GAIT SYNTHESIS AND RESILIENT TASK PLANNING FOR AMBULATING SOFT ROBOTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Scott Bowes Hamill

May 2022

GAIT SYNTHESIS AND RESILIENT TASK PLANNING FOR AMBULATING SOFT ROBOTS

Scott Bowes Hamill, Ph.D.

Cornell University 2022

Soft robots are capable of motions and environmental interactions that are not typically achievable by rigid robotic systems. In addition, the materials and fabrication methods used often result in actuators that can be designed and produced quickly. The combination of these attributes presents an opportunity for developing rapidly manufacturable and versatile soft robots capable of addressing a variety of environments and tasks. However, taking full advantage of these properties requires tools to automatically synthesize motions and controllers for these systems while also addressing the potential for actuator failure. This work addresses the challenges of motion and control synthesis for ambulating soft robots. First, we present an algorithm for synthesizing gaits for arbitrarily composed, modular soft robots. The algorithm, which makes no assumptions about dynamics or specific soft material models, is demonstrated on two different composed robots. Second, we present a framework for synthesizing controllers for multigait soft robots that are resilient to actuator failure. This abstraction, which utilizes Linear Temporal Logic (LTL) to encode multigait behavior and a sensor based abstraction of actuator performance, is demonstrated on a physical robot. Finally, we present methods for modeling ambulating soft robots as a precursor towards gait synthesis.

BIOGRAPHICAL SKETCH

Scott Hamill was born in Houston, Texas and completed his undergraduate degree at The University of Texas at Austin in 2008. He completed his M.S. Degree from The University of Texas at Austin in 2014 and joined the Verifiable Robotics Research Group in the fall of that same year. His research interests are in gait synthesis and the control of walking soft robots. He spends his time breaking things rigorously in pursuit of his degree.

For Anna and Ewan.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Hadas Kress-Gazit, for her guidance, input, and encouragement over the years. I would also like to thank my committee members, Dr. Rob Shepherd and Dr. Mark Campbell, for their technical expertise and constructive comments on my work.

Further, I'd like to thank the members of the Verifiable Robotics Research Group and the rest of the members of the Autonomous Systems Lab. Research cannot progress in a vacuum, and I would like to thank my fellow researchers for their continued support and input during my time at Cornell. In addition, I'd like to thank Dr. Marcia Sawyer, Katrina Overton, and Lyn Park, whose hard work and dedication underpin the entirety of the MAE department.

Lastly, I'd like to thank my family, especially my wife, Anna, whose infinite patience and unwavering support made all of this possible.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Soft robots and actuators, fabricated from compliant, often extensible materials, are capable of unique motions and environmental interactions that are typically unachievable by rigid robotic systems. The variety of materials and methods of manufacture used to create soft actuators such as cast elastomers [29][47][77], 3D printed elastomers [56][46], sculpted foams [2][64], bonded plastic and fabric [60][49][52][71], laser cut plastic [1], and sewn fabric [8][6], provide an expansive design space from which to create soft robots capable of novel deformations. This design space has been leveraged to create soft robotic systems capable of motions such as walking [40][17], swimming [43], inching [41][69], and jumping [67], as well as robots capable of multiple types of ambulatory gaits [61]. Further, soft actuators have been integrated into hard robotic components, creating hybrid systems that are able to leverage the strengths of both types of systems [65], [38].

The expansive design space of motions and behaviors of soft robots, coupled with the range of manufacturing methods, allows for the development of libraries of soft actuator components that may be used to compose soft robots (e.g. [50]). However, leveraging libraries of components requires methods to synthesize motions and controllers for these composed systems.

This work addresses the challenges of generating motions for composed soft robots, and we present three main contributions related to gait and controller synthesis for soft robots. The first, presented in Chapter 2, is an algorithm for synthesising gaits for modular soft robots. This algorithm does not require a dynamic model of actuator behavior or models of soft material mechanics. We

demonstrate gaits synthesized with this algorithm on two different, composed robots. The second contribution, presented in Chapter 3 is a framework for synthesizing gait controllers for soft robots that are resilient to actuator failure. This framework is comprised of a sensing based abstraction of actuator health and robot multigait behavior encoded in Linear Temporal Logic (LTL) that is then used to synthesize reactive controllers. Synthesized controllers are subsequently demonstrated on a physical robot. The final contribution, presented in Chapter 4 is two different methods for modeling soft robots and actuators that, unlike the contributions presented in Chapter 2, consider actuator dynamics, specific deformations, and friction forces. These performance of these modeling methods are compared on two different physical soft robots. We present in Chapter 5 a discussion of utilizing the models presented in Chapter 4 for gait synthesis.

CHAPTER 2

OPEN LOOP GAIT SYNTHESIS FOR MODULAR SOFT ROBOTS

2.1 Introduction and Background

As discussed in the introduction, taking advantage of the large design space of soft actuators requires techniques to automatically synthesize motions for soft robotic systems. This chapter, which presents the work in [25], addresses the problem of synthesizing gaits for modular soft robots with a focus on soft robotic systems that can be arbitrarily composed from libraries of modular components.

The gait synthesis problem is primarily one of motion planning: given a robot and knowledge of the kinematics and/or dynamics of the system, generate a sequence of limb and body motions that result in a desired chassis displacement. This problem has been widely studied for rigid robot systems and a variety of methods have been proposed for generating and regulating limb and body motions of ambulating systems. These include techniques related to geometric mechanics that exploit the relationship between robot body shape and propulsion (e.g. snake robots moving through granular media) [15][79], central pattern generators (CPGs) which regulate limb coordination for specified gait patterns [42][59], evolutionary algorithms and reinforcement learning [21] [55][72], footstep planning methodologies which account limb kinematics and terrain properties [16][14][23], and dynamic gaits (e.g. hopping) that exploit the underlying dynamics of the robotic system [74][58][24][63].

In comparison, the central challenge in generating motions for soft robots

is the difficulty in modeling soft material deformations, i.e. accurately capturing the critical modes of deformation of compliant structures, which includes modeling the ground contact forces generated by soft materials. Some of the aforementioned methods of gait synthesis that do not rely as heavily on accurate dynamic models have been successfully applied to soft robotic systems, such as CPGs [70][68], evolutionary algorithms [11][12][9], and reinforcement learning [73][30]. In addition, several authors have addressed the the problem of generating dynamic models for soft actuators. For example, the authors of [28], [18], and [66] developed modeling methods based in finite element analysis (FEA) for highly compliant materials that were then used to enable gait generation or control of ambulating soft robots, and the authors of [44] created a dynamic model of a soft robot arm that relied on geometrically consistent (e.g. circular) cross sections for which they they generated motions using trajectory optimization.

While effective and capable of addressing a range of complex morphologies, the aforementioned methods of gait synthesis are either highly specific to the composition of the system (e.g. FEA, evolutionary, reinforcement learning approaches, consistent and simple cross sections) or still require expert input (e.g. CPG methods). In this chapter, we present a more generalized methodology for synthesizing gaits for soft robots, specifically gaits for robots composed from a library of components. In this case, we do not require an explicit model of actuator dynamics, nor do we require an explicit model of actuator deformations. Relying solely on knowledge of the unloaded actuator kinematics, we present a graph search based synthesis method for synthesizing gaits.

2.2 Technical Approach

The actuators considered in this work are pneumatic and multi-input. The geometry of each actuator is controlled by regulating the air pressure in each pressure chamber, and for a given chamber pressure for each of an actuator's inputs, the actuator assumes some steady-state deformation, thereby producing some force and torque. In this work, we assume that the pressure is controlled by a three-way solenoid valve such that the actuator pressure chamber is either exposed to a common rail pressure or to atmosphere (vented), and that the valve state is modeled by a Boolean variable, i.e. the valve is either *open* or *closed*. The Boolean vector of all valve settings (*open*/*closed*) is referred to as the valve state, s. The set of all valve states is denoted S. The robot assumes a discrete, steady state configuration for each valve state, and we assume that the steady state geometry is known. By evaluating the geometries of two different configurations, we can estimate the motion a walking robot may undergo when transitioning between the two valve states. The approach in this work is to evaluate the steady state geometries of the discrete configurations of the robot and then to determine a repeatable sequence of configurations expected to produce a desired chassis motion.

2.2.1 The Ground Contact Polygon and Stability

For each valve state, s_i, the robot assumes a steady state geometry and we assume that the unloaded geometry, i.e. the contact-free curvature of each soft actuator with respect to a chassis-fixed coordinate system, is known. In this work we assume that each actuator contacts the ground surface at one location,

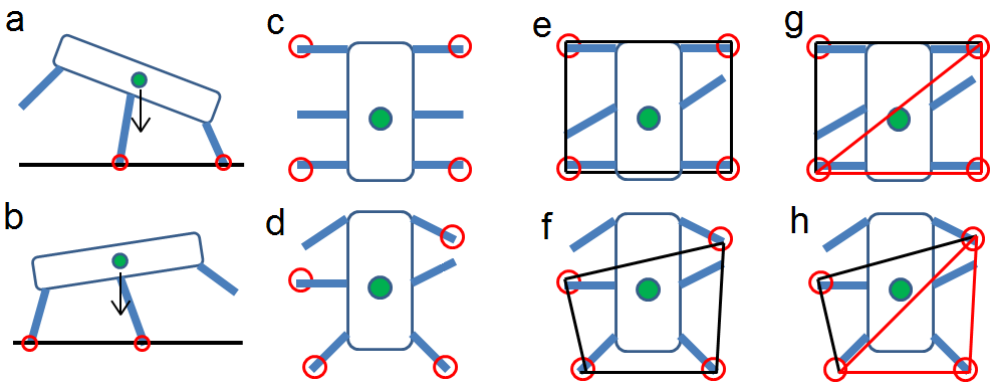

Figure 2.1: Robot poses and contact polygons. Images (a) and (b) show a side view of two different poses for a robot with the same configuration. Images (c)-(h) depict a top down view of two different robot configurations. Images (c) and (d) depict the ground contact points (circled in red) of example poses 1 and 2, respectively. Images (e) and (f) depict the ground contact polygon outlined in black for both poses. Images (g) and (h) depict the shared vertices of the contact polygons, outlined in red. The center of gravity of the robot is depicted as a green dot.

referred to as the end effector (EFF). A robot with n total pressure chambers has 2^n valve states, each of which may have more than one stable pose as shown in Figure 2.1(a-b). A pose is stable if the projection of the center of gravity (CG) of the robot onto the ground surface lies within the the contact polygon - that is, the the polygon formed by the points of contact between the actuators and the ground surface, shown in Figure 2.1(c-f). Each actuator is assigned an index, i, and we define c to be the set of indices of actuators in contact with the ground surface. For consistency with notation later in this document, we deviate from the publication [25] slightly in our definition of a pose, which we define to be the pair $p = \{s, c\}$ such that $s \in S$ and c is stable for s.

For example, assume the actuators shown in Figure 2.1(c) are labeled $\{1, 2, 3, 4, 5, 6\}$ in a clockwise manner starting from the top right actuator, and those actuators in contact with the ground surface have EFFs circled in red. In this case, $c = \{1, 3, 4, 6\}$. In the case of Figure 2.1(d), $c = \{1, 3, 4, 5\}$.

2.2.2 Predicting Robot Motion

We define the translation and rotation that occurs during a transition between two stable poses as t and r respectively. For a given pair of poses, $p = \{s, c\}$ and $p' = \{s', c'\}$, we estimate the chassis translation, t, and rotation, r, by evaluating the displacement of those actuator end effectors that remain in contact with the ground surface in a chassis fixed frame. That is, we evaluate the motion of those common contact points defined by the commonly in-contact actuators, $k = \{c \cap c'\}$ relative to the robot chassis. This is depicted in Figure 2.1(g-h). We do so by comparing the contact polygon b and post transition polygon b' as defined by the EFF locations of actuator indices k in poses p and p', respectively. We utilize the Kabsch algorithm [31] to compute the optimal rotation matrix, R, that maps the pre-transition polygon vertices, b, to those of the post transition polygon b' and subsequently assume that Euler angles that comprise R describe the chassis rotation experienced during the transition. Further, the model assumes that the translation of the chassis is accurately described by the displacement of the centroids of b and b'.

We assume a transition between two poses is valid if one of two conditions are met:

1. (Condition 1) The pre and post transition polygons are stable w.r.t. the actuator indices in k. That is, the contact polygon formed by actuator indices k is stable for valve setting s and for valve setting s'.
2. (Condition 2) The projection of the center of gravity of the robot is estimated to move outside the contact polygon of the pre-transition pose but into the contact polygon of the post transition pose. In other words, the

robot becomes unstable during the transition but "falls" into the post transition contact polygon.

2.3 The Gait Graph and Gait Synthesis

We define a gait as a repeatable sequence of poses, $\{p_0, p_1, ..., p_i\}$, and subsequently the controller for a given gait is the valve state sequence $\{s_0, s_1, ..., s_i\}$. Valve states are set at a fixed time step interval. We then pose the problem of synthesizing a gait for a given robot as a graph search problem. We formulate a graph, $G = \{V, E\}$, wherein each node $v \in V$ of the graph represents a stable pose $v = p$, and each edge $e \in E$ represents a transition between two stable poses, $e = \{p, p', t, r\}$. Previous work has utilized graph-based methods to search for gaits as in [35], but the authors of that work construct a graph from a corpus of motion capture data while we construct the graph by analyzing actuator geometry.

Gait synthesis is a process of three parts: selecting a desirable transition from E about which to form a gait, e^*, assigning edge weights to all other $e \in E$, then performing a shortest-path graph search connecting the post-transition pose in e^* to the pre-transition pose to create a cycle. In this work we select the edge e^* w.r.t. a given input vector from the user such that the angular difference between e^* and the given input vector is minimized, and we subsequently assign assign to each edge in E the weight $(1 - cos(\alpha))^4$, where α is the angle between the vectors t and t^*. An arbitrarily small weight is assigned to edges in the case where α is zero, in this case 0.0001, and a similarly small weight assigned to edges with no chassis motion (i.e. no chassis motion, $t = 0$), in this case 0.01. A

gait may then be synthesized by temporarily removing e^* from the graph and performing shortest-path graph search from v^* to v'^*. This process is detailed in Algorithm 1.

Algorithm 1 Building a Graph of Robot Poses and Motions

procedure ADDSTATE(s,G(V,E))
  for each unique pose p for state s **do**
    $L \leftarrow$ set of coordinates of the endpoints of the actuators in p
    $Q \leftarrow$ plane defind by the points in L
    if projection of CG onto Q lies within the polygon L **then**
      $v \leftarrow \{p, s\}$
      add v to V
      for all $v^* \in V \neq v$ **do**
        $\{p^*, s^*\} \leftarrow$ pose and state of v^*
        $k = \{p \cap p^*\}$
        $x \leftarrow$ set of endpoint coordinates for each act. in k w.r.t state s
        $x^* \leftarrow$ set of endpoint coordinates for each act. in k w.r.t. state s^*
        if *Condition 1* or *Condition 2* **then**
          $[R] = kabsch(x, x^*)$
          $r \leftarrow$ rotation angles derived from R
          $t \leftarrow$ centroid translation vector (x, x^*)
          $e \leftarrow \{v, v^*, r, t\}$
          add e to E
        end if
      end for
    end if
  end for
end procedure

2.4 Example Soft Robot Design

The gait synthesis method described above is actuator agnostic as long as the pressurized or actuated geometries are known and the single-point-of-contact EFF assumption holds. To demonstrate this method, we present a silicone actuator and modular chassis system design.

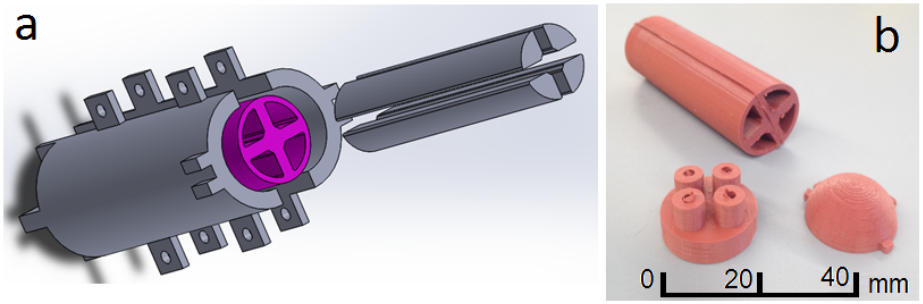

Figure 2.2: Actuator design. (a) Exploded view of actuator mold design (actuator body highlighted in purple). (b) Image of completed actuator body and end caps before final assembly.

2.4.1 Actuator and Robot Design

The actuators used in this work are created from two-part cast silicone (Elastosil® M 4601 silicone rubber). Each actuator has four radially symmetric pressure chambers as shown in Figure 2.2 and the material along the central axis of the actuator acts as a strain limiting component. Each actuator is approximately 95mm in length and weighs approximately 34 grams. Pressurizing combinations of actuator chambers cause the actuator curvature as seen in Figure 2.3(a-b), which is assumed to be constant radius. The actuator in Figure 2.3(b) has only one chamber inflated. The coordinate system used in this work is shown in Figure 2.3(c) wherein the curvature is parameterized by the constant of curvature k, and a rotation angle ϕ. The valving and pressure generation hardware is mounted off-board.

Example robots are shown in Figure 2.3(d-e), each comprised of six actuators. A symmetric design is shown in (d) and an asymmetric design shown in (e). Each robot chassis is composed of modular pieces, 3D printed in ABS plastic.

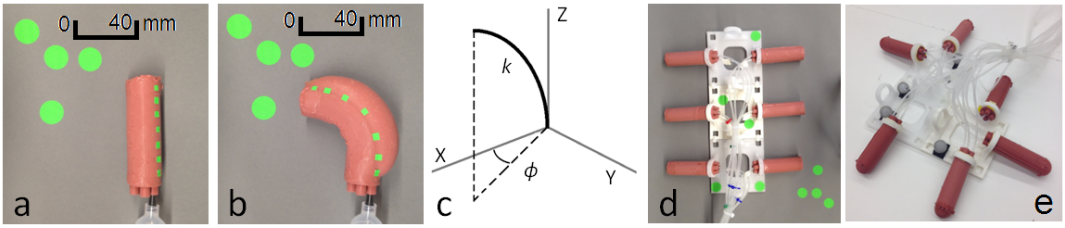

Figure 2.3: Actuator geometry and assembled example robots. (a-b) Depiction of actuator curvature. The constant radius of curvature assumption is evaluated via optical markers shown in green. (c) Diagram of coordinate system and parameters used for actuator bending geometry. (d-e) Fully assembled example robots - symmetric design shown in (d), and asymmetric design shown in (e).

2.4.2 Actuator Characterization

Elastomeric materials exhibit strain softening behavior in that the stiffness of the material decreases with the time spent at a specific strain - the Mullins Effect [48]. As the gait synthesis method presented in this section depends on a pressure/curvature model, we conducted a set of experiments to characterize this behavior in which we recorded the curvature of a series of actuators during repeated pressurizations. In each case, one chamber of an actuator was repeatedly pressurized to to 93 kPa for 10 seconds, and then depressurized (exposed to atmospheric pressure) for 10 seconds. The resulting maximum steady-state curvature was recorded with a Vicon motion capture system. The resulting curvature data is shown in Figure 2.4. Part (a) of Figure 2.4 shows the curvature of four single actuator chamber inflations without pre-pressurization. The curvature increases most rapidly in the early cycles but never appears to reach steady state - the constant of curvature continues to increase. Part (b) of Figure 2.4 shows the same procedure repeated for actuators that have undergone pre-pressurization. In this case, each actuator pressure chamber was pre-pressurized to 97 kPa approximately 20 times (over pressurized) and the resulting 93 kPa pressurization data shows no increase in curvature. The pre-pressurization process mitigates

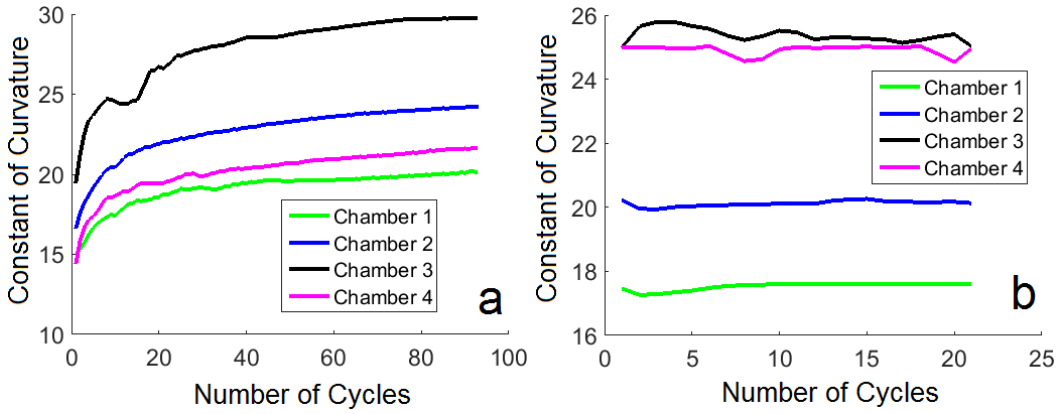

Figure 2.4: Example actuator pressure-curvature data. (a) Stress softening of unprepared actuator. (b) Constant performance of "pre-pressurized" actuator.

the stress softening behavior. We subsequently used pre-pressurized actuators for all gait demonstrations.

2.5 Gait Synthesis Demonstrations

As previously stated, the number of possible valve configurations, and subsequently the number of possible robot poses, grows exponentially with the number of valves. As such, we do not generate the entire gait graph before attempting to synthesize a gait. We present two methods of graph generation: one in which valve states are added to a gait graph randomly, and one in which valve states are added by searching around transitions of interest.

2.5.1 Random Graph Generation

In this method, the gait graph is constructed randomly. A unique valve state, s, is generated randomly and added to the graph wherein all possible stable poses

and possible pose transitions are evaluated and added to G. The gait search process is then attempted after 100 vertex additions. If a cycle is found, the pose and valve sequences are returned as a feasible gait.

It was observed, however, that the transition and no-slip assumptions were regularly violated when gaits synthesized in this manner were demonstrated on the robots shown in 2.3(d-e). The robot routinely failed to demonstrate the predicted pose transitions, and would regularly "fall" into a different pose (other than predicted), which invalidated the synthesized gait. Actuator compliance and friction assumption violations resulted in errors in predicted actuator contact.

2.5.2 Searching Around A Single Graph Edge

The second method of gait generation is intended to mitigate many of the confounding factors observed when demonstrating randomly synthesized gaits. In order to avoid "falling" or "sagging" into unpredicted poses, this method instead only allows transitions (other than the initially selected e^*), to be added to the graph if they demonstrate zero (or very small) chassis displacement. During the subsequent graph search synthesis process, each edge in the graph is weighted equally with a value of 1 so that the shortest path algorithm returns a cycle with the minimum number of transitions. The result is a gait that is comprised of a main, driving transition, and a sequence of zero motion transitions that allow the robot to "reset" the main transition. The results of this method are shown in Figures 2.5 and 2.6.

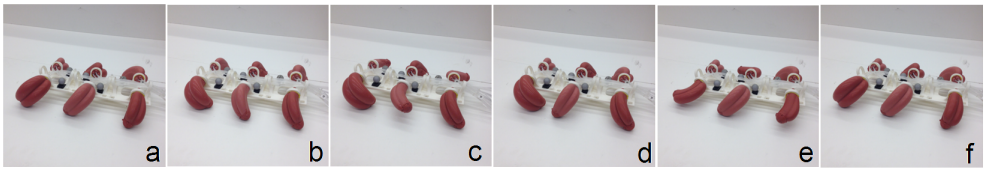

Figure 2.5: Five pose gait synthesized for the symmetric example robot, moving right to left. The main transition (the edge of interest) is shown in (a-b). The sequence of four "reset" transitions is shown in (d-f), with the initial pose attained again in (f).

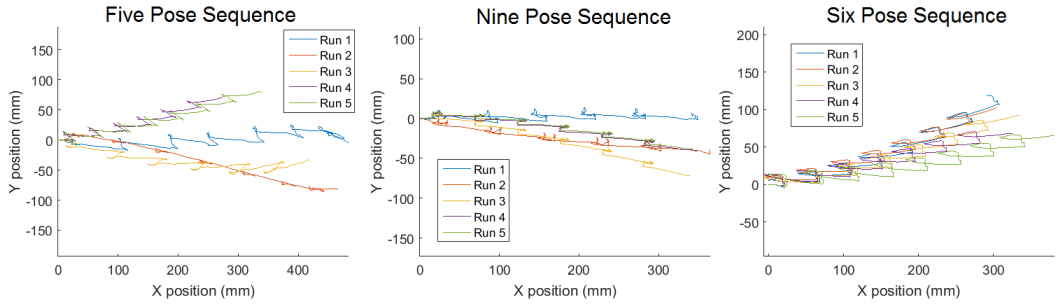

Figure 2.6: Recorded chassis displacements for three synthesized gaits. (a) Five pose sequence for the symmetric robot. (b) Nine pose sequence for the symmetric robot. (c) Six pose sequence for the asymmetric robot.

2.5.3 Demonstrations of Synthesized Gaits

We demonstrate three different synthesized gaits, two for the symmetric robot (Figure 2.3(d)) and one for the asymmetric robot (Figure 2.3(e)). Five runs were executed for each gait and the resulting chassis motions were recorded using a Vicon motion capture system. Each run consisted of multiple cycles of the gait. Figure 2.6 shows the recorded Vicon chassis motion data and Figure 2.5 shows the pose sequence for one of the gaits synthesized for the symmetric robot. A video of the robot may be found at the following link: https://www.youtube.com/watch?v=GNiNd51Hp-A.

The estimated chassis displacement for all three gaits was approximately 6 cm in the positive x direction with no lateral or rotational displacement - ap-

proximately one fifth of a body length per cycle for the symmetric robot and one third of a body length per cycle for the asymmetric robot. The robot experienced several actuator failures during the gait demonstrations and as such, gait performance was inconsistent, as can be seen most notably in Figure 2.6(a).

The trajectories for each of the runs in the 9 pose sequence and 6 pose sequence were largely similar (despite the discrepancies due to actuator replacement) and the transition cycle displacements were consistent within each robot configuration. The transitions in the 5 pose sequence were less consistent. This is likely due to the shorter sequence - the "resetting" action occurred in fewer transitions, and as such, the robot had fewer legs consistently in contact with the ground surface during the reset poses. In comparison, the consistent trajectories of the asymmetric robot with a similar number of sequence poses was likely due to the geometry of the chassis in that the asymmetry of the robot and actuator placement improved the stability of the system, i.e. the robot was able to better "balance" during reset transitions.

2.6 Discussion

This chapter presents (1) a method for synthesizing gaits for modular soft robots, (2) experimental data for on soft actuator performance, and (3) experimental data for synthesized gaits for composed robots - both symmetric and asymmetric. Gait synthesis is highly sensitive to the assumptions made about compliance, friction, and points of contact. The method detailed in this chapter predicts motions based on "rigid" actuator behavior, that is, the unloaded geometry of each component, and no assumptions are made about the deforma-

tion of actuators in contact with the ground surface. In order for this assumption to hold completely, the contact patch polygon would have to remain consistent in shape during a transition between to poses and, only such transitions could then be admissible as node additions to the gait graph. This may be possible for certain morphologies (e.g. completely symmetric robots), but this is not feasible for others such as the asymmetric robot. As such, it was assumed that these assumptions would be violated to some extent, but the unreliability of the gaits returned by random graph node addition clearly indicated the need for a more robust method - the "reset" pose method, which was demonstrated to produce viable (i.e. repeatable) gaits.

The controllers presented in this chapter are all open loop - no sensing is provided. Knowledge of the pose of the robot would improve the ability to implement robust gaits. As will be discussed in later chapters, which also address the issue of reasoning about actuator deflections under load, even the addition of simple, Boolean contact sensors has a marked impact on the ability of the robot to determine its pose and subsequently reason about transitions.

CHAPTER 3

RESILIENT GAIT PLANNING

3.1 Introduction and Background

In this section we address the issue of resilience in soft robot behavior and present the work published in [26]. While resilience to mechanical damage has been demonstrated in some soft robot systems, [45][62], the compliant materials utilized in soft actuator fabrication are prone to failure either by rupturing or developing leaks. These failures may inhibit actuation, either partially or fully, thereby restricting mobility and preventing the robot from accomplishing a task or goal. The performance of these actuators is difficult to predict due to the difficulty in modeling highly compliant materials.

Methods have been developed to detect and quantify the deformation and curvature of soft actuators [22][78]. The ability to monitor soft actuator performance enables a robot to detect and subsequently react to actuator performance degradation and failure.

Consider the example shown in Figure 3.1 in which a multigait, legged robot is performing a patrolling task, continually moving between regions H and E. The legs of the robot may degrade during usage and, additionally, may be damaged by moving through regions with detected hazards. Additional, replacement components are available to the robot in region C. In this particular case, the robot detects a hazard in region F and subsequently takes a more circuitous route to region E, while leveraging the set of available gaits in order to ensure task completion despite actuator failure as depicted in region A.

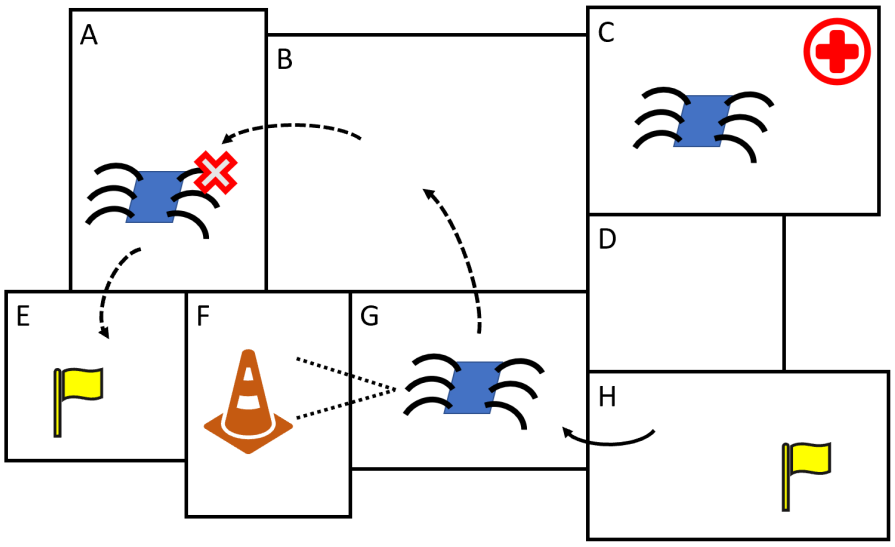

Figure 3.1: Example: A patrolling task for a multigait robot. The robot must continually visit regions E and H. An actuator failure is sensed in region A as the robot avoids a hazard detected in region F. The robot must utilize different gaits in order to continue the patrolling task as well as reach replacement components in region C.

Previous work has addressed gait resilience in walking robots [13][34] and soft robots [73], and some soft robotic systems have been demonstrated to continue to ambulate despite the total loss of one or more limbs [39]. However, these methods do not provide guarantees on behavior and we subsequently address this problem by utilizing methods of formal synthesis.

Recent research has addressed the issue of generating controllers for high-level robot behaviors [36]. These methods abstract robot behaviors and environmental interactions using Linear Temporal Logic (LTL) and provide techniques for synthesizing correct-by-construction controllers. These controllers guarantee the ability of the robot to satisfy a task despite adversarial environment behaviors. Similar techniques have been used to generate controllers that are resilient to system failures in different domains such as vehicle power management [75][53], but utilize different system abstractions.

High-level abstractions provide an expressive and flexible framework for encoding robot and environmental behaviors, and actuator performance and environmental hazards are easily encoded. In this chapter, we present three contributions: 1) a sensing based abstraction of actuator performance, 2) a framework for encoding reactive, multigait behavior in LTL, and 3) a demonstration of synthesized, correct-by-construction controllers for a soft robot.

3.2 Preliminaries

We provide an overview of Linear Temporal Logic (LTL) and controller synthesis. The reader is referred to [36] for further discussion.

3.2.1 High-Level Control

The problem, in this case, is posed as a two player game between the robot (the *system*) and an adversarial *environment*. The robot has a set of actions it may perform, the environment controls the states of the sensors of the robot, and the goal is to generate a strategy that guarantees that the robot will achieve a goal (satisfy a specified task) despite the actions of the environment.

Linear Temporal Logic

Linear Temporal Logic (LTL) is a formal language that contains the temporal operators $\bigcirc$ ("next") and $\mathcal{U}$ ("until") in addition to the Boolean operators $\neg$ ("not"), and $\wedge$ ("and"). We define a set of Boolean propositions, $\mathcal{X}$, that contains

the propositions controlled by the environment, and we subsequently define the set $\mathcal{Y}$ that contains the propositions controlled by the system. The total set of propositions is $AP = \mathcal{X} \cup \mathcal{Y}$. LTL formulas are defined over AP in the following way:

$$\varphi ::= \pi \in AP \mid true \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathcal{U} \varphi_2 \tag{3.1}$$

The operators $\Box$ ("always") and $\Diamond$ ("eventually") are derived from $\bigcirc$ and $\mathcal{U}$, and the operators $\vee$ ("or"), $\rightarrow$ ("implies"), and $\leftrightarrow$ ("bi-implication") are derived from $\wedge$ and $\neg$.

LTL formulas are evaluated over an infinite sequence of truth assignments, $\sigma = \sigma_0\sigma_1\sigma_2...$, where $\sigma_i \in 2^{AP}$. The expression σ_i denotes those propositions in AP that are true in position i. The temporal operators subsequently indicate the following regarding formulas:

1. $\Box\varphi$ is true at position i if it is true for all positions $j \geq i$
2. $\Diamond\varphi$ is true at i if φ is true in at least one position $j \geq i$
3. $\bigcirc\varphi$ is true at i if φ is true at position $i + 1$

The LTL specification we use in this work is denoted the following way [7]:

$$\varphi = (\varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e) \rightarrow (\varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s) \tag{3.2}$$

The initial conditions of the environment and the system are given in the Boolean formulas φ_i^e and φ_i^s, respectively. The specifications φ_t^e and φ_t^s are of the form $\wedge_{m \in M} \Box C_m$ where C_m are Boolean formulas that may contain the $\bigcirc$ operator

and represent the environment assumptions and system guarantees. The specifications φ_g^e and φ_g^s are of the form $\wedge_{o \in O} \square \diamond D_o$, where D_o are Boolean formulas that encode the environment liveness assumptions and system liveness guarantees (goals), respectively.

The synthesis process is presented in detail in [7]. The result of the synthesis process is a finite state automaton (FSA), $A = (\mathcal{X}, \mathcal{Y}, Q, Q_0, \delta, L)$ wherein:

- $\mathcal{X}$ is the set of environment propositions
- $\mathcal{Y}$ is the set of system propositions
- Q is the set of states
- $Q_0 \subseteq Q$ is the set of initial states
- $\delta : Q \times 2^{\mathcal{X}} \rightarrow Q$ is a transition function
- $L : Q \rightarrow 2^{\mathcal{Y}}$ is the labeling function of the states

The system transition relation, δ maps a current state and environment proposition values to the next state. The labelling function, L, maps a state to the propositions that are *true* in that state. The specification is *realizable* if there exists a solution that satisfies the specification regardless of the actions of the environment. We utilize synthesis tool SLUGS [20] in this work.

3.3 Robot Gaits, Actuator Health, and the Environment

3.3.1 Regions and Multigait Behavior

We assume the robot operates in a workspace comprised of a finite set of p discrete regions. Associated with this set of regions $Reg = \{R_1, ..., R_p\}$ is a set of Boolean propositions $reg = \{r_1, ..., r_p\} \subset \mathcal{Y}$ that denote the position of the robot such that the proposition r_i is true if and only if the robot is currently occupying region R_i. The possible region transitions are denoted by the adjacency relation $\tau \subset Reg \times Reg$.

We denote the set of m gaits available to the robot as $G = \{G_1, ..., G_m\}$. The robot my only utilize one gait at a time as they are mutually exclusive. We define an associated set of Boolean propositions $gaits = \{g_1, ..., g_m\} \subset \mathcal{Y}$ such that the proposition g_i is true if and only if the robot is currently utilizing gait G_i. Each gait is directional and we subsequently define the function $\mathcal{D}_\tau : \tau \to 2^G$ that maps each region transition in τ to the set of gaits able to make the transition. Included in G is the gait proposition *noGait* so that the robot is not forced to make a transition. We abuse notation slightly in this work by using G_i/g_i and R_i/r_i interchangeably.

3.3.2 Actuator Sensing and Health

We denote by $Legs$ the set of all actuators available to the robot. Each gait uses a subset of the available actuators and, as such, we define a function $\mathcal{L}_G : G \to 2^{Legs}$ that maps each gait in G to a subset of actuators in $Legs$ that utilize that

gait. We further define the function $G_{\mathcal{L}} : Legs \rightarrow 2^G$ that maps each actuator to the subset of gaits in G that utilizes that actuator.

In this work, we assume that actuators degrade during use. As such, we define a number of "health states" for each of the n actuators, denoted by $H = \{h_1, ..., h_n\}$. We then define a set of propositions for each actuator i that describes its health state, $P_i = \{\cup_{(j=1,...,h_i)} p_{i,j}\} \subseteq \mathcal{X}$. The variables in P_i represent a sequential evaluation of the health of the actuator, which may degrade any time a gait is utilized to make a region transition that uses that actuator. If degradation is sensed while the health proposition $p_{i,j}$ is true, then at the next time step the proposition $p_{i,j-1}$ is true and $p_{i,j}$ is false. Health propositions are mutually exclusive. The "health state" proposition $p_{i,h}$ of actuator i is *true* if the actuator is in perfect working condition. Conversely, the proposition $p_{i,1}$ is true if the actuator is in the lowest health state and is subsequently considered inoperable.

Additional components are available to the robot. Should the robot experience one or more actuator degradations or failures, the robot may access replacement components in regions of the workspace called *cache regions*, $Reg_c \subseteq Reg$. Should the robot enter a cache region, the health states of every actuator are reset to the "healthy" state. It is possible for the set Reg_c to be empty.

3.4 Soft Robot Design and Gaits

The example robot used in this work is shown in Figure 3.2. The robot is comprised of eight actuators that are mounted radially to a rigid polyurethane (Carbon® RPU 70) chassis. The actuator body and chassis components are printed on a Carbon M1 printer. Actuators are mounted by press-fit and are

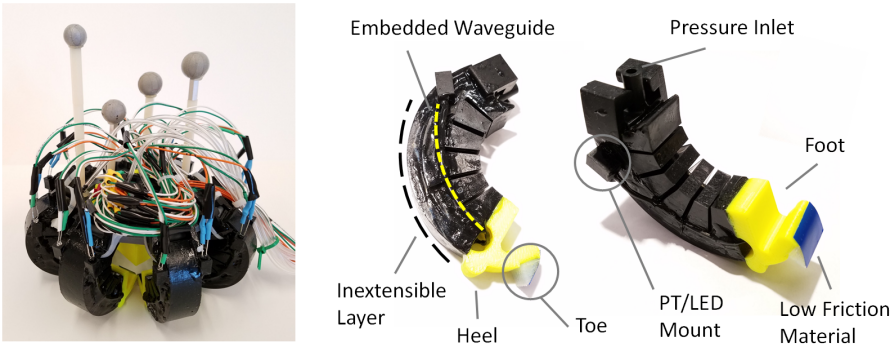

Figure 3.2: Example Robot and Actuator. Left: assembled physical robot. Eight actuators are arranged in a radial pattern around a rigid chassis. The silver spheres are markers for a Vicon motion capture system. Right: Actuator design.

subsequently quickly replaced. The actuators are pneumatic, and each is controlled by a solenoid valve that is off-board.

Each actuator has a body and a "foot," both of which are shown in Figure 3.2. The "foot" (yellow in the image) is 3D printed from ABS plastic and is comprised of a heel and a mounting location for a "toe" cast from Dragon Skin® silicone rubber. A band of low friction material is adhered to one side of the toe.

The backing layer of the actuator, shown in Figure 3.2 is significantly more thick than the walls of the bellowed section and subsequently creates an inextensible component. When the actuator is pressurized, it "unfurls" around this inextensible component and the actuator extends. During the extension phase, the high friction material of the toe contacts the ground surface and propels the robot. The resulting chassis motion is shown in Figure 3.3.

The low friction element on the other side of the toe allows the actuator to return to its resting orientation without resulting in an opposing displacement. The heel provides a low friction contact point while the actuator is at rest, allowing the chassis to slide.

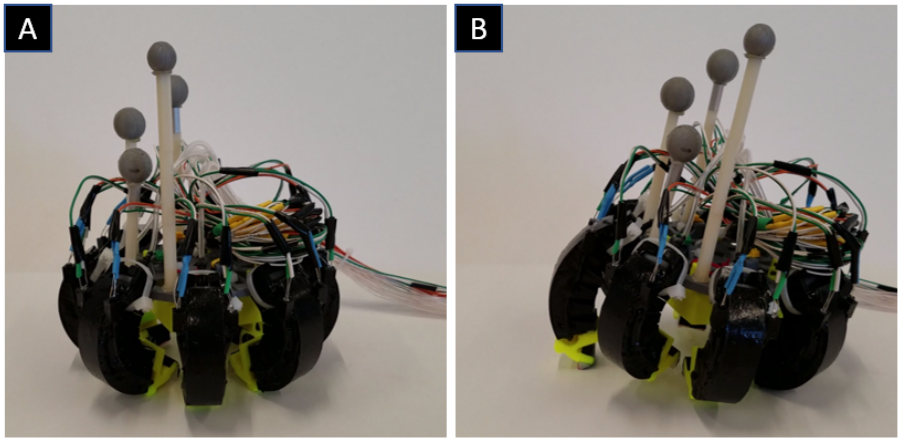

Figure 3.3: Example motion during actuator pressurisation. The actuator on the far left of the robot is pressurized, propelling the robot to the right.

3.4.1 Robot Gait Primitives

For this work we developed three different gait motion primitives, which are shown in Figure 3.4. Each gait utilizes two actuators pressurized in rapid succession, as shown in Figure 3.4 (A-C), (D-F), and (G-I). For example, the sequence (D-F) pressurizes actuator 4, propelling the robot in the positive x and positive y directions, before pressurizing actuator 7, pushing the robot in the negative x direction and back toward the center of the image. The net displacement is in the robot chassis y axis The radial symmetry of the robot results in twelve gaits, capable of propelling the robot in any of the $\pm x$ or $\pm y$ directions. Though the robot is capable of moving in diagonal directions utilizing these gaits, we restrict the number of gaits used to twelve as a result of the grid workspace. It should be noted that we include an adjustment gait to correct for errors in orientation. It is assumed that the correction gaits do not affect actuator health states.

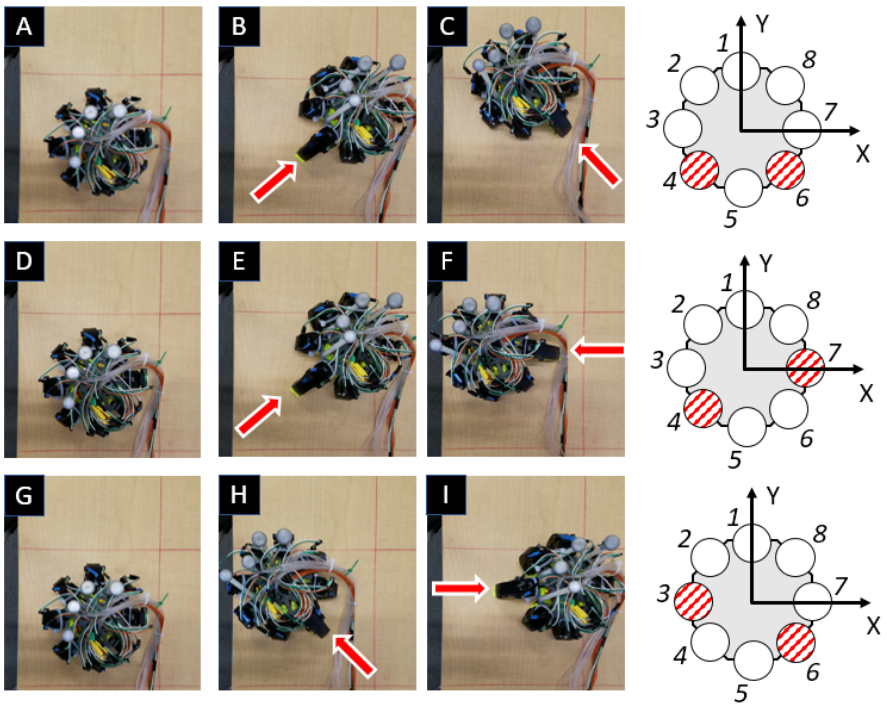

Figure 3.4: Example robot gaits. Three gait sequences are depicted in (A-C), (D-F), and (G-I). Each propels the robot in the positive y axis w.r.t. the robot frame. Shown on the right are the respective actuators used in each gait, shaded in red.

3.4.2 Detecting Curvature and Degradation

Each actuator has an incorporated optical waveguide as indicated in Figure 3.2. We evaluate actuator performance by leveraging the work in [78] in which optical waveguides embedded in a soft prosthetic hand are used to detect curvature and contact. In this case we sense only curvature. An infrared LED provides a light source, the intensity of which is detected by a phototransistor (PT). As the actuator deforms, the waveguide experiences strain which reduces the transmissivity of the waveguide material and the level of light detected by the PT decreases, providing a mechanic for quantifying actuator curvature.

The output of an example waveguide sensor is shown in Figure 3.5. Shown left in the figure is a plot of the phototransistor intensity during several actu-

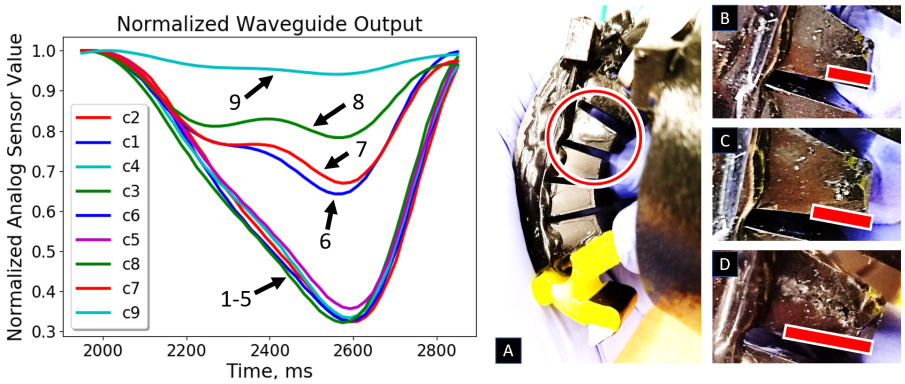

Figure 3.5: Waveguide behavior and actuator performance degradation. Left: normalized waveguide data for a progressively damaged actuator. Right: A-D highlight damage, induced by an X-ACTO blade.

ation cycles. The analog intensity values are filtered and normalized w.r.t. the at-rest value. Nine different extension (actuation) cycles are shown in the figure and correspond to various levels of actuator health. After each actuation trial, the actuator was deliberately and progressively damaged using an X-ACTO blade, as depicted on the right. Curves 1-5 depict "healthy" actuator behavior (complete actuation as a function of pressure input) and correspond roughly with the level of damage shown in Figure 3.5(B). Curves 6-8 correspond roughly with the level of damage shown in C. The cut, expanded and enlarged, caused the actuator performance to diminish, but the actuator was still capable of propelling the robot. Cycle 9 corresponds to the level of damage shown in D. In this case, the performance of the actuator was completely reduced and the actuator was ineffectual in moving the robot.

In this specific case, the performance levels of the actuator can be grouped into three sets: cycles 1-5, for which no significant degradation in performance was detected, cycles 6-8, for which partial degradation was detected, and cycle 9, in which state the actuator was deemed inoperable. This particular actuator could be considered to have 3 actuator health states, that is $h_i = 3$. It should be

noted that this data corresponds to a very specific type of damage to a specific actuator design. As such, thoroughly validating this damage evaluation method for different actuators in different operating conditions requires further testing.

3.5 Encoding Multi-Gait Behavior

3.5.1 Regions and Gait Selection

We encode gait selection behavior and region transitions by appending the following LTL formulas to φ_t^s. Those formulas that encode proposition mutual exclusivity are omitted for brevity.

Each gait propels the robot in a given direction (as depicted in Figure 3.4) and subsequently allows the robot to make certain region transitions. We encode this behavior in the following way:

$$\bigwedge_{r_s \in reg} \square \left(r_s \rightarrow \bigcirc \left(\bigvee_{\{r_e | (r_s, r_e) \in \tau\}} \quad \bigvee_{g \in \mathcal{D}_T(r_s, r_e)} r_e \wedge g \right) \right) \tag{3.3}$$

If the robot is in region r_s, at the next time step the robot is required to to transition to one of the regions as specified by the transition relation τ and must also select a gait compatible with that direction of motion. The robot may not move to a region if there are no gaits available to make that transition. Same region transitions, enabled by the inclusion of the gait proposition *noGait*, are included in τ.

As described previously, damage to an actuator may cause one or more ac-

tuators to become inoperable. In this case, the robot is restricted from selecting any gait associated with a failed actuator. We encode such gait restrictions in the following way:

$$\bigwedge_{L_i \in Legs} \square \left(\bigcirc p_{i,1} \rightarrow \left(\bigwedge_{g \in \mathcal{G}_L(L_i)} \neg \bigcirc g \right) \right) \tag{3.4}$$

We encode the degradation of actuator performance and actuator replacement by appending the following LTL formulas to φ_t^e as the environment controls the state of each actuator. The environment is only able to influence the health state of an actuator that is in use by the currently selected gait. This behavior is captured in the following way:

$$\bigwedge_{g \in gaits} \square \left(\left(g \wedge \neg \left(\bigvee_{r \in reg_c} r \right) \right) \rightarrow \left(\bigwedge_{L_i \in \mathcal{L}_G(g)} \varphi_{D_i} \wedge \bigwedge_{L_k \notin \mathcal{L}_G(g)} \varphi_{D_k} \right) \right) \tag{3.5}$$

where φ_{D_i}, and φ_{D_k} are defined as:

$$\varphi_{D_i} = \left(\bigwedge_{j=h_i, h_i-1, \ldots, 2} (p_{i,j} \rightarrow (\bigcirc p_{i,j} \vee \bigcirc p_{i,j-1})) \right) \wedge (p_{i,1} \rightarrow \bigcirc p_{i,1})$$

$$\varphi_{D_k} = \bigwedge_{j=h_k, h_k-1, \ldots, 1} (p_{k,j} \leftrightarrow \bigcirc p_{k,j})$$

Assuming the robot is not in a cache region, the environment may choose to alter health states of in-use actuators according to the previously described rule of consecutive degradation.

If a robot moves into a cache region, all damaged actuators are considered replaced, and the health states of all are actuators are reset to "healthy" in the next time step. This is captured in the following way:

$$\square \left(\left(\bigvee_{r \in reg_c} r \right) \rightarrow \left(\bigwedge_{i=1,\ldots,n} \bigcirc p_{i,h_i} \right) \right) \tag{3.6}$$

Appending these statements to the mission specification captures the gait selection and actuator degradation behavior of a multigait robot. If the specification is realizable, the resulting synthesized controller ensures the resilience of the robot to actuator failure.

3.5.2 Example Scenario

An example scenario is depicted in Figure 3.6. The robot operates in a grid world and is able to move "north," "south," "east," and "west" at each time step. The goal of the robot is a patrolling task - the robot must continually visit the regions of the map marked by yellow diamonds, 11 and 12. These goals are encoded in the following LTL expression, appended to φ_g^s

$$(\square\diamond r_{11}) \wedge (\square\diamond r_{12}) \tag{3.7}$$

Caches of replacement components, marked with green circles, are available in regions 3 and 14.

We demonstrate in this example how one can encode more complex environment behaviors by including hazards, detectable by the robot, that may be "activated" or "deactivated" by the environment. If the robot moves through a region with an active hazard, all actuators associated with the currently selected gait are considered failed and are rendered inoperable regardless of the health state. These regions are denoted by the set $T = \{T_1, ..., T_v\}$. We define a mapping function $\mathcal{F}_T : T \rightarrow 2^{Reg}$ that maps each hazard to a set of workspace regions. Sets of hazards have associated Boolean propositions, $t = \{t_0, ..., t_v\} \subseteq \mathcal{X}$ such that the

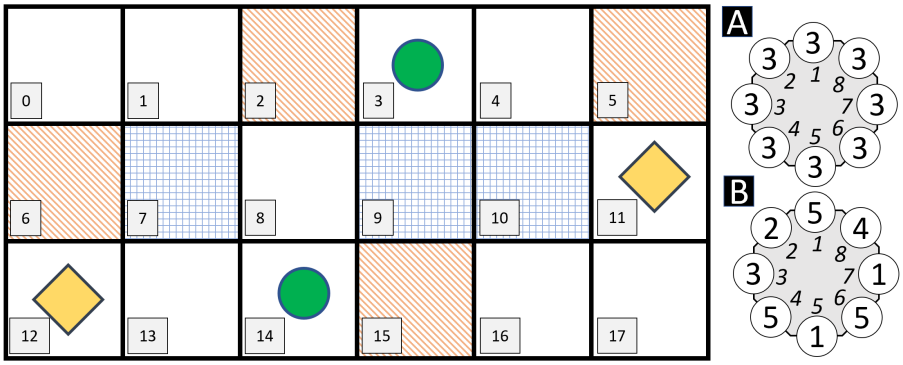

Figure 3.6: Left: Example grid environment. Goal regions are marked with yellow diamonds. Cache regions are marked with green circles. One set of hazard regions is indicated by blue checkers, and a second set is marked with red stripes. Right: actuator reliability for robots **A** and **B**. Actuator health levels are indicted inside the circles and the italicized numbers adjacent indicate actuator indices.

robot senses active hazards for all regions in $\mathcal{F}(T_i)$ if proposition t_i is true. We assume propositions in t are mutually exclusive and we further assume the set of cache regions and set of hazard regions are disjoint.

We add to the antecedent in Equation 3.5 ($\wedge\neg\varphi_H$):

$$\varphi_H = \left(\bigvee_{t_k \in t} t_k \wedge \left(\bigvee_{r \in \mathcal{F}_T(t_k)} r\right)\right) \tag{3.8}$$

and the statement:

$$\bigwedge_{g \in gaits} \square\left(g \wedge \neg\left(\bigvee_{r \in reg_c} r\right) \wedge \varphi_H\right) \rightarrow \left(\left(\bigwedge_{L_i \in \mathcal{L}_G(g)} \bigcirc p_{i,1}\right) \wedge \bigwedge_{L_k \notin \mathcal{L}_G(g)} \varphi_{D_k}\right) \tag{3.9}$$

is added to φ_t^e, where φ_{D_k} is defined as before.

As shown in Figure 3.6, There are two sets of hazard regions. One set containing, regions 7,9, and 10, is marked with blue checkers, and the other, containing regions 2, 5, 6, and 15, is marked with red stripes.

3.6 Demonstrations

We present demonstrations of the behaviors of two robots with different levels of actuator health states for the example task described in the previous section. Robot **A** is equipped with actuators of equal levels of reliability, $\mathbf{A} : H_A = \{3, 3, 3, 3, 3, 3, 3, 3\}$, while robot **B** is equipped with actuators of asymmetric reliability, $\mathbf{B} : H_B = \{5, 2, 3, 5, 1, 5, 1, 4\}$. Robot **B** has two actuators that only have one health state, which means that they are considered inoperable, are not affected by cache regions, and may not be used for any gait.

The synthesis algorithm subsequently produces markedly different behaviors for each robot. Robot **A** has a greater level of gait redundancy than **B** due to the even levels of actuator health. Robot **B**, in contrast, lacks the same level of redundancy due to the two permanently failed actuators, but does have several actuators (1,4,6) with higher levels of health states, which permits robot **B** a greater reachable space than **A**, meaning that robot **B**, unimpeded by hazards, may travel further than robot **A** before requiring component replacement.

The specifications for both robots are realizable, and example runs of each are depicted graphically in Figure 3.7. The corresponding physical behavior is depicted in Figure 3.8. In both scenarios, the robot begins in region 12. Regions containing active hazards are highlighted in both figures (active hazard regions not in proximity of the robot are not highlighted for clarity).

In the graphical description, currently selected actuators (of the currently selected gait) are boxed in green. Healthy actuators are shown in blue, degraded actuators are depicted in yellow, and failed actuators are marked with red stripes. Actuators that fail (are reduced to the lowest health state) during a

transition are indicted with a red X.

In the images of the physical demonstrations, goal regions are marked with with pink tags, cache regions with teal tags. Actuator degradation during the physical demonstrations was simulated by manually setting the proposition values. Actuator failures were similarly simulated by physically crimping the pneumatic channels, preventing complete actuation. No actuators were physically harmed during these demonstrations. The robot successfully sensed all "failures" and reacted according to the synthesized controllers. Localization was provided by a Vicon motion capture system.

Robot A: Robot **A** initially starts in region 12 and begins by moving "north" to region 0 and then "west" to region 3 (a cache region). This is shown in Figures 3.7 and 3.8 in *a1*, *b1*, and *c1*. In this scenario, the environment activates all hazard regions and causes actuator degradations with each transition. Ultimately, actuators 3, 6, 2, and 5 are caused to fail, yet the robot takes the same route it would have if no hazards had been activated and no degradations sensed. The robot, in this case, leverages gait redundancy to mitigate the effect of hazards and degradations.

The robot then moves through regions 4 and 10 to the goal in 11 before turning back toward the other goal in region 12. At this point, there are two possible routes the synthesized controller considers. As the robot moves from region 11 to region 10 (with an inactive hazard), actuator 5, previously in a degraded state, is caused to fail (*d1*). The robot senses that the hazard in region 9 is inactive and proceeds to region 11 via regions 8 and 14 as shown in white, dashed arrows in 3.8. Had the actuator not degraded and failed, however, the robot would have proceeded to 11 along the bottom of the map via region 16, again leveraging gait

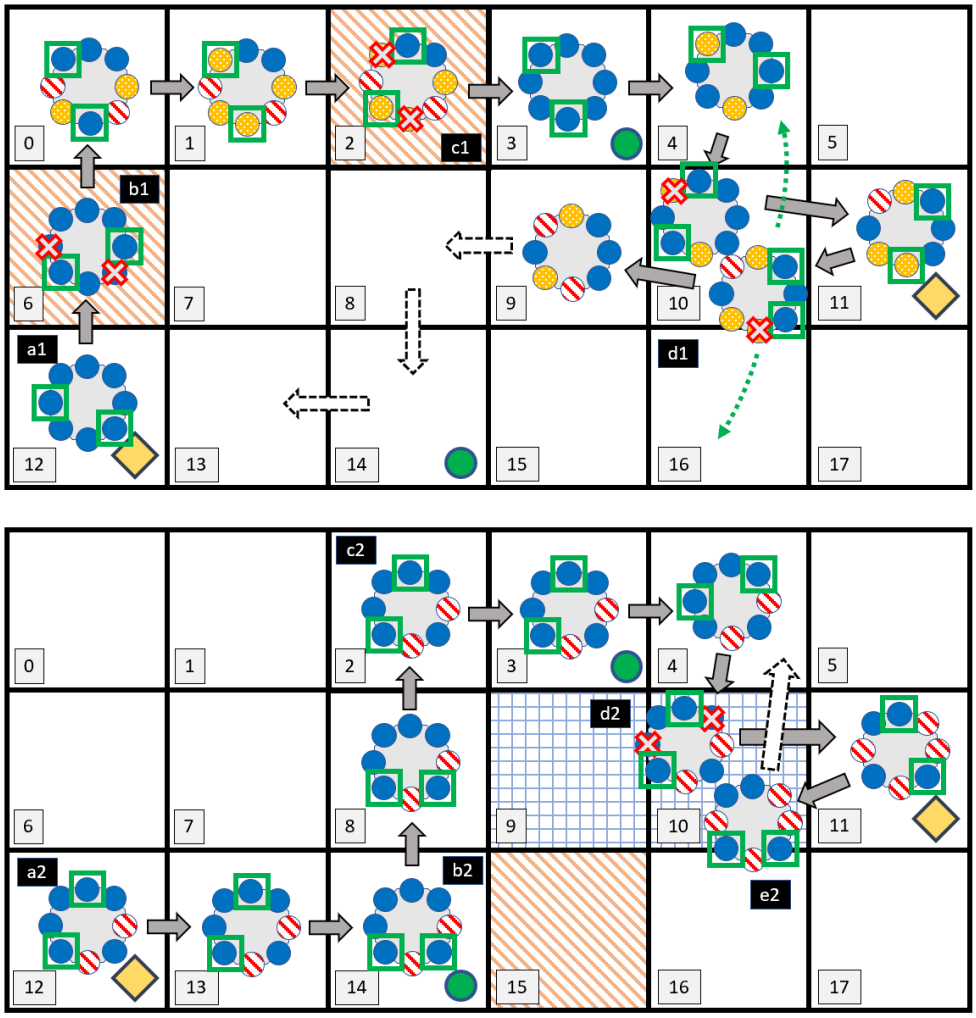

Figure 3.7: Example scenario robot behavior, robot **A** shown top, robot **B** shown bottom. The blue and red patterns depict active hazard regions. Healthy actuators are marked in blue, degraded actuators marked in yellow, and failed actuators marked with red stripes. Currently in-use actuators marked with green boxes. Actuators that fail during a transition are marked with a red X.

redundancy and ignoring the hazard in region 15. Had the hazard in region 10 been active, rather, the robot would have been forced to take a more circuitous route back to region 11 via the cache region 3. Both alternate paths are shown as dashed green arrows in Figure 3.8.

Robot B: In the absence of activated hazards, robot **B** would be able to take a direct route to the goal in region 11. However, the lack of gait redundancy due

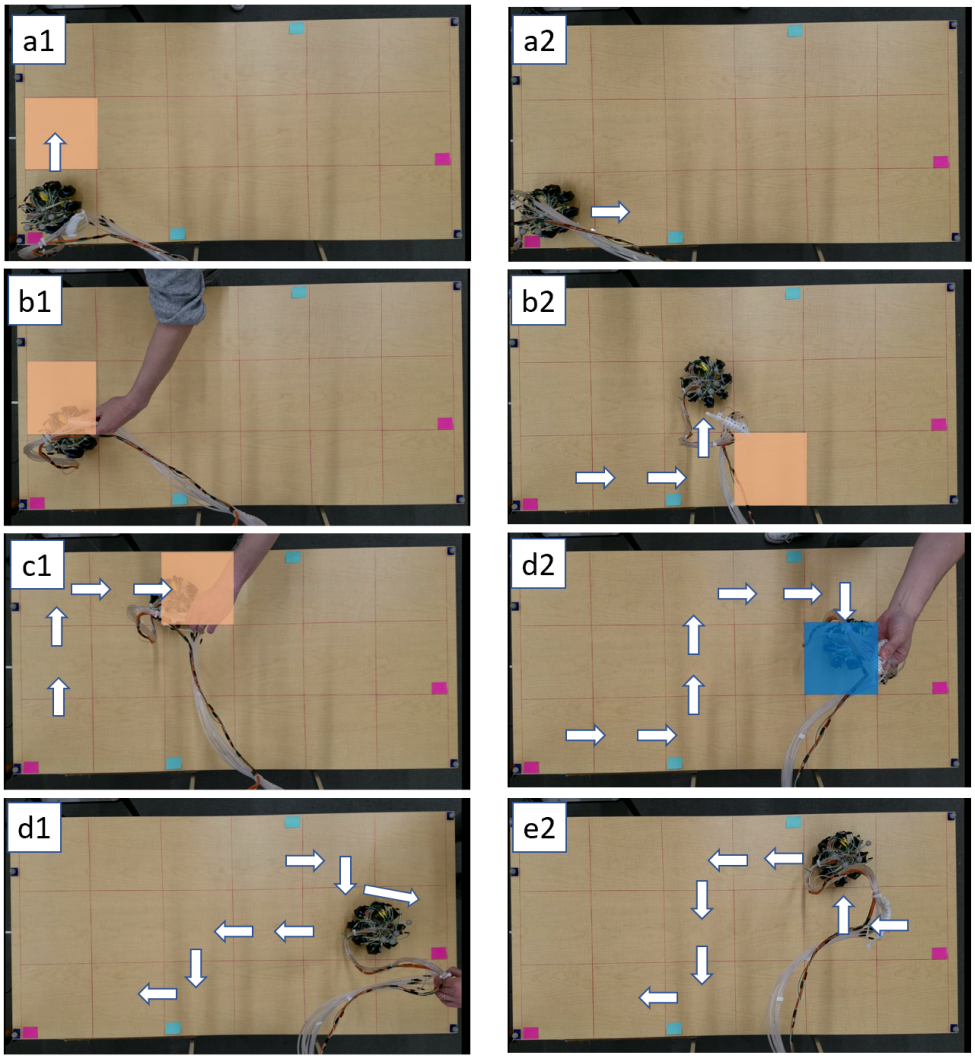

Figure 3.8: Physical demonstration. Robot **A** shown left, robot **B** shown right. Image labels and highlighted active hazards correspond to instance in Figure 3.7. Images *b1*, *c1*, *d1*, and *d2*, show the physical crimping of pneumatic lines to simulate actuator failure.

to the two permanently failed actuators means the robot must avoid all possible hazards. The robot senses an active hazard in region 15, and subsequently takes a conservative route through both cache regions (*a2*, *b2* and *c2*). In moving from region 3 toward region 11, however, the robot moves through region 10 despite the active hazard (*d2*), causing actuators 3 and 8 to fail. As a result, the robot must return to the cache region 3 before again moving toward the goal in region

12 as denoted by a large dashed arrow (*e2*), once again conforming to a more conservative route.

3.7 Discussion

In this chapter we address the problem of generating controllers for multigait, walking soft robots that are resilient to actuator failure. We presented (1) an abstraction of multigait behavior and actuator sensing, (2) provided a framework for encoding gait selection and actuator failure in LTL, and (3) demonstrated synthesized automata on a physical soft robot system operating in an adversarial environment.

These demonstrations show only a part of the complex behaviors of the synthesized automata presented above, but they highlight the effectiveness of the synthesis algorithm applied to the framework presented in this work. The multigait abstraction and straightforward, easily altered behavioral encoding in LTL produced nuanced controllers for both robots and the flexibility of the behavioral encoding allows specifications to be altered rapidly for different robots and different environments. In this particular case, the only difference was the levels of health states specified in H for each robot, but this concept could be easily extended to other actuator types, robot morphologies, or environmental behaviors. For example, if actuators on the robot were able to grasp an object as well as contribute to a gait, holding or carrying objects may restrict the robot from using associated gaits. Further, encoding robot behaviors in more complex environments rather than the simple, two dimensional, obstacle free environment presented in this chapter (e.g. different gaits for different terrains) can

be done with the same framework. Encoding these behaviors involves simple additions to the specification.

CHAPTER 4

MODELING OF AMBULATING SOFT ROBOTS

As discussed in the introduction, models of soft robots and actuators range from simplistic [25] to complex [19][28] depending on the degree to which the infinite degrees of freedom of soft materials are represented in the model. Further, some models incorporate contact mechanics [18][30], but these representations are computationally intensive or are specific to soft actuator compositions or morphologies. Our interest in this chapter is generalizable models for ambulating soft robots that do not require significant computation or are highly actuator or material specific, yet are able to capture the key modes of deformation of soft actuators as well as contact forces.

To that end, we present two different techniques for modeling the dynamics of soft actuators that differ in their level of abstraction but are nonetheless simplistic in implementation - one in which soft actuators or appendages are modeled as generalized spring/damper systems such that the *explicit* deformations of each actuator under load are not considered, and one in which multiple modes of deformation are considered but only a subset are characterized dynamically, and complex models are not subsequently required. These models are presented as methods to generate dynamic models of soft robotic systems without relying on highly specific and computationally intensive tools. We subsequently present two different walking soft robot systems in order to demonstrate the two modeling methods and present a comparison of simulated and recorded chassis motion data.

4.1 Modeling Techniques

4.1.1 Generalized Spring-Damper Model

This section presents a generalized method for modeling soft actuator chains as spring-damper systems. Many soft actuators are designed for a specific deformation or deformations. For example, the actuators presented in [25] are designed to exhibit constant curvature around a central axis. However, state variables capturing the intended deformations of a given soft actuator did not accurately quantify the shape of the actuator under certain loading conditions - e.g. ground contact forces, which were not considered when modeling the kinematics of the actuators. It may be the case that actuator deformation may be satisfactorily quantified under some conditions but not all, as was noted when the kinematic assumptions for the actuators in [25] were observed to be violated during gait execution (as discussed in Chapter 2). To that end, the model presented in this section is intended to dynamically characterize actuators that have a known unloaded geometry but an unknown loaded geometry.

Similar to the model we presented in Chapter 2, we make the assumption that each soft actuator appendage makes contact with the ground surface at exactly one location, hereby referred to as the end effector (EFF). Subsequently, the appendage is modeled as an actuator chain with a proximal attachment point at the chassis and a distal point at the EFF.

Generalized Actuator and Robot Representation

The following is a formalization of the functions and definitions required for the generalized spring-damper model. We define a limb or appendage, a, as the tuple $(q_a, u_a, f_q, f_e, f_m, K_q, b_q)$ wherein:

- q_a is a column vector of n_a limb state variables
- u_a is a column vector of m_a actuator input variables.
- $f_q : u_a \rightarrow \mathbb{R}^{n_a}$ is a vector function describing the dynamics of the state variables w.r.t. the inputs
- $f_e : q_a \rightarrow \mathbb{R}^3$ maps the limb state to the location of the EFF, p_e
- $f_m : q_a \rightarrow \mathbb{R}^3$ maps the limb state to the location of the limb center of mass
- K_q is a $[3 \times 3]$ stiffness matrix as a function of q_a
- b_q is a scalar damping coefficient

The locations of the center of mass (CoM) and the EFF in the above definitions are with respect to a local limb reference frame, $\mathcal{F}_a$. Note that the assumptions of a singular point of contact with the ground surface and a single point of attachment mean that the actuator chain may be any combination of serial and parallel actuators. We subsequently define a robot, R, as the tuple (A, P, R), wherein:

- A is a set of s limbs
- P is a list of vectors in $\mathbb{R}^3$, $[p_0, ..., p_s]$, such that each p_i locates the local origin frame of limb a_i, $\mathcal{F}_{a,i}$, w.r.t. a local chassis origin frame, $\mathcal{F}_c$

- R is a list of rotation matrices in $\mathbb{R}^{[3x3]}$, $[R_0, ..., R_s]$, such that each R_i describes the orientation of $\mathcal{F}_{a,i}$ w.r.t. $\mathcal{F}_c$

We assume that values of the vectors and matrices specified in the sets P and R are fixed for a given robot. We subsequently define the robot state, q, as the vector of all limb state vectors, $q = [q_{a,0}, ..., q_{a,s}]^\top$, and we similarly define the robot control input, $u = [u_{a,0}, ..., u_{a,s}]^\top$. We further specify n as the total sum of all $n_{a,i}$ and m as the sum of all $m_{a,i}$.

Limb Forces and Chassis Motion

The model assumes that each limb of a composed robot, R, acts as a spring-damper system such that the force on the robot chassis from limb i is:

$$F_{a,i} = K_{a,i}(p_{e,i} - \tilde{p}_{e,i}) + b_{a,i}(\dot{p}_{e,i} - \dot{\tilde{p}}_{e,i}) \tag{4.1}$$

wherein $p_{e,i}$ and $\dot{p}_{e,i}$ are the actual (loaded) EFF position and velocity, respectively, and $\dot{p}_{e,i}$ and $\dot{\tilde{p}}_{e,i}$ are the unloaded EFF position and velocity, respectively, as defined in $\mathcal{F}_a$. In this work, we assume a centroidal dynamics [51] model in which the robot and limbs are modeled as a single mass at each time step. Subsequently, the spring/damper limb abstraction and connection definitions provide a dynamic model for a composed robot. This model is depicted in Figure 4.1. Note that because internal forces aren't directly computed (as would be the case in a traditional, rigid robot model), the relationship between actuator input and limb force application is indirect and described entirely by the unloaded EFF position, spring matrix values, and damping coefficient value.

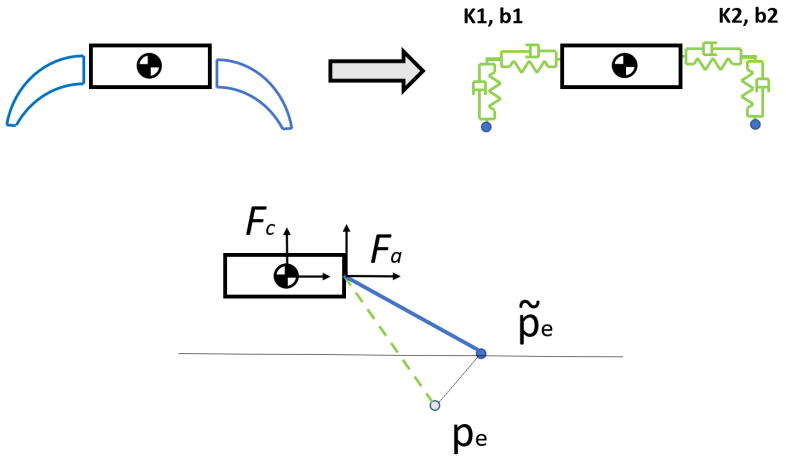

Figure 4.1: Generalized spring/damper model. Shown above is the abstraction of each soft limb as a multi-axis spring/damper. Shown below is the deflection vector represented by the difference between unloaded and loaded EFF position ($\tilde{p}_e - p_e$).

We present a robot to which we apply this model in Section 4.2.1 and demonstrate how the dynamics functions, stiffness values, and damping coefficients are determined for this particular system.

4.1.2 Active/Passive Model

In this section we present a method for modeling walking soft robots that captures specific deformations. While the generalized spring-damper model does not make any assumptions about the specific modes of deformation under load, this model assumes that some state variables are known under load. We differentiate this model from others that capture specific deformations by splitting actuator state variables into two groups: *active* and *passive*. Those state variables in the *active* set are those that capture deformations that are determined largely by actuator inputs, e.g. curvature along a principle axis. In other words,

the dynamics of the *active* state variables are dominated by the actuator inputs, and the effects of both ground contact forces/moments, and internal forces are, by comparison, negligible. Conversely, the dynamics of the *passive* state variables are those that are not principally determined by the actuator inputs and are instated dominated by the ground contact and internal forces.

Active/Passive Actuator and Robot Representation

The following is a formalization of the functions and definitions required for the active/passive model. We define an actuator, a, as the tuple $(q_a, u_a, f_q, f_c, f_m)$ wherein:

- q_a is a column vector of n limb state variables
- u_a is a column vector of m actuator input variables.
- $f_q : u_a \rightarrow \mathbb{R}^n$ is a vector function describing the dynamics of the state variables w.r.t. the inputs
- $f_c : q_a \rightarrow \mathbb{R}^3$ is a vector function that maps the state of interest to contact point locations on the actuator body
- $f_m : q_a \rightarrow \mathbb{R}^3$ is a vector function that maps the state of the actuator to the center of mass of the actuator

Note that the output of the dynamics function, f_q, is assumed to be 0 for any state variable considered to be *passive*. Similar to the definitions for the generalized model, all locations specified in the definition above are with respect to a a local actuator coordinate frame, $\mathcal{F}_a$. Unlike the generalized formulation specified above, we make no assumptions about where the actuator makes contact

with the ground surface other than there exists the function f_c that locates any given contact point in an actuator frame (i.e. the overall shape of the actuator is known, represented, for example, by a wireframe).

We define a robot, R by the tuple (A, C, M) wherein:

- A is a set of actuators
- C is a set of non-actuating components (e.g. touch sensors)
- M is a set of mounting connections

Note, in the following definitions, the word "component" is overloaded to refer to include both actuators and sensors. We subsequently define a connection, c, between two components (the *from* component and the *to* component), as a tuple: $(ID_f, ID_t, p_f, p_t, f_f, f_t, d, \mathbb{R})$ wherein:

- ID_f is the identifier or name of the *from* component
- ID_t is the identifier or name of the *to* component
- p_f is a point defined in the local origin reference frame of the *from* component, $F_{a,f}$
- p_t is a point defined in the local origin reference frame of the *to* component, $F_{a,t}$
- f_f is a function that locates the point p_f in $F_{a,f}$
- f_t is a function that locates the point p_t in $F_{a,t}$
- d is a vector in $\mathbb{R}^3$ that specifies p_t w.r.t. p_f in frame $F_{a,f}$
- R is a $[3x3]$ rotation matrix that describes the orientation of a reference frame at p_t w.r.t. p_f

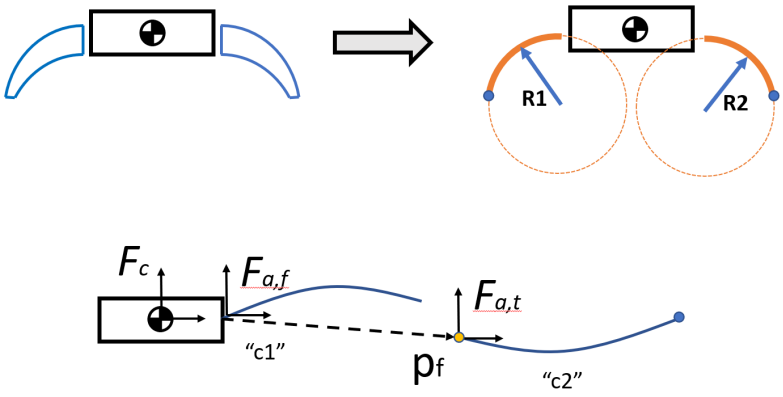

Figure 4.2: Active/passive model. Shown above is the abstraction of soft limbs as a constant radius, continuum geometry. Shown below is a representation of a connection for a serial chain of soft actuator elements.

This model is depicted graphically in Figure 4.2.

Modeling Passive Elements

The dynamics of the *active* state variables are assumed to be satisfactorily described by f_q despite potential loading conditions. The dynamics of the *passive* state variables, however, are defined to be zero even though the values may evolve over time. The evolution of passive state variables is therefore determined through optimization processes during simulations with this model, and through online state estimation on the physical robot, as will be discussed in later sections.

4.1.3 Friction Models

Both of the models presented above are able to incorporate a friction model. The generalized spring damper model makes the assumption that the actuator only makes contact with the ground surface at a single point, the EFF, defined by p_e, and the ground forces at each contact point are determined through Equation 4.1. The active/passive model, in contrast, makes no assumptions about where the actuators contact the ground surface, but one of the requirements of the model is the function f_c, which provides a tool to define the surface geometry of each actuator as a function of the actuator state. In later sections we utilize wireframe representations for soft actuators via this function f_c that provide the ability to assess contact. As will be discussed, the loading condition and subsequent normal/reaction forces on each actuator may then be determined through optimization. Once the normal forces or loading condition is known, frictional forces may then be computed.

4.2 Example Soft Actuators and Robots

In this section we present two different soft robotic systems in order to demonstrate both modeling techniques. The first is a robot fabricated from elastomeric polyurethane, to which we apply the generalized spring-damper model, and the second is fabricated from cast silicone, to which we apply the active/passive model. We then present recorded motion data and an analysis of the performance of each model.

4.2.1 Elastomeric Polyurethane Actuators and Actuator Chains

The robot presented in this section is fabricated on a Carbon® 3D Printer from two different materials: soft actuators printed in Carbon® Elastomeric Polyurethane (EPU) and chassis components printed in Carbon® Rigid Polyurethane (RPU). Both types of components are printed on a Carbon® M1 printer, which utilizes a stereolithographic process to cure parts from resin material. The chassis elements and actuator mounting components are modular and, as a result, several different robot geometries may be composed from the same actuator components.

Each soft appendage or actuator chain in this example system is fabricated from identical prints of the soft actuator segment depicted in Figure 4.3, which is similar in shape an identical in material to the actuators of the robot presented in Chapter 3. The actuator is printed in two main pieces and assembled and cured post-print. The individual components of each segment are shown in Figure 4.3 (a), a completed and assembled segment is shown in Figure 4.3 (b), and an exploded model view of the component pieces is shown in Figure 4.3 (c) and (d). The curved back piece of the actuator as seen in Figure 4.3 is thicker than the material of the bellowed piece, and in a manner that is, again, similar to the deformation of the actuator presented in Chapter 3, acts as a strain limiting component. As such, the actuator "unfurls" as pressure is applied and the curvature of the actuator decreases, in effect, straightening out.

In the unloaded state, this "unfurling" is the dominant mode of deformation and is modeled as a single variable, the constant of curvature, k, which is equivalent to $1/r$ where r is the radius of curvature. This is the mode of deformation central to the propulsive force generated on the robot presented in Chapter 3

as can be seen in Figure 3.3. Of note are the spline attachment piece shown in Figure 4.3 (a), which allows the actuator to be attached to chassis components, and a foot cap piece, shown in the lower portion of Figure 4.3 (b), which allows various foot attachments to be included, as seen in Figure 4.4 (a). This image depicts a series of actuators composed from identical, single chamber actuator segments. Each end cap has attached a foot extension piece, which terminates in the EFF of each appendage. Of the appendages in 4.4(a), the two flanking appendages (marked 'A' and 'B') are composed of three segments each, and the two middle appendages (marked 'C' and 'D') are composed of two segments each. Each foot extension in this image has a silicone foot pad for providing traction. A robot composed from these four actuator chains is shown in Figure 4.4 (b). In this image, each of the 10 total pressure chambers is connected to the control board (not pictured). On top of the robot is a visual marker for recording chassis motion via USB webcam.

Modeling Actuator Curvature

In order to model unloaded actuator deformation, we utilized a Vicon motion capture system to record the curvature of a single segment for a series of step inputs. We then fit a 2^{nd} order model to the recorded curvature and apply this curvature model to each segment of each serial chain appendage, assuming that the dynamics for each individually printed (but identical) piece were sufficiently similar, resulting in a dynamic model of the unloaded EFF position for each serial chain. It is assumed that the gravitational effects on the unloaded EFF location are negligible.

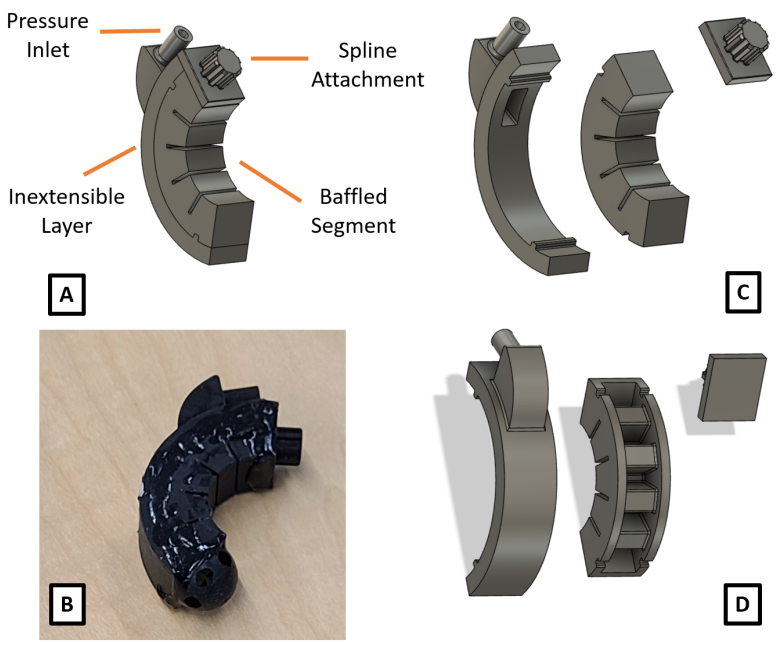

Figure 4.3: **a)** Model of single chamber carbon actuator. **b)** Printed and assembled single chamber actuator. **c,d)** Model of single chamber carbon actuator, exploded view, front and rear.

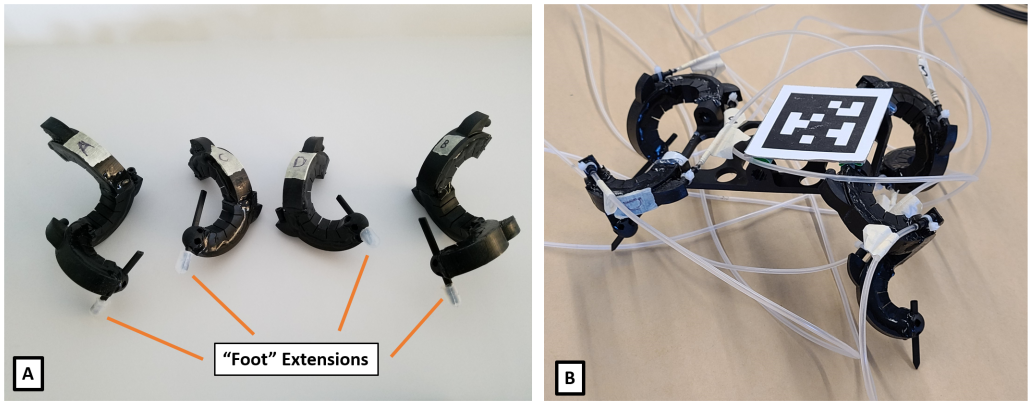

Figure 4.4: **a)** Example robot appendages, each constructed as a serial chain of identical segments of the single chamber actuator. **b)** Assembled robot with rigid polyurethane chassis. A visual tracking marker is mounted on top of the chassis for motion tracking purposes.

Measuring Actuator Stiffness and Damping

We model the stiffness of the actuator empirically by measuring the force/displacement of the EFF of the actuator in several different, steady state operating conditions. For example, characterizing a limb with three chamber segments (3 inputs) requires taking a series of force/displacement measurements for all 8 possible steady state pressurization inputs. From a set of force/displacement measurements taken in a given state, we compute a stiffness matrix by performing a linear fit to the force/deflection data. This is done for each set of force/displacement data. The subsequent stiffness matrix values as a function of curvature are then determined by interpolating between the measured, steady state stiffness matrices. This process is depicted in Figures 4.5, 4.6, and 4.7.

Shown in Figure 4.5 are the components used to collect data. Figure 4.5(a) depicts an actuator mount, clamped to a table and fitted with Vicon tracking markers (the silver orbs). Figure 4.5(b) depicts an EFF attachment also fitted with Vicon markers. The foot extension of the actuator to be tested (shown in (d)) is replaced by the EFF tracking attachment as shown in (e). The entire actuator with tracker setup is shown in Figure 4.5 (f). Shown in Figure 4.5 (c) is the load cell "wand," which is comprised of an Arduino UNO, a load cell, and a series of Vicon markers attached to a RPU printed frame. The silver appendage at the top of the image is the load cell, attached to which is a bolt that was used for alignment as seen in Figure 4.6.

The process of collecting stiffness data is shown in Figure 4.6. Depicted in the larger image is a cable driven EPU actuator, the development and operation of which is detailed in [5]. Note that as long as the functions specified in sec-

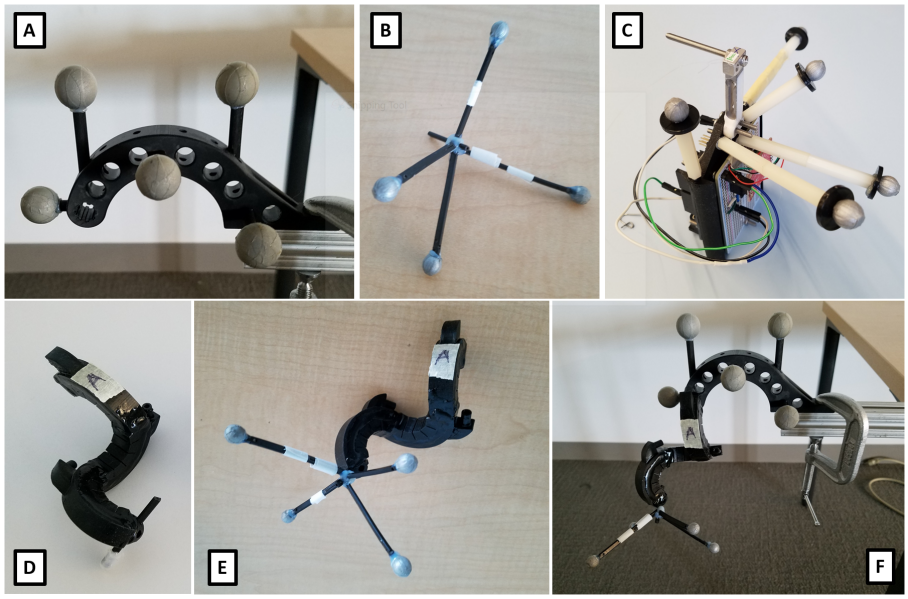

Figure 4.5: Components for measuring actuator stiffness. **a)** Mounting frame for actuator with Vicon markers (silver orbs). **b)** Vicon EFF tracking element. **c)** Load cell "wand" with Vicon tracking markers. **d)** Example actuator to be evaluated. **e)** Example actuator of **(d)** shown with EFF Vicon tracking element. **f)** Actuator with Vicon tracking element, mounted in tracking frame shown in **(a)**.

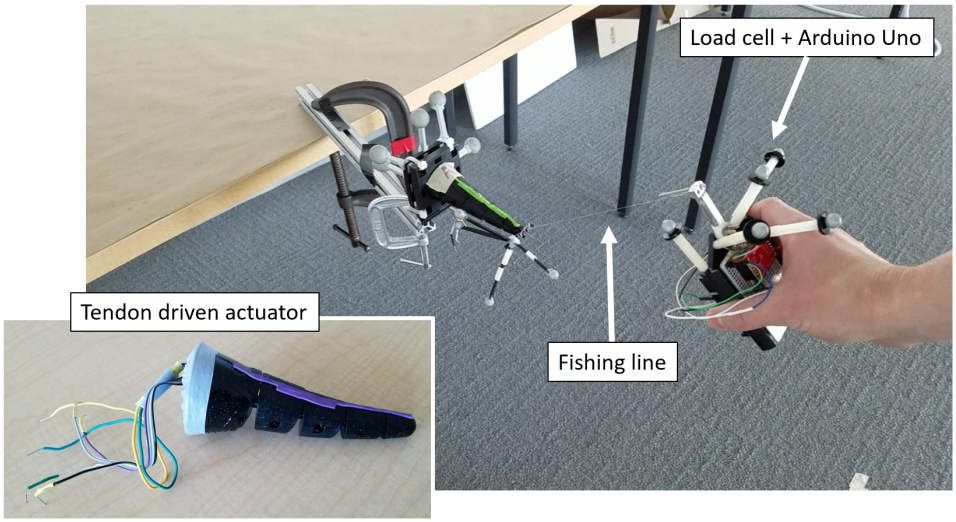

Figure 4.6: Measuring the stiffness of an actuator. The "wand" is used to displace the EFF of the actuator to be evaluated while load cell and Vicon data are recorded. Shown inset: cable driven EPU actuator.

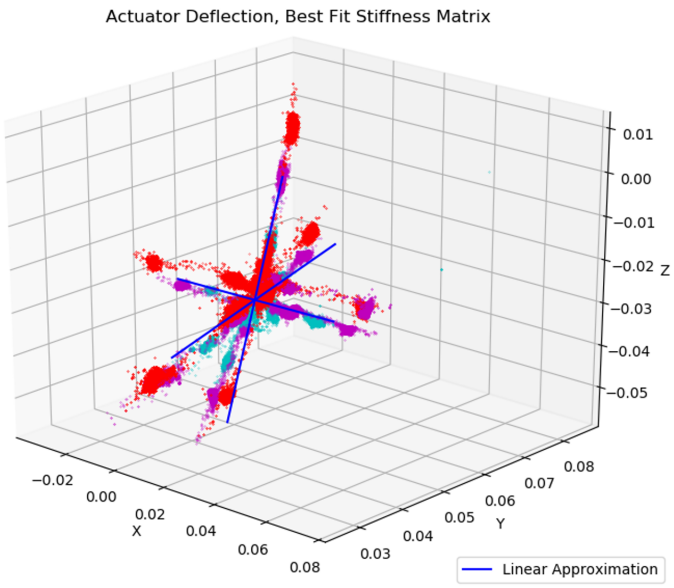

Figure 4.7: Resulting actuator EFF deflection data with linear fit.

tion 4.1.1 previously are defined, this model is actuation and material agnostic. The load cell attached to the wand is connected to the EFF of the actuator with fishing line, and the bolt on the load cell is used to align the attachment with the EFF. In this case, three sets of data were collected for each actuator - a series of small deflections, a series of medium deflections, and a series of large deflections. The locations of the actuator anchor, wand, and EFF tracking attachment were recorded continuously along with the load cell values. A resulting data set is shown in Figure 4.7, in which the three data sets are shown in different colors: light in cyan, medium in purple, and large in red. The linear fit applied to the aggregated data is depicted as a series of blue lines.

The damping coefficient was determined by exciting one of the actuators with a known mass as a step input, recording the response, and fitting a second order model to the data (with the known stiffness matrix). The damping coefficient is recorded once and is assumed to be constant for all appendages and appendage configurations. This is done to speed up the collection of data and

overall characterization process.

The combination of unloaded actuator dynamics (unloaded EFF position) and measured stiffness/damping provide a complete model for a robotic system composed from these EPU printed actuators. A comparison of the predicted and measured chassis displacement for a sequence of actuator inputs is presented in Section 4.3.

4.2.2 Silicone Actuators and Nylon Enclosures

The second soft robotic system we present is fabricated from two types of silicone: Smooth-On Ecoflex® and Smooth-On Dragonskin®. An image of this actuator type is shown in Figure 4.8. A side and top view of a representative actuator is shown in Figure 4.8 (a) and (b). The actuator is cured in two pieces - the body is cast in Ecoflex®, and the inextensible layer is cast in Dragonskin®. The molds are depicted in Figure 4.3 (c) and (d). Shown in (c) are the three total mold pieces one for the inextensible layer (far left) and two for the baffled, body section. The assembled body section mold is shown in (d). Embedded in the Dragonskin® inextensible layer is a piece of chiffon fabric. This piece of fabric acts as an inextensible material - as the actuator is pressurized, this planar piece does not extend and instead causes the actuator to "curl" around the inextensible material as shown in Figure 4.8(e).

Also depicted in Figure 4.8(e) is an actuator segment with a nylon "suit" enclosure. The nylon suit is designed to limit curvature and is similar in function to the folding structures of [54] and [76]. As mentioned in Chapter 2, cast silicone exhibits strain softening, which causes the level of actuation (curvature) to

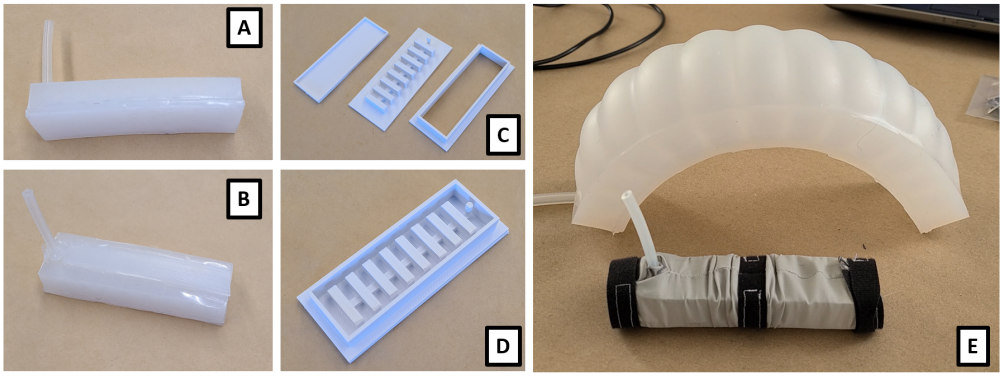

Figure 4.8: **a,b)**Ecoflex® actuator, side and top view. **c,d)** Actuator mold, 3D printed in PLA+. **e)** Foreground: actuator with grey nylon enclosure. Background: inflated actuator without enclosure.

increase with repeated pressurizations unless the actuator has been "broken in" by a series of deliberate pre-pressurizations. The first purpose of the nylon enclosure is to negate the need to pre-pressurize each actuator as the curvature is physically restricted. The second purpose is repeatability. The nylon enclosure is designed and sewn such that the curvature of the actuator is limited to a specific value. Subsequently, the maximum curvature of the actuator is constant, regardless of the input pressure. Increasing the input pressure only increases the rate of deformation, but the suit enclosure saturates the curvature at a specified value. This allows the stiffness of the inflated actuator to be tuned without also changing the range of motion.

A robot constructed from six identical silicone actuators is shown in Figure 4.9. Each of the four distal legs is a single segment actuator which curls down toward the ground surface, and the middle segment is a pair of opposing actuators that cause lateral curvature. A robot without nylon enclosures is shown in (a), and a robot with enclosures is shown in (c). Each actuator segment has its own enclosure and they are fastened together with Velcro. A robot with one segment enclosure removed is shown in Figure 4.9(b). Shown in (d) is the complete

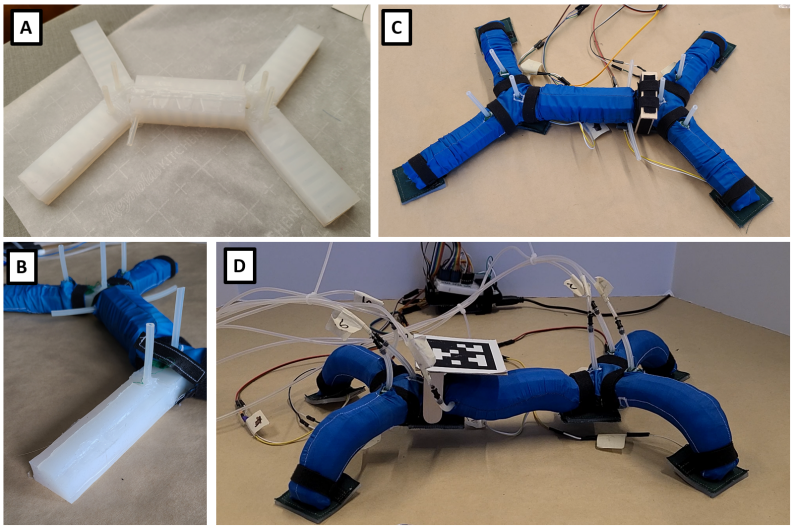

Figure 4.9: **a)** Robot composed from 6 silicone actuators. **b)** Robot complete with nylon enclosure. **c)** Robot with one silicone leg exposed. **d)** Robot complete with enclosure, all pressure chambers inflated.

robot with all actuators inflated. This particular robot contains two small sections that are pressurized but do not deform - one on either end of the double chamber middle actuator segment. These sections, which serve as the attachment point for the limbs, are included to simplify the process of putting on and securing the nylon enclosures.

Active and Passive Deformations

As mentioned above, the deformation specific model we describe divides the robot state space in to *active* and *passive* state variables. We make the same curvature assumptions with this actuator type as with the EPU actuator described above - that the curvature is approximately constant, and this constant curvature mode of deformation is considered an active degree of freedom. However, the middle segment of this robot design exhibits a second main mode of deformation: torsion. A demonstration of this mode of deformation is shown in

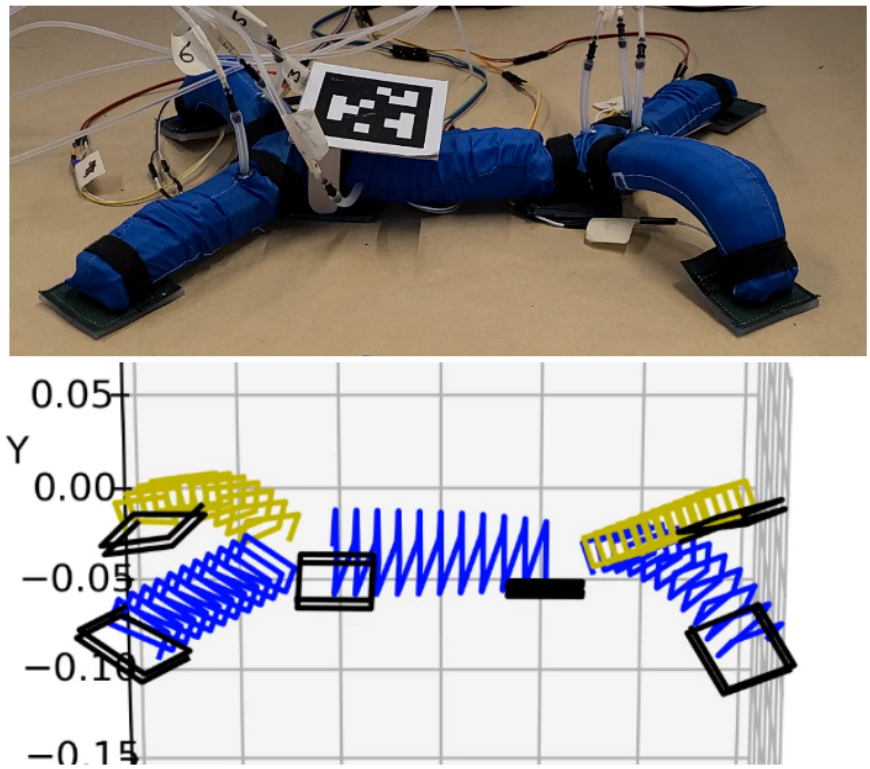

Figure 4.10: Above: example of silicone robot with opposite front/rear limbs pressurized, resulting in torsion of the middle actuator. Bottom: wireframe representation of the same robot.

Figure 4.10. This degree of freedom is considered passive and is governed primarily by internal forces and moments. We represent the curvature and torsion of this actuator using the Frenet-Serret formulas [37], integrated for constant curvature and torsion values. We further include two passive hinge joints in the model where each front/rear leg pair are attached to the middle segment. It was observed during operation that these particular attachment locations were prone to deflection. As the passive degrees of freedom of this robot have no associated dynamics, we determine their values via one of two methods: optimization in the case of simulation, and pose estimation in the case of the physical robot.

In the case of simulation, we determine the values of the unmodeled degrees of freedom by solving a minimum-energy optimization problem at each simulation time step that minimizes the height of the center of gravity of the robot but constrains the boundaries of the actuators to remain above the ground surface. In this case, we represent the boundary of each actuator as a wireframe as depicted in Figure 4.10 (bottom) and constrain the y coordinate (vertical axis) of each wireframe vertex be non-negative. We restrict the range of the passive hinge joints by imparting bounds on the values of the joint. These bounds, representing the rotation values at which the joints effectively "bind," were determined empirically. This optimization, though appropriate for simulation, was too slow to run online. As such, we implemented contact sensing and pose estimation to determine passive state values on the physical robot.

Sensing Contact and Online Pose Estimation

In order to sense contact, we developed the touch sensors as shown in Figure 4.11, which are comprised of two parts: a pressure sensitive, conductive material (Velostat fabric, Adafruit part no. 1361) with leads attached via conductive fabric and tape, and an enclosure comprised of a canvas backing with either a silicone or canvas contact pad. The underside of a fully instrumented, nylon enclosed robot is shown in Figure 4.11 (a). Six sensors are attached - one at the end of each appendage (shown with white/clear silicone contact pads) and two on the middle segment (shown with green canvas contact pads). Two enclosures (front and back) are depicted in Figure 4.11 (b). An enclosure (top view) with sensor inserted is shown in (c). Sewn to the canvas backing is a strip of Velcro for attachment to the robot. Figure 4.11 (d) shows the underside of the same

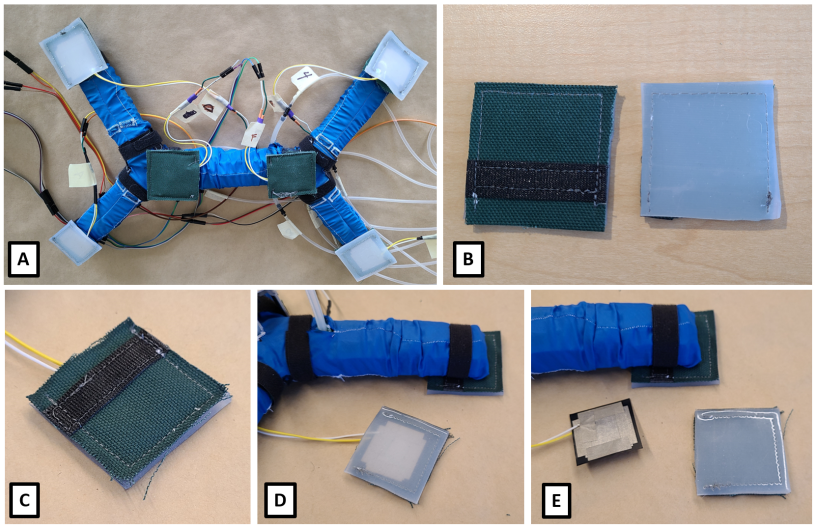

Figure 4.11: Touch sensor developed for pose estimation. **a)** Instrumented, nylon enclosed robot. Six sensors are attached to the robot - one at the end of each appendage (shown with silicone pads) and two on the middle segment (shown with green canvas pads). **b)** Touch sensor enclosure composed of a canvas backing with a Velcro strip and Ecoflex® pad. **c)** Contact pad with sensor inserted. **d)** Underside of silicone touch pad with sensor inserted. **e)** Touch pad enclosure with sensor removed.

sensor next to a robot appendage with a touch sensor attached. Figure 4.11 (e) shows the enclosure with sensor insert removed.

Each touch sensor is sensitive to bending as well as contact pressure, and those touch sensors at the end of each appendage experienced both modes of deformation during operation. As such, we treat each sensor as a Boolean value rather than attempt to characterize these two modes of sensitivity and determine the exact load on each contact point. Threshold sensor values are determined empirically each time the robot is operated. We then determine the pose of the robot, i.e. the values of the total state of the robot, both active and passive, via an Extended Kalman Filter [4].

Robot Angular Momentum

The rigid chassis and spring-damper limb abstraction of the model presented in Section 4.1.1 present a complete model for the motion of the robot. The active/passive model, in contrast, does not assume a central, rigid chassis and such, we do not utilize centroidal dynamics to represent the motions of the robot during operation.

The dynamics of the robot are computed in two different phases. The first is a conservation of angular momentum calculation. The locations of the centers of mass of each actuator are treated as a mass-particle system and at each time step the planar rotation of a body fixed reference frame is computed such that the angular momentum of the particle system is conserved. The second step is to compute the frictional forces acting on the robot. In the case of the silicone robot, we apply a viscous friction model. Each vertex of the wireframe representation is assigned a friction coefficient value offline, and we compute during simulations the normal loads on each vertex point via an optimization that finds the minimum average vertex load that satisfies static force/moment constraints for any vertex point within some ϵ of the ground surface.

Generating the gait graph

For the purposes of model evaluation as a precursor to gait synthesis, we generate the gait graph for this robot, thereby extending the gait graph concept from Chapter 2. Computing the gait graph is done by first finding the steady state values of each of the passive state variables for each valve state. This is done via optimization - minimizing the height of the CG as previously discussed.

We then simulate each of the pose transitions and the resulting planar motion (displacement and rotation) is recorded in the graph.

4.3 Model Validation

4.3.1 Robot Control

Both robots utilize pneumatic actuators and flow to both is controlled by a set of three-way solenoid valves. Similar to the control scheme described in Chapter 2, each valve is at any given time exposed to a common rail pressure or to the atmosphere. As such, each actuator is either pressurized or depressurized and control inputs are subsequently modeled as Boolean variables. We define a controller as a sequence of Boolean (open/closed) valve state vectors and a fixed controller frequency - at each time step the next sequential valve state vector is applied.

4.3.2 Model Validation: Generalized Spring Damper Representation.

Figures 4.12 and 4.13 show a comparison of the generalized spring damper model and chassis motion data collected via USB webcam. The robot was subjected to an open loop valve state sequence and the position and orientation of the visual marker on top of the robot were recorded. In this case, the robot was evaluated on two different surfaces: bench paper and carpet, and the EFF ex-

tensions used terminated in RPU material rather than the silicone as previously shown. The intention of operating the robot on a carpeted surface with RPU EFF extensions was to negate the complications of friction - ideally the sharp, RPU EFF extensions would not exhibit any sliding friction, and, subsequently make the comparison between the model and actual robot behavior more straightforward. The simulated robot, therefore, assumes a no-slip condition for the EFF contacts (i.e. there is no friction model included in this instance). Note - the simulated robot does not begin the simulation from a steady state condition. Rather than solve for the steady state leg spring displacements during each simulation, the robot is "dropped" from a small height and the system allowed to settle, as can be seen in the fluctuations of the position and orientation values at appx. $t = 0$.

Figure 4.12 shows the $x/y/z/$ position of the chassis. In this case, the X axis is aligned with the "forward" orientation of the chassis (to the right in Figure 4.4(b), the Y axis is oriented vertically, and the Z axis subsequently oriented to the "right" of the chassis. Overall, the model tracks the general trends of the the motion of the physical robot, but doesn't align closely with the measured chassis displacements. There are two main reasons for this: 1) error in measuring the stiffness of each actuator, and 2) errors in contact modeling.

To address the first issue, errors in measuring actuator stiffness, we attempted to find optimal leg stiffness values for the data set collected on the carpet. As mentioned previously, the stiffness of each limb as a function of limb state is determined by interpolating linearly between stiffness matrices measured at steady state limb configurations. As such, each element of the $[3x3]$ stiffness matrix for each of the four limbs contains several parameters for this

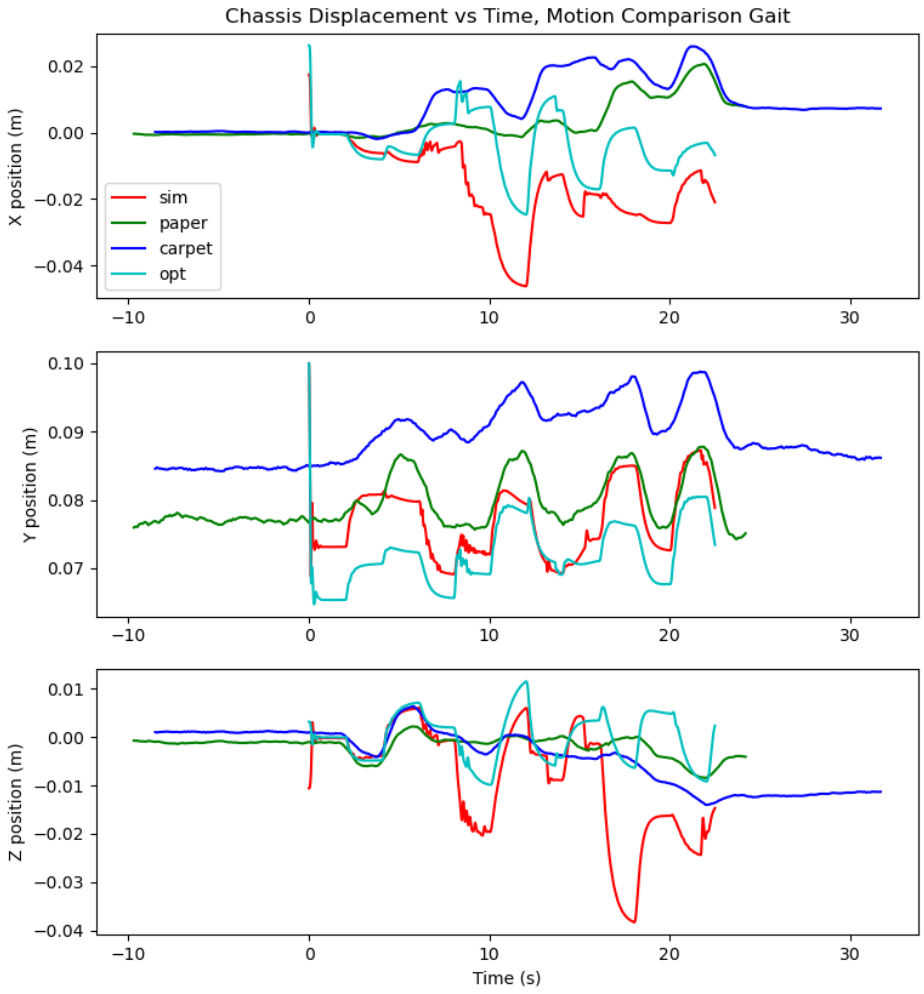

Figure 4.12: Generalized spring damper model performance - chassis position.

linear interpolation function, resulting in a large parameter space should we wish to find the optimal linear fit for each element of each stiffness matrix. As such, rather than attempt to find the optimal values for each linear expression for each element of each matrix, we optimize over four parameters - one stiffness matrix modifier for each limb, α, such that the value of the stiffness matrix is then $\alpha_i * K_i$ for limb i. We performed a bounded minimization on the aggregate

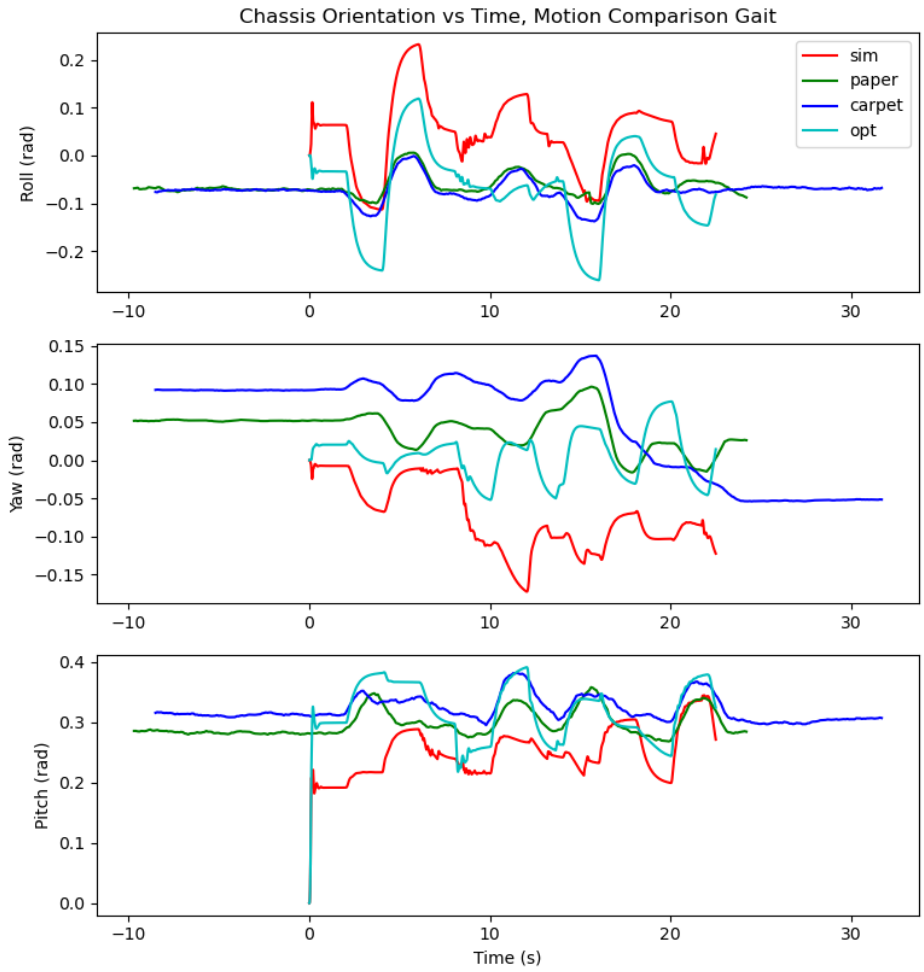

Figure 4.13: Generalized spring damper model performance - chassis orientation.

chassis position and orientation error and the results are shown in cyan (labelled "opt"). The modified stiffness matrices showed better performance in matching the orientation of the chassis than the position, but the overall the results were improved.

Optimizing the stiffness values for each limb, while effective in improving the accuracy of the model, does not mitigate the second major factor contribut-

ing to model error: errors in limb contact. Specifically, this refers to modeling when a limb (EFF) breaks or makes contact with the ground surface. It was observed during each trial on the physical robot that the rear limbs broke contact several times and that these contact events were not reflected in simulation. The nature of the compliance of the limbs of the robot means that the chassis of the robot is often subject to step-input like disturbances when a limb breaks contact with the ground surface - in effect, a limb breaking contact causes discontinuities in the forces applied to the chassis. Creating an effective model requires the ability to accurately predict these contact events.

4.3.3 Model Validation: Active/Passive Model.

Figures 4.14, 4.15, and 4.16 show a comparison of the specific deformation model against chassis motion data collected via USB webcam. As was the case with the generalized model, the silicone robot was subjected to an open loop valve sequence. Three trials were conducted, each for different materials placed under the touch pads: nylon (low friction), canvas (some friction), and nothing (the as-fabricated silicone pads). The measured chassis displacement for each trial is compared against the EKF (using the Boolean touch sensors), the simulated robot, and the motion predicted by the gait graph. The model and the EKF, in this case, utilizes a viscous friction model along with with coefficients of friction empirically determined for each material.

As is the case with the EPU robot and the generalized spring/damper model, the trends of the model align with those of the measured chassis motion (those of the lateral, y, displacement and chassis orientation angle), but the exact val-

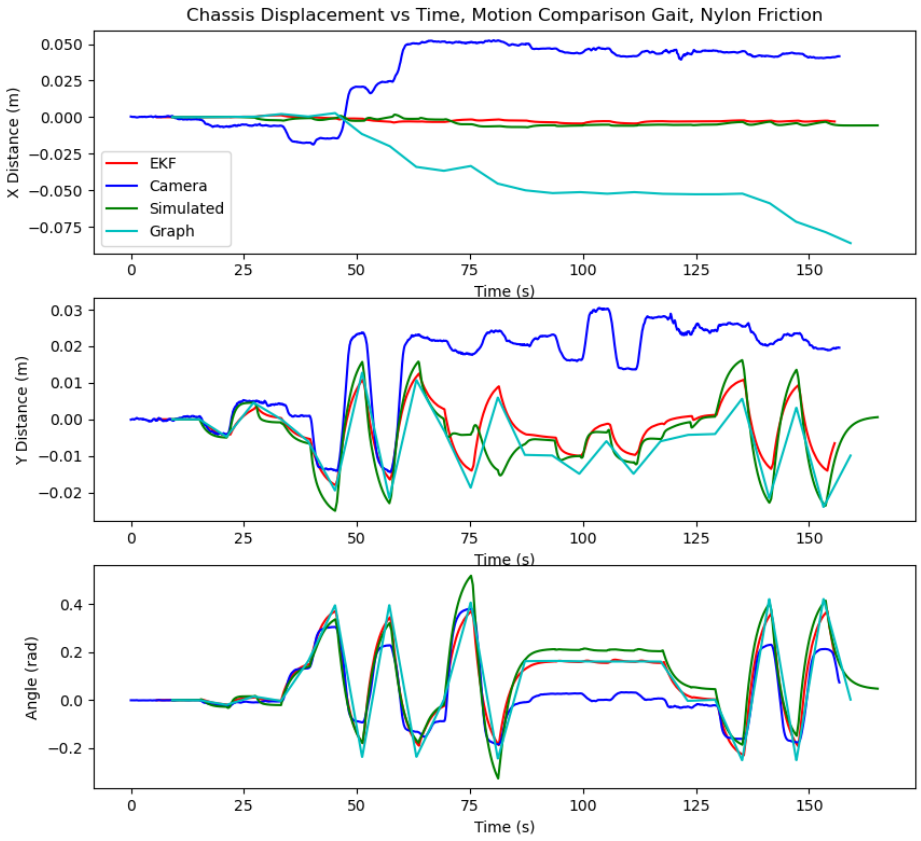

Figure 4.14: Active/passive model performance - nylon friction pads.

ues do not correlate. Of note is the deviation between the simulated values (shown in green) and the gait graph predicted values (shown in cyan). The gait graph, as previously discussed, is generated in this case by simulating the robot transitioning between poses. However, due to the large number of possible transitions, over 4000 in this case, the gait graph simulations are conducted at a larger time-step than the simulation conducted for these motion comparisons. This difference has a significant impact on the gait synthesis process, as will be discussed in the following chapter.

Also of note is the similar behavior of the robot in the x direction in each trial

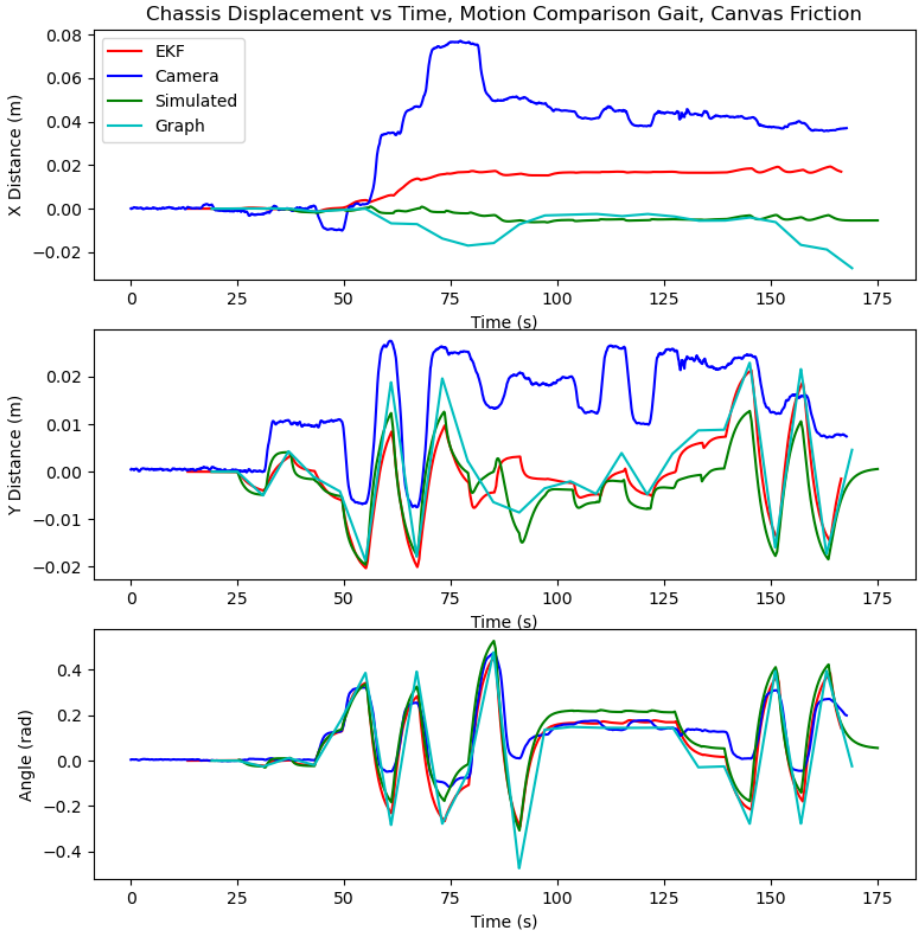

Figure 4.15: Active/passive model performance - canvas friction pads.

at approximately $t = 40$. The valve sequence (identical for all trials) contains valve assignments that briefly mimic a "salamander" type gait at this point, causing the robot to propel itself forward. Though the exact motion varies between friction material, the robot moves forward in all cases and it is at this point that the simulated values in the x direction diverge significantly for each trial. The main reason for this is error in the friction model. It was observed during the trials that the robot contacts do not adhere to viscous friction behavior,

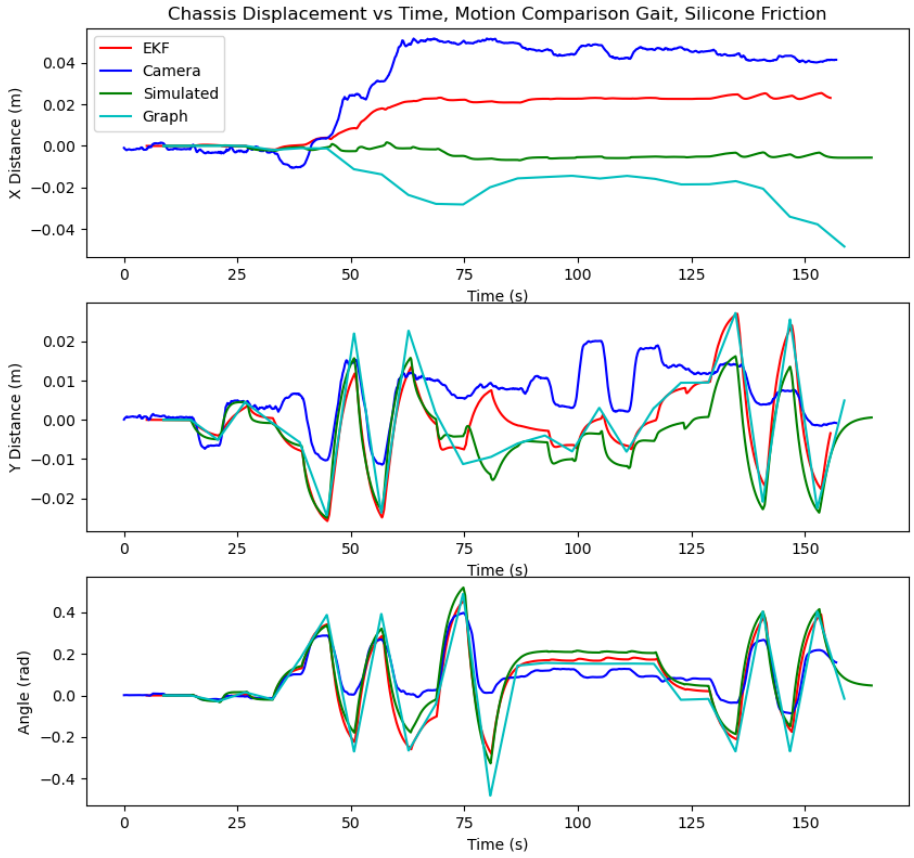

Figure 4.16: Active/passive model performance - silicone friction pads.

rather, the contact points exhibit static/dynamic friction behavior, which is difficult to model if the exact values of the static and dynamic friction coefficients are not known.

4.3.4 Overcoming modeling limitations.

The models detailed in this chapter are not (as currently presented) sufficiently accurate to use for gait synthesis in simulation. In addition the EKF result for

the silicone robot is also insufficiently accurate to estimate motion (dead reckoning via touch-sensor) online. For example, there are areas of the state space in which the passive states of silicone robot model are not observable - namely when the two middle touch sensors are not in contact with the ground surface. These limitations can be overcome with further sensing capabilities but there exist obstacles for both models.

The generalized spring-damper model is predicated on what can be considered "virtual" quantities - that is, the unloaded EFF positions of each limb. This quantity, the at-rest spring geometry, cannot be measured directly once the robot is in contact with a ground surface, and the limbs of the robot may not be directly instrumented as there are no corresponding explicit modes of deformation in the robot state vector. The only quantities available to measure directly are the height and orientation (roll, pitch, and yaw) of the chassis.

Instrumenting the silicon robot, in comparison, is considerably more straightforward as the state values correlate directly to modes of deformation and may therefore be directly measured, but there still exist problems with implementing accurate contact sensing. As previously discussed, the contact sensors attached to the silicone robot are sensitive to both compression and bending and are subsequently only used to detect contact. Implementing a contact sensor that provided an accurate load value is difficult due to the nature of the contact of the limbs - the bottom of the appendage is in contact with the ground surface when the actuator is flat, but the end of the appendage is in contact when the limb is pressurised (as can be seen in Figure 4.10), effectively creating a "rolling" type contact, which is difficult to instrument. The EKF performance would be enhanced if the loading distribution of the robot were measured di-

rectly rather than inferred from the state estimate, and if the passive modes of deformation (hinges and torsion) were also instrumented.

4.4 Discussion

This Chapter presents (1) two different methods for modeling ambulating soft robots that consider actuator compliance and specific actuator deformations, (2) two different physical systems, and (3) an evaluation of the performance of both models. As presented, both of the models are insufficiently accurate for use in gait synthesis for the presented robot systems.

The problem of modeling the infinite degrees of freedom of compliant materials remains an obstacle for creating accurate models of soft robots. However, modeling errors may be overcome with improved sensing and state feedback. Despite the limitations of the Boolean contact sensors, unobservable modes, and friction model errors of the silicone robot, the EKF state estimate was more accurate than the simulated robot behavior. Further development of these models will focus on techniques to improve sensing capabilities and improve contact models.

CHAPTER 5

TOWARDS GAIT SYNTHESIS FOR COMPOSED SOFT ROBOTS

5.1 Introduction

In the previous chapter, we described two different modeling methods for soft robots and presented two different example robots - one modeled using each method. In this section, we present a discussion on leveraging the previously presented models for gait synthesis. To that end, we seek to leverage the gait graph concept presented in Chapter 2. All robot hardware presented in this work utilizes the same control strategy - pneumatic actuators controlled by three-way solenoid valves, the states of which are represented as Boolean variables. This type of control scheme is represented well by the discrete gait switching and pose-to-pose control methods presented in previous chapters.

One of the main drawbacks to the graph search method presented in Chapter 2 is that the edge weights are all specified according to motions presented in a chassis centered frame. That is, there is no notion of global chassis displacement during the graph search process and the global, planar motion of the robot was not assessed until the termination of the algorithm. This section discusses two different graph search gait synthesis methods applicable to the aforementioned models and control system - hybrid graph search and Monte Carlo Tree Search (MCTS) that seek to synthesize gaits in a global frame.

Hybrid Graph Search. The authors of [57] utilize a hybrid graph search algorithm to generate motion plans for an autonomous vehicle. In this formulation, the state or configuration space of the vehicle is decomposed, forming a

graph over which the shortest path search technique A* [27] is used to generate a motion plan. However, included in this search algorithm is a control history and subsequent trajectory of the vehicle. The adjacency relation for the graph search is not determined by the configuration space decomposition, but rather by the propagation of the dynamics of the vehicle. In other words, cell adjacency is determined by simulating the dynamics of the vehicle forward in time. The control history is retained during search and is returned with the final motion plan result. We leverage this concept to extend the graph search method presented in Chapter 2 to search over global chassis displacements.

Monte Carlo Tree Search. The authors of [10] present a method of gait generation for a robot that utilizes Monte Carlo Tree Search to search over sequences of possible actuator commands for a biped robot in a cluttered environment. In this case, the search process utilizes a set of motion primitives as possible control inputs. This method is useful in situations in which the branching factor of possible motion commands is high and we propose a similar method in this section.

5.2 Offline and Online Gait Search

The models presented in Chapter 4 provide a fully dynamic representation of a soft robot that may then be simulated to predict robot motion. Were the models sufficiently accurate, gaits could be generated in simulation and then demonstrated on the physical system. In practice, however, the error of both models is sufficient to preclude the possibility of generating a gait entirely in simulation. As such, we propose a hybrid of offline and online gait synthesis. We apply this

combination method only to the silicone based robot due to the aforementioned difficulties in state estimation with the 3D printed, EPU robot.

5.2.1 Offline Transition Prediction

As previously stated, the control inputs each robot are discrete and applied at a fixed interval, in this case, approximately 6 seconds. As such, we can utilize the dynamic model of the silicone robot to pre-generate all possible motion transitions. The 6 DOF silicone robot showed in Figure 4.9 in Chapter 4 has $2^6 = 64$ possible poses and subsequently $64^2 = 4096$ possible transitions. Note - this may not be feasible for a robot with more than 6 pressure chambers, but in this case processing the entirety of the gait graph online is feasible. The gait graph is then utilized as the basis for both methods - hybrid graph search and MCTS.

5.2.2 Gait synthesis through hybrid graph search

Similar to the authors of [57], we discretize the configuration space, in this case the 3 DOF planar motion of the robot dx, dy, $d\theta$, and determine cell adjacency by propagating the dynamics from a given cell using the gait graph. Each cell that has been visited has associated a shortest path control history. Should a cell be visited more than once according to the gait graph propagation, the overall shortest path is retained and any previous path history discarded. In this manner, it is possible to generate a control history (a gait) driving the robot from a given initial state to a feasible desired final state, dynamics permitting. Self transitions, that is, transitions that fail to move the robot from the current cell

are discarded.

However, as previously stated, modeling errors preclude an entirely offline gait-generation approach. As such, we leverage the algorithms D* Lite [32], and Lifelong Planning A* (LPA*) [33], which address the problem of updated edge weights during graph search. The central idea in this case is to use the online EKF feedback to update gait graph transition values (and subsequently edge weights) as the robot executes a gait, thereby refining the gait graph online. The difference between D* Lite and LPA* is the direction of search: LPA* searches from an initial node to a goal node and updates weights accordingly, and D* Lite works in reverse - from the goal to the start node. Both algorithms are applicable in this case depending on when gait graph edge weights are updated during gait execution. If the robot updates the gait graph at each transition, the D* will continue to drive the robot to a goal state despite deviations. If an entire gait cycle is executed before the gait graph is updated, LPA* is more useful for synthesizing a gait from a desired initial condition. We utilized LPA* in this work.

As stated, the gait graph is first generated offline. The robot subsequently synthesizes a gait for a desired chassis displacement, and the robot executes the gait. As the robot is operating, the pose of the robot is estimated with the EKF and the displacement is estimated from the pose and incorporated friction model (dead reckoning) as previously described in Chapter 4. As the robot executes each transition in the gait, the gait graph is updated according to the estimated displacements, the executed transitions are marked "explored", and the shortest path is then updated according to the modified edge weights. The algorithm then runs until a gait is synthesized in which all gait transitions have

been explored. This process is detailed in Algorithm 2.

Algorithm 2 Hybrid Graph Search Synthesis

```
Synthesize gait via gait graph, G
while Gait sequence not fully explored do
    Synthesize gait via gait graph
    Execute gait and record displacements via EKF
    Update edge weights per D* or LPA*, marking as explored
end while
```

5.2.3 Issues With Hybrid Graph Search

We determined that while hybrid LPA* was capable of synthesising offline gaits for the gait graph associated with the silicone robot, the edge weight updating process was time consuming - approximately 15 minutes of run time to update one edge of the graph. This was largely due to the granularity of the configuration space decomposition - the small displacements estimated in the offline gait graph computation necessitated a fine configuration space discretization, leading to a large configuration graph. We subsequently evaluated a second synthesis method, hybrid MCTS.

5.2.4 Monte Carlo Tree Search

Monte Carlo Tree Search works similarly to the hybrid graph search approach but rather that decompose the configuration space and evaluate adjacency during a standard graph search process, MCTS generates a tree graph by propagating each leaf of the tree (a pose), recording the continuous configuration space state of the robot (along with the control history) as a new node of the tree graph,

and then deciding which new nodes to propagate further though a weighting process that involves "playing out" new nodes. The playout process involves executing a sequence of random moves from a given node and then recording the total chassis displacement as the "score" of the playout and then updating the weights of each node according to the UCB1 criterion [3]. As with the hybrid graph search method, a gait may be synthesize offline by using the gait graph to propagate the state of the robot.

The hybrid graph search approaches (LPA* and D*) propagate cost-to-go and cost-from-goal values each time an edge is updated. Our proposed MCTS method, in contrast, does not propagate costs but rather prunes branches entirely from the tree each time an edge is traversed and its true displacement (according to the EKF) determined. Once a branch is pruned, a rollout is performed for each leaf from which the pruned transition was previously propagated. This processes is described in Algorithm 3.

Algorithm 3 MCTS Synthesis

```
Synthesize gait via gait graph, G
while Gait sequence not fully explored do
    gait ← control history of leaf with best score
    Execute gait and record displacements via EKF
    Prune tree for each explored gaits and perform new rollouts
end while
```

5.3 Bridging the Gap

Though the MCTS method was faster in exploring the gait space of the robot, neither method was successfully implemented on a robot. There are two main issues that need to be addressed for the synthesis methods proposed above to

be viable: improving the contact model and improving the state estimate.

As discussed in Chapter 4, the contact (friction) model is they key to any reliable estimate of robot locomotion. It is central to both the offline simulation and gait graph generation as well as the online EKF state estimate. Though some error may be tolerated, a good initial estimate of the motion of the robot (offline gait graph) and good estimate of the executed transitions (EKF feedback) are required in order for a gait to be successfully synthesized.

5.4 Discussion

This chapter presents the gait synthesis methods intended for use with the models presented in Chapter 4. Both of these methods are predicated on accurately simulated and measured estimates of soft robot motion, and, as discussed in Chapter 4, there are significant hurdles to creating accurate models of soft robot locomotion. Future work with these synthesis methods will involve improving state estimation and friction models for the soft robot utilized in this work.

CHAPTER 6

CONCLUSION

This work addresses the issues of gait synthesis and resilient gait behavior for ambulating soft robots. The infinite degrees of freedom and large deformations of soft materials present a litany of challenges for modeling and controlling soft robots. Though there exist a variety of methods for quantifying soft material behavior and generating dynamic models of soft systems, they are typically highly specific to a given robot morphology or composition, or they are computationally intensive. As such we focus on generalizable methods for representing and abstracting soft robot behavior that may subsequently be used to both synthesize gaits for ambulating systems, and synthesize gait controllers for soft robots that are resilient to actuator failure.

Chapter 2 details the work in [25], which presents a graph search based gait synthesis method for modular soft robots that is agnostic to soft actuator dynamics and specific modes of compliance. Several gaits synthesized using this method are demonstrated on a physical system. Chapter 3 describes the work in [26], which presents a method for synthesizing gait controllers for soft robots that are resilient to actuator failure. This method, which involves a sensor based abstraction of actuator health and performance, provides the ability to synthesize correct-by-construction controllers that enable reactive behaviors for robots operating in adversarial environments.

The work presented in chapter 4 seeks to extend the modeling concepts presented in Chapter 2 by addressed two of the main complications of the simplistic model - friction and specific modes of deformation. The models presented in Chapter 4 provide the ability to characterize actuator compliance and specific

modes of deformation without resulting to highly complex models, as well as allow for the inclusion of friction models. Chapter 5 subsequently seeks to extend the synthesis concepts of Chapter 2 by proposing a hybrid offline/online gait synthesis process that leverages the models presented in Chapter 4.

The models presented in Chapter 4, however, are, as presented, insufficiently accurate for use with the gait synthesis methods proposed in Chapter 5. Specifically, the contact dynamics (e.g. friction) do not accurately represent the observed behavior of the robotic systems developed to test and validate the models. These shortcomings may be overcome by improving the friction model as well as improving the sensing capabilities of the soft robot systems, thereby improving motion prediction as well as robot behavior feedback. As such, future work will focus on further robot instrumentation and a more thorough analysis of the collected motion data.

BIBLIOGRAPHY

[1] Amir Ali Amiri Moghadam, Seyedhamidreza Alaie, Suborna Deb Nath, Mahdie Aghasizade Shaarbaf, James K. Min, Simon Dunham, and Bobak Mosadegh. Laser cutting as a rapid method for fabricating thin soft pneumatic actuators and robots. *Soft Robotics*, 5(4):443–451, 2018.

[2] Alfredo Argiolas, Benjamin C. Mac Murray, Ilse Van Meerbeek, John Whitehead, Edoardo Sinibaldi, Barbara Mazzolai, and Robert F Shepherd. Sculpting soft machines. *Soft Robotics*, 3(3):101–108, 2016.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, (47):235–256, 2002.

[4] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley & Sons, 2004.

[5] Jose Barreiros, Kevin W. O'Brien, Samantha Hong, Michael F. Xiao, Ho Jung Yang, and Robert F. Shepherd. Configurable tendon routing in a 3d-printed soft actuator for improved locomotion in a multi-legged robot. In *IEEE International Conference on Soft Robotics*, pages 94–101, 2019.

[6] James M. Bern, Grace Kumagai, and Stelian Coros. Fabrication, modeling, and control of plush robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3739–3746, 2017.

[7] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'Ar. Synthesis of Reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.

[8] Leonardo Cappello, Kevin C. Galloway, Siddharth Sanan, Diana A. Wagner, Rachael Granberry, Sven Engelhardt, Florian L. Haufe, Jeffrey D. Peisner, and Conor J. Walsh. Exploiting textile mechanical anisotropy for fabric-based pneumatic actuators. *Soft Robotics*, 5(5):662–674, 2018.

[9] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 11–23, 2013.

[10] Patrick Clary, Pedro Morais, Alan Fern, and Jonathan Hurst. Monte-Carlo planning for agile legged locomotion. *International Conference on Automated Planning and Scheduling (ICAPS)*, 28(1):446–450, 2018.

[11] Francesco Corucci, Marcello Calisti, Helmut Hauser, and Cecilia Laschi. Novelty-based evolutionary design of morphing underwater robots. In *Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 145–152, 2015.

[12] Francesco Corucci, Nick Cheney, Francesco Giorgio-Serchi, Josh Bongard, and Cecilia Laschi. Evolving soft locomotion in aquatic and terrestrial environments: Effects of material properties and environmental transitions. *Soft Robotics*, 5(4):475–493, 2018.

[13] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[14] Hongkai Dai and Russ Tedrake. Planning robust walking motion on uneven terrain via convex optimization. In *IEEE-RAS International Conference on Humanoid Robots*, pages 579–586, 2016.

[15] Jin Dai, Hossein Faraji, Chaohui Gong, Ross L. Hatton, Daniel I. Goldman, and Howie Choset. Geometric swimming on a granular surface. In *Robotics: Science and Systems*, pages 1–7, 2016.

[16] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *IEEE-RAS International Conference on Humanoid Robots*, pages 279–286, 2014.

[17] Dylan Drotman, Saurabh Jadhav, Mahmood Karimi, Philip de Zonia, and Michael T. Tolley. 3D printed soft actuators for a legged robot capable of navigating unstructured terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5532–5538, 2017.

[18] Christian Duriez. Control of elastic soft robots based on real-time finite element method. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3982–3987, 2013.

[19] Christian Duriez and Thor Bieze. Soft robot modeling, simulation and control in real-time. *Biosystems and Biorobotics*, 17:103–109, 2017.

[20] Rüdiger Ehlers and Vasumathi Raman. Slugs: Extensible GR(1) synthe-

sis. In *28th International Conference on Computer Aided Verification (CAV)*, volume 2, pages 333–339. Springer, Cham, 2016.

[21] Mustafa Suphi Erden and Kemal Leblebicioğlu. Free gait generation with reinforcement learning for a six-legged robot. *Robotics and Autonomous Systems*, 56(3):199–212, 2008.

[22] Wyatt Felt, Khai Yi Chin, and C. David Remy. Smart braid feedback for the closed-loop control of soft robotic systems. *Soft Robotics*, 4(3):261–273, 2017.

[23] Siyuan Feng, X. Xinjilefuz, Christopher G. Atkeson, and Joohyung Kim. Robust dynamic walking using online foot step optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5373–5378, 2016.

[24] Moritz Geilinger, Roi Poranne, Ruta Desai, Bernhard Thomaszewski, and Stelian Coros. Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels. *ACM Transactions on Graphics*, 37(4):1–12, 2018.

[25] Scott Hamill, Bryan Peele, Peter Ferenz, Max Westwater, Robert F. Shepherd, and Hadas Kress-Gazit. Gait synthesis for modular soft robots. In *International Symposium on Experimental Robotics*, pages 669–678. Springer, Cham, 2016.

[26] Scott Hamill, John Whitehead, Peter Ferenz, Robert F. Shepherd, and Hadas Kress-Gazit. Resilient task planning and execution for reactive soft robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5148–5154, 2019.

[27] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics*, 4(2):100–107, 1968.

[28] Jonathan Hiller and Hod Lipson. Dynamic simulation of soft multimaterial 3D-printed objects. *Soft Robotics*, 1(1):88–101, 2014.

[29] Filip Ilievski, Aaron D. Mazzeo, Robert F. Shepherd, Xin Chen, and George M. Whitesides. Soft robotics for chemists. *Angewandte Chemie - International Edition*, 50(8):1890–1895, 2011.

[30] Matthew Ishige, Takuya Umedachi, Tadahiro Taniguchi, and Yoshihiro Kawahara. Exploring behaviors of caterpillar-like soft robots with a central pattern generator-based controller and reinforcement learning. *Soft Robotics*, 6(5):579–594, 2019.

[31] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica*, 32(5):922–923, 1976.

[32] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[33] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong planning A*. *Artificial Intelligence*, 155(1-2):93–146, 2004.

[34] Sylvain Koos, Antoine Cully, and Jean-Baptiste Mouret. Fast damage recovery in robotics with the T-resilience algorithm. *International Journal of Robotics Research*, 32(14):1700–1723, 2013.

[35] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *SIGGRAPH - Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, 2002.

[36] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:211–247, 2018.

[37] Wolfgang Kühnel. *Differential geometry: curves - surfaces - manifolds*. American Mathematical Society, 2002.

[38] Naveen Kumar Uppalapati, Benjamin Walt, Aaron Havens, Armeen Mahdian, Girish Chowdhary, and Girish Krishnan. A berry picking robot with a hybrid soft-rigid arm: Design and task space control. In *Robotics: Science and Systems*, 2020.

[39] Sen W. Kwok, Stephen A. Morin, Bobak Mosadegh, Ju-Hee Hee So, Robert F. Shepherd, Ramses V. Martinez, Barbara Smith, Felice C. Simeone, Adam A. Stokes, and George M. Whitesides. Magnetic assembly of soft robots with hard components. *Advanced Functional Materials*, 24(15):2180–2187, 2014.

[40] Shuguang Li, Samer A. Awale, Katharine E. Bacher, Thomas J. Buchner, Cosimo Della Santina, Robert J. Wood, and Daniela Rus. Scaling up soft

robotics: A meter-scale, modular, and reconfigurable soft robotic system. *Soft Robotics*, 9(2):324–336, 2022.

[41] Huai-Ti Lin, Gary G. Leisk, and Barry Trimmer. GoQBot: a caterpillar-inspired soft-bodied rolling robot. *Bioinspiration biomimetics*, 6(2):026007, 2011.

[42] Matteo Lodi, Andrey Shilnikov, and Marco Storace. Design of synthetic central pattern generators producing desired quadruped gaits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(3):1028–1039, 2018.

[43] Andrew D. Marchese, Cagdas D. Onal, and Daniela Rus. Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators. *Soft Robotics*, 1(1):75–87, 2014.

[44] Andrew D. Marchese, Russ Tedrake, and Daniela Rus. Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2528–2535, 2015.

[45] Ramses V. Martinez, Ana C. Glavan, Christoph Keplinger, Alexis I. Oyetibo, and George M. Whitesides. Soft actuators and robots that are resistant to mechanical damage. *Advanced Functional Materials*, 24(20):3003–3010, 2014.

[46] Shane K. Mitchell, Xingrui Wang, Eric Acome, Trent Martin, Khoi Ly, Nicholas Kellaris, Vidyacharan Gopaluni Venkata, and Christoph Keplinger. An easy-to-implement toolkit to create versatile and high-performance HASEL actuators for untethered soft robots. *Advanced Science*, 6(14), 2019.

[47] Bobak Mosadegh, Panagiotis Polygerinos, Christoph Keplinger, Sophia Wennstedt, Robert F. Shepherd, Unmukt Gupta, Jongmin Shim, Katia Bertoldi, Conor J. Walsh, and George M. Whitesides. Pneumatic networks for soft robotics that actuate rapidly. *Advanced Functional Materials*, 24(15):2163–2170, 2014.

[48] L. Mullins. Softening of Rubber by Deformation. *Rubber Chemistry and Technology*, 42(1):339–362, 1969.

[49] Ryuma Niiyama, Daniela Rus, and Sangbae Kim. Pouch Motors: Printable/inflatable soft actuators for robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6332–6337, 2014.

[50] Cagdas D. Onal and Daniela Rus. A modular approach to soft robots. In *IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 1038–1045, 2012.

[51] David E. Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35(2-3):161–176, 2013.

[52] Jifei Ou, Mélina Skouras, Nikolaos Vlavianos, Felix Heibeck, Chin-Yi Cheng, Jannik Peters, and Hiroshi Ishii. AeroMorph - Heat-sealing inflatable shape-change materials for interaction design. In *Symposium on User Interface Software and Technology (UIST)*, pages 121–132, 2016.

[53] Necmiye Ozay, Ufuk Topcu, and Richard M. Murray. Distributed power allocation for vehicle management systems. *IEEE Conference on Decision and Control and European Control Conference*, pages 4841–4848, 2011.

[54] Laura Paez, Gunjan Agarwal, and Jamie Paik. Design and analysis of a soft pneumatic actuator with origami shell reinforcement. *Soft Robotics*, 3(3):109–119, 2016.

[55] Chandana Paul, John William Roberts, Hod Lipson, and Francisco J. Valero Cuevas. Gait production in a tensegrity based robot. In *IEEE International Conference on Advanced Robotics (ICAR)*, pages 216–222, 2005.

[56] Bryan N Peele, Thomas J. Wallin, Huichan Zhao, and Robert F. Shepherd. 3D printing antagonistic systems of artificial muscle using projection stereolithography. *Bioinspiration Biomimetics*, 10(5):055003, 2015.

[57] Janko Petereit, Thomas Emter, Christian W. Frey, Thomas Kopfstedt, and Andreas Beutel. Application of hybrid A* to an autonomous mobile robot for path planning in unstructured outdoor environments. In *7th German Conference on Robotics, ROBOTIK 2012*, pages 227–232, 2012.

[58] Marc H. Raibert, Michael Chepponis, and H. Benjamin Brown. Running on four legs as though they were one. *IEEE Journal on Robotics and Automation*, 2(2):70–82, 1986.

[59] Ludovic Righetti and Auke Jan Ijspeert. Programmable central pattern generators: an application to biped locomotion control. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1585–1590, 2006.

[60] Harpreet Sareen, Udayan Umapathi, Patrick Shin, Yasuaki Kakehi, Jifei Ou,

Pattie Maes, and Hiroshi Ishii. Printflatables: Printing human-scale, functional and dynamic inflatable objects. In *Conference on Human Factors in Computing Systems*, pages 3669–3680, 2017.

[61] Robert F. Shepherd, Filip Ilievski, Wonjae Choi, Stephen A. Morin, Adam A. Stokes, Aaron D. Mazzeo, Xin Chen, Michael Wang, and George M. Whitesides. Multigait soft robot. *Proceedings of the National Academy of Sciences*, 108(51):20400–20403, 2011.

[62] Robert F. Shepherd, Adam A. Stokes, Rui M. D. Nunes, and George M. Whitesides. Soft machines that are resistant to puncture and that self seal. *Advanced Materials*, 25(46):6709–6713, 2013.

[63] Alexander Shkolnik, Michael Levashov, Ian R. Manchester, and Russ Tedrake. Bounding on rough terrain with the LittleDog robot. *International Journal of Robotics Research*, 30(2):192–215, 2011.

[64] Luca Somm, David Hahn, Nitish Kumar, and Stelian Coros. Expanding foam as the material for fabrication, prototyping and experimental assessment of low-cost soft robots with embedded sensing. *IEEE Robotics and Automation Letters*, 4(2):761–768, 2019.

[65] Adam A. Stokes, Robert F. Shepherd, Stephen A. Morin, Filip Ilievski, and George M. Whitesides. A hybrid combining hard and soft robots. *Soft Robotics*, 1(1):70–74, 2014.

[66] Maxime Thieffry, Alexandre Kruszewski, Christian Duriez, and Thierry-Marie Guerra. Control design for soft robots based on reduced order model. *IEEE Robotics and Automation Letters*, 4(1):25 – 32, 2018.

[67] Michael T. Tolley, Robert F. Shepherd, Michael Karpelson, Nicholas W. Bartlett, Kevin C. Galloway, Michael Wehner, Rui Nunes, George M. Whitesides, and Robert J. Wood. An untethered jumping soft robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 561–566, 2014.

[68] Takuya Umedachi, Takeshi Kano, Akio Ishiguro, and Barry A. Trimmer. Gait control in a soft robot by sensing interactions with the environment using self-deformation. *Royal Society Open Science*, 3(12):160766, 2016.

[69] Takuya Umedachi and Barry A. Trimmer. Design of a 3D-printed soft robot with posture and steering control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2874–2879, 2014.

[70] Takuya Umedachi, Vishesh Vikas, and Barry A. Trimmer. Highly deformable 3-D printed soft robot generating inching and crawling locomotions with variable friction legs. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4590–4595, 2013.

[71] Nathan S. Usevitch, Zachary M. Hammond, Mac Schwager, Allison M. Okamura, Elliot W. Hawkes, and Sean Follmer. An untethered isoperimetric soft robot. *Science Robotics*, 5(40):492, 2020.

[72] Vinod K Valsalam, Jonathan Hiller, Robert MacCurdy, Hod Lipson, and Risto Miikkulainen. Constructing controllers for physical multilegged robots using the ENSO neuroevolution approach. *Evolutionary Intelligence*, 5(1):45–56, 2012.

[73] Vishesh Vikas, Piyush Grover, and Barry Trimmer. Model-free control framework for multi-limb soft robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1111–1116, 2015.

[74] Alexander W. Winkler, C. Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

[75] Huan Xu, Ufuk Topcu, and Richard M. Murray. Specification and synthesis of reactive protocols for aircraft electric power distribution. *IEEE Transactions on Control of Network Systems*, 2(2):193–203, 2015.

[76] Meng Yu, Weimin Yang, Yuan Yu, Xiang Cheng, and Zhiwei Jiao. A crawling soft robot driven by pneumatic foldable actuators based on Miura-ori. *Actuators*, 9(2), 2020.

[77] Huichan Zhao, Yan Li, Ahmed Elsamadisi, and Robert Shepherd. Scalable manufacturing of high force wearable soft actuators. *Extreme Mechanics Letters*, 3:89–104, 2015.

[78] Huichan Zhao, Kevin O 'brien, Shuo Li, and Robert F. Shepherd. Optoelectronically innervated soft prosthetic hand via stretchable optical waveguides. *Science Robotics*, 1(1), 2016.

[79] Baxi Zhong, Yasemin Ozkan-Aydin, Guillaume Sartoretti, Jennifer Rieser, Chaohui Gong, Haosen Xing, Howie Choset, and Daniel Goldman. A hierarchical geometric framework to design locomotive gaits for highly articulated robots. *Robotics: Science and Systems*, 2019.