Interactive, Iterative Robot Design

Bradley Canaday[1], Samuel Zapolsky[2] and Evan Drumwright[3]

***Abstract*— Consider how one designs a robot. Starting from a relative size (e.g., nanometer, centimeter, meter), the roboticist picks a morphological type (manipulator, quadruped, biped), and then uses intuition, experience, biological inspiration, or some combination of the three to select kinematic, dynamic, and geometric parameters. The designer then conducts preliminary checks to see whether the robot can satisfy its intended function: manipulation, locomotion, or both. Subject to performance on sample tasks, the designer will iteratively adjust the physical parameters and conduct these checks, presumably until convergence is reached. This paper describes a means to automate part of this approach by combining interactive elements with powerful tools using multi-rigid body simulation.**

We describe and demonstrate a virtual testing phase that explores the space of robot morphological parameters to determine whether the physically situated robot will serve its intended purpose. If the robot is not capable of performing a given target task, the virtual testing phase can determine which of the robot's morphological parameters should be modified. The process keeps a human in the loop to help account for hard to quantify design aspects like appearance, quirks of the fabrication procedure (i.e., laser cutting, milling, 3D printing processes), or even expert knowledge. We therefore intend for the described approach to be used as an interactive tool that gives a robot designer feedback on what morphological parameters are likely to limit the performance of a robot and how to modify the design to fix or offset such limitations. We demonstrate that, through simulated prototyping and testing methods, we can improve a robot design and iteratively locate morphological parameters that make efficient use of available hardware.

I. INTRODUCTION

We wish to optimize a robot's model within simulation to maximize its performance on a task given known hardware constraints. We present an approach that updates a robot's morphological parameters to improve the robot's ability to perform an intended task. We demonstrate that the predicted improvements made in our virtual framework translate to improved task performance during physically situated testing.

One could imagine designing an expert system—or the modern extensions of them, deep neural networks—to optimize robot designs; these paradigms either require, in the case of expert systems (or significantly benefit from) in the case of deep neural networks, significant domain knowledge. Another possibility is using global optimization algorithms, like CMA-ES, to optimize gait and morphological parameters, as done in [8]. This kind of automatic approach can clearly be successful, though removing humans from the loop is not necessarily beneficial: it is challenging to select objective functions that capture secondary criteria and do not result in unintended side effects. Additionally, synthesis of ad hoc solutions have to date proven far superior, at least with respect to robot locomotion, to tabula rasa optimization; see [20], which requires priming and shaping the optimization, and [13], in which an ad hoc robot design and locomotion control system yields high performance.

This paper focuses on using computation to guide robot design, rather than automating such design completely. We envision a robot model editor[1] paired with an analysis tool that identifies the robot design aspects that limit performance. This tool would allow iterative improvement upon an initial design.

For a given task there is a volume of operational space that the robot must be able to access in order to succeed at the task. If a robot is unable to perform that task, then some operational space states defined in the task are inaccessible to the robot due to limiting factors acting on its dynamic and kinematic system (e.g., maximum torque, kinematic reachability limits). We find that the design of a robot morphology can be guided toward a model parameterization that mitigates the effect that these limits have on its task performance. We propose and demonstrate an interactive robot optimization process that informs a designer how to update a model parameterization and steer away from quantifiable continuous limits (i.e. actuator torque and velocity limits) while permitting the designer to modify the suggested update to incorporate expert, domain specific knowledge that is harder to quantify (e.g., fabrication constraints, traction requirements, size an weight limits, or aesthetic concerns).

The approach we present does not seek to optimize an objective function. Instead, this method iteratively translates the region of operational space accessible to the robot to satisfy a prescribed task. Once a task is within the feasible region of the constraints imposed by a robot's provided hardware, the process is complete. This approach thereby allows adjusting the parameterization of a robot's model to achieve a specific task; such morphological modification allows balancing capabilities of a robot's hardware to achieve specific tasks. The described approach also provides an intermediate virtual prototyping and stress-testing stage between design and fabrication stages. We demonstrate the performance of

[1]Bradley Canaday is an undergraduate researcher at the Computer Science Dept., George Washington University, Washington, DC, USA `bradcanaday@gwu.edu`

[2]Samuel Zapolsky is a Ph.D. student at the Computer Science Dept., George Washington University, Washington, DC, USA `samzapo@gwu.edu`

[3]Evan Drumwright is faculty in the Computer Science Dept., George Washington University, Washington, DC, USA `drum@gwu.edu`

[1]such as the *model editor* by Open Source Robotics Foundation as part of their *Gazebo* simulator

the approach by generating robot designs from both "supervised" (human in-the-loop) and fully automated processes and then compare their relative performance with respect to a simple locomotion task. We perform experiments with two fabricated robots that are 3D printed using morphological and mass parameters of the tuned model.

II. Background and Related work

This paper builds on top of work in evolutionary robotics, physical simulation, rapid prototyping, and operational space locomotion. We seek to build upon this work by providing an interactive robot design suite toward improving a robot's morphology.

A. Evolutionary robotics

Evolutionary robotics seeks to implement a morphology exploration strategy of iteratively modifying a robot's model by optimizing over a fitness function [15]. Constraints on robot model limits can be incorporated into the problem in two ways; (1) Only generate feasible offspring so these limits are never violated; (2) Penalize infeasible offspring proportionally to their infeasibility. The latter approach allows for a larger region to explore, possibly avoiding falling into local minima. Some evolutionary design work to create systems in which both the morphology and control scheme of the robot are modified following a genetic algorithm approach to develop robots based on geometric primitives to accomplish simple tasks such as trotting or pushing an object [3], [21].

B. Morphological computation

The virtual creatures generated through a genetic algorithmic approach (see [1]) shed light on a deeper insight that morphological design may have just as much sway on certain aspects of control as modifications tot he control system itself. Changes to a robot's morphology affect its physical behavior, but do not accrue additional computational cost, they are computationally relevant for control and stability of the locomoting system [18].

C. Situated robotics & embodied cognition

The interaction of a robot's morphology with its environment can have such a great impact on robot behavior, that these interactions must be considered developing a robotic system [5], [9]. This importance of working with *situated* systems is emphasized by work with *embodied cognition* which seek to understand the interaction of cognition with the sensory information from a physical environment. The necessity of considering situated performance becomes especially important with locomoting systems that continuously make and break contact with their environments and their behavior is heavily determined by how those interactions affect its dynamics. We seek to *embody* our locomoting robotic system in a physics simulation. Though sensors are poorly modeled in such system, physical interaction with the environment will still lead to behavioral differences between morphologies that can be used to improve a well-modeled robot [6].

D. Prototyping

Rapid prototyping of 3D-printable robots has found some success in previous work [14] with robots having a variety of morphologies and leg counts being successfully built and tested using the same base software to determine footfall pattern. A base robot was provided for users to edit to their liking in terms of shape and footfall patterns, and then transformed from the simple inputs into usable robots and gaits by the program from the user inputs.

E. Rigid body & nonsmooth mechanics

This work considers robot dynamics that are well modeled by rigid bodies and rigid or nearly rigid contact. The approach is not necessarily predicated on these assumptions. Deformable body simulations albeit much slower will provide a more thorough appraisal of a robotic system.

In addition to the challenges of analyzing nonlinear dynamics (of multi-rigid body systems), the approach discussed in this paper requires consideration of *nonsmooth mechanical systems* [4], for which velocities can change discontinuously due to impacts and even non-impacting contact with Coulomb friction [22].

Multi body dynamics with rigid contact and Coulomb friction—which captures important stick-slip transitions—can be modeled as a differential algebraic equation (DAE):

$$\ddot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \tag{1}$$

$$\boldsymbol{0} = \Phi(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \tag{2}$$

where $\Phi(.)$ is a set of algebraic constraints. Some constraints are always active, like bilateral joint constraints. Other constraints are only active if certain conditions are met; e.g., a contact constraint between two polyhedra would only be active when the bodies are in contact at that point and they would otherwise (i.e., without the constraint in place) interpenetrate at that point. Other algebraic constraints acting on a robot system may include motor torque and velocity limits, which constrain how much torque can be applied to the system through actuation. These kind of problems can be formalized as a differential complementarity problem [16]:

$$\dot{\Phi}_i(.) = 0 \text{ if } \Phi_i(.) = 0 \text{ and } \lambda_i > 0, \text{ for } i = 1, \ldots, m \tag{3}$$

where λ_i acts as a Lagrange Multiplier (i.e., it is zero if the constraint is active and non-negative otherwise).

F. Locomotion

This work considers improving morphological designs of a robot independently from the control system. This implies that the operational space state and external forces remain fixed while the morphological and dynamic design are improved to consider interaction of the controlled mechanical system with its environment.

Operational space control for locomoting systems allows the robot to decouple the configuration of its morphology from its end effector and base configurations [2]—assuming all operational space goals are kinematically reachable. We implement such a locomotion controller [23], which borrows lessons in foot placement and simple trotting behavior from [11], and has similar parameterizations to other gait planning systems for quadrupedal robots [7], [12]. We assume for this work that a planned gait is open-loop stable. This assumption simplifies the presentation of our approach without loss of generality.

Though this transition to operational space may lead to computationally difficult to handle redundancies in configuration space, these can be dealt with through careful use of inverse kinematics [17], [19]. The system we focus upon is not redundant, but these considerations are important for complex morphological modifications (e.g., adding joints).

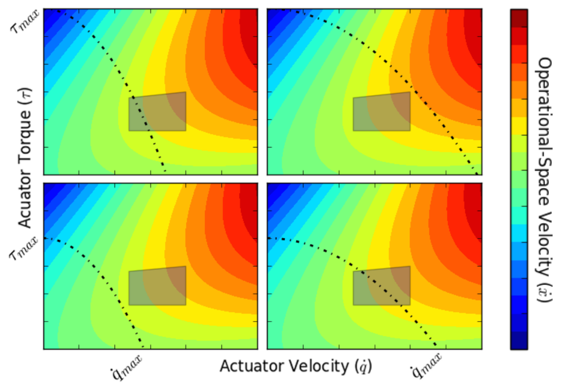
III. Approach

Our idea relies upon adjusting a robot's morphological and dynamic properties using inverse kinematics and inverse dynamics to map operational space velocities and forces to generalized velocities and forces. The remainder of this section introduces the concepts of the accessible operational space (§III-A) and limit functions (§III-B), outlines the iterative process (§III-C), and describes how model parameters are adjusted to alter limit function outputs (§III-D).

A. Accessible operational space

Improving a robot's actuation capabilities increases the maximum operational space velocities and forces that the robot can generate. If actuation is improved enough, then the robot might be able to perform all the motion requirements of an intended task. Kinematic reachability limitations imposed by the robot's morphology will remain unchanged as a result of actuator modification. Figure 1 shows an illustrative example of how actuators can increase the magnitude of operational space velocities that the robot can achieve.

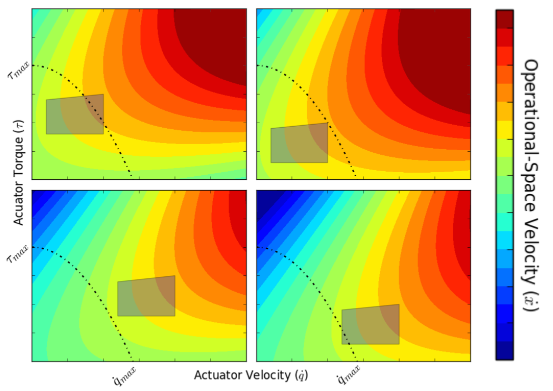
Figure 1: Actuator Modification



The requirements of a target task (shaded box) are sometimes outside the capabilities of the robot, bounded in this plot by the torque-speed curve (under the dotted line). the actuators of a robot can be modified to increase the maximum operational space velocity that the robot can achieve
Bottom Left: A robot whose actuators are not suitable to perform the target task will not fit the requirements of the task within it's capabilities (under the dotted line).
Top Left: Decrease motor gearing or increase maximum voltage of the actuator to increase the maximum torque.
Bottom Right: Increase gearing or improve efficiency of the actuator to achieve a higher max actuator velocity.
Top Right: Improving both actuator efficiency and max torque results in the robot being able to perform all the operational space motions required by the task (top right).

Beyond actuator improvement (expanding the boundary of the accessible volume of operational space), the model of a robot can be modified to a similar effect (shifting and deforming the accessible volume of operational space). Model modifications such as adjusting the size of the base link, decreasing leg lengths, and decreasing stride length will transform the mapping between configuration and operational space for the model. Figure 1 provides an illustrative example of how a robot's model can be modified to transform the accessible volume of operational space.

Figure 2: Morphological Parameter Modification



The requirements of a target task (shaded box) are sometimes outside the capabilities of the robot, bounded in this plot by the torque-speed curve (under the dotted line). the robot's morphological parameterization can be modified to change which operational space motions, positions, and forces it can access with the generalized velocity and forces it is capable of generating. If one carefully updates the robot's morphological parameters, the robot might become capable of performing the target task even with a fixed torque-speed curve.
Bottom Left: A robot whose actuators or morphology are not capable of performing the task will not be able to perform the motions required by the task (bottom left).
Top Left: Increasing leg length, foot radius, or foot friction will decrease the actuator speed requirements of a task by providing the robot a larger lever or greater reaction force when interacting with the environment.
Bottom Right: Decreasing leg length, foot radius, or foot friction will decrease the actuator torque requirements of a task by providing the robot a smaller lever or less reaction force when interacting with the environment.
Top Right: Improving robot morphology in a directed manner will allow the robot to make the most of its actuator limitations, allowing it to perform the motions required by the task (top right).

Of the robot modifications mentioned in Figures 1 and 2, we focus on implementing the latter. We modify modeling parameters of the robot morphology while assuming fixed actuation constraints (i.e., torque and speed limits).

Some limiting factors are hardware related and some are control related. All factors are interrelated and the interdependencies are not always clear. *There does not yet exist a straightforward way to predict the interaction of physical design and control system parameters for a particular robot performing a given task.*

Other limitations that are less apparent than the torque-speed trade-off may be equally important when gauging the capabilities of a robot. For example, a robot must avoid front-back foot collisions at high locomotion speeds when performing asymmetric gaits about the sagittal-plane (e.g., during walking and trotting). Additionally, certain robot designs may have a kinematic limitation for how high or far they can reach, preventing them from performing certain pick-and-place tasks.

B. *Task-robot limit functions*

We find that by observing how a *limit function* evolves in relation to modeling parameters, we can steer the design of a robot morphology toward a model that avoids violating constraints while respecting task constraints by modifying the model parameterization to steer away from quantifiable limits. If a specific constraint becomes "active" during the robot's virtual performance of the task, the process is halted and the model is updated to avoid violating a constraint, or reduce constraint violation:

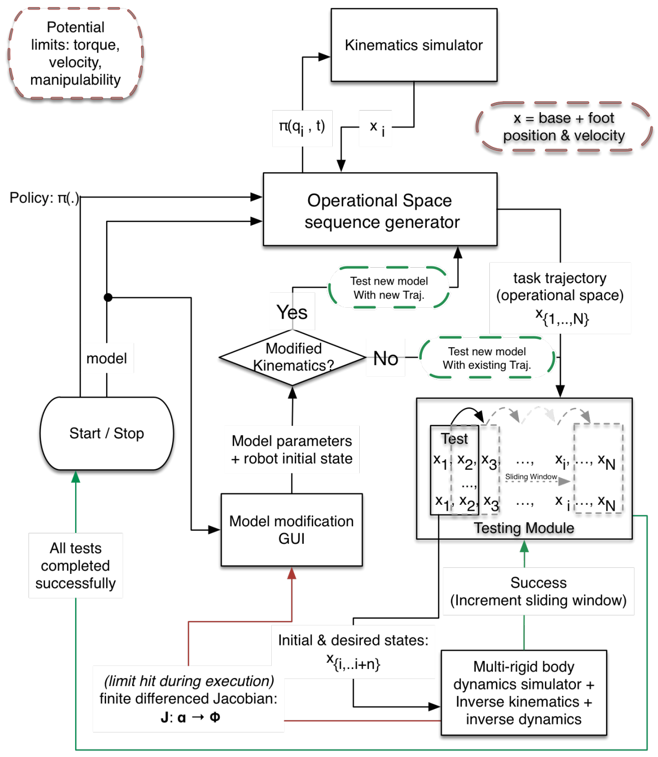
$$\text{if} \quad (\Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) < \epsilon)$$
$$\Phi_{\boldsymbol{p}_i+1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) > \Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \tag{4}$$
$$\text{else}$$
$$\Phi_{\boldsymbol{p}_i+1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \geq 0 \tag{5}$$

Subscript $\boldsymbol{p}_i$ refers to the parameterization of the model at iteration i. The limit function Φ produces a vector of unitless "distances", representing how close each constraint is to being violated. Every component should be non-negative. Negative values represent how far over a limit boundary the system has passed, while positive values represent distance from the limit boundary. For example, if an actuator torque limit violation ($u_i > u_{i,max}$) causes the system to halt, the morphological and dynamic properties of the robot can be updated to produce a lower magnitude u_i toward realizing the operational space trajectory (by, e.g., reducing leg length, decreasing foot friction, or reducing the weight parameterization for child links of the joint).

C. *Process iteration*

Because constraint violation might be rather large immediately upon executing a difficult task with a novel robot design, the process should be bootstrapped by performing an "easy" version of the task. For a pick-and-place task, this may entail starting with a lightweight object and slower movements; simplifications are straight forward for locomotion where lower velocity gaits are typically easier to perform.

Once the robot can successfully complete a simplified version of the task, the user should repeat the process on an incrementally harder version of the task. At each successive model update, the configuration space trajectory is updated to achieve the same operational space movement (a task constraint). While the operational space trajectories of a task (e.g., for the torso, foot, and hand) remain fixed throughout the optimization process, the inverse kinematics solution for each robot changes as the morphology evolves. Our software tool physically simulates the robot using inverse kinematics and inverse dynamics tools with contact force prediction to determine the instantaneous actuator forces needed to follow the desired trajectory while the robot contacts the environment. See Figure 3 for an illustration of this algorithm and its sub-processes.



Potential limits: torque, velocity, manipulability
Kinematics simulator
π(q_i , t)
x_i
x = base + foot position & velocity
Policy: π(.)
Operational Space sequence generator
Test new model With new Traj.
Yes
task trajectory (operational space)
x_{1,..,N}
model
Modified Kinematics?
No
Test new model With existing Traj.
Start / Stop
Model parameters + robot initial state
Test
x_1, x_2, x_3 ..., x_i, ..., x_N
Sliding Window
x_1, x_2, x_3 ..., x_i ..., x_N
Testing Module
Model modification GUI
All tests completed successfully
Success (Increment sliding window)
Initial & desired states: x_{i,..i+n}
(limit hit during execution) finite differenced Jacobian: J: α → Φ
Multi-rigid body dynamics simulator + Inverse kinematics + inverse dynamics

Fig. 3. The user will likely want to generate multiple pose sequences and use them for testing. These sequences can be added to as the robot starts to pass tests (after trotting is successful, we can move to running)

D. *Adjusting model parameterization to alter limit function outputs*

Each model update of the optimization process requires constructing a gradient for how each model parameter affects each active limit function at the moment in the task where a limit function first becomes active. Gradient vector $\nabla_j \Phi_{\mathtt{model}(i)}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$ is computed using finite differencing (as described in Figure 4) with respect to modeling parameter j of m total parameters.

The limit gradients are concatenated to form a limit Jacobian, $\mathbf{J}_\Phi$ that describes the slope of the constraint-model space at a specific point in configuration space $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ under the current model parameterization $(\boldsymbol{p}_i)$.

$$\mathbf{J} = \left[\nabla_0 \Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \cdots \nabla_m \Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})\right]$$

The model parameters to be included in the parameter vector $(\boldsymbol{p}_i)$ must be chosen by the designer. This parametrization must be expressive enough to generate a wide range of morphologies but concise enough to preclude any easily rejected morphologies and keep the finite differencing process fast. The gradient descent direction $\mathbf{J}^+\boldsymbol{l}$ where $^+$ is the pseudo inverse operator, an update direction to the robot modeling parameters that should decrease the limit function violations $(\boldsymbol{l} = \Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}))$. The current modeling parameters $(\boldsymbol{p}_i)$ can then be updated to its next iteration $\boldsymbol{p}_{i+1}$:

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i - \boldsymbol{\alpha}\mathbf{J}^+\boldsymbol{l}$$

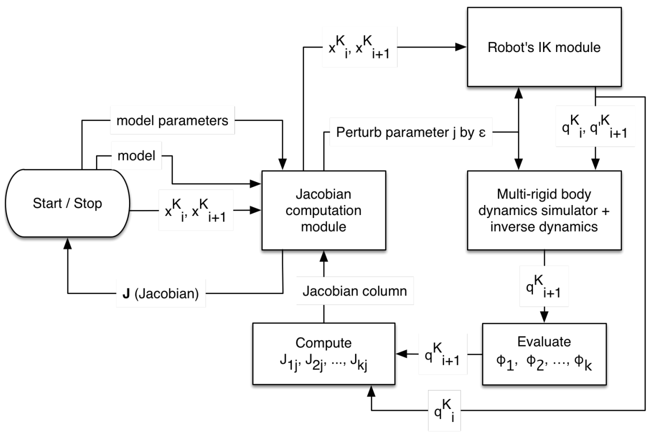


Fig. 4. Jacobian generation flowchart.

Where matrix $\boldsymbol{\alpha}$ is a PD matrix. In our process, alpha is tuned manually. The designer incorporates expert knowledge into this tuning stage, weighting the gradient update and perhaps making additional modifications to the robot model.

IV. Experiments

We perform a test of our optimization framework and then an experiment assessing the validity of our simulation results against 3D printed robots. Our method focuses on, but is not limited to, locomoting in a straight-line across a planar environment. We optimize the morphological parameters of a quadrupedal robot to perform a trot at a variable forward velocity. This experiment tries to adjust the continuous geometric and mass parameters of the links of the quadruped in order to achieve the highest speed possible given known actuator stall torque and velocity limits. We will use the described system to iteratively update the robot model to increase its maximum trotting speed. We focus on trotting because it is a highly dynamic, known challenging task that requires a robot to make full use of its physical ability.

This section assesses the performance of our design system and its final product in a virtual environment. The final robot will be derived from our working *Links* quadrupedal robot (depicted in Figure 9). After verification in simulation, we transfer our changes to the physical robot and observe whether the improvements made in simulation also result in better in situ performance.

A. Software and simulation setup

Pacer [23] runs alongside the open source simulator *Moby*[2], which was used to simulate the legged locomotion scenarios used in the experiments. *Moby* simulates contact with a rigid contact model. *Moby* provides accurate time of contact calculation for impacting bodies, yielding more accurate contact point and normal information. Other simulators typically step past the moment of contact and approximate contact information based on the intersecting geometries. The accurate contact information provided by *Moby* allows us to test the robot under more realistic simulated conditions.

[2]Obtained from `https://github.com/PositronicsLab/Moby`

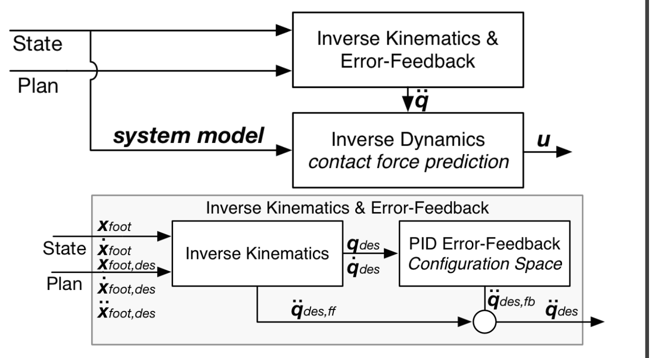
Parameter	**Value**
relative-torso-pose	$\{0,0,0,0,0,0\}$
step-height	0.01
gait-duration	0.3
length	2
height	0.75
width	$\{1.1, 1.1\}$
gait	$\{0.25, 0.75, 0.25, 0.75\}$
duty-factor	$\{0.6, 0.6, 0.6, 0.6\}$

Fig. 5. Gait parameterization used in the morphological optimization process.

B. Tasks

We simulate the quadruped locomoting with an initial gait parameterization shown in Figure 15. These values were chosen as a generally functional set of parameters for a 16 cm tall quadruped. The robot uses a closed loop error-feedback controller which accumulates feedback accelerations which are input to a QP-based inverse dynamics controller with contact force prediction [24].

Figure 6: Controller



Inverse dynamics controller with predictive contact forces generates an estimate of contact forces at the current time given contact state and internal forces [24]. Stabilization and error-feedback are accumulated as accelerations and input into inverse dynamics controller.

C. Morphological parameterization

One simplification we made for our application assumes that a locomoting robot will need to be symmetric across the sagittal plane. Because we are likely to reject any robots that are laterally asymmetric, we only adjust the modeling parameterization of half of the robot and reflect the parameters to the other side when the model is updated.

a) Complexity of topological updates: Adding joints or links to the robot's topological structure introduces complexities in planning and large discontinuous jumps in our algebraic limit functions. Though we are considering topological updates as a target for future work, we currently consider only a fixed topology (one base with four, three-jointed limbs) and adjust the geometric, inertial, and actuation parameters of that model.

Given a fixed topological structure, there are two types of modeling parameters that we consider. In this work, we update only continuous modeling parameters (cylindrical limb, box-shaped base, and spherical foot dimensions as well as link mass). Additionally, we might consider discrete modeling parameters (e.g., integer gear ratios or fabrication materials implying discrete link densities). Relevant morphological and actuator limitation for our particular use-case, a quadrupedal robot, are shown in Figures 7 and 8, respectively.

Robot Limitations

Parameter	DoF	Value
Actuator Torque Limits	$N_{\texttt{joints}}$	1.1 N·m
Actuator Velocity Limits	$N_{\texttt{joints}}$	6.55 rad/s

Fig. 8. Limits to robotic hardware considered in the experiment. These are the reported stall torque and max velocity of a cheap hobby servo.

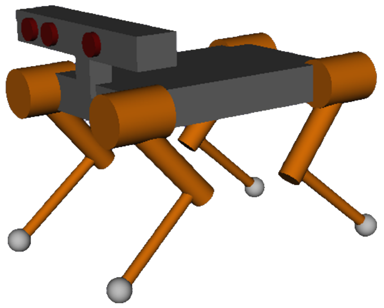


Fig. 9. The base link (grey) has a box geometry; limbs, originating from the corners of the base have a cylindrical geometry; feet have a spherical geometry. All links also have a density that, with the calculated volume, determine the mass of the link. A Kinect type sensor (unused) is rendered also.

D. Experimental design

To determine the usefulness of the proposed interactive design process, we compare three candidate robot models: (1) an **`initial`** robot, taken without modification from previous, successful locomotion experiments [24], [10]; (2) an updated robot design created by following an **`automated`** approach, seeded from the `initial` robot and allowing the design software to iteratively update the model according to the path of steepest descent until it can make no further progress; an (3) an updated robot design created by following a **`supervised`** approach, seeded from the `initial` robot using the software to guide the designer through modifications using the direction of steepest descent. After generating the three candidate models, we fabricated the `initial` and `supervised` robot models for testing in situ. We did not attempt to fabricate the `automated` model, as we did not limit it to our fabrication constraints; we comment on this model in Section V.

We tested both fabricated robots by comparing their average velocity over a duration of trotting against the intended trotting speed. We used this metric to determine whether in situ performance improved at each target gait velocity for the `supervised` robot over the `initial` robot. We expected that the `supervised` robot would be much more successful at performing the higher speed gaits, as it was modified to respect its actuator limitations for the prescribed gaits.

a) Initial robot: The optimized robot designs (`automated` and `supervised`) were seeded from a simple robot model (`initial`). A CAD model of the `initial` morphology designed for a 3D printer is depicted in Figure 10.

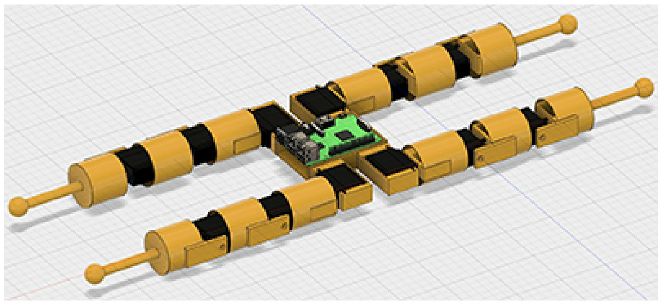


Fig. 10. The `initial` (non-optimized) robot design used to seed the optimized models.

V. Results

This section presents in situ results from the morphological optimization process. Both `supervised` and `automated` design processes began using the `initial` model attempting to perform a trotting gait (see Figure 15) at the starting forward velocity of 10 cm/s. It was necessary for the model to improve to proceed to the next velocity (velocities were increased in 10 cm/s increments).

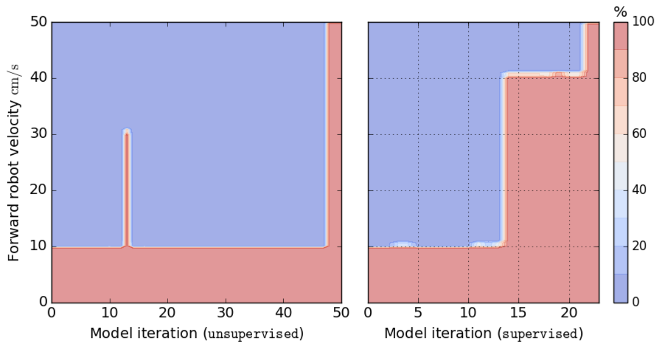


Fig. 11. The `automated` (left) and `supervised` (right) robot design progress over several model updates. Regions of the plot are colored according to the percentage of the gait that robot model can complete at the specified velocity before violating an actuator limit. Colors blue, white and red coincide with 0, 50 and 100 percent task completion, respectively.

This section describes the two robots generated through the `supervised` (this work) and `automated` (gradient descent) optimization processes. Figure 11, shows that modifying the robot enough to progress past one limit violation is usually enough to progress through the remainder of the planned gait, at least for a walking trot. This phenomenon expresses itself in the plots as either 0% or 100% progress in the plot (blue or red, respectively), and there are very few examples of 50% progress (white). *Both models achieve the same theoretical maximum forward trotting velocity of 50 cm/s.*

Free Variables in Hardware Design

Parameter	Parameters	Description
cylindrical link dims (cm)	$\{\text{length, radius}\}\times N_{\text{limb links}}$	Limb link (one parent, one child) collision and inertial geometry
box link dims (cm)	$\{\text{length, width, height}\}\times 1$	Base link (no parent, multiple children) collision and inertial geometry (determined point where limbs branch from robot (at each of the bottom corers of the base))
sphere link dims (cm)	$\{\text{radius}\}\times N_{\text{foot links}}$	Foot link (one parent, no children) collision and inertial geometry
link mass (g)	N_{links}	Mass inertial parameter of each link

Fig. 7. Robot design parameters.

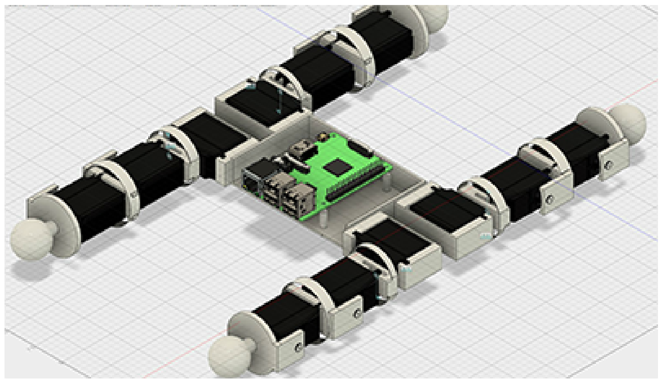


Fig. 12. The supervised optimized robot design using a design process with gradient descent directions suggested to the designer.

a) Optimized robot (supervised): Our human-in-the-loop modification process permitted us to make informed updates to the robot model. A CAD model of this morphology designed for a 3d printer is depicted in Figure 12. We were given the gradient descent direction during this process but allowed to make arbitrary updates to the model parameters. As already noted, the resulting model (supervised) can theoretically achieve a forward trotting velocity of 50 cm/s (see Figure 11).

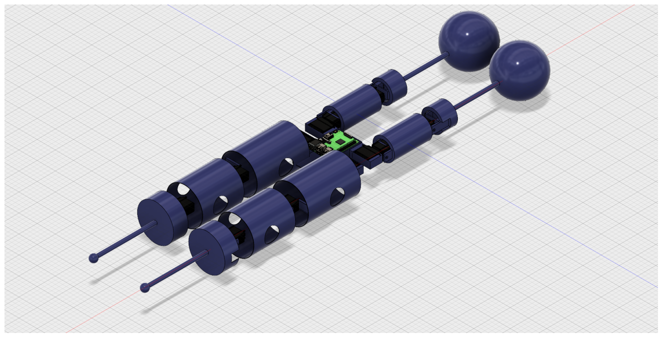


Fig. 13. The automated optimized robot design using gradient descent method. This design produces link lengths that are too short to fit our actuators and link radii that are too wide to achieve a reasonable range of motion.

b) Optimized robot (automated): The automated optimization process following a purely gradient decent approach produced a model (automated) that can theoretically achieve the same maximum speed as the supervised model. A CAD model of this morphology designed for a 3D printer is depicted in Figure 13. The model produced through this approach created a robot characterized by extreme values (e.g., 1 g front foot mass and 187 mm hind foot radius). Although fabrication of this robot is possible, hard to quantify issues—such as self collision and thin, brittle link geometries—would have resulted in the robot destroying itself during the first test. In Section V-A, we mention that the initial model broke in the fifth trial during physically situated testing. The 3D printed automated robot would have undoubtedly broken as well.

Parameter	initial	automated	supervised
$l_{\text{Front,Hip}}$	62	137	54
$l_{\text{Front,ULeg}}$	72	96	53
$l_{\text{Front,LLeg}}$	140	189	55
$l_{\text{Hind,Hip}}$	62	1	54
$l_{\text{Hind,ULeg}}$	72	81	53
$l_{\text{Hind,LLeg}}$	140	106	55
l_{Base}	87	165	72
w_{Base}	64	72	64
h_{Base}	20	34	25
$r_{\text{Front,\{Hip,ULeg,LLeg\}}}$	23	92	23
$r_{\text{Hind,\{Hip,ULeg,LLeg\}}}$	23	37	23
$r_{\text{Front,Foot}}$	20	8	27
$r_{\text{Hind,Foot}}$	20	187	39
$m_{\text{Front,Hip}}$	17	148	12
$m_{\text{Front,ULeg}}$	18	3	12
$m_{\text{Front,\{LLeg,Foot\}}}$	29	1	15
$m_{\text{Front,Hip}}$	17	200	12
$m_{\text{Front,ULeg}}$	18	166	12
$m_{\text{Hind,\{LLeg,Foot\}}}$	29	69	15

Fig. 14. Model parametrization for each robot. Masses (m) are in grams an length (l), width (w), radius (r) and height (h) are in mm.

A. Robot performance in situ

This section presents situated test results for 3D printed robots conforming to the morphological and dynamics parameterizations of the initial and supervised models.

Planning and control for the 3D printed robots were performed by a Raspberry Pi computer looping over a prerecorded configuration space trajectory for the specific robot morphology. The trotting gait trajectory is generated using the *Pacer* planning and control software [23]. Gait parameters are defined in Figure 15.

The commanded forward trotting velocity of 16.26 cm/s was determined by the approximately 60 Hz publish rate of the Raspberry Pi computer. Any faster movement would produce jumpy behavior when the robot tried to follow the joint trajectory in real-time.

Results: The initial robot trotted an average distance of 64 cm over 20 gait cycles, or about 12.22 seconds of trotting during the trials where it managed to complete the task. During the five trials that were performed only three

Parameter	**Value**
relative-torso-pose	$\{0, 0, 0, 0, 0, 0\}$
step-height	0.01
gait-duration	0.608
length	2
width	$\{1.1, 1.1\}$
height	0.75
gait	$\{0.25, 0.75, 0.25, 0.75\}$
duty-factor	$\{0.6, 0.6, 0.6, 0.6\}$
forward velocity	0.1626

Fig. 15. Gait parameterization for in situ trials.

successfully completed the 20 gait cycles. After the first three the next fell to its side and the following trial irreparably damaged the robot, stripping the plastic servo horn. Excessive torque applied from the long legged morphology was likely the cause of this hardware failure. The robot had trouble supporting its own weight during most trials. The average velocity during the successful trials was 5.23 cm/s, or about one-third of the commanded velocity. Much of this speed discrepancy can be attributed to the robot's inability to produce the torque necessary to properly move its long limbs.

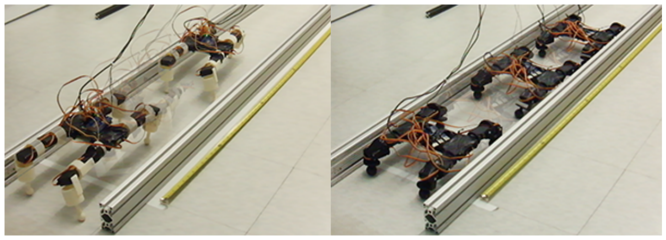


Fig. 16. A time-lapse of the `initial` (left) and `optimized` (right) robot trotting for 20 gait cycles. Videos of all 20 situated trials are provided with the supplemental material.

The `supervised` robot trotted at an average distance of 99 cm over 20 gait cycles, or about 12.22 seconds of trotting. The average velocity during the trials was 8.04 cm/s, about one-half the commanded velocity (and 54% faster than the `initial` robot). Much of this robot's discrepancy in trotting speed to the desired value can be attributed to the robot's feet tending to slide during the stance phase portion of the gait. The robot remained stable and did not have trouble supporting its own weight during the experiment and successfully finished all trials.

VI. Discussion

We described a morphological modification process that allowed us to significantly optimize real-world performance. We observed a substantial performance increase between morphologies within our situated trails, reflecting the results from the simulated optimization process. We expect that further development of such tools will allow hobbyists and roboticists to maximize robot performance.

Acknowledgements

This work was supported by ARO grant W911NF-16-1-0118 and by a GWU SEAS SUPER fellowship to Bradley Canaday.

References

[1] J. E. Auerbach and J. C. Bongard. On the relationship between environmental and mechanical complexity in evolved robots. In *Intl. Conf. on the Synthesis and Simulation of Living Systems (ALife)*, volume 13, pages 309–316, 2012.

[2] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. de Pieri, and D. G. Caldwell. A reactive controller framework for quadrupedal locomotion on challenging terrain. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*, Karlsruhe, Germany, 2013.

[3] J. C. Bongard. Why morphology matters. *The Horizons of Evolutionary Robotics*, pages 125–152, 2014.

[4] B. Brogliato. *Nonsmooth Impact Mechanics: Models, Dynamics, and Control*. Springer-Verlag, London, 1996.

[5] A. Cangelosi, J. Bongard, M. Fischer, and S. Nolfi. *Handbook of Computational Intelligence*, chapter Embodied intelligence, pages 697–714. Springer, 2015.

[6] N. Cheney, J. Bongard, V. SunSpiral, and H. Lipson. On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Intl. Conf. on the Synthesis and Simulation of Living Systems (ALife)*, volume 15, Cancun, Mexico, 2016.

[7] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne. Locomotion skills for simulated quadrupeds. In *Proc. ACM SIGGRAPH*, 2011.

[8] K. M. Digumarti, C. Gehring, S. Coros, J. Hwangbo, and R. Siegwart. Concurrent optimization of mechanical design and locomotion control of a legged robot. In *Proc. Intl. Conf. Climbing Walking Robots (CLAWAR)*, 2014.

[9] M. Dorigo and M. Colombetti. Robot shaping: Developing situated agents through learning. *Artificial Intelligence*, 70(2):321–370, 1994.

[10] E. Drumwright. Rapidly computable viscous friction and no-slip rigid contact models. *arXiv*, 2015.

[11] B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut. Omnidirectional locomotion for quadupred robots. In *Proc. RoboCup 2001: Robot Soccer World Cup V*, pages 368–373, 2002.

[12] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal. Learning, planning, and control for quadruped locomotion over challenging terrain. *Intl. J. Robot. Res.*, 30(2):236–258, 2011.

[13] J. Lee, D. J. Hyun, J. Ahn, S. Kim, and N. Hogan. On the dynamics of a quadruped robot model with impedance control: Self-stabilizing high speed trot-running and period-doubling bifurcations. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*, Chicago, 2014.

[14] V. Megaro, B. Thomaszewski, M. Nitti, O. Hilliges, M. Gross, and S. Coros. Interactive design of 3d-printable robotic creatures. In *Special Interest Group on Computer GRAPHics and Interactive Techniques (SIGGRAPH)*, 2015.

[15] A. L. Nelson, G. J. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345—370, April 2009.

[16] J.-S. Pang and D. E. Stewart. Differential variational inequalities. *Math. Program., Ser. A*, 113:345–424, 2008.

[17] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal. A unifying methodology for robot control with redundant DOFs. *Autonomous Robots*, 24(1–12), 2008.

[18] R. Pfeifer and F. Iida. *Creating Brain-Like Intelligence*, volume 5436 of *Lecture Notes in Computer Science*, chapter Morphological Computation: Connecting Body, Brain, and Environmentand environment, pages 66–83. Springer, 2009.

[19] B. Satzinger, J. Reid, M. Bajracharya, P. Hebert, and K. Byl. More solutions means more problems: Resolving kinematic redundancy in robot locomotion on complex terrain. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*, Chicago, 2014.

[20] G. Schultz and K. Mombaur. Modeling and optimal control of running. *IEEE/ASME Trans. on Mechatronics*, 15(5), 2010.

[21] K. Sims. Evolving virtual creatures. In *Special Interest Group on Computer GRAPHics and Interactive Techniques (SIGGRAPH)*, pages 15–22, 1994.

[22] D. E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, Mar 2000.

[23] S. Zapolsky. Pacer. https://github.com/PositronicsLab/Pacer.

[24] S. Zapolsky and E. Drumwright. Quadratic programming-based inverse dynamics control for legged robots with sticking and slipping frictional contacts. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*, 2014.