Design and Control of a Highly-Articulated Salamander-Inspired Robot for Future Search and Rescue Applications

Final Project Report

ECE 4012 Senior Design Project

Salamander Robot Team - Group LM2

ECE 4012 Principal Advisor:	Dr. Linda S. Milor
ECE Advisor:	Dr. Patricio A. Vela
ME Advisor:	Dr. Amit S. Jariwala
MSE Advisor:	Dr. Mary Lynn Realff

Austin Bush	abush34@gatech.edu
Sunit Kulkarni	skulkarni38@gatech.edu
Hariank Mistry	hmistry8@gatech.edu
Alex Popescu	apopescu@gatech.edu
Jonathan Rundquist	jrundquist6@gatech.edu
Shashwat Sitesh	ssitesh3@gatech.edu
Brian Weaver	bweaver38@gatech.edu
Calvin Yao	cyao31@gatech.edu

Submitted

May 4, 2017

Executive Summary

Robots have the potential to serve as extensions of emergency response professionals. During the immediate response to a disaster, rescue robots can provide valuable real-time data to help assess and monitor a situation and potentially save lives. In environments with hazardous materials or precarious structures, robots can safely navigate in places where humans cannot.

However, large-scale catastrophes are often characterized by rough, uneven terrain that may result from rubble [1]. Rough, uneven terrain poses a challenge for traditional tracked or wheeled robots to traverse. Because simply increasing overall vehicle size prohibits vehicles from entering confined spaces, mobile, yet small, robots are necessary.

We have developed a salamander-inspired robotic platform that resolves these terrain-related challenges by virtue of the salamander's unique kinematic properties. Compared to legged and tracked robots, salamanders provide increased mobility (articulated legs and spine, compliant feet), stability (low center of gravity), and portability (small size and weight). Our robot builds upon a biomimetic design [2] by optimizing the salamander's physical dimensions and gait parameters for good locomotion performance.

Preliminary locomotion testing was done over a small range of terrains with various feet types and produced some promising results. However, extensive testing over more terrains with different feet types is necessary to fully validate our robot and compare its rescue-robot capabilities with those of other unmanned ground vehicles (UGV). The total development costs were $154,394.46, which nets a 22.8% profit margin for a $200,000.00 price point. The final total parts cost was $7,307.10.

Table of Contents

Design and Control of a Highly-Articulated Salamander-Inspired Robot for Future Search and Rescue Applications

1. Introduction

Robots can augment the work of of emergency-response professionals. In a disaster situation, rescue robots can provide valuable real-time data, help assess and monitor a situation and potentially save lives. In environments with hazardous materials, such as the Fukushima reactor meltdown, robots are able to safely navigate to places inaccessible to humans.

Biological principles were leveraged to develop the first iteration of a salamander-inspired robotic platform that can be utilized in disaster-response scenarios. The mechanical design, fabrication, interfacing, and control design subproblems and their solutions are documented. In addition, some preliminary testing data is given.

1.1 Objective

The objective of this project is to apply biological principles to develop a salamander like robotic testbed that can be used for disaster robotics research. This is only a first iteration and future developments will lead to a rugged reconnaissance vehicle that can be used for disaster-response applications. The development process consisted of mechanical design, fabrication, control, and testing. This mechanical design and control of this robot (Supermander) is based on the Pleurobot robot developed by the EPFL (École Polytechnique Fédérale de Lausanne) BioRobotics Lab (Figure 1).

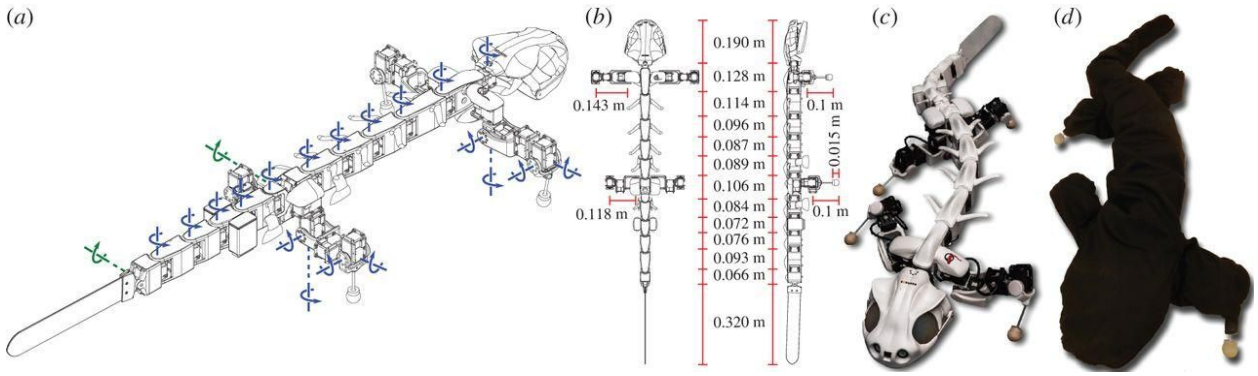


Figure 1. Degrees of freedom locations (*a)* and measurements (*b*) of the Pleurobot, with completed robot design (*c*) and swimming suit (*d*) [2]

1.2 Motivation

The motivation behind Supermander is to develop a controllable scouting robot. Post-disaster scouting is a subset of disaster robotics in which first-response ground robots can immediately enter the disaster area and move through tight spaces inaccessible to humans. To overcome rugged terrain, robot programmers have implemented a wide variety of designs and features ranging from tread-based movement to serpentine crawling [1].

For instance, Pleurobot, a state-of-the-art salamander robot designed by the BioRob Lab at EPFL, has the ability to negotiate rugged terrain due to its compact and low frame, untethered battery supply, and adaptability to onboard I/O devices such as sensors and cameras [2]. Supermander's control system consists of hand maneuvered gaits in addition to pre-programed ones to deal with unforeseen terrain challenges.

1.3 Background and Prior Work

The proposed salamander-based robot is similar to a large class of bioinspired snake-like robots due to its flexible spine. Snake-like robotics is an expanding research field which had its origins in the work of Hirose in the 1980s [3]. Recent snake robots include the Omnitread OT-4, a

7-segmented snake utilizing pneumatic actuators and moving tracks on the exterior [4], and a more traditional servo-based 16 degrees-of-freedom (DOF) snake by Choset et al. [5]. The IVALab at Georgia Tech has built Snakey, a 12-DOF servo-based 3D-printed snake that utilizes scale-induced frictional anisotropy to induce locomotion [6].

Free-serpentine rescue robots have also previously been developed, including the International Rescue System Institute (IRS) Soryu robot [7]. The IRS Soryu consists of three pods, each with caterpillar treads, linked by spherical joints. It also hosts a thermographic camera. Rescue robots with legs, such as the RHex [8], have demonstrated good performance over rough terrain. The RHex is a six-legged robot which uses compliant "C"-shaped-legs to achieve extremely high locomotion speeds, exceeding five body lengths per second on even terrain.

Finally, previous work in the area of biomimetic salamander-inspired robots has been done by the EPFL's Biorobotics Lab. The most advanced biomimetic robot is the Pleurobot. Pleurobot is the third salamander-inspired robot designed by the Biorobotics Lab following their previous robots, Salamandra Robotica I and II. Unlike the simplified skeleton of the previous two iterations, this biologically inspired robot was designed to more accurately replicate the kinematics structure and scaled dimensions of Pleurodeles waltl (P. waltl) with an Intel Atom 1.6 GHz computer, a more articulated spine using Dynamixel MX-64R servomotors, and additional motors placed in each limb give two more degrees of freedom. The Pleurobot's walk cycle (gait) was derived from transforming and scaling existing P. waltl gait data gathered from tracked cineradiography analysis and applying it to the Pleurobot's robotic joints, giving the robot terrestrial and aquatic gaits emulating those of actual salamanders [2].

2. Project Description and Goals

The goal of this project was to build a rugged salamander-like robot capable of traversing uneven terrain for use in disaster reconnaissance missions. The targeted users of this robot are government agencies and researchers who will use it as a testbed for future disaster robotics research.

The final robot consists of 29 body servo motors and a forward-looking camera. Power and data are supplied to the motors via a tether. The servo motors are controlled with software utilizing the ROS (Robot Operating System) for compatibility with custom gait controllers. Supermander uses custom vertical-lift brackets to allow both vertical and horizontal spinal movement and a simple joystick user interface to allow a human operator to control the speed and direction of the robot without engaging in low-level motor control. Several walking gait options were developed and tested.

To further increase mobility, several interchangeable foot options were developed. The key features of the robot are summarized as follows:

- Highly-articulated quadruped body consisting of 29 servo motors
- Custom spine brackets allowing for vertical spine flexibility
- Gait control system capable of executing various gait options
- ROS compatibility
- Joystick user interface functionality for easy operator control
- 3 tested foot options

Future work may extend these features to include detection and avoidance of obstacles, localization sensors, and autonomous navigation.

3. Technical Specifications and Verification

3.1 Disaster-Response Deployment Specifications

This salamander robot system is to be deployed in a disaster-response scenario. There are several specifications which are relevant to disaster-response teams who will actually be using this platform, as shown in Table 1. For example, the weight and volume need to be small enough to be man-portable for deployment, which is a useful feature in areas which may have damaged infrastructure [1]. Table 1 shows that both weight and volume desired specifications were met. The weight actual specification fluctuates depending on the specific feet or forward-looking camera that is utilized.

In Table 2, the relevant performance specifications for a disaster-response robot are displayed. The only specification from Table 2 that was met completely was the traversable terrain height deviation. The other specifications were partially met. In future iterations of this robot, the traversable grade, traversal speed, and tether length can all be increased. Turning was not implemented due to time constraints.

Table 1. Specifications Applicable to Disaster-Response Team

Item	Desired Specification	Actual Specification
Weight	< 25 kg	5 ± 2 kg
Volume	< 2m x 0.5m x 0.5m	0.63 x 0.41 x 0.2 m

Table 2. Disaster-Response Related Performance Specifications

Item	Desired Specification	Actual Specification
Traversable terrain height deviation (max)	> 5 cm	3 cm
Traversable grade (max)	> 3 %	3 %
Turn radius (min)	< 2 m	not implemented
Terrain traversal speed (max)	> 10m / minute	4m / minute
Tether length	> 3 m	2.44 m

3.2 Electrical Specifications

Table 3 shows specifications for the electrical power needed for robot locomotion. For a typical gait, less than 60W (RMS) power was necessary, which met the specification of < 100W. This means that the tether needed to contain only two 20-gauge wires, power and ground, to power the robot. Thicker or duplicate power lines were not necessary.

Table 3. Robot Power Specifications

Item	Desired Power	Actual Power
Typical total servomotor load during gait	< 100W	< 60 W

4. Design Approach and Details

4.1 Design Approach

The design of this robot was accomplished through the use of three concurrent design subteams that worked in collaboration: mechanical team, interfacing team, and controls team.

4.1.1 Mechanical Team: Design & Fabrication

The mechanical engineering (ME) team took on the mechanical fabrication of the robot, which consisted of off-the-shelf motors and brackets, custom brackets, 3D printed parts, and custom sheet metal parts. The robot design was inspired by the Pleurobot [2] from EPFL as described in Section 1.3. Using this design as a starting point, the ME members of the team re-designed the robot to optimize it for the proposed disaster-response application. The servo motors which were utilized for joint actuation are the Robotis Dynamixel MX-28AT and Robotis Dynamixel MX-64AT. The MX-28AT is shown in Figure 2. The MX-64AT is not shown as it's just a slightly larger version of the MX-28AT.

Figure 2. Robotis Dynamixel MX-28AT Servomotor [9]

The general design is based on the EPFL Pleurobot by utilizing similar joint connection and degrees of freedom. To understand the overall layout of the robot, each section was designed and then assembled in Autodesk Inventor and Solidworks. The length of both the spine and legs were optimized using previous research and gait simulations which will be discussed in Section 4.1.5.

Next, a mechanical analysis of the necessary torques and strengths for each joint was performed using the same software. The analysis identified that the motors that were provided to the team were, in fact, sufficiently powerful enough to move the robot. In addition, it confirmed that the brackets provided off the shelf by Robotis would also be sufficiently structural to make the robot.

4.1.1.1 Spine Design

A central spine was used to form the body to which attached leg, head, and tail assemblies. It is a chain of seven MX-28AT type motors beginning and ending with the “shoulder” joints that will be used to attach the legs. It was important that the spine’s length could support the desired gaits without the legs colliding, as well as meet a design specification that climbing stairs would be

physically possible. Between the first and second motors from each end was a MX-64 that sat on top of the spine which allowed another degree of freedom in the spine to account for a vertical motion in the spine. These motors may be used in the future to accomplish climbing stairs or higher grades.

This layout, as shown in Figure 3, was made possible by some custom brackets which preserved the spinal node lengths modelled in the aforementioned simulations. This was important to be able to accurately simulate a salamander's gait. MX-64's were used because this specific joint requires much more torque than the other spinal joints.

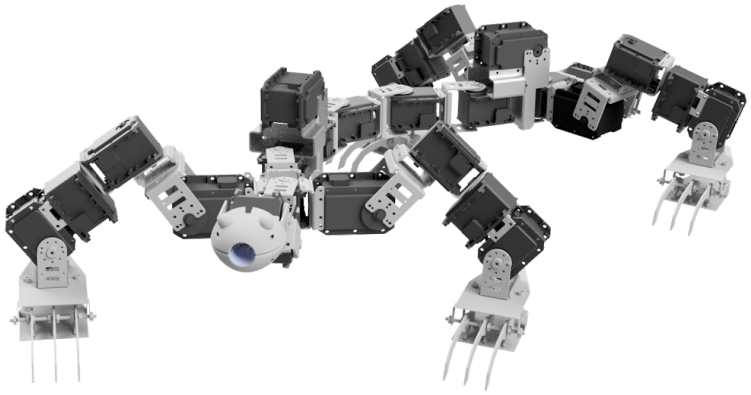


Figure 3. The Final Design of the Spine.

4.1.1.2 Leg Design

Contrary to as mentioned in the initial proposal, passive spring actuated legs were not used, but instead fully controllable legs were made by using MX-28s and more off the shelf brackets. Figure 4 describes the design for the legs. This 4-DOF layout allows for the necessary range of motion necessary to host a salamander-like gait. The to redundant degrees of freedom allow for the foot to raise while still remaining horizontal to the ground. This is important to allow for the foot to adapt as necessary to step over possible objects and conform better to the current terrain.

Each leg is attached to the first or last motors of the spine, two on each one to form a set of fore and aft legs. When including this motor into the leg actuation, each leg acts as if it has 5 degrees of freedom. This additional vertical axis of rotation allows for a redundant axis which allows for a linear motion between leg and foot motion; the feet can remain pointing forward during the whole gait, which would otherwise not be possible. However, as the shoulder joint is shared between the legs, the legs must act as a locked pair, ie, as one foot steps forward, the other must be stepping back to retain this linear characteristic with the feet.

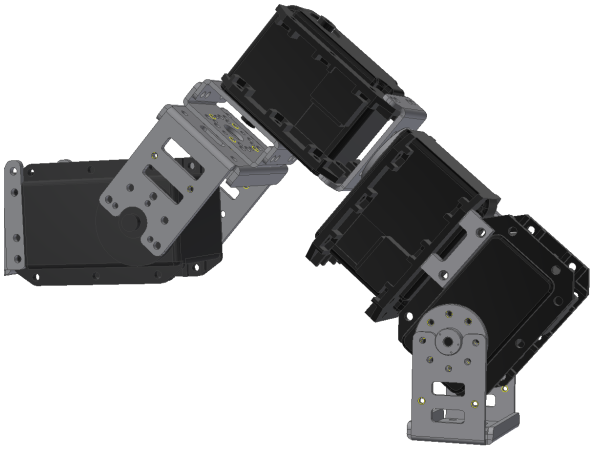


Figure 4. The Final Design of the Legs.

4.1.1.3 Foot Design

Different foot designs were explored by measuring simply the speed at which the robot traversed different terrains with the different feet. Feet were categorized into two types: static and dynamic. Dynamic feet are characterized by their ability to either passively or actively adapt to their environment by changing their shape. The remaining feet were considered static. There were 4 feet built, 2 static and 2 dynamic. One static “ball” foot was built with intention to be versatile in all environments. The “squirrel” foot is a static foot meant to grip the ground more

sharply. With a certain gait type, this foot also acts like a shovel in gravel, pushing the robot forward as if it was swimming through the loose rock. The “badger” foot is a dynamic version of the “squirrel” foot, which gripped the ground below it with metal claws when pressure was applied to the foot. This foot performed similarly to its static counterpart, as seen in Table 4, but its capabilities were not fully explored due to time. The best environment for this type of foot was envisioned to be a steep grade, as the claws could actively dig into dirt, crevices, or loose rock to hang on. A fourth dynamic foot was imagined but not fully built, as seen in Figure 8. This foot would have bent to the environment around it, and it is imagined that it would function best in sand. Table 4 lists data from preliminary tests performed with the robot walking using the same gait on different surfaces with different feet. While more testing with more applicable terrains would have been preferred, more time was needed. Scenarios that were affected by insufficient testing are indicated on the table.

Table 4. Foot Performance on Different Surfaces

Traversing Speed (ft/s)			
Foot Design	Tile	Carpet	Gravel
Ball (Static)	0.3	0.3	0.27
“Squirrel” (Static)	0.12	0.02	.22
“Badger” (Dynamic)	0.2	0.02	0.19
Dual Material Compliant (Dynamic)	NA*	NA*	NA*

* Insufficient testing data

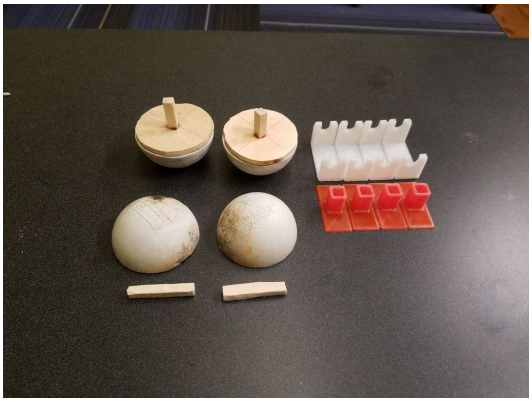


Figure 5. “Ball” Foot

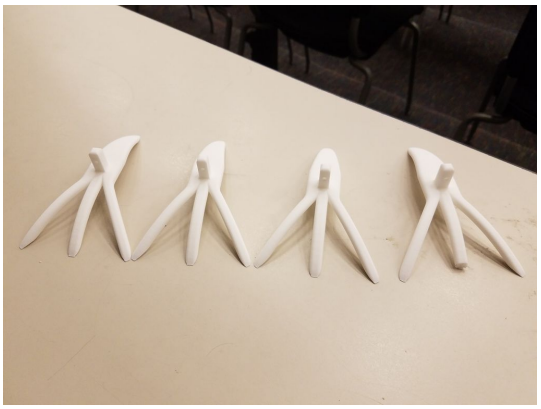


Figure 6. “Squirrel” Foot

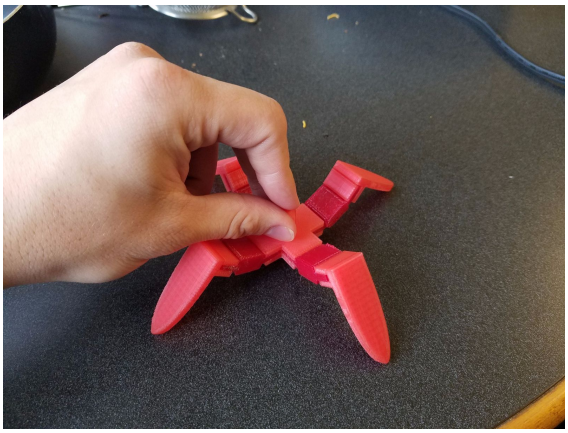


Figure 7. Dual Material Compliant Foot

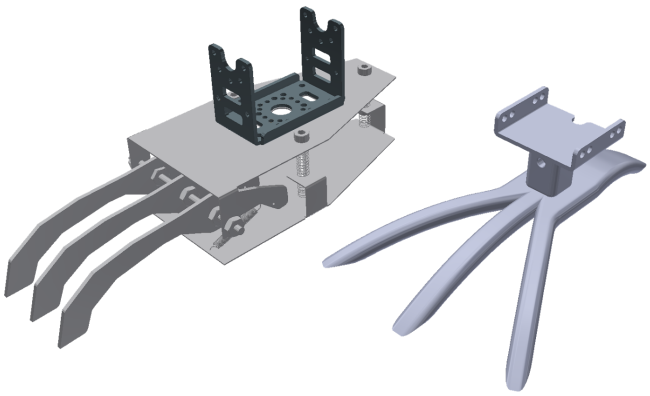


Figure 8. "Badger" Foot Compared to "Squirrel" Foot

4.1.2 Interfacing Team: Communications and Control Framework Design

4.1.2.1 System Overview

A high-level overview of the components of the salamander robot system is displayed in Figure 9. The three main components are the operator control station (OCS), the motor interfacing board, and the robot itself. The tether, shown in gold, contains an asynchronous serial data line, as well as a 12V power line, which connects the on-board servo motors to the OCS through the motor interfacing board. All control processing is done off-board the robot on the OCS, to reduce complexity and weight on-board.

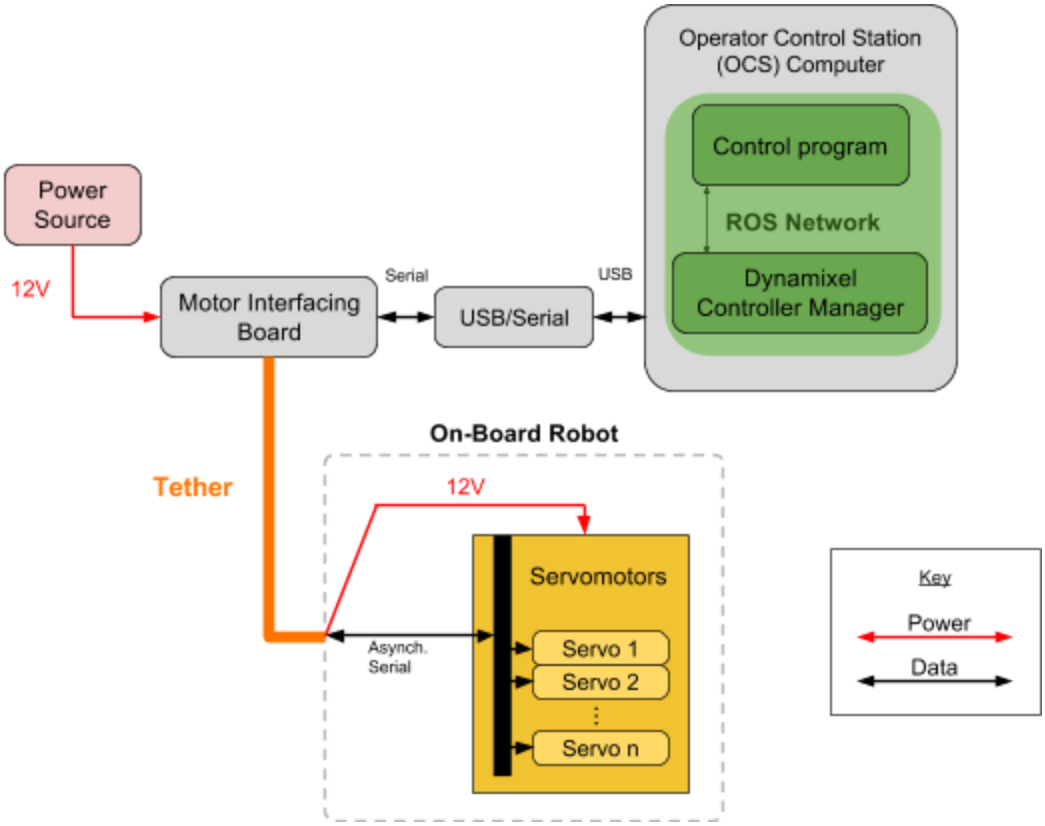


Figure 9. High-level overview of salamander robot system showing power and data connections.

4.1.2.2 Electrical Fabrication

The electrical wiring for the robot was assembled with pre-made 3-wire connectors from the Dynamixel catalog, and attached to the servos. The motor interfacing board is the OpenCM 9.04 board manufactured by Dynamixel.

4.1.3 ROS Control Framework

Robot Operating System (ROS) is a suite of programs that can run on distributed computers in a TCP/IP network, which send and receive messages via a publisher-subscriber model. Each program is called a *node*. Each node publishes or subscribes to certain message *topics*. A ROS *stack*, or set of packages, was specifically created for communicating with the Dynamixel motors, and is called the dynamixel_motor stack (http://wiki.ros.org/dynamixel_motor). This ROS stack is Python-based and allows Python ROS nodes to easily send position commands,

read states, and execute other utility commands, to/from the Dynamixel motors.

The dynamixel_motor stack enables communication with the motors through a "controller manager" ROS node, which manages the motor serial connection. The controller manager node is shown in Figure 9. The stack also enables creation of joint controller ROS services for each joint. These joint controllers allow commanding of motor positions just by publishing to a particular topic.

4.1.3.1 On-Board Actuators

The servomotor positions are commanded via an asynchronous serial connection running at 1Mbps. The motor electronics interfacing was tested by publishing servo motor position commands to the relevant topics, and observing the response.

4.1.3.2 Power

Power is provided to the servo motors by a nominal 12V power line in the tether. A 12V, 5A, portable laptop power supply was used to power the robot because of its compactness and ability to supply 60W of power. Normal walking gaits were able to run without errors on the 60W supply, indicating the robot consumed less than 60W on average when walking.

4.1.4 Controls Team: Full-Body Gait Optimization and Control

4.1.4.1 Controls Overview

The controls subteam was tasked with making the salamander robot walk and move. To accomplish this task, we first studied the previous Pleurobot [2] control strategy. The Pleurobot controller simply replayed joint data collected from cineradiography conducted on walking and swimming salamanders. The replay speed was scaled to be slower, because the robot salamander was larger than a real one [2].

This design needed to go beyond simply replaying gait data obtained from a salamander; therefore several other methods of generating gaits for the robot were developed. First, we utilized kinematic simulations to "lock" feet down and understand the effect spinal undulations had on locomotion. Optimizing a gait for speed using this approach utilized the Optragen MATLAB package. Next, dynamics simulations were carried out in Gazebo in order to accurately capture the effects of foot geometry, foot slippage, and motor torques. Finally, a genetic algorithm (GA) was used to optimize the gait of the physics-based salamander model.

4.1.4.2 Kinematics-Based Control Approaches

In the kinematics-based control approach, the dynamics of the robot system are ignored for simplicity, and only geometry is considered. Figure 10a shows a simple model for the salamander robot: a 1-degree-of-freedom (DOF) spine with 0-DOF legs. This model was created with the MATLAB package. In this model, only 1 foot is "locked down" to the ground at a time by fixing its position only. The "locked" foot is allowed to pivot (rotate) and the central spine joint is controlled to keep the rest of the body straight. With a 1-3-2-4 foot touchdown pattern (Figure 11 shows the leg labeling convention), this model produces forward "locomotion," showing that a very simple system of just 1 spine DoF coupled with foot pivoting is enough to create forward motion. However, this model is unrealistic because it assumes only one foot may contact the ground at a time.

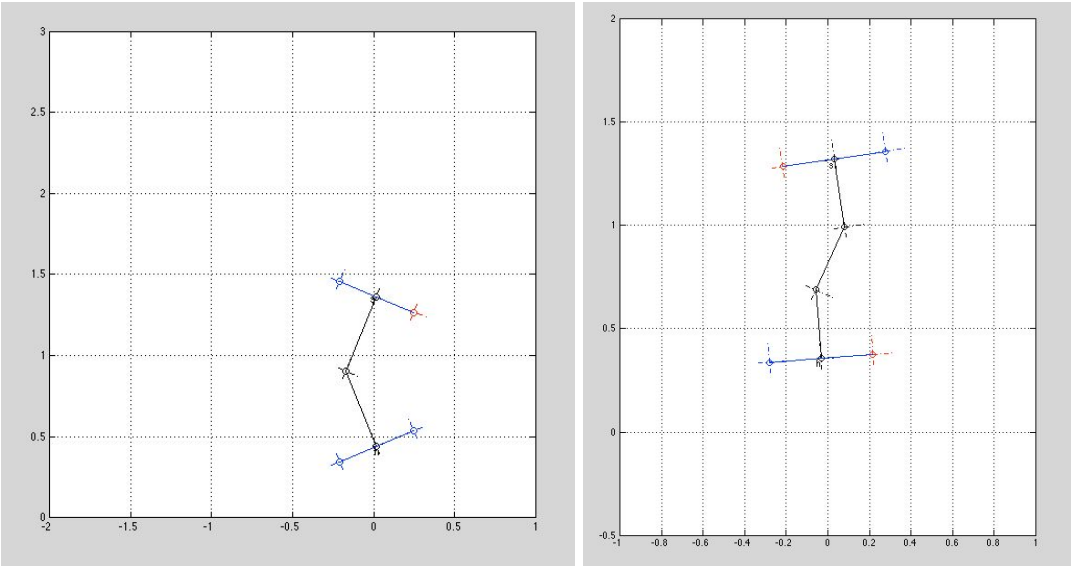


Figure 10. Foot locking gait for different numbers of spine segments. (a) 2 spine segments, (b) 3 spine segments. The "locked" feet are shown in red.

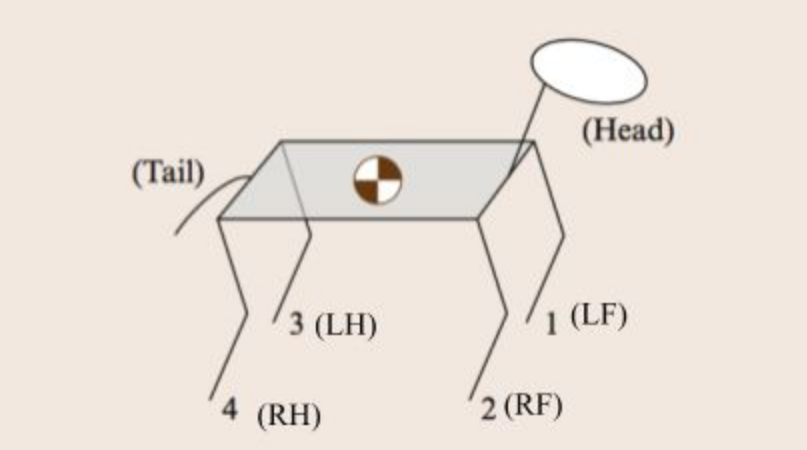


Figure 11. Leg labeling of quadruped robot, adapted from [14].

Figure 10b shows a more detailed quadruped locomotion model, where the spine contains 2 DOF and legs still contain 0 DOF. Because there are now 2 DOF in the spine, 2 feet may now be "locked down", simulating ground contact, while still allowing the spine joints freedom to move. This model is simple, yet it allows simulation of the "trot" gait, which is characterized by the two diagonally opposite legs touching down and lifting off simultaneously. The "footfall diagram", "touchdown diagram", or "gait diagram" for this gait is shown in Figure 10a. The trot

gait is still relatively unstable because the robot does not support itself with a support polygon and can tip over.

In order to simulate more stable gaits, such as the walking gaits shown in Figure 12, assuming no foot slippage, at least 3 DOF are needed in the spine. Thus, the models shown in Figures 14 and 15, with 3 and 4 DOF in the spine, respectively, were created to allow 3 feet to contact the ground simultaneously. When 3 or more feet contact the ground, the robot's stance is very stable because there are enough contact points to form a support polygon.

In order to generate a gait for these more complex robot models, a custom MATLAB-based tool was created to allow commanding of up to 3 feet to be in stance (ground contact) or swing (non-ground-contact) phases, at numerous time points throughout the gait cycle. The tool then generates the corresponding spine joint angles for forward locomotion, and finally visualizes the generated gait cycle using a stick model. In other words, the MATLAB-based tool allows the user to input a custom gait diagram and the program then simulates what the robot's locomotion would look like, using inverse kinematics implemented by the resolved-rate trajectory generation (RRTG) method. This MATLAB tool also allows users to modify the number of DOF in the spine to observe the effect on the gait.

Using this MATLAB tool, the actual robot gaits (joint trajectories) for the gait diagrams shown in Figure 13 were generated, and are visualized in Figures 14 and 15. Figure 14 shows the execution of the trot gait with gait diagram Figure 13a. Figure 15 shows the execution of the walking gait of Figure 13b, with up to 3 feet contacting the ground simultaneously. Figure 15 shows a walking gait that is a mix of a very stable "wave gait" where 3 feet are always in contact with the ground, and the less stable trot gait. This gait produces a reasonable walking motion that

is stable, but also relatively high-speed.

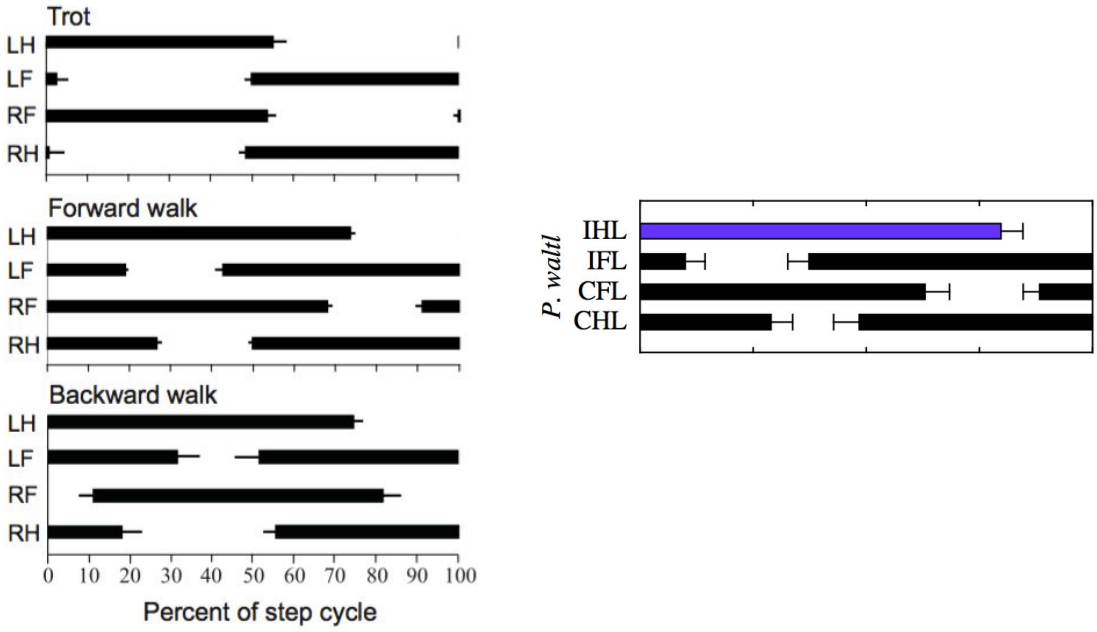


Figure 12. (a) Footfall diagrams for terrestrial stepping for various salamander species[13] and (b) forward walk for just *P. waltl* [2].

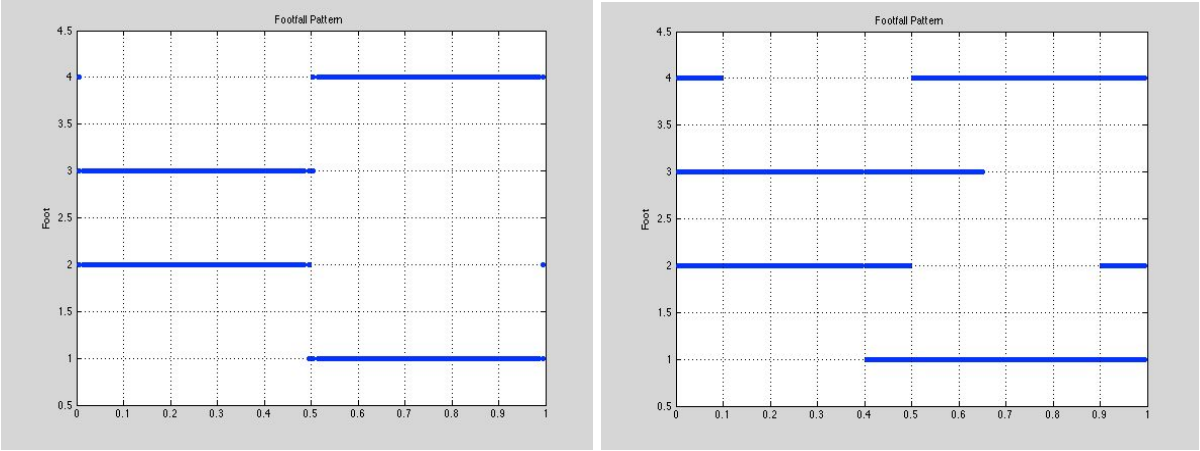


Figure 13. Footfall diagrams for MATLAB kinematics simulation for (a) trot gait and (b) walking gait.

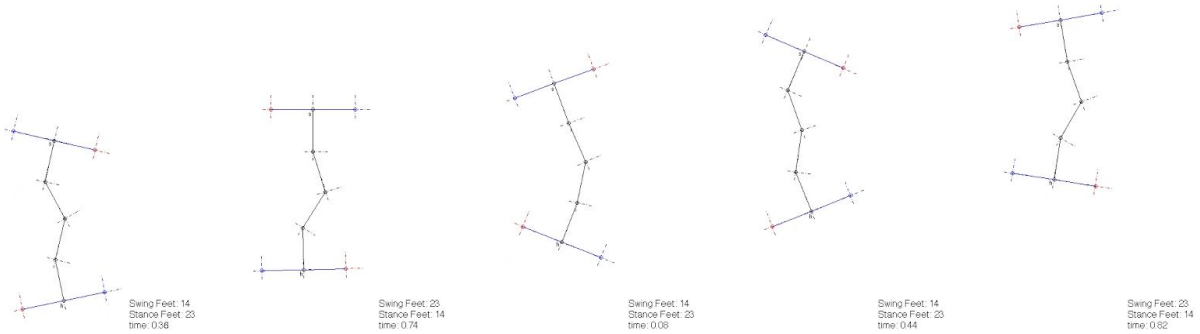


Figure 14. A trot gait kinematics simulation, time increasing to the right.

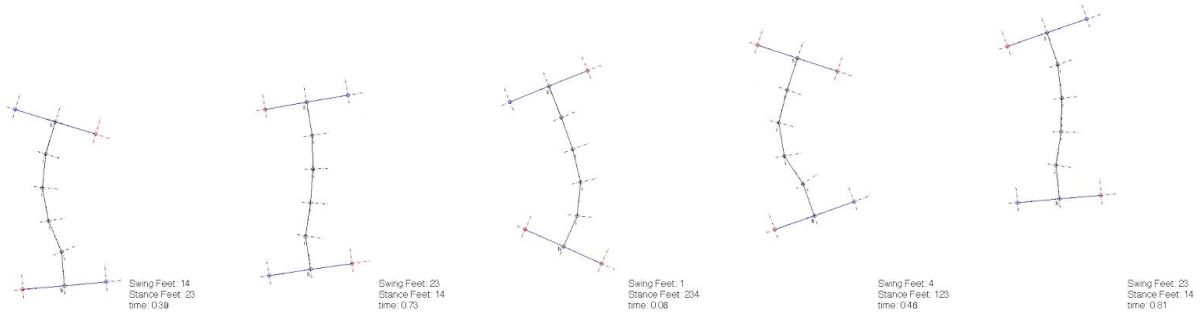


Figure 15. A walking gait kinematics simulation, with up to 3 feet on the ground, time increasing to the right.

One drawback of the RRTG-based gaits created by the MATLAB tool are that these gaits do not consider whether the center of mass (COM) resides inside the support polygon defined by the foot contact points. If the COM resides outside the support polygon, the robot stance is unstable and will tip over assuming that the robot is moving very slowly) [14]. Another drawback of the RRTG-based MATLAB tool is that MATLAB tool requires evaluating symbolic expressions that grow larger and get slower to evaluate as the spine DOF increases. So this approach is not very scalable to a highly-articulated robot with many spine and leg joints.

In order to efficiently generate stable gaits for high-DOF spine and leg quadrupedal systems, the Optragen MATLAB toolbox was used. Optragen is a transcription package that transcribes an optimal control problem into a nonlinear programming problem that can be solved

by a nonlinear programming solver. Optragen was used to reformulate the optimal control problem of generating a gait that is fast yet stable, into a nonlinear programming optimization problem that can be solved with the SNOPT optimization package. Optragen's transcription method is to parametrize the trajectories in the optimal control problem as B-splines. The optimal control problem given to Optragen to optimize was formulated as follows:

The cost function is what SNOPT attempts to minimize. The cost function for this problem contains a cost for not traveling a desired distance in a set time and a cost for any unstable stances the quadruped may assume during the gait. The gait space for a quadruped with many spine and leg joints is very large, so continuation methods were utilized to achieve sufficiently low cost function values. This means that multiple cost functions were optimized by SNOPT, one after the other. The two cost functions that were used were:

$$L_1 = L_{movement} + \int_0^{t_f} k_{g1} L_{grip}(t) dt,$$

$$L_2 = L_{movement} + \int_0^{t_f} k_{g2} L_{grip}(t) + k_s L_{stability}(t) dt,$$

where:

$$L_{movement} = \|\vec{p}_{goal} - \vec{p}_{head}\|^2,$$

$$L_{grip}(t) = \sum_{i=1}^{4} c_i(t) \|\vec{p}_i(t)\|^2,$$

$$L_{stability}(t) = \sum_{i=1}^{8} \text{contacts}(i,t) \exp(-k_{sp} d_{margin}(i,t)),$$

$$\text{contacts}(i,t) = \begin{cases} c_1 c_3, & i = 1 \\ c_3 c_4, & i = 2 \\ c_4 c_2, & i = 3 \\ c_2 c_1, & i = 4 \\ c_1 c_4 c_2 (1 - c_3), & i = 5 \\ c_4 c_1 c_3 (1 - c_2), & i = 6 \\ c_2 c_3 c_4 (1 - c_1), & i = 7 \\ c_3 c_2 c_1 (1 - c_4), & i = 8 \end{cases} \qquad d_{margin}(i,t) = \begin{cases} d_{marginp}(\vec{p}_1, \vec{p}_3), & i = 1 \\ d_{marginp}(\vec{p}_3, \vec{p}_4), & i = 2 \\ d_{marginp}(\vec{p}_4, \vec{p}_2), & i = 3 \\ d_{marginp}(\vec{p}_2, \vec{p}_1), & i = 4 \\ d_{marginp}(\vec{p}_1, \vec{p}_4), & i = 5 \\ d_{marginp}(\vec{p}_4, \vec{p}_1), & i = 6 \\ d_{marginp}(\vec{p}_2, \vec{p}_3), & i = 7 \\ d_{marginp}(\vec{p}_3, \vec{p}_2), & i = 8 \end{cases},$$

$$d_{marginp}(\vec{p}_a, \vec{p}_b) = [\vec{p}_{com} - \vec{p}_a - [(\vec{p}_{com} - \vec{p}_a) \cdot \hat{n}_{ab}]\hat{n}_{ab}] \cdot \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \hat{n}_{ab},$$

$$\hat{n}_{ij} = \frac{\vec{p}_j - \vec{p}_i}{\|\vec{p}_j - \vec{p}_i\|}.$$

The first cost function L_1 is somewhat less restrictive than L_2, because there is no stability cost. This allows SNOPT to optimize the cost function faster, and guides SNOPT in a "good" direction. It was found that only using the second cost function resulted in SNOPT getting caught in local minima and producing only mediocre results.

The stability cost $L_{stability}$ computes a cost that is very high when the COM is outside of the support polygon, and low when the COM is inside the edges of the support polygon. $L_{stability}$ is loosely related inversely to the stability margin S_M, which is defined in Figure 16. $L_{stability}$ uses the exponential function to compute a "soft min" of all the distance margins. The distance margins are the distances from the COM to each edge of the support polygon. Note that the argument to the exponential function has a negative sign, resulting in the inverse property: low stability margin results in high cost. By utilizing the exponential function, we can deal with *negative* stability margins (COM outside the support polygon). Negative stability margins lead to very high stability cost.

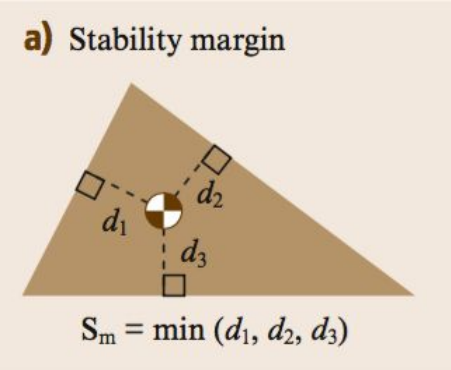


Figure 16. Definition of stability margin, taken from [14]. The distance margins are the shortest distances from the COM to each edge.

Because SNOPT uses a gradient-descent technique to optimize the cost function, the problem given to it needs to be continuous. However, a robot contacting or lifting up its foot is a discontinuous dynamics problem, because ground contact forces forces "turn on" or "turn off" based on whether the foot is contacting. To resolve this, contact variables $c_1(t), ...c_4(t)$ were created, which represent whether each foot of the quadruped is contacting the ground. 0 indicates no contact, and 1 indicates contact. Then, these c_i can be optimized by Optragen as a trajectory, in effect optimizing the footfall pattern of the robot.

The $contacts(i,\ t)$ and $d_{margin}(i,\ t)$ functions represent relevant foot contacts and distance margins, for each possible edge of the support polygon (there are 8 possible edges, so i goes from 1,...,8). Note that the support polygon changes based on which feet are contacting the ground. The various support polygon edges possible are illustrated in Figure 17. Finally, the p_i functions represent the spatial position in (x, y) coordinates of each foot i=1,...,4.

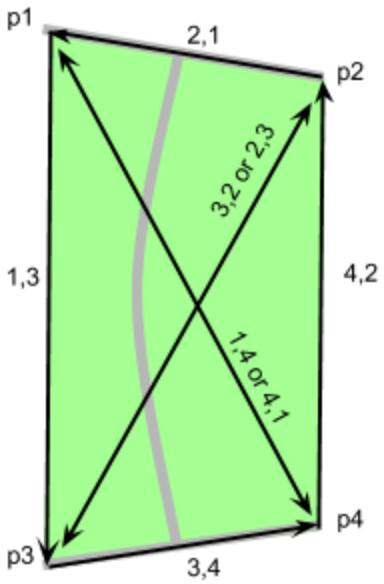


Figure 17. Possible support polygon edges for 3 or 4 feet in stance.

Optragen was given a robot kinematics model with 3 trunk DoF and 1 tail DoF for a total of 4 spine DoF, plus legs with 1 DoF each. The above cost functions were utilized to optimize a gait, including joint angles and foot contact functions $c_i(t)$. The cost function constant parameters k_{g2}, k_s were varied to observe the effect on gait. Figure 17 shows the results of this optimization. For no stability and grip costs, the gait is very fast (going offscreen), but the COM often exceeds the support polygon boundary (Figure 18, row 1). As the stability and grip cost constants are increased, the optimal gait transitions into a gait where the COM is within the support polygon. For very high stability costs, SNOPT finds a gait where all 4 feet are down, with the COM exactly in the middle (Figure 18, row 3), which is a very stable pose.

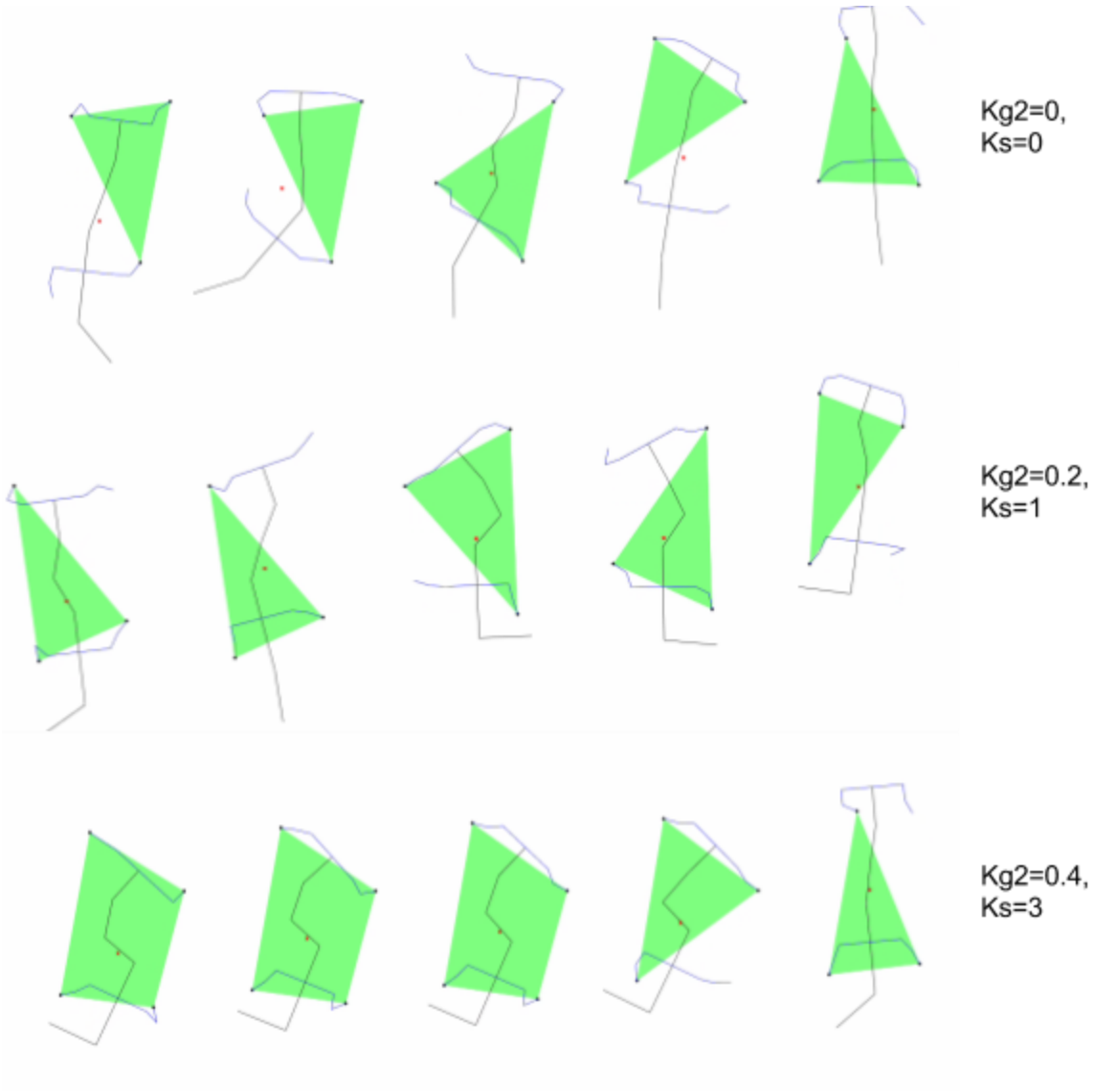


Figure 18. Optragen gait kinematics simulation for three different gaits (each row) with time increasing to the right. The COM is marked with a red x, and the support polygon is green. Feet in the stance phase are shown with black dots. Different rows show different gaits generated with various stability penalty constants.

The results in Figure 18 look like promising candidates for salamander gaits that could be implemented on the real robot. However, this kinematics approach to gait generation ignores the dynamics, or physics, of locomotion. As a result, relevant physical phenomena are not modelled, including: inertia of body elements, foot friction/slippage, and PID torque control of servo motors. Furthermore, the previous kinematics optimization was done in 2 dimensions, or SE(2),

and the real robot is in SE(3), or is 3-dimensional, so motion in the *z*-axis has not been considered yet.

4.1.4.3 Dynamics-Based Control Approaches

In order to model realistic physics phenomena such as foot slippage and motor torque limits, and to include all 3 dimensions in our gait optimization, we used the Open Source Robotics Foundation (OSRF) Gazebo robotics simulation package. Gazebo supports the universal robot description format (URDF) for description of robot models, so we created a URDF file that describes the salamander's DoF and zero configuration, as shown in Figure 19. 3D models taken from the Dynamixel website were used for realistic renderings of the servo motors and brackets. The Gazebo URDF model of the robot consists of 29 total servo motors and joints, identical to the real physical robot.

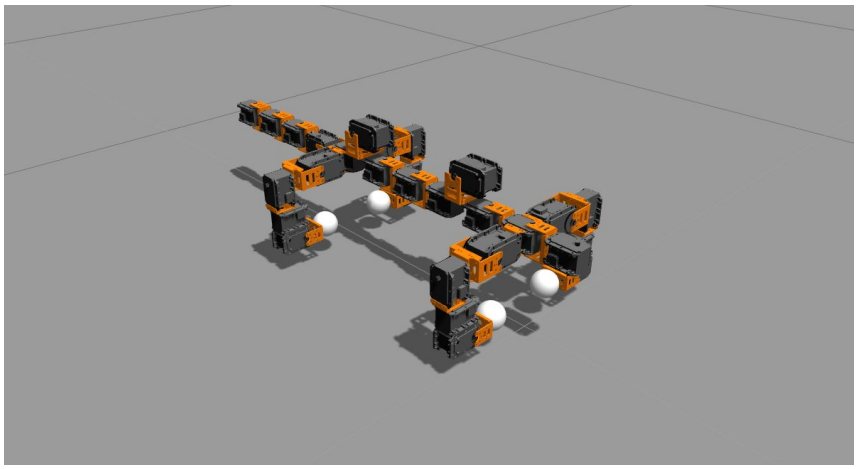


Figure 19. Robot in zero configuration.

Next, in order to simulate PID servo control for each motor in the body, a custom Gazebo plugin was created. Gazebo plugins are C++ programs that are compiled as a shared library, and are directly inserted into the physics simulation. Our custom plugin kept track of 28 PID controllers as well as the state (angle) of each motor, and applied a torque to each virtual servo

motor each physics timestep. The physics timestep for simulation was 1ms. The C++ plugin, URDF model, and associated code can all be found on the Sally/GazeboDynamicsOptimization GitHub repository: https://github.gatech.edu/Sally/GazeboDynamicsOptimization.

Using this Gazebo dynamics simulation allowed us to experiment with different feet geometries and terrain environments. Figure 20 shows a Gazebo simulation with "bird feet", or asymmetrical feet.

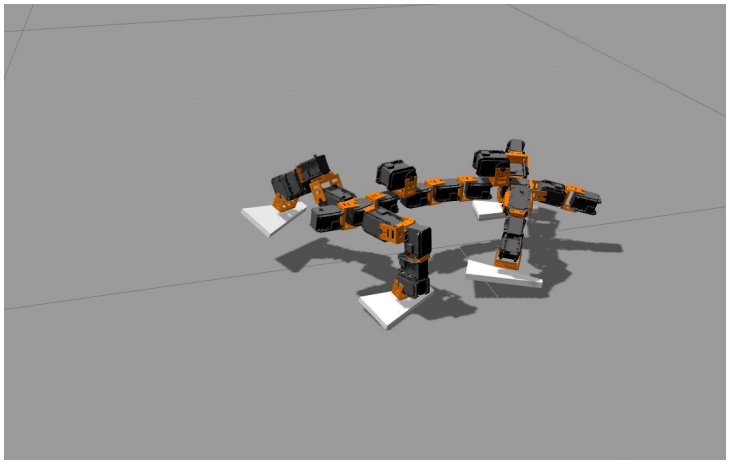


Figure 20. Gazebo simulation with asymmetrical bird-foot geometry.

4.1.4.3.1 Gait Parameterization

Without loss of generality, any gait can be parametrized by a set of splines with a sufficiently high number of knots (one spline for each controlled DoF). However, the higher the number of knots in each spline, the more parameters it takes to parametrize and the larger the gait search space. In order to keep things simple, sine wave parametrization was chosen instead of splines, for each controlled DoF. That is, each motor was assigned a trajectory $\theta_i(t)$:

$$\theta_i = A_i + B_i \sin(\omega t + \phi_i)$$

where ω is the gait frequency. Thus, the goal trajectory of each motor can be parametrized by just 3 numbers. A naïve approach to parameterizing a gait by just appending 3 parameters for

each motor, to a gait description vector $\vec{g}$, would result in a vector of length $3 \times 28 = 84$. This results in a large gait space: if only 1 bit is chosen to represent each gait description vector element, this results in a gait space of size $2^{84} \approx 1.9 \times 10^{25}$. In order to reduce the size of this gait space, we exploit some symmetries in normal walking gaits, generating a gait parameter vector of length 29, resulting in a space of size $2^{29} \approx 5.4 \times 10^{8}$. The main simplifications are that the amplitudes and biases of joints 1 and 2 of each leg are the same, and leg joint 3 is not used. The gait parameter vector is described in Table 5.

Table 5. Gait Parameter Listing

Parameter Indices (0-28)	**Name**
0-3	leg joint 1 phases (4)
4	leg joint 1 amplitude
5	leg joint 1 bias
6-9	leg joint 2 phases (4)
10	leg joint 2 amplitude
11	leg joint 4 bias
12-20	spine amplitudes (9)
21-28	spine phases (8)

Using this 29-dimensional parameter vector, we are able to describe a large range of gaits, including forward walking, backward walking, and turning, in a simple, efficient way. Future work could include higher dimensional representations for spline parametrization of each joint, enabling an even richer gait space.

4.1.4.3.2 Genetic Algorithm Optimization

Once the dynamics model had been created and the gait has been parametrized, an optimization algorithm needed to be chosen to actually generate an optimized gait. Previously for kinematics simulation, gradient-descent-based optimization had been implemented with SNOPT. While gradient descent would also probably work for dynamics simulation, one problem with gradient

descent is that it can get caught in local extrema. Because we would like to find a globally optimal gait, and because of its bio-inspired origins, a genetic algorithm (GA) was chosen as the gait optimization algorithm.

The GAlib C++ library created by Matthew Wall (http://lancet.mit.edu/ga/GAlib.html) was utilized to implement a genetic algorithm within the Gazebo plugin that was mentioned earlier. GA parameters including population size, number of generations, types of mutations, fitness function scaling, and others were adjusted to achieve good GA convergence in a relatively short time. The final GA parameters that provided good performance are shown in Table 6.

Table 6. Genetic Algorithm Parameters

Parameter	**Value**
Population size	20
Phenotype Format	Bit string
Phenotype size	29
Num. bits per phenotype attribute	16
Mutation function	Roulette wheel
Prob. of mutation	0.01
Crossover function	1-point crossover
Prob. of crossover	0.9
Fitness function scaling	sigma truncation
Max. num. of generations	100
Initial population genomes	Uniform random

GA requires an objective function to maximize. We have constructed an objective function that rewards gaits that produce body velocity close to a desired goal velocity. Our objective function also penalizes gaits that use high motor torque, and further penalizes gaits that utilize motor torques that are over the specified Dynamixel motor torque limits. This allows the GA to be aware of motor torque limits and favor gaits which are feasible by the real motors. The objective function used for gait optimization is:

$$F(\vec{g}) = -\|\xi - \xi_{goal}\| - k_1 \int_0^{t_f} \sum_{i=1}^{n} |\hat{\tau}_i(t)| dt$$

where

$$\xi = \begin{pmatrix} \vec{v} \\ \omega \end{pmatrix}, \quad \hat{\tau}_i(\tau_i) = \begin{cases} \tau_i & 0 < \tau_i < 1/2 \\ 1/2 + k_2(\tau_i - 1/2)^2 & \tau_i > 1/2 \end{cases}.$$

Note that by "velocity," we mean the generalized velocity ξ, which includes both linear and angular velocity. We specify the goal velocity ξ_{goal} in 2 dimensions, from an overhead view, so it has 3 components: linear x, linear y, and angular velocity. Also note that the motor torques τ_i are normalized so that 1 corresponds to the maximum allowed torque. Thus, the above equation shows that when the normalized torque exceeds half its rated maximum, an additional quadratic cost is added to the torque cost $\hat{\tau}_i$ to disincentivize using that much torque. Penalization constant k_2 is chosen to be large, so that even 1 motor exceeding its limits drastically increases the overall torque cost, and decreases the fitness function $F(\vec{g})$.

Figure 21b shows the optimization curve resulting from utilizing the GA and objective function described above. Figure 21a shows a similar GA with an objective function that has no torque cost. Both GAs seem to converge to a stable objective function population mean by about 30 generations. However, the GA for the objective function with torque costs continued to find better *maximum* population objective function values well past generation 30, up until generation 80. Figure 21a illustrates the random nature of the GA: depending on the initial conditions and random crossover and mutation, different populations can vary significantly. Multiple populations weren't experimented with the more complex torque-optimization due to time

constraints.

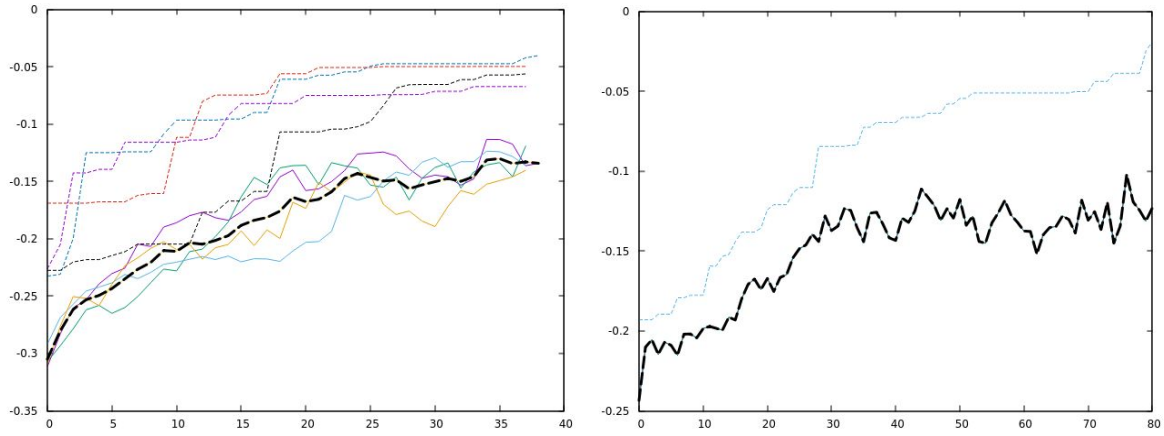


Figure 21. GA objective function (y-axis) as a function of generation number (x-axis) for (a) bird-foot gait optimization with no torque cost, (b) ball-foot gait optimization with torque cost. The solid lines indicate population mean for different populations. The black line represents the average of all population averages. The dashed lines show the maximum objective function in the population.

Each individual simulation as part of the GA was run for 5 seconds with a gait frequency of 2Hz, resulting in 2.5 gait cycles per simulation. Gazebo ran around 3-5x real-time, resulting in about 30s simulation time per generation. Parallel Gazebo instances for use on multi-core processors were experimented with to further speed up simulation time, which improved speed by about 2x.

Figure 22 shows an optimized gait for the salamander robot with bird feet, using the above GA method without torque costs. Figure 23 shows the same gait, after small modifications, running on the real robot. The GA-generated gait was modified to use smaller spine amplitudes than in simulation and to run at 40% of the simulated speed, because of motor overloading issues. Though it is hard to tell in Figure 23, the robot is wearing bird feet. The Gazebo-simulated gait execution and real gait execution are somewhat similar, but they differ in the fact that the real gait turns noticeably, while the simulated gait has a lower angular velocity

(turns less). This could be because friction properties of the walking surface are not correct in the Gazebo simulation, and could also be the result of the described gait modification.

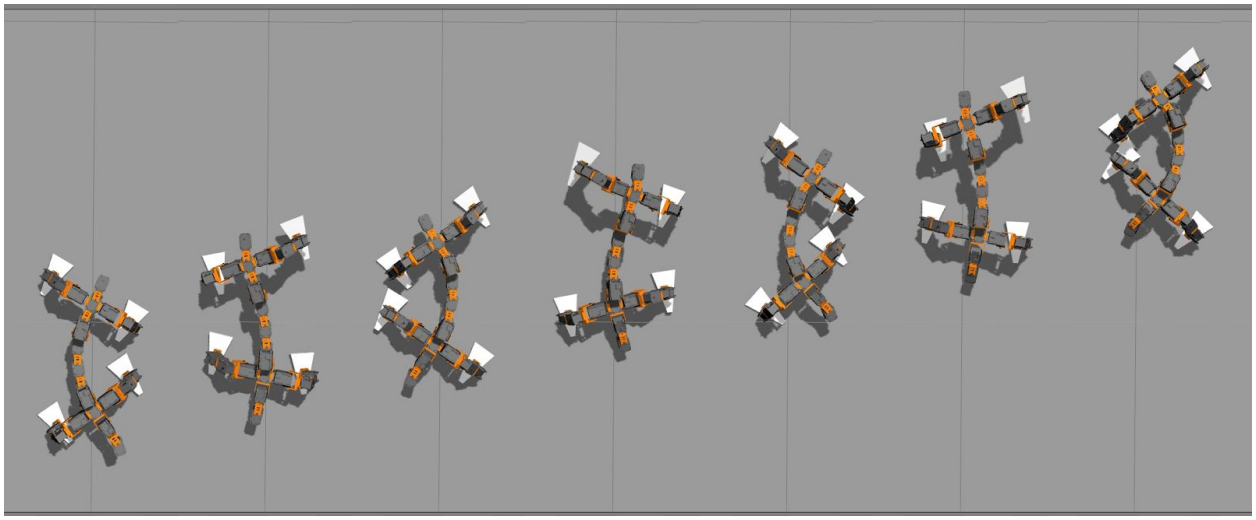


Figure 22. Gazebo dynamics gait simulation with bird feet on robot.

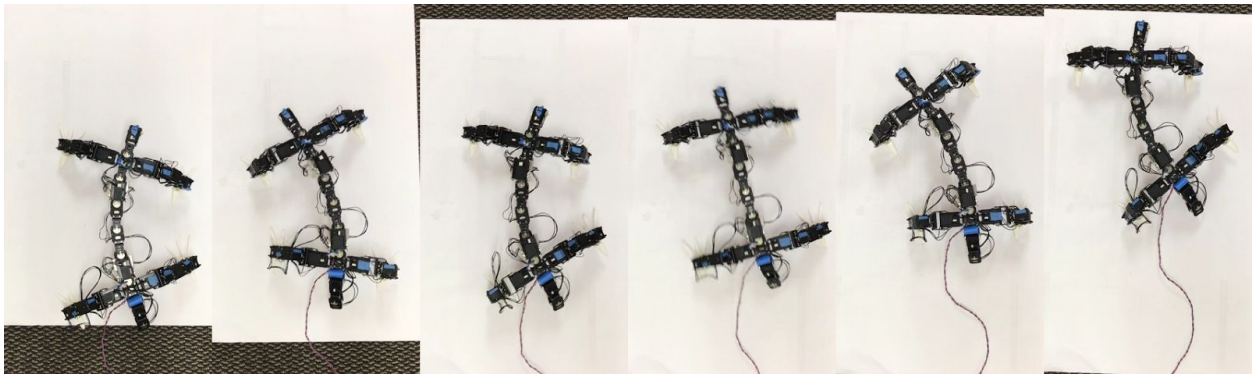


Figure 23. Real robot with modified gait similar to that in Figure 7.

The GA-generated gait including torque costs was run on the real robot at 100% speed and performed quite well (no motor overloading) with only minor gait modifications. This demonstrated that including the torque cost terms in the objective function actually reduced real motor torques.

4.1.4.4 ROS Gait Control via Action Client/Server Interface

In order to actually send the desired trajectories to the servo motors, the ROS actionlib package

was used. A custom gait controller node written in Python was created to send trajectory messages to the actionlib Action Server. The Action Server then manages the trajectory execution, including timing of sending motor control packets over the serial line to individual motors. The Action Server interface is very easy to use, so it is simple to create a custom Python controller node which sends any desired trajectory to the robot's motors.

4.2 Codes and Standards

4.2.1 Disaster Robotics

In 2005, the U.S. Department of Homeland Security and the National Institute of Standards and Technology (NIST) developed a suite of standard test methods to "quantify key capabilities of robots for emergency response and other hazardous applications" [11]. The result was DHS-NIST-ASTM: "International Standard Test Methods for Response Robots measures robot maneuvering, mobility, manipulation, sensing, endurance, radio communication, durability, reliability, logistics, and safety for remotely operated ground vehicles, aquatic vehicles, and small unmanned aerial systems in FAA Group I under 2 kg (4.4 lbs)" [11].

One relevant testing standard for the salamander robot in question is the "Disaster City" standard test bed, as shown in Figure 24. The testbed begins with flat, sloped terrain, and slowly works up to real rubble of various dimensions. The challenge for the robot is to make it from beginning to end of the test bed. Gravel and inclined planes were used to test the robot's speed and capabilities rather than the aforementioned test bed.

Figure 24. NIST "Disaster City" Standard Test Bed from [11]

4.2.2 Interfacing

The main interfacing standards utilized in this salamander robotic system are summarized in Table 7.

Table 7. Communication Standards for the Robot Salamander System

Standard	Features	Components Utilizing
Asynchronous TTL Serial (Transistor-Transistor Logic)	Up to 1 Mbps transmission speed Bidirectional communication 3.3 or 5V No clock signal	Servomotors
USB (Universal Serial Bus) 2.0	480 Mbps Master/slave relationship No clock signal	Communication to interfacing board

4.3 Constraints, Alternatives, and Tradeoffs

4.3.1 Constraint: Weight/Number of DOF

One of the most important constraints for the robot salamander system is the number of servomotors, or DOF. The larger the number of DOF, the more flexible and robust the robot can

be to rough terrain. However, the larger the DOF, the heavier and more unwieldy the robot becomes. In addition, the longer the motor chain, the higher the torque requirements. So, the ideal design would be not too slow and heavy, but with enough DOF to be flexible enough to traverse difficult terrain.

4.3.2 Tradeoff: Tethered vs. Untethered

Eliminating a tether and utilizing an on-board battery would free the robot from the distance-limitations and excess drag force of a power tether. A data tether could be eliminated by using a wireless network such as WiFi. On the other hand, a power tether would be beneficial because it provides unlimited operating life and reduced on-board weight. A battery would significantly increase robot weight and volume. The weight contributions between an on-board battery vs. pulling a long tether are nearly equal, and both limit operating range, so cost and complexity of each solution was the main determining factor. The tether is much cheaper and less-complex solution than utilizing a large battery pack and associated battery monitoring system.

5. Schedule, Tasks, and Milestones

A Gantt chart showing the actual timeline and tasks that were completed is presented in Appendix A. The team was divided into three subteams: mechanical, control, and interfacing. These teams worked in parallel throughout the course of the semester as shown in the Gantt chart. The organizational chart for the team is shown in Figure 25.

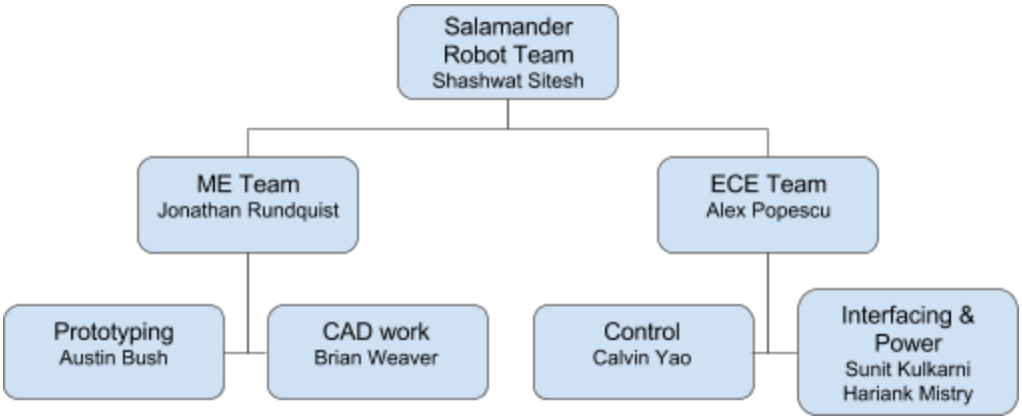


Figure 25. Team organizational chart

Each team member contributed to their subteam, and the subteam contributed to the whole. The team members and their notable contributions are listed below:

Austin Bush: Mechanical team, helped with overall design and fabrication. Built custom brackets for vertical motors and tail, designed and built a dynamic mechanical foot, and helped in testing of the robot.

Sunit Kulkarni: Controls team, helped with preliminary Optragen modeling. Also attempted interfacing with a camera module at the front of the robot. This setup was structurally sound but the camera malfunctioned before the final demonstration rendering it useless. Replacing the camera in the enclosure for future research should solve this issue.

Hariank Mistry: Interfacing team. Created 2 tethers for the robot and learned about the ROS interfacing for motors.

Alex Popescu: ECE team lead, controls team. Led ECE team to develop fully functional robot from simulation to integration and testing. Created kinematic and dynamics models of the robot and used optimization algorithms, including a genetic algorithm, to generate gaits optimized for speed and low torque usage.

Jonathan Rundquist: Mechanical team, overall design and fabrication. Designed leg and shoulder connections, helped with custom bracket fabrication, and helped in testing and gathering data.

Shashwat Sitesh: Interfacing team, engineered protocols to develop the interface through which the ROS software can effectively control the dynamixel servos. Updated firmware and baud rate for each motor and developed the circuitry for effective recognition and communication.

Brian Weaver: Mechanical Team, Designed and fabricated feet. Fabricated two sets of feet and prototyped another. Built method to easily exchange working feet on robot.

Calvin Yao: Controls Team, helped with kinematic and dynamic modeling of the robot. Helped with joystick integration and testing of the robot.

6. Project Demonstration

The project demonstration took place in Georgia Tech's IVALab and tested several gaits on the robot. A simple joystick controller was created for the purposes of demonstration, and allowed commanding forward and backward gaits from the robot with a range of speeds. 5 routines, including 2 gaits were demonstrated:

1. "Stand up" routine
2. "Lay down" routine
3. "Look around" routine (2D joystick control for pointing forward-looking camera left/right/up/down)
4. Simple sine-wave walking controller (forward/backward joystick control)

5. 29-dimensional gait optimized by genetic algorithm (forward joystick control only)

6.1 Demonstration Steps

1. Give joystick controller to operator
2. Operator can choose the direction and speed of locomotion using the joystick, or the direction to point the forward-looking camera in the “look around” routine
3. The robot responds

6.2 Specifications and Modules

The specifications originally proposed for this project did not change significantly. Originally we had desired a turning gait, a maximum traversal speed of about 10m/minute, and a traversable terrain height of greater than 5 cm. These goals, however, became stretch goals for the project, once the demand of time and effort needed for simple walking was understood. A turning gait was not implemented at all, but because of the nature of normal walking, data for the other two specifications was gathered, but goals not met. the maximum traversal speed was 4m/minute, and the traversable terrain height was 3 cm.

This project is organized into three modules; Mechanical, Interfacing, and Controls. Mechanical prototyped various spine lengths, leg configurations, and foot designs in the CAD models before fabricating. Some foot prototypes were also created to test the structural integrity of the foot connections. Interfacing prototyped each motor, wire, and tether subsystem before creating the overall system to power and control the robot. Lastly, Controls prototyped many different types of gait using simulation software and Matlab.

Videos of testing over gravel and carpet can be seen here:

https://www.dropbox.com/sh/wetbxzfp4nkfh8z/AACUnQ-ilbYv591N1Do5TvRQa?dl=0&previe

w=salamander_run2_rgb.avi

https://www.dropbox.com/sh/wetbxzfp4nkfh8z/AACUnQ-ilbYv591N1Do5TvRQa?dl=0&preview=2017-04-25+17.27.09.mov

https://drive.google.com/file/d/0BwphcFNORi5OMFRYdnBnYS1DMFU/view

7. Marketing and Cost Analysis

7.1 Marketing Analysis

Disaster robots are not routinely sold in quantity, as this field is still under research and any completed robots are used by governments, companies, and universities. According to [12], government agencies and humanitarian organizations dealing in emergency response can expect to pay in increments of $50,000 for "small ground robots." However, larger caterpillar robots are valued in the range of $100,000. The market for disaster robots is also small: only 37 deployments have been reported for disaster robots since 2001 [1]. Clearly, disaster robots comprise a small and expensive market. The total development costs of our salamander robot (detailed in the next two sections) fit in this high price range, as expected.

7.2 Cost & Funding Analysis

The final total parts cost for this salamander robot is $7,307.10, as shown in Table 5. The most costly components are the Dynamixel servomotors, which cost $239.90 each for the Dynamixel MX-28AT model. All individual component components and associated costs are displayed in Table 8.

Table 8. Parts Costs

Item	Unit Price	Quantity	Total Price
Dynamixel MX-28AT Servomotor (Dr. Vela)	$239.90	24	$5,736.00
Dynamixel MX-64AT Servomotor (Dr. Vela)	$258.25	6	$1,549.50
Tether materials (senior design lab)	$10.00/m	5	$0.00
Total Parts Cost			$7,307.10

Labor costs were calculated assuming $50 per hour. Gait analysis/control and testing are expected to have the most labor-hours due to the difficulty of implementing and testing the walk cycles of the robot. The breakdown of the development costs are listed in Table 9.

Table 9. Development Costs

Task	Hours	Labor Cost
Design	120	$6,000
Fabrication	40	$2,000
Assembly	80	$3,100
ROS/Microcontroller Programming	100	$5,800
Gait Analysis/Control	160	$9,800
Testing	200	$3,600
Meetings	32	$10,00
Reports	56	$4,800
Demo Preparation	56	$5,200
Total Labor Cost	**844**	**$50,300**

Assuming fringe benefits are 30% of labor and overhead on materials, and overhead costs are 120% of parts, labor, and fringe benefits, the total development cost of this salamander robot is $159,933.62, according to Table 10.

Table 10. Total Development Costs

Development Component	Cost
Parts	$7,307.10
Labor	$50,300.00
Fringe Benefits (30% labor)	$15,090.00
Subtotal	$72,697.10
Overhead (120% of parts, labor, fringe)	$87,236.52
Total Development Cost	$159,933.62

Funding for our project came from two main sources: Dr. P. Vela's lab, as well as the ECE department. The amount from each funding source is shown in Table 11. Funding from the ECE department was $800.00.

Table 11. Funding Sources and Amounts

Funding Source	Amount
Dr. P. Vela/IVALab	$7,307.10
ECE Department	$800.00
Total Available Funds	$8,107.10

The available funding of $8,107.10 was enough to cover the parts costs of $7,307.10.

7.3 Profit Analysis

As described in Section 7.1, medium-size ground-based disaster robots near the scale of our salamander robot are routinely sold for $100,000 and more. The total development costs of our robot are estimated at $159,933.62. So, if the proposed salamander robot is sold for $200,000.00, the profit will be $40,066.38. This price is in the normal competitive price range for disaster robots of this scale, and still yields a reasonable 25.1% net profit margin.

8. Conclusion

Supermander went through several iterations throughout the design process as there were very loose initial design specifications. Therefore, the team continuously experimented with various design standards, gaits, body configurations, foot types, and testing environments. This has resulted in a salamander-inspired robotic testbed with a complex system of servo motors, 29 degrees of freedom, three sets of feet, front facing camera, multiple adjustable gaits, and an integrated system for user controls.

The first specifications were to walk over environments ranging from flat to rubble to an inclined grade, with a stretch goal to climb stairs. The robot was then sized around these capabilities. Previous work also influenced the design of Supermander, as another similar robot named “Pleurobot” gave insight for motor orientations and leg designs. During construction, the

realistic goals for the behavior of this robot became apparent, and so it was decided upon that walking at all by the end of the timeline would be a success. Research and design were conducted by each subteam for most of the project length. The mechanical team designed the motor layout of the spine and legs, designed the feet, and fabricated the robot. The interfacing team designed the communications with the robot, the power specifications, and the electrical hardware. And the controls team designed and optimized the various gaits for the Supermander.

The final product is a working robot that can walk forward and backwards with both automated (genetic algorithm) and hand maneuvered gaits. As per the test results, supermander performed well in mild terrain ranging from flat tile to gravel to an incline of 3 degrees. The various feet all seem to work best in a specific environment, and with a specific, optimized gait. The fastest feet, however, were the simplest for the tested gaits and environments, the ball feet, at a max speed of 0.3 ft/s.

Supermander serves as an excellent testbed for various optimized gait patterns, foot designs, and terrain. Electromagnetic feet to climb metal in buildings; many more types of gait need to be explored such as climbing stairs, turning, ascending, and descending rough terrain. Future improvements can also include structural and functional changes, such as a z-axis degree of freedom in the spine, a smaller body to fit in cracks in rubble, microphones and speakers, and a battery pack and remote controller for greater mobility. Supermander is a strong testbed to drive research in the area of disaster robotics and will help pave the way for future innovations.

9. Acknowledgements

The team would like to thank Dr. Vela for his advice and guidance throughout the semester. We would also like to thank him for providing us the unique opportunity to work on this project.

10. References

[1] R. R. Murphy, S. Tadokoro and A. Kleiner, "Disaster Robotics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Springer-Verlag Berlin Heidelberg, 2008, pp. 1577-1604.

[2] K. Karakasiliotis, R. Thandiackal, K. Melo, T. Horvat, N. K. Mahabadi, S. Tsitkov, J. M. Cabelguen and A. J. Ijspeert, "From cineradiography to biorobots: an approach for designing robots to emulate and study animal locomotion," Journal of the Royal Society Interface, vol. 13, 2016. doi: 10.1098/rsif.2015.1089

[3] J. K. Hopkins, B. W. Spranklin, and S. K. A. Gupta, “A Survey of Snake-Inspired Robot Designs,” vol. 4, no. 2, 2009.

[4] J. Borenstein, G. Granosik, and M. Hansen, “The OmniTread Serpentine Robot - Design and Field Performance,” in Proceedings of the SPIE Defense and Security Conference, Unmanned Ground Vehicle Technology VII, G. R. Gerhart, C. M. Shoemaker, and D. W. Gage, Eds., Orlando, FL, May 2005, pp. 324–332. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=864275

[5] A. Johnson, C. Wright, M. Tesch, K. Lipkin, and H. Choset, “A Novel Architecture for Modular Snake Robots,” Tech. Rep., 2011.

[6] A. Chang, M.M. Serrano, P.A. Vela. “Incorporating frictional anisotropy in the design of a robotic snake through the exploitation of scales,” IEEE International Conference on Robotics and Automation (ICRA), 2015.

[7] "Rescue Robots / Machines Play Vital Roles in Disaster Relief." Trends in Japan. Trends in Japan / Sci-Tech, Sept. 2010. Web. 27 Jan. 2017. <http://web-japan.org/trends/09_sci-tech/sci100909.html>.

[8] P. Lin, H. Komsuoglu, D.E. Koditschek. “Sensor Data Fusion for Body State Estimation in a Hexapod Robot With Dynamical Gaits,” IEEE Transactions on Robotics, Vol. 22, No. 5, 2006.

[9] ROBOTIS e-manual, "MX-28AT / MX-28AR" [Online]. Available: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28at_ar.htm.

[10] A. J. Ijspeert, *A robot that runs and swims like a salamander*, TED Talk, 2016.

[11] Intelligent Systems Division, "Guide for Evaluating, Purchasing, and Training with Response Robots Using DHS-NIST-ASTM International Standard Test Methods," National Institute of Standards and Technology, 2011.

[12] "Tornadoes, Natural Disasters: How High-tech Robots Help in Search and Rescue." *The Washington Post*. WP Company, 25 May 2011. Web. 27 Jan. 2017. <https://live.washingtonpost.com/robot-technology-tornadoes-joplin-missouri.html>.

[13] Karakasiliotis, Konstantinos. "Legged Locomotion with Spinal Undulations." Diss. École Polytechnique Fédérale De Lausanne, 2013. Print.

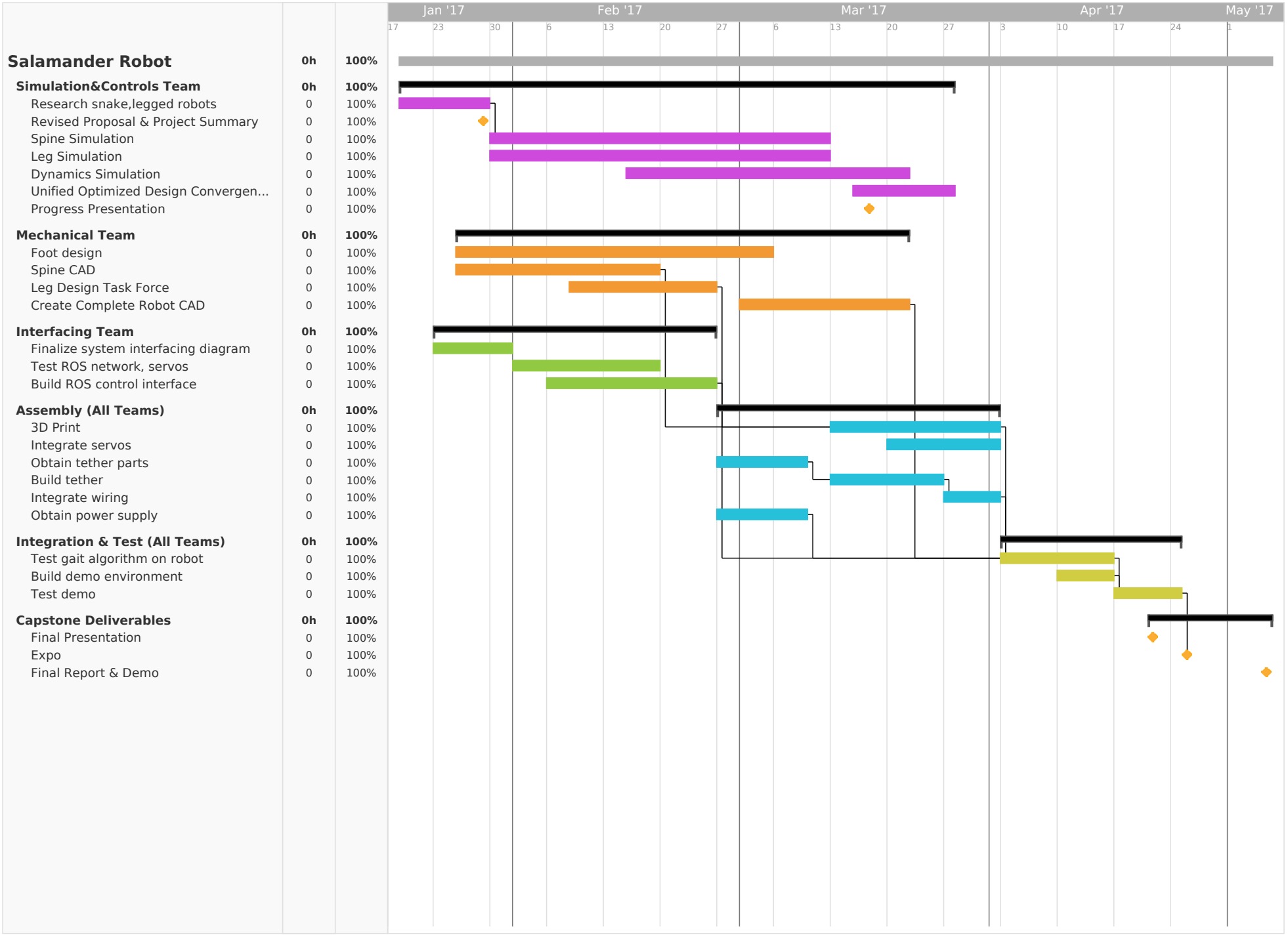
[14] Kajita, Shuuji, Bernard Espiau, and Oussama Khatib. "Legged Robots." *Springer Handbook*

of Robotics. Ed. Bruno Siciliano. 1st ed. N.p.: Springer-Verlag Berlin Heidelberg, 2008. N. pag. Print.

Appendix A: Gantt Chart

The Gantt chart for this project is on the following page.

teamgantt
Created with Free Edition



Salamander Robot	**0h**	**100%**
Simulation&Controls Team	**0h**	**100%**
Research snake,legged robots	0	100%
Revised Proposal & Project Summary	0	100%
Spine Simulation	0	100%
Leg Simulation	0	100%
Dynamics Simulation	0	100%
Unified Optimized Design Convergen...	0	100%
Progress Presentation	0	100%
Mechanical Team	**0h**	**100%**
Foot design	0	100%
Spine CAD	0	100%
Leg Design Task Force	0	100%
Create Complete Robot CAD	0	100%
Interfacing Team	**0h**	**100%**
Finalize system interfacing diagram	0	100%
Test ROS network, servos	0	100%
Build ROS control interface	0	100%
Assembly (All Teams)	**0h**	**100%**
3D Print	0	100%
Integrate servos	0	100%
Obtain tether parts	0	100%
Build tether	0	100%
Integrate wiring	0	100%
Obtain power supply	0	100%
Integration & Test (All Teams)	**0h**	**100%**
Test gait algorithm on robot	0	100%
Build demo environment	0	100%
Test demo	0	100%
Capstone Deliverables	**0h**	**100%**
Final Presentation	0	100%
Expo	0	100%
Final Report & Demo	0	100%