Computational Design of Robotic Devices from High-Level Motion Specifications

Sehoon Ha, *Member, IEEE,* Stelian Coros, *Member, IEEE,* Alexander Alspach, *Member, IEEE,*
James M. Bern, *Student Member, IEEE,* Joohyung Kim, *Member, IEEE,* Katsu Yamane, *Member, IEEE*

***Abstract*—We present a novel computational approach to designing robotic devices from high-level motion specifications. Our computational system uses a library of modular components—actuators, mounting brackets, and connectors—to define the space of possible robot designs. The process of creating a new robot begins with a set of input trajectories that specify how its end effectors and/or body should move. By searching through the combinatorial set of possible arrangements of modular components, our method generates a functional, as-simple-as-possible robotic device that is capable of tracking the input motion trajectories. To significantly improve the efficiency of this discrete optimization process, we propose a novel heuristic that guides the search for appropriate designs. Briefly, our heuristic function estimates how much an intermediate robot design needs to change before it becomes able to execute the target motion trajectories. We demonstrate the effectiveness of our computational design method by automatically creating a variety of robotic manipulators and legged robots. To generate these results we define our own robotic kit that includes off-the-shelf actuators and 3D printable connectors. We validate our results by fabricating two robotic devices designed with our method.**

***Index Terms*—Mechanism Design, Legged Robots, Manipulation Planning, Kinematics.**

I. Introduction

Over the past five decades, robots have fundamentally transformed large-scale industrial manufacturing. Nowadays, as hardware platforms are becoming increasingly versatile and affordable, robots promise to have an equally profound impact on other aspects of our lives. Indeed the way we work, learn and play may forever be changed by service robots that provide help with chores, exploration, search-and-rescue missions and educational activities.

While mass-produced robots have well-established advantages, the ability to custom-create designs on demand presents distinct opportunities. For example, inspection robots can be created specifically for the environment they need to operate in and the tasks they need to complete, while assembly robots for short run manufacturing can be reconfigured to meet the demands imposed by different types of products while remaining as simple and easy to operate as possible.

Partly due to the need to easily configure customized devices, and partly due to the economy of mass production, it is common practice to employ a standard set of modular components (e.g. servomotors, mounting brackets and other structural elements) when creating robotic systems [1], [2], [3]. In this setting, the task of designing a new robot amounts to choosing which components to use, how to combine them to form a functional system that is sufficiently versatile, and how to control the resulting device in order to achieve a desirable set of motions or behaviors. Due to the intimate coupling between these sub-tasks, the design process is notoriously challenging. Most robotic systems available today are therefore the product of meticulous, time-consuming and largely manual efforts led by experienced engineers. As the diversity of robotic devices that enter our lives grows, today's design methodologies are likely to become too limiting or prohibitively expensive.

We propose a novel computational method that efficiently automates the challenging problem of designing robotic devices. Our computational system takes as input a library of modular components, as well as basic rules that define compatible connections between them. Starting from a high-level description of a motion task, which is provided in the form of desired motion trajectories for end effectors or the robot's body, our method creates a functional, as-simple-as-possible design. The output designs are obtained by searching through the space of possible arrangements of modular components. To guide this discrete optimization process, we propose a novel heuristic that estimates the ability of any intermediate design to reproduce the input motion trajectories.

We demonstrate the effectiveness of our computational method by automatically designing an assortment of robotic manipulators and walking robots. To evaluate the scalability of our approach, we define our own robotic kits that are based on Dynamixel actuators [4] and 3D printable connectors. We validate our results by fabricating one of the designs generated by our method.

II. Related Work

Manual Robot Design Designing robots is a difficult problem that requires expert knowledge in a range of different areas, including mechanical and electrical engineering, motion planning, and control. In creating new types of robots,

Manuscript received April 28, 2017; revised August 21, 2017.

Sehoon Ha was with Disney Research, Pittsburgh, PA, 15213 when the research was conducted. He is now with Carnegie Mellon University, Pittsburgh, PA, 15213. sehoon.ha@gmail.com

Stelian Coros and James M. Bern are with ETH Zürich, Switzerland. scoros@inf.ethz.ch, jamesmbern@gmail.com

Alexander Alspach was with Disney Research, Pittsburgh, PA, 15213 when the research was conducted. He is now with Toyota Research Institute, Cambridge, MA, 02139 alexalspach@gmail.com

Joohyung Kim is with Disney Research Los Angeles, Glendale, CA, 91201. joohyung.kim@disneyresearch.com

Katsu Yamane was with Disney Research, Pittsburgh, PA, 15213 when the research was conducted. He is now with Honda Research Institute USA, Mountain View, CA, 94043. kyamane@honda-ri.com

designers oftentimes look to nature for inspiration. Examples of robots mimicking real-life animals include salamanders [5], cheetahs [6], kangaroos [7], chimpanzees [8], and cockroaches [9]. Virtual characters can likewise be used as a source of inspiration for roboticists [10]. In these cases, however, the design process begins with a vision for the overall appearance of the robot. Our computational approach, in contrast, begins with a description of what the robotic device should be able to do. Furthermore, rather than relying on mostly manual efforts, the robots designed with our method are generated automatically such that they are as simple as possible, yet sufficiently versatile to execute the motion-based tasks that are provided as input.

Task-based Robot Design The challenge of creating robotic devices that are custom-made to perform different tasks has received considerable attention in the field of manipulator design. In particular, previous work has addressed the design of general [11], [12], [13] and parallel manipulators [14], [15], [16] that reach desired configurations in a specified workspace while avoiding joint singularities. Other types of robots optimized for specific tasks include pipe-cleaning robots [17], stair-climbing mobile robots [18], and legged robots [19]. In the computer graphics community, Wampler and Popović [20] demonstrated a simultaneous optimization of gaits and body proportions for virtual animals. Our work is inspired by this body of research. However, most prior methods focus on optimizing continuous parameters, such as limb lengths, while keeping the morphology (e.g. number of actuators and their connectivity) of the robot design fixed. In contrast, the method we propose designs robotic devices entirely from scratch, concurrently optimizing their morphological structure, kinematic proportions and motor control signals.

Evolutionary Robot Design Inspired by Sims's pioneering work in character animation [21], many roboticists have explored the use of evolutionary algorithms to aid in the design process for manipulators [22], tensegrity robots [23], [24], and soft robots [25]. These approaches rely on the ability to generate and test a large variety of robot designs in a physically-simulated environment. Each variation is generated by combining or mutating existing designs in a stochastic fashion. Although evolutionary approaches have proven to be simple and effective in designing robotic devices, they often lead to local minima and results can be difficult to reproduce on subsequent runs. The goal of our approach, in contrast, is to provide a systematic exploration of the space of feasible robotic devices and their motor capabilities.

Physical Character Design Recent advances in 3D printing technologies fueled a large body of research on fabrication and design optimization for physical objects. In this research domain, the problems investigated have evolved from static objects to functional devices, including articulated characters [26], [27], self-balancing artifacts [28], [29], foldable furniture [30], gliders [31], mechanical toys [32] and automata [33], [34], [35], some of which are even specifically-designed to walk [36]. Moving towards the design of increasingly more complex robotic systems requires the development of predictive models that capture the interplay between actuation capabilities and morphological characteristics. Megaro and his colleagues [37], for example, presented an interactive design system that generates stable locomotion patterns for legged robots with arbitrary, user-created morphologies, while Du *et al.* [38] proposed a design method to optimize the structure and flight controllers of multicopters based on a notion of desired capabilities. Inspired by this body of work, our goal is to computationally design different types of robotic devices, from the ground up, based on high-level descriptions of the tasks they need to perform.

III. Overview

The goal of our work is to automate the design of robotic devices based on high-level descriptions of a desired motion. These high-level descriptions, which are depicted in Fig. 1, typically consist of center of mass and end effector trajectories, and they specify how a robot should move.

As illustrated in Fig. 1, the computational method we propose operates in terms of libraries of modular *design elements*, or *modules*, such as actuators and connectors. The space of feasible robot designs is implicitly defined through the way in which these modules can be combined to form complex systems. Our computational system efficiently explores this combinatorial design space in search for an optimized, as-simple-as-possible robotic device that can carry out the desired motion.

The optimization problem that our computational system must solve is very difficult: the mechanical design of a robot—the number and type of design elements to be used, as well as their connectivity—needs to be optimized concurrently with the control signals that its actuators must execute to recreate the input motion trajectories. This observation highlights two important challenges that must be addressed. First, we must solve for an unknown number of variables (e.g. number of design elements) that can take on both discrete (e.g. element type) and continuous values (e.g. motor control signals). Second, while searching for a functional robot that is as simple as possible, *incomplete designs* that are unable to achieve the desired motion will also need to be evaluated. In order to effectively guide the search process, our system must assess the possibility that each incomplete design will eventually lead to an optimal robot design.

We address this difficult design optimization problem using a heuristic-guided tree search algorithm. As illustrated in Fig. 1, nodes in the search tree correspond to feasible *intermediate robot designs* that may or may not be able to complete the desired motion. To evaluate the cost of any intermediate design, we begin by augmenting it with an idealized *virtual end effector*. The virtual end effector, although not physically realizable, has sufficient degrees of freedom to solve any task. We employ a specialized numerical solver to generate control inputs for the resulting *hybrid device*, such that it optimally recreates the input motion trajectories. The solver heavily regularizes the contribution of the virtual end effector. As a result, its degrees of freedom are only used to account for deficiencies in the intermediate design that it augments. If the intermediate design can complete the input motion by itself, for example, the virtual end effector will

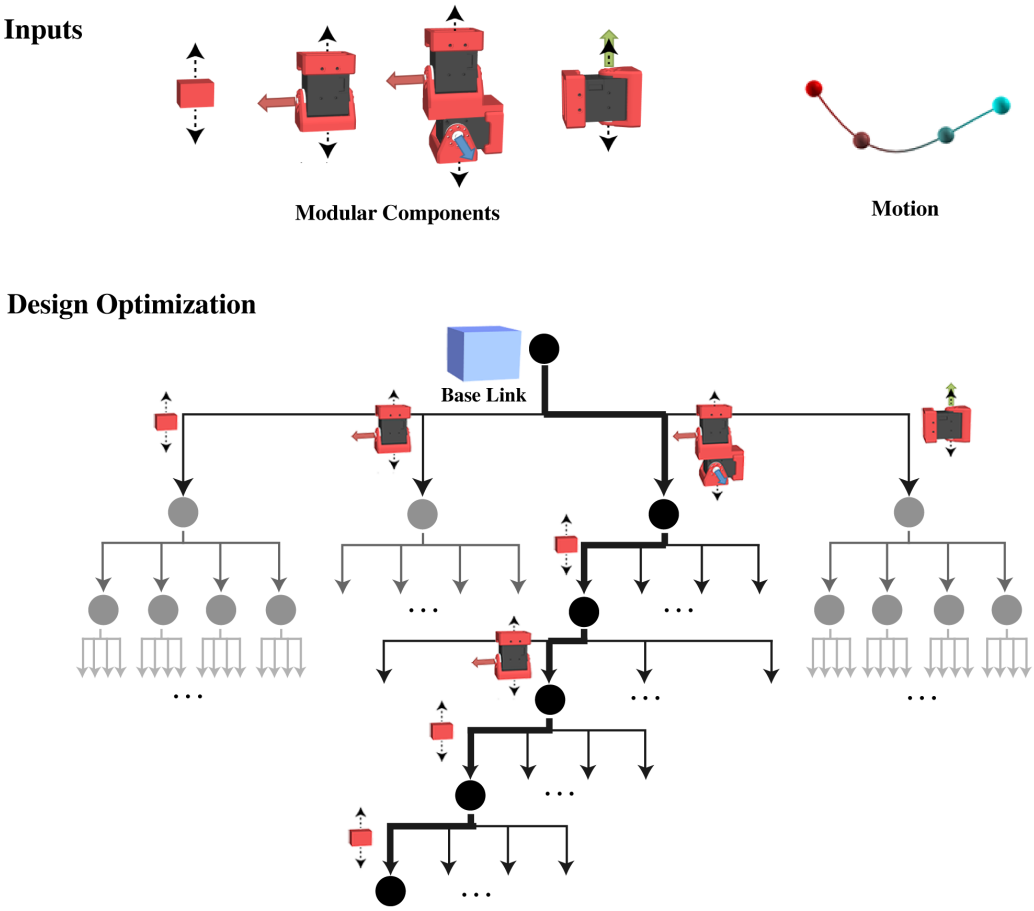


Fig. 1. Overview. **Top:** The input modular components (an available set of servos and connectors) and the desired end-effector trajectory. **Middle:** An example search tree, which is simplified for visualization purpose. **Bottom:** The output design and its motion.

not be used at all. Conversely, the fewer (useful) degrees of freedom an intermediate design has, the more the virtual end effector will contribute to the motion of the hybrid device. The degree to which the virtual end effector has to help out therefore provides a heuristic measure of how well-suited an intermediate design is for a given input motion.

In Section IV, we formally describe the coupled design and control problem that is solved by our computational method. To solve this problem efficiently, we develop an A* search algorithm as described in Section V. Expanding a node during the search process amounts to creating new robots by iterating through all compatible modules and appending them to the current intermediate design. The heuristic function we propose allows our computational system to explore variations of the most promising intermediate designs.

IV. Problem Formulation

The goal of our computational design method is to automatically generate the simplest robot that is capable of executing user-specified motions. Two main ingredients are required to formalize this design synthesis problem. First, we must define the space of feasible robot designs that needs to be explored. Second, we need to establish the mathematical foundation

that allows the design process to be cast as an optimization problem.

A. Design Space

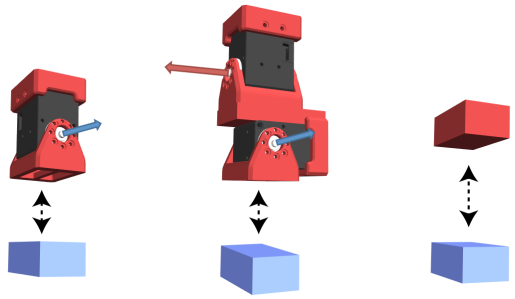


Fig. 2. Connection rules which specify how design elements (e.g. motor assemblies and structural links) can be combined to form complex robotic systems.

Our computational framework takes as input an implicit definition of the space of attainable robot designs through a library of *design elements* and associated *connection rules*. The design elements are modular building blocks, such as motors and structural links. The connection rules define compatibilities between these modular building blocks. Each connection rule is defined as $e = \{x, y, {}_x\mathbf{T}_y\}$, where x and y denote a pair of compatible modules, and ${}_x\mathbf{T}_y$ is the rigid transformation that specifies how module x is positioned relative to module y. Fig. 2 illustrates a few design elements and connection rules between them. The highlighted structural link can be connected to different motor assemblies or to other structural links.

Starting from a base link z, a robotic device $\mathcal{D} = \{\mathcal{V}, \mathcal{E}\}$ is represented by a collection $\mathcal{V}$ of interconnected design elements and their connections $\mathcal{E}$. The connection rules are used to recursively compute the location and orientation of each module $v \in \mathcal{V}$ (robots designed with our method will not feature kinematic loops). The resulting configuration corresponds to the pose of the robot when all joint positions are set to 0. We let $\mathbf{q}$ represent an arbitrary pose of the robot, where $\mathbf{q}^j$ denotes the angle attained by motor $m_j (j \leq M)$ and M is the number of motors in $\mathcal{D}$. For any vector $\mathbf{q}$, the configuration of the robot is computed through forward kinematics.

B. Robot Design as an Optimization Problem

Our computational method aims to find the simplest robot design that can successfully execute user-provided motions. To quantify the complexity of a design $\mathcal{D}$, and implicitly its fabrication cost, we define two terms, g_s and g_a. The former estimates the amount of structural material required to physically realize $\mathcal{D}$; this is achieved by integrating over the volume of its structural elements. The latter term, g_a, is proportional to the number of actuators included in the design. Combining these two terms, the complexity of a design $\mathcal{D}$ is given by:

$$g(\mathcal{D}) = \sum_{v \in \mathcal{V}} g_s(v) + w_a \sum_{v \in \mathcal{V}} g_a(v) \tag{1}$$

TABLE I
MOTION CONSTRAINTS

Description	Expression
End-effector position constraint	${}^a\mathbf{E}^j(\mathcal{D}, \mathbf{q}) = \bar{\mathbf{E}}^j$
End-effector orientation constraint	${}^b\mathbf{R}^j(\mathcal{D}, \mathbf{q}) = \bar{\mathbf{R}}^j$
Base link constraint	${}^c\mathbf{P}(\mathcal{D}, \mathbf{q}) = \bar{\mathbf{P}}$
Self collision constraint	${}^d d_{jk}(\mathcal{D}, \mathbf{q}) \geq 0$
External collision constraint	${}^e e_j(\mathcal{D}, \mathbf{q}, \Phi) \geq 0$

${}^a\mathbf{E}^j$ and $\bar{\mathbf{E}}^j$ evaluate the actual and desired positions of the jth end-effector.
${}^b\mathbf{R}^j$ and $\bar{\mathbf{R}}^j$ evaluate the actual and desired orientations of the jth end-effector.
${}^c\mathbf{P}$ and $\bar{\mathbf{P}}$ evaluate the actual and desired base link positions and orientations.
${}^d d_{jk}$ evaluates the distance between the jth and kth links.
${}^e e_j$ evaluates the distance between the jth link and the environment Φ.

where w_a is the weight term.

The desired functionality of a robot is specified through a set of motion trajectories for its end-effectors or base link. We discretize these motion trajectories in time to obtain a set of N *target frames*, where the time complexity of the entire search process is near linear to N. A robot design $\mathcal{D}$ is able to achieve the user-specified motions if there exists a set of poses $\mathbf{q}_i$ $(i \leq N)$ that satisfy all $M + P$ high-level motion constraints $c_m^{eq} = 0$ $(m \leq M)$ and $c_p^{ineq} > 0$ $(p \leq P)$, which are collected from all time N frames.

Table I summarizes the types of constraints supported by our computational method. They include the necessary end-effector and base link terms, as well as other functional constraints that ensure the robot's motion is collision-free. Although we only provide five types of motion constraints, they are general enough to describe the input tasks of a wide range of robotic devices, including manipulators and legged robots. It is also easy to add new constraints for future applications if needed.

For any desired motion, we must concurrently generate an optimized robot design $\mathcal{D}$ and control inputs for its motors. Given the set of objectives and constraints introduced above, this coupled design and control problem is defined as follows:

$$\begin{aligned} \min_{\mathcal{D}, \mathbf{q}_i, \cdots, \mathbf{q}_N} \; & g(\mathcal{D}) + w_r \sum_{i=1}^{N-1} |\mathbf{q}_{i+1} - \mathbf{q}_i|^2 \\ \text{s.t.} \quad & c_m^{eq}(\mathcal{D}, \mathbf{q_i}) = 0 \;\; \forall m \leq M \\ & c_p^{ineq}(\mathcal{D}, \mathbf{q_i}) \geq 0 \;\; \forall p \leq P \\ & \mathbf{l}_p \leq \mathbf{q}_i \leq \mathbf{u}_p \;\; \forall i \leq N \\ & \mathbf{l}_v \leq \dot{\mathbf{q}}_i \leq \mathbf{u}_v \;\; \forall i \leq N, \end{aligned} \tag{2}$$

where the discrete set of poses $\mathbf{q}_i$ defines the control trajectories that the robot's actuators must execute such that it achieves the desired motion. The term w_r is the regularization weight for increasing the smoothness of the resulting motion. The terms $\mathbf{l}_p$, $\mathbf{u}_p$, $\mathbf{l}_v$, and $\mathbf{u}_v$ refer to the minimum and maximum positions and velocities of actuators.

Eq. (2) describes the design optimization in a general form that can be applied to a wide range of applications. We apply Eq. (2) in examples for two nominal scenarios: *manipulators* (Eq. 8) and *legged robots* (Eq. 9).

The formulated optimization is a very challenging problem that features an unknown number of parameters taking on

both discrete and continuous values. To solve it efficiently, we present a heuristic-guided tree-search algorithm as described in the following section.

V. A* Search for Robot Designs

The use of design elements as modular building blocks presents a natural way of synthesizing novel robotic devices. Consider a robot design $\mathcal{D}$ composed of design elements $\mathcal{V}$. The connection rules specify which modules $\mathbb{V}$ in the design library are compatible with those in $\mathcal{V}$. Appending any module $y \in \mathbb{V}$ to its compatible counterpart x in $\mathcal{V}$ gives rise to new *offspring* designs, each with potentially different motor capabilities. Consequently, if design $\mathcal{D}$ is not well-suited for a user-specified motion, one of its offsprings might be.

We use this insight to formulate the problem of generating optimized robot designs as a shortest path problem. As illustrated in Fig. 1, a directed acyclic graph represents all robotic devices that are realizable given a library of design elements. Each node in this graph corresponds to a feasible design. The root is a *base link* – a special module equipped with a computing unit and a power source. A directed edge between two nodes corresponding to designs $\mathcal{D}_1$ and $\mathcal{D}_2$ implies that $\mathcal{D}_2$ is an offspring generated by appending one compatible module to $\mathcal{D}_1$. The cost associated with this edge quantifies the added complexity of the offspring design: $g(\mathcal{D}_2) - g(\mathcal{D}_1)$. We further designate *goal nodes* to be those corresponding to robot designs that are capable of executing the user-specified motion. The shortest path from the root of the design graph to a goal node reveals the robotic device that we seek.

A* is a widely used algorithm for path-finding problems. It works in a best-first search manner by exploring paths that are deemed most promising. For our problem, A* explores the design that minimizes the estimated total design cost:

$$f(\mathcal{D}) = g(\mathcal{D}) + h(\mathcal{D}), \tag{3}$$

where g quantifies the complexity of the current design (Eq. 1) and h is a heuristic function that estimates the design cost of required modules that are not yet included in $\mathcal{D}$. By fetching the most promising design (Eq. 3), we expect A* to effectively find the solution for the given design optimization problem (Eq. 2).

The nature of the heuristic measure governs the overall performance of the A* search process. If it fails to appropriately estimate the promise of different (incomplete) robotic devices, it might lead to an exploration of unnecessarily many designs or to sub-optimal results. However, estimating future design costs is difficult because even minor updates, such as adding an actuator or changing the length of a link, can significantly affect the workspace of the robot, and therefore its motor capabilities. To address this technical challenge, we introduce the concept of a *hybrid device*, which we use to estimate the changes required to complete any robot design such that it can achieve the desired motion.

A. Hybrid Devices

We define a hybrid device $\tilde{\mathcal{D}}$ to be a robotic design $\mathcal{D}$ augmented with virtual end effectors. This virtual component

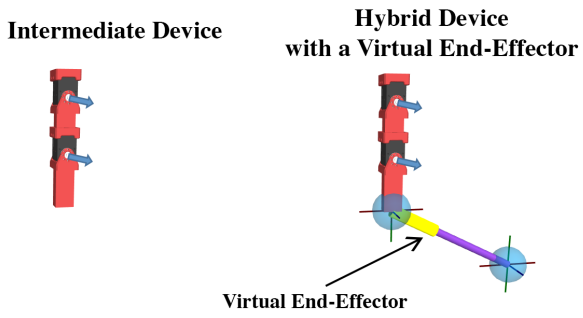


Fig. 3. An intermediate design (left) is augmented with a virtual end effector to create a hybrid device (right). The virtual end effector consists of two actuated spherical joints connected by a telescoping link. The hybrid device can solve any motion task described through desired end-effector positions and/or orientations.

is an idealized device with sufficient degrees of freedom to solve any motion. The concept of virtual end effectors is quite general, and they can in principle be defined in different ways. Given that we describe desired motions in terms of Cartesian-space motion trajectories, we define virtual end effectors as assemblies of two actuated spherical joints connected by a telescoping link, as illustrated in Fig. 3. We note that although fully functional, virtual end effectors are not physically-realizable. Further, we assume there is no explicit correspondence between their degrees of freedom and the library of modular building blocks that is employed by our computational design method. Consequently, hybrid devices only serve the purpose of guiding the search for an appropriate design, rather than providing a direct recipe for how robots should be made.

A user can input multiple target trajectories for designing multi-limb robots. In this case, we augment each leg with a virtual end effector. We assume that the correspondence between the limbs and the end-effector trajectories are given.

B. Heuristic Function based on Hybrid Devices

Virtual end effectors are, by design, sufficiently versatile to solve any input motion. However, our goal is to use them only to complement the robotic device they augment. This strategy allows our computational method to estimate how much a robotic design needs to change before it can complete the input motion by itself. The answer to this question forms the basis of the heuristic that guides the search for optimal robot designs. More formally, to compute the value of the heuristic function $h(\mathcal{D})$, we begin by solving the following optimization problem:

$$\begin{aligned} &\min_{\tilde{\mathbf{q}}_1,\cdots,\tilde{\mathbf{q}}_N} \sum_{i=1}^{N} ||\mathbf{W}\mathbf{q}_i^v||_1 \\ \text{s.t.} \quad & c_m^{eq}(\tilde{\mathcal{D}}, \tilde{\mathbf{q}}_\mathbf{i}) = 0 \;\; \forall m \leq M \\ & c_p^{ineq}(\tilde{\mathcal{D}}, \tilde{\mathbf{q}}_\mathbf{i}) \geq 0 \;\; \forall p \leq P \\ & \tilde{\mathbf{l}}_p \leq \tilde{\mathbf{q}}_i \leq \tilde{\mathbf{u}}_p \;\; \forall i \leq N \\ & \tilde{\mathbf{l}}_v \leq \dot{\tilde{\mathbf{q}}}_i \leq \tilde{\mathbf{u}}_v \;\; \forall i \leq N, \end{aligned} \tag{4}$$

where $\tilde{\mathcal{D}}$ is the hybrid device corresponding to intermediate design $\mathcal{D}$. The configuration of the hybrid device at any moment in time, $\tilde{\mathbf{q}}$, is defined as $[\mathbf{q}, \mathbf{q}^v]$, where $\mathbf{q}^v$ describes the positions of the virtual end effector's joints. The objective minimizes the contribution of each DOF of the virtual end effector, based on a sparsity-inducing L1-norm. Matrix $\mathbf{W}$ is a diagonal weight matrix for adjusting the scales and priorities of linear and angular joints. We prioritize linear joints by assigning larger weights while assigning smaller weights to angular joints.

With this objective in place, a non-zero value for any virtual DOF highlights the need for a change in the mechanical structure of the current design $\mathcal{D}$, and indicates that the design is lacking in terms of actuation capabilities. Throughout the motion optimization process, the degrees of freedom of the original robot design can be used as much as necessary, without penalty. Consequently, if $\mathcal{D}$ is already well-suited for the input motion, $\tilde{\mathcal{D}}$'s virtual DOFs have no reason to be used.

In Eq. (4), the objective function only considers the virtual end effector because our goal is to estimate additional design costs until being able to execute the given task. However, we still need to solve the entire motions for both virtual and non-virtual joints because the functionality of the robotic device depends on both the design and motion. On the other hand, we do not need to solve the motion to evaluate the current design cost (Eq. 1).

Based on the solution to Eq. (4), we evaluate the heuristic term $h(\mathcal{D})$ in Eq. (3) as follows:

$$\begin{aligned} h(\tilde{\mathcal{D}}, \tilde{\mathbf{q}}_1, \cdots, \tilde{\mathbf{q}}_N) =& w_s \max_{1 \le i \le N} t(\tilde{\mathcal{D}}, \mathbf{q}_i^v) \\ &+ w_a n(\mathbf{q}_1^v, \cdots, \mathbf{q}_N^v), \end{aligned} \tag{5}$$

where t extracts the length of the telescopic segment and n counts the number of joints with non-zero velocities. More precisely, the two terms are defined as:

$$t(\tilde{\mathcal{D}}, \mathbf{q}) = \sum_{j \in \tilde{\mathcal{J}}^{tl}(\tilde{\mathcal{D}})} |\mathbf{q}(j)| \tag{6}$$

$$n(\mathbf{q}_1, \cdots, \mathbf{q}_N) = \sum_j |\max_i \mathbf{q}_i(j) - \min_i \mathbf{q}_i(j)|_0 \tag{7}$$

where $\tilde{\mathcal{J}}^{tl}$ refers to the indices of the telescopic joints in the virtual end-effectors of the given hybrid device $\tilde{\mathcal{D}}$. Following the reasoning behind the definition of $g(\mathcal{D})$ (Eq. 1), the first term in Eq. (5) estimates the volume of material needed by the virtual end effector. Its value is proportional to the maximum length of the telescoping segment. The second term is proportional to the number of non-zero elements in velocities. We combine terms using the same weights w_s and w_a because both terms estimate the future costs of the corresponding terms in Eq. (1). We illustrate the examples of heuristic values in Fig. 4.

VI. Results

We validated our design optimization algorithm on two main types of robotic devices: manipulators and legged robots. The simulation and optimization are implemented in Python with the PyDART [39] and SciPy library [40] on Ubuntu Linux, and the computations are conducted on a single core of 3.40GHz Intel i7 processor. The parameters used in the examples are summarized in Table II.

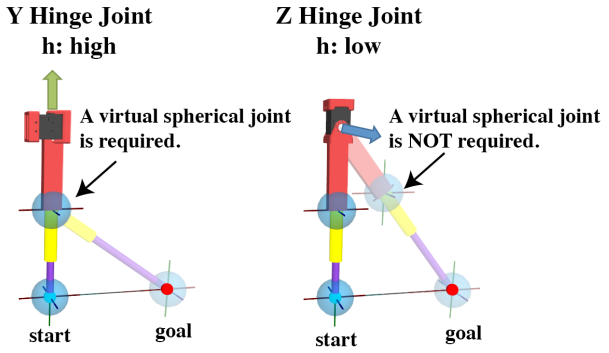


Fig. 4. Our hybrid-design based heuristic estimates the future design cost using virtual end effectors. The left design must extensively use the first spherical joint of the virtual end effector, which results in a high heuristic value. On the other hand, the right design can track the trajectory only using the telescopic joint, which results in a low heuristic value.

TABLE II
Problem Parameters

	w_a	w_r	W_{lnr} [a]	W_{ang} [b]	Δl [c]
Manipulators	0.3	0.001	1.0	0.01	0.1m
Legged Robots	0.1	0.001	1.0	0.005	0.01m

[a] W_{lnr} is the value of the weight matrix $\mathbf{W}$ for a linear joint in Eq. (4).
[b] W_{ang} is the value of the weight matrix $\mathbf{W}$ for an angular joint in Eq. (4).
[c] Δl represents the length of the unit link.

A. Robotic Kits

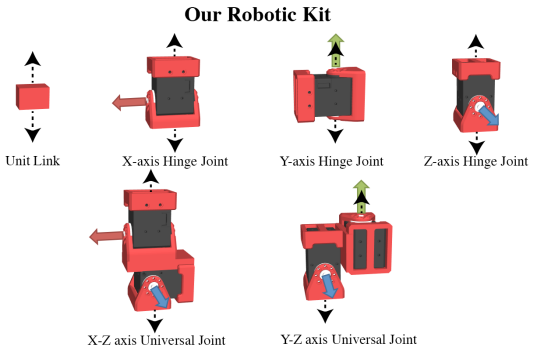


Fig. 5. Our robotic kit that include off-the-shelf actuators and 3D printable connectors.

We designed the modular robotic kit that uses commercial actuators [4] and 3D-printed connectors. Dynamixel is a commercial actuator with a controller and a network module. We used Dynamixel servos as our actuators, and fabricated links and brackets using 3D printing. We define seven design elements: a link with unit length, three hinge joints (X, Y, and Z axes), and two universal joints (X-Z and Y-Z axes), as shown in Fig. 5. We do not include the Y-X universal joint because it

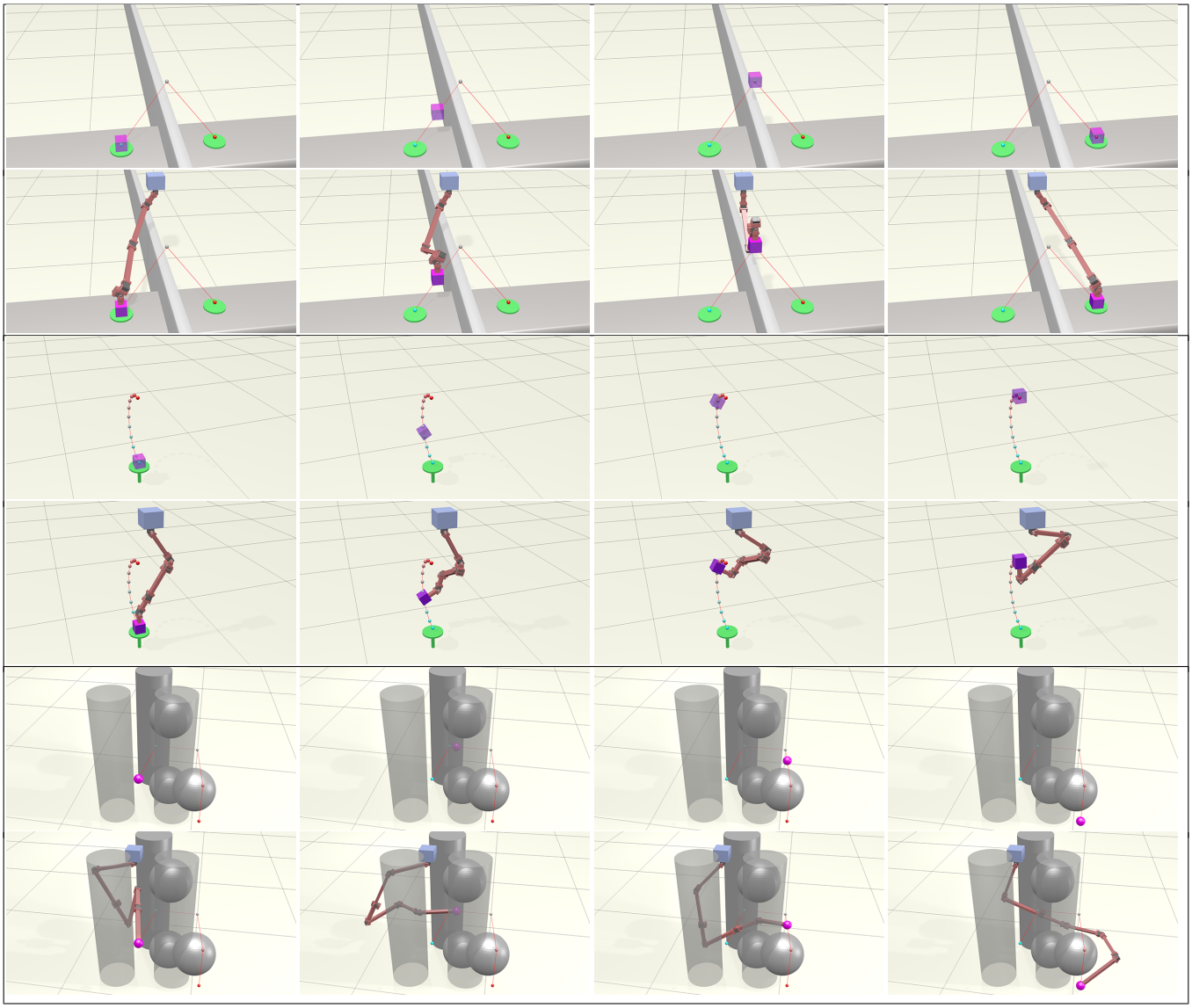


Fig. 6. The co-optimized designs and motions of three manipulators. Each row demonstrates the input trajectories and optimized designs for the *object moving* task (the first and second rows), the *object picking* task (the third and fourth rows), and the *inspection* task (the fifth and sixth rows).

is identical to the Y-Z joint with the 90° offset at the Y joint. The elements can be connected to each other at pin locations, which are rendered as arrows in Fig. 5. We differentiate X and Z axes because we fixed relative orientations of modules. We used the unit link with the different lengths for different sets of the problems, as listed in Table II.

B. Manipulators

Using our own robotic kit, we first designed a set of manipulators based on different desired capabilities. Concisely designed manipulators for specific tasks have many advantages, such as reduced complexity and costs.

We defined three types of tasks for our manipulator examples: *object moving*, *object picking*, and *inspection*, all specified through end-effector trajectories. The goal is to track these input target trajectories with end-effectors while avoiding collisions with the environment. The *object moving* and *object picking* examples have additional orientation constraints for the target object, while the *inspection* example only has the positional constraints. We assume that the position of the base link is given as input and kept fixed. Therefore, the design optimization (Eq. 2) can be written as:

$$\begin{aligned} \min_{\mathcal{D},\mathbf{q}_i,\cdots,\mathbf{q}_N} \; & g(\mathcal{D}) + w_r \sum_{i=1}^{N-1} |\mathbf{q}_{i+1} - \mathbf{q}_i|^2 \\ \text{s.t.} \quad & \mathbf{E}^1(\mathcal{D},\mathbf{q}_i) = \bar{\mathbf{E}}_i^1 \;\; \forall i \le N \\ & \mathbf{R}^1(\mathcal{D},\mathbf{q}_i) = \bar{\mathbf{R}}_i^1 \;\; \forall i \le N \\ & d_{jk}(\mathcal{D},\mathbf{q}_i) \ge 0 \;\; \forall i \le N, \;\; j,k \le L \\ & e_j(\mathcal{D},\mathbf{q}_i,\Phi) \ge 0 \;\; \forall i \le N, \;\; j \le L \\ & \mathbf{l}_p \le \mathbf{q}_i \le \mathbf{u}_p \;\; \forall 1 \le i \le N \\ & \mathbf{l}_v \le \dot{\mathbf{q}}_i \le \mathbf{u}_v \;\; \forall 1 \le i \le N, \end{aligned} \tag{8}$$

where L refers to the number of rigid links. The functions $\mathbf{E}^1$ and $\mathbf{R}^1$ evaluate the position and orientation of the single end-effector at the ith frame, while $\bar{\mathbf{E}}_i^1$ and $\bar{\mathbf{R}}_i^1$ are taken from the

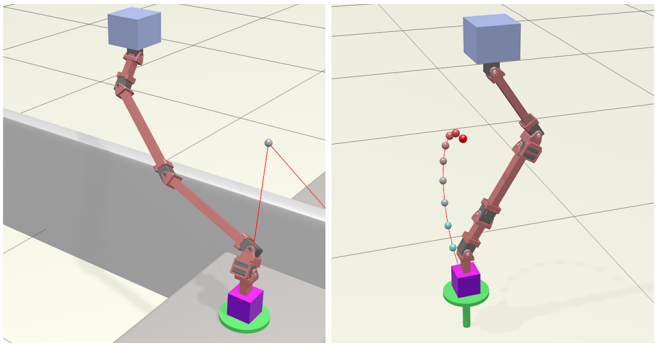


Fig. 7. Different task specifications lead to different robot designs (left: *object moving*, right: *object picking*).

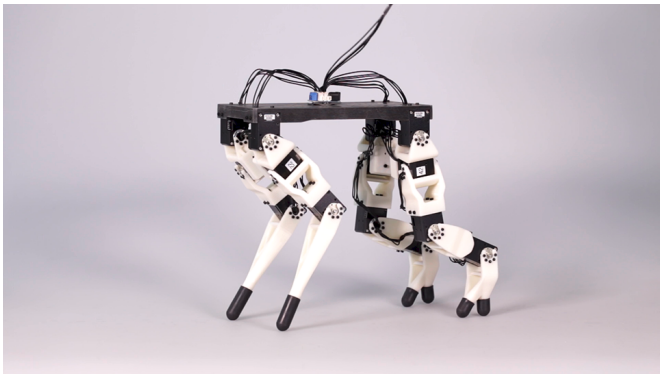


Fig. 8. The fabricated Puppybot.

input trajectory. Also note that the orientation constraint $\bar{\mathbf{R}}_i^1$ is optional, as shown in the *inspection* example. Finally, d_{jk} and e_j prevent self-collision and collision against the given environment Φ for all L links.

Please see Fig. 6 and the supplementary video for the optimized designs and their motions. Although we expected the design with six servos for the *object moving* example due to additional three orientation constraints, the algorithm finds the manipulator design with five servos (Fig. 6). In addition, the optimized manipulator for the *object picking* example also has the simple design with five servos including one universal joint, which is less than our expectation. Also please note that different task specifications result in different manipulator designs as shown in Fig. 7. The *inspection* example has the cluttered environment with three pillars and three balls. Although three servos is enough when there are no obstacles, the optimized manipulator has six servos to track the trajectory while avoiding collisions.

Since the objective is to find the minimal design that can track the entire target trajectory, the optimized design may cause near-singular poses. To avoid singularity issues, we suggest to expand the workspace by enlarging the input trajectory.

C. Legged Robots

Although we illustrated our algorithm in the context of a single manipulator, we can also apply the algorithm to designing legged robots by combining with any operational-space trajectory generation algorithm. In the pipeline, a user only needs to specify the desired initial and final positions of the base-link and end-effectors. Then our system generates complete trajectories by applying the space-time optimization [19]. Finally, our algorithm takes as input the generated trajectories and finds the optimal designs of limbs.

1) Puppybot: We first tested our algorithm on designing an animal-like robot that can walk forward. The center of mass and foot trajectories are generated by solving space-time optimization using centroidal dynamics [19]. The base link trajectory is set to a few centimeters above the center of mass trajectory. Besides the generated base link and end-effector trajectories, we have additional constraints that all links must remain above the ground. For testing different specifications, we also placed additional orientation constraints to rear feet so that they remain upright during locomotion. To account for extra orientation constraints, we independently designed the front and rear legs based on their desired trajectories relative to the base link. Therefore, the details of the design optimization (Eq. 2) are:

$$
\begin{aligned}
&\min_{\mathcal{D},\mathbf{q}_i,\cdots,\mathbf{q}_N} g(\mathcal{D}) + w_r \sum_{i=1}^{N-1} |\mathbf{q}_{i+1} - \mathbf{q}_i|^2 \\
\text{s.t.}\quad &\mathbf{E}^h(\mathcal{D},\mathbf{q}_i) = \bar{\mathbf{E}}_i^h \;\; \forall i \le N, h \le H \\
&\mathbf{R}^h(\mathcal{D},\mathbf{q}_i) = \bar{\mathbf{R}}_i^h \;\; \forall i \le N, h \le H \\
&\mathbf{P}(\mathcal{D},\mathbf{q}_i) = \bar{\mathbf{P}}_i \;\; \forall i \le N \\
&d_{jk}(\mathcal{D},\mathbf{q}_i) \ge 0 \;\; \forall i \le N, \;\; j,k \le L \\
&e_j(\mathcal{D},\mathbf{q}_i,\Phi) \ge 0 \;\; \forall i \le N, \;\; j \le L \\
&\mathbf{l}_p \le \mathbf{q}_i \le \mathbf{u}_p \;\; \forall 1 \le i \le N \\
&\mathbf{l}_v \le \dot{\mathbf{q}}_i \le \mathbf{u}_v \;\; \forall 1 \le i \le N,
\end{aligned}
\tag{9}
$$

where L is the number of rigid links in the design $\mathcal{D}$ and H is the number of end-effectors. As L and $\mathcal{D}$ change during the search, H remains constant ($H = 4$ for the Puppybot). The foot-fall pattern of the operational space motion plan is implicitly encoded in the desired end-effector trajectories (the rth end-effector is in contact if and only if the desired position $\bar{\mathbf{E}}_i^r$ is on the ground). Finally, e_j only checks for collisions with the ground.

The optimized quadruped has three servos for the front legs and four servos for the rear legs. Based on the optimized design, an engineer generated required link meshes and base boards (Fig. 8, Top). Feet are fabricated as rubber-coated hemispheres. We tethered the fabricated quadruped to a computer and replayed the trajectories that were concurrently optimized with the morphological design. The joint trajectories were tracked by local servos and we did not implement any balance controller. Although the fabricated Puppybot had larger movements in the side direction due, e.g., to joint slackness and link deformation, it was able to walk for multiple gait cycles. Please refer to the supplemental video for simulated and real motions of the Puppybot.

2) Tetrabot: Inspired by the work of Pai *et al.* [41], we optimized the design of a spherically symmetric quadruped that is reminiscent of a tetrahedron, so called *Tetrabot*. The

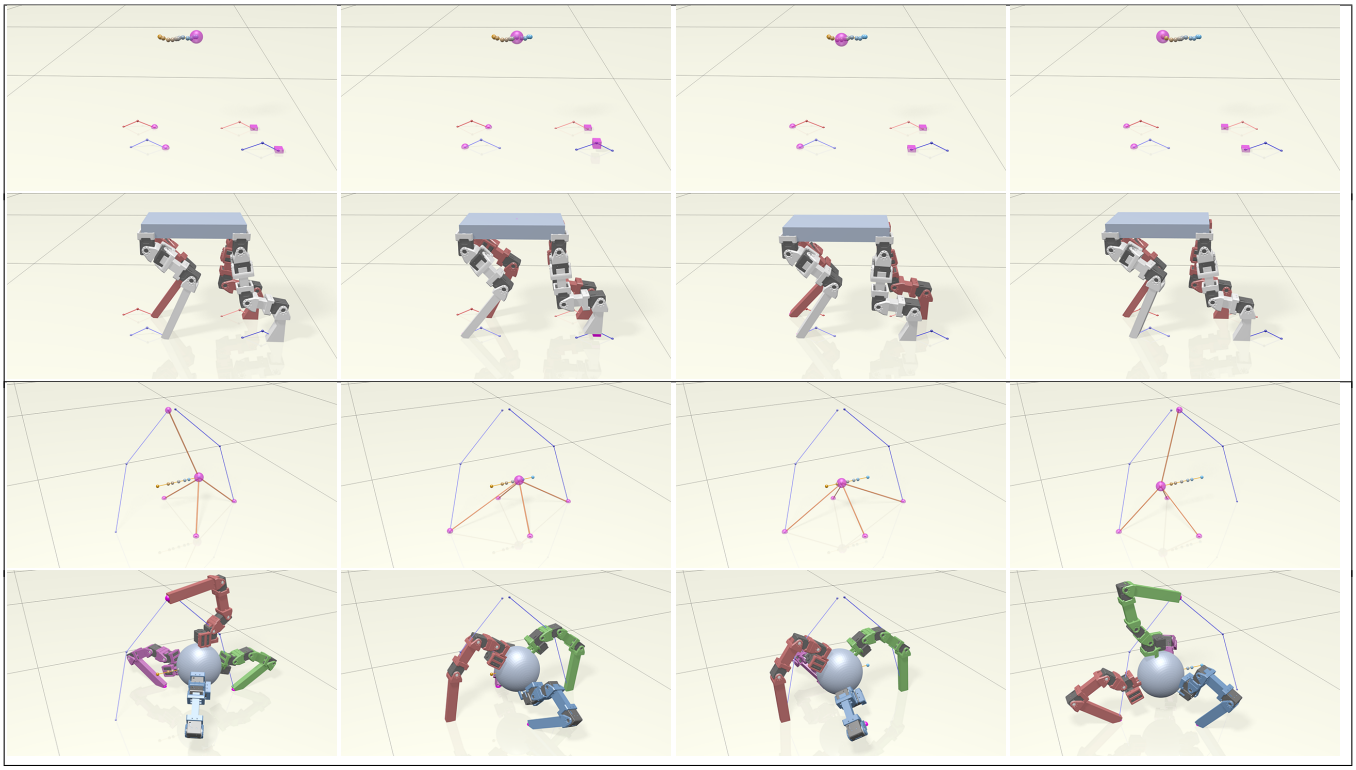


Fig. 9. The co-optimized designs and motions of two legged robots. Each row demonstrates the input trajectories and optimized designs of the *Puppybot* (the first and second rows) and the *Tetrabot* (the third and fourth rows).

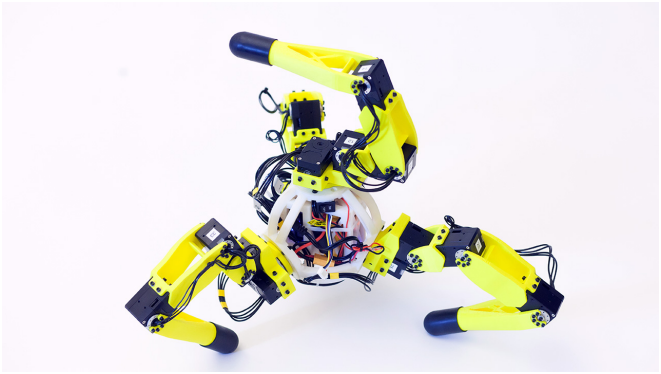


Fig. 10. The fabricated Tetrabot.

walking mechanism of this *Tetrabot* is unintuitive yet surprisingly simple: starting from the initial tetrahedron shape, it gradually pushes its base link toward the edge while preparing the landing with the leg on the top. By moving the base link beyond the edge of the support polygon, the robot gently falls toward the edge, and lands on the adjacent equilateral triangle. After landing, it goes to the initial tetrahedron shape once again. Based on this intuition, we generated the input trajectories by interpolating positions and orientations of two adjacent tetrahedrons while keeping the horizontal position of the base-link in the support polygon with 2 cm margins.

In our experience, designing a spherically symmetric robot was not intuitive even for experts due to non-orthogonal coordinate frames, collisions, and joint angle limits. However, our optimizer was able to find a minimal robot design that has one Y-Z universal joint and two Z-axis hinge joints (Fig. 9 and Fig. 10). One noticeable difference to the design of Pai *et al.* with three servos per legs is that our design uses four servos per legs due to the joint angle limits on the defined Dynamixel kit, which was against our intuition. We verified the minimality of the design by running an exhaustive search with three DoFs limits, which resulted in no solutions. The algorithm found a three DoF solution only when we increased the joint angle limits from $\pm 135^\circ$ to $\pm 160^\circ$, which is difficult to realize as a durable physical prototype. Once again, we fabricated the untethered Tetrabot using Dynamixels and 3D-printing. The fabricated Tetrabot was able to walk multiple steps without losing its balance: please refer to the supplemental video for its motion.

D. Comparison of Algorithms

We compared the algorithm we propose, A* with a hybrid device based heuristics function (**Hybrid**), to two baseline algorithms: breadth-first search (**BFS**) and A* with an error-based heuristic function (**Error**). The error-based heuristic function estimates the effectiveness of the intermediate device by converting all constraints into soft constraints and calculating a sum of errors. Because this error-based heuristic has different units than the cost function (Eq. 1), it is difficult to intuitively interpret the effects of combining the cost and heuristic terms.

We use the number of expanded nodes until finding the best design as comparison criteria. Note that it is not intuitive to analytically prove the optimality of the generated designs due to the combinatorial nature of the problem. To ensure that we

TABLE III
THE RESULT STATISTICS.

Problem				Optimal Design		# Expanded Nodes (Elapsed Time in Minutes) [a]		
Type	**Task**	**# Frames**	**Duration(sec)**	**# Servos**	**$\sum$ Length**	**BFS**	**Error**	**Hybrid**
Manipulator	Object moving	21	4.0	5	0.60	10000+	10000+	2267 (76)
Manipulator	Object picking	21	4.0	5	0.80	10000+	10000+	2224 (79)
Manipulator	Inspection	21	4.0	6	1.60	20000+	20000+	13203 (402)
Legged Robot	Puppybot	13	1.5	14	1.14	501 (81)	408 (72)	276 (56)
Legged Robot	Tetrabot	21	4.0	16	0.96	10000+	1430 (307)	165 (37)

[a]The plus sign (+) indicates that the number of expanded nodes reached the limit without finding a feasible design.

find the optimal design, we let the optimization run until it enumerates all the available configurations with lower design costs to show the optimality of the solutions.

The result statistics are presented in Table III. For all examples, the hybrid device based heuristic shows the best performance by finding the optimal design with a minimal number of expanded nodes. We have noticed that sometimes it finds non-optimal functional designs first and requires additional 10% to 20% more nodes to find the true optimal design for the subset of problems (*object moving*, *object picking*, and *Puppybot*). We believe the local solutions in Eq. (4) are the cause of the inadmissibility, which requires further examination. BFS shows good performance for the simple problem (*Puppybot*) but fails to find any functional design within the evaluation limits for all the other problems due to its exponential time complexity with respect to the number of modules in the optimal design. For instance, it needs to evaluate at least a few million designs to solve the *inspection* example. The A* with the error-based heuristic generally expands three to ten times more nodes than the A* with the hybrid device based heuristic that we propose.

VII. DISCUSSION AND FUTURE WORK

We presented a novel heuristics algorithm that optimizes designs of robotic devices for user-provided motion tasks. The design parameters include the number of links per limbs, joint types, link lengths, and motor control signals. Our key insight is to formulate the design optimization as a shortest path-finding problem, which can be solved by an A* algorithm driven by a novel heuristic function. The robustness of the proposed algorithm is validated by generating optimized designs for various manipulators and legged robots. Further, the efficiency of the algorithm is demonstrated by comparing with two baseline algorithms, a breadth-first search and an A* algorithm with a simple error-based heuristic function.

The discrete nature of our algorithm prevents it from optimizing finer details of continuous variables, such as link lengths, link offsets, or servo orientations, which may lead to better designs with lower costs. Further, the optimization of these variables has other practical values such as avoiding self-collisions or adjusting ranges of motions. We expect that these extra continuous parameters can be further optimized as a post-process as suggested in prior works [12], [20], [19].

In practice, our heuristic search is not always admissible. Although the admissibility is not a necessary condition for A* search, it can be a convenient tool for guaranteeing the optimality of the solution. We suspect that non-convex optimization problems in Eq. 1 and Eq. 4 cause non-admissibility. If a globally-optimal solver could be devised, we believe our heuristic function would be admissible, which is one possible direction for future work. Another interesting approach is to adjust admissibility by statically or dynamically weighting the intermediate and heuristic costs during the search process [42], [43].

Another direction of future work is to consider dynamics or control of robots, rather than focusing on only kinematic capabilities. For instance, we can add an additional objective term for minimizing a sum of squared torques or constraints on maximum torques. Otherwise, we can test whether a robot can exert desired forces at its end-effectors by checking joint singularities. We believe the proposed searching framework can be extended with additional constraints.

In all examples, we only took a single motion plan as an input to find specialized robot designs. In practice, many working environments require versatile robots that can handle multiple related tasks, such as manipulating different types of objects. Therefore, it will be interesting to design a robot for a family of parameterized motion plans.

ACKNOWLEDGEMENTS

This research was fully funded by Disney Research.

REFERENCES

[1] N. Sporer, M. Hdmle, R. Krenn, A. Pascucci, and M. Schedl, "DLR's torque-controlled light weight robot," *Mechatronics*, no. May, pp. 1710–1716, 2002.

[2] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaffer, A. Beyer, O. Eiberger, S. Haddadin, G. Stemmer, Andreas Grunwald, and Others, "The KUKA-DLR Lightweight Robot arm-a new reference platform for robotics research and manufacturing," *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–8, 2010.

[3] M. Fujita and H. Kitano, "Development of an Autonomous Quadruped Robot for Robot Entertainment," *Autonomous Robots*, vol. 5, pp. 7–20, 1998.

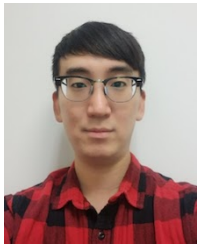
[4] Dynamixel, *http://robotis.com*, 2016.

[5] A. Crespi, K. Karakasiliotis, A. Guignard, and A. J. Ijspeert, "Salamandra Robotica II: An amphibious robot to study salamander-like swimming and walking gaits," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 308–320, 2013.

[6] S. Seok, A. Wang, M. Y. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, "Design Principles for Energy-Efficient Legged Locomotion and Implementation on the MIT Cheetah Robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1117–1129, 2014.

[7] K. Graichen, S. Hentzelt, A. Hildebrandt, N. Kärcher, N. GaiBert, and E. Knubben, "Control design for a bionic kangaroo," *Control Engineering Practice*, vol. 42, pp. 106–117, 2015.

[8] D. Kuehn, F. Bernhard, A. Burchardt, M. Schilling, T. Stark, M. Zenzes, and F. Kirchner, "Distributed computation in a quadrupedal robotic system," *International Journal of Advanced Robotic Systems*, vol. 11, no. 1, 2014.
[9] G. Nelson, R. Quinn, R. Bachmann, W. Flannigan, R. Ritzmann, and J. Watson, "Design and simulation of a cockroach-like hexapod robot," *Proceedings of International Conference on Robotics and Automation*, vol. 2, no. April, pp. 3–8, 1997.
[10] S. Song, J. Kim, and K. Yamane, "Development of a bipedal robot that walks like an animation character," *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
[11] C. J. Paredis and P. K. Khosla, "An Approach for Mapping Kinematic Task Specifications into a Manipulator Design," *International Conference on Advanced Robotics*, vol. 1, 1991.
[12] M. Ceccarelli and C. Lanni, "A multi-objective optimum design of general 3R manipulators for prescribed workspace limits," *Mechanism and Machine Theory*, vol. 39, no. 2, 2004.
[13] E. Van Henten, D. Vant Slot, C. Hol, and L. Van Willigenburg, "Optimal manipulator design for a cucumber harvesting robot," *Computers and Electronics in Agriculture*, vol. 65, no. 2, pp. 247–257, 2009.
[14] S. G. Kim and J. Ryu, "New dimensionally homogeneous jacobian matrix formulation by three end-effector points for optimal design of parallel manipulators," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 4, pp. 731–737, 2003.
[15] J.-F. Collard, P. Fisette, and P. Duysinx, "Contribution to the Optimization of Closed-Loop Multibody Systems: Application to Parallel Manipulators," *Multibody System Dynamics*, vol. 13, no. 1, pp. 69–84, 2005.
[16] Y. Yun and Y. Li, "Optimal design of a 3-PUPU parallel robot with compliant hinges for micromanipulation in a cubic workspace," *Robotics and Computer-Integrated Manufacturing*, 2011.
[17] C. D. Jung, W. J. Chung, J. S. Ahn, M. S. Kim, G. S. Shin, and S. J. Kwon, "Optimal mechanism design of in-pipe cleaning robot," *IEEE International Conference on Mechatronics and Automation*, pp. 1327–1332, 2011.
[18] D. Kim, H. Hong, H. S. Kim, and J. Kim, "Optimal design and kinetic analysis of a stair-climbing mobile robot with rocker-bogie mechanism," *Mechanism and Machine Theory*, vol. 50, 2012.
[19] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Task-based Limb Optimization for Legged Robots," *International Conference on Intelligent Robots and Systems*, 2016.
[20] K. Wampler and Z. Popović, "Optimal gait and form for animal locomotion," *ACM Transactions on Graphics*, vol. 28, p. 1, 2009.
[21] K. Sims, "Evolving Virtual Creatures," *ACM Transactions on Graphics*, no. July, pp. 15–22, 1994.
[22] C. Leger, "Automated Synthesis and Optimization of Robot Configurations : An Evolutionary Approach," *Design Engineering*, pp. 1–234, 1999.
[23] H. Lipson and J. Pollack, "Towards continuously reconfigurable self-designing robotics," *IEEE International Conference on Robotics and Automation. Symposia Proceedings*, vol. 2, no. April, pp. 1761–1766, 2000.
[24] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms." *Nature*, vol. 406, no. 6799, pp. 974–978, 2000.
[25] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding," *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation*, p. 167, 2013.
[26] M. Bächer, B. Bickel, D. L. James, and H. Pfister, "Fabricating articulated characters from skinned meshes," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 1–9, 2012.
[27] J. Cali, D. A. Calian, C. Amati, R. Kleinberger, A. Steed, J. Kautz, and T. Weyrich, "3D-Printing of Non-Assembly, Articulated Models," *Acm Transactions on Graphics*, vol. 31, no. 6, 2012.
[28] R. Prévost, E. Whiting, S. Lefebvre, and O. Sorkine-Hornung, "Make It Stand: Balancing Shapes for 3D Fabrication," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 81:1—-81:10, 2013.
[29] M. Bächer, E. Whiting, B. Bickel, and O. Sorkine-Hornung, "Spin-it: Optimizing Moment of Inertia for Spinnable Objects," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.
[30] B. Koo, W. Li, J. Yao, M. Agrawala, and N. J. Mitra, "Creating works-like prototypes of mechanical objects," *ACM Transactions on Graphics*, vol. 33, no. 6, pp. 1–9, 2014.
[31] N. Umetani, Y. Koyama, R. Schmidt, and T. Igarashi, "Pteromys: Interactive Design and Optimization of Free-formed Free-flight Model Airplanes," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 1–10, 2014.
[32] L. Zhu, W. Xu, J. Snyder, Y. Liu, G. Wang, and B. Guo, "Motion-guided Mechanical Toy Modeling," *ACM Transactions on Graphics*, vol. 31, no. 6, pp. 127:1–127:10, 2012.
[33] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R. W. Sumner, W. Matusik, and B. Bickel, "Computational design of mechanical characters," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 1, 2013.
[34] D. Ceylan, W. Li, N. J. Mitra, M. Agrawala, and M. Pauly, "Designing and fabricating mechanical automata from mocap sequences," *ACM Transactions on Graphics*, 2013.
[35] B. Thomaszewski, S. Coros, D. Gauge, V. Megaro, E. Grinspun, and M. Gross, "Computational Design of Linkage-Based Characters," *ACM Transactions on Graphics*, vol. 33, 2014.
[36] G. Bharaj, S. Coros, B. Thomaszewski, J. Tompkin, B. Bickel, and H. Pfister, "Computational Design of Walking Automata," *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 93–100, 2014.
[37] V. Megaro, B. Thomaszewski, M. Nitti, O. Hilliges, M. Gross, and S. Coros, "Interactive Design of 3D-Printable Robotic Creatures," *ACM Transactions on Graphics*, 2015.
[38] T. Du, A. Schulz, M. Csail, B. Zhu, B. Bickel, and W. Matusik, "Computational Multicopter Design," *ACM Transactions on Graphics*, vol. 35, 2016.
[39] PyDART, *A Python Binding of Dynamic Animation and Robotics Toolkit, http://pydart2.readthedocs.io*.
[40] E. Jones, T. Oliphant, and P. Peterson, "{SciPy}: open source scientific tools for {Python}," 2014.
[41] D. K. Pai, R. A. Barman, and S. K. Ralph, "Platonic beasts: Spherically symmetric multilimbed robots," *Autonomous Robots*, vol. 2, no. 3, pp. 191–201, 1995.
[42] J. Pearl, "Heuristics: intelligent search strategies for computer problem solving," 1984.
[43] I. Pohl, "The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving," in *Proceedings of the 3rd international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., 1973, pp. 12–17.



Sehoon Ha Sehoon Ha is currently a postdoctoral fellow at Carnegie Mellon University (CMU). Before joining CMU, he worked at Disney Research Pittsburgh as an associate research scientist. He received his Ph.D. degree in Computer Science from the Georgia Institute of Technology. Sehoons research lies at the intersection between computer graphics and robotics, including physics-based control, fabrication, design optimization, and deep reinforcement learning. He developed computational approaches to automate the complex robot development process by jointly optimizing hardware and software. He also proposed a set of general computational methods for designing a variety of agile motion controllers for virtual and real robots.

Stelian Coros Dr. Stelian Coros leads the Computational Robotics Lab (CRL), a research group within the Institute for Pervasive Computing at ETH Zurich. His research bridges the fields of Computer Graphics, Robotics and Computational Fabrication. His research group draws insights from computer science, applied mathematics and control theory to establish the foundations for algorithms that address a variety of computational problems in robotics. Applications of the work range from studying the principles of dexterous manipulation and legged locomotion to computation-driven design of novel types of robots.

Alex Alspach Alex Alspach has been working and researching in robotics since the outset of his university career. While still in school, he spent time in Drexel Universitys Autonomous Systems Lab (DASL) and South Koreas Advanced Institute of Science and Technology (KAIST) studying the manufacture and maintenance of their humanoid robot, HUBO. After graduating from Drexel, he spent two more years in Korea at SimLab in Seoul where he developed and marketed hardware and software tools for robotic manipulation research. While in Korea, he also worked with a professional production company to develop artists software tools for animating complex, constrained and synchronized robot motions. After that, he spent two years on huggable robots and various other systems at Disney Research in Pittsburgh. Now an engineer at Toyota Research Institute (TRI) in Cambridge, MA, Alex designs and fabricates soft systems for sensing and manipulation. Personal website: http://alexalspach.com

James M. Bern James M. Bern is a Ph.D. candidate in computer science at ETH Zurich. He received his B.S. in mechanical engineering (2015) from the California Institute of Technology, and his M.S. in robotics (2017) from the Carnegie Mellon University Robotics Institute. His current research is on the development of highly accessible soft robots, with a focus on simulation-based modeling and control.

Joohyung Kim Joohyung Kim is currently a Research Scientist in Disney Research, Los Angeles. His research interests include design and control of animation character robots, balancing and walking control for humanoid robots, and safe human-robot interaction. He received BSE and Ph.D. degrees in Electrical Engineering and Computer Science from Seoul National University, Korea, in 2001 and 2012. Prior to joining Disney Research, he was a Postdoctoral Fellow in CMU for DARPA Robotics Challenge in 2013. From 2009 to 2012, he was a Research Staff Member at Samsung Advanced Institute of Technology, Korea, developing biped walking controllers for humanoid robots.

Katsu Yamane Dr. Katsu Yamane is a Senior Scientist at Honda Research Institute, USA. He received his B.S., M.S., and Ph.D. degrees in Mechanical Engineering in 1997, 1999, and 2002 respectively from the University of Tokyo, Japan. Prior to joining Honda in 2018, he was a Senior Research Scientist at Disney Research, an Associate Professor at the University of Tokyo and a postdoctoral fellow at Carnegie Mellon University. Dr. Yamane is a recipient of King-Sun Fu Best Transactions Paper Award and Early Academic Career Award from IEEE Robotics and Automation Society, and Young Scientist Award from Ministry of Education, Japan. His research interests include humanoid robot control and motion synthesis, physical human-robot interaction, character animation, and human motion simulation.