Interactive, Iterative Robot Design

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

Abstract— **Consider how a new robot is designed. 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. The designer then iteratively adjusts the physical parameters and conducts checks using sample tasks, presumably until convergence is reached. This paper describes a means to automate part of this approach by combining interactive elements with powerful tools that use multi-rigid body simulation.**

We describe and demonstrate a virtual testing phase that first determines whether the physically situated robot would serve its intended purpose. If the robot is not capable of performing its target task, the virtual testing phase can determine which of the robot's morphological parameters should be modified in order to do so. 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 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

Efficiently designing robots that interact physically with their environments through contact, like robots that walk and manipulate, is an open problem. Current physical design procedures of such robots requires considerable expertise and is taught in few academic programs. This work aims to improve this *status quo* using computed-aided engineering.

Our suppositions are that the robot's control policy is given (we use *Pacer* [21], a generic legged locomotion gait planning and control system, for that purpose in this work) and that the robot designer starts from a morphological template (i.e., a kinematic, dynamic, and geometric "sketch"). This latter assumption is not a particularly stringent one, as roboticists copy each others' designs (the first author borrowed from an existing working design by the second author in this paper) and borrow others from nature.

We wish to optimize a robot's model within simulation to maximize its performance on a task[1] 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.

Toward the purpose of computer aided robot design, 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 greatly benefit from (in the case of deep neural networks) significant domain knowledge. Another possibility is using non-convex optimization algorithms to optimize gait and morphological parameters, as done in [6]. This kind of automatic approach can clearly be successful, though removing humans from the design process 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 [18], which requires priming and shaping the optimization, and [11], in which an ad hoc robot design and locomotion control system yields high performance.

We began this research from the following observation. For a given task there is a volume of *operational space* (which for our purposes include poses in 3D, wrenches, and twists) that the robot must access to perform that 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 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 set of model parameters that mitigates the effect that these limits have on the robot's task performance. We propose and demonstrate an interactive robot optimization process that informs a designer how to update a models parameters and steer away from quantifiable constraints (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 and weight limits, or aesthetic concerns).

The approach we present does not seek to optimize an ob-

[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]In the context of this work, we informally define a *task* as requiring the robot to effect a desired state (or one of a set of desired states) for both itself and its environment; this definition clearly lacks consideration of time and path dependence, among other factors.

jective function. Instead, this method iteratively translates the region of operational space accessible to the robot to satisfy a prescribed task. Once a task lies within the feasible region of the constraints imposed by a robot's provided hardware, the process is complete. This approach thereby allows adjusting a robot's modeling parameters to achieve a specific task. We demonstrate the performance of 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 and match the morphological and mass parameters of the resulting designs.

II. Background and Related work

This paper builds upon work in evolutionary robotics, physical simulation, rapid prototyping, and operational space control 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 [13]. 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], [19].

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 to the control system itself. A robot's morphology affect the stability and performance of its locomotion system [16].

C. Prototyping

Rapid prototyping of 3D-printable robots has found some success in previous work [12] 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.

D. 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 [20].

E. 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 [21], which borrows lessons in foot placement and simple trotting behavior from [8], and admits a similar parameterization to other gait planning systems for quadrupedal robots [5], [9].

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 [15], [17]. The quadrupedal robots we focus on throughout this paper do not have redundant actuation, but these considerations are important for many walking robots.

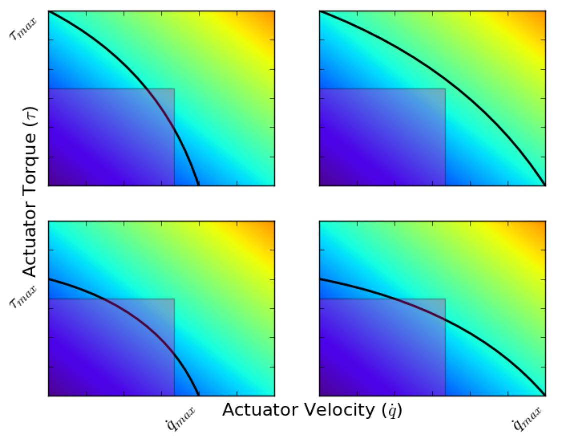
III. Approach

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).

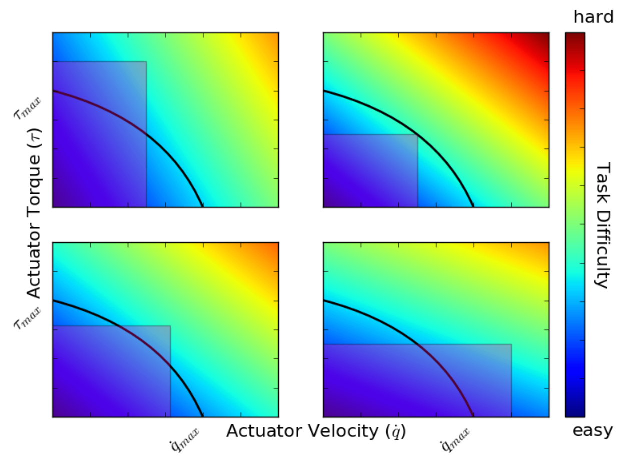
A. Accessible operational space

Improving a robot's actuation capabilities increases the maximum operational space velocities and forces that the robot can generate. If actuation is improved sufficiently, then the robot might be able to perform all of 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 1a shows an illustrative example of how actuators can increase the magnitude of operational space velocities that the robot can achieve.

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



(a) ***Actuator Modification***
Bottom Left: A robot whose actuators are not suitable to perform the target task will not fit the requirements of the task within its capabilities (under the torque-speed curve).
Top Left: Increase the maximum torque $\tau_{\max}$.
Bottom Right: Increase the maximum actuator velocity $\dot{q}_{\max}$.
Top Right: Improving the maximum torque and maximum velocity results in the robot being able to perform all required torques at each desired actuator velocity of the task.



(b) ***Morphological Parameter Modification***
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.
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.

Fig. 1: *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 curved line). Actuators (a) or morphological parameters (b) can be modified to increase the capabilities of the robot to fit a given task. If we carefully update the robot's morphological parameters, the robot might become capable of performing a target task, even with a fixed torque-speed curve.*

link, decreasing leg lengths, and decreasing stride length will transform the mapping between configuration and operational space for the model. Figure 1a provides an illustrative example of how a robot's model can be modified to transform the accessible volume of operational space.

Of the robot modifications mentioned in Figures 1a and 1b, 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).

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). Similarly, mechanical designs may impose limits on kinematic reachability, possibly precluding 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 parameter setting that avoids violating constraints without compromising task performance by modifying the model parameters to steer away from quantifiable constraints (also referred to as "limits", hereafter). 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 to reduce constraint violation.

We define a vector-valued limit function $\boldsymbol{\Phi}_{\boldsymbol{p}}$: $\mathbb{R}^{n_q \times n_v \times n_u} \rightarrow \mathbb{R}$, where $\boldsymbol{p}$ refers to the model parameters, n_q is the dimension of robot generalized coordinates, n_v is the dimension of robot generalized velocities, and n_u is the dimension of the *controllable* generalized forces) produces a m-dimensional vector of unit-less "distances" represent how close each constraint is to being exactly satisfied. The limit functions depend on the robot's state $\{\boldsymbol{q}, \dot{\mathbb{q}}\}$ and input $\boldsymbol{u}$ and specify inequality constraints:

$$\boldsymbol{\phi}_{\boldsymbol{p}}(\boldsymbol{q}, \dot{\mathbb{q}}) \geq \mathbf{0} \tag{1}$$

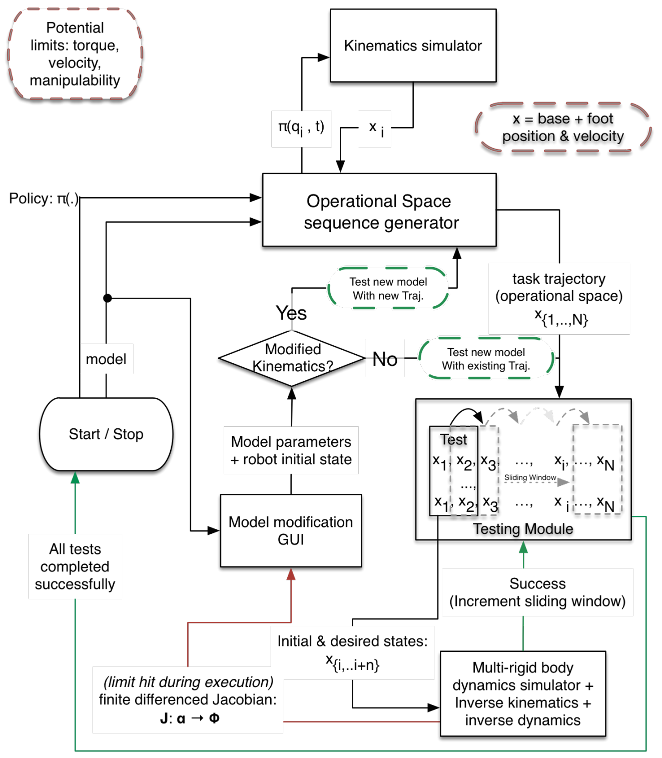
Note that we assume that spatial velocities ($\dot{\mathbb{x}} \in \mathbb{R}^6$) and generalized velocities ($\dot{\mathbb{q}} \in \mathbb{R}^{nv}$) are not equivalent to the time derivatives of rigid body coordinates ($\dot{\boldsymbol{x}} \in \mathbb{R}^7$) and generalized coordinates ($\dot{\boldsymbol{q}} \in \mathbb{R}^{nq}$), respectively, as our focus has been locomoting, underactuated robots and there exists no physically meaningful, minimal representation of orientation in 3D [10]. Implementation-wise, we represent the 3D orientation of the robot's "floating base" using unit quaternions, the rotational velocity using a 3D angular velocity vector, and convert between the two using linear operations (see, e.g., [14]).

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 using 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 straightforward 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, which is driven by an inverse dynamics-based control policy with contact force prediction, to determine the instantaneous actuator forces necessary for the robot to follow the desired trajectory. See Figure 2 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
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. 2: *The user will likely want to generate multiple pose sequences and use them for testing. These sequences can be expanded as the robot begins passing tests (e.g., after trotting is successful, we can move to running, while retaining the sequence for trotting).*

D. Adjusting model parameters 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 $\nabla_j \Phi_{\texttt{model}(i)}(\boldsymbol{q}, \dot{q}, \boldsymbol{u})$ is computed using finite differencing (as described in Figure 3) 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-parameter space at a specific point in state space $(\boldsymbol{q}, \dot{q})$ under the current model parameters $(\boldsymbol{p}_i)$.

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

The parameters to be included in the 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

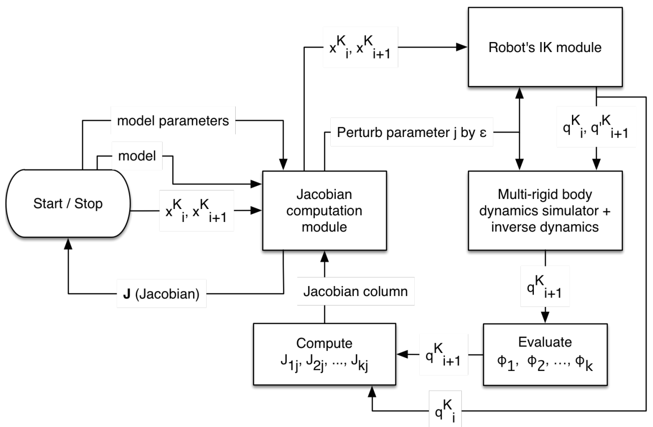


Fig. 3: *Jacobian generation flowchart.*

descent direction $-\mathbf{J}^\dagger \boldsymbol{\ell}$, where † is the pseudo inversion operator, yields an update direction to the robot modeling parameters that should decrease the limit function violations $(\boldsymbol{\ell} = \Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{q}, \boldsymbol{u}))$. The current modeling parameters $(\boldsymbol{p}_i)$ can then be updated to those at the next iteration $\boldsymbol{p}_{i+1}$:

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

for $\alpha \ll 1$ (we currently tune α manually). The designer incorporates expert knowledge into this tuning stage, weighting the gradient update and perhaps making additional modifications to the robot model. Specifically, a designer is permitted to roll-back an update or apply either of two changes repeatedly to the update: (1) zeroing any element of the parameter update vector; (2) applying front-hind symmetry to a specific parameter's update.

E. Algorithmic correctness

There do not exist tractable algorithms for determining whether one or more nonlinear inequality constraints (i.e., limit functions) is/are satisfiable over a discretization of a continuous trajectory; this problem generally corresponds to a non-convex optimization problem. In the absence of a tractable approach to an optimal solution, we considered several alternatives, including optimizing parameters over the entire execution of a trajectory (somewhat similar to trajectory optimization) and optimizing parameters over a "window" sliding over the trajectory. We explored the latter approach, since it was simpler, with the understanding that this approach was liable to modify the parameters one way at time t_a during the trajectory execution only to revert this change at time $t_b > t_a$. We did not encounter this phenomenon but note that the user supervision would be able to prevent such regressions.

IV. Experiments

We perform a test of our interactive design 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 optimized the morphological parameters of a quadrupedal robot to perform a trot at a variable forward velocity. This experiment attempted to adjust the continuous geometric and mass parameters of the links of the quadruped toward achieving the highest speed possible given known actuator stall torque and velocity limits. We used the described system to iteratively update the robot model to increase its maximum trotting speed. We focused on trotting because it is a highly dynamic, known challenging task that requires a robot to make full use of its physical ability.

The first author used our working *Links* quadrupedal robot (depicted in Figure 6) as a design template to create a new quadrupedal robot platform, and used the design software to adapt this design. 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.

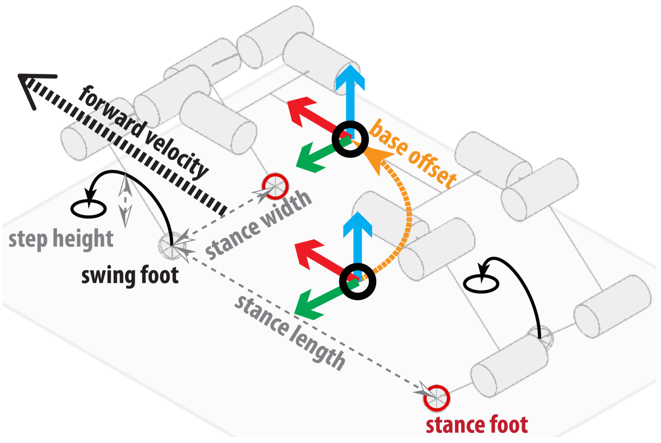


Fig. 4: *Visualization of gait parameters used in our locomotion planner.*

Parameter	**Value**
forward velocity (goal)	50 cm/s
base offset (linear)	$\{0, 0, 0.75 \times l_{longest,\mathrm{ULeg}} + l_{longest,\mathrm{LLeg}}\}$
base offset (angular)	$\{0, 0, 0\}$
step height	1 cm
gait duration	0.3 s
length	$2 \times l_{\mathrm{Base}}$ cm
width	$\{1.1, 1.1\} \times w_{\mathrm{Base}}$ cm
liftoff	$\{0.25, 0.75, 0.25, 0.75\}$
duty factor	$\{0.6, 0.6, 0.6, 0.6\}$

TABLE I: *Gait parameters used in the morphological optimization process. The* duty factor *refers to the proportion of a gait cycle that a single foot spends in stance phase. Other parameters are depicted in Figure 4.*

A. Tasks

We simulated the quadruped locomoting with an initial gait parameters shown in Table I. These values were chosen as a generally functional set of parameters (based on our experience) for a 16 cm tall quadruped. The virtual robot uses a closed loop error-feedback controller (see Figure 5) which accumulates feedback accelerations that are input to a QP-based inverse dynamics controller with contact force prediction [22]. Contact force prediction permits us to simulate the forces acting on the robot at individual time slices of controlled locomotion. If we were to rely on an error-feedback control scheme, the full actuation force required for locomotion would be time-delayed and would thereby hinder accurate limit evaluation.

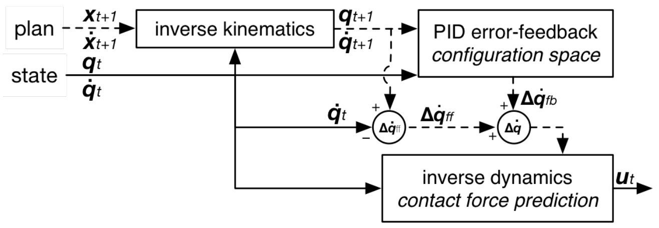


Fig. 5: *Block diagram of the controller utilized by the robot in simulation during the model update process.*

B. Morphological parameters

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, our parameterization requires adjusting only the modeling parameters of half of the robot and reflecting 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 plan to explore topological updates in 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 Tables II and III, respectively.

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 bottom corner 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

TABLE II: *Robot design parameters.*

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

TABLE III: *Limits to robotic hardware considered in the experiment. These are the reported stall torque and max velocity of an inexpensive hobby servo.*

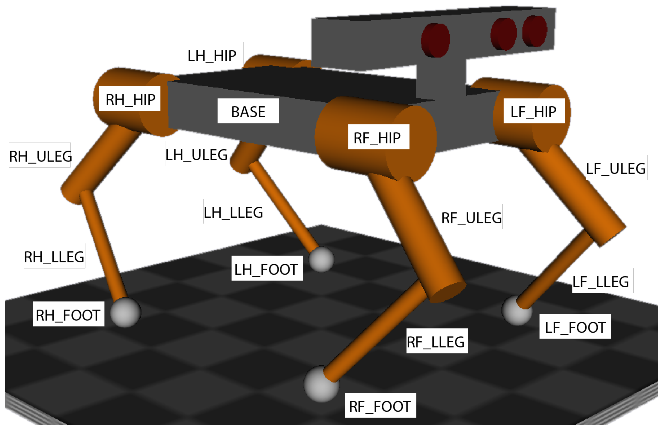


Fig. 6: *The base link (grey) has a box geometry; limbs, originating from the corners of the base have a cylindrical geometry; feet have a spherical geometry. All links also have a density that, with the calculated volume, determine the mass of the link. A Kinect type sensor (unused) is rendered also.*

C. Experimental design

To determine the usefulness of the proposed interactive design process, we compared three candidate robot models: (1) an **`initial`** robot, taken without modification from previous, successful locomotion experiments [22], [7]; (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; and (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 adhere 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.

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 7.

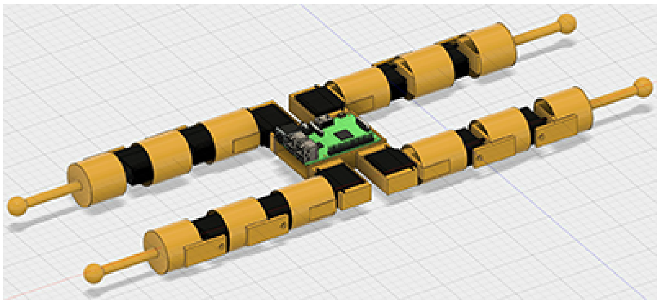


Fig. 7: *The `initial` (non-optimized) robot design used to seed the optimized models.*

V. Results

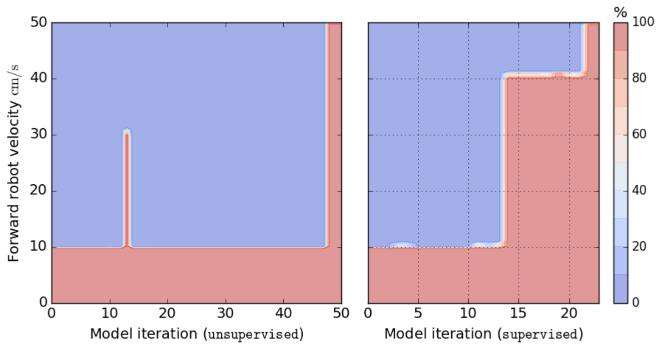


Fig. 8: *The `automated` (left) and `supervised` (right) robot design progress over several virtual model updates. Regions of the plot are colored according to the percentage of the gait that robot model can complete at the specified velocity before violating an actuator limit. Colors blue, white and red coincide with 0, 50, and 100 percent task completion, respectively.*

This section 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 Table I) at the starting forward velocity of 10 cm/s.

The result of the **human in the loop modification process** generated the morphology depicted (as a CAD model) in Figure 9. The **automated process** generated the morphology designed for a 3D printer depicted in Figure 10. The `automated` model yielded 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—likely would have resulted in the robot destroying itself during the first test. Section V-B will note that the seemingly more robust `initial` model broke in the fifth trial during physically situated testing.

A. *Robot performance* in sim

Figure 8 shows that modifying the robot enough to progress past one limit violation was 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 designs achieved the same maximum forward trotting velocity of 50 cm/s in simulation*, leading us to conclude that the automated process resulted in a objectively equivalent, yet qualitatively inferior robot (as will be demonstrated shortly).

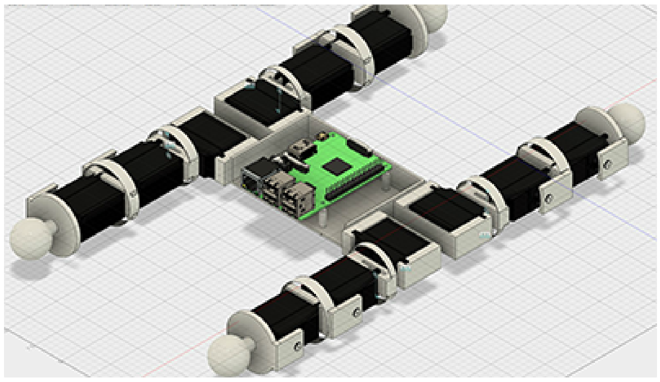


Fig. 9: *The `supervised` optimized robot design produced using the interactive design process.*

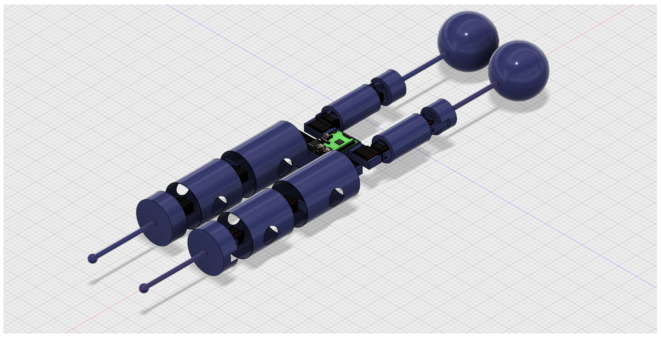


Fig. 10: *The `automated` optimized robot design produced using gradient descent and no human supervision. This design has links that are too short to fit our actuators and link radii that are too wide to achieve a reasonable range of motion.*

B. *Robot performance* in situ

Planning and control for the 3D printed robots was performed by a Raspberry Pi computer looping over a prerecorded configuration space trajectory specific to each

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
$r_{\text{Front},\{\text{Hip,ULeg,LLeg}\}}$	23	92	23
$r_{\text{Hind},\{\text{Hip,ULeg,LLeg}\}}$	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
$m_{\text{Front},\{\text{LLeg,Foot}\}}$	29	1	15
$m_{\text{Front,Hip}}$	17	200	12
$m_{\text{Front,ULeg}}$	18	166	12
$m_{\text{Hind},\{\text{LLeg,Foot}\}}$	29	69	15

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

robot morphology. The operational space trajectory was generated by our gait planner (Figure 4) for the `initial` and `supervised` morphologies—taking into account the model-parameter-dependent gait parameters—and then output for the Raspberry Pi to replay. We defined gait parameters (see Table I) relative to each robot's body dimensions. The situated robot was controlled at a forward velocity of 16.26 cm/s, determined by the approximately 60 Hz publish rate of the Raspberry Pi computer: attempting faster movement produced jumpy behavior when the robot tried to follow the joint trajectory in real-time.

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. The robot successfully completed the 20 gait cycles in only three of the five trials. The robot had trouble supporting its own weight during most trials and then fell to its side in the fourth trial; after repeating this failure in the fifth trial, the robot hardware was irreparably damaged—the servo horn-link interface was stripped. The average velocity during the successful trials was 5.23 cm/s, or about one-third of the commanded velocity. The robot shuffled its feet throughout the trials indicating that it was unable to produce the necessary torque to properly move its long limbs. This discrepancy was readily detected by the morphological modification process, and the violation was then corrected via modifications to the robot parameters (yielding the `supervised` and `unsupervised` models).

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). We attribute the discrepancy between actual and desired trotting speeds to the robot's feet tending to slide during the gait stance phase portions. The robot remained stable and successfully completed all trials.

VI. Discussion

We described a morphological modification process that used simulation, a human in the loop, and a design template

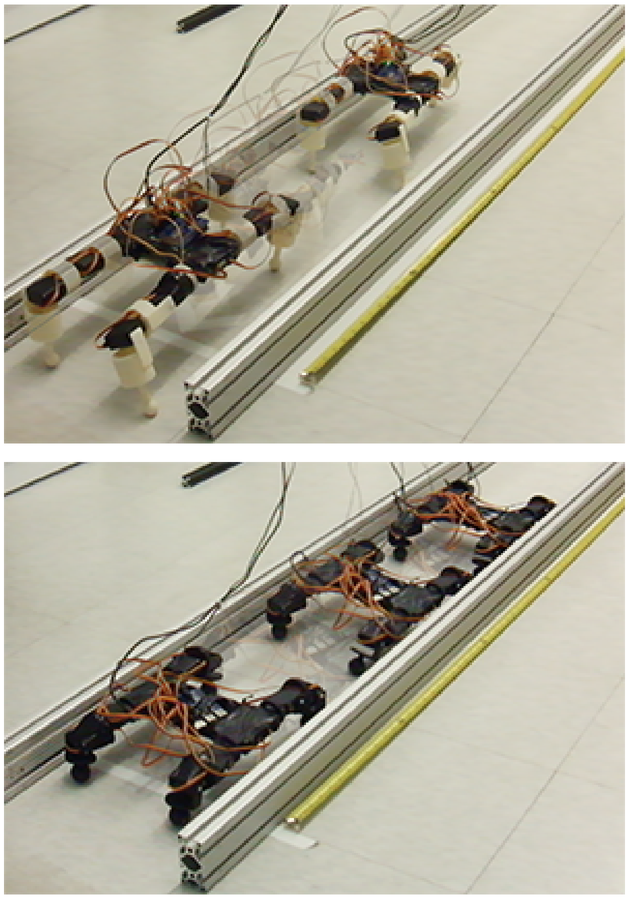


Fig. 11: *A time-lapse of the* `initial` *(left) and* `optimized` *(right) robot trotting for 20 gait cycles. Videos of all 20 situated trials are provided with the supplemental material.*

to modify a poorly performing robot to an effective one. We observed a substantial performance increase between morphologies within our situated trials, reflecting the results from the simulated design process. We expect that further development of such tools will allow hobbyists and professional 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] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne. Locomotion skills for simulated quadrupeds. In *Proc. ACM SIGGRAPH*, 2011.

[6] 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.

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

[8] 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.

[9] 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.

[10] T. R. Kane and D. A. Levinson. *Dynamics: Theory and Applications*. McGraw-Hill, New York, 1985.

[11] 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.

[12] 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.

[13] 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.

[14] P. E. Nikravesh. *Computer-Aided Analysis of Mechanical Systems*. Prentice Hall, 1988.

[15] 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.

[16] 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.

[17] 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.

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

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

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

[21] S. Zapolsky. Pacer. https://github.com/PositronicsLab/Pacer, 2015.

[22] 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.