Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels

MORITZ GEILINGER, ETH Zurich
ROI PORANNE, ETH Zurich
RUTA DESAI, Carnegie Mellon University
BERNHARD THOMASZEWSKI, Université de Montréal
STELIAN COROS, ETH Zurich

Fig. 1. Robotic creatures created with our computational design system employ arbitrary arrangements of legs and wheels to locomote.

We present a computation-driven approach to design optimization and motion synthesis for robotic creatures that locomote using arbitrary arrangements of legs and wheels. Through an intuitive interface, designers first create unique robots by combining different types of servomotors, 3D printable connectors, wheels and feet in a mix-and-match manner. With the resulting robot as input, a novel trajectory optimization formulation generates walking, rolling, gliding and skating motions. These motions emerge naturally based on the components used to design each individual robot. We exploit the particular structure of our formulation and make targeted simplifications to significantly accelerate the underlying numerical solver without compromising quality. This allows designers to interactively choreograph stable, physically-valid motions that are agile and compelling. We furthermore develop a suite of user-guided, semi-automatic, and fully-automatic optimization tools that enable motion-aware edits of the robot's physical structure. We demonstrate the efficacy of our design methodology by creating a diverse array of hybrid legged/wheeled mobile robots which we validate using physics simulation and through fabricated prototypes.

CCS Concepts: • **Computer graphics → Computational geometry and object modeling**; *Physically based modeling*;

ACM Reference Format:
Moritz Geilinger, Roi Poranne, Ruta Desai, Bernhard Thomaszewski, and Stelian Coros. 2018. Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels. *ACM Trans. Graph.* 37, 4, Article 1 (August 2018), 12 pages. https://doi.org/10.1145/3197517.3201368

Authors' addresses: Moritz Geilinger, ETH Zurich; Roi Poranne, ETH Zurich; Ruta Desai, Carnegie Mellon University; Bernhard Thomaszewski, Université de Montréal; Stelian Coros, ETH Zurich.

0730-0301/2018/8-ART1
https://doi.org/10.1145/3197517.3201368

1 INTRODUCTION

Whether it is to help with chores, keep us company, or entertain us, personal robots promise to play a central role in our increasingly technology-driven society. Echoing the trend of mass customization and leveraging recent advances in digital fabrication, our long term goal is to develop algorithmic foundations that will enable these robots to be created on-demand according to the individual needs and preferences of those they serve. In this quest, we join recent research efforts that bridge the fields of animation, fabrication-oriented design and robotics [Bern et al. 2017; Du et al. 2016; Schulz et al. 2017]. Complementing this body of work, we introduce a novel design system for a rich class of mobile robots that employ arbitrary arrangements of legs and wheels for locomotion. Such hybrid robots enjoy the combined versatility of legged and wheeled systems, but they also inherit their compounded challenges: they have many actuated degrees of freedom that need to be precisely coordinated in order to generate motions that are balanced, elegant, and efficient; their kinematics and dynamics are governed by highly non-linear equations; and their motor capabilities and physical design characteristics are inseparably intertwined. For these reasons, creating hybrid mobile robots remains a very difficult and error-prone task.

We present a computation-driven approach to designing, optimizing and synthesizing motions for different breeds of legged/wheeled robots. At the core of our work lies an efficient trajectory optimization formulation tailored to the specific challenges of this class of robotic creatures. Through a unified treatment of feet and wheels, our model enables automatic generation of stable, physically-valid

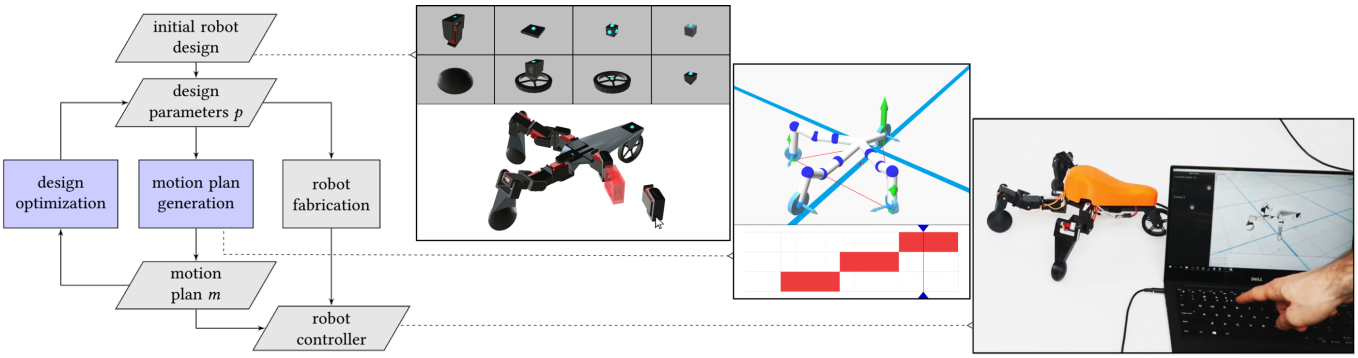


Fig. 2. *High-level overview of our design system:* through a simple drag-and-drop interface, designers can interactively generate a vast array of robotic creatures. Once designs are finished, our system automatically generates physically-valid motions that are skilled and agile. We also present computational solutions for user-driven or automatic optimization of the robot's physical dimensions. The resulting designs can be easily fabricated, leading to compelling physical robots.

walking, rolling and skating motions for user-designed robots. Although these motions are optimal with respect to the morphological characteristics of each individual robot, not all robots are created equal. Indeed, the motor capabilities of different robots can vary drastically. Optimizing design parameters for user created robots is therefore an indispensable piece of the puzzle which we also address in this work. To this end, we develop a suite of computational tools that leverage sensitivity analysis to support manual, semi-automatic and fully automatic design exploration and optimization.

To validate our work, we designed a variety of robotic creatures and corresponding motions, all of which were tested using off-the-shelf physics-based simulators. We further fabricated three of our designs to assess the degree to which physical prototypes match our simulation results.

Succinctly, our main contributions are:

- A versatile trajectory optimization formulation that is used to generate stable, physically-valid motions for a large variety of robots that employ legs and wheels for locomotion
- An analysis of the underlying numerical solver that reveals an effective way to drastically increase convergence rates for the motion optimization process
- A suite of user-guided computational tools that support manual, semi-automatic and fully automatic optimization of the robot's physical dimensions

2 RELATED WORK

Fabrication-aware design is a flourishing topic in Computer Graphics research. This is not surprising, since content generation has been a core topic of the field since its very beginnings, and digital fabrication machines are simply new types of output devices. However, going from virtual environments to the real world introduces many new challenges that must be addressed. For example, in recent years we have seen computational design approaches for objects that are lightweight yet strong [Lu et al. 2014; Stava et al. 2012], objects whose optimized mass distribution allows them to stand, spin or float stably [Bächer et al. 2014; Musialski et al. 2015; Prévost et al. 2013], physical characters that mirror the range of motion of their virtual counterparts [Bächer et al. 2012; Calì et al. 2012; Ureta et al. 2016], and increasingly complex mechanisms and mechanical automata designed to generate specific motions [Ceylan et al. 2013; Coros et al. 2013; Megaro et al. 2017; Song et al. 2017; Zhang et al. 2017]. These computational tools share the same high-level goal as ours: enabling non-experts to create complex physical artifacts without requiring domain specific knowledge.

To increase the range of functionality for digitally-fabricated objects, researchers are also investigating computational approaches to embedding sensors and various other electromechanical components into their designs [Bächer et al. 2016; Follmer et al. 2015; Villar et al. 2012; Weichel et al. 2013]. These research efforts build a bridge between the fields of HCI, Computer Graphics and Robotics, and our work follows this spirit. In particular, closely related to our work are algorithmic methods to design origami-inspired robots [Schulz et al. 2017] as well as walking automata [Bharaj et al. 2015] and robotic creatures [Megaro et al. 2015]. The computational techniques we describe in this paper complement this body of work by targeting a diverse class of mobile robots that move using arbitrary arrangements of legs and wheels. The designs we support have many degrees of freedom and their hybrid wheeled/legged nature demands motion repertoires that are much richer and more intricate than those of robots relying largely on quasi-static walking [Megaro et al. 2015]. We therefore present a new, highly efficient trajectory optimization approach that automatically generates walking, rolling, skating or gliding motions, as appropriate given the morphological designs of different robots.

Building on a growing body of literature [Coros et al. 2013; Ha et al. 2017; Megaro et al. 2017; Pérez et al. 2017; Umentani et al. 2015], we leverage sensitivity analysis to establish a relationship between the motions a robot can generate and its physical design parameters. In particular, the method described in [Ha et al. 2017] nicely complements our work: while their formulation fine-tunes robot designs such that actuation forces are reduced, the suite of computational tools that we propose are specifically developed to support motion-aware manual, semi-automatic and fully automatic design exploration and optimization. Our work also draws inspiration from a number of specific, hybrid robot designs presented in the robotics literature [BostonDynamics 2017; Endo and Hirose 2008;

Smith et al. 2006]. These one-off designs are feats of engineering developed by teams of seasoned domain experts. Our long term goal is to allow even casual users to create robotic creations that approach the same level of sophistication.

3 OVERVIEW

Our system allows its users to create unique robot designs by connecting together different types of mechanical components in a mix-and-match manner. This design process is illustrated in Fig. 2 and can also be seen in the accompanying video. Our implementation of the underlying graphical user interface is similar to the one employed by [Desai et al. 2017], and the database of components we use for all our results consists of servomotors, 3D printable connectors and three types of end effectors: *actuated wheels* whose angular speed is controlled by motors, *passive wheels* that can spin freely about their rotation axis, and *welded wheels* that afford no motion relative to the body part they are attached to. Welded wheels are used to model feet that roll on the ground as the robots are moving, and when their radii are set to 0, they become equivalent to the point foot model commonly used by motion planning algorithms.

The *morphological design* of each robot is generated from a user-specified hierarchical arrangement of components: servomotors correspond to actuated joints, connectors define the geometric shape of each rigid link of the robot, and end-effectors specify the mechanical behavior of the components that will come into contact with the environment as the robot moves. This input directly defines the inertial parameters for each body part, 3D models for fabrication, as well as the rotation axes for each wheel and each joint actuator. With the resulting robot design as input, our trajectory optimization method (Sec. 4) generates physically-valid walking, rolling, gliding or skating motions. These motions are automatically tailored according to the morphological characteristics of each individual design. We leverage the particular structure of our motion synthesis formulation to significantly accelerate the underlying numerical solver (Sec. 4.3). This allows designers to interactively choreograph motions that are stable, agile and compelling.

To ensure that a robot functions as envisioned by its designer, we also develop computational solutions for manual, semi-automatic and fully automatic optimization of the robot's physical dimensions (Sec. 5). These tools leverage sensitivity analysis and allow even non-experts to explore the often unintuitive relationship between design parameters and motor capabilities. Prior to fabrication, designs are validated through off-the-shelf physics simulators. We use proportional-derivative controllers to generate torques for every actuated joint of the simulated robot, and feed-forward velocity controllers for its actuated wheels. Time-varying joint angle and wheel speed targets for these low-level controllers are directly generated from the optimized motions. We use the Open Dynamics Engine [ODE 2007] as a black-box simulator for all our results.

4 MOTION GENERATION

Our motion generation model builds on trajectory optimization techniques that reason in terms of a robot's *centroidal dynamics* [Orin et al. 2013]. This class of methods exploits the fact that modeling the evolution of the robot's aggregate linear and angular momenta over time is much simpler than considering its full-body dynamics. Nevertheless, this simplified dynamics representation can be easily complemented by geometric constraints to ensure that the generated motions are consistent with the robot's kinematics [Dai et al. 2014]. Because they strike a favorable balance between predictive power, simplicity and computational efficiency, models based on centroidal dynamics are quickly gaining in popularity.

In this section, we describe a new mathematical formulation that leverages the concept of centroidal dynamics to efficiently generate dynamic motions for a diverse array of hybrid robotic creatures. The general nature of our formulation allows physically-valid walking, rolling, gliding and skating motions to emerge naturally as a function of the morphological design of each individual robot.

4.1 Optimization Model

The input to our motion optimization model consists of robots with arbitrary user-generated morphology. The robots interact with the environment through their end effectors, which can be located on any of their body parts. Without loss of generality, each end effector is assumed to be a wheel described by its radius r, mounting location $\hat{\mathbf{l}}$ on body part b, and rotation axis $\hat{\mathbf{a}}$ expressed in the local coordinate frame of b. Our motion optimization model supports passive and actuated wheels. Depending on the type of wheel, different constraints are instantiated as discussed in below.

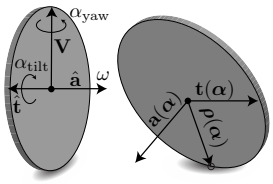
Using a direct transcription approach, we turn to a time-discretized setting and represent a *motion plan* $\mathbf{m} = \{\mathbf{m}_1, \ldots, \mathbf{m}_T\}$ as a set of vectors $\mathbf{m}_i$ that span a planning horizon with length of time hT, where h is the amount of time between consecutive time samples. The subscript indexes specific samples in time, and $\mathbf{m}_i$ is defined as:

$$\mathbf{m}_i = \{\mathbf{q}_i, \mathbf{c}_i, \mathbf{e}_i^1, \ldots \mathbf{e}_i^n, \mathbf{f}_i^1, \ldots \mathbf{f}_i^n, \omega_i^1, \ldots \omega_i^n, \boldsymbol{\alpha}_i^1, \ldots \boldsymbol{\alpha}_i^n\} \quad (1)$$

For every time index i, the vector $\mathbf{q}_i = \{q_i^1, \ldots, q_i^K\}$ denotes the pose of the robot, which consists of the position and orientation of the root link, as well as the joint angles that describe the relative orientation between articulated body parts; $\mathbf{q}_i$ is thus a vector of size $K = 6 + N$, where N is the number of joints. The corresponding *centroidal coordinate frame* $\mathbf{c} = \{\mathbf{x}, \boldsymbol{\theta}\}$ represents the robot's center-of-mass (COM) position $\mathbf{x}$ and global orientation $\boldsymbol{\theta}$. Each of the robot's n end effectors are referenced by superscript $j, 1 \leq j \leq n$. The points $\mathbf{e}$ define the location of the end effectors in a global coordinate frame (e.g. the points where each wheel should make contact with the environment) at different moments in time and $\mathbf{f}$ represents the force imparted by the robot onto the environment through each end effector. For each wheel in the robot's design we also store its instantaneous, world-relative angular speed ω, and two rotation angles $\boldsymbol{\alpha} = \{\alpha_{\text{tilt}}, \alpha_{\text{yaw}}\}$ that define the global orientation of its rotation axis:

$$\mathbf{a}(\boldsymbol{\alpha}) = W(\boldsymbol{\alpha}, \hat{\mathbf{a}}) = R_{\mathbf{v}}(\alpha_{\text{yaw}}) R_{\hat{\mathbf{t}}}(\alpha_{\text{tilt}}) \hat{\mathbf{a}}, \quad (2)$$

where R denotes a typical rotation operator. Note that we use the $\hat{}$ accent to denote quantities expressed in local coordinates. In the equation above, $\hat{\mathbf{a}}$ and $\mathbf{a}$ therefore represent the rotation axis of the wheel expressed in local and global coordinate frames, respectively. As illustrated in the inset figure, the tilt axis $\hat{\mathbf{t}}$ is defined as



lie at the intersection of the ground plane with the wheel plane. It is computed in the local coordinate frame of the wheel as $\hat{\mathbf{t}} = \hat{\mathbf{a}} \times \mathbf{v}/|\hat{\mathbf{a}} \times \mathbf{v}|$, where $\mathbf{v}$ is the vertical axis in global coordinates. The tilt axis in global coordinates, $\mathbf{t}(\boldsymbol{\alpha})$, is computed as $W(\boldsymbol{\alpha}, \hat{\mathbf{t}})$, and carries special meaning: it represents the only valid direction of movement for the wheel at any specific moment in time. Another important quantity, $\hat{\boldsymbol{\rho}} = \hat{\mathbf{t}} \times \hat{\mathbf{a}} r$, where r is the radius of the wheel, is the vector from the center of the wheel to the point where the end effector will make contact with environment (assuming locomotion on flat ground).

Our decision to model and parameterize wheels as separate entities warrants a brief discussion. We initially considered treating wheels as additional rigid bodies in the robot's morphological structure (i.e. their motion would be stored as part of $\mathbf{q}$, as for any other joint). While this modeling choice would reduce the overall number of parameters we need to optimize for, it presents two major downsides. First, even simple operations such as determining the point on a wheel that contacts the environment would require complex kinematic computations, because in the coordinate frame of a rigid body, this point does not remain fixed. In contrast, it is easily seen that upon transformation to global coordinates, $\boldsymbol{\rho} = W(\boldsymbol{\alpha}, \hat{\boldsymbol{\rho}})$ maintains its meaning, i.e. it still represents the vector from the center of the wheel to the point that is closest to the ground. As a result, the objectives and constraints formulated below take on much simpler forms and become faster to evaluate. Second, as discussed in Sec 4.3, optimization terms that include the full kinematic model of the robot are highly non-convex, and if left untreated they negatively affect convergence rates. The auxiliary variables we introduce for wheels allow us to localize these numerical issues to just two consistency constraints, which we can therefore analyze and address in isolation.

With the parameterization of the motion plan in place, we now turn our attention to the constraints and objectives that govern the motions of wheeled/legged hybrid robots.

Kinematics and Dynamics: Leveraging the centroidal dynamics representation, the global motion of the robot is governed by the familiar Newton-Euler equations. For our discrete setting, these equations take the form:

$$\sum_{j=1}^{n} \mathbf{f}_i^j + M\mathbf{g} \qquad = M\ddot{\mathbf{x}}_i \,, \forall i \tag{3}$$

$$\sum_{j=1}^{n} (\mathbf{e}_i^j - \mathbf{x}_i) \times \mathbf{f}_i^j \quad = \mathbf{I}\ddot{\boldsymbol{\theta}} + \dot{\boldsymbol{\theta}} \times \mathbf{I}\dot{\boldsymbol{\theta}} \,, \forall i \tag{4}$$

As before, the subscript i denotes a specific time index and the superscript j refers to individual end effectors. The total mass of the robot is M and its moment of inertia I is computed about the robot's COM. The center of mass acceleration, $\ddot{\mathbf{x}}_i$, is estimated using finite differences: $\ddot{\mathbf{x}}_i = (\mathbf{x}_{i-1} - 2\mathbf{x}_i + \mathbf{x}_{i+1})/h^2$, where h is the time step. Similarly, $\ddot{\boldsymbol{\theta}}$ is computed using finite differences that operate on axis-angle representations of the centroidal coordinate frame's change in orientation between time steps.

The motion of the centroidal coordinate frame is a function of the forces that the robot's end effectors impart onto the environment. To ensure their physical feasibility, these forces are subjected to constraints imposed by a typical Coulomb friction model:

$$\mathbf{f}_n \geq 0 \,, |\mathbf{f}_t| \leq \mu \mathbf{f}_n \,, \tag{5}$$

where $\mathbf{f}_t$ and $\mathbf{f}_n$ denote the tangential and normal component of $\mathbf{f}$, and μ is the coefficient of friction. Forces generated by unactuated wheels are further constrained to have a vanishing component in the direction along which the wheel is free to move (i.e. they can only push on the ground in an orthogonal direction):

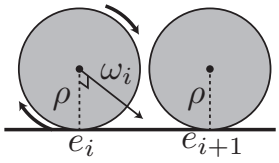
$$\mathbf{f} \cdot \mathbf{t}(\boldsymbol{\alpha}) = 0 \,, \tag{6}$$

Furthermore, end effectors can only generate ground reaction forces when they are in contact with the environment. This behavior is enforced through constraints of the form:

$$(1 - c)\mathbf{f} = 0 \,, \tag{7}$$

where the binary contact flags c are specified by a foot fall pattern per end effector, per time step [Megaro et al. 2015]. These flags represent the schedule of contacts that characterize different locomotion gaits; $c = 1$ indicates that an end effector must be in contact with the ground, while $c = 0$ denotes swing phases during which end effectors cannot generate ground reaction forces. Constraints 5-7 are applied for all time samples i and all end effectors j.

When end effectors are in contact with the ground, their motion must be subject to kinematic no-slip constraints. Recall that $\mathbf{e}_i^j$ represents the location where end effector j makes contact with the ground at time index i. We wish the evolution of these points to be consistent with the motion of the wheel, as shown in the inset figure. If at some moment in time the wheel has angular velocity $\boldsymbol{\omega}$ and the relative velocity between the wheel and the ground at the contact point is $\mathbf{0}$, then the center of the wheel must have velocity $\boldsymbol{\omega} \times \boldsymbol{\rho}$. The time derivative of $\mathbf{e}_i^j$ must take on the same value, so the no-slip constraint is formulated as:



$$\left(\frac{\mathbf{e}_{i+1}^j - \mathbf{e}_i^j}{h} + \omega_i^j \mathbf{a}(\boldsymbol{\alpha}_i^j) \times \boldsymbol{\rho}(\boldsymbol{\alpha}_i^j) \right) c_i^j = 0 \tag{8}$$

While the constraint above bounds end effector velocities such that their motions model rolling wheels, the vertical component of their motion must also be prescribed. We accomplish this through a simple constraint that asks that the normal component of the end effector position is 0 in stance (i.e. $c = 1$), or attain a user-specified value w_h otherwise:

$$c_i^j \mathbf{e}_i^j \cdot \mathbf{n} + (1 - c_i^j)(\mathbf{e}_i^j \cdot \mathbf{n} - w_h) = 0 \tag{9}$$

The motion of the end effectors is further optimized to ensure that the motions that are generated are collision-free. For this purpose, we use inequality constraints applied to every pair of end-effectors:

$$||\mathbf{e}_i^j - \mathbf{e}_i^k||_2^2 \geq (r^j + r^k + \beta)^2, \forall j, k \leq n \tag{10}$$

where $\beta = 2cm$ is a safety factor.

We note that the constraints imposed on the motion of end effectors and the forces they generate are a function of the types of wheels employed in each design. These constraints, while each simple in formulation, lead to interesting and often surprising motions.

Consistency Constraints: The terms described above operate on the centroidal coordinate frame and the set of auxiliary end effector variables introduced by our model. We ensure that the motion of the robot is in sync with these quantities through a set of consistency constraints. First, we ask that the trajectory of the robot's center of mass matches the linear motion of the centroidal coordinate frame:

$$\varphi_{CoM}(\mathbf{q}_i) - \mathbf{x}_i = 0\,, \tag{11}$$

where $\varphi_{CoM}(\mathbf{q})$ outputs the robot's center of mass given pose $\mathbf{q}$. Likewise, the orientation of the robot's body, $\varphi_\theta(\mathbf{q})$, must match that of the centroidal coordinate frame:

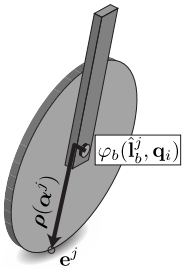
$$\varphi_\theta(\mathbf{q}_i) * R(\boldsymbol{\theta}_i)^{-1} = I \tag{12}$$

To mirror the motion prescribed through the auxiliary variables for wheels, two constraints must be satisfied:

$$\varphi_b(\hat{\mathbf{a}}^j, \mathbf{q}_i) - \mathbf{a}(\boldsymbol{\alpha}_i^j) = 0\,, \tag{13}$$

$$\varphi_b(\hat{\mathbf{l}}^j, \mathbf{q}_i) + \boldsymbol{\rho}(\boldsymbol{\alpha}_i^j) - \mathbf{e}_i^j = 0\,, \tag{14}$$

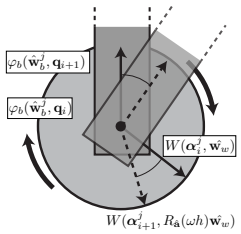
Here, φ_b is the forward kinematics function that computes world coordinates of points or vectors defined in the local coordinate frame of rigid body b which wheel j is mounted on. The first constraint therefore demands that the wheel axis, as seen from the coordinate frame of b, is aligned with the wheel axis computed through the auxiliary variables. The second constraint further requires that the position of the wheel's center of rotation satisfies the kinematic relationship illustrated in the inset figure.



The last consistency constraint is instantiated only for welded wheels that must have zero velocity relative to the robot body part they are mounted on. Before we provide the formulation of the constraints that model *welded wheels*, recall that the wheel speed ω is specified in a global coordinate frame, and not relative to b. Consequently, we must establish a correspondence between the global motion of body part b, and the motion represented by the auxiliary parameters of the wheel. We can do this efficiently by requiring that vectors lying in the plane of the wheel, as seen from the two different coordinate frames, follow equivalent movement patterns. The constraint therefore becomes:

$$\varphi_b(\hat{\mathbf{w}}_b^j, \mathbf{q}_i) \times \varphi_b(\hat{\mathbf{w}}_b^j, \mathbf{q}_{i+1}) = W(\boldsymbol{\alpha}_i^j, \hat{\mathbf{w}}_w) \times W(\boldsymbol{\alpha}_{i+1}^j, R_{\hat{\mathbf{a}}}(\omega h)\hat{\mathbf{w}}_w) \tag{15}$$

where $\hat{\mathbf{w}}_b$ and $\hat{\mathbf{w}}_w$ are two *reference vectors* that lie in the plane of the wheel, specified in the local coordinate frame of the parent body b and the auxiliary wheel frame, respectively. We note that the end effector parameterization does not explicitly store the orientation of the wheel about its rotation axis. For this reason, for the second term on the right hand side, we first rotate the vector $\hat{\mathbf{w}}_w$ according to the angular speed of the wheel, and then compute its world coordinates, as illustrated in the inset figure. This formulation does not require $\hat{\mathbf{w}}_b$ and $\hat{\mathbf{w}}_w$ to be the same vector, as the cross product operator outputs the same result for any pair of reference vectors we choose.



Physical Hardware Constraints: With the motion of the robot consistent with its centroidal dynamics, it is important to ensure that the limitations of physical actuators are also respected. We therefore implement bound constraints for the range of motion of each joint angle $\mathbf{q}_k$, its rate of change $\dot{\mathbf{q}}_k$ and the angular speed ω of active wheels. The values for the bound constraints are hardware-specific, but they otherwise take on standard forms, which we omit for brevity.

Boundary Conditions: Our trajectory optimization model supports the generation of periodic motions as well as motions with prescribed starting and end states. For periodic motions, we implement constraints that ask that the poses of the robot at the start and end of the motion plan to be identical, i.e. $\mathbf{q}_1 = \mathbf{q}_T$. Constraints for prescribed starting or end states take on similar forms, and the target values can be poses generated for any other motion. In this way, our method can easily generate transitions between periodic motions to create motion graphs [Kovar et al. 2002].

Functional Objectives and Motion Regularizers. Complementing the constraints detailed above, we also implement several objectives that provide interactive control over the generated motions. The walking speed, for example, is controlled by specifying a target offset between the first and last configuration of the centroidal coordinate frame: $||(\mathbf{x}_T - \mathbf{x}_1) - \mathbf{t}||_2$. The turning rate is similarly controlled by specifying a target yaw angle between $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_T$. To promote the generation of smooth motions, we also include a regularizing term defined as $(\mathbf{q}_{i-1} - 2\mathbf{q}_i + \mathbf{q}_{i+1})$.

To provide further control over the generated motions, we allow users to interactively specify target positions for the robot's end effectors or COM trajectory. This interaction mode, which is demonstrated in the accompanying video, is supported by simple objectives of the form $||\mathbf{e}_i^j - \mathbf{e}_{\text{target}}||_2$ or $||\mathbf{x}_i^j - \mathbf{x}_{\text{target}}||_2$. We call these *choreography* objectives, and they are directly instantiated or removed by the user as desired. Supported by the real-time feedback enabled by the efficient solver described in Sec. 4.3, we found this interactive motion synthesis mode to be very effective. This user-in-the-loop optimization scheme can be used both to help the optimization overcome undesirable local minima when they occur, as well as to shape the style of the resulting motions.

4.2 Numerical Solution

One common approach to solving constrained optimization problems is through Sequential Quadratic Programming. We initially pursued this approach but, arguably due to the highly non-linear constraints required for our formulation, we could not find a strategy (i.e., merit function and line search parameters) that would reliably

lead to good convergence. We therefore resorted to a penalty-based approach that allows us to isolate the most challenging constraints into specific objectives that can then be treated in a numerically stable way. To transform equality constraints into penalty terms, it suffices to minimize their inner product: constraint $\mathbf{f}(\mathbf{x}) = \mathbf{b}$ becomes $(\mathbf{f}(\mathbf{x}) - \mathbf{b})^T(\mathbf{f}(\mathbf{x}) - \mathbf{b})$. Inequality constraints are slightly more involved. As in [Bern et al. 2017], we first define a C^2, piece-wise polynomial function $\psi(x)$ as:

$$\psi(x) = \begin{cases} 0 & x \leq -\varepsilon \\ \frac{1}{6\varepsilon}x^3 + \frac{1}{2}x^2 + \frac{\varepsilon}{2}x + \frac{\varepsilon^2}{6} & -\varepsilon \leq x < \varepsilon \\ x^2 + \frac{\varepsilon^2}{3} & \text{otherwise} \end{cases} \tag{16}$$

Each scalar inequality constraint $f(\mathbf{x}) \leq b$ is then modeled as $\psi(f(\mathbf{x}) - b)$. The function $\psi(x)$ is a quadratic penalty term when $x \geq \epsilon$, it is zero if $x \leq -\epsilon$, and it behaves as a smooth interpolant otherwise. The constant ϵ often affords an intuitive interpretation. For example, when placing a bound on the velocity of a motor, we set ϵ to 10% of its maximum speed.

To solve the resulting optimization problem, we define a function $E(\mathbf{m})$ as a weighted sum of all objectives and penalty terms associated with the equality and inequality constraints introduced earlier. We use a weight of 10000 for all penalty terms. The weights for the functional objectives are set to 50, while the motion regularizers are assigned a weight of 0.1.

To minimize $E(\mathbf{m})$, we use Newton's Method coupled with a backtracking line search method. Once the optimization process converges, we alert the user if the residual of any constraint penalty term is above an acceptable threshold. This could be an indication of an invalid robot design, as discussed in Sec. 5.

4.3 Further Analysis and Optimization Speedup

Although all gradients and Hessians are computed analytically in our framework (we verified our implementation against numerical differentiation estimates), the trajectory optimization algorithm is still quite slow to converge, as illustrated in Fig. 3. This behavior is often caused by objectives with an indefinite Hessian. If the Hessian is not positive semi-definite (PSD), the Newton step may result in a non-descent search direction, which explains the "jittery" nature of the convergence plot. To avoid this artifact, indefinite Hessians must be modified such as to remove negative eigenvalues. In many cases, one might opt to simply *regularize* the Hessian by adding a scaled identity matrix to it. However, determining the optimal coefficient value is costly. If it is too low, the Hessian will remain indefinite, and too high a value will slow down progress. Dynamic regularization schemes, which we also tested, start out with a small coefficient that is progressively increased if the search direction is invalid. Nevertheless, every iteration requires an attempt to solve the underlying linear system, so a different strategy is needed. Furthermore, for multi-objective problems, while summing up the contributions from different objectives might result in an overall PSD Hessian, individual objectives that are badly behaved might still hinder progress. To pinpoint the root cause of the problem, we examined the Hessians of all our objectives. We found that the culprit was hidden in the consistency constraints (11)-(14). For example, when written as a penalty term, (14) has the form

$$E(\mathbf{q}_i, \boldsymbol{\alpha}_i^j) = \|\varphi_b(\hat{\mathbf{a}}^j, \mathbf{q}_i) - \mathbf{a}(\boldsymbol{\alpha}_i^j)\|^2 = \|\varphi_b - \mathbf{a}(\boldsymbol{\alpha}_i^j)\|^2 \tag{17}$$

where we use $\varphi_b := \varphi_b(\hat{\mathbf{a}}^j, \mathbf{q}_i)$ for brevity. Its gradient w.r.t $\mathbf{q}_i$ is

$$\nabla_{\mathbf{q}_i} E(\mathbf{q}_i, \boldsymbol{\alpha}_i^j) = \mathbf{J}_{\mathbf{q}_i}\varphi_b^T(\varphi_b - \mathbf{a}(\boldsymbol{\alpha}_i^j)) \tag{18}$$

where $\mathbf{J}_{\mathbf{q}_i}\varphi_b$ is the Jacobian of φ_b w.r.t. $\mathbf{q}_i$. The Hessian with respect to $\mathbf{q}_i$ is therefore

$$\nabla^2_{\mathbf{q}_i} E(\mathbf{q}_i, \boldsymbol{\alpha}_i^j) = \sum_{k=1}^{3} (\mathbf{J}_{\mathbf{q}_i}\varphi_b)_{(k)} (\mathbf{J}_{\mathbf{q}_i}\varphi_b)_{(k)}^T + \left[\frac{(\partial \mathbf{J}_{\mathbf{q}_i}\varphi_b)^T}{\partial q_i^k} (\varphi_b - \mathbf{a}(\boldsymbol{\alpha}_i^j)) \right]_{k=1}^{K}, \tag{19}$$

where we use $A_{(k)}$ for the k'th column of a matrix A, and $[\mathbf{v}_k]_{k=1}^k$ for the concatenation of K vectors $\mathbf{v}_k$. The first term is always PSD since it is a sum of outer products of vectors. However, we found that the second term is often indefinite. We therefore simply exclude it when computing the Hessian. We apply the same modification to the other consistency terms and call the result the *Filtered Hessian*. This simplification is akin to a Gauss-Newton approximation, but we note we only remove second derivatives of (17) with respect to $\mathbf{q}_i$. A standard Gauss-Newton approximation would also remove second derivatives with respect to other parameters (e.g. $\boldsymbol{\alpha}_i^j$)), as well as mixed derivative terms. Furthermore, Gauss-Newton is traditionally used on the entire objective when second-order derivatives are too difficult or too expensive to evaluate. We use our approximation on only a select few objectives. This simple filtering operation results in remarkably faster and smoother convergence, as can be seen in Fig. 3. This result also highlights another benefit of our formulation that employs auxiliary variables to model end effectors: it allows these types of numerical problems to be isolated and addressed in a targeted way.

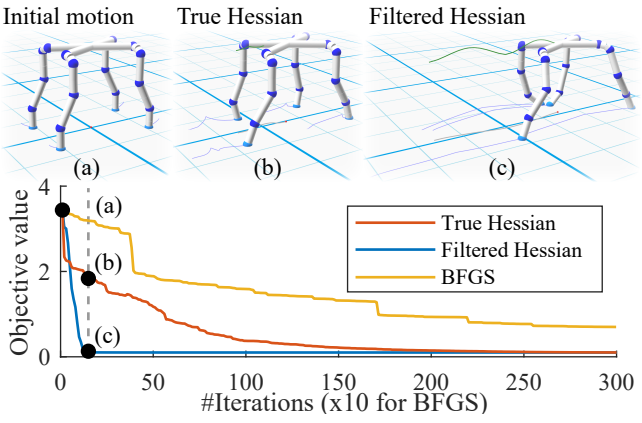
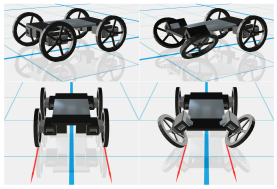


Fig. 3. Comparison of convergence rates for the motion optimization process using the True Hessian and the Filtered Hessian approximation (Sec 4.3). The configurations of the robot in (b) and (c) correspond to the motion plan after 10 optimization steps. The non-smooth end effector trajectories shown in (b) provide a visual indication that the motion has not converged. L-BFGS, a common quasi-Newton method, features very poor convergence for our problem.

5 GUIDED EXPLORATION AND OPTIMIZATION OF MORPHOLOGICAL DESIGNS

Motivation: Our early experiments with the motion optimization model described in the previous section exposed an interesting challenge: the relationship between the morphological design of a robot and the motions it can generate can be very unintuitive. We illustrate this challenge with the example shown in the inset figure. The design on the left is a simple car with four parallel wheels, all of which are actuated. When asked to move forward at constant speed, the trajectory optimization method generated the trivial motion plan that one would expect. The design shown on the right is the same car, but with the front wheels tilted by 45 degrees. This seemingly innocent editing operation resulted in the robot no longer being able to move forward as expected. This is because tilting the front wheels has the unintended consequence of lifting them off the ground. The pitch angle of the robot's body must therefore be adjusted such that all four wheels are once again in contact with the environment. In this new configuration, however, as a result of the combined tilt and pitch of the front wheels, the directions along which they can move are no longer parallel – recall that these directions, which are visualized as red arrows, are given by the intersection of the ground plane with the plane that the wheels lie in. Slip-free motion is therefore no longer possible. With this design flaw identified, the robotic car is easy to fix: lowering the front wheels by just the right amount eliminates the need for the body pitch, ensuring the directions of movement for all wheels are once again in agreement.



Technical solution: The simple example described above highlights the subtleties and potential pitfalls inherent to the task of creating mobile robots. Needless to say, analyzing and finding problems with designs becomes much more difficult as they increase in complexity. Computational approaches to correcting flaws and improving user-generated designs is therefore of utmost importance. We explore solutions to this technical challenge through a suite of computational tools that enable system-guided manual, semi-automatic and fully automatic design methodologies. These tools leverage the fact that the sensitivities of optimal motions with respect to morphological design parameters encode very valuable information.

We treat each user-generated design as a *parameterized morphological template* Ψ which takes as input a vector $\mathbf{p}$. For our implementation, elements of $\mathbf{p}$ encode the location of each joint and each end effector in the local coordinate frame of their parent rigid bodies. $\Psi(\mathbf{p})$ therefore outputs a specific robot design, and different vectors $\mathbf{p}$ result in robots that have the same morphology but different body proportions. Robots generated with our graphical design system provide both a morphology, which remains fixed, and an initial set of parameters $\mathbf{p}_0$. The motions generated through trajectory optimization and the robot's morphological parameters are deeply intertwined through the forward kinematics functions φ_b, φ_{CoM} and φ_θ that appear in Eq. 11-15. Without loss of generality, we can therefore express the motion of the robot as a function of its design parameters,

$$\mathbf{m}(\mathbf{p}) = \arg\min_{\tilde{\mathbf{m}}} E(\tilde{\mathbf{m}}, \mathbf{p}) \,, \tag{20}$$

where the optimization energy E was defined in Sec. 4.2.

Although the function $\mathbf{m}(\mathbf{p})$ does not afford an analytic solution, we leverage the fact that $\mathbf{m}$ and $\mathbf{p}$ are coupled through E to compute an explicit map relating them. To derive this map, we note that as $\mathbf{p}$ changes, we can always compute a new motion such that $\mathbf{G}(\mathbf{m}, \mathbf{p}) = \partial E/\partial \mathbf{m} = 0$, i.e. $\mathbf{m}$ is the minimizer of E for the new design parameters. Consequently, the total derivative of $\mathbf{G}(\mathbf{m}, \mathbf{p})$ with respect to $\mathbf{p}$, $d\mathbf{G}/d\mathbf{p}$, vanishes always. Applying the chain rule, we obtain

$$\frac{d\mathbf{G}}{d\mathbf{p}} = \frac{\partial \mathbf{G}}{\partial \mathbf{m}} \frac{\partial \mathbf{m}}{\partial \mathbf{p}} + \frac{\partial \mathbf{G}}{\partial \mathbf{p}} = \mathbf{0} \,. \tag{21}$$

This expression exposes the Jacobian $\mathbf{J} = \frac{\partial \mathbf{m}}{\partial \mathbf{p}}$, which captures to first order how motion parameters $\mathbf{m}$ need to change as the design parameters $\mathbf{p}$ change such that the solution remains on the manifold of optimal motions, i.e., $\mathbf{G}(\mathbf{m}, \mathbf{p}) = \mathbf{G}(\mathbf{m} + \partial\mathbf{m}, \mathbf{p} + \partial\mathbf{p}) = \mathbf{0}$. Computing this Jacobian requires the Hessian $\partial\mathbf{G}/\partial\mathbf{m}$, which we compute analytically, and the term $\partial\mathbf{G}/\partial\mathbf{p}$, which we estimate numerically. With $\mathbf{J}$ at hand, we describe three system-guided design editing modes supported by our computational framework.

Manual Design Mode: As shown in the accompanying video, our computational framework provides an intuitive interface to directly edit the morphological parameters of a robot design. The goal of the *manual design mode* is to enable users to freely explore the design space. As the body proportions of robotic creatures are interactively adjusted, our computational system provides near-instantaneous feedback by generating and displaying corresponding optimal motions. We achieve this performance by coupling the efficient numerical treatment described in Sec. 4.3 with a simple warm-starting scheme. Briefly, through the Jacobian $\partial\mathbf{m}/\partial\mathbf{p}$, user-provided edits $\Delta\mathbf{p}$ are explicitly mapped to the changes in motion $\Delta\mathbf{m}$ that they induce, $\Delta\mathbf{m} = \partial\mathbf{m}/\partial\mathbf{p}\Delta\mathbf{p}$. We therefore update the current motion plan by $\Delta\mathbf{m}$ and then proceed with numerical optimization.

Semi-Automatic Design Mode: While manual editing enables free-form exploration of the relationship between robot designs and corresponding optimal motions, it is also important to have the ability to optimize designs according to specific functional goals. In general, these goals can be defined in a variety of ways. The option we explore for our *semi-automatic design mode* is the following: starting from an initial robot $\Psi(\mathbf{p}_0)$ and associated motion $\mathbf{m}(\mathbf{p}_0)$, we enable user-driven generation of design variations that ensure specific features of the motion $\mathbf{m}^f \subseteq \mathbf{m}$ are affected as little as possible. In other words, we allow the user to navigate the null space of their robot design. The desired set of motion features is selected by the user from a dropdown menu, and it can consist of the linear or angular motion of the body, the robot's joint angle trajectories, or the paths that its end effectors move along. As before, designers can manually edit the robot's body proportions as they desire. These user-provided morphological edits, $\Delta\mathbf{p}^u$, are complemented by synergistic changes to all other design parameters, $\Delta\mathbf{p}^s$, that are automatically computed such that changes to the motion features, $\Delta\mathbf{m}^f$, are minimal. Noting that $\Delta\mathbf{m}^f = \mathbf{J}^f(\Delta\mathbf{p}^u + \Delta\mathbf{p}_s)$,

where $\mathbf{J}^f = \frac{\partial \mathbf{m}^f}{\partial \mathbf{p}}$, $\Delta \mathbf{p}_s$ is the optimum of the following quadratic program:

$$\frac{1}{2}||\frac{\partial \mathbf{m}^f}{\partial \mathbf{p}}(\Delta \mathbf{p}^u + \Delta \mathbf{p}^s) - \Delta \bar{\mathbf{m}}^f||_2^2 \quad \text{subject to} \quad \Delta \mathbf{p}_i^s = 0, \forall i : \Delta \mathbf{p}_i^u \neq 0 \tag{22}$$

The constraints ensure that the changes to design parameters that are automatically computed are 0 for all components of $\mathbf{p}$ that are user-specified, and $\Delta \bar{\mathbf{m}}^f = \mathbf{m}^f(\mathbf{p}) - \mathbf{m}^f(\mathbf{p}_0)$ helps to combat drift. With the solution to this optimization problem computed, the design parameters are updated as $\mathbf{p} = \mathbf{p} + (\Delta \mathbf{p}^u + \Delta \mathbf{p}^s)$, and the corresponding optimal motion is recomputed. The updated robot design respects the morphological edits specified by the user while minimally changing the selected set of motion features. The process, of course, can repeat as desired.

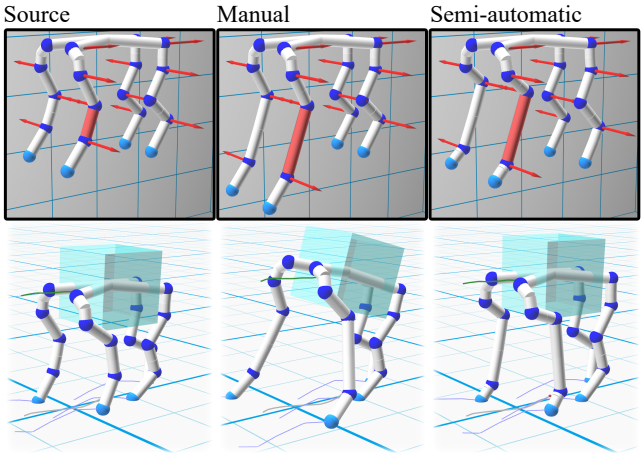


Fig. 4. Comparison of manual and semi-automatic design editing modes. In this example, we increase the length of the highlighted link and apply symmetric edits to the right side of the body. In manual mode, the lengthening of the leg results in a noticeable body pitch – the transparent boxes are added to help better visualize body orientations. In semi-automatic mode, we ask that the orientation of the body does not change. As a result, when the highlighted link is lengthened, synergistic adjustments to other design parameters are automatically applied as well. In both cases, the weights of the objectives governing the motion of the body are exactly the same, so its orientation is only influenced by the robot's design.

Automatic Design Optimization: As a more general solution, our mathematical framework also supports the optimization of designs according to arbitrary functions defined in terms of the robot's motion parameters $\mathbf{m}$. The choreography and functional objectives described in the previous section, as well as the overall energy E minimized during the trajectory optimization process are examples of such functions, which we denote as $O(\mathbf{m})$. Through the chain rule, we can easily compute the gradient of $O(\mathbf{m})$ with respect to the design parameters: $\frac{\partial O}{\partial \mathbf{p}} = \frac{\partial \mathbf{m}}{\partial \mathbf{p}}^T \frac{\partial O}{\partial \mathbf{m}}$. Updating the robot's design parameters $\mathbf{p}$ using a step along this gradient (i.e. $\mathbf{p} = \mathbf{p} + \beta \frac{\partial O}{\partial \mathbf{p}}$) induces the change in the robot's optimal motion that most improves O. We have used this optimization scheme to confirm that the design of the simple car described at the start of this section can be fixed automatically. In this case, $O(\mathbf{m})$ was simply $E(\mathbf{m})$. As we demonstrate in the results video, based on the choreography objectives, the design of the robot and its motions can be concurrently generated in a co-optimization process that is guided by the designer.

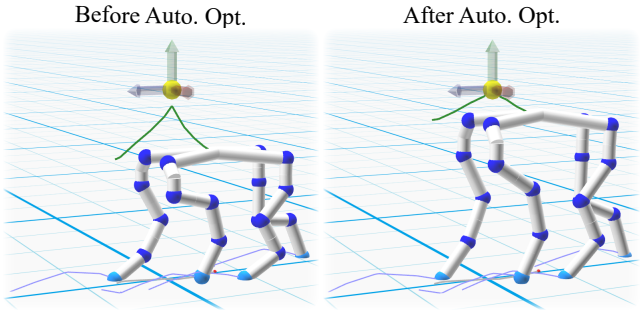


Fig. 5. Result of design optimization. The robot is asked to reach the high target position as shown, but its initial design makes it unable to comply. By optimizing the design parameters, the robot's physical dimensions automatically change such that it can achive the user-specified motion goal.

6 RESULTS

Here we discuss our results, which showcase a variety of robots and corresponding motions generated with our method. Thanks to our interactive design system, all of these results were very easy to create. We emphasize that all motions emerged automatically as a function of the morphological design of each robot (i.e. number of limbs, types of wheels, etc). Users provided only high-level guidance in the form of a desired moving speed, or optionally, in the form of a sparse set of targets for the robot's body over time. The accompanying results video shows real-time screen captures of our design system, including the motion optimization and physics simulation steps.

6.1 Wheeled robots

Actuated wheels. Active wheels are highly successful locomotive devices in their own right, as evidenced by the abundance of vehicles found on our roads. This begs the question: what can be gained by placing motorized wheels on robotic legs? In addition to providing the option to switch to legged locomotion whenever convenient, the extra flexibility afforded by combining legs and wheels enables increased agility. We demonstrate this in the results video by asking one of our robots, Agilebot, to first accelerate and then quickly slow to a halt. In order to not lose balance and topple over, the robot extends its front legs forward to maintain the center of pressure within the support polygon. It is worth noting that the optimization process *discovers* this strategy by itself, without any intervention from the user. We further demonstrate automatically generated step-and-drive and turn-in-place motions that are enabled by the robot's hybrid wheeled/legged design.

Passive wheels. Passive wheels also provide ample opportunities for efficient locomotion, *skating* being the prime example. Skating is an elegant and highly efficient form of human locomotion. Even at high speeds, muscles move slowly and can thus exert a large amount

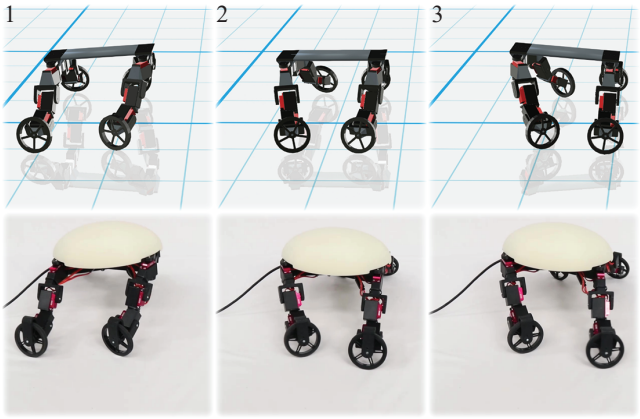


Fig. 6. Swizzle.

of force for propulsion. However, as anyone who has ever tried to roller-skate will attest, mastering this skill requires a high degree of motor coordination, and one that is radically different from other, more native, forms of movement. Unlike walking, skating does not produce ground reaction forces aligned with the desired direction of motion; it is precisely the absence of friction in this direction that makes skating so elegant and efficient. We demonstrate two types of natural skating motions that our optimization automatically generates: swizzling and stroking.

Swizzling is a skating technique where wheels remain in constant contact with the ground. This technique is based on wave-like motions of the feet which exploit the directional friction of the wheels and cause the whole body to be propelled forward. Our motion optimization process discovered swizzling motions automatically for robot designs that feature only one passive wheel per leg, as shown in Fig. 6 and in the accompanying video.

Stroking is a slightly more advanced technique that requires the legs to be lifted off the ground periodically. This is necessary when several wheels are attached to a foot, since this type of design makes it impossible for end effectors to rotate about the vertical axis without sliding while they are touching the ground. This mode of locomotion requires the robot to place its back leg on the ground, orienting the wheels in a direction almost orthogonal to the direction of motion, while the front leg glides forward. A stroke of the back leg pushes the body forward. Next, the legs switch placement and the motion is repeated. Fig. 1(left) depicts a robot that automatically learns to perform this type of movement. We note that to obtain this motion, it is necessary to provide an input gait (e.g. a walk or trot) that then defines the stroking pattern.

Welded wheels. Welded wheels, which are supported by our design system, enable a more general representation of physical robot feet as compared to the widely used point foot model. Wheels can either be welded permanently to a body part of the robot (e.g. they are printed together as one piece), or they can be actuated wheels that are actively blocked. We use the former option to demonstrate a design with three legs that end in welded wheels and one passive wheel, as shown in Fig. 1(middle). The results video includes a real-time demonstration of the motion optimization process for this robot. As can be seen, optimal motions are generated in response to the user changing the desired speed and turning rate.

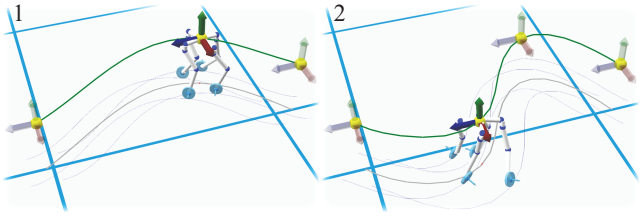


Fig. 7. A demonstration of a slalom motion design. The user can quickly place position objectives and observe the result in real-time.

6.2 Interactive Design

The improved performance of our motion optimization algorithm allows the user to interactively specify motion choreography objectives. A wide variety of motions can therfore be created in a very short time-span, and the user may fine-tune the movements of the robot to any desired degree. One example, shown in Fig. 7, is a slalom motion, which also benefits greatly from the versatility of wheeled legs. For this example, the user simply specifies a desired speed and provides several lateral target positions for the center of mass at different moments in time. The optimal motion is generated almost instantaneously.

6.3 Design optimization

As discussed in the previous section, the three different design editing modes assist users with the non trivial and often unintuitive task of optimizing their robots' physical dimensions. These editing modes are supported by our efficient motion optimization method which allows the user to vary design parameters and observe the change in motion immediately.

Semi-automatic optimization allows the user to keep certain aspects of the motion unchanged while manually tweaking their design, as shown in Fig. 4. Fully automatic design optimization is used both to fix subtle flaws, as discussed in Sec. 5, or to make more drastic changes that enhance a robot's ability to generate the motions envisioned by the designer. Fig. 8 presents convergence plots for both of these use cases. For both plots we show the energy of the optimal motion corresponding to the morphological parameters at each design optimization step. For the car example, shown on the left, the initial design was unable to move forward due to the flaw we identified earlier. After three design optimization steps, the flaw was successfully fixed. The example on the right corresponds to the use case shown in Fig. 5. Here, user-specified choreography objectives asked the robot to move in ways that were incompatible with its initial design. After one design optimization step, this robot could already reach its target, while subsequent iterations further reduced the total energy of its motion.

6.4 Further discussion

Fig. 10 shows a collection of robots designed with our system. For all these robots we also designed a variety of motion plans that were validated in a physically-simulated environment. Table 1 provides statistics regarding the complexity of all the robots shown in the results video. As can be seen, generating each motion takes only a few seconds of compute time.

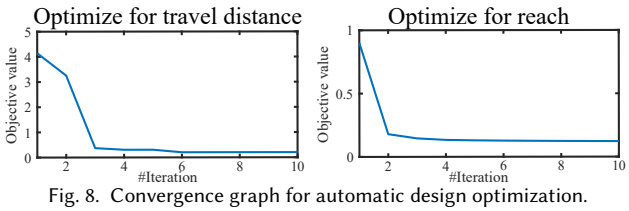


Fig. 8. Convergence graph for automatic design optimization.

We validate our designs through three physical prototypes which are shown in Fig. 1. To create these robotic creatures, we used off-the-shelf micro-sized servo motors (e.g. *Turnigy TGY-306*), the *Maestro* USB controller board from *Pololu*, a standard 7.4V battery and 3D printed connectors (e.g. limb segments). Fig. 9 shows all the components used for the leg of one of our robot prototypes. For each of our designs, the connectors took less than 24h to 3D print on a Stratasys F370 machine. Assembling, calibration and testing took an additional 2-4h.

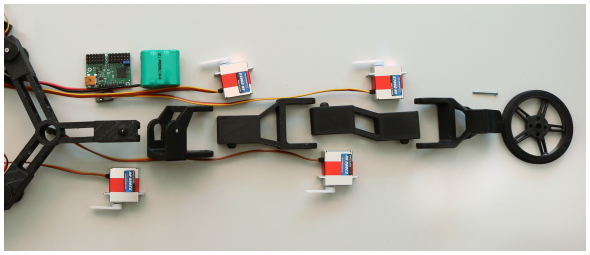
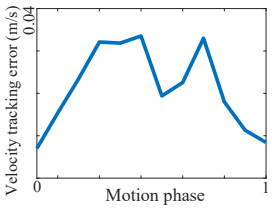


Fig. 9. A robot leg before assembly.

Micro-sized servos are small, light and easy to work with. However, they are also limited in terms of the torque they can produce, which is particularly problematic for robots with long legs, and thus large moment arms. For this reason, while we noticed that our smaller robots were able to reproduce the motions generated through optimization quite well, the *Skatebot* design was noticeably slower as compared to the simulation results. Our initial wheel design, which relied largely on 3D printed parts (e.g. no ball bearings), further contributed to discrepancies between the motions of the physical robot and its simulated counterpart. To quantify the degree to which our robots were able to follow the optimized motion plans, we measured the relative difference in speed for Agilebot as it tracked the slalom motion (Fig. 7). As can be seen in the inset figure, the relative error in the velocity of the COM is less than $4cm/s$ (about 6% of the speed of the motion plan). We note that this tracking error is recorded as the robot operates in a physically-simulated environment, which bypasses mismatches arising from sub-optimal hardware components.



7 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

We presented a novel design system for a rich class of hybrid legged/wheeled robotic creatures. Thanks to our versatile trajectory optimization formulation, physically-valid walking, rolling, gliding and skating motions arise naturally as a function of the design characteristics of different robots. We further showed how to leverage the structure of our formulation to significantly accelerate the underlying numerical solver. This, in turn, allows designers to interactively choreograph compelling motions. Given that motor capabilities and the design of a robot are inseparably intertwined, we also developed a suite of user-guided computational tools that support manual, semi-automatic and fully automatic optimization of the robot's physical dimensions. We demonstrated the effectiveness of our method by creating a variety of unique robot designs, three of which we fabricated.

Our results highlight exciting avenues for future work. For example, we are encouraged by the significant improvements in convergence rates achieved through the Filtered Hessian strategy described in Sec. 4.3. We will continue this line of investigation by analyzing all other objectives and experimenting with different filtering operations. Our ultimate goal here is to achieve faster than real time performance for the trajectory optimization process. This will allow motion plans to be computed on-line, taking into account dynamic environmental obstacles, and capable of providing full-body feedback strategies in response to unplanned disturbances. We also plan to investigate more sophisticated controllers to track the optimized motion plans. The performance of the stiff position-based controllers we currently employ would likely degrade as they attempt to locomote on rougher ground. To this end, there are many techniques developed in the field of physics-based character animation that we can build on. To increase the complexity of the behaviors exhibited by user-designed robots, character animation techniques that appropriately re-sequence motion clips [Hyun et al. 2016; Kovar et al. 2002] show great promise as well.

There are also interesting opportunities to improve our motion optimization model, as well as the fabrication process for the robots designed with our framework. For example, the physical prototype for our SkateBot robot moves noticeably slower than the simulation model. This is not surprising, given that in simulation wheels have no play, nor friction losses, and motors can very accurately track the planned motions. Better engineered physical wheels (i.e. equipped with professional-grade ball bearings rather than 3D printed parts) and higher performance actuators would help close this gap. Likewise, increasing the accuracy of our predictive model to better account for real-world limitations would be beneficial as well. Last, we plan to address the generation of increasingly agile motions. Drifting manouvers, for example, leverage slipping to great effect, which is currently outside the capabilities of our model.

ACKNOWLEDGMENTS

We would like to thank Vittorio Megaro and Jim Bern for their help with physical prototypes and video editing. We are also grateful to the anonymous reviewers for their insightful suggestions.

Robot	# joints	# end-effectors	motion	motion goals	# time samples	# motion parameters	planning horizon	opt. time
Swizzlebot	12	4	swizzle forward	$d = 0.5\text{m}, \alpha = 0°$	12	648	1.2s	4.5s
			swizzle turning	$d = \text{any}, \alpha = 20°$	12	648	1.2s	3.4s
Creature3	9	4	walk	$d = 0.1\text{m}, \alpha = 0°$	12	504	0.8s	1.3s
			fast walk	$d = 0.3\text{m}, \alpha = 0°$	12	504	0.8s	2.5s
			turning	$d = \text{any}, \alpha = 20°$	12	504	0.8s	1.5s
Brainbot	9	3	swizzle forward	$d = 0.2\text{m}, \alpha = 0°$	24	1008	2s	5.7s
			swizzle forward & turn	$d = \text{any}, \alpha = 20°$	12	504	2s	5.9s
Agilebot	16	4	start & stop	$d = 1.0\text{m}, \alpha = 0°$	12	696	0.8s	5.2s
			turn in place	$d = 0.0\text{m}, \alpha = 0°$	12	696	1.6s	1.1s
			slalom	choreography	24	1392	1.6s	-[1]
			side-step and roll	choreography	36	2088	3s	-[1]
Skatebot	16	8	two-leg skating	$d = 0.3\text{m}, \alpha = 0°$	12	696	1.2s	11.3s
Dinobot	17	8	skating	$d = 0.8\text{m}, \alpha = 0°$	12	1140	2s	11.0s

Table 1. With our system, the user can create various robot designs and easily generate motions satisfying different motion goals.In the fifth column, d defines the desired distance travelled, and α the desired turning angle. Column 7 displays the number of motion parameters which equals the size of $\mathbf{m}$.
[1]: Motion is generated at interactive rates, while user edits choreography objectives.

Fig. 10. A variety of unique legged/wheeled robots designed with our system.

REFERENCES

Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. 2012. Fabricating Articulated Characters from Skinned Meshes. *ACM Trans. Graph.* 31, 4, Article 47 (July 2012), 9 pages. https://doi.org/10.1145/2185520.2185543

Moritz Bächer, Benjamin Hepp, Fabrizio Pece, Paul G. Kry, Bernd Bickel, Bernhard Thomaszewski, and Otmar Hilliges. 2016. DefSense: Computational Design of Customized Deformable Input Devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3806–3816. https://doi.org/10.1145/2858036.2858354

Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it: optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 96.

James M. Bern, Kai-Hung Chang, and Stelian Coros. 2017. Interactive Design of Animated Plushies. *ACM Trans. Graph.* 36, 4, Article 80 (July 2017), 11 pages. https://doi.org/10.1145/3072959.3073700

Gaurav Bharaj, Stelian Coros, Bernhard Thomaszewski, James Tompkin, Bernd Bickel, and Hanspeter Pfister. 2015. Computational Design of Walking Automata. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '15)*. ACM, New York, NY, USA, 93–100. https://doi.org/10.1145/2786784.2786803

BostonDynamics. 2017. Handle, `https://www.bostondynamics.com/handle`. (2017).

Jacques Calì, Dan A. Calian, Cristina Amati, Rebecca Kleinberger, Anthony Steed, Jan Kautz, and Tim Weyrich. 2012. 3D-printing of Non-assembly, Articulated Models. *ACM Trans. Graph.* 31, 6, Article 130 (Nov. 2012), 8 pages. https://doi.org/10.1145/2366145.2366149

Duygu Ceylan, Wilmot Li, Niloy J Mitra, Maneesh Agrawala, and Mark Pauly. 2013. Designing and fabricating mechanical automata from mocap sequences. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 186.

Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 83.

Hongkai Dai, Andres Valenzuela, and Russ Tedrake. 2014. Whole-body motion planning with centroidal dynamics and full kinematics. In *Humanoids*.

R. Desai, Y. Yuan, and S. Coros. 2017. Computational Abstractions for Interactive Design of Robotic Devices. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*.

Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational Multicopter Design. *ACM Trans. Graph.* 35, 6, Article 227 (Nov. 2016), 10 pages. https://doi.org/10.1145/2980179.2982427

G. Endo and S. Hirose. 2008. Study on Roller-Walker - Adaptation of characteristics of the propulsion by a leg trajectory -. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1532–1537. https://doi.org/10.1109/IROS.2008.4650823

Sean Follmer, Valkyrie Savage, Jingy Li, and Bjoern Hartmann. 2015. Makers' Marks: Physical Markup for Designing and Fabricating Functional Objects. In *UIST'15 Proceedings of the 28th annual ACM symposium on User interface software and technology*.

Sehoon Ha, Stelian Coros, Alex Alspach, Joohyung Kim, and Katsu Yamane. 2017. Joint Optimization of Robot Design and Motion Parameters using the Implicit Function Theorem. In *Robotics: Science and Systems*. RSS.

Kyunglyul Hyun, Kyungho Lee, and Jehee Lee. 2016. Motion Grammars for Character Animation. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics (EG '16)*. Eurographics Association, Goslar Germany, Germany, 103–113. https://doi.org/10.1111/cgf.12815

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion Graphs. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*. ACM, New York, NY, USA, 473–482. https://doi.org/10.1145/566570.566605

Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. 2014. Build-to-last: strength to weight 3D printed objects. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 97.

Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive Design of 3D-printable Robotic Creatures. *ACM Trans. Graph.* 34, 6, Article 216 (Oct. 2015), 9 pages. https://doi.org/10.1145/2816795.2818137

Vittorio Megaro, Jonas Zehnder, Moritz Bächer, Stelian Coros, Markus Gross, and Bernhard Thomaszewski. 2017. A Computational Design Tool for Compliant Mechanisms. *ACM Trans. Graph.* 36, 4, Article 82 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073636

Przemyslaw Musialski, Thomas Auzinger, Michael Birsak, Michael Wimmer, Leif Kobbelt, and TU Wien. 2015. Reduced-order shape optimization using offset surfaces. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 102.

ODE. 2007. Open Dynamics Engine, `http://www.ode.org/`. (2007).

David E. Orin, Ambarish Goswami, and Sung-Hee Lee. 2013. Centroidal Dynamics of a Humanoid Robot. *Auton. Robots* 35, 2-3 (Oct. 2013), 161–176. https://doi.org/10.1007/s10514-013-9341-4

Jesús Pérez, Miguel A. Otaduy, and Bernhard Thomaszewski. 2017. Computational Design and Automated Fabrication of Kirchhoff-plateau Surfaces. *ACM Trans. Graph.* 36, 4, Article 62 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073695

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make it stand: balancing shapes for 3D fabrication. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 81.

Adriana Schulz, Cynthia Sung, Andrew Spielberg, Wei Zhao, Robin Cheng, Eitan Grinspun, Daniela Rus, and Wojciech Matusik. 2017. Interactive robogami: An end-to-end system for design of robots with ground locomotion. *The International Journal of Robotics Research* 36, 10 (2017), 1131–1147. https://doi.org/10.1177/0278364917723465 arXiv:https://doi.org/10.1177/0278364917723465

J. A. Smith, I. Sharf, and M. Trentini. 2006. PAW: a hybrid wheeled-leg robot. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* 4043–4048. https://doi.org/10.1109/ROBOT.2006.1642323

Peng Song, Xiaofei Wang, Xiao Tang, Chi-Wing Fu, Hongfei Xu, Ligang Liu, and Niloy J. Mitra. 2017. Computational Design of Wind-up Toys. *ACM Trans. Graph.* 36, 6, Article 238 (Nov. 2017), 13 pages. https://doi.org/10.1145/3130800.3130808

Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. 2012. Stress relief: improving structural strength of 3D printable objects. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 48.

Nobuyuki Umentani, Takeo Igarashi, and Niloy J. Mitra. 2015. Guided Exploration of Physically Valid Shapes for Furniture Design. *Commun. ACM* 58, 9 (Aug. 2015), 116–124. https://doi.org/10.1145/2801945

Francisca Gil Ureta, Chelsea Tymms, and Denis Zorin. 2016. Interactive Modeling of Mechanical Objects. In *Proceedings of the Symposium on Geometry Processing (SGP '16)*. Eurographics Association, Goslar Germany, Germany, 145–155. https://doi.org/10.1111/cgf.12971

Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. 2012. .NET gadgeteer: a platform for custom devices. In *Pervasive Computing*. Springer, 216–233.

Christian Weichel, Manfred Lau, and Hans Gellersen. 2013. Enclosed: a component-centric interface for designing prototype enclosures. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*. ACM, 215–218.

Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. 2017. Functionality-aware Retargeting of Mechanisms to 3D Shapes. *ACM Trans. Graph.* 36, 4, Article 81 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073710