle to only need to go H_s high and X_s far; H_s will vary with the step height and X_s will vary with the step length. The same concept applies to the ramp but the only difference is the orientation of the foot. The orientation would equal the angle of the ramp.

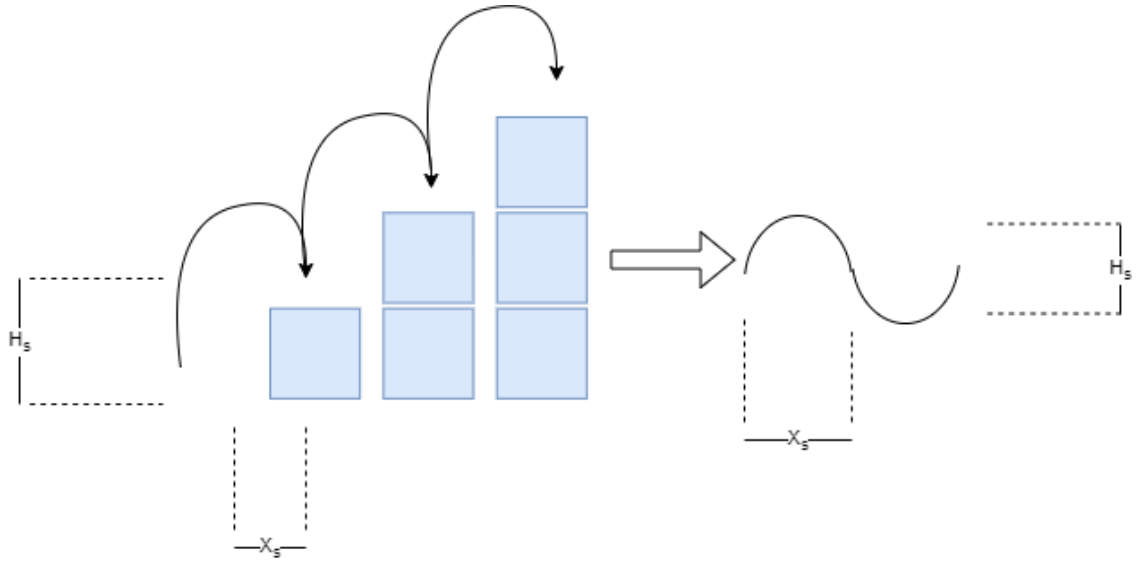


Figure 4.3: Stair Trajectory Planning

Chapter 5: Simulation

Webots is an open-source application made specifically for robot simulation and is developed by Cyberbotics. It provides an in-depth environment package to model, program, and simulate robots with a robust physic engine and graphic interface [4]. Webots is available on all platforms and the main advantage is that the base installation comes with the Darwin-Op already model and with some example programs to learn how to simulate it. The ease of use however comes with some limiting factors, any robot model that came with the installation cannot be modified. This will be discussed in Section 5.1 but another limiting factor is that if a user has a computation-intensive task or the program was made inefficiently, then the user comes with a risk or either dramatically slowing down the simulation or crashing it.

5.1 Darwin-Op Model

The image in Fig 5.1 shows what the Darwin-Op looks like on simulation. The simulated version comes with 6 modifiable slots (head, body, legs, arms). Each slot can equip nodes like IMU, force sensor, and more. The simulated version also allows users to modify some of the native Darwin-Op setting such as backlash which senses if the robot has bumped into an object.

The limitation to this model is that if a user wanted to change the hand attachment for the Darwin-Op, they would have to build a new Darwin-Op model with the new hand attachment instead of being able to modify the model directly. This can be mitigated by making a model of the Darwin-Op elsewhere and having it imported to Webots as a supported file type.

Figure 5.1: Simulation of Darwin-Op

5.2 Environments

For the simulation, a figure can be made by adding a group of shapes and transforming them as needed. 2 different environments were model:

- Stairs going up and down as shown on Fig 5.3
- Ramp going up and down as shown on Fig 5.2

5.3 Programming

For the programming portion of this thesis, it was done in Python 3.8 as some python packages could be leveraged such as NumPy and OpenCV. NumPy is short for Numerical Python and is a library useful for extensive math calculations. NumPy also comes with the

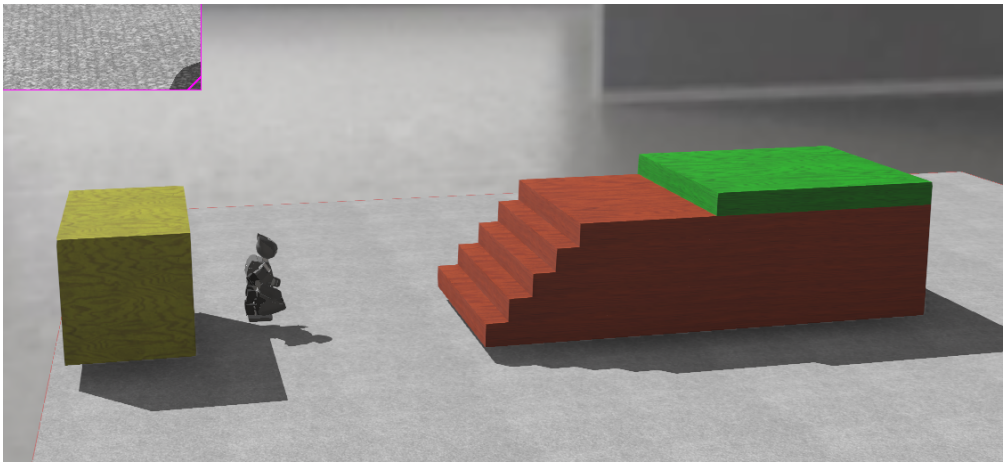


Figure 5.2: Stair Environment

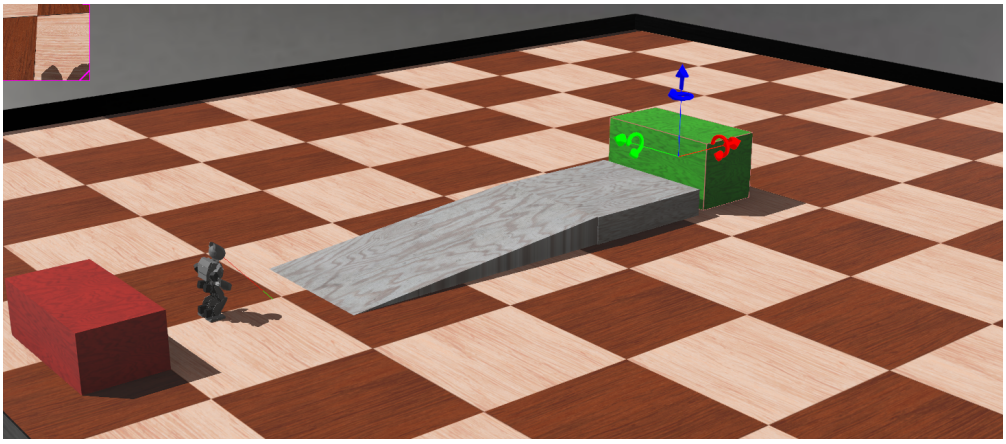


Figure 5.3: Ramp Environment

ability to use broadcasting which allows for matrix operation of different shapes. OpenCV is short for Open Computer Vision and is a library useful for image processing and using current computer vision algorithms.

5.3.1 Control System Flowchart

Figure 5.4 shows the flowchart of the control system and what happens during the simulation runs. The simulation loads any foreknowledge of the environment to the robot before the main loops start; foreknowledge of the environment also includes the initial position of the robot. From there, the simulation goes into the main loop and looks for the nearest object in sight. Once it has determined what the object is, it then goes to the start of the

object. Then it does some gait planning to decide how it will walk up the object. Once it has walked up the object it will then walk up to the goal and then return to its starting position.

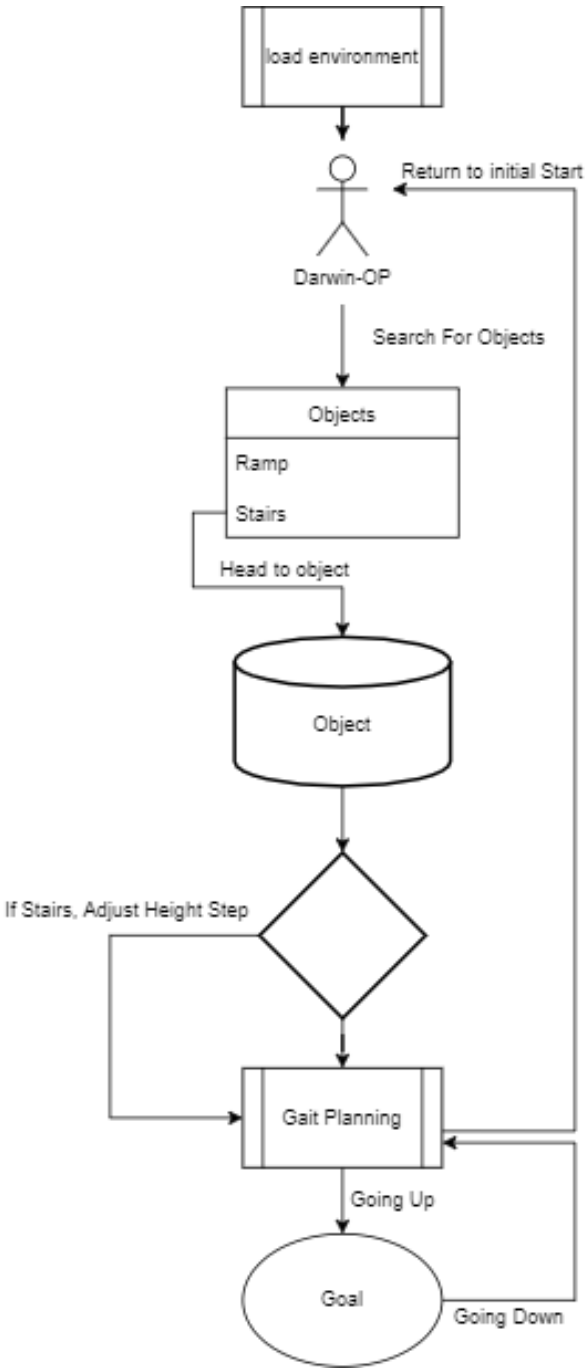


Figure 5.4: Control System Flowchart

5.3.2 Controller Code

This section covers some of the implementation details of the flowchart and the algorithm analysis to determine the time complexity of the simulation; Finding the time complexity helps to identify where the simulation is using up the most resource because the simulation can crash if the time complexity is more than what it can handle.

5.3.3 Color Detection

The standard way of finding color in an image is to go pixel by pixel and check the RGB (red, green, blue) value of the pixel as shown on Algorithm 5.5. Since there are two nested for loops, it makes the time complexity $O(n^2)$. The simulation can handle this time complexity before the main loop but because the color detection happens inside the main loop, then the color detection algorithm had to be optimized to be faster. An optimization

Algorithm 1: Color Detection

```
   Input: Image
   Output: Arrays of RGB
1  for x ∈ camera.getWidth() do
2  |   for y ∈ camera.getHeight() do
3  |   |   red = image[x][y][0]
4  |   |   green = image[x][y][1]
5  |   |   blue = image[x][y][2]
6  |   end for
7  end for
8  return red, green, blue
```

Figure 5.5: Color Detection Algorithm

method is by using the raw image data. camera.getImage() returns a string which can then be interpreted as a buffer using the frombuffer function in NumPy. The frombuffer will then cast the image to usable numbers and can then be reshape to a 3d array. Using the broadcasting capabilities of NumPy, we can drop the last column in the image to have a 3d array which is width x height x [R,G,B]. This 3d array is then usable by OpenCV, OpenCV

is able to convert the array to a HSV (Hue, Saturation, Value) equivalent and see if the desire color is in the image. For example, taking the image in Figure 5.6, it was able to parse out the color brown as shown in Figure 5.7. This process can be shown in the code snippet on Figure 5.8

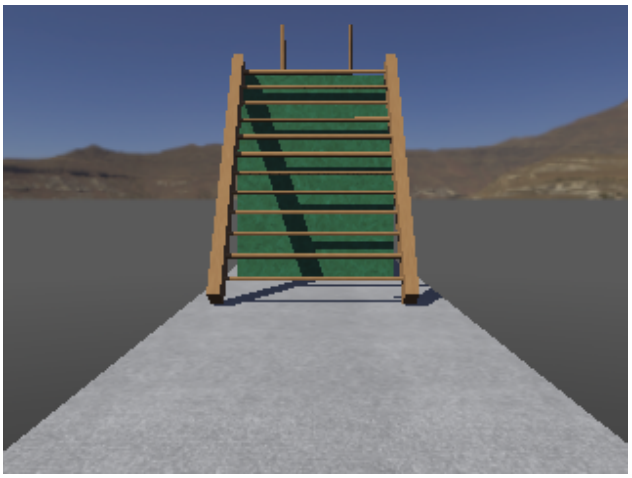


Figure 5.6: Sample Image Used for Color Parsing

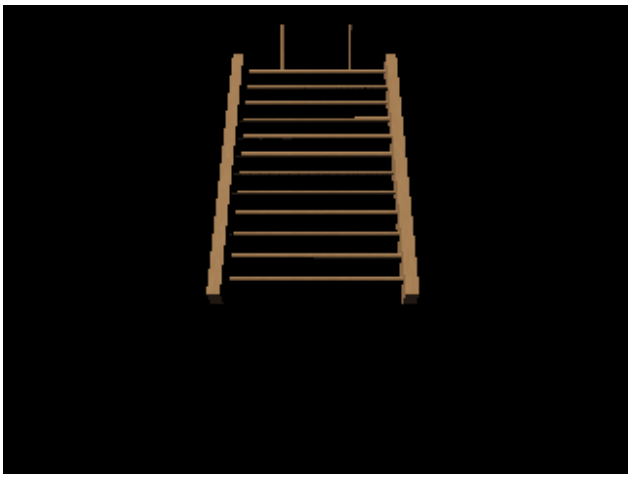


Figure 5.7: Ladder Parsed Out of Image

5.3.4 Gait Planning

Figure 5.9 shows a simple breakdown of what happens during the initial gait planning phase. The initial transformation matrix, T_i is where the robot currently is, while the final transformation matrix, T_f is where the robot should be. In the final transformation matrix,

```python
#take the raw camera data and convert it using numpy
def get_image(camera):
    cameraData = camera.getImage()
    height = camera.getHeight()
    width = camera.getWidth()
    #returns [R, G, B, 255]
    image = np.frombuffer(cameraData, np.uint8).reshape((height, width, 4))
    #drops the last column since the last one is just 255 constant
    image = image[:,:,:-1]
    #cv2.imwrite('current_View.png', image)
    return image

#detection method for finding the ladder, currently only color is implemented
def ladder_detected(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    #hsv scale for brown
    lower_brown = np.array([10, 100, 20])
    upper_brown = np.array([20, 255, 200])
    mask = cv2.inRange(hsv, lower_brown, upper_brown)
    if len(mask[mask > 0]) != 0:
        return True
    return False
```

Figure 5.8: Color Detection Code Snippet

the orientation is kept the same but the position changes in the x and z direction. The

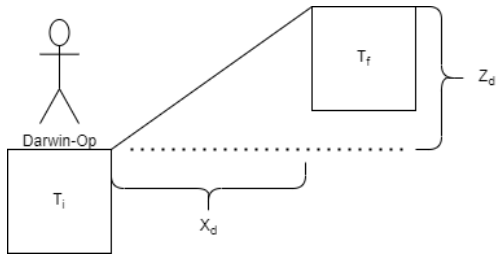


Figure 5.9: Trajectory Planning Diagram

important thing to note is that the transition from the obstacle to final position may not be smooth. In the case of the ramp, because the foot is oriented at an angle different from the final position there are chances of the robot falling down after getting to the final position.

One simple ratification is to make a simple "fallen down" function to help the robot get back up if it had fallen down after traversing the ramp. The fallen function detects the robot has fallen down if the robot is facing down or up for too long. A fall motion is

detected by checking the y axis of the accelerometer and comparing it against a tolerance. After the robot has detected that it has fallen down, it gets back up using motions already implemented by the Darwin-Op motion manager. Since this function is called in the middle of the main loop, the robot constantly checks if it has fallen down. The cost of the fall function is O(1) since there are no For-loops and all of the functions are a constant time to do. This process is shown in Figure 5.10.

```
def checkIfFallen():
    acc_tolerance = 60.0
    acc_step = 100  # Counter upper limit
    acc = accelerometer.getValues()  # Obtain the corresponding value of three axes through acceleration sensor
    if acc[1] < 512.0 - acc_tolerance :  # The value of y-axis will be smaller when face down
        face_up += 1
    else :
        face_up = 0
    if acc[1] > 512.0 + acc_tolerance : # The value of the y-axis will increase when you fall back to the ground
        face_down += 1
    else :
        face_down = 0
    if face_up > acc_step :
        MotionManager.playPage(10) # Perform a face down, floor up motion
        MotionManager.playPage(9) # Get ready to walk
        face_up = 0
    elif face_down > acc_step :
        MotionManager.playPage(11) # Perform a back down and up motion
        MotionManager.playPage(9) # Get ready to walk
        face_down = 0
```

Figure 5.10: Fallen Function

Another function that had to be implemented was a path correction function. Because the trajectory is only a projected trajectory, there are possibilities of the robot straying from the intended path. The path correction function simply checks the robot's current transformation matrix and compares it with the intended transformation matrix. Depending on the margin of error, the function will update some gait parameters to get it back to its intended path. This is mostly to fix any deviation in the y axis since being a distance away from the x and z-axis means the speed isn't suitable for the intended path.

Chapter 6: Experiment and Results

6.1 Experiment

Establishing that a robot can autonomously walk a set of non-flat surfaces means that experiments must be run to verify the performance of the robot. For the research, the experiments conducted observe the performances while changing the parameters on the obstacle and on the robot. Below shows the parameters changed for performance analysis.

- Robot: Speed
- Ramp: Angle
- Stairs: Step Height

The testing explored how well the robot could walk the obstacles on three different speeds: slow, normal, fast. The different speeds is defined as the length of time needed to complete one step. Normal speed is the base case, slow is twice as long as the normal speed and fast is half the time of the normal speed. Section 6.1.1 and 6.1.2 goes into detail about the results of each run. For each experiment, it was conducted by performing 10 runs per trial and slowly increasing the parameter of the experiment. A run would be a success if the robot was able to walk up and down the obstacle without falling down.

6.1.1 Ramp Experiment

The purpose of the ramp experiment was to see how high of an incline the robot could handle without any external support. According to the Americans with Disabilities Act (ADA) standard, the maximum ramp slope should be 18 degrees[16]. Therefore the first

trial starts at an angle of 2 and then increase by 4 each trial with an additional trial being 30 degrees which is consider steep for humans.

6.1.2 Stair Experiment

The purpose of the stair experiment was to see how high of a stair step the robot could handle without any external support. According to Florida's Building Codes, the residential stairway riser must be at least 4 inches with a maximum of 7.75 inches[2]; This range is relative to the ankle height of a human. Since Darwin-Op's lower leg is only 3.66 inches and the ankle height 1.32 inches, that means the robot has a maximum step height of 3.66 inches. The residential stairway riser minimum is greater than the maximum that the robot can handle, therefore the simulated stairs were scaled down to have a minimum riser height of .5 inches and a maximum of 1.5 inches. The first trail starts an a riser height of .5 inches and then increase by .3 inch each trial.

6.2 Results

After running the experiments, the results can be seen below in Section 6.2.1 and 6.2.2. For a basic explanation of the results, "pass" indicated that it passed all trials. "fail" indicated that it failed all trials, and "x/10" indicates the amount of trials passed.

6.2.1 Ramp Results

From the results shown in Table 6.1, we can observe that there are no issues walking up a ramp with an incline of 14 degrees. On the 18 degree mark, there were some failure on the fast speeds due to slippage. For the ambitious 30 degree goal, it fail on all speeds because the foot wasn't able to get a good grip on the ramp and slipped.

Table 6.1: Ramp Experiment

Angle (degrees)	Slow	Normal	Fast
2	pass	pass	pass
6	pass	pass	pass
10	pass	pass	pass
14	pass	pass	pass
18	pass	pass	7/10
30	fail	fail	fail

6.2.2 Stair Results

From the results shown in Table 6.2, any step height that were lower than the robot's ankle height had some successful trials, the failures on the lower height were often from misalignment of where the foot needed to be. For the step height higher than 1.1, it mostly fail due to the gait not having a high enough gait swing. Since the gait swing wasn't high enough, it either hit the edge of the step or didn't hit it at all. Making the robot to have a higher step height often cause the the robot to be unbalance while trying to increase the swing distance often cause the robot to walk backwards by itself.

Table 6.2: Stairs Experiment

Step Height (inch)	Slow	Normal	Fast
0.5	4/10	4/10	4/10
0.7	5/10	5/10	3/10
0.9	3/10	3/10	1/10
1.1	2/10	2/10	1/10
1.3	fail	fail	fail
1.5	fail	fail	fail

6.3 Remarks

During the course of this thesis, running the experiment helped to identify weaknesses in the system and where things could be improved upon. Running this experiment also help to define what was considered a failure. As acknowledge in Section 5.3.4, transitioning from a ramp to a flat surface can cause the robot to fall down; The transition issue is mostly cause

by alignment issue of the ramp height and the block height connected to the ramp. Although the robot falls down, because the robot doesn't fall down during the ramp traversal it's not consider a failure.

Apart from alignment issues, it was noticed that the angle of the upper body had an impact on the walking. This is understandable as the robot is under the effects of gravity meaning the pose of the upper body can change the center of mass. To resolve this issue, the angle of the upper body had to be changed before walking up or down.

Another thing to note is that the difference between a ramp and a stair is the smoothness of the surface. On a ramp, there is nothing to stumble on since the entire surface is smooth but a stair can be considered a jagged ramp. Since the stair has edges, stepping on an edge can cause significant problems for traversing the next step. This means proper foot placement was crucial as seen in the results between the stair and ramp experiment.

Chapter 7: Conclusion and Future Work

7.1 Conclusion

From the initial conception of the research, it was shown that the Darwin-Op can handle autonomously locomotion for non-flat terrains. Though effectiveness on non-smooth surfaces will vary, by using a cart-table model to do ZMP trajectory planning, the robot was able to plan a path to keep itself balanced. Combine with the inverse kinematics, the robot was able to do its locomotion with high precision. From the results in Chapter 6.1, we can see that it's able to handle most steep challenges.

Overall, this research has taught me invaluable things about the current state of things and what already exists and doesn't. With what's already possible in modern robotics, it won't be soon until we see robots walking normally like humans. I came in with high expectations and I can say I satisfy most of them.

7.2 Future Work

While there is much more that can be done for this research, such as testing on the physical robot. This section highlights some of the bigger key areas of research that could have been done given more time.

7.2.1 Walking Refinement

As shown from Section 6.2.2, not a single trial had a 100% pass rate. To increase the pass rate, the stair climbing method would be refined to a more reactive approach. Instead of planning the trajectory ahead of time, the trajectory would be planned as the robot climbs

a step. The robot could use some computer vision logic to detect the straight edges of a stair, form a rectangle from it, and use that as the guiding point for the next step to take.

For the ramp walking, the only improvement there would be to improve the transition from the ramp to the flat surface and vice versa. Some improvement can be making sure the environment doesn't change during execution.

7.2.2 Object Detection

The current object detection algorithm is not robust enough since the obstacle could share the same color as another random object. The robustness can be solved by deploying machine learning techniques to have objects be recognized through detection and classification; the usage of machine learning would also allow the robot to estimate the pose of the object more accurately.

7.2.3 Ladder Climbing

During the initial research for this thesis, the goal was to autonomously climb a set of ladders by using hook grippers native to the Darwin-Op [7]; The focus of the thesis had to shift due to the current limitations of the simulation software. The proposed methodology to autonomously climb a set of ladders is separated into two steps. First is identifying the ladder and the rungs through the use of the color filter function such as in Chapter 4 or through the use of localizing rungs with cloud point data [6]. The second step is to climb the ladder by making a whole body trajectory calculation with each rung pose [12] until the final rung has been grasp.

References

[1] Ali Fawzi Abdul Kareem and Ahmed Abdul Hussein Ali. Experimental and theoretical optimal regulator control of balance zero moment point for bipedal robot. *Journal of Engineering and Sustainable Development*, January 2020.

[2] Ryan Cooks. Florida building codes for residential stairs. https://www.hunker.com/13402630/florida-building-codes-for-residential-stairs.

[3] Manuel Crisostomo and Antonio Coimbra. Consideration on dynamic and static stability of a biped robot. February 2005.

[4] Cyberbotics. Webots: robot simulator. https://cyberbotics.com/, 1998.

[5] Ruoyu Fang and Cheng Cai. Computer vision based obstacle detection and target tracking for autonomous vehicles. *MATEC Web of Conferences 336*, 2021.

[6] Prashanta Gyawali and Jeff McGough. Simulation of detecting and climbing a ladder for a humanoid robot. 2013.

[7] Inyoung Ha, Yusuke Tamura, and Hajime Asama. Development of open platform humanoid robot darwin-op. *Advanced Robotics*, 27:223–232, January 2012.

[8] Chin Yun Han. Vision guided soccer robot. 2017.

[9] G. Clark Haynes and Alfred A Rizzi. Gaits and gait transitions for legged robots. *2006 IEEE International Conference on Robotics and Automation*, May 2006.

[10] Yunn-Lin Hwang, Thi-Na Ta, Chien-Hsin Chen, and Kun-Nan Chen. Using zero moment point preview control formulation to generate nonlinear trajectories of walking patterns on humanoid robots. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 2405–2411, 2015.

[11] Antonio R Jimenez, Fernando Seco, Francisco Zampella, Jose C Prieto, and Jorge Guevara. Pdr with a foot-mounted imu and ramp detection. *Sensors*, September 2011.

[12] Masao Kanazawa, Shunichi Nozawa, Yohei Kakiuchi, Yoshiki Kanemoto, Kei Okada, and Masu Inaba. Robust vertical ladder climbing and transitioning between ladder and catwalk for humanoid robots. September 2015.

[13] Xiao Li, Yangmin Li, and Xinzhe Cui. Kinematic analysis and gait planning for a darwin-op humanoid robot. *International Conference on Robotics and Biomimetics*, December 2016.

[14] Philippe Sardain and Guy Bessonnet. Forces acting on a biped robot. center of pressure - zero moment point. *IEEE Transactions On Systems, Man, and Cybernetics - Part A: Systems and Humans*, September 2004.

[15] Robert L Williams. Darwin-op humanoid robot kinematics. *IDETC/CIE 2012*, August 2012.

[16] Americans with Disabilities Act. Building access ramp slope or pitch requirements. https://inspectapedia.com/Stairs/Access_Ramp_Slope.

Appendix A: Nomenclature

Frame:	A standalone image from a video sequence
WeBots:	Simulation Software
Library:	Collection of implemented techniques and algorithms
OpenCV:	Open Computer Vision, library containing image processing and computer vision algorithms
NumPy:	Numerical Python, library containing comprehensive mathematical functions
Humanoid:	Resembling a human in shape

Appendix B: Copyright Permissions

The permission below is for the reproduction of material in Chapter 5.

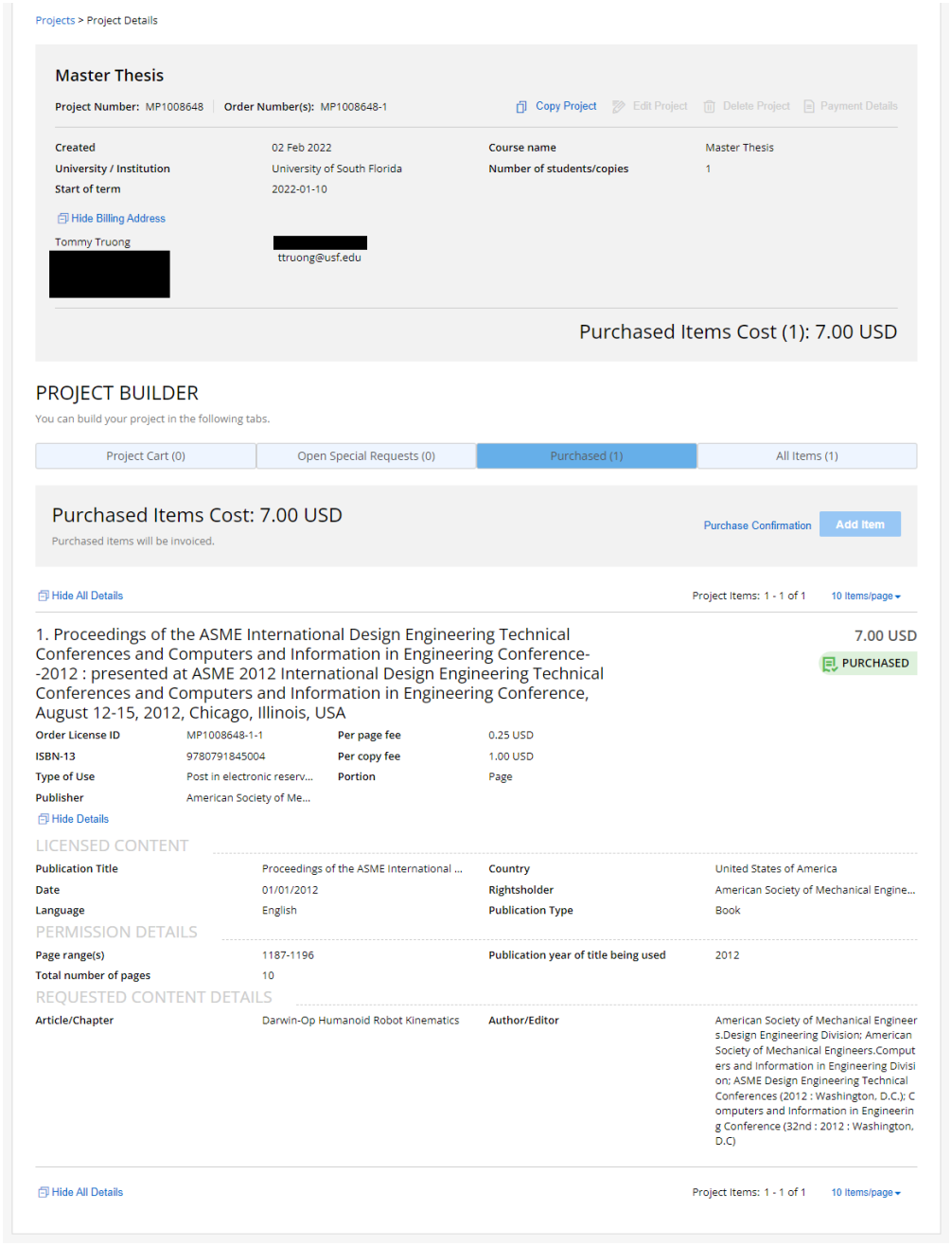
Webots User Guide R2022a

Webots User Guide

Release R2022a

Trademark Information

The permission below is for the figures used in Chapter 3



Projects > Project Details

Master Thesis

Project Number: MP1008648 | Order Number(s): MP1008648-1 — Copy Project | Edit Project | Delete Project | Payment Details

Created	02 Feb 2022	Course name	Master Thesis
University / Institution	University of South Florida	Number of students/copies	1
Start of term	2022-01-10		

Hide Billing Address

Tommy Truong
[redacted]
[redacted]
ttruong@usf.edu

Purchased Items Cost (1): 7.00 USD

PROJECT BUILDER

You can build your project in the following tabs.

Project Cart (0) | Open Special Requests (0) | Purchased (1) | All Items (1)

Purchased Items Cost: 7.00 USD

Purchased items will be invoiced.

Purchase Confirmation | Add Item

Hide All Details — Project Items: 1 - 1 of 1 — 10 Items/page

1. Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference--2012 : presented at ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, August 12-15, 2012, Chicago, Illinois, USA

7.00 USD

PURCHASED

Order License ID	MP1008648-1-1	Per page fee	0.25 USD
ISBN-13	9780791845004	Per copy fee	1.00 USD
Type of Use	Post in electronic reserv...	Portion	Page
Publisher	American Society of Me...		

Hide Details

LICENSED CONTENT

Publication Title	Proceedings of the ASME International ...	Country	United States of America
Date	01/01/2012	Rightsholder	American Society of Mechanical Engine...
Language	English	Publication Type	Book

PERMISSION DETAILS

Page range(s)	1187-1196	Publication year of title being used	2012
Total number of pages	10		

REQUESTED CONTENT DETAILS

Article/Chapter	Darwin-Op Humanoid Robot Kinematics	Author/Editor	American Society of Mechanical Engineers.Design Engineering Division; American Society of Mechanical Engineers.Computers and Information in Engineering Division; ASME Design Engineering Technical Conferences (2012 : Washington, D.C.); Computers and Information in Engineering Conference (32nd : 2012 : Washington, D.C)

Hide All Details — Project Items: 1 - 1 of 1 — 10 Items/page