INTERACTIVE CO-DESIGN OF FORM AND FUNCTION FOR LEGGED ROBOTS USING THE ADJOINT METHOD

RUTA DESAI*, BEICHEN LI, YE YUAN and STELIAN COROS

Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA-15213, USA.
**Email: rutad@cmu.edu*

Our goal is to make robotics more accessible to casual users by reducing the domain knowledge required in designing and building robots. Towards this goal, we present an interactive computational design system that enables users to design legged robots with desired morphologies and behaviors by specifying higher level descriptions. The core of our method is a design optimization technique that reasons about the structure and motion of a robot in a coupled manner to achieve user-specified robot behavior and performance. We are inspired by the recent works that also aim to jointly optimize robot's form and function. However, through efficient computation of necessary design changes, our approach enables us to keep user-in-the-loop for interactive applications. We evaluate our system in simulation by starting with initial user designs that are physically infeasible or inadequate to perform the user-desired task. We then show optimized designs that achieve user-specifications, all while ensuring an interactive design flow.

Keywords: Legged robots, Automatic robot design, Design optimization

1. Introduction

Even from a cursory inspection, it is clear that the morphological features of living creatures are intimately related to their motor capabilities. For e.g., the long limbs and flexible spine of a cheetah lead to extreme speeds. Therefore, roboticists often look to nature for inspiration.[1–3] However, rather than copying the designs that we see in nature, we are interested in beginning to address the question: can we develop mathematically-rigorous models with the predictive power to inform the design of effective legged robots? Furthermore, we are interested in integrating such computational models into interactive design tools that make robotics accessible to casual users.

Towards this goal, we develop a computationally efficient interactive design system that allows users to create legged robots with diverse functionalities, without requiring any domain-specific knowledge. Our system automatically suggests required changes in order to achieve a specified be-

havior or task performance. In particular, we focus on periodic locomotion tasks characterized by footfall patterns, motion speed and direction. The core of our system consists of a mathematical model that maps the morphological parameters of a robot to its motor capabilities. Equipped with this model, we present an automatic design framework to co-optimize robot's structure and motion in a hierarchical manner. To deal with the computational complexity for user-interactivity, we leverage the Adjoint method.[4]

Our long term goal is to make the process of creating customized robots highly accessible.[5] In our prior work, we developed an interactive design system for rapid, and on-demand generation of custom robotic devices.[6] A major limitation of our system was that it did not provide any feedback about design improvements. To overcome this limitation, we take inspiration from past work.[7–9] However, instead of directly weaving in the robot's physical parameters within motion optimization, we establish a mapping between them using the implicit function theorem.[10] Our approach is also complementary to evolutionary approaches.[11] Rather than synthesizing designs from scratch, we adopt a user-in-the-loop approach to allow the users to guide the design process as they desire – thereby converging onto their needed outcome much faster. Further, unlike evolutionary approaches that provide no guarantees, our gradient-based optimization is locally optimal.

Finally, we validate our system in a physics-based simulation, through various task-based robot design scenarios that are challenging for casual users. We demonstrate how our system aids users in issues ranging from physical in-feasibility of the design, to sub-optimality in task performance.

2. Interactive Design

Our interactive design system is rooted in a design abstraction that allows us to combine off-the-shelf and 3D printed parts for designing robots.[6] As figure 1(a) illustrates, our graphical user interface (GUI) consists of a design workspace for designing (left) and a simulation workspace for design testing (right). The users can browse through various modules from a menu (top) in the design workspace, and drag-and-drop them into the scene to construct or modify a robot design. We assume that all parts of a robot's articulated structure (other than motors for joints) are 3D printed. We automatically create these 3D printable connecting structures between actuated joints. These structure geometries are updated with every manual user operation or automatic design update that changes the robot morphology.

To enable this, we define a parameterized 3D printable connector module (see fig. 1(b)) whose position and orientation can be updated interactively with changes in the design. Each connector module has 'virtual' attachment

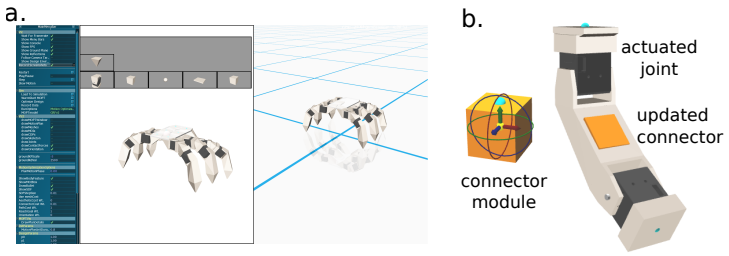

Figure 1. (a) GUI. (b) A parameterized 3D printable connector (orange) dynamically updates to connect actuated joints. Our system's capabilities are highlighted in our video.

points on its face that get updated based on the positions of the connector and the motors. These attachment points are used to update the shape and size of the connector's convex hull structure as needed. The users can also pause and restart the optimization at any point, to update the structure manually in between for achieving desired aesthetics.

Although our interactive design interface is a powerful approach for forward design, modifying designs to achieve a desired task requires domain knowledge. We next present our design optimization framework, which automatically optimizes the robot's form and behavior for a desired task.

3. Automatic Design Optimization

Designing robots with task-specific behaviors is highly skill-intensive and time-consuming. One must decide the robot's structure – physical dimensions of its body, and its articulated parts, as well as the placement of motors. One must then define how to control the motors for a co-ordinated movement that achieves a task. The robot's structure has a huge effect on the tasks it can perform. Therefore, designers typically iterate back and forth between physical and behavior design to create a task-specific robot. To capture this coupling between the robot's form and function, we parameterize a robot with a set of structure parameters $\mathbf{s}$, and motion parameters $\mathbf{m}$. However, instead of treating $\mathbf{m}$ and $\mathbf{s}$ independently, our goal is to represent robot motions as a function of its structure $\mathbf{m}(\mathbf{s})$. Apart from being intuitive, such a representation allows us to solve for an optimal task-specific behavior and design hierarchically, in a computationally efficient manner enabling interactivity during design.

3.1. *Parameterization*

A larger variety of robots including manipulators, and walking robots are composed of articulated chain like structures, in particular, of serially connected and actuated links. Such robot morphologies can be well described as kinematic trees starting at the root of the robot. The design parameters

$\mathbf{s}$ is used to specify such robot morphology, which is given by:

$$\mathbf{s} = (l_1, \ldots, l_g, \mathbf{a}_1, \ldots, \mathbf{a}_n, b_w, b_l) \ , \tag{1}$$

where g is the number of links, $l_i \in \mathbb{R}$ is the length of each link, n is the number of actuators, and $\mathbf{a}_i \in \mathbb{R}^3$ is the actuator parameters. For linear actuators, $\mathbf{a}_i$ defines the 3D attachment points, while for rotary actuators, it corresponds to orientation of axis of rotation. Apart from these parameters that represent the kinematic tree morphology of the robot, we use two additional parameters b_w and b_l to represent the physical dimensions of the robot's body (width and length respectively). Likewise, the motion parameters $\mathbf{m} = (\mathbf{P}_1, \ldots, \mathbf{P}_T)$ are defined by a time-indexed sequence of vectors $\mathbf{P}_i$, where T denotes the time for each gait cycle. $\mathbf{P}_i$ is defined as:

$$\mathbf{P}_i = \left(\mathbf{q}_i, \mathbf{x}_i, \mathbf{e}_i^1, \ldots, \mathbf{e}_i^k, \mathbf{f}_i^1, \ldots, \mathbf{f}_i^k, c_i^1, \ldots, c_i^k, \right) \ , \tag{2}$$

where $\mathbf{q}_i$ defines the pose of the robot, i.e., the position, and orientation of the root as well as joint information such as angle values, $\mathbf{x}_i \in \mathbb{R}^3$ is the position of the robot's center of mass (COM), and k is the number of end-effectors. For each end-effector j, we use $\mathbf{e}_i^j \in \mathbb{R}^3$ to represent its position and $\mathbf{f}_i^j \in \mathbb{R}^3$ to denote the ground reaction force acting on it. We also use a contact flag c_i^j to indicate whether it should be grounded ($c_i^j = 1$) or not.

3.2. *Method Overview*

Given an initial robot design and a task specification, our goal is to change $\mathbf{s}$ and $\mathbf{m}$ (eq. 1, 2) to obtain a design better suited for the task. Users typically define the initial design using our GUI. Various task descriptions such as preferred motion direction, speed, motion styles (walking, trotting, turning) etc. can also be specified with the GUI. These task specifications can then be encoded into a cost function $F(\mathbf{s}, \mathbf{m})$. Assuming $\mathbf{p}$ to be the parameter vector containing both structure and motion parameters $\mathbf{p} = [\mathbf{s}, \mathbf{m}]$, one can search for an optimal $\mathbf{p}$ along the direction of $F(\mathbf{p})$'s gradient $\frac{\partial F}{\partial \mathbf{p}}$. However, $\mathbf{s}$ and $\mathbf{m}$ are inherently coupled. Hence, instead of searching $\mathbf{s}$ and $\mathbf{m}$ independently, we first update $\mathbf{s}$, and then update $\mathbf{m}$ within a constrained manifold that ensures the validity and optimality of $\mathbf{m}$'s update, given $\mathbf{s}$. By constructing a manifold of structure and motion parameters of a robot design, we can explore the sensitivity of robot's motion $\mathbf{m}$ to its structure $\mathbf{s}$. Starting with an initial design $(\mathbf{s_0}, \mathbf{m_0})$ on the manifold, one can search for $\mathbf{s}$, and corresponding $\mathbf{m}(\mathbf{s})$ on this manifold, such that $F(\mathbf{s}, \mathbf{m})$ is minimized. This dependency of $\mathbf{m}$ on $\mathbf{s}$ is captured by the Jacobian $\frac{d\mathbf{m}}{d\mathbf{s}}$ (see Sec. 3.3). This Jacobian is used to compute the search direction $\frac{dF}{d\mathbf{s}}$ for updating $\mathbf{s}$

within the manifold. However, $\frac{d\mathbf{m}}{d\mathbf{s}}$ is expensive to compute. Therefore, we further simplify this computation by using the Adjoint method (Sec. 3.4).

At each iteration i, an update $\mathbf{s}'$ is proposed along the gradient direction $\frac{dF}{d\mathbf{s}}$ with step $\delta_\mathbf{s}$. For each such update of $\mathbf{s}$, multiple updates of $\mathbf{m}$ are executed to obtain the corresponding optimal $\mathbf{m}'$. Specifically, $\mathbf{m}$ is updated in the search direction defined by Newton's method $(\frac{\partial^2 F}{\partial \mathbf{m}^2}^{-1} \frac{\partial F}{\partial \mathbf{m}})$ by $\delta_\mathbf{m}$ step obtained using line-search.[12] Note that $\frac{\partial^2 F}{\partial \mathbf{m}^2}$ and $\frac{\partial F}{\partial \mathbf{m}}$ represent the Hessian and gradient of F with respect to $\mathbf{m}$ respectively. If $F(\mathbf{s}', \mathbf{m}') < F(\mathbf{s}_i, \mathbf{m}_i)$, the updates are accepted, else a new $\mathbf{s}'$ is proposed along $\frac{dF}{d\mathbf{s}}$ with smaller step size $(\frac{\delta_\mathbf{s}}{2})$. We iterate over this procedure till $F(\mathbf{s}_i, \mathbf{m}_i) < threshold$ implying that $\mathbf{s}_i$ and $\mathbf{m}_i$ are optimal. A more detailed overview of this algorithm can be found in the extended version of our paper on arXiv.

3.3. *Coupling form and function for robot design*

It is hard to analytically represent the dependency of robot's motion on its structure. Instead, we assume a manifold that relates robot's structure and behavior capabilities, given a specific task: $\mathbf{G}(\mathbf{s}, \mathbf{m}) = 0$, where $\mathbf{G}(\mathbf{s}, \mathbf{m}) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_m} \rightarrow \mathbb{R}^{n_m}$. Such an implicit manifold between structure and function can be converted into an explicit relation between the two within a small region around a point $P_0(\mathbf{s_0}, \mathbf{m_0})$ on the manifold, using the implicit function theorem.[10] The theorem states that when we change $\mathbf{s_0}$ and $\mathbf{m_0}$ by $\Delta\mathbf{s}$ and $\Delta\mathbf{m}$, the change in the function $\Delta\mathbf{G}$ should be zero to remain on the manifold. Using chain rule to compute $\Delta\mathbf{G}$, we obtain the following explicit relation between $\Delta\mathbf{s}$ and $\Delta\mathbf{m}$:

$$\Delta\mathbf{G} = \frac{\partial \mathbf{G}}{\partial \mathbf{s}}\Delta\mathbf{s} + \frac{\partial \mathbf{G}}{\partial \mathbf{m}}\Delta\mathbf{m} = 0 \implies \Delta\mathbf{m} = -\left(\frac{\partial \mathbf{G}}{\partial \mathbf{m}}\right)^{-1} \frac{\partial \mathbf{G}}{\partial \mathbf{s}}\Delta\mathbf{s} \tag{3}$$

where $\left(\frac{\partial \mathbf{G}}{\partial \mathbf{m}}\right)$ and $\left(\frac{\partial \mathbf{G}}{\partial \mathbf{s}}\right)$ represents the Jacobian of $\mathbf{G}(\mathbf{s}, \mathbf{m})$ with respect to $\mathbf{m}$ and $\mathbf{s}$ respectively.

To compute such a manifold, we start with a task-specific cost function $F(\mathbf{s}, \mathbf{m})$. For each robot morphology $\mathbf{s}$, there exists an optimal $\mathbf{m}^*$ that minimizes $F(\mathbf{s}, \mathbf{m})$. Hence, the gradient of F with respect to $\mathbf{m}$ at point $(\mathbf{s}, \mathbf{m}^*)$ should be zero. One can then search for an optimal $\mathbf{s}^*$ along the manifold defined by this gradient $\mathbf{G}(\mathbf{s}, \mathbf{m}) = \frac{\partial F(\mathbf{s}, \mathbf{m})}{\partial \mathbf{m}}$. An optimal $\mathbf{s}^*$ on such a $\mathbf{G}(\mathbf{s}, \mathbf{m})$ would automatically ensure a corresponding valid and optimal $\mathbf{m}^*$ for the task. For searching such an optimal $\mathbf{s}^*$, we thus need to solve the following optimization problem:

$$\min_s F(\mathbf{s}, \mathbf{m}) \qquad s.t. \quad \mathbf{G}(\mathbf{s}, \mathbf{m}) = 0 \tag{4}$$

where $F(\mathbf{s}, \mathbf{m})$ is the cost function; $\mathbf{G}(\mathbf{s}, \mathbf{m})$ denotes the gradient of $F(\mathbf{s}, \mathbf{m})$ with respect to motion parameters $\mathbf{m}$. Empowered by the Jacobian $\frac{d\mathbf{m}}{d\mathbf{s}}$ that essentially encodes $\mathbf{m}(\mathbf{s})$ (eq. 3), we can define the search direction for $\mathbf{s}$ as:

$$\frac{dF}{d\mathbf{s}} = \frac{\partial F}{\partial \mathbf{m}} \frac{d\mathbf{m}}{d\mathbf{s}} + \frac{\partial F}{\partial \mathbf{s}} \implies \frac{dF}{d\mathbf{s}} = -\frac{\partial F}{\partial \mathbf{m}} \left(\frac{\partial \mathbf{G}}{\partial \mathbf{m}} \right)^{-1} \frac{\partial \mathbf{G}}{\partial \mathbf{s}} + \frac{\partial F}{\partial \mathbf{s}}. \quad (5)$$

3.4. *The Adjoint method*

Computing $\frac{dF}{d\mathbf{s}}$ requires the calculation of the Jacobian $\frac{d\mathbf{m}}{d\mathbf{s}}$ which is computationally very expensive. It requires solving n_s linear equations (for each column in Jacobian matrix $\frac{\partial \mathbf{G}}{\partial \mathbf{s}}$), and the procedure gets very costly for large n_s. Instead, we use the Adjoint method to efficiently compute the gradient $\frac{dF}{d\mathbf{s}}$. This method formulates the computation of gradient as constrained optimization problem, and then uses the dual form of this optimization problem for faster computation.[4] Other applications have also sought out the Adjoint method for similar purposes in the past.[13] In particular, $\frac{dF}{d\mathbf{s}}$ takes on the following form using the adjoint method:

$$\frac{dF}{d\mathbf{s}} = \lambda^{\intercal} \frac{\partial \mathbf{G}}{\partial \mathbf{s}} + \frac{\partial F}{\partial \mathbf{s}}, \quad (6)$$

where λ is called the vector of *adjoint variables*. Such a computation of $\frac{dF}{d\mathbf{s}}$ now involves solving only one linear equation to obtain λ, followed by one matrix-vector multiplication and one vector addition (eq. 6). This is more efficient as compared to solving n_s linear equations for $\frac{d\mathbf{m}}{d\mathbf{s}}$ earlier.

4. Results

We explore three simulated examples to study the utility and effectiveness of our approach. Although, we only show our current results in simulation, we have confirmed that the simulation matches physical results previously.[6]

When novices design robots, it can be hard for them to decide where the actuators should be located and how they should be oriented for achieving a specific behavior. Fig. 2(a) shows one such example of a 'puppy' robot with three motors per leg. Even with enough number of actuators, the robot can only walk in one direction (forward) owing to its actuator placements. Parameterization of the actuator orientations $\mathbf{a_i}$ in eq. 1 enables design optimization to change them for equipping the robot to walk in any specific direction. Without the optimization of such structural parameters, it may be impossible to achieve such tasks (see fig. 2(b)).

Even when a design can theoretically achieve the desired behavior, it may be rendered infeasible due to real world constraints such as collisions. Fig. 3(a) shows a robot that can walk in the user-specified direction at

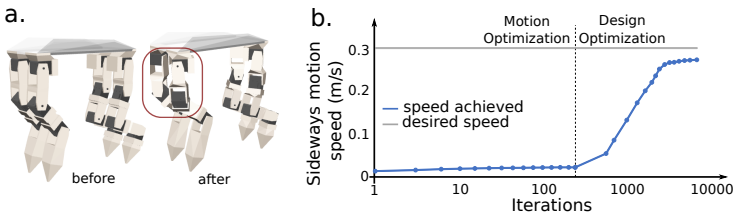

Figure 2. (a) The initial design of 'puppy' robot can only walk forward. Our design optimization enables the robot to walk sideways. (b) Optimizing motion parameters is not sufficient and optimization of the structure parameters is essential in this example.

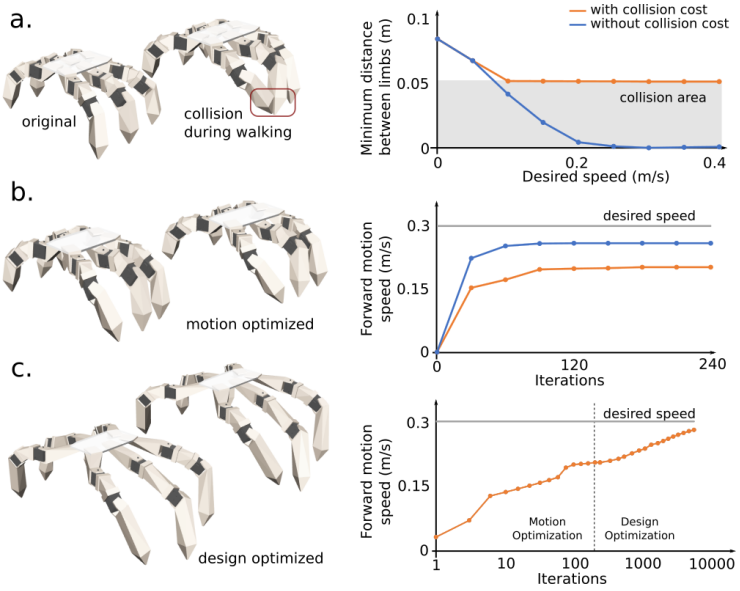

Figure 3. (a) Collision in a hexapod's limbs at high speeds. (b) Accounting for collisions in motion optimization prevents this, but also restricts the robot from walking faster. (c) Instead, design optimization increases spacing between limbs and their lengths.

desired speeds. However, when walking speeds increase above 0.1 ms^{-1}, the robot's limbs start colliding. It is hard to anticipate such issues *a priori*. Along with helping the user to test such scenarios in simulation, our system can also automatically prevent them by using feasibility constraints during motion optimization. However, these constraints prevent the required range of limb motions needed for fast walking, limiting the ability of the robot to walk at desired speed (fig. 3(b)). Instead, design optimization changes the design to achieve both these contradicting requirements (see fig. 3(c)).

Finally, designing robots for multiple tasks is also highly challenging, especially if the tasks demand opposing design characteristics. Consider the task of walking and pacing for a quadruped robot shown in fig. 4(a).

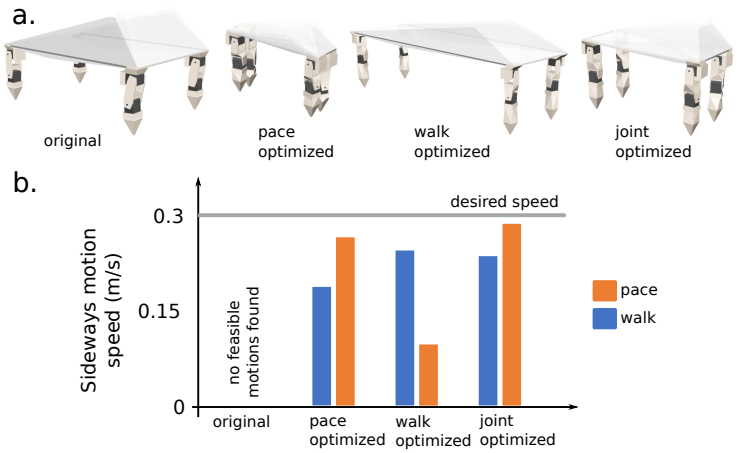

Figure 4. (a) A quadruped robot design that can only walk forward, is optimized for pace, sideways walking, and jointly optimized for both these behaviors. (b) Jointly optimized design achieves a reasonable trade-off between the performance of both tasks.

The original design can only walk forward owing to its actuator placements (similar to the 'puppy' robot in fig. 2(a)). Its wider body and shorter limbs prevent it from pacing in stable manner. Individually optimizing the design for pacing and walking may not be sufficient for enabling the robot to perform both tasks. Pace-based design optimization generates a slim bodied robot, while walk-based design optimization produces a wider body size to increase stability during fast walking (see fig. 4(a)). Such a wider body in turn, negatively affects the pacing behavior (fig. 4(b)). To achieve reasonable performance for both these tasks, a trade-off is thus required. The individual requirements for each task $F_i(\mathbf{s}, \mathbf{m_i})$ can be combined in weighted manner into $F_{joint}(\mathbf{s}, \mathbf{m}) = \sum w_i F_i(\mathbf{s}, \mathbf{m_i})$. Weights w_i representing the importance of each task can be set by the users. Such joint optimization of walking and pacing (with $w_1 = w_2 = 0.5$) for quadruped in fig. 4(a) succeeds in achieving the necessary trade-off as illustrated in the resultant medium bodied optimized design, and the corresponding task performance.

Table 1. Design optimization statistics for example robots

Robot	Number of Parameters	Motion Opt. Iterations	Design Opt. Iterations	Time (s)
Puppy (Fig. 2)	614	6207	32	107.66
Hexapod (Fig. 3)	1044	5013	53	97.47
Quadruped (Fig. 4)	1050	14873	100	124.47

Table 1 shows the design times for optimizing the designs of robots in fig. 2, 3, 4. For quadruped in fig. 4 these statistics are reported for the joint optimization scenario. Note that, even when its number of optimization pa-

rameters are roughly similar to that of the hexapod, there is a significant difference in the number of optimization iterations, and the time required. This is because of the contradicting requirements that the two tasks demand, making the problem more challenging. Also note that for each iteration of design optimization, multiple iterations of motion optimization are executed. However, as shown in the statistics, the large number of these iterations are executed in minutes. Such computational efficiency is at the core of interactivity in our system. Apart from an efficient implementation in C++, a scalable approach using the Adjoint method enables the same.

5. Discussion and Future Works

We introduced an interactive robot design and optimization system that allows casual users to create customized robotic creatures for specific tasks. Apart from generating feasible behaviors, our system improves the performance of robot through an automatic design optimization process. In the future we plan to extend our design optimization technique for a broader class of motions and behaviors, including climbing, carrying weights or avoiding obstacles. Further, to find the right balance between automation and user control during design, we plan to perform an extensive user study as well.

References

1. A. Crespi, K. Karakasiliotis, A. Guignard and A. J. Ijspeert, *TRO* **29** (2013).
2. K. Graichen, S. Hentzelt, A. Hildebrandt, N. Kärcher, N. Gaißert and E. Knubben, *Control Engineering Practice* **42**, 106 (2015).
3. M. Spenko, G. C. Haynes, J. Saunders, M. R. Cutkosky, A. A. Rizzi, R. J. Full and D. E. Koditschek, *Journal of Field Robotics* **25**, 223 (2008).
4. M. B. Giles and N. A. Pierce, *Flow, turbulence and combustion* **65**, 393 (2000).
5. N. Bezzo, A. Mehta, C. D. Onal and M. T. Tolley, *IEEE Magazine* **22** (2015).
6. R. Desai, Y. Yuan and S. Coros, Computational abstractions for interactive design of robotic devices, in *IEEE International Conference on Robotics and Automation*, 2017.
7. B. Canaday, S. Zapolsky and E. Drumwright, Interactive, iterative robot design, in *ICRA*, 2017.
8. S. Ha, S. Coros, A. Alspach, J. Kim and K. Yamane, Joint optimization of robot design and motion using implicit function theorem, in *RSS*, 2017.
9. A. Spielberg, B. Araki, C. Sung, R. Tedrake and D. Rus, Functional co-optimization of articulated robots, in *ICRA*, 2017.
10. K. Jittorntrum, *An implicit function theorem, JOTA* **25**, 575 (1978).
11. N. Cheney, J. Bongard, V. SunSpiral and H. Lipson, On the difficulty of co-optimizing morphology and control in evolved virtual creatures, in *Proceedings of the Artificial Life Conference*, 2016.
12. J. Nocedal and S. Wright, *Springer, New York, USA, 2006* (2006).
13. A. McNamara, A. Treuille, Z. Popović and J. Stam, Fluid control using the adjoint method, in *ACM Transactions On Graphics (TOG)*, (3)2004.