An End-to-End Differentiable Framework for Contact-Aware Robot Design

Jie Xu[†], Tao Chen[†], Lara Zlokapa[†], Michael Foshey[†], Wojciech Matusik[†], Shinjiro Sueda[‡] and Pulkit Agrawal[†]

[†]Massachusetts Institute of Technology

[‡]Texas A&M University

http://diffhand.csail.mit.edu

Abstract—The current dominant paradigm for robotic manipulation involves two separate stages: manipulator design and control. Because the robot's morphology and how it can be controlled are intimately linked, joint optimization of design and control can significantly improve performance. Existing methods for co-optimization are limited and fail to explore a rich space of designs. The primary reason is the trade-off between the complexity of designs that is necessary for contact-rich tasks against the practical constraints of manufacturing, optimization, contact handling, etc. We overcome several of these challenges by building an end-to-end differentiable framework for contact-aware robot design. The two key components of this framework are: a novel deformation-based parameterization that allows for the design of articulated rigid robots with arbitrary, complex geometry, and a differentiable rigid body simulator that can handle contact-rich scenarios and computes analytical gradients for a full spectrum of kinematic and dynamic parameters. On multiple manipulation tasks, our framework outperforms existing methods that either only optimize for control or for design using alternate representations or co-optimize using gradient-free methods.
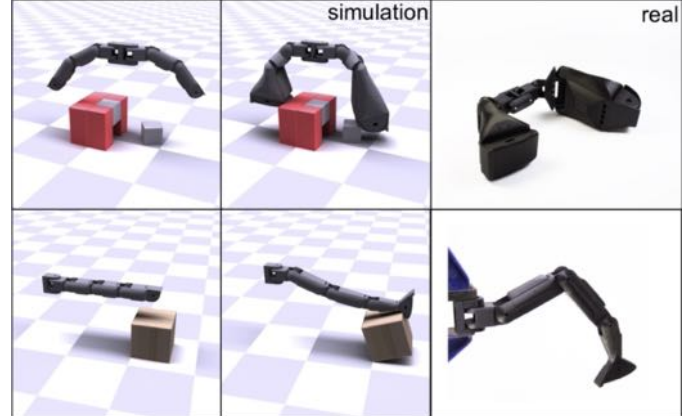
Fig. 1. *Left column*: only optimizing the control algorithm using a nominal robot design fails to complete the task; *Middle*: co-optimization of morphology and control results in success; *Right:* pictures of 3D-printed manipulators. Our method outputs designs that are easy to print and assemble.

I. INTRODUCTION

The design, control, and construction of manipulators is the cornerstone of robotics. Today this process is manual and time-consuming as concurrent design of many different components is required. For example, hardware components and control algorithms are typically constructed sequentially making the integration of different modules difficult which necessitates many design iterations. Ensuring that the designed manipulator meets the desired specifications is challenging since there is a complex interplay between the robot design, manufacturing constraints, and the control algorithm.

Due to the long iteration cycle, in practice, roboticists either (i) explore a rich design space, but make use of simple control algorithms [13, 45] or (ii) develop complex algorithms to control existing robots [1, 4, 38]. The end result is a sub-optimal system for the given task. Co-optimizing both the design and the control scheme can significantly improve the performance of today's robotic systems. One challenge in co-optimization is the substantial increase in the number of parameters to be optimized. The other and arguably more significant challenge is in defining a representation of the robot design that is amenable to optimization. A good representation should: (a) result in designs that can be manufactured; (b)

enable design of articulated robots with complex geometric shapes; and (c) allow the use of powerful optimization methods such as gradient descent.

Several recent works have studied the design representation problem [17, 18, 34, 46, 54]. One strategy is to represent the design as a graph, where each edge denotes connection between two components of the robot [17, 54]. However, just defining the topology is insufficient: it is also necessary to parameterize the shape of each component. While there are many ways to represent the shape of a single object, the shape representation of an articulated system is a challenge. It is because the shape representation must span a rich space of geometric designs while simultaneously satisfying connectivity constraints between components and the manufacturing limitations. A popular strategy is to model each component by a simple primitive shape (*i.e.*, cylinder, cuboids, etc.) [17, 24, 34, 41, 46]. However, while easy to optimize, these primitives are often an over-simplification of the desired shape and are insufficient to model complex gripper designs as shown in Figure 1.

An alternative to basic shape primitives is the CAD parameterization. However, this approach has several downsides: generating models from CAD parameterization is slow [47], and updates in CAD model parameters often results in failures such as the model being disconnected or even failure in gener-

ating the model. Other options for rich shape representations are voxel grid, point cloud, signed distance functions [18], etc. Unfortunately, these are not suitable for articulated robot design because it is hard to impose connectivity constraints between individual components. A drastic alternative to overcoming the connectivity problem without compromising rich shape parameterization is to directly optimize the shape of the entire robot instead of individual components. But this does not solve the problem either because now identifying individual joints and components, which is necessary for simulation and manufacturing, becomes a problem. Another consideration worth discussing is that for learning to control one must first import the shape into the simulator. While CAD designs can be easily imported they do not provide analytical gradients. Other representations discussed above must be first converted to a mesh, a step that is non-differentiable. Without analytical gradients one must rely on data inefficient gradient free methods for solving the shape optimization problem.

In this paper, we study the effective parameterization for the continuous shape morphology assuming a fixed robot topology. We propose a general morphology representation for articulated robot designs based on cage-based deformation (CBD) models. Cage-based deformations [29, 31] have been widely used in computer graphics to deform a mesh through a few number of cage "handles" (*i.e.*, cage vertices) in real-time while preserving local geometric features, as shown in Figures 2 and 4. Instead of specifying a large number of optimization parameters for modelling complex shapes, CBD maintains an *expressive* shape design space with only a few parameters. Furthermore, as we will describe in Section III-A the cage representation allows imposition of a rich set of manufacturing and connectivity constraints. Most importantly, CBD is not tied to a specific shape representation and can be easily used with different representations such as meshes, point clouds etc. It is computationally inexpensive for inference, flexible (*i.e.*, user can easily control the degrees of freedom describing the shape by changing the number of cage handles), and differentiable.

To exploit the differentiability of the proposed deformation-based parameterization, we develop a differentiable articulated rigid body simulator for *contact-rich tasks*, that makes three key improvements over prior work. First, we generalize the frictional contact model proposed by Geilinger et al. [16] that only supports contact between a single dynamic body (e.g., the manipulator) and a static object (e.g., wall) to support contact between multiple dynamic bodies, a key requirement for object manipulation. Second, we propose a modified contact damping force that is free of discontinuities. Third, we derive the analytical gradients for a full spectrum of simulation parameters shown in Table II, including the positions of contact points.

The combination of the proposed deformation-based parameterization and the differentiable simulator allows us to build an end-to-end differentiable framework (Figure 2) for co-optimizing robot morphology and control for contact-rich manipulation tasks using analytical gradients. We test our framework on multiple manipulation problems, some of which are shown in Figure 1. The experiments show that our

deformation-based parameterization provides us an expressive design space and the optimized designs can be easily manufactured (Figure 1 right). Furthermore, due to the key feature of end-to-end differentiability, the proposed method outperforms several state-of-the-art gradient-free approaches and model-free reinforcement learning methods at jointly optimizing the *control scheme* and the *robot morphology*.

II. RELATED WORK

A. Differentiable Physics-Based Simulation

Physics-based simulation has been widely used for various robotic applications [7, 10, 49]. Among them, differentiable physics-based simulators have gained increasing popularity recently since their differentiability facilitates efficient gradient-based optimization for robotic control. Due to their inherent differentiability, neural network based simulators have also been proposed to approximate physics [6, 8, 32, 37]. However, these works sacrifice generality and accuracy of physics for differentiability of the neural network. Differentiable simulators have been developed for rigid bodies [3, 11, 12, 16, 25, 52], soft bodies [15, 22, 26, 27, 28, 30], and cloth [33, 42]. Our simulator lies in the rigid body category, with some key modifications and improvements that enable contact-rich tasks through differentiability of frictional contact between multiple dynamic bodies. Several approaches have been proposed for making the frictional contact response differentiable such as differentiation of the coefficient matrices and vectors of the linear complementarity problem arising from collisions [3, 11] or those that use impulse-based velocity stepping methods [12]. However, the discontinuity stemming from such approaches can cause difficulties for contact-rich tasks such as ours. Recently, Geilinger et al. [16] proposed a differentiable penalty-based frictional contact model. However, their contact model was only demonstrated to work for contact between the robot and a stationary surface (*e.g.*, ground and walls), and the simulation uses stiff springs to approximate articulation. We extend the frictional contact model to support a more general inter-object contact and develop a differentiable simulator based on the reduced coordinate formulation of RedMax [52], which allows a more compact representation and a more accurate articulated dynamics. More importantly, all the previous differentiable rigid body simulators only have gradient information for control variables and a small set of material parameters. In contrast, we derive the analytical gradients for a full spectrum of parameters, including the kinematic and dynamic parameters listed in Table II.

B. Computational Robot Co-Design and Morphology Parameterization

Co-design of robots typically involves optimizing geometry, mass properties, and control parameters. For articulated robot co-design, one must consider extra variables for kinematics and dynamics relationships among links, such as joint and body translations/orientations. Most existing works on rigid robot co-design use simple primitive shapes to approximate

Parameterization Method	Mesh Inference	Complex Shape	Differentiability	Dimension	Feature-preserving	Articulated Design
Primitive Shapes [19, 24, 41, 48, 50]	Fast	No	Support	Low	No	Support
CAD Parameterization [21, 47]	Slow	Support	No	Controllable	Yes	No
TSDF [18]	Slow	Support	No	High	No	No
Mesh-based	Fast	Support	Support	High	No	No
Deformation-based (ours)	Fast	Support	Support	Controllable	Yes	Support

the geometry of each robot link with gradient-based optimizations [19, 24], gradient-free approaches [14, 36, 41, 50], or reinforcement learning (RL) based methods [9, 34, 46]. However, such over-simplified morphology parameterization based on primitive shapes precludes the possibility of generating complex geometric morphology for the robot. It is inadequate especially when the geometry does affect the task dynamics with a rich amount of contact, such as in in-hand manipulation tasks. In contrast, our deformation-based parameterization works for a wide range of complex shapes while preserving the geometric features, narrowing down the robot morphology representation gap between the simulation environment and real fabricated scenarios. CAD-based parameterization can also support natural organic shapes, but it typically suffers from its slow inference speed and non-differentiability. Schulz et al. [47] proposed an interactive system for CAD models with an expensive precomputation cost. Hafner et al. [21] developed a differentiable parameterization of CAD models for FEM simulation, but the differentiability is not preserved in our rigid, multi-body simulation setting. Furthermore, CAD-based parameterization requires extra expertise to select and constrain each parameter in order to preserve model manufacturability throughout the optimization. It is also non-trivial to support the connectivity constraints required in an articulated robot structure. Truncated Signed Distance Functions (TSDF) and mesh-based parameterization are used by Ha et al. [18] to optimize the shape of a free-form gripper. These methods only work for a single body system, introduce a large number of parameters, and usually do not result in natural organic shapes. Compared to these, our deformation-based parameterization allows us to have a constrained but expressive design space for natural shapes and seamlessly works for articulated robot designs. We summarize the comparison among different morphology parameterization methods in Table I.

III. METHOD

We now describe our end-to-end differentiable framework for contact-aware robot design. In Section III-A, we describe our novel deformation-based design space for articulated robot morphology. A key insight of our morphology design space is the use of cage-based deformation, which allows us to morph the underlying mesh using a small number of cage "handles". More specifically, as shown in the top-left block of Figure 2, we use a two-level hierarchy to parameterize the shape. The high-level morphology parameters (shown in red, green, blue, and yellow arrows) controls the positions of the cage vertices (handles), and the cage handles in turn deform the underlying mesh. In Section III-B, we describe

our differentiable articulated rigid body dynamics simulation. As shown in the right block of Figure 2, the simulation takes the deformed meshes and a control sequence as input, executes the forward steps, and computes the objective loss $\mathcal{L}$. By combining the two key techniques above, in Section III-C, we describe our end-to-end framework for robot design. Since each step is differentiable, the overall framework is differentiable, allowing us to use a gradient-based optimization method to search for the design parameters and the controls.

A. Hierarchical Morphology Parameterization

Morphology optimization relies on an effective morphology design space, which further depends on an effective morphology shape parameterization. In this section, we describe our approach to leverage the cage-based deformation as the morphology parameterization for articulated robots with *complex* component shapes. As shown in the top-left block in Figure 2, we use a two-level hierarchy to parameterize the shape of the robot: the cage-based deformation $\mathcal{M}$ and high-level morphology parameterization $\mathcal{H}$.

Cage-based Deformation ($\mathcal{M}$) Cage-based deformation (CBD) is a classic geometry processing technique in computer graphics used to deform a high-resolution mesh in a real-time and feature-preserving manner. With a coarse, closed cage, CBD controls the enclosed space's deformation by moving the cage vertices, or *cage handles* around. Let $\mathcal{C}$ denote the cage and H denote the cage vertices (*i.e.*, handles) of $\mathcal{C}$ with $\tilde{\psi}_h$ being the positions of the handles in H in the rest configuration, and let $\mathcal{S}$ be the space enclosed by cage $\mathcal{C}$. For any arbitrary point $\tilde{s} \in \mathcal{S}$, CBD computes a normalized barycentric coordinate $\boldsymbol{w} \in \mathbb{R}^{|H|}$ for the point, called *deformation weights*, which satisfies:

$$\tilde{s} = \sum_{j}^{|H|} \boldsymbol{w}_j \tilde{\psi}_h(j) \quad \text{and} \quad \sum_{j}^{|H|} \boldsymbol{w}_j = 1. \tag{1}$$

These deformation weights then define a linear function to transfer the translation of handle vertices to the movement of the associated point at run time through:

$$s = \sum_{j}^{|H|} \boldsymbol{w}_j \psi_h(j), \tag{2}$$

where ψ_h is the new positions of the cage handles, and s is the new position of $\tilde{s}$ under deformation. The deformation weights are precomputed for each particular point in the space $\mathcal{S}$ and kept constant at run time. CBD methods preserve various features of the underlying, high-resolution mesh after deformation by carefully designing the weight construction
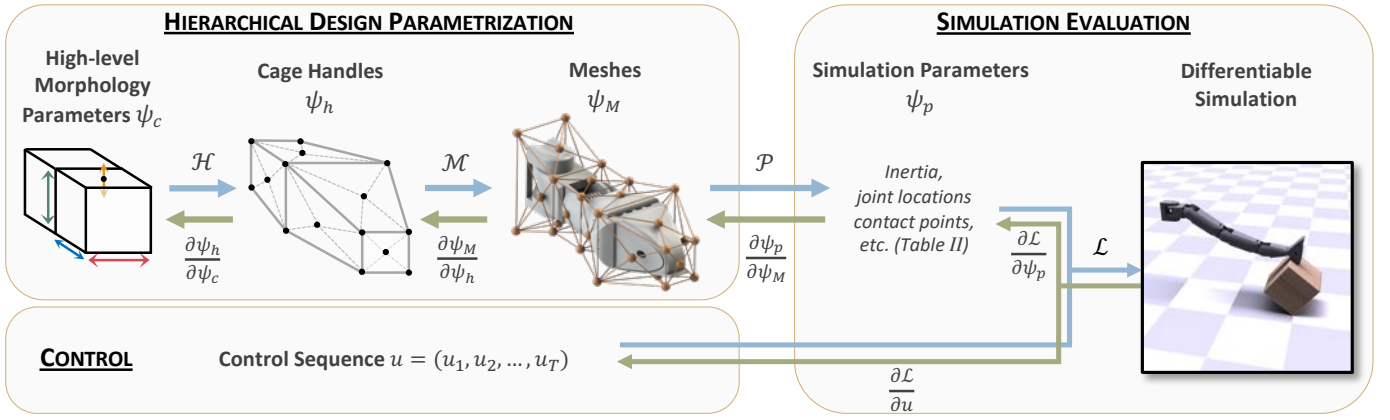
Fig. 2. **End-to-end differentiable framework for morphology and control co-optimization.** Blue arrows labeled as $\mathcal{H}, \mathcal{M}, \mathcal{P}$, and $\mathcal{L}$ are hierarchical functions that evaluate the loss function given the high-level morphology parameters, ψ_c and controls, u. The corresponding green arrows are the derivatives.
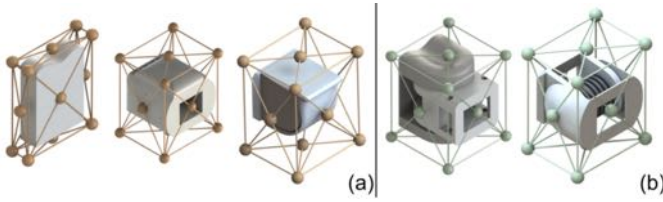

Fig. 3. Our component database for our manipulators. From left to right: finger base, phalanx segment, finger tip, knuckle, and joint. Each component comes with its own deformation cage. (a) The components in the yellow cages can be deformed arbitrarily with the cage, whereas (b) the components in the green cages can only be expanded along the axis of rotation.

algorithms. Among various CBD methods, we choose the mean value coordinates method [31] for its simplicity, stability, and capability to be extended to deform articulated structures.

Inspired by its power, we apply cage-based deformation to parameterize the shape of each robot component by the positions of a set of cage handles around the shape mesh. In practice, the cage handles of each component can be defined by the users based on their demands. For our purpose in this work, we construct a component database for manipulator construction as shown in Figure 3. Each component is represented by a mesh M^i and is associated with a predefined cage $\mathcal{C}^i$ around it. We then use the mean value coordinates method to precompute the deformation weight matrix D^i for each component mesh M^i, and reuse those weights afterwards. Let V^i be the set of vertices of mesh M^i and H^i be the set of handles in cage $\mathcal{C}^i$. Then D^i is a $|V^i|$-by-$|H^i|$ matrix storing the deformation weight for each vertex on the mesh with respect to each handle on the cage. The deformation weights precomputation by mean value coordinates method is cheap, with a computational cost $\mathcal{O}(|V^i| \cdot |H^i|)$. Such precomputation and the run-time linear combination enable a fast mesh inference given the new cage handle positions. Moreover, by controlling the high-resolution mesh's morphology through a coarse cage, we effectively reduce the morphology space into a relatively low-dimensional space for natural deformed shapes. This provides us a constrained yet expressive morphology design space. Furthermore, the dimension of the deformation

is still fully controllable. By adding more handle points, one can deform the underlining mesh with more degrees of freedom, which makes it possible to find a good trade-off between low-dimensional morphology parameter space and fine-grained morphology deformation for different applications. Most importantly, this linear combination gives us a fully differentiable function $\mathcal{M}$ mapping from handle positions ψ_H to the positions of mesh vertices ψ_M.

High-level Morphology Parameterization ($\mathcal{H}$) The CBD method described above works for any *single* mesh with an arbitrary shape. However, parameterizing an articulated robot poses extra challenges, since independently manipulating the cage handles arbitrarily for different robot components will easily lead to a design that is not connected and not manufacturable, as shown in Figure 4(c). In order to handle proper articulation, we need to take two extra constraints into consideration: the *fabrication constraint* and the *connectivity constraint*.

The *fabrication constraint* requires us to have different deformation constraints for different components based on their manufacturing methods. For instance, the finger body part can be manufactured using a 3D printer, so it can undergo free-form deformation. In contrast, the joint component is usually composed of some commercially-sourced products (*e.g.*, screws, spring pins, etc.), which come in predefined, standard sizes. Thus, components such as joints can only be expanded in directions that maintain the geometric integrity of the model where it interfaces with these standard-sized parts. We achieve this through further parameterizing the cage with a few extra parameters to control the allowed deformation for each component.

Connectivity constraint is a more challenging problem for articulated designs, requiring components to remain connected after deforming each underlying mesh. It is non-trivial for CAD-based and mesh-based parameterizations to satisfy this constraint since they are unable to track the connectedness of the interface surfaces between components while varying the parameters. We instead leverage a special property of mean value coordinates to resolve the connectivity issue. Let
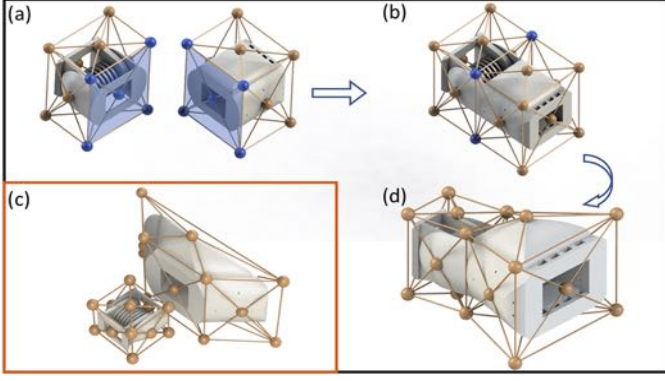
Fig. 4. A *joint* component and a *body phalanx segment* component are shown in the figures. We parameterize the articulated components into lower-dimensional parameters ψ_c by posing different deformation constraints on each component and merging their handle points on the connection surface (highlighted in blue in (a) and (b)). We can then freely explore the ψ_c space to change the underlining articulated robot shape (d). The two components come apart from each other and become to be not manufacturable if they are deformed individually and arbitrarily by their associated cages (c).

$\langle H_1, H_2, H_3 \rangle$ be three handles of a triangle on the cage mesh $\mathcal{C}$, and s be a point in the enclosed space. With mean value coordinates, if s and $\langle H_1, H_2, H_3 \rangle$ are coplanar, and s is inside the triangle $\langle H_1, H_2, H_3 \rangle$, then the deformation weights for s have the following property:

$$\boldsymbol{\omega}_i^s = \begin{cases} 0, & \text{if } i \notin \{H_1, H_2, H_3\} \\ > 0, & \text{if } i \in \{H_1, H_2, H_3\}. \end{cases} \tag{3}$$

In other words, s is fully controlled by the triangle that contains it.

With this property in mind, we construct the cage for each component so that, for each component's connection surface (*i.e.*, interface to a neighboring component), we always construct a cage that is coplanar with and fully contains the connection surface as shown by the blue faces in Figure 4(a). We call the handles on the connection surface plane *connection handles*. For two components that can be potentially connected, their connection handles are constructed to be identical, which ensures that the connection surfaces on two components share the same connection handle layout. While connecting two components, we merge their overlapped connection handles (Figure 4(b)). Such merge operation is equivalent to adding a constraint on the handles on the connection surface so that their connection handles always have the same motion. Due to the property of mean value coordinates in Eq. 3, this placement of connection handles ensures that the connectivity constraint is automatically satisfied.

By considering both the *fabrication constraint* and the *connectivity constraint*, we parameterize the whole cage via a small number of high-level morphology cage parameters ψ_c. This extra layer of parameters implicitly imposes constraints on all cage handle positions to move around in a unified fashion. Specifically, ψ_c consists of: (a) the scale information of the cages (*e.g.*, length of each cage, and width/height of each connection cage, shown as red, blue, and green arrows in Figure 2); and (b) any other auxiliary

cage points (*e.g.*, yellow vertical arrow in Figure 2). The scale parameters are component-dependent and are based on their fabrication constraint. For example, the joint cage can only be scaled along the joint axis direction. Through such cage parameterization, we can map the high-level cage parameters ψ_c to the cage handle positions ψ_h by $\psi_h = \mathcal{H}(\psi_h)$, and effectively construct a high-level morphology design space with only bound constraints. Note that this high-level cage parameterization is not a fixed choice; the parameterization can be modified based on the application and user's demand. We show a more free-form cage parameterzation in the experiment section to show such flexibility.

Although different combinations of the components results in a large amount of different design topologies (*e.g.*, different number of fingers, different number of links, etc.) and thus different final deformation cages (*i.e.*, merged cages), we construct a simple stochastic graph grammar, following the work by Zhao et al. [54], to generate different topologies and automatically construct the merged deformation cage and the high-level cage parameterization. Since we focus on the continuous morphology optimization for a fixed design topology in this work, we leave the details of the grammar in Appendix I.

B. Differentiable Articulated Rigid Body Simulation

From the deformation-based parameterization, we obtain the mesh vertices ψ_M. In order to simulate the constructed morphology, we further convert ψ_M into the simulation parameters ψ_p through an analytical function $\mathcal{P}$. As shown in Table II, the simulation parameters ψ_p include both kinematics- and dynamics-related parameters. Specifically, the kinematic parameters are the relative transformations of the joints with respect to their parent joints, E_j, and the relative transformations of the bodies with respect to their parent joint, E_b; and the dynamic parameters are the generalized inertia, I, contact point positions with respect to the bodies, C_b, and surface area for each contact point, a. For the generalized inertia, we use cuboids for ease of differentiability, but it is also possible to use mesh-based inertia. (Note, however, that the 3D printed parts may not necessarily match the mesh-based inertia, depending on the in-fill.) In order to acquire the contact points C_b in the deformed mesh, we presample a uniformly distributed set of contact points on the surface of each mesh in the rest configuration. We then track the positions of these presampled contact points through the same cage-based deformation as the mesh vertices. Thus, the deformation-based parameterization provides us with differentiability not only for the mesh but also for the contact point positions. The approximate contact point area, a, is used to scale the magnitude of the frictional contact forces. To compute this parameter, we use the change in the total surface area of the cage before and after deformation.

Unlike the previous methods which typically compute the gradients of the simulation only with respect to control parameters $\partial \mathcal{L}/\partial u$ or a small set of material parameters, our simulator also provides the analytical gradients $\partial \mathcal{L}/\partial \psi_p$ for

Type	Notation	Parameter Description	Dimension
Kinematics	E_j	Joint transformation	$SE(3) \times n_b$
	E_b	Body transformation	$SE(3) \times n_b$
Dynamics	I	Generalized inertia	n_{DOF}
	C_b	Contact points on body	$3 \times n_c$
	a	Contact area	n_c

a full spectrum of simulation parameters described above. Such extension is non-trivial and is essential for allowing gradient-based morphology optimization. We provide detailed mathematical formulation of our differentiable simulation in the supplementary document associated with the released code, and briefly introduce the key ideas of our simulation below.

Simulation Dynamics The simulation parameters ψ_p and the control sequence u, are the input to the differentiable articulated rigid body simulator. Our simulator uses reduced coordinates so that the equations of motion are expressed compactly using a minimal set of degrees of freedom [52]. The dynamics equations are implicitly integrated in time with the BDF2 scheme, with SDIRK2 for the initial step [39]. We analytically derive all the derivatives required by these implicit time integration schemes, and we solve the resulting non-linear equations using Newton's Method with line search. To compute the simulation derivatives, $\partial \mathcal{L}/\partial \psi_p$ and $\partial \mathcal{L}/\partial u$, we use the adjoint sensitivity method [16, 35], which requires a forward pass and a backward pass of the simulation. During the forward pass, we store some auxiliary variables, such as the matrix factors of the final Newton iteration of each time step and the partial derivatives of the loss function. Then during the backward pass, we compute the final derivative with a block banded triangular solver using these auxiliary variables stored during the forward pass.

Inter-Object Frictional Contact Model For frictional contact, we extend the differentiable penalty-based approach of Geilinger et al. [16], adapted to our reduced coordinate approach. We make two critical changes to make their penalty-based approach work in our manipulation settings.

First, we add support for frictional contact between *two dynamic* bodies, rather than between a single dynamic body and a stationary surface (*e.g.*, floor and walls). With our approach, as we change the cage parameters, the contact point positions change, and so to allow end-to-end differentiability, we require the gradients of the normal contact force, the dynamic friction force, and the static friction force with respect to these contact point positions as well as with respect to the generalized coordinates of the robot (*i.e.*, joint angles). To solve this, we use a signed distance field augmented with derivative information. This distance field is attached to the manipulated object, which we assume is rigid. Let the generalized coordinates of the robot and the manipulated object be q_b and q_o, respectively, and the generalized velocity be $\dot{q}_b$ and $\dot{q}_o$, respectively. Then the world position and velocity of the contact point C_b on the robot body are computed

as $\mathbf{x}(q_b, C_b)$ and $\dot{\mathbf{x}}(q_b, \dot{q}_b, C_b)$. Using this world point, we query the signed distance function attached to the manipulated object, which gives us the following quantities:

$d, \dot{d}$ penetration distance and speed
$\mathbf{n}$ contact normal
$\dot{\mathbf{t}}$ tangential velocity

as well as the *derivatives* of these quantities with respect to q_b, q_o, and C_b. The frictional contact forces are computed from these quantities and then scaled by the approximate contact area, a. The derivatives of the frictional contact forces with respect to q_b and q_o are used to step the simulation forward in time with the implicit integrator, and the derivatives with respect to C_b and a are used to calculate the sensitivities during the backward pass of the simulation.

Second, we modify the contact damping force so that it is continuous. The original formulation by Geilinger et al. [16] has a discontinuity, which we found can cause convergence problems when there are collisions between two moving dynamic objects. In their formulation, the contact damping force is proportional to the penetration speed, $\dot{d}\mathbf{n}$, which means that there will be a sudden change in the magnitude of this damping force at $d = 0$, since $\dot{d}$ is not necessarily zero when d is zero. To fix this, we instead make the damping force be proportional to both the penetration distance and speed, $d\dot{d}\mathbf{n}$, which ensures that the force remains continuous at the moment of collision. If needed, we can also use a sigmoid function around d to make the damping force match the original formulation when d becomes large.

C. End-to-End Differentiable Co-Design Framework

By combining the proposed deformation-based parameterization and the differentiable simulator, we build an end-to-end differentiable framework for robot co-design as shown in Figure 2.

Mathematically speaking, our co-design framework starts with a three-layer morphology parameterization $\mathcal{F} = \mathcal{P} \circ \mathcal{M} \circ \mathcal{H} : \mathbb{R}^m \to \mathbb{R}^{|H| \times 3} \to \mathbb{R}^{|V| \times 3} \to \mathbb{R}^p$, where m is the number of high-level morphology parameters, $|H|$ is the total number of cage handles, $|V|$ is the total number of mesh vertices, and p is the number of low-level kinematic and dynamic parameters in the simulation. This hierachical parameterization converts high-level morphology parameters ψ_c into low-level simulation parameters ψ_p, going through three analytically differentiable steps including the morphology parameterizations $\mathcal{H}$ and $\mathcal{M}$ in Section III-A, and the simulation parameter computation $\mathcal{P}$ in Section III-B:

$$\psi_p = \mathcal{P}(\psi_M), \ \psi_M = \mathcal{M}(\psi_h), \ \psi_h = \mathcal{H}(\psi_c). \quad (4)$$

The differentiability of each step allows us to efficiently compute the derivatives from the simulation parameters ψ_p all the way to the high-level morphology parameters ψ_c through the chain rule:

$$\frac{\partial \psi_p}{\partial \psi_c} = \frac{\partial \psi_p}{\partial \psi_M} \frac{\partial \psi_M}{\partial \psi_h} \frac{\partial \psi_h}{\partial \psi_c}. \quad (5)$$

Fig. 5. **Morphology design space**. The initial morphology of the single finger and the two-finger gripper designs are shown on the left. We randomly sample different parameters for each configuration and show the deformed morphology on the right.

The framework then proceeds with the simulation described in Section III-B with the simulation parameters ψ_p and a control sequence u as input, and computes the task-specific objective loss $\mathcal{L}$. As our simulator is differentiable with respect to both the kinematic/dynamic parameters and control variables, we are able to compute the analytical derivatives $\frac{\partial \mathcal{L}}{\partial \psi_c} = \frac{\partial \mathcal{L}}{\partial \psi_p} \frac{\partial \psi_p}{\partial \psi_c}$ and $\frac{\partial \mathcal{L}}{\partial u}$ for the full framework efficiently.

IV. EXPERIMENTS

A. Implementation

We implemented our differentiable rigid body simulation in C++ and the deformation-based design parameterization in Python. The two components of the code are connected through Python bindings. The control sequence input to the simulation consists of the torques applied to the joints at each simulation time step. In all the tasks shown in below, we use L-BFGS-B [40] for co-optimization with our analytical derivatives, $\partial \mathcal{L}/\partial \psi_c$ and $\partial \mathcal{L}/\partial u$.

B. Quality of Morphology Design Space

To demonstrate the quality of the morphology design space represented by our deformation-based parameterization, we randomly sample the high-level morphology parameters ψ_c for two manipulator topologies: *single-finger* and *two-finger gripper* configurations (Figure 5). Our deformation-based parameterization method has a compact, yet expressive and rich design space. As shown in Figure 5, using only 9 and 17 design parameters respectively for the two configurations, our method is able to generate various natural and fabricable morphology designs of articulated robots while maintaining the connectivity of the articulated manipulators.

TABLE III
LIST OF HYPER-PARAMETERS FOR EACH EXAMPLE.

Task	Δt_s	n_t	n_{ctrl}	$|u|$	$|\psi_c|$	$|\psi_p|$
Finger Reach	0.005	600	20	120	9	376
Flip Box	0.005	150	5	180	9	1478
Rotate Rubik's Cube	0.005	200	5	240	9	1478
Assemble	0.001	500	5	800	17	1226
Free-form Gripper	0.005	400	1	-	396	9228

C. Morphology and Control Co-Optimization

Tasks In order to test the performance of our differentiable contact-aware co-optimization framework, we designed four manipulation tasks as shown in Figure 7, consisting of three single-finger tasks and one two-finger task:

1) *Finger Reach*: In this task, the base of the finger is mounted on the wall, and the finger is required to reach four scattered target points in the space sequentially.
2) *Flip Box*: This task requires the finger to flip a heavy box by $90°$ and be as energy-efficient as possible.
3) *Rotate Rubik's Cube*: A finger is required to rotate the top layer of a Rubik's cube by $90°$. The bottom of the Rubik's cube is fixed on the ground.
4) *Assemble*: Two fingers need to collaborate to push and insert a small cube into its movable mount. The cube and the hole on the mount have similar sizes, making the task much more challenging and requiring high-accuracy manipulation.

The detailed description and loss function $\mathcal{L}$ of each task is provided in the Appendix II-A. The hyper-parameters of each task are listed in Table III, where Δt_s is the simulation time steps size, n_t is the total number of simulation steps of the task, $|u|$ and $|\psi_c|$ are the total numbers of control and morphology variables in optimization respectively, and $|\psi_p|$ is the number of simulation parameters. We optimize for the control signals not every simulation step but every n_{ctrl} steps, giving us $|u| = (n_t/n_{ctrl}) \cdot |u_i|$ where $|u_i|$ is the number of control degrees of freedom of the robot.

Baselines We adopted the following three baseline algorithms for comparison.

1) *ES*: Evolutionary strategy is widely used to search for optimal design and control parameters for robots [14, 36]. We tried various ES algorithms in the open-sourced Nevergrad library [43] and found that the $(1 + 1)$-*ES* [5, 44] algorithm and *CMA-ES* [23] work best on the proposed tasks.
2) *RL*: Luck et al. [34] is one of the state-of-the-art morphology and control co-optimization approaches using sample-efficient reinforcement learning (soft actor critic, SAC) algorithm and particle swarm optimization. We used their released implementation as a baseline.
3) *Control Only*: In this algorithm, we freeze the morphology parameters and only optimize the control sequence with L-BFGS-B.

Experiment Setup We use the same morphology parameterization for baselines and our method. The control parameter for the *RL* baseline is a neural network controller (a policy network) as proposed by Luck et al. [34], and is an open-loop control sequence for all other methods. We try both *ES* algorithms due to their different performances on different tasks. For fair comparison, we finetune the parameters of *ES* and *RL* baselines and run the experiments with the best-performing parameters. For the *Control Only* baseline and our method, we use the default parameters provided in the Scipy's L-BFGS-B optimizer. While our method can solve the

TABLE IV

Normalized Metric Comparison. We design the task-related metrics to measure how successful each method performs on the tasks. For *Finger Reach* task, the metric is the time-averaged distance to the target tracking points. For *Flip Box* and *Rotate Rubik's Cube*, the metrics are the flipping/rotating angle error at the end of the task. For *Assemble*, we measure the distance between the center of the small box and the center of the hole on the movable mount. All the metrics are normalized.

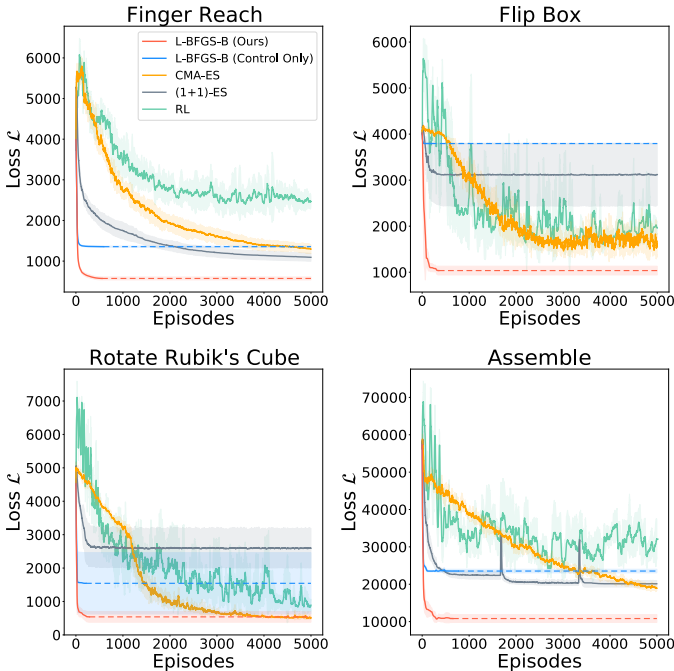| Task	Finger Reach	Flip Box	Rotate Rubik's Cube	Assemble
CMA-ES	0.39 ± 0.02	0.00 ± 0.00	$\mathbf{0.02 \pm 0.01}$	0.28 ± 0.03
(1+1)-ES	0.35 ± 0.04	0.69 ± 0.39	0.87 ± 0.15	0.39 ± 0.09
RL	0.61 ± 0.05	0.41 ± 0.48	0.79 ± 0.31	0.91 ± 0.11
L-BFGS-B (Control Only)	0.41 ± 0.00	1.00 ± 0.00	0.42 ± 0.39	0.77 ± 0.03
L-BFGS-B (Ours)	$\mathbf{0.17 \pm 0.01}$	$\mathbf{0.00 \pm 0.00}$	0.07 ± 0.09	$\mathbf{0.12 \pm 0.11}$

Fig. 6. **Optimization curves comparison**. We run all the methods on all tasks 5 times with different random seeds. Mean and standard deviation in the loss objective are reported. The horizontal axis of each plot is the number of simulation episodes, and vertical axis is the objective loss value. L-BFGS-B optimization can terminate early once it satisfies the termination criterion. For better visualization, we extend the actual learning curves that use L-BFGS-B horizontally using dotted lines. We also smooth out the curves with a window size of 10.

Assemble task with a high success rate, we found that the loss objective can be further decreased by using a continuation method [2, 16]. Specifically, on the *Assemble* task, we scale down the contact forces at the beginning of the optimization to provide a smoother objective function space, and scale it up as the optimization proceeds. We set three stages with contact force scale equal to 0.01, 0.1 and 1, and start the next optimization stage once the previous stage converges. To apply the continuation method on the baseline algorithms, we fix the number of simulations in each stage and proceed to the next stage after the previous stage uses up the budget. We run all the baseline algorithms with and without continuation method on *Assemble* task, and plot the best performing one. We run all the methods on each task for five times with different

random seeds and plot the average training curves in Figure 6. We further measure the successfulness of the tasks for each method by task-related metrics, and report in the Table IV.

Results The results show that our differentiable co-optimization framework is able to find better morphology and control solutions with significant better sample efficiency (10-30 times fewer simulated episode data) compared to the gradient-free *ES* baselines and model-free *RL* baseline. On *Finger Reach* task, while most methods (except *Control Only*) find a finger configuration that can reach the four target points, our method can find a morphology and a control sequence that can track the target points most accurately. On the most challenging and contact-rich task, *Assemble*, our method is the only one that is able to solve the task successfully.

We also performed an ablation study on the importance of morphology design by comparing the performance of our method and a *Control Only* baseline. The significant performance gain of our method over *Control Only* baseline reveals that incorporating the optimization of morphology design leads to easier optimization and better solutions. We show some of the optimized morphology designs from our method in Figure 7. On the *Flip Box* task, the optimized morphology has a hook-like structure at the finger tip, so that it is able to hook on the back surface of the box to flip over the box more easily. For the *Assemble* task, the optimized morphology has fingers of different lengths, so that the long right finger is able to push the smaller cube while the short left finger can hold the mount. Moreover, the design has flat and larger fingers, which allows the manipulator to push the object much more stably than a thin finger. More visual results can be found in our supplementary video.

D. Flexibility of the Morphology Parameterization

By adding more cage handles, we can easily increase the degrees of freedom for the morphology design. We show such flexibility of our deformation-based morphology parameterization in this section using a free-form gripper task motivated by Ha et al. [18]. As shown in Figure 8, the algorithm needs to optimize the shape of a pair of gripper fingers such that the gripper can grasp a diamond-like object using a predefined control sequence. Each finger starts with a cube-like shape and is optimized with free-form deformation.

To support deformation in higher degrees of freedom (DoF), we add more handles on the cage around each gripper finger.
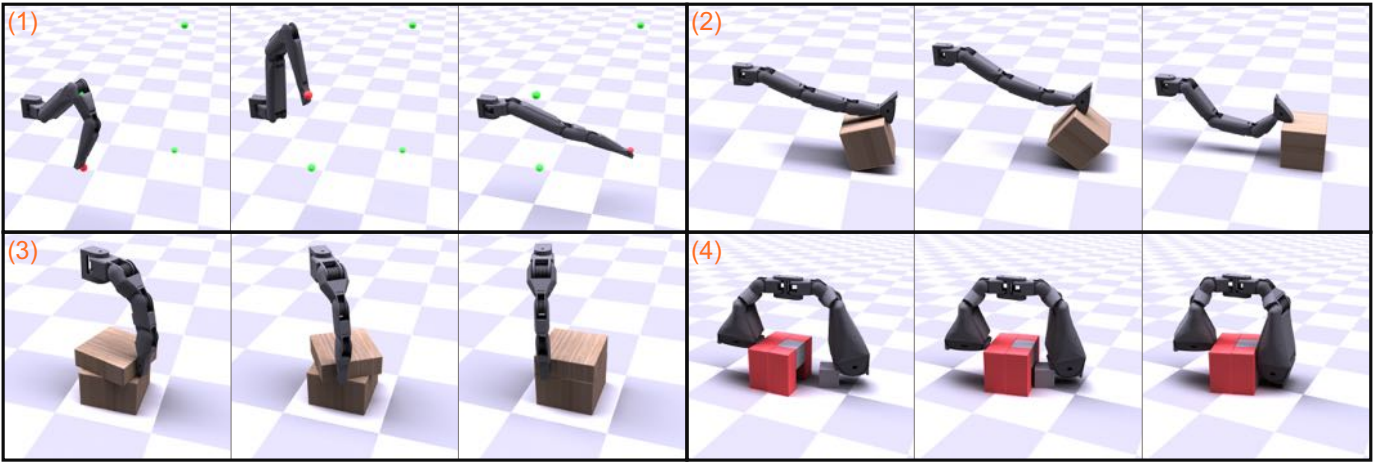
Fig. 7. **Optimized designs and controls for four manipulator tasks**. (1) *Finger Reach*. (2) *Flip Box*. (3) *Rotate Rubik's Cube*. (4) *Assemble*. More visual results are provided in the supplementary video.
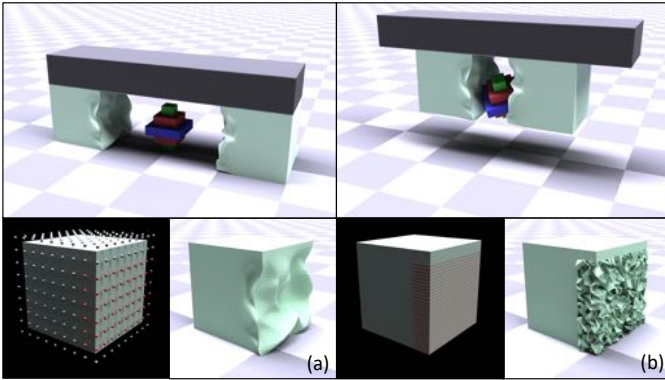

Fig. 8. **Free-form Gripper**: The task is to pick up the object, as shown in the top row. We compare deformation-based parameterization (ours) and mesh-based parameterization. The optimization variables and optimized gripper morphology for the left gripper finger using both methods are shown in the bottom row. (a) *our parameterization method*: all the cage handles are shown on the left sub-figure and the ones used as optimization variables are highlighted in red. (b) *mesh-based parameterization*: we allow the optimization to directly optimize all the mesh vertices highlighted in red in the left sub-figure. In both cases, we do not modify the areas near the top of the gripper. The gripper morphology generated by our method is much smoother.

To show the advantage of using a deformation cage based parameterization, we compare it to a differentiable mesh-based parameterization (similar to the Truncated Signed Distance Function (TSDF) parameterization used in Ha et al. [18]) which directly optimize over the vertex positions of the mesh. For both parameterization methods, we only optimize the handles/mesh vertices on the inner side for each gripper finger as shown in Figure 8 (bottom row). We test both parameterization methods in our differentiable framework, and conduct 30 independent experiments for each method with different initial parameters.

Even though our deformation cage based parameterization has a much smaller optimization space and DoFs (396 optimization variables in the grasping task) than the mesh-based parameterization (8946 variables), our method is able to generate comparable success rates on the grasping task than

the mesh-based parameterization (ours: 97%, mesh: 100%), and achieves better average loss (plot shown in Appendix II-B). Moreover, as shown in the Figure 8, our method is able to generate much smoother morphologies than the mesh parameterization which creates many reverted triangles on the mesh. Such advantage comes from the smoothness and feature-preserving properties of the cage-based deformation method.

E. Manufacturing of Optimized Designs

We manufactured two optimized finger models from our method, one generated by the *Flip Box* task and the other generated by the *Assemble* task, as shown in Figure 1. The finger components were 3D printed on a Markforged printer using Onyx, a micro carbon fiber filled nylon, and assembled together after print. Minimal modification was required to prepare the program-generated models for printing, demonstrating a streamlined design process. This shows that our deformation-based morphology paramterization successfully maintains design manufacturability.

We further tested the functionality of the manufactured finger for the *Flip Box* task. Vectran cables were routed through the 3D printed finger and controlled by dynamixel DC servos at the base of the finger. The tendon-driven finger system was then mounted on a UR5 arm. We manually programmed a control sequence on the dynamixel motors and the UR5 arm to follow a series of waypoints from the trajectory optimized by the algorithm. The experiment shows that the manufactured finger can effectively flip the cube in real world. We also test the robustness of the optimized design on the cubes of various sizes and the experiment demonstrates that the finger can also perform the flipping tasks successfully. Please see the supplementary video for this real-world experiment.

V. DISCUSSION

In this work, we present an end-to-end differentiable framework for contact-aware robot designs. At the core of our contribution is a novel deformation-based morphology parameterization for articulated robot designs, and a differentiable rigid body simulation carefully developed for contact-

rich manipulation tasks. The experiments show that our innovative morphology parameterization approach provides us with an effective and expressive morphology design space. We also demonstrate that for a given manipulation task, by applying gradient-based optimization algorithm in our fully differentiable framework, our method is able to find a better morphology and control combination with significantly fewer number of simulation episodes than the state-of-the-art approaches. Furthermore, the optimized designs can be easily manufactured and are functional in real world. We release our implementation for reproducibility.

Note that although in this work we focus on optimizing the continuous component shape given a fixed design topology, the database we constructed can actually produce a large amount of different design topologies (*e.g.*, different number of fingers, different number of links, etc.) via different combinations of the components through our constructed grammar in Appendix I. Furthermore, our deformation-based parameterization allows us to reuse the cages and the precomputed deformation weights for each individual component across different manipulator structures. Such automatic and repeatable process enables the potential research on manipulator structure optimization in the future. It is also worth mentioning that while our examples only show the components with identical connection surfaces, it is not a limitation of our proposed deformation-based parameterization. One can easily applies our method for components with connection surfaces of different sizes by constructing proper cages for them.

There are several directions which can be explored in the future. First, though analytical derivatives allows for an efficient gradient-based optimization algorithm, the local minimum issue of the gradient-based optimization is a known problem. Combining the exploration ability from model-free (gradient-free) methods and the gradient-based optimization can be an interesting direction to better leverage the advantages from both worlds. Second, our proposed method is applicable to optimize the continuous morphology parameters given the fixed robot topology. There is another category of morphology optimization that focuses on searching the discrete topology of the robot designs [20, 51, 53, 54]. It is desirable to develop an advanced optimization algorithm to effectively search over both the discrete robot topology space and the continuous morphology space. Finally, narrowing down the gap between the reality and our simulation-based framework can be another important task in the future in order to seamlessly transfer the optimized morphology and control from simulation directly in to real world.

Acknowledgements

References

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. arXiv preprint arXiv:1910.07113, 2019.

[2] Eugene L Allgower and Kurt Georg. Introduction to numerical continuation methods. SIAM, 2003.

[3] Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks, 2019.

[4] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. The International Journal of Robotics Research, 39 (1):3–20, 2020.

[5] Anne Auger. Benchmarking the (1+ 1) evolution strategy with one-fifth success rule on the bbob-2009 function testbed. In Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, pages 2447–2452, 2009.

[6] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pages 4509–4517, 2016.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.

[8] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. arXiv preprint arXiv:1612.00341, 2016.

[9] Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning. arXiv preprint arXiv:2008.04460, 2020.

[10] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[11] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. Advances in neural information processing systems, 31:7178–7189, 2018.

[12] Jonas Degrave, Michiel Hermans, Joni Dambre, et al. A differentiable physics engine for deep learning in robotics. Frontiers in neurorobotics, 13:6, 2019.

[13] Raphael Deimel and Oliver Brock. A novel type of compliant and underactuated robotic hand for dexterous grasping. The International Journal of Robotics Research, 35(1-3):161–185, 2016.

[14] KM Digumarti, C Gehring, S Coros, J Hwangbo, and R Siegwart. Concurrent optimization of mechanical design and locomotion control of a legged robot. In 17th International Conference on Climbing and Walking Robots (CLAWAR), pages 315–+. WORLD SCIENTIFIC PUBL CO PTE LTD, 2014.

[15] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics with contact. arXiv preprint arXiv:2101.05917, 2021.

[16] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: analytically differentiable dynamics for multi-body systems with frictional contact. ACM Transactions on Graphics (TOG), 39(6):1–15, 2020.

[17] Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. arXiv preprint arXiv:2102.02202, 2021.

[18] Huy Ha, Shubham Agrawal, and Shuran Song. Fit2form: 3d generative model for robot gripper form design. arXiv preprint arXiv:2011.06498, 2020.

[19] Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. Joint optimization of robot design and motion parameters using the implicit function theorem. In Robotics: Science and systems, volume 8, 2017.

[20] Sehoon Ha, Stelian Coros, Alexander Alspach, James M Bern, Joohyung Kim, and Katsu Yamane. Computational design of robotic devices from high-level motion specifications. IEEE Transactions on Robotics, 34(5):1240–1251, 2018.

[21] Christian Hafner, Christian Schumacher, Espen Knoop, Thomas Auzinger, Bernd Bickel, and Moritz Bächer. X-cad: Optimizing cad models with extended finite elements. ACM Trans. Graph., 38(6), November 2019. ISSN 0730-0301. doi: 10.1145/3355089.3356576. URL https://doi.org/10.1145/3355089.3356576.

[22] David Hahn, Pol Banzet, James M Bern, and Stelian Coros. Real2sim: Visco-elastic parameter estimation from dynamic motion. ACM Transactions on Graphics (TOG), 38(6):1–13, 2019.

[23] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. Evolutionary computation, 9(2):159–195, 2001.

[24] Christopher Hazard, Nancy Pollard, and Stelian Coros. Automated design of manipulators for in-hand tasks. In 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), pages 1–8. IEEE, 2018.

[25] Eric Heiden, David Millard, Hejia Zhang, and Gaurav S Sukhatme. Interactive differentiable simulation. arXiv preprint arXiv:1905.10706, 2019.

[26] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In 2019 International conference on robotics and automation (ICRA), pages 6265–6271. IEEE, 2019.

[27] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. In International Conference on Learning Representations, 2020.

[28] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B. Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=xCcdBRQEDW.

[29] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. ACM Trans. Graph., 30(4):78, 2011.

[30] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. arXiv preprint arXiv:2104.02646, 2021.

[31] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In ACM SIGGRAPH 2005 Papers, SIGGRAPH '05, page 561–566, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 9781450378253. doi: 10.1145/1186822.1073229. URL https://doi.org/10.1145/1186822.1073229.

[32] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. arXiv preprint arXiv:1810.01566, 2018.

[33] Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/28f0b864598a1291557bed248a998d4e-Paper.pdf.

[34] Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In Conference on Robot Learning, pages 854–869. PMLR, 2020.

[35] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. ACM Trans. Graph., 23(3):449–456, August 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015744. URL https://doi.org/10.1145/1015706.1015744.

[36] Andre Meixner, Christopher Hazard, and Nancy Pollard. Automated design of simple and robust manipulators for dexterous in-hand manipulation tasks using evolutionary strategies. In 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), pages 281–288. IEEE, 2019.

[37] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel LK Yamins. Flexible neural representation for physics prediction. arXiv preprint arXiv:1806.08047, 2018.

[38] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In Conference on Robot Learning, pages 1101–1112. PMLR, 2020.

[39] Hiroaki Nishikawa. On large start-up error of bdf2. 392: 456 – 461, 2019. ISSN 0021-9991.

[40] Jorge Nocedal and Stephen Wright. Numerical optimization. Springer Science & Business Media, 2006.

[41] Xinlei Pan, Animesh Garg, Animashree Anandkumar, and Yuke Zhu. Emergent hand morphology and control from optimizing robust grasps of diverse objects. arXiv preprint arXiv:2012.12209, 2020.

[42] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. Scalable differentiable physics for learning and control. In International Conference on Machine Learning, pages 7847–7856. PMLR, 2020.

[43] J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[44] Ingo Rechenberg. Evolutionsstrategien. In Simulationsmethoden in der Medizin und Biologie, pages 83–114. Springer, 1978.

[45] Daniela Rus and Michael T Tolley. Design, fabrication and control of soft robots. Nature, 521(7553):467–475, 2015.

[46] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In 2019 International Conference on Robotics and Automation (ICRA), pages 9798–9805. IEEE, 2019.

[47] Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. Interactive design space exploration and optimization for cad models. ACM Transactions on Graphics (TOG), 36(4):1–14, 2017.

[48] Andrew Spielberg, Brandon Araki, Cynthia Sung, Russ Tedrake, and Daniela Rus. Functional co-optimization of articulated robots. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 5035–5042. IEEE, 2017.

[49] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5026–5033. IEEE, 2012.

[50] Kevin Wampler and Zoran Popović. Optimal gait and form for animal locomotion. ACM Transactions on Graphics (TOG), 28(3):1–8, 2009.

[51] Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. Neural graph evolution: Towards efficient automatic robot design. arXiv preprint arXiv:1906.05370, 2019.

[52] Ying Wang, Nicholas J. Weidner, Margaret A. Baxter, Yura Hwang, Danny M. Kaufman, and Shinjiro Sueda. REDMAX: Efficient & flexible approach for articulated dynamics. ACM Trans. Graph., 38(4), July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3322952. URL https://doi.org/10.1145/3306346.3322952.

[53] Jie Xu, Andrew Speilberg, Allan Zhao, Daniela Rus, and Wojciech Matusik. Multi-objective graph heuristic search for terrestrial robot design. IEEE, 2021.

[54] Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. ACM Transactions on Graphics (TOG), 39(6):1–16, 2020.