Interactive Design of 3D-Printable Robotic Creatures

Vittorio Megaro[1] Bernhard Thomaszewski[2] Maurizio Nitti[2]
Otmar Hilliges[1] Markus Gross[1,2] Stelian Coros[3]

[1]ETH Zürich [2]Disney Research Zurich [3]Carnegie Mellon University

Abstract

We present an interactive design system that allows casual users to quickly create 3D-printable robotic creatures. Our approach automates the tedious parts of the design process while providing ample room for customization of morphology, proportions, gait and motion style. The technical core of our framework is an efficient optimization-based solution that generates stable motions for legged robots of arbitrary designs. An intuitive set of editing tools allows the user to interactively explore the space of feasible designs and to study the relationship between morphological features and the resulting motions. Fabrication blueprints are generated automatically such that the robot designs can be manufactured using 3D-printing and off-the-shelf servo motors. We demonstrate the effectiveness of our solution by designing six robotic creatures with a variety of morphological features: two, four or five legs, point or area feet, actuated spines and different proportions. We validate the feasibility of the designs generated with our system through physics simulations and physically-fabricated prototypes.

CR Categories: I.2.9 [Artificial Intelligence]: Robotics—Kinematics and dynamics; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: physical characters, robotics, 3D-printing.

1 Introduction

The desire to create mechanical creatures that are capable of life-like motions and behaviors dates back to ancient times. However, it was only during the last century that this vision started to become reality. Today, mobile robots are ubiquitous in industry and they start to enter our daily life in the form of electro-mechanical toys, robotic pets, and household assistants. The recent progress in 3D-printing technology and the advent of powerful, simple-to-program hardware platforms like Arduino now open the door to a new generation of *personal robots*—unique companions that we custom-design according to our needs and preferences. Already now, the rapidly growing community of makers and technology enthusiasts indicates that there is significant interest in this topic. Nevertheless, creating compelling robotic creatures is currently a formidable task that only experienced engineers can successfully undertake.

Driven by the progress in rapid manufacturing technology, the graphics community has recently started to embrace the challenge of translating digital characters into physical artifacts [Zhu et al. 2012; Coros et al. 2013; Ceylan et al. 2013; Thomaszewski et al. 2014]. While current methods can create physical characters whose motion closely resemble their digital counterparts, this motion is merely for display; stable locomotion, however, poses complex requirements on the physics and geometry of motion—criteria that digital animations fail to fulfill in general.

Figure 1: *Digital designs (left) and physical prototypes (right) for our* Ranger *(top),* Bobby *(middle) and* Predator *(bottom) designs, fabricated using 3D-printing and off-the-shelf servo motors.*

We present an interactive design system that allows casual users to quickly design 3D-printable robotic creatures. We are inspired by the simplicity, power and flexibility of the character design system by Hecker et al. [2008], empowering novice users to quickly create digital characters with custom shape and motion. Our ambition is to make the design of compelling robotic creatures as accessible and intuitive. A number of challenges have to be overcome in order to achieve this goal. First, as opposed to virtual characters, the motions designed for robotic creatures have to be physically correct; otherwise, the robot will not move as expected, fail to move, or simply fall over. This *physical feasibility* requires a high degree of coordination between the motions of different body parts, which is difficult to achieve with traditional animation approaches. Second, digital characters often exhibit a large number of degrees of freedom in order to allow for highly expressive animations. When creating robotic creatures, however, a much more careful balance between complexity of design, and therefore cost, and range of achievable motions is required.

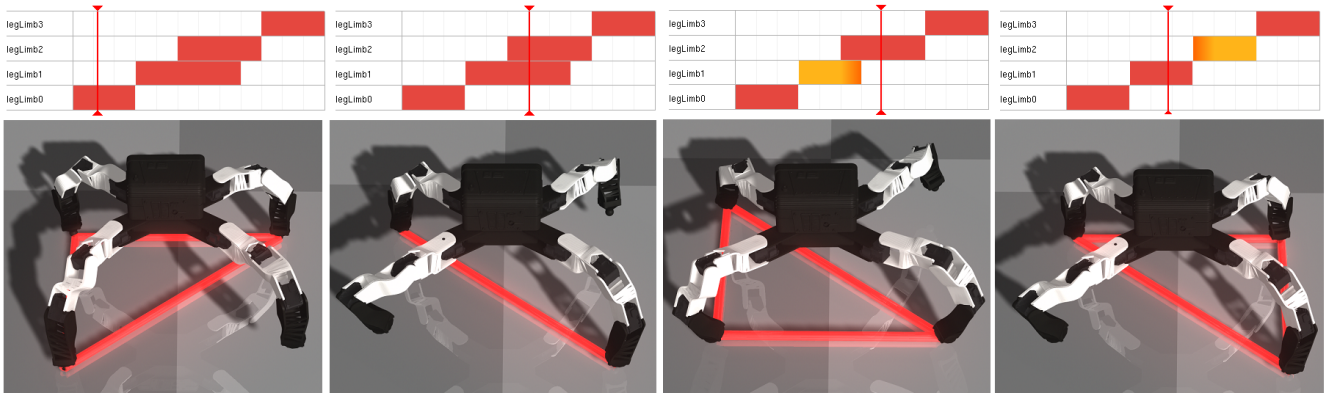


Figure 2: *The footfall pattern indicates which leg is in stance (white) or in swing (red) mode. In this example, the user interactively adjusts the footfall pattern such that three legs are always in stance mode.*

Overview & Contributions To address the challenges outlined above, we propose a *forward design* approach that automates the tedious parts of the design process while providing ample room for creativity and personalization. The core contribution of our method is a fast optimization-based solution that generates stable motions for legged robots of arbitrary designs. To warrant feedback at interactive rates, we depart from conventional space-time approaches, leveraging a geometric view of trajectory optimization. Tightly integrated with the optimization, an intuitive set of interactive tools allows the user to design and edit the morphology, proportions, gait and motion style of a robotic creature. In order to translate the digital designs to the physical world, we automatically generate geometry for the mechanical structure of the robot such that it can be fabricated using 3D-printing and off-the-shelf servo motors.

2 Related Work

Fabrication-Oriented Design Fueled by advances in rapid manufacturing technologies, the graphics community has seen a growing number of works that propose computational approaches for generating physical artifacts. Examples include design systems for physically-valid furniture pieces [Lau et al. 2011; Umetani et al. 2012], objects whose mass distribution can be precisely controlled such that they are able to stand [Prévost et al. 2013] or spin stably [Bächer et al. 2014], free-form gliders with optimized aerodynamic properties [Umetani et al. 2014] and prototypes that test and validate the functionality of finished products [Koo et al. 2014]. Our computational design framework shares the same high-level goal as these works: empowering casual users in creating complex physical artifacts without requiring domain specific knowledge. More specifically, we address the challenge of designing 3D printable robotic creatures whose morphological features, behaviors and motion styles can be easily personalized.

Physical Character Design Within the field of fabrication-oriented design, our work is most closely related to previous methods for creating physical representations of virtual characters. For instance, the design systems proposed by Bächer et al. [2012] and Calì et al. [2012] can be used to design 3D printable characters with functional joints. Skouras et al. [2013] proposed a method to automatically optimize the internal distribution of material parameters in order to control the ways in which 3D printed characters deform when subjected to external forces. Several methods that create mechanical automata from animated digital characters have also been proposed [Zhu et al. 2012; Coros et al. 2013; Ceylan et al. 2013; Thomaszewski et al. 2014]. These methods focus on the design of passive mechanical structures, such as linkages and gears, that propagate the motion of one input motor to the entire system. Given the limited actuation of these mechanical characters, they can only perform periodic motions. We remove this restriction by designing robotic creatures that can generate a variety of compelling motions and stable behaviors.

Robotics Our work is also closely related to the field of robotics, where a long-standing goal is to automate the design of robotic systems based on high-level functional specifications. Inspired by Sims' work on evolving virtual creatures [1994], a variety of evolutionary methods that aim to co-design a robot's structure and control inputs have been investigated [Leger 1999; Lipson and Pollack 2000], and this effort continues today [Auerbach et al. 2014]. Rather than relying on stochastic algorithms, our system puts the user in the loop with a set of easy-to-use editing tools that provide control over the creative process. Sharing a goal similar to ours, several methods that allow users to create origami-inspired, foldable robots have been proposed recently [Mehta and Rus 2014; Mehta et al. 2014]. However, while this body of work relies on a set of pre-defined building blocks that are linked together using a custom scripting language, our design system allows users to freely explore a vast space of legged robot designs: two or more legs, point or area feet and articulated spines are all handled in a unified manner by our framework.

Motion Generation To generate motions for our physical creatures we draw inspiration from the rich literature on this topic from both computer animation and robotics. In particular, the motions we generate are similar to those used by sophisticated robots like Boston Dynamic's *Little Dog* [Neuhaus et al. 2011]. However, while control solutions reported in the literature are usually linked intimately to the robots that they are designed for, we rely on a fully automated approach to generate optimal walking motions for creatures of arbitrary morphologies. The trajectory optimization method we use for this purpose is inspired by a well-known class of animation methods based on space-time constraints [Witkin and Kass 1988] which continue to be actively investigated to date [Wampler and Popović 2009a; Mordatch et al. 2012]. However, motivated by the need to develop a computationally efficient system that supports interactive design, as well as by our goal of creating robotic creatures capable of walking stably using only commodity hardware, we develop a model that presents important differences, as will become clear throughout the remainder of the paper. Briefly,

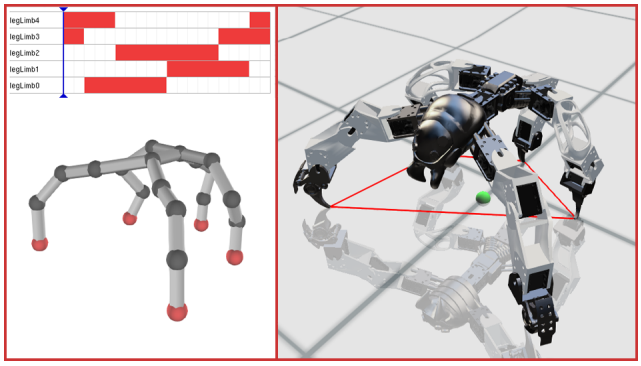


Figure 3: *Snapshot of the design interface.* Left*: the design viewport with the footfall pattern graph.* Right*: the preview window showing the center of pressure of the robot (green) and the support polygon (red).*

rather than modeling full system dynamics and assuming idealized joint torque actuators, we directly use joint angles as control variables and explicitly enforce a dynamic stability objective.

3 Method Overview

We propose an end-to-end solution for creating robotic creatures, implemented as an interactive tool that allows the user to design the structure and motion of a robot while receiving immediate feedback on its expected real-world behavior.

Design Interface Our design interface is structured into two viewports (see Fig. 3): one for editing the structure and motion of the robot, one for displaying the resulting real-world behavior as predicted by our optimization or through physics simulation. The heart of the interface is formed by a set of easy-to-use editing tools.

Structure Editing The user starts by loading a description file that specifies an initial skeletal structure of the robot, as defined through a typical hierarchy of bones connected by joints. Initial geometry is created from this information and a virtual, uni-axial motor is placed at each joint position. The user can freely edit the robot's structure at all times by adding or removing motors, thus altering the morphology of the design, or by adjusting the position or orientation of the motors.

Motion Editing The motion of a robotic creature is completely described by the trajectories of its joints. However, authoring motions directly in this high-dimensional space is unintuitive, tedious, and very unlikely to lead to stable movements. We therefore propose a set of higher-level motion authoring and editing tools, designed to be mutually orthogonal and intuitive for the user.

Motions are largely characterized by their footfall pattern, indicating for each instant during a gait cycle which of the legs are in contact with the ground (stance mode) and which ones are in flight (swing mode). Our interface displays these information as a time-dependent graph, allowing for quick inspection and direct editing by the user (see Fig. 2). In particular, the user can change the duration of the stance and swing phases for any leg and change the relative ordering of the footfalls. While not all patterns lead to desirable motions, this is immediately clear upon inspecting the evolution of the support polygon through time or by observing the motions generated by our framework. The immediate feedback provided by our framework allows the user to interactively adjust the footfall pattern in order to find satisfying solutions.

Higher-level goals such as the walking direction, speed or turning rate can be provided by the user so as to specify the behavior of the robotic creatures that are being designed. Further, the user can control the overall motion style by editing the movement of the robot's center of mass and of the feet trajectories. The motions generated by our optimization-based framework are guided by this user input, while a set of feasibility constraints ensures that they are always stable.

Optimization Given the structure and motion goals for a robotic creature, our system computes time-varying motor values for dynamically-stable motions using a trajectory optimization approach. The user can preview the optimized motions using physics-based simulation and iteratively adapt the design to explore the solution space and converge on a desired result. In order to enable this seamless forward design experience, to robustly support a wide variety of morphological features in a unified manner, and to ensure that the resulting robotic creatures function well using off-the-shelf components, the model that we propose requires a departure from conventional approaches.

Space-time optimization methods that consider the full dynamics of the systems they compute motions for are most general. However, the complexity of the underlying models brings about a significant computational overhead—even for simple creatures, generating motions is a matter of several minutes [Wampler and Popović 2009b; Mordatch et al. 2012; Wampler et al. 2014], thus clearly prohibiting the type of interactive design process we seek to enable. Furthermore, space-time methods are notoriously characterized by challenging optimization landscapes that often lead to undesirable local minima. Our model avoids the complexity of considering full system dynamics for three main reasons. First, we achieve important gains in efficiency, with the process of optimizing motions taking at most a few seconds. Second, in conjunction with the optimization scheme we employ, our system consistently converges to high-quality solutions. Last, because the internal torques and ground reaction forces computed through space-time optimization cannot be directly reproduced by off-the-shelf servomotors, and as physical actuators present considerable limitations in terms of speed, bandwidth and strength, we focus on generating motions that are more conservative (e.g. no flight phases). In this setting, the dynamic interplay between the instantaneous center of pressure and the motion of the center of mass is captured sufficiently well through an inverted pendulum approximation.

The simplified dynamics model we use is adopted from robotics, where it is commonly used for *model predictive control* (MPC). MPC formulations typically decouple the generation of center of mass trajectories, motion of the feet and full-body joint angles [Kajita et al. 2003; Dimitrov et al. 2008; Mastalli et al. 2015]. When the morphology and proportions of a robot are fixed, this strategy is typically sufficient. However, various heuristics are required to ensure that constraints between different modules are satisfied, e.g., stepping locations can be reached given the center of mass trajectory and robot kinematics. Given that our design system allows users to generate robot designs with a vast range of morphological features, such a decoupling is not feasible. Rather, as detailed in Sec. 4, our trajectory optimization method provides an efficient, unified formulation for concurrently computing trajectories for the center of mass and feet that are consistent with the structure and range of motion of the robotic creatures.

Finishing Once the design iterations have converged, we automatically generate 3D geometry for all body parts, including con-

nectors for the motors, which are then sent to a 3D printer for manufacturing (see Sec. 5).

4 Motion Plan Generation

Given a robot morphology which includes an arbitrary number of legs with point or area feet, and possibly an actuated spine, our goal is to compute time-varying motor values that lead to user-controllable, stable motions. We formulate this task as a trajectory optimization problem whereby a set of objectives is used to define an optimal motion. We represent a *motion plan* $\mathbf{P} = (\mathbf{P}_1, \ldots, \mathbf{P}_T)$ as a time-indexed sequence of vectors

$$\mathbf{P}_i = (\mathbf{q}_i, \mathbf{x}_i, c_i^1, \ldots, c_i^n, \mathbf{e}_i^1, \ldots, \mathbf{e}_i^n\, w_i^1, \ldots, w_i^n)\,, \qquad (1)$$

where $\mathbf{q}_i$ represents the pose of the creature, i.e., the position and orientation of the root as well as the angle values for all motors, and $\mathbf{x}_i$ denotes the desired position of the creature's center of mass. The feet of each limb are defined by one or multiple end effectors. For each end effector $j, 1 \leq j \leq n$, we use a contact flag c_i^j to indicate whether it should be grounded ($c_i^j = 1$) or not ($c_i^j = 0$) at a given time t_i. Furthermore, we denote the desired position of the end effectors as $\mathbf{e}_i^j$ and store a scalar weight w_i^j for each of them. Fig. 4 (left) visualizes our motion plan representation.

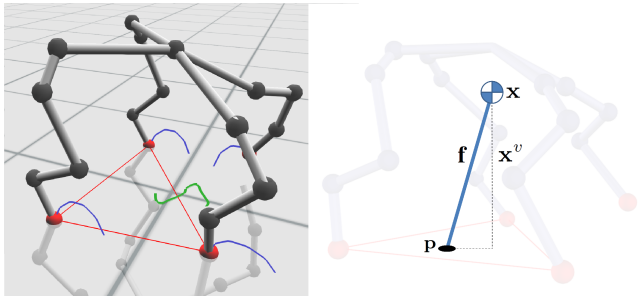


Figure 4: *The motion plans generated by our framework consist of trajectories for the center of mass (green), feet (blue), and corresponding full-body poses for the robotic creature (left). An inverted pendulum model is employed to obtain a relationship between the center of pressure and the center of mass (right).*

4.1 Constraints

We refer to a motion plan as *consistent* if, for every time sample i, the following set of conditions is satisfied:

$$\varphi_{\mathrm{CoM}}(\mathbf{q}_i) - \mathbf{x}_i = \mathbf{0}\,, \qquad (2)$$

$$\varphi_{\mathrm{EE}}(\mathbf{q}_i)^j - \mathbf{e}_i^j = \mathbf{0}\,, \ \forall j\,. \qquad (3)$$

Here, $\varphi_{\mathrm{CoM}}(\mathbf{q})$ is a forward kinematics function outputting the position of the creature's center of mass (COM) given pose $\mathbf{q}$. Similarly, the function $\varphi_{\mathrm{EE}}(\mathbf{q})$ computes the end effector positions for the robot's limbs given pose $\mathbf{q}$.

In order to generate motions that do not require sophisticated sensors and complex feedback mechanisms, our framework computes motion plans that are naturally stable. Formally, this criterion is expressed as a constraint that ensures that at every discrete moment in time i, the *center of pressure* (COP) $\mathbf{p}_i$ falls within the support polygon defined by the end effectors that are in contact with the ground:

$$\sum_{j=1}^{n} c_i^j w_i^j \mathbf{e}_i^j - \mathbf{p}_i = 0\,, \qquad (4)$$

with the additional constraint that only convex combinations of grounded end effector positions are allowed to define the location of the COP:

$$\sum_{j=1}^{n} w_i^j c_i^j = 1\,, \quad w_{lb} \leq w_i^j \leq 1\,, \forall j\,, \qquad (5)$$

where the lower bound limit w_{lb}, which is set to 0.1 for all our experiments, prevents the COP from moving too close to the boundary of the support region.

The COP position $\mathbf{p}_i$ at each time step i is not an explicit parameter of the motion plan. However, using an inverted pendulum model, a simple relationship between it and the optimized COM trajectory can be readily obtained [Kajita et al. 2003]. As illustrated in Fig. 4 (right), the vertical component of the force $\mathbf{f}$ applied along the vector between the COM and the COP is $\mathbf{f}^v = mg + m\ddot{\mathbf{x}}^v$, where $g = 9.8m\backslash s^2$. Consequently, by computing the horizontal component of $\mathbf{f}$, and by observing the trigonometric relationship between the height of the COM, $\mathbf{x}^v$, and the horizontal projection of the vector from $\mathbf{p}$ to $\mathbf{x}$, the following relationship emerges:

$$\mathbf{p}_i = \mathbf{x}_i - \frac{\mathbf{x}_i^v \ddot{x}_i}{\ddot{\mathbf{x}}_i^v + g} \qquad (6)$$

The acceleration of the COM trajectory, $\ddot{\mathbf{x}}_i$, including its vertical component, is expressed using finite differences as a function of $\mathbf{x}_{i-1}$, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$: $\ddot{\mathbf{x}}_i = (\mathbf{x}_{i-1} - 2\mathbf{x}_i + \mathbf{x}_{i+1})/h^2$, where h is the time step. We note that Eq. (4) represents a *dynamic stability* criterion: while the COP is guaranteed to lie within the support polygon at all times, the projection of the COM on the ground plane does not have to. Consequently, the motions generated by our optimization framework are less conservative than if a static stability criterion acting solely on the COM was employed.

As the positions of the end effectors are independently defined at discrete moments in time, an additional constraint is needed in order to ensure temporal consistency of the motion plan and to avoid foot-slipping:

$$(\mathbf{e}_{i-1}^j - \mathbf{e}_i^j)c_i^j = \mathbf{0}\,, (\mathbf{e}_i^j - \mathbf{e}_{i+1}^j)c_i^j = \mathbf{0}\,, \qquad (7)$$

for all $2 \leq i \leq T-1$. This expression implies that the target positions of the end effectors are only allowed to change freely when they are not in contact with the ground.

If a periodic motion is desirable, an additional constraint that relates the robot's joint angles, $J(\mathbf{q})$, at the start and end of the motion is added:

$$J(\mathbf{q}_1) - J(\mathbf{q}_T) = \mathbf{0}\,. \qquad (8)$$

The robotic creatures generated with our system reproduce the planned motions by directly tracking the optimized joint angle trajectories. It is therefore crucial to ensure that the resulting motion plans fall within the range of capabilities of the physical motors that are used. As a simplified actuation model, we place bounds on the maximum angular velocity at each joint j:

$$-\omega_{max} \leq \frac{J(\mathbf{q}_{i+1})^j - J(\mathbf{q}_i)^j}{h} \leq \omega_{max}, \forall i, \qquad (9)$$

where ω_{max} is the maximum achievable speed of the motor.

4.2 Motion Style Objectives

If Eqs. (2-8) are satisfied, the motion plan is termed *admissible*, as it corresponds to a stable motion. In general, many such motion plans exist for a given robot morphology and footfall pattern. We therefore provide several objectives and high-level controls that allow users to intuitively explore the space of admissible motions.

The smoothness of the motion is the first attribute that users can control via an objective defined through a finite difference approximation of the second derivatives of the robot pose trajectory:

$$E_{\text{Smooth}} = \frac{1}{2}\sum_{i}^{T} ||\mathbf{q}_{i-1} - 2\mathbf{q}_i + \mathbf{q}_{i+1}||^2 \,. \tag{10}$$

With the motion editing interface provided by our system, users can also directly influence the motion style of the robotic creatures they design. To this end, two regularizing terms provide target trajectories for the motion of the COM and the end effectors:

$$E_{\text{StyleCOM}} = \frac{1}{2}\sum_{i=1}^{T} ||\mathbf{x}_i - \mathbf{x}_i^D||^2 \tag{11}$$

$$E_{\text{StyleEE}} = \frac{1}{2}\sum_{i=1}^{T}\sum_{j=1}^{n} ||\mathbf{e}_i^j - \mathbf{e}^{D}{}_i^j||^2 \tag{12}$$

We note that although the target trajectories may lead to unstable motions if employed directly, they are used just as a guide. Consequently, the user can edit them freely, without needing to worry about feasibility constraints, in order to uncover the range of motion styles achievable by the robot they are designing.

The walking and turning speed of the robotic creatures are controlled through two separate terms which measure the difference between the pose of the robot at the start and end of the motion trajectory:

$$\mathbf{x}_T - \mathbf{x}_1 = \mathbf{d}^D \tag{13}$$

$$\tau(\mathbf{q}_T) - \tau(\mathbf{q}_1) = \tau^D \tag{14}$$

where $\mathbf{d}^D$ and τ^D are the desired values for the net distance traveled and turning angle, respectively, and the function $\tau(\mathbf{q})$ returns the turning angle from pose $\mathbf{q}$.

4.3 Optimization

We cast the problem of generating feasible motion plans as a multi-objective, constrained optimization problem where the degrees of freedom are the robot poses at each time instance, the trajectories of the end effectors and the center of mass, and the end effector weights which define the trajectory of the center of pressure. The contact flags are directly specified by the foot-fall pattern and are thus not treated as free parameters during the optimization. As described above, we have structured the conditions on the robot's motion into constraints that model vital requirements for successful locomotion, and objectives that influence the style of the motion. When translating these terms into an optimization problem however, we have to carefully balance the importance of exact constraint satisfaction against the numerical difficulties associated with non-linear, non-convex systems of equations. We therefore treat the nonlinear Eqs. (2-4) as soft constraints by minimizing their squared residuals weighted by a large constant (10^4 for all experiments). The linear equality and inequality relations described by Eqs. (5), (7), (8) and (9) are treated as hard constraints. The weights associated with the motion style objectives (10-14) can be interactively set by the users of our system to emphasize different priorities they might have. However, they are kept constant for all our experiments. To minimize the resulting constrained optimization problem we use OOQP [Gertz and Wright 2003], as the inner loop of our Sequential Quadratic Programming solver.

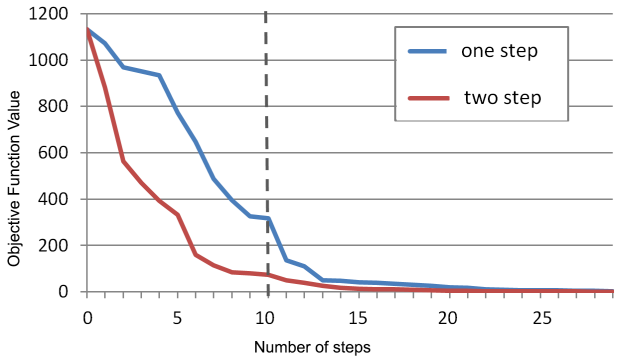


Figure 5: *Convergence plot showing the value of the objective function while optimizing all parameters at once (blue) versus a two-step optimization scheme (red).*

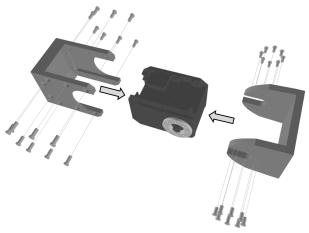
As can be seen in Table 1, generating motion plans requires our framework to solve non-linear systems of equations with hundreds of unknowns—the exact number depends on the complexity of the robot's design and it is linear in the number of time samples that define the motion trajectories. Given that analytic derivatives for the constraints and objectives we formulate can be readily computed, and because the resulting Hessians are sparse, computing optimal motions can be done efficiently, typically requiring less than $3-5s$ of computation time when starting from scratch. During the iterative design process, as the user adjusts the proportions or motion style of their robot, previously computed motion plans are used to warm-start the optimization process, thus further reducing the computational burden and leading to a seamless interactive experience.

To increase convergence rates when optimizing a motion plan from scratch we employ a two-step optimization process. As illustrated in Fig 5, which shows the value of the objective while generating in-place walking motions for our *Ranger* robot, 10 iterations are first executed while the end effector trajectories are kept constant (546 parameters in total), followed by additional iterations where the full set of parameters (714) is optimized for. In contrast to a typical scheme where all parameters are optimized from the beginning, we observe improvements in convergence rates of up to 50%. This performance improvement is due to Eq. (4) becoming convex, which gives rise to a smoother optimization landscape where intermediate solutions can be efficiently arrived at. These intermediate solutions then furnish a good starting point for the global optimization which quickly convergences to a nearby local minimum. Each step of the global optimization takes on average $0.08s$.

5 Generation of 3D Printable Mechanical Structures

The designs thus far produced by our system provide only an abstract description of the robotic creatures our users intend to create. In particular, the placement and orientation of virtual motors, the relative location of the end effectors and the skeletal structure that specifies their connectivity define the morphol-

ogy and body proportions of the designs. Before proceeding to fabrication, our system automatically generates 3D printable geometry for the robot's mechanical structure. To this end, our system starts with a CAD model of available servomotors, as visualized in the inset figure. In a pre-processing step we generate two tight-fitting 3D models that directly attach to the servomotors with screws or rivets, enclose them, and become part of the geometry generated by our system. The geometry of the enclosing models allows for a ± 90 degrees range of motion for each servomotor.



We collectively refer to the attachment parts of each servomotor and the models that define the geometry of the end effectors as *structural features*. The problem of creating the mechanical design a robot's body parts thereby reduces to generating geometric models that connect consecutive pairs of structural features sf_i and sf_j. We note that mounting brackets are typically used for this purpose. However, as the shape and functionality of mounting brackets are pre-defined such that they can be mass-produced, they significantly restrict the set of possible relative positions and orientations between pairs of structural features. We therefore opt to create custom, 3D printable connections instead.

As illustrated in Fig. 6 a), each structural feature outputs a set of possible attachment points (green). As a step towards generating a geometric model that connects a pair of structural features, our system computes a mapping between the attachment points output by sf_i and sf_j. More specifically, we first compute the convex polygons of the attachment points projected on a plane perpendicular to the vector between sf_i and sf_j. We then compute the convex hull of the attachment points whose projections define the two convex polygons, as seen in Fig. 6 b). If fabricated using a *Fused Filament Fabrication* device, where the infill rate can be used to control the trade-off between weight and structural integrity, then the geometry of the body part can be obtained by performing a union operation between the computed convex hull and the models that enclose the servomotors. However, for other 3D printing techniques, such as *Selective Laser Sintering*, our system can automatically create a lightweight structure inspired by engineering principles. More specifically, our system generates truss structures directly from the convex hull computed at the previous step. The edges of the convex hull that connect one attachment point on sf_i to another on sf_j define the main elements of the structure, and additional connective struts are added procedurally, as shown in Fig. 6 c). The density of the connective struts and their thicknesses are user-specified, as they depend on the material used for fabrication.

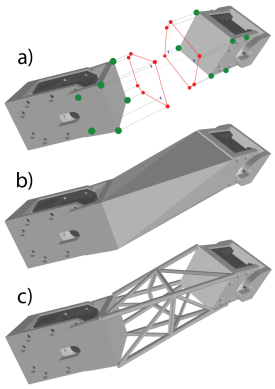


Figure 6: *Generating 3D-printable geometry.*

The geometric structures automatically generated by our framework are functional and they allow the robotic creatures designed with our system to be fabricated through 3D printing. However, in order to enhance the aesthetics of the designs, we allow users to augment the generated geometry with existing 3D models, if desired. To accomplish this, users position the existing 3D models relative to the desired robot body part, and a union operation is performed to generate the fused mesh. As a simplification, the added 3D models are assumed to not significantly alter the mass distribution of the robotic creature (i.e., they are lightweight shells). If this assumption is not valid, it is trivial to recompute the mass and moment of inertia of each body part based on the user-provided geometry and re-optimize the motion plans.

6 Results and Discussion

We have used our interface to design a diverse set of robotic creatures, two of which were physically fabricated for validation. The results that we present in this section demonstrate that our method is indeed a powerful tool, allowing users to author a broad range of designs with explicit and intuitive control over morphology and motion style. Our design methodology offers a number of key benefits over the alternative of manual design. In particular, while keyframing and other conventional methods are very successful for digital animation, the motions of our robotic creatures are subject to real-world physics. Anticipating and incorporating these effects during the design process is very difficult, even for experts. The gaits generated using our motion optimization, in contrast, precisely coordinate the movements of the feet and the body such as to ensure smoothness and stability for robots of various designs. Consequently, our system allows for an intuitive exploration of the relationship between a robot's morphological features and its ability to produce compelling, purposeful motions.

Below we highlight several aspects that are key to our approach and discuss observations from experimental validation.

6.1 Design Interface & Workflow

Structure Editing Thanks to the fast turnaround rates of the underlying optimization, our interface allows for quick, easy, and intuitive editing of a creature's structure and motion. As shown in the accompanying video, the user can freely edit the morphology of the robot by dragging on motor handles until the predicted motion—always visible and up-to-date in the preview viewport—is satisfying. For these edits in particular, convergence is very good since we can warm-start the optimization with a previously optimized motion plan. However, when changing axes of rotation, the joint-angle trajectories used for warm-starting can lead to drastically different trajectories of the end effectors. We therefore simply re-initialize the optimization process, which then takes about 3 seconds on average to arrive at a solution. We also restart the optimization process from scratch when the morphology of a design is altered by adding or removing motors.

Morphology Editing A particularly compelling feature of our method is that it supports arbitrary morphologies. As demonstrated by our results, we support creatures with arbitrary numbers of legs, including bipeds, quadrupeds, and more exotic cases such as the five-legged creature shown in Fig. 3. Furthermore, the configuration of the legs is very flexible: the robotic creatures we show are designed with three or four motors per leg, while our biped robot (Fig. 7) also features actuated ankles for a total of five motors for each leg. Area or point feet are specified by designating one or more end effectors for each limb. Actuated bodies are also supported: our *Ranger* (Fig. 1) and *Predator* (Fig. 3) robots each have one motor that allows their shoulders to rotate relative to their pelvis, while the *Salamander* design (Fig. 8) uses two actuators for the tail and five for its flexible spine. It is worth noting that this generality is a direct consequence of our formulation—no special treatment is required.

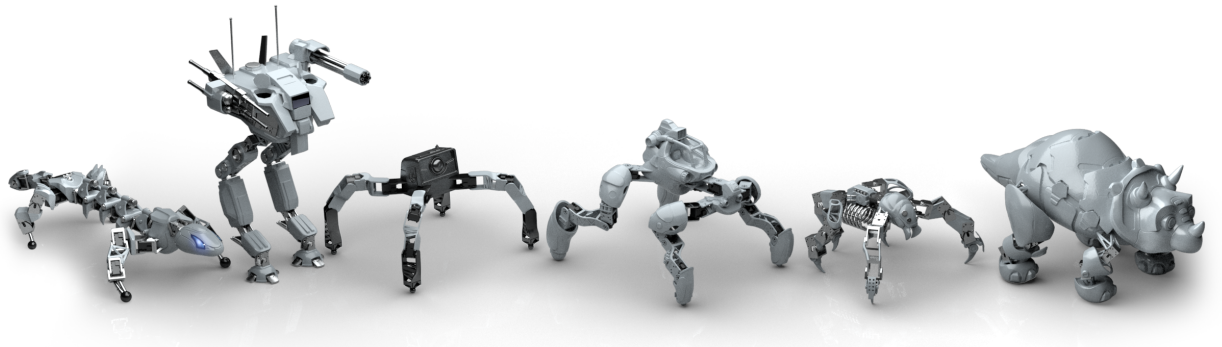


Figure 7: *Six robotic creatures designed with our interactive system: one biped, four quadrupeds and one five-legged robot.*

Motion Editing Our system offers three sets of tools that allow the user to author and edit the motion of a given robot in largely orthogonal dimensions: the footfall pattern, the velocity of the center of mass, and trajectories for the feet and the center of mass. The accompanying video illustrates these tools through a range of motion edits on the *Bobby* model (Fig. 2), a quadruped robot with 12 motors and thus similar in complexity to commercially-available mid-range robots. The footfall pattern graph allows the user to quickly explore different gaits, simply by dragging on the stance/swing phase widgets of the individual legs. As the video shows, the different gaits are also well-reflected by the physical prototype. High-level motion goals are formulated simply by prescribing constant linear or angular velocity for the body of the robot to achieve forward or sideway motion, or to turn in place. Once a rough motion has been laid out, the trajectory editing tool can be used to further flesh out the characteristics of the gait such as to give it personality and style.

Creature Finishing Our method automatically generates geometry for the various body parts based on the shape and location of the motors and end-effectors. However, in order to increase the aesthetic appeal of the final designs, the user can replace these meshes or augment them with manually designed shells. In that case, we assume that the mass and inertia properties for the new parts do not significantly deviate from the old geometry such that the robot can walk as expected with the original animation. We note that it is always possible to recompute angle trajectories in order to account for changed mass properties, but if these changes are significant, the optimization might have to substantial alter the style of the resulting motion.

Figure 8: Salamander*: our framework automatically generates swaying tail and spine motion, leading to a natural-looking gait.*

On-Board Control After the robotic creatures are fabricated and assembled, off-the-shelf servos are used to drive their motions. We use a combination of position and velocity control to ensure that the servo motors produce smooth motions and remain in sync with each other. Control signals are computed at fixed time intervals. Briefly, for motor i at time t we estimate the target joint position $q^i(t+\delta t)$ by interpolating the corresponding optimized motion trajectory, and read the servo's current position, $\alpha^i(t)$. The control board then sets the motor's maximum angular speed to $(q^i(t+\delta t) - \alpha^i(t))/\delta t$, while its goal position is set to $q^i(t+\delta t)$. We use $\delta t = 0.15s$ for all our experiments.

6.2 Validation

We performed a set of experiments in order to assess the feasibility of the motion plans generated by our method. First, we employ black-box physics simulations as a way of providing a preview of the expected real-world behavior of each robot design. We use the Open Dynamics Engine [ODE 2007] for this purpose and model the robots as articulated rigid body systems. The joint trajectories computed during the motion optimization stage are used to define time-varying targets for Proportional-Derivative controllers, used to model the actuators at each joint.

In order to achieve motion planning at interactive rates, our optimization scheme uses an approximate dynamics model. More concretely, asking the center of pressure to fall within the support polygon is a necessary but not sufficient condition, as it ignores the friction-induced limitation on the tangential contact forces and dynamic effects that may become significant for fast limb movements. In order to assess the impact of this simplification, we measured the error in tracking the planned center of mass trajectories over a full motion cycle for our *Ranger* model. We ran this experiment on three different motions with stride durations of 0.5, 1 and 2 seconds, and observed a net final error in center of mass position of 1.57, 0.62 and $0.18cm$ respectively, when comparing the planned motion against the result of the simulation. As a point of reference, each limb of this robotic creature is about $50cm$ in length and it was tasked with walking forward using step lengths of $20cm$. Although a growing discrepancy from the motion plan becomes apparent as the speed of the motion increases, the simulated robot was able to walk successfully each time. However, for biped designs like *Hunter*, such errors can lead to failures due to the high COM and comparatively small area of the support polygon. Physics simulations will immediately reveal such unwanted behaviors, allowing the designer to take appropriate measures by adjusting their design.

To further validate the results of our system, we created physical prototypes for the *Bobby* and the *Ranger* character. We used 3D-printed body parts and off-the-shelf hardware—12 *Dynamixel*

MX-28 actuators daisy-chained to a *CM-700 Robotis Servo Controller* board and powered by a LiPo battery. Even before comparing the motion predicted in simulation to the results observed on the real-world protoypes, we can identify several sources of inaccuracy resulting from idealizing assumptions made by the simulation model. In particular, we assume ideal actuators and perfectly rigid body parts. In reality, the amount of torque that motor can exert is limited, but the maximum torque also decreases with increasing angular velocity—a complex behavior that is more difficult to model. Furthermore, while 3D-printing allows for quick and easy customization of the robot's geometry, the limited accuracy of current consumer-end printers together with the finite compliance of the printed body parts leads to deformations that are not accounted for in the simulation. As a concrete manifestation, we observed that the feet of the *Ranger* model do not rise as high as seen in simulation (approximately $8cm$ measured at the apex vs. $10cm$ in simulation). Despite these sources of error, we observed good agreement between the overall motions of our physical prototypes and the behavior predicted in simulation.

Finally, it should be noted that it takes on the order of minutes to the design these creatures, but hours to assemble and even days to print. This fact implies that building prototypes is very time-consuming and expensive—and it is the ambition of our method to produce final *digital* designs without the need for *physical* iterations.

7 Limitations and Future Work

We presented an interactive, end-to-end solution for designing 3D-printable robotic creatures whose morphology, proportions, gaits and motion styles can be easily personalized. Using our system, we were able to design a diverse set of legged robots, each created in a matter of minutes. Our method efficiently generates stable, user-controllable walking motions for robots with a vast array of morphological features. The most immediate benefit of our system is therefore that it enables an interactive exploration of the space of feasible robot designs without requiring any domain specific knowledge from its users.

Although the space of robots and motions that our solution can currently generate is substantial, our system does have limitations that present exciting avenues for future work. For example, motions that exhibit flight phases cannot currently be produced, as our model requires the existence of at least one point of contact between the robot and the ground at any moment in time. Furthermore, the model simplifications that we exploit for efficiency result in differences between the predicted and real-world behavior of the robots. These differences are most pronounced for fast motions and increase as the size of the support polygon decreases relative to the height of the center of mass. For instance, the bipedal robot that we designed is capable of walking, but unsurprisingly, it can fall quite easily when perturbed. This limitation highlights the need to integrate feedback mechanisms that adapt the motion plan in real-time based on sensor data.

	# Legs	Spine DOFs	# motors	# motion plan parameters
Bobby	4	0	12	429
Dino	4	0	14	735
Salamander	4	8	20	861
Ranger	4	1	13	714
Hunter	2	0	10	387
Predator	5	1	16	840

Table 1: *An overview of the complexity of robot designs we created with our system.*

We are also excited about the challenge of making the process of authoring complex behaviors easily accessible to casual users. We are encouraged by the ease with which gaits and motion styles can be specified using our easy-to-use editing tools. We plan to extend these tools such that users can specify a motion repertoire that includes switching between gaits, portraying rich personalities and interacting appropriately with objects and humans. Finding appropriate abstractions for intuitively authoring such high-level behaviors is an interesting subject for future work.

Acknowledgments

We would like to thank Chiara Daraio and Mirko Meboldt for giving us access to their 3D-printing facilities, as well as Jung-Chew Tse and Jan Wezel for helping us with the prints. We are also grateful to Alessia Marra for her creative assistance with some of the robot designs.

References

AUERBACH, J., AYDIN, D., MAESANI, A., KORNATOWSKI, P., CIESLEWSKI, T., HEITZ, G., FERNANDO, P., LOSHCHILOV, I., DALER, L., AND FLOREANO, D. 2014. RoboGen: Robot Generation through Artificial Evolution. In *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, The MIT Press, 136–137.

BÄCHER, M., BICKEL, B., JAMES, D. L., AND PFISTER, H. 2012. Fabricating articulated characters from skinned meshes. In *Proc. of ACM SIGGRAPH '12*.

BÄCHER, M., WHITING, E., BICKEL, B., AND SORKINE-HORNUNG, O. 2014. Spin-It: Optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 33*, 4, 96:1–96:10.

CALÌ, J., CALIAN, D., AMATI, C., KLEINBERGER, R., STEED, A., KAUTZ, J., AND WEYRICH, T. 2012. 3D-printing of non-assembly, articulated models. In *Proc. of ACM SIGGRAPH Asia '12*.

CEYLAN, D., LI, W., MITRA, N. J., AGRAWALA, M., AND PAULY, M. 2013. Designing and fabricating mechanical automata from mocap sequences. In *Proc. of ACM SIGGRAPH Asia '13*.

COROS, S., THOMASZEWSKI, B., NORIS, G., SUEDA, S., FORBERG, M., SUMNER, R. W., MATUSIK, W., AND BICKEL, B. 2013. Computational design of mechanical characters. In *Proc. of ACM SIGGRAPH '13*.

DIMITROV, D., WIEBER, P.-B., FERREAU, H., AND DIEHL, M. 2008. On the implementation of model predictive control for online walking pattern generation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2685–2690.

GERTZ, E. M., AND WRIGHT, S. J. 2003. Object-oriented software for quadratic programming. *ACM Trans. Math. Softw. 29*, 1, 58–81.

HECKER, C., RAABE, B., ENSLOW, R. W., DEWEESE, J., MAYNARD, J., AND VAN PROOIJEN, K. 2008. Real-time motion retargeting to highly varied user-created morphologies. In *Proc. of ACM SIGGRAPH '08*.

KAJITA, S., KANEHIRO, F., KANEKO, K., FUJIWARA, K., AND YOKOI, K. H. K. 2003. Biped walking pattern generation by using preview control of zero-moment point. In *in Proceedings of the IEEE International Conference on Robotics and Automation*, 1620–1626.

KOO, B., LI, W., YAO, J., AGRAWALA, M., AND MITRA, N. J. 2014. Creating works-like prototypes of mechanical objects. *ACM Transactions on Graphics (Special issue of SIGGRAPH Asia 2014)*.

LAU, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2011. Converting 3D furniture models to fabricatable parts and connectors. In *Proc. of ACM SIGGRAPH '11*.

LEGER, P. C. 1999. *Automated Synthesis and Optimization of Robot Configurations: An Evolutionary Approach*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

LIPSON, H., AND POLLACK, J. B. 2000. Towards continuously reconfigurable self-designing robotics. In *ICRA*, IEEE, 1761–1766.

MASTALLI, C., WINKLER, A., HAVOUTIS, I., CALDWELL, D. G., AND SEMINI, C. 2015. On-line and on-board planning and perception for quadrupedal locomotion. In *2015 IEEE International Conference on Technologies for Practical Robot Applications (TEPRA)*, IEEE.

MEHTA, A. M., AND RUS, D. 2014. An end-to-end system for designing mechanical structures for print-and-fold robots. In *IEEE International Conference on Robotics and Automation (ICRA)*.

MEHTA, A. M., DELPRETO, J., SHAYA, B., AND RUS, D. 2014. Cogeneration of mechanical, electrical, and software designs for printable robots from structural specifications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

MORDATCH, I., TODOROV, E., AND POPOVIĆ, Z. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph. 31*, 4 (July), 43:1–43:8.

NEUHAUS, P., PRATT, J., AND JOHNSON, M. 2011. Comprehensive summary of the institute for human and machine cognition's experience with littledog. *International Journal Of Robotics Research 30*, 2 (Feb.), 216–235.

ODE, 2007. Open dynamics engine, `http://www.ode.org/`.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make it stand: Balancing shapes for 3d fabrication. In *Proc. of ACM SIGGRAPH '13*, 81:1–81:10.

SIMS, K. 1994. Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '94, 15–22.

SKOURAS, M., THOMASZEWSKI, B., COROS, S., BICKEL, B., AND GROSS, M. 2013. Computational design of actuated deformable characters. In *Proc. of ACM SIGGRAPH '13*.

THOMASZEWSKI, B., COROS, S., GAUGE, D., MEGARO, V., GRINSPUN, E., AND GROSS, M. 2014. Computational design of linkage-based characters. In *Proc. of ACM SIGGRAPH '14*.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. In *Proc. of ACM SIGGRAPH '12*.

UMETANI, N., KOYAMA, Y., SCHMIDT, R., AND IGARASHI, T. 2014. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph. 33*, 4 (July), 65:1–65:10.

WAMPLER, K., AND POPOVIĆ, Z. 2009. Optimal gait and form for animal locomotion. *ACM Trans. Graph. 28*, 3 (July), 60:1–60:8.

WAMPLER, K., AND POPOVIĆ, Z. 2009. Optimal gait and form for animal locomotion. In *ACM SIGGRAPH 2009 Papers*, ACM, New York, NY, USA, SIGGRAPH '09, 60:1–60:8.

WAMPLER, K., POPOVIĆ, Z., AND POPOVIĆ, J. 2014. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Trans. Graph. 33*, 4 (July), 49:1–49:11.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '88, 159–168.

ZHU, L., XU, W., SNYDER, J., LIU, Y., WANG, G., AND GUO, B. 2012. Motion-guided mechanical toy modeling. In *Proc. of ACM SIGGRAPH Asia '12*.