

MIT Open Access Articles

DiffAqua

The MIT Faculty has made this article openly available. *Please share* how this access benefits you. Your story matters.

As Published: 10.1145/3450626.3459832

Publisher: Association for Computing Machinery (ACM)

Persistent URL: <https://hdl.handle.net/1721.1/134208>

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Terms of use: Creative Commons Attribution 4.0 International license



DiffAqua: A Differentiable Computational Design Pipeline for Soft Underwater Swimmers with Shape Interpolation

PINGCHUAN MA, MIT CSAIL, USA
TAO DU, MIT CSAIL, USA
JOHN Z. ZHANG, ETH Zurich, Switzerland
KUI WU, MIT CSAIL, USA
ANDREW SPIELBERG, MIT CSAIL, USA
ROBERT K. KATZSCHMANN, ETH Zurich, Switzerland
WOJCIECH MATUSIK, MIT CSAIL, USA

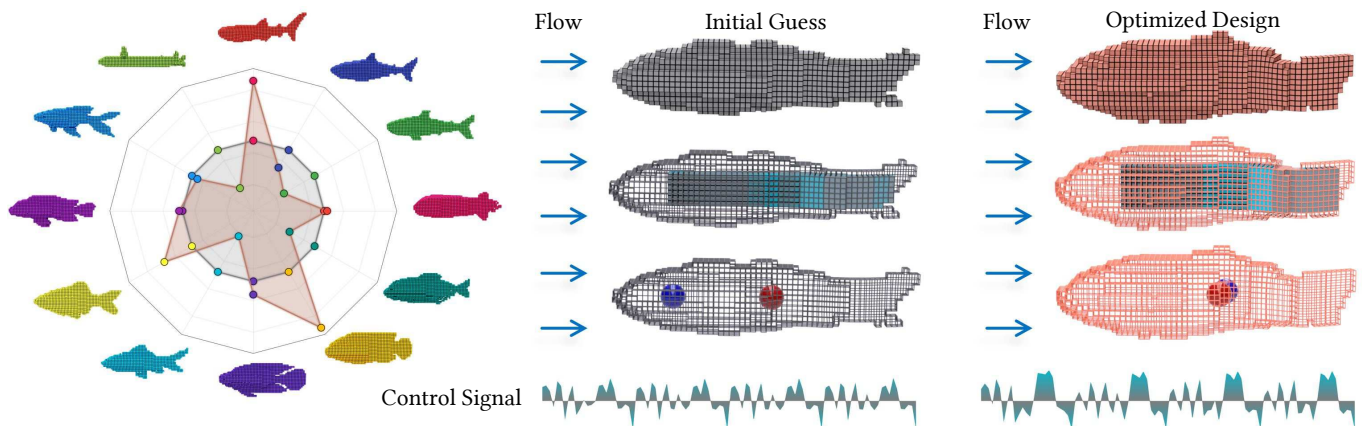


Fig. 1. In this example, we co-design a swimmer’s geometry and controller for position-keeping in running water (blue arrows in upper middle and upper right). The blue and red spheres indicate the desired and actual position of the swimmer. The swimmer’s geometric design weights are initialized as the average of 12 base shapes with the Wasserstein distance metric shown in gray (left). After optimization, the final geometric design weights of each base shape are shown in orange (left). With the initial shape and actuator locations (upper middle) and the initial controller (lower middle), the position-keeping performance is poor, as indicated by the large distance between the red and blue dots. With the optimized shape and actuator locations (upper right) and the optimized controller (lower right), performance is markedly improved, i.e., blue and red are closer together.

The computational design of soft underwater swimmers is challenging because of the high degrees of freedom in soft-body modeling. In this paper, we present a differentiable pipeline for co-designing a soft swimmer’s geometry and controller. Our pipeline unlocks gradient-based algorithms for discovering novel swimmer designs more efficiently than traditional gradient-free solutions. We propose Wasserstein barycenters as a basis for the geometric

design of soft underwater swimmers since it is differentiable and can naturally interpolate between bio-inspired base shapes *via* optimal transport. By combining this design space with differentiable simulation and control, we can efficiently optimize a soft underwater swimmer’s performance with fewer simulations than baseline methods. We demonstrate the efficacy of our method on various design problems such as fast, stable, and energy-efficient swimming and demonstrate applicability to multi-objective design.

Authors’ addresses: Pingchuan Ma, MIT CSAIL, Cambridge, MA, USA, pcma@csail.mit.edu; Tao Du, MIT CSAIL, Cambridge, MA, USA, taodu@csail.mit.edu; John Z. Zhang, ETH Zurich, Zurich, Switzerland, john.zhang@sr.ethz.ch; Kui Wu, MIT CSAIL, Cambridge, MA, USA, kuiwu@csail.mit.edu; Andrew Spielberg, MIT CSAIL, Cambridge, MA, USA, aespielberg@csail.mit.edu; Robert K. Katzschmann, ETH Zurich, Zurich, Switzerland, rkk@ethz.ch; Wojciech Matusik, MIT CSAIL, Cambridge, MA, USA, wojciech@csail.mit.edu.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**; Physical simulation; Volumetric models.

Additional Key Words and Phrases: Computational design, differentiable simulation, optimal transport, geometry and control co-design, multi-objective optimization



This work is licensed under a Creative Commons Attribution International 4.0 License.
© 2021 Copyright held by the owner/author(s).
0730-0301/2021/8-ART132
<https://doi.org/10.1145/3450626.3459832>

ACM Reference Format:

Pingchuan Ma, Tao Du, John Z. Zhang, Kui Wu, Andrew Spielberg, Robert K. Katzschmann, and Wojciech Matusik. 2021. DiffAqua: A Differentiable Computational Design Pipeline for Soft Underwater Swimmers with Shape Interpolation. *ACM Trans. Graph.* 40, 4, Article 132 (August 2021), 14 pages. <https://doi.org/10.1145/3450626.3459832>

1 INTRODUCTION

Designing bio-inspired underwater swimmers has long been an exciting interdisciplinary research problem for biologists and engineers [Berlinger et al. 2018; Fish and Lauder 2006; Katzschmann et al. 2018; Marchese et al. 2014; Triantafyllou and Triantafyllou 1995]. Aquatic locomotive performance of underwater swimmers is governed by two interrelated aspects: the control policy responsible for coordinated actuation and the geometry by which the actuation is transformed into motion through hydrodynamic forces. While the computational co-design of geometry and control has been explored in the context of articulated rigid walking and flying robots [Du et al. 2016; Ha 2019; Pathak et al. 2019; Zhao et al. 2020], the co-design of geometry and control of a *soft* robot consisting of substantially deformable materials has been sparsely studied. Since a soft robot is governed by continuum physics with many more degrees of freedom (DOFs), existing methods are not readily transferable to soft robotic design problems. Optimizing over all degrees of freedom of an infinite-dimensional continuum elastic body is computationally intractable, even when approximated with a large number of discrete elements. Lower-dimensional geometric representations are needed to allow for efficient shape exploration without sacrificing expressiveness.

To remedy the lack of an intuitive and low-dimensional design space suitable for soft underwater swimmers, we propose representing a soft swimmer's shape and actuators as probability distributions in 3D space and interpolate between designs with optimal transport [Rubner et al. 2000]. Given a set of base shapes representing swimmer archetypes, we define a vector space spanned by these shapes according to the Wasserstein distance metric [Solomon et al. 2015]. Such a representation of the soft swimmer's design space brings a few key benefits. First, it allows for each design to be represented as a low-dimensional Wasserstein barycentric weight vector. Second, and more importantly, this interpolation procedure is differentiable [Bonneel et al. 2016], enabling gradient-based optimization algorithms for fast exploration of the design space. We further combine this design space with a differentiable simulator [Du et al. 2021] and control policy, creating a full pipeline for co-optimizing the body and brain of soft swimmers using gradient-based optimization algorithms.

We evaluate our co-optimization pipeline on a set of soft swimmer design problems whose objectives include forward swimming, stable swimming under opposing flows, and energy conservation. We further demonstrate the pipeline's applicability to multi-objective design and the generation of Pareto fronts. We show that our algorithm converges significantly faster than gradient-free optimization algorithms [Hansen et al. 2003] and strategies that alternate between optimizing geometric design and control.

In this paper, we contribute:

- a low-dimensional differentiable design space parametrizing the shape and actuation of soft underwater swimmers with the Wasserstein distance metric,
- a differentiable pipeline for co-optimizing the geometric design and controller of a soft swimmer concurrently, and
- demonstrations of this algorithm on a set of bio-inspired single- and multi-objective underwater swimming tasks.

2 RELATED WORK

Our approach builds upon recent and seminal work in differentiable simulation, soft robot and character control, computational co-design of robots, and shape parametrizations.

Differentiable simulation. Differentiable simulation allows for the direct computation of the gradients of continuous parameters affecting the system performance, such as control, material, and geometric parameters. Gradients computed from differentiable simulation can be directly fed into numerical optimization algorithms, immediately unlocking applications such as system identification, computational control, and design optimization. Differentiable simulation has a rich history in robotics and physically-based animation across various domains, including rigid-body dynamics [Belbute-Peres et al. 2018; Degraeve et al. 2019; Geilinger et al. 2020; Popović et al. 2003], fluid dynamics [Holl et al. 2020; McNamara et al. 2004; Schenck and Fox 2018] and cloth physics with rigid coupling [Liang et al. 2019; Qiao et al. 2020]. In cases where manually deriving gradients is difficult, automatic differentiation frameworks [Gifftaler et al. 2017; Hu et al. 2020] or learned, approximate models [Chen et al. 2018; Li et al. 2019; Sanchez-Gonzalez et al. 2020] have been employed. Most related to our work are differentiable simulation methods for soft bodies [Du et al. 2021; Hahn et al. 2019; Hu et al. 2019; Huang et al. 2021]. We build upon the work of Du et al. [2021], which proposes a differentiable projective dynamics framework capable of implicit integration.

Dynamic soft robot and character control. Soft robotic control is traditionally more difficult than rigid robotic control, since the infinite-dimensional state spaces of soft robots are difficult to computationally reason about. Previous papers [Geijtenbeek et al. 2013; Won and Lee 2019] demonstrated learning general controllers for articulated, rigid-body walking robots in a variety of forms. Tan et al. [2011] explored model-based control optimization of articulated rigid swimmers in an environment coupled with fluid. Control using model-free reinforcement learning for systems with solid-fluid coupling has been studied in Ma et al. [2018]. Simplified dynamical models which treat soft tendril structures compactly and accurately as rod-and-spring-like structures have allowed for the natural adoption of modern control algorithms like those used for articulated rigid robots [Della Santina et al. 2018; Katzschmann et al. 2019a; Marchese et al. 2016]. Grzeszczuk and Terzopoulos [1995] proposed representing the actuators of swimmers with a simplified biomechanical model and automatically generating their controllers. There are also works, such as Hecker et al. [2008], synthesizing animation independent of character morphology for the purpose of downstream retargeting, with the motion described by familiar posing and key-framing methods. More general approaches [Barbič et al. 2009; Barbič and Popović 2008; Katzschmann et al. 2019b; Spielberg et al. 2019; Thieffry et al. 2018] present dimensionality reduction strategies for soft robots to compact representations used in model-based control. Most similar to our work are the approaches of Hu et al. [2019] and Min et al. [2019]. In Hu et al. [2019], a differentiable soft-body simulator is coupled with parametric neural network controllers. By tracking the positions and velocities of manually specified regions as control inputs, a loss function measuring

the forward progress in robot locomotion is directly backpropagated to control parameters, enabling efficient model-based control optimization *via* gradient descent. Min et al. [2019] present model-free control of soft swimmers using reinforcement learning with similar handcrafted features. Our approach for control optimization is similar to the model-based optimization framework of Hu et al. [2019], but instead of considering legs and crawlers, we optimize underwater swimmers. Our work further distinguishes itself from prior art through the realization of the non-trivial co-optimization of shape and control of soft underwater swimmers.

Computational robot co-design. Much of the existing work on the co-design of rigid robots has focused on the interplay between geometric, inertial and control parameters. Megaro et al. [2015]; Schulz et al. [2017b] explored the interactive design of robot control and geometry, in which CAD-like front-ends guided human-in-the-loop, simulation-driven control and design optimization. Ha et al. [2017]; Spielberg et al. [2017]; Wampler and Popović [2009] presented algorithms for co-optimizing geometric and inertial parameters of robots with open-loop controllers; Ha [2019]; Schaff et al. [2019] extended these ideas to the space of closed-loop neural network controllers *via* reinforcement learning approaches. Du et al. [2016] presented a method for co-optimizing the control and geometric parameters for optimal multicopter performance. Sims [1994] models the morphology as a graph and co-optimizes it with control using an evolutionary strategy. In each of the approaches described above, the geometric representations are simple. Each robotic link is parameterized by no more than a handful of geometric parameters, substantially limiting the morphological search. More complex representations are used in the works by Ha et al. [2018]; Wang et al. [2018]; Zhao et al. [2020], presenting algorithms for searching over various robot topologies. These techniques lead to more geometrically and functionally varied robots; however, this discrete search space cannot be directly optimized by the continuous optimization approaches to which differentiable simulation lends itself.

Computational co-design of soft robots has been more sparsely explored. Cheney et al. [2014]; Corucci et al. [2016]; Van Diepen and Shea [2019] presented heuristic search algorithms for searching over soft robotic topology and control, including swimmers in the case of Corucci et al. [2016]. These sampling-based approaches focus on geometry and (discrete) materiality, typically leaving actuation as open-loop, pre-programmed cyclic patterns. Contrasted with these approaches are those of Hu et al. [2019]; Spielberg et al. [2019], which exploit system gradients to co-optimize over closed-loop control, observation models, and spatially varying material parameters, but not geometry. Our approach combines advantages of both lines of research, relying on the differentiable simulation for fast gradient-based co-optimization of neural network control and complex geometry.

Shape interpolation. Many bases for shape representations have been proposed over the years for applications in shape analysis. Geometrically-based spaces [Baek et al. 2015; Bonneel et al. 2016; Lewis et al. 2014; Ovsjanikov et al. 2012; Schulz et al. 2017a; Solomon et al. 2015] parameterize shape collections based on intrinsic geometric metrics computed across a data set; learning-based spaces [Averkiou et al. 2014; Bronstein et al. 2011; Fish et al. 2014; Mo

et al. 2019; Park et al. 2019; Yang et al. 2019], by contrast, learn parametrizations based on statistical features of that dataset and can also easily incorporate non-geometric information (such as labels). In this work, we opt for the geometrically-based convolutional Wasserstein basis [Solomon et al. 2015], which interpolates between meshes using the Wasserstein distance. This basis smoothly interpolates between shapes with “as few” in-between modifications as necessary, keeping our mesh parametrizations well-behaved and minimizing the chance of artifacts that might cause difficulties in simulation. This continuous and differentiable basis makes it amenable to continuous co-optimization.

3 SYSTEM OVERVIEW

Our design optimization procedure starts with a collection of base shapes from which to build a geometric design space using the Wasserstein distance metric. Each base design also specifies a multivariate normal distribution for each of its actuators. For our bio-inspired exploration of swimmers, we have selected base shapes inspired by nature (Sec. 4), *e.g.*, sharks, manta rays, and goldfish. Next, the user specifies a controller which is either an open-loop sinusoidal signal or a neural network. The state observations of the closed-loop neural network controller are defined by the user (Sec. 5). The user then evaluates the soft swimmer *via* differentiable simulation (Sec. 6), which returns both the swimmer’s performance and its gradients with respect to geometric design and control parameters. Finally, we embed both the design space and the differentiable simulator in a gradient-based optimization procedure, which co-optimizes both the geometric design and the controller until convergence (Sec. 7), closing our design loop. We present the overview of our method in Fig. 2.

4 GEOMETRIC DESIGN

Our soft swimmer’s geometric design consists of two parts: its volumetric shape and its actuator locations. This section explains our differentiable parameterization of this geometric design space. The key idea is to represent geometric designs as probability density functions and generate novel swimmers by interpolating the probability density functions of the base designs. After the design space has been parametrized, any swimmer in this space can be represented compactly by low-dimensional discrete and continuous parameters.

4.1 Shape Design

Given a set of user-defined base shapes, we interpolate between these bases with the Wasserstein distance metric to form a continuous and differentiable shape space. The reason behind this choice is twofold. First, the Wasserstein barycentric interpolation encourages smooth, plausible intermediate results even between shapes with substantially different topology [Solomon et al. 2015], which is common in underwater creatures. Second, efficient numerical solutions exist for solving Wasserstein barycenters, and their gradients are also readily available [Bonneel et al. 2016; Solomon et al. 2015]. These two features make Wasserstein distance a proper choice for building a fully differentiable shape design space for soft swimmers.

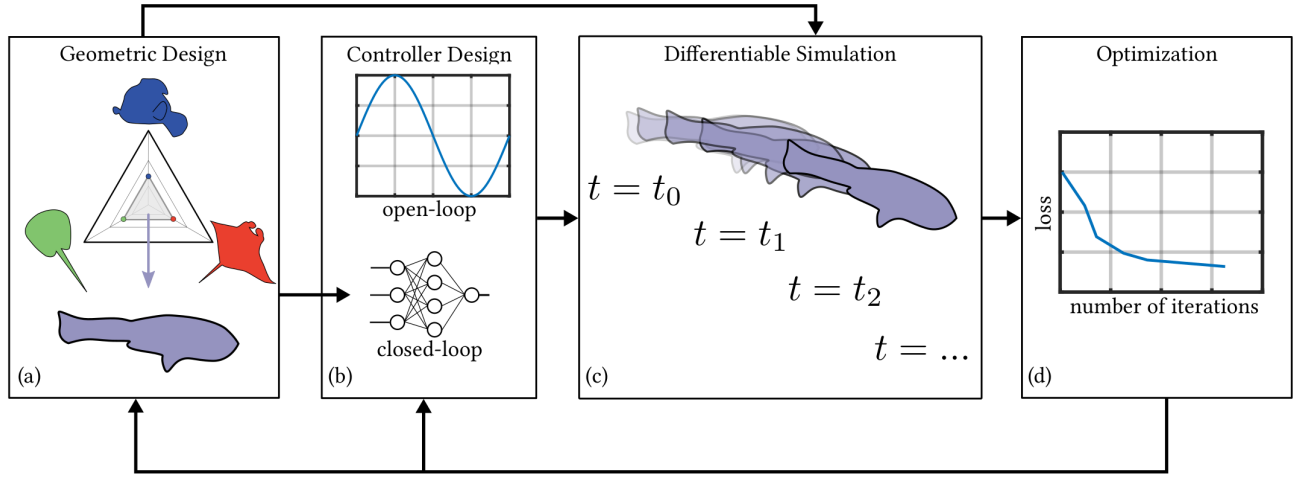


Fig. 2. In our computational design pipeline, we begin with (a) the shape and actuator design using the Wasserstein distance to interpolate smoothly between base shapes and between actuators. Given a set of base shapes by the user, we initialize the shape parametrization by assigning equal weights to all bases. (b) Next, we consider the controller design for two cases: an open-loop controller and a closed-loop controller. (c) Given this initial shape, actuators, and controller, we simulate the design using a differentiable simulator to evaluate its performance. (d) Finally, we take advantage of the differentiability of our framework to compute the gradients of a given objective with respect to geometry and control parameters simultaneously. These gradients are used in a gradient descent optimizer to concurrently adapt the geometric and control design parameters.

Formally, we consider the designs of our soft swimmers to be embedded in an axis-aligned bounding box $\Omega \subseteq \mathcal{R}^d$, where $d = 2$ or 3 is the dimension. Without loss of generality, we rescale Ω uniformly and shift it so that it has unit volume and is centered at the origin of \mathcal{R}^d . We use $d_E : \Omega \times \Omega \rightarrow \mathcal{R}_+$ to denote the Euclidean distance function, $\mathcal{P}(\Omega)$ the space of probability measures on Ω , and $\mathcal{P}(\Omega \times \Omega)$ the space of probability measures on $\Omega \times \Omega$.

For any probability measure $P \in \mathcal{P}(\Omega)$, we define the following set $S \subseteq \Omega$ based on P 's probability density function p to represent a soft swimmer's shape:

$$S(p) = \{x | p(x) \geq 0.5 \sup_{x \in \Omega} p(x)\}. \quad (1)$$

In other words, a soft swimmer's shape occupies the volumetric region where the probability density is over half the peak density. Similarly, the surface of the shape, which is primarily used for computing hydrodynamic forces and visualizing the shape, is defined as follows:

$$\partial S(p) = \{x | p(x) = 0.5 \sup_{x \in \Omega} p(x)\} \quad (2)$$

Shape bases. Although these probability density functions provide us a design space that can express almost all possible soft swimmer shape designs, its infinite degrees of freedom makes design optimization computationally challenging. Moreover, the probability measure space includes many physically implausible shapes that would be rejected instantly by a human user. To encourage findings of physically plausible swimmers from a low-dimensional shape space, we build a library consisting of m soft swimmers designed by human experts and define the shape space as the vector space

spanned from these designs. Specifically, let $S_1, S_2, \dots, S_m \subseteq \Omega$ be the m shapes in the library, we define the probability density function p_i for $i = 1, 2, \dots, m$ as follows:

$$p_i(x) = \begin{cases} \frac{1}{|S_i|} & \text{if } x \in S_i, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Furthermore, we use P_i to refer to the corresponding probability measures induced by p_i , which serve as the basis for shape interpolation.

Shape interpolation. With all probability measures P_i at hand, we consider the shape space parametrized by a weight vector $\alpha \in \mathcal{R}^m$ in the probability simplex $\{\alpha | \alpha_i \geq 0, \sum_i \alpha_i = 1\}$. Specifically, given a weight vector α , we compute the Wasserstein barycenter $P(\alpha)$, which can be interpreted as a weighted average of P_1, P_2, \dots, P_m [Solomon et al. 2015]:

$$P(\alpha) = \arg \min_{P \in \mathcal{P}(\Omega)} \sum_i \alpha_i \mathcal{W}_2^2(P, P_i). \quad (4)$$

Here, $\mathcal{W}_2(\cdot, \cdot) : \mathcal{P}(\Omega) \times \mathcal{P}(\Omega) \rightarrow \mathcal{R}_+$ is the 2-Wasserstein distance:

$$\mathcal{W}_2(P, Q) = \left[\inf_{\pi \in \Pi(P, Q)} \iint_{\Omega \times \Omega} d_E^2(x, y) d\pi(x, y) \right]^{\frac{1}{2}} \quad (5)$$

where $\Pi(P, Q)$ is the set of transportation maps from probability measure P to Q :

$$\Pi(P, Q) = \{\pi \in \mathcal{P}(\Omega \times \Omega) | \pi(\cdot, \Omega) = P, \pi(\Omega, \cdot) = Q\}. \quad (6)$$

In short, for each weight vector α , we compute the Wasserstein barycenter $P(\alpha)$ and use its probability density function to define a new shape $S(\alpha)$, with some abuse of notation S . This way, we

establish an isomorphic mapping from the probability simplex to the shape space. Exploring the shape space is then equivalent to navigating the continuous, low-dimensional probability simplex.

4.2 Actuator Design

In this work, we use the contractile muscle fiber model introduced by Min et al. [2019] to actuate soft swimmers. Like the shape interpolation scheme above, users first specify the geometric representation of actuators for all soft swimmers in the library. Then, our method interpolates between them to obtain actuators for shapes at Wasserstein barycenters.

Actuator representation. For a given soft swimmer, we define its actuators by a set of discrete and continuous labels. Each actuator has one discrete parameter denoting its category and a small number of continuous parameters defining its location, size, and orientation. The actuator categories supported in this work include “left fin”, “right fin”, and “caudal fin”, which indicate the actuator’s rough location (Fig. 5). Each soft swimmer has at most one actuator for each category. The actuator’s continuous parameters represent its geometric design with a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ where $\boldsymbol{\mu} \in \mathcal{R}^d$ and $\Sigma \in \mathcal{R}^{d \times d}$ are its mean and variance. Similar to the shape function S , the actuator’s shape A is defined by the locations whose probability density is over half of the peak density from $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$:

$$A_S^c(\boldsymbol{\mu}, \Sigma) = B(\{\mathbf{x} \mid \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})) \geq 0.5\}) \cap S. \quad (7)$$

Here, A_S^c stands for the volumetric region occupied by an actuator of category c in a soft swimmer whose shape is S , and $B: \Omega \rightarrow \Omega$ is a function that takes as input a d -dimensional ellipsoid and returns the minimum bounding box whose directions are aligned with the principal axes of the ellipsoid (Fig. 3). Note that the intersection with S ensures the actuator stays in the interior of S . With some abuse of notation, we use A_P^c and A_α^c to denote an actuator of category c in a soft swimmer defined by a probability measure P or the Wasserstein barycentric interpolation with a weight vector α , respectively. When designing a soft swimmer S_i in the library, the human expert also specifies its actuators’ discrete and continuous parameters to obtain $\{A_i^c\}_c$, defined as follows:

$$A_i^c = A_{S_i}^c(\boldsymbol{\mu}_i^c, \Sigma_i^c) \quad (8)$$

where $\boldsymbol{\mu}_i^c$ and Σ_i^c are prespecified mean and variance for the actuators with category c in shape S_i .

Actuator interpolation. We now describe how to generate actuators for a soft swimmer interpolated using the Wasserstein distance. Let α be the weight vector defined above. For the soft swimmer defined by the probability measure $P(\alpha)$ and for each actuator category c , we define A_α^c as follows:

$$A_\alpha^c = A_{S(\alpha)}^c(\boldsymbol{\mu}_\alpha^c, \Sigma_\alpha^c) \quad (9)$$

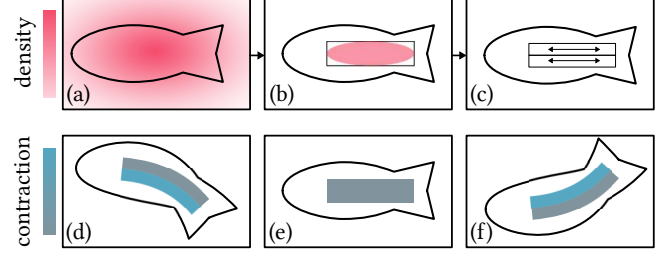


Fig. 3. We explain our actuator design with a 2D fish illustration. Sec. 4: (a) We use a multivariate normal distribution to parametrize the location of the actuators. The gradient of light pink to dark pink represents the increasing probability density function. (b) The actuator’s shape is defined by the bounding box of the probability density’s isocontour at half of its peak density. We draw a black bounding box around the clipped distribution to form the actuator. Sec. 5: (c) Our actuator is divided into a pair of antagonistic muscle fibers acting in opposition. We show the extreme deformations of the fish tail in (d) the left-most position, (e) the neutral position, and (f) the right-most position. More contraction of each muscle fiber is indicated by a darker blue. The rest shape of the muscle fiber is depicted in gray.

where $\boldsymbol{\mu}_\alpha^c$ and Σ_α^c are continuous parameters computed as follows:

$$\boldsymbol{\mu}_\alpha^c = \text{LinearInterpolation}(\alpha, \boldsymbol{\mu}_i^c), \quad (10)$$

$$\Sigma_\alpha^c = (R^c)^\top S^c R^c, \quad (11)$$

$$R^c = \text{RotationalInterpolation}(\alpha, R_i^c), \quad (12)$$

$$S^c = \text{LinearInterpolation}(\alpha, S_i^c). \quad (13)$$

Here, both $\boldsymbol{\mu}_\alpha^c$ and Σ_α^c are defined by interpolating the means and variances of the actuators from the base shapes with the weight vector α . If the actuator of category c is not used by the i -th base shape, we set the corresponding weight α_i to zero so that it is excluded from the interpolation. The mean $\boldsymbol{\mu}_\alpha^c$ is computed by linearly interpolating between the $\boldsymbol{\mu}_i^c$. The variance matrix Σ_α^c is obtained by linearly interpolating the eigenvalues S_i^c and the Euler angles of the rotational matrices R_i^c from the actuator of category c in the i -th base shapes. Note that, for simplicity, the actuators are not interpolated using the Wasserstein distance. We leave solving for the exact actuator interpolation as future work.

5 CONTROL

In this section, we first explain the actuation model built upon the actuator’s geometric design defined in the previous section. Next, we introduce two controllers for the soft underwater swimmer’s motion: an open-loop controller defined by analytic functions and a closed-loop neural network controller.

5.1 Actuation Model

As we explain in the previous section, each actuator is a cubic region defined by a clipped multivariate normal distribution. Following the muscle fiber model from previous work [Du et al. 2021; Min et al. 2019], we model each actuator as a group of parallel muscle fibers. Each muscle fiber can contract itself along the fiber direction based on the magnitude of the control signal. For the “caudal fin” actuator,

the parallel muscle fibers are placed antagonistically, allowing us to actuate bilateral flapping (Fig. 3).

5.2 Open-Loop Controller

The muscles of marine animals are usually actuated periodically like waves. Inspired by this, the first controller we consider in this work is a series of sinusoidal waves:

$$a(t) = a \sin(\omega t + \varphi) \quad (14)$$

where t stands for the time and a , ω , and φ are the control parameters to be optimized and may vary between different actuator categories. When combined with the parallel muscle fibers in each actuator, the open-loop controller generates an oscillating motion sequence from the soft swimmer's body.

5.3 Closed-Loop Controller

In addition to open-loop controllers, we also consider using closed-loop neural network controllers to achieve more precise control over a soft swimmer's motion. Our neural network controller takes sensor data as input and returns control signals used to activate the actuation of the parallel muscle fibers.

Sensing. For a soft underwater swimmer in our design space, we gather position and velocity information from a swimmer's head, center, and tail. More concretely, we first align all swimmers in the library so that their heading is along the positive x -axis. After this alignment, we ensure the heading of any interpolated design is also along the positive x -axis. We then place the head, center, and tail sensor at locations along the x -axis with the maximum, zero, and minimum x values within the swimmer's shape. We stack the outputs from all three sensors into a single vector and send it to the neural network controller.

Neural network controller. Our neural network controller is a standard multilayer perceptron network with two layers of 64 neurons. We use tanh as the activation function. The input to the network includes the velocities from all three sensors and also the positional offsets from the center sensor to the head and tail sensors. Additionally, our network also takes as input a 20-dimensional temporal encoding vector $\phi(t)$ to sense the temporally contextual information and encourage periodic control output, which is defined as follows [Vaswani et al. 2017]:

$$\phi(t) = [\sin(2^0 \pi \tau(t)), \sin(2^1 \pi \tau(t)), \dots, \sin(2^9 \pi \tau(t)), \cos(2^0 \pi \tau(t)), \cos(2^1 \pi \tau(t)), \dots, \cos(2^9 \pi \tau(t))]. \quad (15)$$

Here, $\tau: \mathcal{R}_+ \rightarrow [0, 1]$ wraps the actual t with a predefined period T : $\tau(t) = \frac{t \bmod T}{T}$. We use $T = 25h$ where h is the time step in each experiment. We concatenate the sensor feedback and the temporal encoding as a 35-dimensional vector, which is used as the input to the neural network controller. The neural network then outputs control signals for all actuators.

6 DIFFERENTIABLE SIMULATION

Given a soft swimmer's geometric design and controller, we now describe how to evaluate the swimmer's performance and gradients in a differentiable simulation environment. We build our differentiable

simulator upon Min et al. [2019] and Du et al. [2021], which use projective dynamics, a fast finite element simulation method amenable to implicit integration. We consider the geometric domain Ω as the material space and the swimmer's shape $S(p)$ as the rest shape of a deformable body. One way to forward simulate the design is to extract the surface boundary $\partial S(p)$ from the level set of its probability density function p , discretize $S(p)$ into finite elements, and simulate its motion by tracking its vertex locations. While this is a viable option for the forward simulation, this Lagrangian representation of the geometric design brings a few challenges for design optimization. First, whenever S is updated, we need to regenerate the volumetric mesh to avoid narrow finite elements, and such a geometric processing step could be computationally expensive. Second, gradient computation depends on the exact discretization of $S(p)$, but the way to partition $S(p)$ into finite elements is not unique. Therefore, picking any specific partition may bias the gradient computation unintentionally.

Spatial and time discretization. As a result, we choose to evolve the soft swimmer's motion with an Eulerian view based on the probability density function p . Specifically, we simulate the full geometric domain Ω with a spatially varying stiffness defined by p . For the volumetric region outside $S(p)$, we assign close-to-zero stiffness so that simulating it has a negligible effect on the soft swimmer's motion. More formally, we consider a uniform grid that discretizes Ω . Let n be the number of nodes in the grid and let $\mathbf{q}_i, \mathbf{v}_i \in \mathcal{R}^{dn}$ be the nodal positions and velocities at the beginning of the i -th time step. We simulate the grid based on the implicit time-stepping scheme:

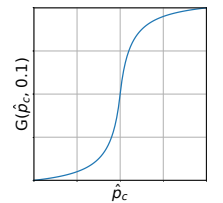
$$\mathbf{q}_{i+1} = \mathbf{q}_i + h\mathbf{v}_{i+1} \quad (16)$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + h\mathbf{M}^{-1}[\mathbf{f}_{\text{int}}(\mathbf{q}_{i+1}) + \mathbf{f}_{\text{ext}}] \quad (17)$$

where h denotes the time step, $\mathbf{M} \in \mathcal{R}^{dn \times dn}$ is the mass matrix, and $\mathbf{f}_{\text{int}}, \mathbf{f}_{\text{ext}} \in \mathcal{R}^{dn}$ are the internal and external forces applied to the n nodes. As we inherit the constitutive model and the actuator model from Min et al. [2019], we skip their implementation details and focus on explaining how our spatially varying stiffness field is used to define the constitutive model. Specifically, we define a cell-wise constant Young's modulus field $E(c)$:

$$E(c) = E_0 G(\hat{p}_c, 0.1) \quad (18)$$

where c is the cell index, E_0 is a base Young's modulus and \hat{p}_c is the discretized value of p in cell c . We further divide \hat{p}_c by its maximum value from all cells so that \hat{p}_c is between 0 and 1. $G(\cdot, 0.1): [0, 1] \rightarrow [0, 1]$ is the Schlick's function [Schlick 1994], and choosing its second argument to be 0.1 pushes the output of G towards the binary value 0 or 1 (see the inset). Recall that the soft swimmer's shape is defined at locations where p is at least half of the peak density, using $G(\cdot, 0.1)$ encourages the soft swimmer's body to have a Young's modulus close to E_0 . Additionally, it suppresses the volumetric region outside the swimmer to have a close-to-zero Young's modulus so that its effect on the swimmer's motion is negligible (Fig. 4).



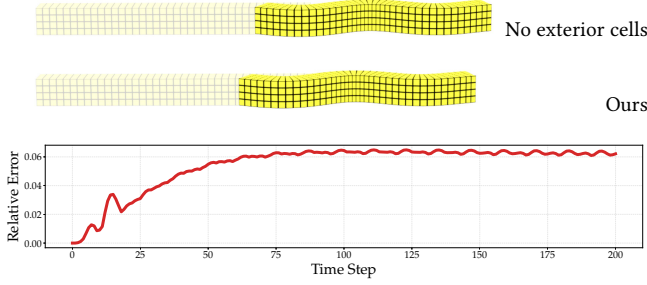


Fig. 4. To understand the effects of simulating a volumetric region outside a swimmer with a close-to-zero Young’s modulus, we simulate an eel without the volumetric region outside its shape (top) and compare it to our method (middle), which simulates the whole domain Ω but assigns small Young’s modulus to the volumetric region outside the eel. The relative error for each time step converges to around 6% after 100 time steps (bottom). We compute the relative error by computing the average difference between nodal positions from the two simulations and dividing it by the length of the eel.

We stress that the choice of simulating the entire domain Ω with spatially varying Young’s modulus brings us two key benefits. First, discretization is done trivially on the background grid. We avoid the expensive discretization process involving the level set of p , and evolving the shape design is merely updating the probability density function on the regular grid. Second, and more importantly, since a differentiable simulator provides gradients about the material stiffness, and algorithms for Wasserstein barycentric interpolation offer gradients about the probability density function [Bonneel et al. 2016], formulating the material stiffness as a function of the probability density connects the gradients from shape design and simulation into a uniform pipeline seamlessly.

Hydrodynamics. To efficiently mimic the interaction between the water and the swimmer, we develop a hydrodynamics formulation based on Min et al. [2019]. In this model, the thrust and drag forces are computed on each quadrilateral from the swimmer’s surface mesh after discretization:

$$\mathbf{f}_{\text{drag}} = \frac{1}{2} \rho A C_d(\Phi) \|\mathbf{v}_{\text{rel}}\|^2 \mathbf{d}, \quad (19)$$

$$\mathbf{f}_{\text{thrust}} = -\frac{1}{2} \rho A C_t(\Phi) \|\mathbf{v}_{\text{lat}}\|^2 \mathbf{n}, \quad (20)$$

where A is the area of the surface quadrilateral, ρ is the density of the fluid, $\mathbf{d} = \frac{\mathbf{v}_{\text{rel}}}{\|\mathbf{v}_{\text{rel}}\|}$ is the direction of the relative surface velocity, and \mathbf{n} is the surface normal. $C_d(\Phi)$ and $C_t(\Phi)$ are dimensionless drag and thrust coefficients that only depend on the angle of attack $\Phi = \cos^{-1}(\mathbf{n} \cdot \mathbf{v}_{\text{rel}}) - \frac{\pi}{2}$. The relative velocity for the surface quadrilateral is calculated as follows:

$$\mathbf{v}_{\text{rel}} = \mathbf{v}_{\text{water}} - \frac{1}{4}(\mathbf{v}_0 + \mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3), \quad (21)$$

where $\mathbf{v}_{\text{water}}$ is the velocity of the surrounding water and \mathbf{v}_0 to \mathbf{v}_3 are the velocities of the four corners of the quadrilateral. We define the lateral velocity in Eqn. (20) as follows:

$$\mathbf{v}_{\text{lat}} = \mathbf{v}_{\text{rel}} - (\mathbf{s} \cdot \mathbf{v}_{\text{rel}}) \mathbf{s}, \quad (22)$$

where \mathbf{s} is the direction of the fish spine. In our framework, \mathbf{s} is set as an unit vector pointing from fish tail to the head. The thrust and drag forces are then distributed equally to the four corners of each quadrilateral.

Lastly, the thrust force model from Min et al. [2019] creates an additional “drag-like” force even when there is no tail motion, which prevents forward swimming. To alleviate this, we modify Eqn. (20), and scale the thrust by $\|\mathbf{v}_{\text{lat}}\|^2$, a physically-based modification which is inspired by the large amplitude elongated-body theory of fish locomotion [Lighthill 1971].

Backpropagation. For a given soft swimmer’s design and controller, a loss function provides a quantitative metric for evaluating the soft swimmer’s performance. In this work, we define our loss function L on nodal positions, velocities, and hydrodynamic forces. The choices of these input arguments allow us to measure travel distance, speed, or energy efficiency. We provide the definitions of each loss function in our experiments in Sec. 8.

After we evaluate the loss function for a given design and controller, we can compute their gradients *via* backpropagation. To calculate the gradients about the design parameter α , we first obtain gradients about $E(c)$, the material stiffness inside each cell, from the differentiable simulator. We then backpropagate these gradients through Wasserstein barycentric interpolation to obtain $\frac{\partial L}{\partial \alpha}$ through the mapping function $G(\cdot, 0.1)$ as described above. In particular, $\frac{\partial L}{\partial \alpha}$ is computed using auto-differentiation as implemented by deep learning frameworks [Paszke et al. 2019]. All of our controllers are also clearly differentiable, allowing gradients with respect to control parameters to also be computed *via* backpropagation.

7 OPTIMIZATION

Our fully differentiable pipeline enables the usage of gradient-based numerical optimization algorithms in co-designing the shape, actuator shape, and controller of soft underwater swimmers. Starting with an initial guess of the soft swimmer’s geometric design and controller, we apply the Adam optimizer [Kingma and Ba 2015] with gradients computed for both design and control parameters. We terminate the optimization process when the results converge or when we exhaust a specified computational budget. Since we parametrize both geometry and control with continuous variables in the same framework, we can optimize over all variables simultaneously.

8 RESULTS

We evaluate our method’s performance with six experiments, covering shape design, co-design of geometry and control, and multi-objective design. We summarize their setup in Table 1. We run all experiments on a virtual machine instance from Google Cloud Platform with 16 Intel Xeon Scalable Processors (Cascade Lake) @ 3.1 GHz and 64 GB memory with 8 OpenMP threads in parallel.

We collect a set of meshes representing various fishes in nature as the design bases. In the pre-processing phase, we normalize and voxelize them using the same grid resolution. Grid resolutions can be found in Table 1. We use the following strategies in all our experiments to improve the numerical stability of Wasserstein barycentric interpolation: choosing basis shapes with the same topological

Table 1. A summary of all experiments mentioned in Sec. 8. The “# Bases” and “# Parameters” columns report the number of base shapes used for interpolation and the number of trainable parameters, respectively. The “Resolution” column gives the voxel resolution of base bounding boxes. The “Shape” and “Control” columns indicate whether we optimize a soft swimmer’s geometric design and controller, respectively. The “Objective” column shows the experiment’s objective type (none, single, or multiple). The “Time” column records the total time cost of optimization in minutes.

Section	Name	# Bases	# Parameters	Resolution	Shape	Control	Objective	Time (min)
8.1	Shape Exploration	3	-	$60 \times 54 \times 14$	-	-	None	-
	Shape Optimization	2	2	$60 \times 16 \times 26$	✓	-	Single	136
8.2	Open-Loop Co-Optimization	2	5	$40 \times 8 \times 8$	✓	✓	Single	60
	Closed-Loop Co-Optimization	3	7572	$60 \times 54 \times 14$	✓	✓	Single	354
	Large Dataset Co-Optimization	12	6541	$60 \times 16 \times 26$	✓	✓	Single	193
8.3	Multi-Objective Co-Optimization	12	6541	$60 \times 16 \times 26$	✓	✓	Multiple	195

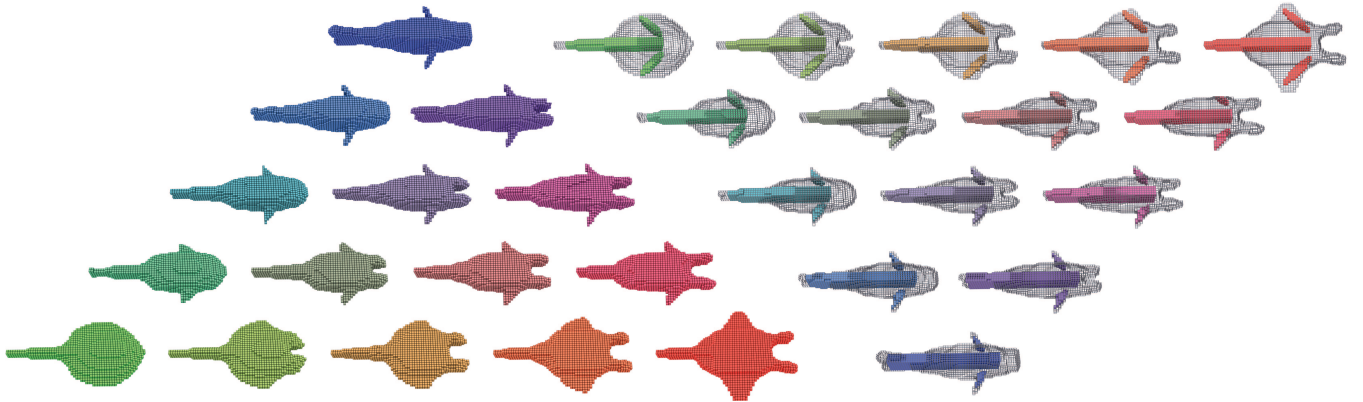


Fig. 5. “Shape exploration” experiment: Interpolation between three base shapes and their actuators. Left triangle: 12 intermediate shapes interpolated from three base shapes (red, green, and blue shapes at the corners). Right triangle: the corresponding actuator placements (colored regions) for the 12 intermediate shapes shown as wire frames, also generated by interpolating actuators in the three base shapes at the corners.

genus, aligning their centers of mass, and running the interpolation iteration until convergence.

8.1 Design Space Exploration

Shape exploration. In this experiment, we demonstrate the capability of our shape interpolation scheme without considering optimization. In Fig. 5, we show shape and actuator interpolation between three morphologically different hand-designed base shapes from our library: a clownfish, a manta ray, and a stingray. We see from Fig. 5 that the Wasserstein barycentric interpolation is capable of generating smooth and biologically plausible intermediate shapes. Additionally, we note that our actuator interpolation generates designs adaptive to the shape’s size, as can be seen from the change of muscle size in the left and right fins.

Shape optimization. In this experiment, we demonstrate the power of our shape interpolation scheme when used in tandem with our differentiable simulator for optimizing shape design and actuator placement. We choose a shark with a tall tail and an orca with a flat tail as the base shapes. The control input is a prespecified open-loop sine wave leading to oscillatory motions on the horizontal plane

(spanned by the x - and y -axes in our coordinate system), which is ill-suited for the orca but natural for the shark. We initialize the open-loop controller with $a = 1$, $\omega = \frac{\pi}{6h}$, and $\varphi = 0$. The objective is to find a geometric design that traverses the longest forward distance in a fixed time period Nh , where N is the number of time steps and h the time interval. We purposefully choose the shark and orca bases since we know *a priori* that the shark will outperform the orca in this task with the prespecified controller. Therefore, this experiment also serves as a small test for our pipeline. Formally, the loss function is defined as

$$L = -\frac{1}{|Sp|} \sum_{j \in Spine} (\mathbf{q}_{N,j})_x - (\mathbf{q}_{0,j})_x \quad (23)$$

$$Spine = \{j | (\mathbf{q}_{0,j})_y = 0\} \quad (24)$$

where $\mathbf{q}_{i,j} \in \mathcal{R}^d$ is the j -th node’s location at the i -th time step and $(\cdot)_x$ and $(\cdot)_y$ extracts its x and y coordinate, respectively. The objective encourages faster swimming speed along the desired heading direction, which is defined as the x -axis in all our experiments. We estimate the swimmer’s x -offset by averaging the x -offsets between the first and last time step from nodes near the spine of the swimmer,

which is formally defined by a *Spine* set consisting of nodes whose y coordinate (the lateral offset) is zero when undeformed.

Fig. 6 summarizes our results in this experiment. We run our optimization algorithm with two different initial weight vectors α : $\alpha = (0, 1)$ which puts all weights in the orca (Fig. 6 top) and $\alpha = (0.5, 0.5)$ (Fig. 6 middle). As the control signal is known to be ill-suited for the orca, we expect the orca to have a poor performance, as is correctly reflected in the top row of Fig. 6. The optimized results from both starting shapes align with our expectation of the shark as the optimal shape for this task (Fig. 6 bottom). Our algorithm robustly converges to the optimal solution in both cases.

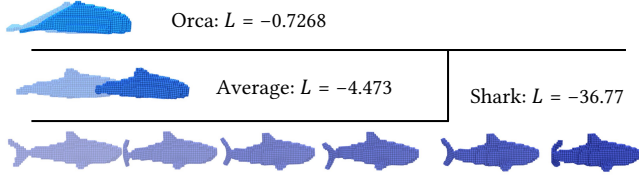


Fig. 6. “Shape optimization” experiment: Performances of two initial shapes and the optimized shape. Top: the orca; Middle: the average shape between the orca and the shark; Bottom: the optimized shape found by our method, which is similar to the shark. The color on the shape indicates the weight on the two base shapes (light blue: orca, dark blue: shark). A smaller loss L indicates a longer traveling distance within the same duration and is preferred in this experiment.

8.2 Co-Design of Shape and Control

Open-loop co-optimization. We now present our first co-design example, illustrating the value of our method compared to other baseline methods. The case we consider is co-optimization of both the shape and controller of an eel for the same objective described in the “shape optimization” experiment. The base shapes are two eels, *i.e.*, slender bodies with their undulations’ wavelengths smaller than their length. One *vertical* eel is flag-like with a greater height than width, and the other *horizontal* eel is pancake-like with a greater width than height. We use the same open-loop sine wave control sequence as in the “shape optimization” experiment, except that we leave its amplitude, phase, and frequency as variables to be optimized from the initial guess with 0.5, 0, and $\frac{\pi}{6h}$ respectively. The decision variables for this co-optimization problem are four-dimensional, including one geometric design parameter and three control parameters. The goal is to find both an optimal shape *and* an optimal controller that leads to the longest distance traveled within a fixed time span. Fig. 7 shows the initial design (top row) and the optimized design (bottom row) returned by our algorithm. Since the actuation is manifested in form of a sine-wave on the horizontal plane, the optimal shape should be a vertical eel. As expected, our co-optimization algorithm correctly finds such a physical design, as well as an intensified control signal to maximize traveling distance.

For comparison, we also examine the performances of a few baseline algorithms: **alt**: alternating between shape and control optimization; **shape-only**: fixing the initial controller and optimizing the shape; **control-only**: fixing the initial shape and optimizing the controller; **cma-es**: co-optimizing both shape and controller with

CMA-ES [Hansen et al. 2003], a gradient-free evolutionary algorithm. By comparing the loss-iteration curves from all of these methods (Fig. 8), we reach the following conclusions: First, co-optimizing both the shape and controller (**ours**, **alt**, and **cma-es**) reaches a much lower loss than only optimizing shape or control (**shape-only** and **control-only**), which is as expected. Second, gradient-based co-optimization (**ours** and **alt**) converges significantly faster than the gradient-free baseline **cma-es**, demonstrating the value of gradients in design optimization. Lastly, we find that simultaneously optimizing both shape and controller (**ours**) converges to similar results but faster than the alternating strategy **alt**. This observation highlights the value of our differentiable geometric design space, which makes simultaneously co-optimizing shape and control with a gradient-based algorithm possible.

Closed-loop co-optimization. In this experiment, we replace the open-loop controller from the last experiment with a closed-loop neural network controller. We also use the three base shapes in the “shape exploration” experiment (Fig. 5) for shape interpolation. The objective is the same as in the previous experiment.

Fig. 9 and Fig. 10 show the optimal shape and controller from our method. We observe that the optimal shape assigns more weight to the clownfish and the stingray than the manta ray, likely the clownfish has a larger tail. For the controller, we also notice that the optimal neural network controller discovers a strongly oscillating pattern for the tail (Fig. 10), which aligns with our expectation for a fast swimmer. We point out that the source of periodicity in the control signal is from the temporal encoding technique: As shown in Fig. 10, learning a periodic control output becomes much more difficult when temporal encoding is disabled, leading to much worse performance (Fig. 11).

To show the value of co-optimizing both shape and control in this experiment (rather than just control), we test fixing the shape design to the three bases and optimizing the controller only. We present the loss-iteration curves for these methods in Fig. 11 and conclude that co-optimizing both the shape and control achieves significantly better performance.

Large dataset co-optimization. In a more realistic example of design pertinent to roboticists, we consider a position-keeping task in the face of an external disturbance from a constant water flow. We feed our shape interpolation a plethora of bases, including four shark variations, seven goldfish variations, and one submarine (Fig. 1). We again use a closed-loop controller as in the Experiment 4. The swimmer’s goal is to maintain its position and orientation in a stream of fast-flowing water moving against its head. Formally, our loss L is defined as follows:

$$L = L_{\text{perf}} + \gamma L_{\text{reg}} \quad (25)$$

$$L_{\text{perf}} = \sum_i \| \mathbf{q}_{i,c} - \mathbf{q}_{\text{target}} \|_1 \quad (26)$$

$$L_{\text{reg}} = - \sum_i (\mathbf{q}_{i,h_0} - \mathbf{q}_{i,h_1}) \cdot \mathbf{d}_{\text{target}}. \quad (27)$$

Here, the loss function consists of a performance loss L_{perf} and a regularizer L_{reg} . We set the regularizer weight $\gamma = 0.01$. The performance loss is defined as the cumulative offset of a node $\mathbf{q}_{i,c} \in$

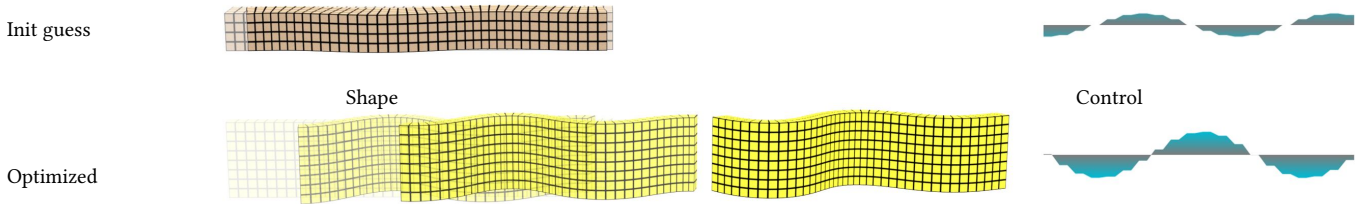


Fig. 7. “Open-loop co-optimization” experiment: The initial guess (top) and the optimized design (bottom) of the shape and control. The shapes are simulated to swim forward with the parameterized sinusoidal controller (right) from its initial position (transparent) to the final location (solid) within a fixed time period.

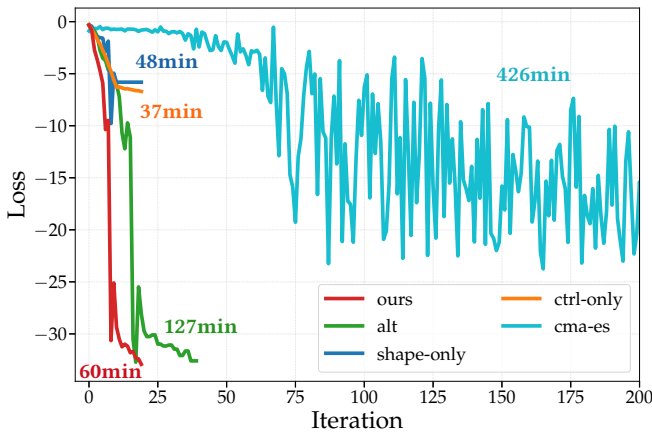


Fig. 8. The loss-iteration curves for our method and four baseline algorithms evaluated in the “open-loop co-optimization” experiment. Lower losses are better. The CMA-ES loss continually oscillates and does not reach a loss lower than our method before exhausting its computational budget (250 iterations, not shown in the figure). We label the total time cost for each method with its corresponding color.

\mathcal{R}^d at the i -th timestep with respect to a target location $\mathbf{q}_{\text{target}} = \mathbf{0}$. Here c is the index of the central node in the aforementioned *Spine* set. In short, the performance loss encourages the swimmer to stay at the origin within the given time period. Additionally, we introduce the regularizer loss to penalize controllers circling around $\mathbf{q}_{\text{target}}$. Here, h_0 and h_1 are indices of two prespecified nodes from the *Spine* set. Therefore, $\mathbf{q}_{i,h_0} - \mathbf{q}_{i,h_1}$ estimates the swimmer’s heading at the i -th timestep. The regularizer computes the dot product between the true heading and a target heading $\mathbf{d}_{\text{target}}$, which is the positive x unit vector, to encourage a controller that maintains orientation along the positive x direction.

We report the optimal shape and controller from our method in Fig. 1. With co-optimized shape and control, the swimmer learns to leverage an oscillating motion to counter the flow and stabilize itself. We use an average of all shape bases as the initial shape for optimization. The swimmer’s shape after optimization appears mildly different from the initial guess (Fig. 1 middle and right), but



Fig. 9. “Closed-loop co-optimization” experiment: Performances of the initial (top) and the optimized geometric design (bottom) of the swimmer. The color of the swimmer is interpolated with their weights α from the base shapes’ colors in Fig. 5. The design is simulated to swim forward from an initial position (transparent) on the right. The swimmer’s final locations after a fixed amount of time are rendered as solid meshes. A longer traveling distance is preferred in this experiment.

that difference has a significant impact on performance. Further, the difference between the initial and optimized control signals are quite noticeable. In particular, the optimizer learns to intensify the magnitude of the control signal to counter the flow. To justify the importance of shape optimization, we compare our method to optimizing controllers with the shape fixed as each of the 12 bases. As shown in Fig. 12, our co-optimized swimmer outperforms all 12 base swimmers by a clear margin, highlighting the necessity of co-optimizing both the geometry and the control of the swimmer.

8.3 Multi-Objective Design

Multi-objective co-optimization. Finally, we employ our method to investigate a multi-objective design problem: What is the optimal design of a swimmer for both fast *and* efficient forward swimming? These two objectives often conflict with each other for real marine creatures [Sfakiotakis et al. 1999]; therefore, they define a gamut of designs with varying preferences on these two objectives and an interesting Pareto front.

Formally, we consider the same set of shape bases as in the “Large dataset co-optimization” experiment with two loss functions L_{speed} and $L_{\text{efficiency}}$. We use the loss function in Eqn. (23) for L_{speed} , which reaches its minimum when the swimmer obtains the maximum

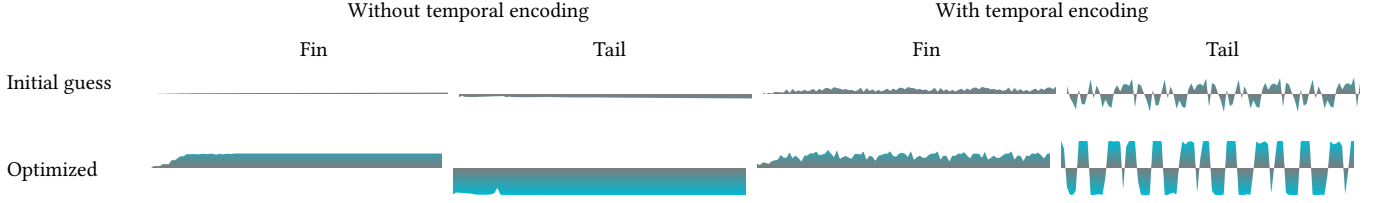


Fig. 10. The initial and optimized control signals generated by running our method with and without temporal encoding in the “closed-loop co-optimization” experiment.

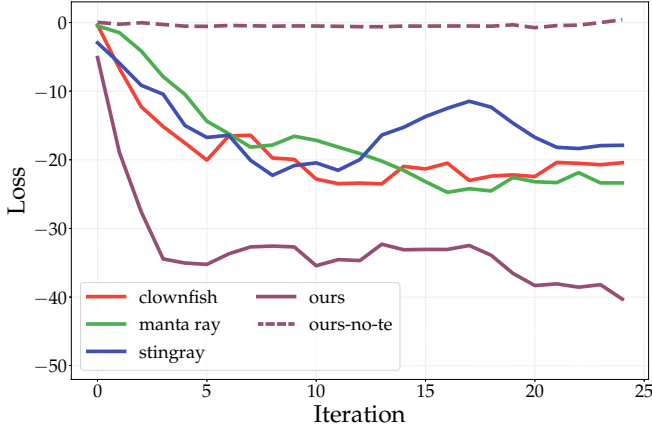


Fig. 11. The loss-iteration curves for five different methods in the “closed-loop co-optimization” experiment: The curves labeled as “ours” and “ours-no-te” represent losses from our methods with and without temporal encoding. The “clownfish”, “manta ray”, and “stingray” curves record the intermediate losses from optimizing controllers with shapes fixed to the corresponding base shape. Lower losses are better.

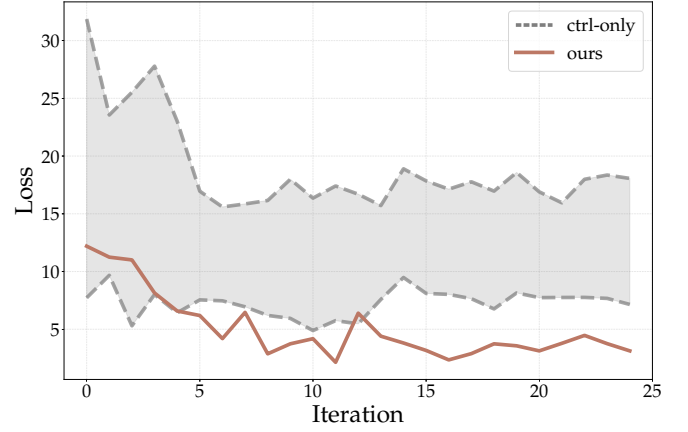


Fig. 12. The loss-iteration curves for our method (ours) and optimizing control parameters with the shape fixed to each of the 12 base shapes (ctrl-only). A lower loss is better. We report the aggregate results about the 12 control-only optimizations (whose loss curves all fall in the gray area), with the lower and upper bounds of the loss curves highlighted by dashed lines.

average forward speed over a given time. The efficiency loss is defined as follows:

$$L_{\text{efficiency}} = - \sum_i \frac{|p_{\text{thrust}}|}{1 + |p_{\text{thrust}}| + |p_{\text{drag}}|} \quad (28)$$

$$= - \sum_i \frac{|\bar{f}_i^{\text{thrust}} \cdot \bar{v}_i|}{1 + |\bar{f}_i^{\text{thrust}} \cdot \bar{v}_i| + |\bar{f}_i^{\text{drag}} \cdot \bar{v}_i|}. \quad (29)$$

In short, $L_{\text{efficiency}}$ provides a measure of the energy dissipation due to the hydrodynamic drag, and minimizing $L_{\text{efficiency}}$ encourages a more efficient usage of the hydrodynamic force. We use P_{thrust} and P_{drag} to denote the power of hydrodynamic thrust and drag, respectively. At the i -th time step, we define the hydrodynamic force’s power p_{thrust} as the dot product between the average hydrodynamic force $\bar{f}_i^{\text{thrust}}$ on the swimmer’s surface and the average velocity \bar{v}_i computed from nodes in the *Spine* set. The hydrodynamic drag’s power p_{drag} is defined similarly. Finally, we add 1 in the denominator to avoid singularities, which occurs when both the water and the swimmer are still.

To understand the implications of different geometry and controller designs on the two losses, we generate a gamut of swimmers and visualize their performances in the $L_{\text{speed}}-L_{\text{efficiency}}$ space (Fig. 13 left). Note that lower losses are better, so designs closer to the lower-left corner are preferred. To generate this gamut, we optimize the weighted sum of the two losses $w_s L_{\text{speed}} + w_e L_{\text{efficiency}}$ with $w_s = 0, 0.1, 0.2, \dots, 1$ and $w_e = 1 - w_s$. We record all intermediate designs discovered during this process to form the gamut, with the designs on the Pareto front highlighted as white circles.

To understand the Pareto optimal designs, we sampled four swimmers along the Pareto front. Fig. 13 further shows the geometric design and neural network control outputs for the sampled swimmers. We find that the four swimmers’ shape designs are quite similar, but their controllers are significantly different. In particular, their control signals show a strong correlation with the losses: controllers preferred by faster swimmers show larger magnitudes lasting for a longer period, which exerting more powerful forces from the muscle fibers in the actuators. These sampled four swimmers, along with many others discovered in the Pareto front, form a diverse set of designs for users to choose from in case they have varying preferences.

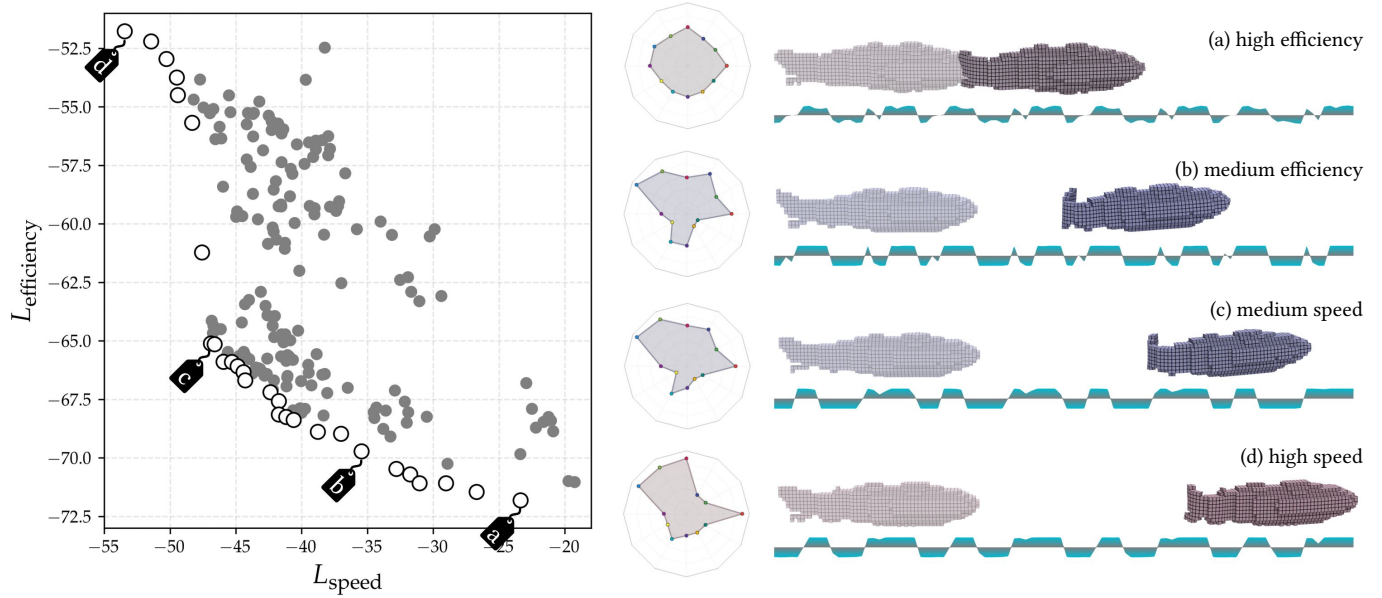


Fig. 13. “Multi-objective co-optimization” experiment: Left: The performance gamut of the intermediate discovered designs (gray dots) and its Pareto front (white circles). Lower losses are better. Right: We show four samples, labeled as (a), (b), (c), and (d), from the Pareto front. For each sample, we show the swimmer’s initial (transparent) and final (solid) location at the beginning and end of the simulation. A larger distance between these two locations indicate a faster average speed (better L_{speed}). The design parameters for each sample are shown in the radar chart to its left. We plot the control signals for each sample below its initial and final locations.

8.4 Ablation Study

Gradient Scaling. The Adam optimizer we use in our experiments is a first-order gradient descent optimizer. Since such algorithms are generally not scale-invariant and we co-optimize parameters from two very different categories (geometry and control), the possible imbalance between the scale of geometry and control parameters may affect the optimizer’s performance. To examine the impact of different scales, we rerun the “closed-loop co-optimization” experiment by scaling the geometry and control parameters in three different settings: First, the default setting repeats the experiment with no changes, in which case we notice the ratio between the gradient from each geometry and control parameter is roughly 3:1. Next, in the balanced setting, we rescale the control parameters by roughly a factor of 3 so that their gradients have magnitudes comparable to those from the geometry parameters. Finally, in the reversed setting, we further rescale the control parameters until the ratio between the geometry and control gradients becomes 1:3, the reciprocal of the ratio in the default setting. We report the training curves in Fig. 14 (left), from which we notice a substantial effect from rescaling these parameters as expected. Using a scale-invariant optimizer instead of Adam could be a potential solution in the future.

Initial Guesses. To better understand the influence of different initial guesses on the performance of our method, we rerun the “large dataset co-optimization” experiment with ten randomly sampled initial shapes and controllers. Fig. 14 (right) reports the resulting ten training curves, from which we observe a consistent decrease in the loss function across all initial guesses. Still, we notice that not all

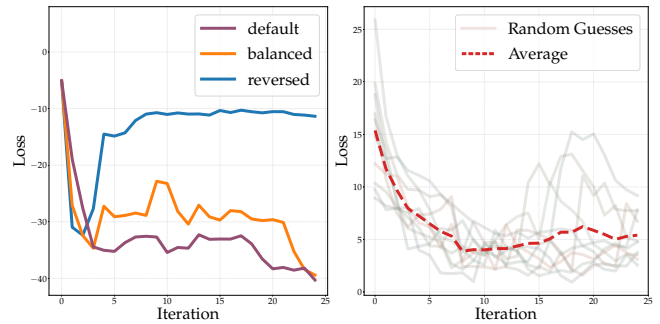


Fig. 14. The loss-iteration curves for ablation studies on the “gradient scaling” experiment (left) and the “initial guesses” experiment (right) in Sec. 8.4. Lower losses are better. Left: The “default” curve reports the result without scaling the gradients. The “balanced” and “reversed” curves represent the strategies normalizing the gradients to the same average L1-norm and switching the scales, respectively. Right: The semitransparent curves represent the losses from different initial guesses. The red dashed line reports the average loss.

random guesses converge to the same optimal solution, and some of them are trapped in different local minima. Such results are not surprising as our gradient-based method is inherently a continuous local optimization algorithm. More advanced global search algorithms might alleviate the issue of local minima, which we consider as future work.

9 CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

We have presented a method for co-optimizing soft swimmers over control and complex geometry. By exploiting differentiable simulation and control and a basis space governed by the Wasserstein distance, we are able to generate biomimetic forms that can swim quickly, resist disturbance, or save energy. Further, we have generated designs that are Pareto-optimal in two conflicting design objectives. Our co-optimization procedure outperforms optimizing over each domain independently, demonstrating the tight interrelation of form and control in swimmer behavior.

While we have provided a first foray into the computational design of soft underwater swimmers, a number of interesting problems remain ripe for exploration. First, while our algorithm was able to interpolate between actuator shapes, those actuators were placed manually on the base shapes. A method for automating the design of muscle-based actuators for soft swimmers would be interesting. Second, our algorithm requires that the base shapes themselves be chosen by hand — it would be interesting to investigate methods for extending the morphological search beyond the Wasserstein basis space. For example, it can be combined with discrete, composition-based design methods, *e.g.*, Zhao et al. [2020], to explore combinatorial design space. Third, certain aspects of soft swimmer design — *e.g.* sensing and material selection — were untouched in this work. Fourth, our simulator can be made more physically realistic by handling environmental contact as well as employing computational fluid dynamics rather than our analytical hydrodynamic model. Fifth, our work presented here investigated only virtual swimmers. It would be interesting to fabricate their physical counterparts, and research methods for overcoming the likely sim-to-real gap for physical soft swimmers. Sixth, the objectives of optimization in our experiments cover only travel distance, the ability for position maintenance, and swimming efficiency. It will be exciting to extend our algorithm to more complex goals, *e.g.*, stability under non-constant current or controllability over a target trajectory. Finally, as our optimization scheme is a local, gradient-based method, there is no guarantee for global optimality. It would be interesting to see if our gradient-based optimization could be combined with more global heuristic searches (such as simulated annealing or evolutionary algorithms) to reap the benefits of both approaches.

ACKNOWLEDGMENTS

We thank Yue Wang for the valuable discussion on the Wasserstein barycentric interpolation. We also thank the anonymous reviewers for their constructive comments. This work is supported by Intelligence Advanced Research Projects Agency (grant No. 2019-19020100001) and Defense Advanced Research Projects Agency (grant No. FA8750-20-C-0075).

REFERENCES

- Melinos Averkiou, Vladimir G Kim, Youyi Zheng, and Niloy J Mitra. 2014. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 125–134.
- Seung-Yeob Baek, Jeonghun Lim, and Kunwoo Lee. 2015. Isometric shape interpolation. *Computers & Graphics* 46 (2015), 257–263.
- Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable object animation using reduced optimal control. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–9.
- Jernej Barbič and Jovan Popović. 2008. Real-time control of physically based simulations using gentle forces. *ACM transactions on graphics (TOG)* 27, 5 (2008), 1–10.
- Filipe de A Belbute-Peres, Kevin A Smith, Kelsey R Allen, Joshua B Tenenbaum, and J Zico Kolter. 2018. End-to-end differentiable physics for learning and control. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 7178–7189.
- Florian Berlinger, Mihai Duduta, Hudson Gloria, David Clarke, Radhika Nagpal, and Robert Wood. 2018. A modular dielectric elastomer actuator to drive miniature autonomous underwater vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3429–3435.
- Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. 2016. Wasserstein barycentric coordinates: histogram regression using optimal transport. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.
- Alexander M Bronstein, Michael M Bronstein, Leonidas J Guibas, and Maks Ovsjanikov. 2011. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)* 30, 1 (2011), 1–20.
- Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 6572–6583.
- Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. 2014. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOLUTION* 7, 1 (2014), 11–23.
- Francesco Corucci, Nick Cheney, Hod Lipson, Cecilia Laschi, and Josh Bongard. 2016. Evolving swimming soft-bodied creatures. In *ALIFE XV, The Fifteenth International Conference on the Synthesis and Simulation of Living Systems, Late Breaking Proceedings*, Vol. 6.
- Jonas Degraeve, Michiel Hermans, Joni Dambre, and Francis Wuyffels. 2019. A Differentiable Physics Engine for Deep Learning in Robotics. *Frontiers in Neurorobotics* 13 (2019), 6.
- Cosimo Della Santina, Robert K Katzschmann, Antonio Biechi, and Daniela Rus. 2018. Dynamic control of soft robots interacting with the environment. In *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 46–53.
- Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational multicopter design. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–10.
- Tao Du, Kui Wu, Pingchuan Ma, Sebastian Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. DiffPD: Differentiable Projective Dynamics with Contact. *arXiv preprint arXiv:2101.05917* (2021).
- FE Fish and George V Lauder. 2006. Passive and active flow control by swimming fishes and mammals. *Annu. Rev. Fluid Mech.* 38 (2006), 193–224.
- Noa Fish, Melinos Averkiou, Oliver Van Kaick, Olga Sorkine-Hornung, Daniel Cohen-Or, and Niloy J Mitra. 2014. Meta-representation of shape families. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.
- Thomas Geijtenbeek, Michiel Van De Panne, and A Frank Van Der Stappen. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–11.
- Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.
- Markus Gifftthaler, Michael Neunert, Markus Stäuble, Marco Frigerio, Claudio Semini, and Jonas Buchli. 2017. Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics* 31, 22 (2017), 1225–1237.
- Radek Grzeszczuk and Demetri Terzopoulos. 1995. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 63–70.
- David Ha. 2019. Reinforcement learning for improving agent design. *Artificial life* 25, 4 (2019), 352–365.
- Sehoon Ha, Stelian Coros, Alexander Alspach, James M Bern, Joohyung Kim, and Katsu Yamane. 2018. Computational design of robotic devices from high-level motion specifications. *IEEE Transactions on Robotics* 34, 5 (2018), 1240–1251.
- Sehoon Ha, Stelian Coros, Alexander 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*, Vol. 8.
- David Hahn, Pol Banzet, James M Bern, and Stelian Coros. 2019. Real2sim: Visco-elastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–13.
- Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation* 11, 1 (2003), 1–18.
- Chris Hecker, Bernd Raabe, Ryan W Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. 2008. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 1–11.
- Philipp Holl, Nils Thuerey, and Vladlen Koltun. 2020. Learning to Control PDEs with Differentiable Physics. In *International Conference on Learning Representations*.

- Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. *International Conference on Learning Representations* (2020).
- Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. 2019. ChainQueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*. IEEE, 6265–6271.
- Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B. Tenenbaum, and Chuang Gan. 2021. PlasticineLab: A Soft-Body Manipulation Benchmark with Differentiable Physics. In *International Conference on Learning Representations*.
- Robert K Katzschmann, Cosimo Della Santina, Yasunori Toshimitsu, Antonio Bicchi, and Daniela Rus. 2019a. Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 454–461.
- Robert K Katzschmann, Joseph DelPreto, Robert MacCurdy, and Daniela Rus. 2018. Exploration of underwater life with an acoustically controlled soft robotic fish. *Science Robotics* 3, 16 (2018).
- Robert K Katzschmann, Maxime Thieffry, Olivier Goury, Alexandre Kruszewski, Thierry-Marie Guerra, Christian Duriez, and Daniela Rus. 2019b. Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 717–724.
- Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.
- John P Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Frederic H Pighin, and Zhigang Deng. 2014. Practice and Theory of Blendshape Facial Models. *Eurographics (State of the Art Reports)* 1, 8 (2014), 2.
- Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua Tenenbaum, and Antonio Torralba. 2019. Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids. In *International Conference on Learning Representations*.
- Junbang Liang, Ming Lin, and Vladlen Koltun. 2019. Differentiable Cloth Simulation for Inverse Problems. In *Advances in Neural Information Processing Systems*. 772–781.
- Michael James Lighthill. 1971. Large-amplitude elongated-body theory of fish locomotion. *Proceedings of the Royal Society of London. Series B. Biological Sciences* 179, 1055 (1971), 125–138.
- Pingchuan Ma, Yunsheng Tian, Zherong Pan, Bo Ren, and Dinesh Manocha. 2018. Fluid directed rigid body control using deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.
- Andrew D Marchese, Cagdas D Onal, and Daniela Rus. 2014. Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators. *Soft robotics* 1, 1 (2014), 75–87.
- Andrew D Marchese, Russ Tedrake, and Daniela Rus. 2016. Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator. *The International Journal of Robotics Research* 35, 8 (2016), 1000–1019.
- Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)* 23, 3 (2004), 449–456.
- Vittorio Megaro, Bernhard Thomašewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive design of 3d-printable robotic creatures. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–9.
- Sehee Min, Jungdam Won, Seunghwan Lee, Jungnam Park, and Jehee Lee. 2019. SoftCon: simulation and control of soft-bodied animals with biomimetic actuators. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.
- Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J Mitra, and Leonidas J Guibas. 2019. StructureNet: hierarchical graph networks for 3D shape generation. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–19.
- Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.
- Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 165–174.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems* 32 (2019), 8026–8037.
- Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. 2019. Learning to Control Self-Assembling Morphologies: A Study of Generalization via Modularity. *Advances in Neural Information Processing Systems* 32 (2019), 2295–2305.
- Jovan Popović, Steven M Seitz, and Michael Erdmann. 2003. Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics (TOG)* 22, 4 (2003), 1034–1054.
- Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. 2020. Scalable Differentiable Physics for Learning and Control. In *International Conference on Machine Learning*.
- Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover’s distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.
- Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. 2020. Learning to Simulate Complex Physics with Graph Networks. In *International Conference on Machine Learning*.
- Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. 2019. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 9798–9805.
- Connor Schenck and Dieter Fox. 2018. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks. *Conference on Robot Learning (CoRL)* (2018).
- Christophe Schlick. 1994. Fast Alternatives to Perlin’s Bias and Gain Functions. *Graphics Gems 4* (1994), 401–404.
- Adriana Schulz, Ariel Shamir, Ilya Baran, David IW Levin, Pitchaya Sithi-Amorn, and Wojciech Matusik. 2017a. Retrieval on parametric shape collections. *ACM Transactions on Graphics (TOG)* 36, 1 (2017), 1–14.
- Adriana Schulz, Cynthia Sung, Andrew Spielberg, Wei Zhao, Robin Cheng, Eitan Grinspun, Daniela Rus, and Wojciech Matusik. 2017b. 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.
- Michael Sfakiotakis, David M Lane, and J Bruce C Davies. 1999. Review of fish swimming modes for aquatic locomotion. *IEEE Journal of oceanic engineering* 24, 2 (1999), 237–252.
- Karl Sims. 1994. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 15–22.
- Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. 2015. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.
- Andrew Spielberg, Brandon Araki, Cynthia Sung, Russ Tedrake, and Daniela Rus. 2017. Functional co-optimization of articulated robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5035–5042.
- Andrew Spielberg, Allan Zhao, Yuanming Hu, Tao Du, Wojciech Matusik, and Daniela Rus. 2019. Learning-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations. *Advances in Neural Information Processing Systems* 32 (2019), 8284–8294.
- Jie Tan, Yuting Gu, Greg Turk, and C Karen Liu. 2011. Articulated swimming creatures. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 1–12.
- Maxime Thieffry, Alexandre Kruszewski, Christian Duriez, and Thierry-Marie Guerra. 2018. Control Design for Soft Robots Based on Reduced-Order Model. *IEEE Robotics and Automation Letters* 4, 1 (2018), 25–32.
- Michael S Triantafyllou and George S Triantafyllou. 1995. An efficient swimming machine. *Scientific american* 272, 3 (1995), 64–70.
- Merel Van Diepen and Kristina Shea. 2019. A spatial grammar method for the computational design synthesis of virtual soft locomotion robots. *Journal of Mechanical Design* 141, 10 (2019).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.
- Kevin Wampler and Zoran Popović. 2009. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–8.
- Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. 2018. Neural Graph Evolution: Towards Efficient Automatic Robot Design. In *International Conference on Learning Representations*.
- Jungdam Won and Jehee Lee. 2019. Learning body shape variation in physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.
- Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. 2019. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4541–4550.
- Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2020. RoboGrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.