Task-based design of cable-driven articulated mechanisms

Jian Li
McGill University

Sheldon Andrews
McGill University

Krisztian G. Birkas
McGill University

Paul G. Kry
McGill University

ABSTRACT

We present a framework for the automatic design of articulated cable-driven mechanisms performing push and pick-and-place tasks. Provided an initial topology and task specification, our system optimizes the morphology and cable mechanisms such that the resulting mechanism can perform the desired task successfully. Optimizing for multiple tasks and multiple cables simultaneously is possible with our framework. At the core of our approach is an optimization algorithm that analyzes the kinematics of the design to evaluate the mechanism's ability to perform the task. Dynamical attributes, such as the ability to produce forces at the end effector, are also considered. Furthermore, this paper presents a novel approach for fast inverse kinematics using cable-driven mechanisms, which is used in the morphology optimization process. Several examples of mechanisms designed using our framework are presented. We also present results of physics based simulation, and evaluate 3D printed versions of an example mechanism.

CCS CONCEPTS

• **Computing methodologies** → *Computer graphics*;

KEYWORDS

articulated structures, constraints, cables, inverse kinematics, 3D printing

ACM Reference format:
Jian Li, Sheldon Andrews, Krisztian G. Birkas, and Paul G. Kry. 2017. Task-based design of cable-driven articulated mechanisms. In *Proceedings of SCF '17, Cambridge, MA, USA, June 12-13, 2017,* 12 pages.
DOI: http://dx.doi.org/10.1145/3083157.3083161

1 INTRODUCTION

Designing articulated mechanisms has many challenges that largely prevent the average user from participating in the creation of custom robotics. However, with the democratization of 3D printing technology, low cost actuators, and embedded computers, there has been a growing interest in the research and development of computational tools that work towards removing these barriers. Our interest within this space is the design of cable driven mechanisms. Specifically, we focus on mechanisms that have a minimal number of cable actuators, yet produce varied and useful motions that solve a specified set of tasks thanks to their morphology, the organization of joints and cable attachments, and the inclusion of passive stiffnesses at the joints.

Cable driven mechanisms have the advantage that the actuators are located away from the joints, allowing them to have light structures. But this also poses an interesting challenge for deciding how and where cables should be attached to links. These choices influence the torque that can be produced at joints, and furthermore, these torques will depend on the pose of the structure. Including springs at the joints has the advantage that we can drive a mechanism with many links using fewer actuators. In fact, we could even choose to use just a single cable. However, this makes design problems trickier since compliant joints introduce behavior that can influence the kinematics of the mechanism in unintuitive ways. For a given set of cable actuations, we must solve for the static equilibrium pose of the linkage while also considering joint limits.

In this paper, we describe a method for designing custom, task-specific cable driven mechanisms An overview of our approach is shown in Figure 1. The user selects a given input topology of links and a set of actuators. The user also provides the specification for a set of tasks that involve producing a force and displacement at an end effector at different locations and in different directions. We then use optimization to find a mechanism design that achieves the tasks. This involves varying link lengths, changing cable attachment points, and adjusting the stiffness of compliant joints. We build on common computer animation tools, such as inverse kinematics (IK), analysis of physics based models, and collision detection. An important part of our contribution is the development of a fast cable IK technique that permits efficient optimization.

We evaluate our technique for a variety of different problems, such as pushing buttons on a keypad, or cutting wires in a bomb disposal. We see our design tool as just the first step. We test our designs within physics based simulations, and fabricate and evaluate several versions of one example mechanism. We believe our work will ultimately have applications such as toys, home and office automation, and other scenarios that can benefit from an inexpensive low-fidelity solution.

2 RELATED WORK

Because our work builds upon ideas in different domains, we divide the discussion or related work into three broad categories: computational design tools, robotics, and cable-driven mechanisms.

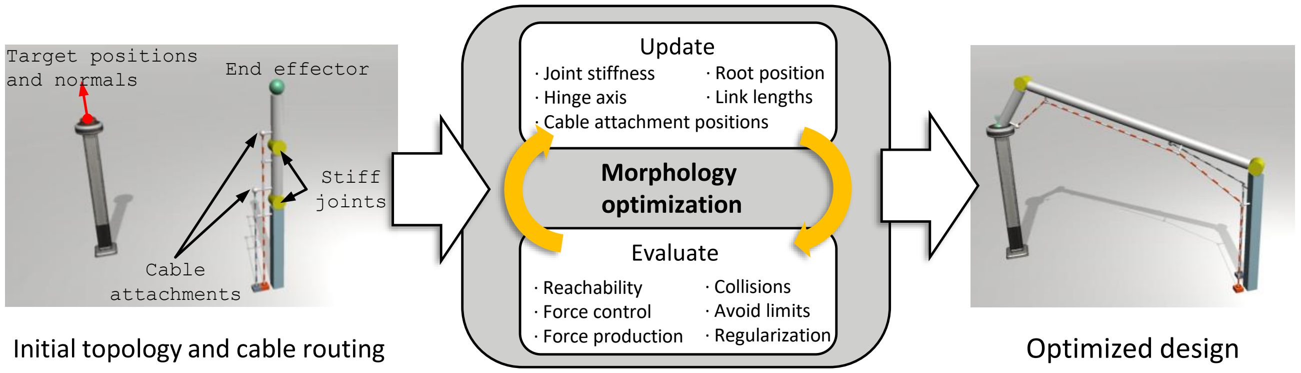


Figure 1: An overview of our system. At left, the user specifies as input both the task, and the topology of the mechanism. In this case, to push the button, the task involves producing a force and a displacement at the location of the button, while the template topology is a three link arm actuated by two cables. At right, the result of our optimization is a mechanism that can perform the specified task due to an appropriate morphology, cable routing, and joint stiffnesses.

2.1 Computational design tools

Design systems that enable non-expert users to easily design personalized 3D structures are quickly gaining interest in the computer graphics community. Specifically, for the design of mechanical toys. Zhu et al. [2012] automatically generate an assembly of parametric gears, cranks, and pulleys to drive rigid components of mechanical toys based on user-defined motions. Similarly, Coros et al. [2013] develop a framework for animated mechanical characters creation where the user can iteratively create an animation by sketching motion curves. Ceylan et al. [2013] developed a fully automatic approach to drive a mechanical automaton and follow an input motion capture sequence. Thomaszewski et al. [2014] present a design system for linkage-based characters that provides different topology options to the user and then performs continuous optimization to improve the design quality of the user selected topology. A similar system was presented by Bächer et al. [2015], but allowing for interactive design with immediate feedback. Other work by Gauge et al. [2014] present a computational design system that allows casual users to interactively create tensegrity characters, in which they use elementary types of tensegrities as building blocks to create complex figures that approximate the shape of digital characters. Megaro et al. [2014] propose interactive design systems for rapid creation of planar mechanical characters, and in other work they build upon these ideas for 3D-printable robotic creatures [Megaro et al. 2015].

These new tools make it possible for novice users to design complicated mechanical structures with the assistance of computation. The goals of our method are similar. We focus on the problem of designing cable-driven articulated mechanisms. As is common to most of these design systems, our system allows users to choose an initial configuration, set high-level functional goals, and automate the majority of design processes that require expert knowledge.

2.2 Robotics

Our work is closely connected to the field of robotics, especially the modeling and design of articulated robots. Recently, a variety of approaches to articulated robot design and actuation have been explored. Bandara et al. [2014] propose a modified underactuated cross-bar mechanism, which has the self-adaptation ability for grasping different geometries, while Mishima and Ozawa [2014] describe methods for series gear chain mechanisms that produce connected motions and adaptive curling. Xu et al. [2012] and Kumar et al. [2013] rely on pneumatic systems for actuation and solve an air dynamic model empirically for controlling the pneumatic system. Odhner et al. [2014] design a versatile cable-driven robot gripper with underactuated fingers thanks to compliant joints. Higashimori et al. [2005] design a cable driven gripper for the dynamic task of catching a ball, and consider in their design the mass of links, pulley positions, joint stiffness, and frictional energy dissipation to achieve the task. In general, flexible robotics is a thriving field in robotics with a large and varied research going into design and analysis (De Luca and Book [2016]; Dwivedy and Eberhard [2006]).

Our focus on task specific design of articulated mechanisms is related to the problem of designing fingers that interact with the objects for touching, grasping, and manipulation. Lotti and Vassura [2002] propose a simplified solution for articulated finger design, where finger structures are made of rigid links connected by flexural hinges. They argue that the stiffness of the hinges can be exploited in order to simplify the actuation system. Similarly, we only consider linkages with stiff hinge joints. We assume that the stiffness of different hinge joints can vary by using different materials for rotational springs. Of course, optimal design of mechanisms and manipulators is an important problem that has received much attention over the years. Paredis and Khosla [1991] design manipulators based on kinematic task specifications. Ceccarelli and Lanni [2004] design manipulators with a multi-objective optimization respecting workspace limits. Yang and Chen [2000] optimize modular robots based on specified tasks, such as path following or obstacle avoidance. Yun and Li [2011] optimize the design of parallel mechanism with compliant hinges for micromanipulation. Ha et al. [2016] optimize the limbs of legged robots for tasks such as jumping, walking, and climbing. Also of note and related to cable driven mechanisms, Geijtenbeek et al. [2013] optimize muscle attachments for bi-pedal creatures.

Inverse kinematics is widely used in computer graphics, but is also critical in robotics for the control robot manipulators. Komura et al. [2003] propose an inverse kinematics method that determines the motion of redundant joints by a weight matrix derived from optimizing for minimal bending energy of the joints. In our framework, the initial cable-routing and actuation scheme of the articulated mechanism is provided by the user. That means the articulated structure we are designing could also be under-actuated and contain redundant joints. For this reason, and similar to most inverse kinematics methods, we use a joint stiffness matrix in our design, thus resolving the problem of underactuation.

2.3 Cable-driven mechanisms

We focus on cable-driven mechanisms due to the advantage of cable-driven systems being able to be controlled by simple actuators (for instance, if we were to create a physical prototype of our results, we could use inexpensive servos). Cheng et al. [2012] present a novel robotic manipulator design that uses jamming of granular media to achieve local stiffness, and tension cables to control the shape and position of the robot. Mao and Agrawal [2012] present a cable-driven upper arm exoskeleton for neural rehabilitation, where the cable attachment points are optimized to improve the workspace of the exoskeleton to be similar to the workspace of a human arm. While these studies focus on the development and analysis of individual designs with specific purposes, our system is a generic design tool through which users can create customized cable-driven articulated mechanisms. Note that while we assume cable mass to be negligible and do not take it into consideration in our analysis, Kozak et al. [2006] address the effect of cable mass in the static analysis of cable-driven robotic manipulators.

One challenge in modeling multi-link cable-driven manipulators is the large number of possible cable-routing schemes. Lau et al. [2013] introduce a generalized model for a multi-link cable-driven manipulator that allows for arbitrary cable routing by using a cable-routing matrix. Our system allows the user to provide an initial cable-routing scheme, which is then optimized in the subsequent continuous-optimization step. However, finding an optimal topology is something we leave for future investigation.

3 OVERVIEW

Our framework enables casual users to design articulated mechanisms that perform push and pick-and-place tasks. There are two main steps in our design process.

The user starts by providing an initial configuration of the articulated mechanism, such as the number of links and joints in the structure and the cable routing scheme, and indicates the tasks they want to perform. To give the user more control over the final design, our system also allows them to decide which part of the morphology should stay fixed in the optimization step.

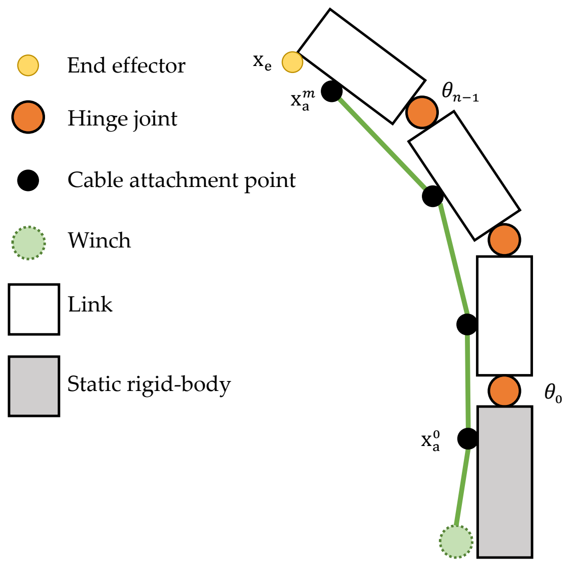


Figure 2: Our cable driven articulated mechanism consists of a set of links connected by rotary joints. We assume the root link is static in the world and a cable is routed through eyelets on a set of joints before being attached at a distal link (not necessarily the end link). An end effector is defined as the point that must interact with the environment to perform a specified task.

With the initial topology defined, our method then automatically optimizes the morphology of the articulated mechanisms using a stochastic algorithm to better perform the desired tasks. It is worth noting that each iteration of the optimization process is fast enough that the best solution can be visualized at interactive framerates, as shown in the supplementary video. This allows the user to quickly evaluate the design, make adjustments to parameters, and restart the optimization if they so desire.

3.1 The design problem

Before we explain our framework in more detail, we will briefly describe the computational model and representations we use in our system.

Our system focuses on cable-driven articulated mechanisms, as illustrated in Figure 2, where a series of rigid bodies, or links, are connected together through hinge joints. We assume that the hinge joints will contain rotational springs of various stiffness values. Without external force, the joints will stay at the rest posture.

We use a cable mechanisms to actuate our articulated mechanism designs. A control cable is modeled as a series of cable attachment points, each of them attached to a link in the structure or a rigid body in the workspace. To increase the flexibility in optimizing the cable routing scheme, we use cylindrical coordinates (ρ, φ, h) to parametrize each cable attachment point, as illustrated in Figure 3, where ρ, φ and h are the radial distance from the attached rigid body to the point, the attachment angle, and the attachment height respectively. Using cylindrical coordinates makes it simple

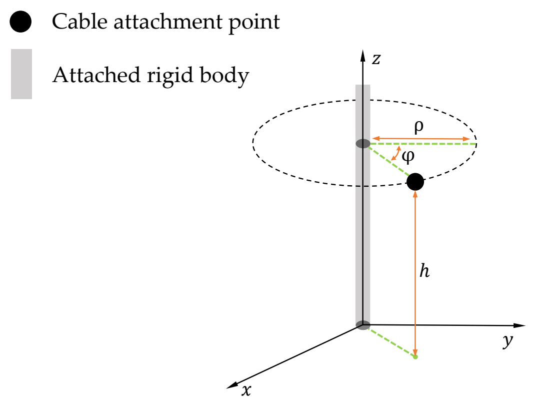


Figure 3: The cable attachment point is either a terminal connection or an eyelet through which the cable passes. The point has three parameters: the distance along the link *h*, the radial distance from the centerline of the link *ρ*, and an angle of attachment *φ* which is defined relative to the link's local reference frame.

for our system to optimize the attachment points while keeping the attachment angle or the radial distance constant, or vice versa.

We specify a task for the articulated mechanism by a position in the workspace, the direction in which a force needs to be applied, and the magnitude of the force required. Our tasks are basic physical actions, but we also sometimes refer to human-level functions, such as cutting a wire or pushing a button, even though our system is not aware of the goal or the context.

4 INVERSE KINEMATICS

Inverse kinematics is one of the most common ways to control the movement of a rigid multi-body. In this section we review the methods. Furthermore, we propose a novel approach for fast IK using cable-driven mechanisms.

4.1 Basic IK

Let us specify the posture of the articulated mechanism by

$$\boldsymbol{\theta} = (\theta_0\,,\ \theta_1\,,\ \ldots,\ \theta_{n-1})\ ,$$

where n is the degree of freedom (DOF) of the mechanism. Note that in our design, the DOF is the same as the number of joints in the structure, and θ_i is the angular displacement of the ith joint.

We use $\mathbf{x}_\mathrm{e}$ to represent the position of the end effector, which can be viewed as a function of the posture, $\boldsymbol{\theta}$, and topology of the articulated mechanism. We usually call it the forward kinematics function. For now, let us assume the topology of the mechanism is fixed.

For small displacements, the relationship between movement at the end effector $\Delta\mathbf{x}_\mathrm{e}$ and changes in the joints angles $\Delta\boldsymbol{\theta}$ can be approximated [Buss 2004; Komura et al. 2003]:

$$\Delta\mathbf{x}_\mathrm{e} \approx \mathbf{J}_e \Delta\boldsymbol{\theta}\ , \tag{1}$$

where $\mathbf{J}_e = \frac{d\mathbf{x}_\mathrm{e}}{d\boldsymbol{\theta}}$. We then find a solution for $\Delta\boldsymbol{\theta}$ using the pseudo-inverse of $\mathbf{J}$, such that

$$\Delta\boldsymbol{\theta} = \mathbf{J}_e^\dagger \Delta\mathbf{x}_\mathrm{e} \tag{2}$$

and $\mathbf{J}^\dagger = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}$. The goal of these incremental updates to the joint angles is to find a posture for the mechanism where the end effector matches the target position, such that $\mathbf{x}_\mathrm{t} = \mathbf{x}_\mathrm{e}(\boldsymbol{\theta})$. The IK problem is solved by calculating $\Delta\boldsymbol{\theta}$ iteratively,

$$\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}^{(k+1)}\ , \tag{3}$$

where the Jacobian matrix and end effector position is re-calculated at each iteration k. These updates are performed until convergence. We use the damped least squares method [Buss 2004] to handle singularities that can arise when using the pseudo-inverse method.

4.2 Cable-driven IK

The basic IK algorithm presented in Section 4.1 only shows how changes in the posture $\boldsymbol{\theta}$ affect $\mathbf{x}_\mathrm{e}$. Given that our goal is the design of mechanisms that are actuated by cables, we would prefer to determine how changes in the length of the control cables affect the position of the end effector in the workspace.

We use l_c to represent the length of a control cable. Assuming the cable is under tension, then l_c can be expressed as the sum of distances between adjacent attachment points:

$$l_c = \sum_{i=0} ||\mathbf{x}_\mathrm{a}^i - \mathbf{x}_\mathrm{a}^{i+1}||\ , \tag{4}$$

where $\mathbf{x}_\mathrm{a}^i$ is the position of the ith attachment point of the cable. It is trivial to determine $\mathbf{x}_\mathrm{a}$ for attachment points that are fixed to a static rigid-body. For attachment points that move with a mechanism link, the position is found by a forward kinematics calculation similar to how we calculate the end effector position $\mathbf{x}_\mathrm{e}$.

Similar to Eq. 1, the relationship between l_c, $\mathbf{x}_\mathrm{a}$ and $\boldsymbol{\theta}$ can be written in the following form:

$$\Delta l_c = \frac{dl_c}{d\mathbf{x}_\mathrm{a}} \Delta\mathbf{x}_\mathrm{a}\ , \tag{5}$$

$$\Delta\mathbf{x}_\mathrm{a} = \frac{d\mathbf{x}_\mathrm{a}}{d\boldsymbol{\theta}} \Delta\boldsymbol{\theta}\ . \tag{6}$$

Combining Eq. 5 and Eq. 6, we get

$$\Delta l_c = \mathbf{J}_c \Delta\boldsymbol{\theta}\ , \tag{7}$$

where $\mathbf{J}_c = \frac{dl_c}{d\boldsymbol{\theta}}$ is the Jacobian matrix of l_c with respect to the posture $\boldsymbol{\theta}$. It can be easily verified through the chain rule that $\frac{dl_c}{d\boldsymbol{\theta}} = \frac{dl_c}{d\mathbf{x}_\mathrm{a}} \frac{d\mathbf{x}_\mathrm{a}}{d\boldsymbol{\theta}}$.

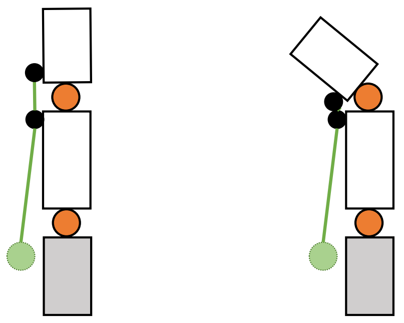
Similar to Eq. 2, we can find a solution for $\Delta\boldsymbol{\theta}$ by

$$\Delta\boldsymbol{\theta} = \mathbf{J}_c^\dagger \Delta l_c\ . \tag{8}$$

Combining Eq. 2 and Eq. 8, we get

$$\Delta l_c = \mathbf{J}_c \Delta\boldsymbol{\theta} = \mathbf{J}_c \mathbf{J}_e^\dagger \Delta\mathbf{x}_\mathrm{e}\ . \tag{9}$$

Eq. 8 and Eq. 9 give the relationship between the incrementation of the position of end effector, cable length and joint angles. We can use them in calculating update values to iteratively solve the inverse kinematics problem for an articulated mechanism with a cable-driven mechanism.



(a) Rest posture **(b) Posture at joint limit**

Figure 4: Joint limits of our simple cable-driven mechanism are determined by link geometry and attachment points.

For mechanisms controlled by multiple cables, we can very easily scale up by expanding the length of a single cable l_c to a vector for all cables $\mathbf{l}_c$ in Eq. 5 to 9.

Note that attaching a cable to the mechanism introduces new limits for the joints. As shown by the simple example in Figure 4, the joints cannot bend any further when adjacent attachment points are at their closest point to each other since we are controlling the mechanism joints by pulling control cables. We need to take the new joint limits into consideration to guarantee the stability of our new IK approach.

4.3 Weighted IK

Our framework allows the user to set the initial topology and cable-routing scheme for the articulated mechanism, which indicates that the structure to be designed can be under-actuated and contain redundant joints. Komura et al. [2003] use a weight matrix to solve this redundancy problem, and similarly, we use a joint spring with tunable stiffness in our mechanism designs.

Let us define an $n \times n$ diagonal weight matrix $\mathbf{W}$, where the entry $\mathbf{W}_{i,i}$ corresponds to the stiffness of the ith joint. Recall that we assume joints will contain springs. Let us also define that the joint angles at the rest configuration to be 0, then $\boldsymbol{\theta}$ can also represent the angular displacement of the joints. The total spring bending energy can thus be calculated by

$$E = \frac{1}{2}\boldsymbol{\theta}^T \mathbf{W} \boldsymbol{\theta} \ . \tag{10}$$

If we define $\hat{\mathbf{J}}^\dagger$ as the weighted pseudo-inverse matrix of $\mathbf{J}$,

$$\hat{\mathbf{J}}^\dagger = \mathbf{W}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T)^{-1} \ , \tag{11}$$

then we can update Eq. 8 using Eq. 11 to produce a new update for our IK solver,

$$\Delta\boldsymbol{\theta} = \hat{\mathbf{J}}_c^\dagger \Delta l_c \ . \tag{12}$$

Here, Eq. 12 gives a solution that optimizes the spring bending energy in the joints [Komura et al. 2003].

Our inverse kinematics solver using cable-driven mechanisms is summarized in Algorithm 1, and takes as a parameter a *design-state* that we define in the next section. Note that in cable-driven systems, multiple joints will typically bend in unison as they are controlled

```
Input: Initial rest posture $\boldsymbol{\theta}_{\text{in}}$
Parameters: Mechanism design-state vector S
            Target position $\mathbf{x}_{\text{t}}$
Output: Solved posture $\boldsymbol{\theta}_{\text{out}}$
while $||\mathbf{x}_t - \mathbf{x}_e(\boldsymbol{\theta})^{(k)}|| > \epsilon$ do
    Calculate $\mathbf{J}_e = \frac{d\mathbf{x}_{\text{e}}^{(k)}}{d\boldsymbol{\theta}^{(k)}}$ and $\hat{\mathbf{J}}_e^\dagger$ by Eq.11
    Calculate $\mathbf{J}_c = \frac{dl_c^{(k)}}{d\mathbf{x}_{\text{a}}^{(k)}} \frac{d\mathbf{x}_{\text{a}}^{(k)}}{d\boldsymbol{\theta}^{(k)}}$ and $\hat{\mathbf{J}}_c^\dagger$ by Eq. 11
    Calculate $\Delta l_c^{(k+1)} = \mathbf{J}_c \hat{\mathbf{J}}_e^\dagger \Delta\mathbf{x}_{\text{e}}^{(k)}$
    Calculate $\Delta\boldsymbol{\theta}^{(k+1)} = \hat{\mathbf{J}}_c^\dagger \Delta l_c^{(k)}$
    $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}^{(k+1)}$
end
```

Algorithm 1: Iterative cable driven IK solver

by the same cable. Calculating $\Delta\boldsymbol{\theta}$ from Δl_c makes sure that joint angles are directly affected by changes of cable length.

5 MORPHOLOGY OPTIMIZATION

The iterative algorithm used to optimize the mechanism morphology is summarized in Algorithm 2. A set of task specifications, $\mathbf{T}$, contains the position and required force (direction and magnitude) for all tasks. Once the algorithm terminates, the optimal design parameters are available in a design-state vector S. These parameters include the root position of the mechanism (3 values), link lengths (1 value per link), stiffness of joints (1 value per joint), hinge axes (1 angle per joint given that we keep the axis orthogonal to the link direction), and attachment positions for cable routing (3 coordinates for each eyelet, and 2 of these for each joint that a cable crosses). In this work, it is for the sake of simplicity that we restrict each rotary joint to be orthogonal to the link direction and that we fix the rest pose of the torsion springs to be zero for each joint. Nevertheless, we note that adding these parameters to the design-state would not be onerous and could lead to more efficient designs.

The CMA-ES algorithm [Hansen and Ostermeier 1996] is used to solve for the final design by minimizing

$$\text{H} = \sum_{t \in \mathbf{T}} H_{\text{task}}^t + H_{\text{reg}}. \tag{13}$$

The function in Eq. 13 is composed of the sum of individual task terms, H_{task}^t, and a regularization term, H_{reg}. A task is evaluated by the ability of the mechanism's end effector to reach a point in 3D space and apply a particular force. The task-based function is therefore the weighted combination of several objectives that evaluate these characteristics, or

$$H_{\text{task}}^t = \sum_{j=1}^{5} w_j H_j^t. \tag{14}$$

Here, w_j are positive scaling factors and H_j is an objective function that evaluates a specific characteristic of the mechanism design. The remainder of this section is dedicated to presenting details of the five objective functions that make-up the task function.

```
Input: Initial design-state vector $S_{in}$
Parameters: Task configuration vector T
Output: Optimized state vector $S_{out}$
while CMA-ES termination criteria not met do
    foreach $t \in T$ do
        Solve IK problem for $\theta^t$ to reach target position $x_t^t$
        Calculate $H_{task}^t$ by Eq. 14
    end
    Calculate $H = \sum_{t \in T} H_{task}^t + H_{reg}$
    CMA-ES algorithm updates design parameters
end
```

Algorithm 2: Morphology optimization

5.1 Reachability

The most basic requirement for a mechanism to perform a task is reaching the target position in the workspace. Therefore, the first term in the objective function evaluate's the ability of a design to reach the target position. We calculate a posture that gets the end effector of the mechanism closest to the task position using the IK solver we described above. We use the distance between the end effector position and the task location $\mathbf{x}_t$ as the reachability of the mechanism:

$$H_1 = ||\mathbf{x}_t - \mathbf{x}_e(\boldsymbol{\theta})|| \ . \tag{15}$$

5.2 Force control

To avoid the cases where the mechanism can reach the task position, but cannot push in the specified directions, we include a task compatibility index in the objective function.

Let $\vec{u}$ denote the unit vector in the direction we want the end effector to push, and γ be the force transmission ratio in the direction of $\vec{u}$. We know from work by Chiu [1988] that

$$\gamma = (\vec{u}^T \mathbf{J}_e \mathbf{J}_e^T \vec{u})^{-\frac{1}{2}} \ . \tag{16}$$

Specifically, Chiu proposes that the optimal direction for fine control of velocity is the direction in which the transmission ratio is at a minimum. Therefore, we use the task compatibility index

$$H_2 = \gamma^2 = (\vec{u}^T \mathbf{J}_e \mathbf{J}_e^T \vec{u})^{-1} \tag{17}$$

in our objective function to achieve maximum force control along the direction $\vec{u}$.

5.3 Force production

The first two terms of the objective function make sure the mechanism can reach the task position and control force in the direction of interest. However, in order to perform a task, the mechanism also has to be able to produce a large enough force while not violating the limits of the motors used as actuation.

The task objective involves applying a force in a particular direction at the end effector position. The joint torques required to generate a force at the end effector $\mathbf{f}_e$ may be computed by

$$\boldsymbol{\tau}_{end} = \mathbf{J}_e^T \mathbf{f}_e \ . \tag{18}$$

Of course, no torques are being generated at the joints, since these are un-actuated. But it is useful to consider in our analysis.

We assume joints contain springs, which generate a torque when a joint is rotated away from a rest configuration. Therefore, we must consider the torques generated by these torsional springs at the joints, such that

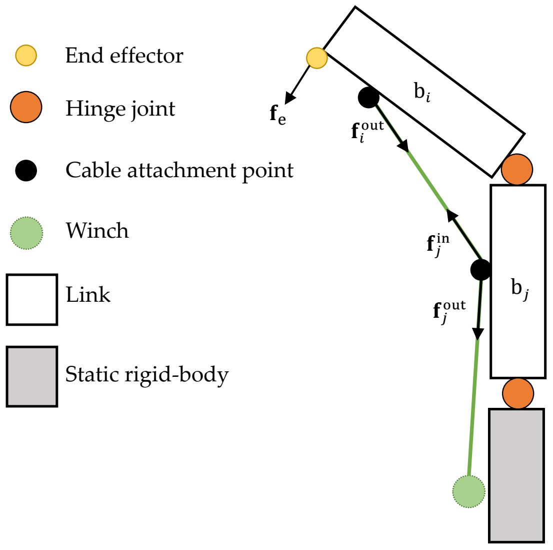


Figure 5: Cable forces, joint torques, and end effector forces are computed using pose and morphological parameters.

$$\boldsymbol{\tau}_{spring} = \mathbf{W}\Delta\boldsymbol{\theta} \ . \tag{19}$$

Recall that $\mathbf{W}$ is the diagonal joint stiffness matrix and $\Delta\boldsymbol{\theta}$ is a vector of joint displacements. This does not take into twisting effects that could occur in our 3D printed compliant joints from the various cables that cross a joint, though we likewise did not observe any twisting in our printed examples.

Since individual joints are un-actuated, the internal torques generated by the system occur via a cable which is routed through attachment points on the mechanism. We assume that frictionless eyelets are used at the cable attachment points, and so focus on the effects of tensile forces. In Figure 5, each attachment site has an incoming and an outgoing force, acting in the direction as drawn. The total effective force acting at each site is the sum of these forces, $\mathbf{f}_i^{in} + \mathbf{f}_i^{out}$, and the torques produced at the ith attachment site is

$$\tau_i = \mathbf{J}_i^T \mathbf{f}_i^{in} + \mathbf{J}_i^T \mathbf{f}_i^{out} \ . \tag{20}$$

Putting Eq. 18, Eq. 19 and Eq. 20 all together, we get the static equilibrium of all generalized forces (torques) acting in the system

$$\mathbf{J}_e^T \mathbf{f}_e = \sum_{i=1} (\mathbf{J}_i^T \mathbf{f}_i^{in} + \mathbf{J}_i^T \mathbf{f}_i^{out}) - \mathbf{W}\Delta\boldsymbol{\theta} \ , \tag{21}$$

where $\mathbf{f}_i^{in}$ and $\mathbf{f}_i^{out}$ represent the incoming and outgoing tensile forces acting on the ith eyelet, respectively. The linear Jacobian $\mathbf{J}_i$ is evaluated at each attachment site. Note that if the cable is rigidly attached to the mechanism (i.e., fixed at the end of the cable), then we can assume that $\mathbf{f}_i^{in} = 0$ in this special case.

Since we are ignoring friction, the tension is even throughout the cable and each input and output force will have a magnitude equal to this tension, and the only thing we need to keep track of in this case is the unit vectors $\boldsymbol{u}_i^{\text{in}}$ and $\boldsymbol{u}_i^{\text{out}}$ that give the direction of each force. We rewrite Eq. 21 as

$$\mathbf{J}_e^T \mathbf{f}_\text{e} = \alpha \hat{\boldsymbol{\tau}} - \mathbf{W}\Delta\boldsymbol{\theta} \ , \tag{22}$$

where α is the tension in the cable, which should be the same as the motor force at the winch, and $\hat{\boldsymbol{\tau}}$ represents the summation term $\sum_{i=1}(\mathbf{J}_i^T \boldsymbol{u}_i^{\text{in}} + \mathbf{J}_i^T \boldsymbol{u}_i^{\text{out}})$. Substituting and multiplying both sides by the pseudo inverse gives

$$\mathbf{f}_\text{e} = \mathbf{J}_e^{\dagger T}(\alpha \hat{\boldsymbol{\tau}} - \mathbf{W}\Delta\boldsymbol{\theta}) \ . \tag{23}$$

We formulate the force as an objective in the morphology optimization. Recall that $\mathbf{f}_\text{e}$ is a virtual quantity, and that $\mathbf{J}_e^T$ is a linear map relating output forces to input torques. Naively, we could assume that if $\mathbf{f}_\text{e}$ is the force specified by the task objective it could be generated simply by computing the α that satisfies the equilibrium equation. But the desired $\mathbf{f}_\text{e}$ may have components that are in the row null space of $\mathbf{J}_e$, and so we need to account for this in the objective function.

We begin by solving the least squares problem for the small, overdetermined system

$$\text{argmin}_\alpha ||\mathbf{f}_\text{e} - \mathbf{J}_e^{\dagger T}(\alpha \hat{\boldsymbol{\tau}} - \mathbf{W}\Delta\boldsymbol{\theta})|| \ , \tag{24}$$

which gives α at the minimum. Also, consider that if $\mathbf{f}_\text{e}$ is completely in the null space of the Jacobian, the optimization should return zero. Now we clamp α to the limits of the motor,

$$\alpha^* = \max(\alpha_\text{min}, \min(\alpha, \alpha_\text{max})) \ , \tag{25}$$

where α_min is set to 0 to guarantee non-negative motor force. Finally, we compute the residual of the force generated by $\alpha^*$and use this in the morphology optimization, such that the penalty equation can be written as

$$H_3 = ||\mathbf{f}_\text{e} - \mathbf{J}_e^{\dagger T}(\alpha^* \hat{\boldsymbol{\tau}} - \mathbf{W}\Delta\boldsymbol{\theta})|| \ . \tag{26}$$

This will give the error of the force we want versus the force that will be generated by pulling the cable with a motor force (tension) of α^*.

5.4 Limit avoidance

Since we only use static kinematics analysis to evaluate the ability to perform tasks, the optimizer sometimes gives a design that is able to reach the task position, but unable to move any further in the direction of interest because the joints in the mechanism are already at their limits. To penalize these designs, we add a limit avoidance objective term that evaluates the magnitude of joint limit violation when the mechanism tries to push further in the direction of interest after the end effector has already reached the task position.

Recall Eq. 2 gives the relationship between changes in the end effector position and small changes in the joint angles. We can thus approximately calculate the joint angles required for the end effector to push in the direction $\vec{u}$ by the distance of d through,

$$\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} + d\,(\mathbf{J}_e^\dagger \vec{u}) \ . \tag{27}$$

The joint angles are also clamped so that joint limits are not exceeded:

$$\tilde{\boldsymbol{\theta}} = \max(\boldsymbol{\theta}_\text{min}, \min(\boldsymbol{\theta}_\text{max}, \bar{\boldsymbol{\theta}})) \ , \tag{28}$$

Mechanism	w_1	w_2	w_3	w_4	w_5
Default	100	10	10	20	30
Light switcher	100	10	10	20	30
Bomb wire cutter	100	10	0	0	30
Whack-a-mole	100	10	10	20	100
Pick-and-place	100	0	0	0	30
Number-pad dialer	1000	10	10	20	45
Button presser	700	10	10	20	30

Table 1: Weights used in optimizing the mechanisms.

where $\boldsymbol{\theta}_\text{min}$ and $\boldsymbol{\theta}_\text{max}$ are the lower and upper bounds of the joint angles, respectively. The limit avoidance term is simply the magnitude of angle limits violation, or

$$H_4 = ||\tilde{\boldsymbol{\theta}} - \bar{\boldsymbol{\theta}}|| \ . \tag{29}$$

5.5 Collision penalization

To ensure that the resulting mechanism designs do not have self collision between the cables and links, or collide with other objects in the workspace while performing tasks, we penalize solutions that would result in collision. We only create a collision set C for the pose computed for the reachability objective. We note it would be an interesting although more costly to ensure that a collision free path exists from the rest pose. A penalty is computed as the sum of interpenetration distances between all pairs of points in the set, such that

$$H_5 = \sum_{c \in \text{C}} ||\mathbf{p}_a^c - \mathbf{p}_b^c|| \ . \tag{30}$$

Here, $\mathbf{p}_a^c$ and $\mathbf{p}_b^c$ are the positions of points in collision c. We use the Bullet physics engine [Coumans 2015] for collision detection. We use collision proxies that are slightly larger than the geometry such that potential collisions are identified before they happen, and can be appropriately penalized. We use interpenetration depth rather than a proximity query for the simplicity of using the tools and libraries at hand.

5.6 Regularization

Finally, we add a regularization term for input values to make sure the optimized parameters stay in a reasonable and realistic range. We will use H_reg to denote the regularization term. For scalar input values, including link length and joint stiffness, we use an ℓ^2 norm. For non-scalar parameters, regularization is handled slightly differently.

The regularization for root position is simply the squared distance between the current root position and the initial one. This helps to ensure that the mechanism does not get placed too far away from the user-specified position. The regularization for a cable attachment is the squared radial distance between the position and central link, ρ^2. Finally, we do not include any regularization for the rotation axis as we do not wish to encourage or penalize any given axis direction.

5.7 Weighted objectives

Each term in the optimization has an important effect on the solution, though in some cases we observe that certain terms could be omitted. Without the reachability term, the end effector will not reach the target at all. Without the collision penalty term we observe solutions can involve links penetrating with the scene as well as self collision. The force control, force production, and limit avoidance terms have a more subtle effect. Without the force control term, the solution can have the mechanism reaching a target position in a way that is impossible to produce a force in the desired direction. The force production term is important for ensuring that the desired force at the end effector is possible given the joint torques and motor forces. If the term is omitted, then we observe random forces when we enable their visualization in the editor (note that we did not verify force production results for any printed examples). Finally, we include the limit avoidance term because it simplifies the cable driven IK in that it ensures that equilibrium configurations will be valid, and likewise, omitting the term can result in solutions with large bends where there is no possibility of producing the pose by pulling cables.

We use the following default values for the scaling factors: $w_1 = 100$ (reachability), $w_2 = 10$ (force production), $w_3 = 10$ (force transmission), $w_4 = 20$ (limit avoidance), and $w_5 = 30$ (collision penalty). However, different task specification require adjustment to these weights. With some experimentation, we did not find it too difficult to find suitable weights, though we also recognize this need for choosing weights as a limitation of our approach. Table 1 gives the weight values used to produce the results for all examples shown in this paper.

6 OPTIMIZATION RESULTS

We use our system to design several articulated mechanism that perform a diverse set of tasks, as shown in Figure 6. The optimization algorithm is implemented in C++ and runs on a single thread. All experiments were performed on a Windows 10 PC with Intel i5 processor and 8 GB of RAM. Our prototype application for configuring the scene and visualizing results is built using the Unity game engine.

Table 2 presents an overview of some basic information about the complexity of our designed mechanisms, while Table 3 provides the timing information on designing the mechanisms. Please also see the accompanying video for animations of the optimization process and the final mechanisms. Note that in all examples, the cable-driven IK provides the lengths of the cables to meet a given target, and much like a keyframe animation we simply drive the cables at the appropriate constant velocity from the rest configuration to the target, and then back.

It is worth noting that all of our designs contain only 3 joints or less, and yet they are capable of performing various tasks. In our experiment with designing mechanisms of higher complexity to perform more complex tasks, our system would find solutions that do not use all the joints.

From Table 3 we can see that it takes on the order of minutes for our system to design a mechanism. The amount of time it takes for our system to generate a design is directly related to the complexity of the mechanism and number of tasks.

Mechanism	Joints	Cables	Tasks	Dim
Light switcher	2	1	2	15
Bomb wire cutter	3	2	2	27
Whack-a-mole player	2	2	4	18
Pick-and-place	3	2	4	27
Number-pad dialer	3	3	9	30
Button presser	3	3	5	30

Table 2: Complexity of designed mechanisms and tasks. At right, we list the dimension of the design space (root position, link lengths, eyelet positions, joint orientation, and joint stiffness, as described in Section 5).

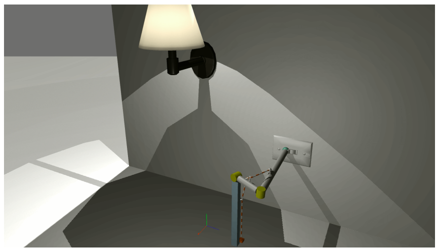
Mechanism	No. of iterations	Avg. time per iteration (ms)	Total time (s)
Light switcher	528	26.67	14.08
Bomb wire cutter	620	70.27	43.57
Whack-a-mole player	456	136.37	62.18
Pick-and-place	733	349.85	256.44
Number-pad dialer	615	474.89	292.06
Button presser	1679	113.94	191.3

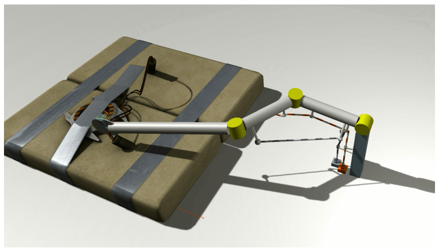
Table 3: Timing information for morphology optimization.

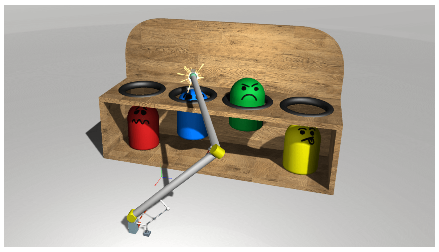
Light switcher. The light switcher in Figure 6a is an example of a mechanism controlled by a single cable that only has two modes. This type of mechanism could be useful in workspaces with on and off switches. This is an under-actuated example where one cable controls both hinge joints. The posture is determined by the stiffness of the joint springs. Note that the two tasks we set for the mechanism requires applying force in the exact opposite direction, and that can only be solved by pulling and releasing the cable in this one cable example.

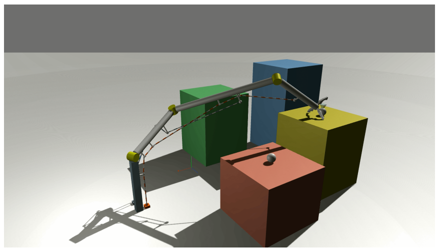
Bomb wire cutter. The bomb wire cutter design (Figure 6b) demonstrates our system's potential in designing mechanisms performing highly customized, and possibly very dangerous, tasks. In this case, we place a cutting device (a circular saw) at the end effector location to perform the cutting task. We exploited the fact that the mechanism itself does not need to push the target with force, and hence reduce the weight of the force production objective function in the optimization.

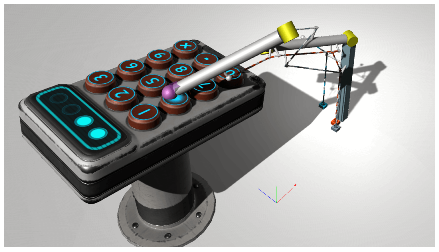
Whack-a-mole. The Whack-a-mole player mechanism in Figure 6c shows that we can achieve slightly more complicated tasks with a very simple 2-link mechanism controlled by 2 cables. The Whack-a-mole player is a typical example of a 2 link mechanism reaching for multiple targets that line up. Our system tends to find solutions where the two rotation axes of the joints are nearly perpendicular; one cable will control the root joint to bend sideways in order to reach multiple target, while the other joint bends so that the end effector could push, or hit in this particular case, the target in the desired direction. Figure 7 shows the convergence plots for optimizing the Whack-a-mole player. We only include plots for this


(a) Light switcher


(b) Bomb wire cutter


(c) Whack-a-mole player


(d) Pick-and-place


(e) Number-pad dialer

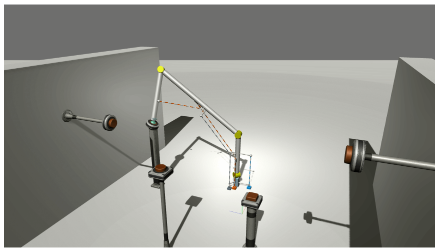

(f) Button presser

Figure 6: Articulated mechanism designed using our framework

one case, but the optimizations for other cases all converged in similar fashions.

Pick-and-place. The pick-and-place gripper (Figure 6d) is an example to show we can do more than just pushing tasks with our articulated mechanisms. Similar to the wire cutter, we reduce the weight of all force related terms in the optimization and focus mostly on the reachability of the mechanism. We purposefully made the platforms to be of different heights and also not in alignment with each other to test our system's ability to reach more random points in the workspace.

Number pad dialer. Figure 6e demonstrates our system's ability to design mechanisms that can perform more nuanced tasks. The number pad dialer is 3-link mechanism controlled by 3 cables that can be used to dial buttons 1 to 9. It is the mechanism that takes the longest time for our system to design due to the amount of tasks specified. In our experiment, we also need to increase the weight on reachability during optimization when there are more tasks.

Button presser. Most of our designs only need to perform tasks that are more or less aligned in terms of position or force direction. We designed the button presser to show our system's ability to find a good design for a complex set of tasks that require placing the end effector at a set of unstructured positions and requiring forces in a variety of directions.

6.1 Discussion and limitations

Our system only analyses the final posture that gets the mechanism to the task location and evaluates its ability to perform this task. That results in cases where the mechanism would take a infeasible trajectory to the task position due to collisions with objects in the scene.

One limitation of our optimization is that the resulting cable attachment scheme does not prevent cables from crossing with each other. This could be avoided by adding additional penalization objective to the optimization in the future.

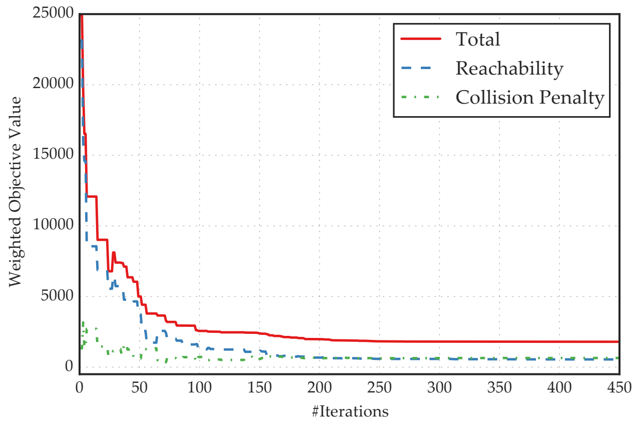
The core of our approach involves a continuous optimization problem, which does not make discrete changes to the morphology of the mechanism. This means that when the user sets too many tasks, but does not increase the DOF, our system will struggle to find a good solution. Thomaszewski et al. [2014] circumvent this problem by providing different topology options for the user to choose from. We would like to also create a system that can evaluate the complexity of user defined tasks and provide suggestions for initial mechanism configuration.

We sometimes observed that subsequent executions of the optimization did not always result in similar solutions, even though parameters remained fixed. This indicated that CMA-ES was periodically converging to a local minimum. Rather than fine tuning low-level parameters of the stochastic optimization, we prefer to spawn multiple instances of our prototype application and run the algorithm with different initial sample populations in parallel. The best solution among all processes is then chosen as the mechanism design. This is comparable to using a restart technique for finding a global minimum, and we found that this gave deterministic behavior when performing the morphology optimization.

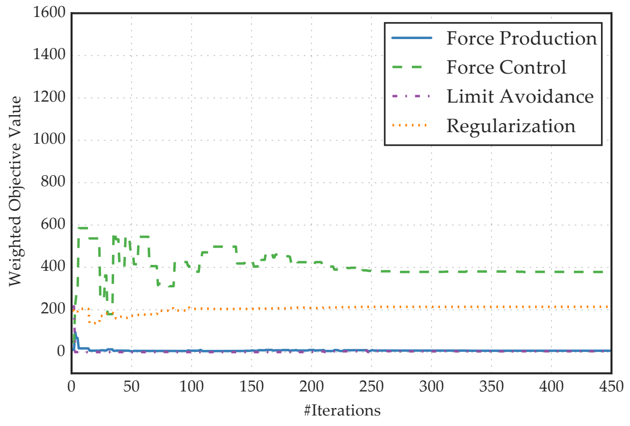
6.2 Dynamical simulation

We simulate several of the mechanisms shown in the previous section using the Vortex Dynamics [2017] physics engine, which supports cable and constrained multibody dynamics simulation.

Links are modeled as cylindrical bodies with collisions enabled. The stiff hinge joints are modeled as 6 DOF rigid constraints, which are then relaxed to allow angular motion about the axis of rotation for the joint. Relaxation parameters are assigned according to the stiffnesses found by our framework. The eyelets and terminal cable attachments are modeled using a specialized toolkit. Lengths of



(a) Convergence plots for the total objective value, the reachability term, and the collision penalty term.



(b) Convergence plots for the force production term, the force transmission term, the limit avoidance term, and the regularization term.

Figure 7: Convergence plots for optimizing the Whack-A-Mole player. The plots are presented in two figures for clarity. Note that the scales of y axes (weighted objective value) are different in the two figures.

individual cables are actuated by attaching one end of the cable to a motorized prismatic joint. Constraint force limits were used to model the torque limits of a typical low-cost servo. We then reach the targets by moving the actuated prismatic joints according to the cable lengths changes from our optimization.

This step of simulating the dynamics allows us to estimate how the mechanism performs under various physical conditions, and to evaluate the viability of a design before fabricating it. For instance, a simulation involving the Whack-a-mole example is shown in the accompanying video. We evaluate it by the ability of the dynamical mechanism to reach the target locations and produce sufficient force to move the virtual moles, which are attached to springs and resist movement. The mechanism could reach and "whack" all of the targets, despite additional motion due to inertial effects.

However, we observed that if the hinge joint limits are not enabled in the simulation, the mechanism could sometimes overshoot the rest position. This case is shown in the video. The mechanism ends up in a configuration where it does not behave according to the designed specification. This caused some concern, since joint limits are not enforced in our fabricated models. But we have not observed this behavior in the fabricated Whack-a-mole example. We attribute this to additional damping in the printed example that was not considered in our simulation.

7 PRINTING

Our goal is to design mechanisms that can be realized with inexpensive servos, cables, and 3D printed parts. To evaluate our results we describe below our process for taking an optimization and creating a 3D printed mechanism.

7.1 Compliant joints

We use a parametric CAD file with OpenSCAD to create geometry for printing. The parameters come directly from the optimization, and include the number of joints, the joint axes, the stiffness of joints, the length of beams, and the position of the eyelets.

Printing in the default and least expensive strong and flexible plastic material available from Shapeways, we target a one-shot printing process for the full mechanism, which only involves threading fishing line through the eyelets of the finished print. This can be accomplished by printing compliant rotary joints, where the compliance at the joints comes from the deformation of the geometry.

We considered two designs for the joints, with the first being the compliant rotary joint championed by Moon et al. [2002] as having the best overall properties. These include compactness, range of motion, axis drift, stress concentration, off-axis stiffness, while having acceptable compactness. While the compactness of the joint is problematic for cable routing, we also observed other problems with a collection of preliminary test prints with different thicknesses for different stiffness. Using beams with a thickness of 2 mm, the resulting joints were too stiff, and exhibited plastic deformation when bent by 90 degrees. The next best alternative in Table 2 of Moon et al. [2002] would seem to be the cross flexure, also known as a cross spring pivot.

While the optimization has real units and the tasks have real positions, we treated the results as uniformly scalable. Additionally, we note that we can change the stiffness of the joints as long as we preserve their ratios, as this is what will determine the static equilibrium solution when we pull on the cables. Because the joints have the smallest parts, they were used as a reference for scaling the rest of the mechanism. The joints were modeled to be just big enough to be printable (a bit bigger than the minimum printable size to ensure successful prints). We set the softest joint as having the thinnest parts, and then scaled the thickness of the beams of the cross-flex joints to account for the ratio of stiffnesses that came from the optimization. Note that joint stiffness can be computed from the material parameters [EOS 2017] and standard formulas for the classic double symmetric cross spring pivot [JPE 2017].

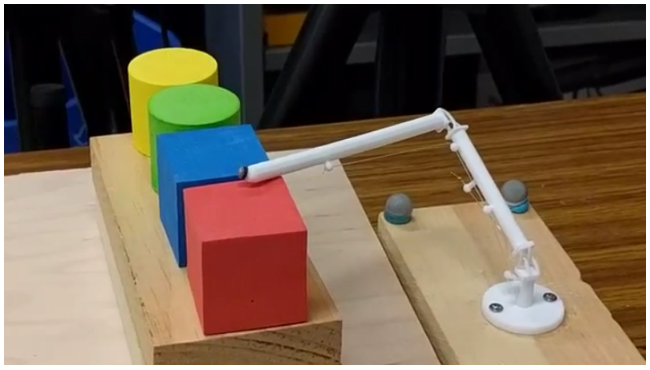


Figure 8: View of the Whack-a-mole mechanism being tested for accuracy. The x axis is along the row of colored blocks into the image, the y axis is up, and the z axis is to the right.

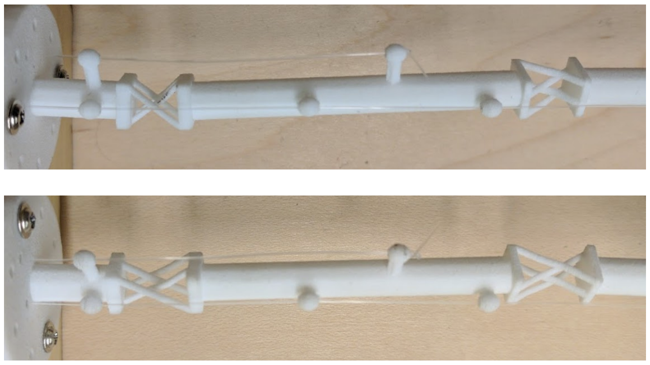


Figure 9: Close up view of two versions of the cross spring pivot with different heights to implement our compliant rotary joints.

7.2 Printing issues

Our printed results focus on the Whack-a-mole example (see Figure 8). We printed a number of variations with varying thickness, and with a varying overall joint height, expecting that the taller joints would be less vulnerable to plastic deformation and ultimately more flexible (see Figure 9) . We discovered that an important limitation of the printing process is that some prints had deformation artifacts, likely because they were warm when removed from printing powder and deformed under their own weight. As such, the rest pose of all the joints in our printed models are not what they should be (i.e., zero bend). However, the Whack-a-mole model has only two joints that are almost orthogonal, therefore the cables actuate the two joints largely independently. This leaves these small thin light models still usable, though in future prints, we plan to make larger prints to avoid any deformation from the print process.

7.3 Evaluation

Figure 8 shows the Whack-a-mole print mounted on a platform and connected to servos that we control with an Arduino (please also see the supplementary video). Because the beams of our model are light compared to the stiffness of the joints, we note that gravity has only a small effect on the static equilibriums (note likewise that gravity was not included in the optimization process). Figure 10 shows an evaluation of one of the prints repeating the tasks that it was designed to perform. The trajectory, recorded with motion capture, reveals important limitations in our final prints. That is, the print exhibited important plastic deformation over use. Furthermore, there are viscous effects that introduces quite a bit of variability in the resulting motion. The servo commands to produce the task positions were manually identified in this case. The x axis position of the motion shows the success for hitting the targets that were each 4 cm apart (i.e., the two middle tasks were well performed, while the mechanism had trouble reaching the two outer tasks).

8 CONCLUSIONS AND FUTURE WORK

We present a method for automating the design of cable-driven mechanisms. With our framework, we can produce designs that can achieve a collection of tasks while also having a small number of actuators as specified by the input topology. Our optimization is efficient thanks to a fast cable IK algorithm. Furthermore, the solutions respect physical requirements since the optimization considers physics-based objectives that ensure valid, collision-free solutions that produce the necessary forces and displacements at the end effector.

Fabrication-oriented design has gained quick interest in the graphics community, fueled by the rapid advances in 3D manufacturing technologies. While optimizing the morphology, we consider physical aspects of the mechanism by taking into account the force transmission, force production, and motor limits. The fabrication of one our designs, the Whack-a-mole example presented in Section 7, demonstrates that these kinematic and physical considerations combined result in mechanisms that are able to perform multiple tasks.

Our current fabricated mechanisms use a one shot printing process. As future work, we would like to automatically generate 3D printable geometries and assembly schemes from high level designs [Luo et al. 2012]. Another future research trajectory is adding supporting to our optimization framework for interaction between multiple linkages. With a single articulated mechanisms the set of tasks that can be performed is limited, and several mechanisms could perform more complicated tasks, such as pinching and grasping.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported by funding from NSERC, FRQNT, and the Tomlinson Scientist Awards.

REFERENCES

Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. 2015. LinkEdit: Interactive Linkage Editing Using Symbolic Kinematics. *ACM Trans. Graph.* 34, 4, Article 99 (July 2015), 8 pages.

DSV Bandara, RARC Gopura, G Kajanthan, M Brunthavan, and HIMM Abeynayake. 2014. An under-actuated mechanism for a robotic finger. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on.* IEEE, 407–412.

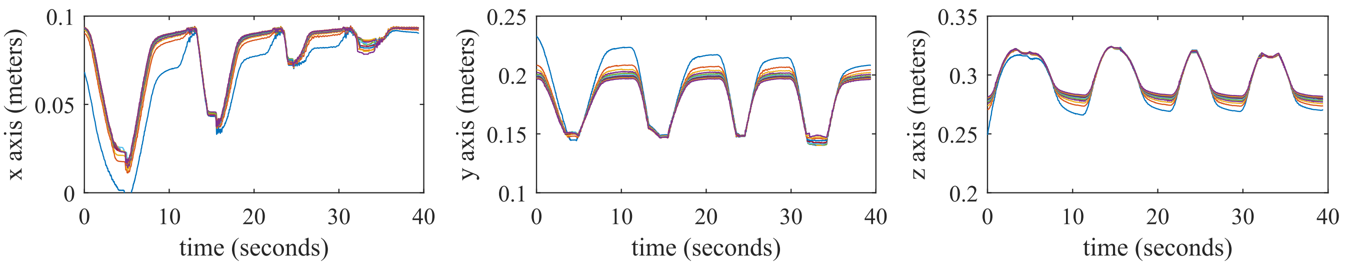


Figure 10: Plots showing end point accuracy over 11 cycles through the Whack-a-mole mechanism hitting 4 targets, in order, from red to yellow (see Figure 8). Viscosity and plastic deformation lead to different trajectories in each cycle, with servos commanded to positions where there is slack in the cables. Nevertheless, the task positions are repeated with reasonable consistency (at 5, 15, 25, and 35 seconds).

Samuel R Buss. 2004. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation* 17, 1-19 (2004), 16.

Marco Ceccarelli and Chiara Lanni. 2004. A multi-objective optimum design of general 3R manipulators for prescribed workspace limits. *Mechanism and Machine Theory* 39, 2 (2004), 119–132.

Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. 2013. Designing and Fabricating Mechanical Automata from Mocap Sequences. *ACM Trans. Graph.* 32, 6, Article 186 (Nov. 2013), 11 pages.

Nadia G Cheng, Maxim B Lobovsky, Steven J Keating, Adam M Setapen, Katy I Gero, Anette E Hosoi, and Karl D Iagnemma. 2012. Design and analysis of a robust, low-cost, highly articulated manipulator enabled by jamming of granular media. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on.* IEEE, 4328–4333.

Stephen L Chiu. 1988. Task compatibility of manipulator postures. *The International Journal of Robotics Research* 7, 5 (1988), 13–21.

Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM Trans. Graph.* 32, 4, Article 83 (July 2013), 12 pages.

Erwin Coumans. 2015. Bullet Physics Simulation. In *ACM SIGGRAPH 2015 Courses (SIGGRAPH '15).* ACM, New York, NY, USA, Article 7. http://doi.acm.org/10.1145/2776880.2792704

Alessandro De Luca and Wayne J Book. 2016. Robots with flexible elements. In *Springer Handbook of Robotics.* Springer, 243–282.

Santosha Kumar Dwivedy and Peter Eberhard. 2006. Dynamic analysis of flexible manipulators, a literature review. *Mechanism and Machine Theory* 41, 7 (2006), 749 – 777. DOI: https://doi.org/10.1016/j.mechmachtheory.2006.01.014

EOS. 2017. Material data sheet, PA 2200. (2017). https://www.shapeways.com/rrstatic/material_docs/mds-strongflex.pdf

Damien Gauge, Stelian Coros, Sandro Mani, and Bernhard Thomaszewski. 2014. Interactive Design of Modular Tensegrity Characters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '14).* Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 131–138.

Thomas Geijtenbeek, Michiel van de Panne, and A Frank van der Stappen. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 206.

Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. 2016. Task-based limb optimization for legged robots. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on.* IEEE, 2062–2068.

Nikolaus Hansen and Andreas Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on.* IEEE, 312–317.

Mitsuru Higashimori, Makoto Kaneko, Akio Namiki, and Masatoshi Ishikawa. 2005. Design of the 100G Capturing Robot Based on Dynamic Preshaping. *The International Journal of Robotics Research* 24, 9 (2005), 743–753. DOI: https://doi.org/10.1177/0278364905057058

JPE. 2017. Construction fundamentals, Cross Spring Pivot. (2017). http://www.janssenprecisionengineering.com/wp-content/uploads/Cross-spring-pivot.pdf

Taku Komura, Atsushi Kuroda, Shunsuke Kudoh, Tai Chiew Lan, and Yoshihisa Shinagawa. 2003. An inverse kinematics method for 3d figures with motion data. In *Computer Graphics International, 2003. Proceedings.* IEEE, 266–271.

Kris Kozak, Qian Zhou, and Jinsong Wang. 2006. Static analysis of cable-driven manipulators with non-negligible cable mass. *IEEE Transactions on Robotics* 22, 3 (2006), 425–433.

Vipin Kumar, Zhe Xu, and Emo Todorov. 2013. Fast, strong and compliant pneumatic actuation for dexterous tendon-driven hands. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on.* IEEE, 1512–1519.

Darwin Lau, Denny Oetomo, and Saman K Halgamuge. 2013. Generalized modeling of multilink cable-driven manipulators with arbitrary routing using the cable-routing matrix. *IEEE Transactions on Robotics* 29, 5 (2013), 1102–1113.

Fabrizio Lotti and Gabriele Vassura. 2002. A novel approach to mechanical design of articulated fingers for robotic hands. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on,* Vol. 2. IEEE, 1687–1692.

Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. 2012. Chopper: Partitioning Models into 3D-printable Parts. *ACM Trans. Graph.* 31, 6, Article 129 (Nov. 2012), 9 pages.

Ying Mao and Sunil Kumar Agrawal. 2012. Design of a cable-driven arm exoskeleton (CAREX) for neural rehabilitation. *IEEE Transactions on Robotics* 28, 4 (2012), 922–931.

Vittorio Megaro, Bernhard Thomaszewski, Damien Gauge, Eitan Grinspun, Stelian Coros, and Markus Gross. 2014. ChaCra: An Interactive Design System for Rapid Character Crafting. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '14).* Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 123–130. http://dl.acm.org/citation.cfm?id=2849517.2849538

Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive Design of 3D-printable Robotic Creatures. *ACM Trans. Graph.* 34, 6, Article 216 (Oct. 2015), 9 pages. http://doi.acm.org/10.1145/2816795.2818137

Yuuki Mishima and Ryuta Ozawa. 2014. Design of a robotic finger using series gear chain mechanisms. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on.* IEEE, 2898–2903.

Yong Mo Moon, Brian Patrick Trease, and Sridhar Kota. 2002. *Design of large-displacement compliant joints.* Vol. 5 A. 65–76.

Lael U. Odhner, Leif P. Jentoft, Mark R. Claffee, Nicholas Corson, Yaroslav Tenzer, Raymond R. Ma, Martin Buehler, Robert Kohout, Robert D. Howe, and Aaron M. Dollar. 2014. A compliant, underactuated hand for robust manipulation. *The International Journal of Robotics Research* 33, 5 (2014), 736–752. DOI: https://doi.org/10.1177/0278364913514466

Chris Paredis and Pradeep K Khosla. 1991. An approach for mapping kinematic task specifications into a manipulator design. (1991).

Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational Design of Linkage-based Characters. *ACM Trans. Graph.* 33, 4, Article 64 (July 2014), 9 pages.

Vortex Dynamics. 2017. CM Labs Simulations. http://www.cm-labs.com/. (2017).

Zhe Xu, Vipin Kumar, Yasutaka Matsuoka, and Emo Todorov. 2012. Design of an anthropomorphic robotic finger system with biomimetic artificial joints. In *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on.* IEEE, 568–574.

Guilin Yang and I-Ming Chen. 2000. Task-based optimization of modular robot configurations: minimized degree-of-freedom approach. *Mechanism and machine theory* 35, 4 (2000), 517–540.

Yuan Yun and Yangmin Li. 2011. Optimal design of a 3-PUPU parallel robot with compliant hinges for micromanipulation in a cubic workspace. *Robotics and Computer-Integrated Manufacturing* 27, 6 (2011), 977–985.

Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. 2012. Motion-guided mechanical toy modeling. *ACM Trans. Graph.* 31, 6 (2012), 127.