International Journal of Humanoid Robotics
Vol. 17, No. 1 (2020) 1950029 (27 pages)

DOI: 10.1142/S0219843619500294

Automated Design of Robotic Hands for In-Hand Manipulation Tasks

Christopher Hazard* and Nancy Pollard†
Robotics Institute, Carnegie Mellon University,
5000 Forbes Ave, Pittsburgh, PA 15213, USA
**chazard@andrew.cmu.edu*
†*nsp@cs.cmu.edu*

Stelian Coros
Computer Science Department, ETH Zurich,
Rmistrasse 101, Zurich 8092, Switzerland
scoros@inf.ethz.ch

Received 30 March 2019
Accepted 1 November 2019
Published 8 January 2020

Grasp planning and motion synthesis for dexterous manipulation tasks are traditionally done given a pre-existing kinematic model for the robotic hand. In this paper, we introduce a framework for automatically designing hand topologies best suited for manipulation tasks given high-level objectives as input. Our pipeline is capable of building custom hand designs around specific manipulation tasks based on high-level user input. Our framework comprises of a sequence of trajectory optimizations chained together to translate a sequence of objective poses into an optimized hand mechanism along with a physically feasible motion plan involving both the constructed hand and the object. We demonstrate the feasibility of this approach by synthesizing a series of hand designs optimized to perform specified in-hand manipulation tasks of varying difficulty. We extend our original pipeline[32] to accommodate the construction of hands suitable for multiple distinct manipulation tasks as well as provide an in depth discussion of the effects of each non-trivial optimization term.

Keywords: Manipulation; trajectory optimization; motion planning; mechanism design.

1. Introduction

Dexterous manipulation has long been a topic of interest in robotic manipulation due to its association with fine motor skills in humans, and the advantages that it can confer upon factory robots and general purpose robots.[13] Dexterous manipulators are able to accomplish motions more efficiently and operate in limited workspace environments more easily.[19] Additionally, we want to develop manipulators that

*Corresponding author.

work in intuitive and human-like ways, particularly if they are meant to work alongside humans.

One line of research in dexterous manipulation focuses on the design of manipulators to mirror the kinematics of the human hand.[17,18] These hands have shown impressive capabilities with regards to dexterous manipulation[27] tasks, however, the problem of dexterous manipulation remains unsolved.[5] One reason for this is that we cannot yet fully replicate the capabilities of the human hands and choices made to simplify the design may end up limiting capabilities of the hand. We have experienced this in our own research when the thumb of a dexterous hand does not have sufficient range of motion or the geometry of the hand's inner surfaces impedes rather than aids performing a manipulation. Progress in this domain is further burdened by the fact that these hands are prohibitively costly.

Rather than trying to approach manipulation from the perspective of human hand kinematics and dynamics, we focus on accomplishing some critical dexterous human hand functions and optimizing mechanisms to perform specific in-hand manipulation tasks. Our vision is to create an optimization pipeline for generating low-cost hands that are well tuned for specific tasks or families of tasks. The possibility of creating useful low-cost hands has been well demonstrated, as in Refs. 7, 8 and 20. In several cases, optimization has been used to tune some of the design parameters for these types of hands.[10] We go beyond the previous work by constructing our hands from scratch based on a given task definition. Our goal is to allow even novice users to easily design a variety of hands for their intended use cases.

In this paper, we introduce an optimization pipeline that takes high-level user specifications such as a sequence of goal poses for a manipulated object and builds a mechanism specifically designed for the given task with no additional parameter tuning required on the part of the user. In this work, we limit ourselves to the class of in-hand manipulations that can be wholly described as reorientation of the object with respect to the palm. However, the pipeline we have developed is extensible to other classes of in-hand manipulations. We show that our pipeline is able to synthesize a wide variety of useful specialized manipulators for various tasks. We build on our previous work[12] by extending our original pipeline to build hands for multiple distinct tasks. We have also generated a comprehensive list of failure cases that arise when each of our non-trivial optimization terms is removed, thereby justifying their inclusion in our final pipeline.

2. Related Work

A large body of work revolves around classifying human manipulation behaviors and replicating them with robotic manipulators inspired by the human hand. Works such as Refs. 1 and 30 attempt to classify the spectrum of human hand manipulations into a hierarchy of grasps and in-hand manipulations covering various phenomena such as rolling motions, controlled slipping, grasp repositioning, and finger gaiting[25] with the intention of mimicking these motions on robotic hands. Platforms such as the NASA

Robonaut hand,[18] GIFU III,[23] and Shadow Dexterous Hand[17] have become standard models on which manipulation algorithms and controllers have been implemented to mimic these types of behaviors. These hands are meant to be generic manipulators that should be able to carry out virtually any manipulation task given an appropriate control policy.

Low DOF hands have the advantage of being easier to build and maintain, easier to control, less expensive, and less prone to mechanical failure since they have fewer moving parts.[9,10] Due to the fact that they are cheaper and easier to build, specifics of the design can be optimized to tune or specialize a given hand. Various works[3,4,6] have optimized continuous parameters such as component lengths, tendon stiffness, and pulley radii to address kinematic concerns such as reachability constraints, avoidance of Jacobian singularities within the workspace, limits on individual joint torques, etc.[28] addresses the problem of discrete optimization of gripper design by chaining together individual modules to build fingers until a desired grasp quality is reached.

We build on previous work by optimizing both discrete and continuous characteristics of hand design to suit specific tasks. A significant portion of our design process consists of trajectory optimizations to test the competence of our hands in performing different tasks. Trajectory optimization methods have shown remarkable ability to synthesize complex motions in both robot locomotion and manipulation, allowing the user to create complicated physically feasible motions from high-level goal specifications. References 14 and 15 developed optimization routines in which an initial grasp pose is specified with a given hand model along with kinematic goals for an object, and a numerical optimizer constructs physically feasible motion plans to synthesize target manipulations. Other work in trajectory optimization for manipulation captures demonstrated manipulations and finds contact forces that explain the motion.[31]

Recent work in trajectory optimization has explored the use of discontinuous contacts in locomotion and manipulation tasks.[21,26] Mordatch *et al.*[21] introduced the concept of contact invariance in which contact is treated as a continuous variable to facilitate optimization with changing contacts. Our work draws inspiration from Ref. 22, which applies the contact invariant method to the domain of manipulation.

Trajectory optimization methods for motion synthesis assume a fixed robot morphology. We do not know of any prior work that attempts to optimize the manipulator design while also creating a motion plan for physically feasible manipulations. Such a task is challenging for in-hand manipulation tasks due to the fact that these tasks are very contact-dependent and hard to model or simulate.

3. Optimization Pipeline Description

We focus on precision in-hand manipulation where the hand manipulates an object with the fingertips in order to change the object's configuration with respect to the base of the hand. The object may be partially supported by the environment.

This type of manipulation is fundamental to acquiring and placing objects, and moving from one grasp to another.[24] Our examples demonstrate 2- and 3-fingered hands, however our approach can be applied to accommodate hands with more fingers.

Our optimization pipeline has three parts, as shown in Fig. 1. The input to our system is a sequence of objective poses for the object, a trajectory for the base, and an initial placement (subject to change) of the contact points on the object. The output of our system is an optimized mechanism, contact points, and forces that meet our objectives in a physically valid motion. The sections below discuss the components of this pipeline originally introduced in Ref. 12, followed by an extension of our original methodology to build hands meant to execute multiple distinct tasks (we refer to these as "multi-objective" designs, whereas hands designed for single manipulations are "single objective" designs).

3.1. *Floating contact optimization*

The floating contact optimization computes optimal contact points and forces that can move the object to its desired objective poses. No information about the robot mechanism is used (or even available) at this point. Specifically, let

$$\mathbf{S_t} = [\mathbf{x_O}\ \mathbf{f_j}\ \mathbf{r_j}\ c_j], \tag{1}$$

be the state at time t of the object, with $\mathbf{x_O}$ denoting the object's position and orientation in the world frame, and $\dot{\mathbf{x}}_\mathbf{O}$ being the derivative at time t of position and

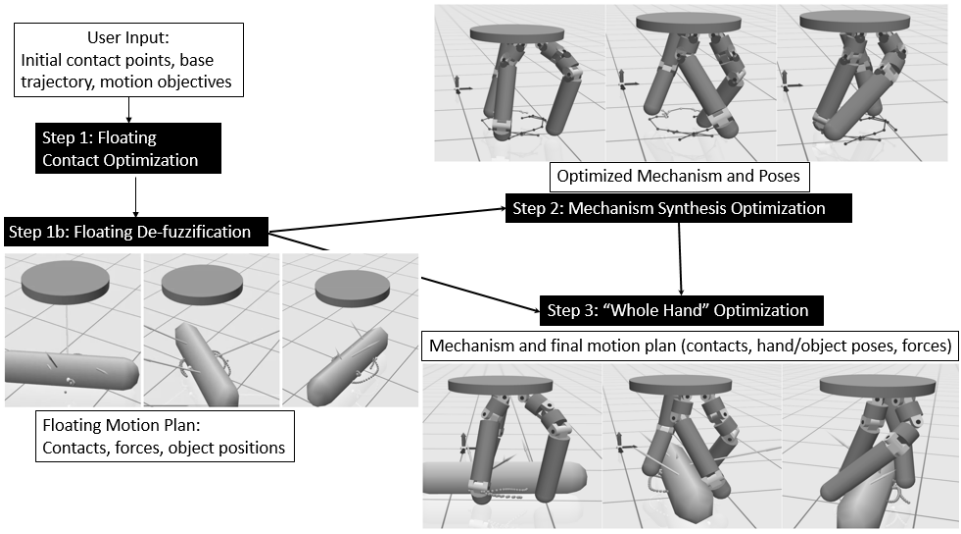


Fig. 1. Given the required user input, our "floating" optimization generates a physically feasible motion plan using disembodied contacts points. We then synthesize a mechanism with fingertips designed to follow these trajectories and provide the required forces. Finally, we combine the floating motion plan with the hand design to adapt the motion to the designed mechanism, outputting the mechanism and a physically valid motion.

orientation. $\mathbf{f_j}$ denotes the force vector at contact point j for $j \in \{1, 2, \ldots, N_{\text{contacts}}\}$ expressed with respect to the world frame, and $\mathbf{r_j}$ denotes the location of contact point j in the local frame of the object. c_j is the contact invariant term described in Ref. 21 that is constrained to lie in the interval [0,1], with 0 representing an inactive contact and 1 being fully active.

We wish to find a trajectory $\mathbf{S} = \{\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S}_{N_{\text{keyframes}}}\}$ such that

$$\mathbf{S} = \underset{\mathbf{S}}{\text{argmin}} \ \Sigma_t \Sigma_i \ w_i * L_i(t), \tag{2}$$

$$\text{s.t. } c_j \in [0, 1] \text{ for } 0 \leq t \leq T, \tag{3}$$

where each cost L_i is in the set $\{L_{\text{physics}}, L_{\text{forceReg}}, L_{\text{frictionCone}}, L_{\text{task}}, L_{\text{ci\_object}}, L_{\text{floatingContactAccel}}, L_{\text{objectAccelReg}}, L_{\text{angularAccelReg}}\}$, which we detail below.

- Physics terms:

$$L_{\text{physics}}(t) = L_{\text{linMom}}(t) + L_{\text{angMom}}(t), \tag{4}$$

$$L_{\text{angMom}}(t) = ||\Sigma_i c_i(t) * (r_i \times f_{i,\text{local}}) - (\omega \times (I^{\text{local}}_{\text{object}}\omega) + I\dot{\omega})||^2, \tag{5}$$

$$L_{\text{linMom}}(t) = \sum_i c_i(t) f_i - m\ddot{x}, \tag{6}$$

  where I_{object} is the moment of inertia of the object in its own local frame, ω is the angular velocity and m is the object mass. The L_{physics} term is responsible for ensuring that the forces acting on the object impart the necessary accelerations to move the object to its destination.

$$L_{\text{forceReg}}(t) = \sum_i ||c_i(t) f_i||^2, \tag{7}$$

$$L_{\text{frictionCone}}(t) = \sum_i c_i * \exp(\alpha(||f_{i,\text{local}} - n * (f_i \cdot n_i)|| - \mu f_i \cdot n_i)), \tag{8}$$

  where n is the local surface normal, μ is the coefficient of friction, and α is a sharpening factor for the exponent that controls how much we penalize contact forces that are close to the friction cone bounds. $L_{\text{frictionCone}}$ is responsible for ensuring that our contact forces are physically feasible, while L_{forceReg} is a regularization term to discourage excessive contact forces.

- Task objectives:

$$L_{\text{task}} = \frac{1}{k} \sum_k ||\text{pos}(k) - \text{pos}_{\text{goal}}(k)||^2 + \text{dist}(o(k), o_{\text{goal}}(k))^2, \tag{9}$$

  where k is the set of keyframes for which we define object goal positions pos_k and orientations o_k. The function dist (q_1, q_2) refers to the quaternion distance formula which is essentially the angle required to rotate from one frame of reference to the other. This particular task objective dictates a set of objective poses (position and orientation) that our object is required to meet. Other task objectives can be

specified depending on the behavior we want to see from our system: for example, we can also specify goals such as tracing out a desired path with an end effector point on the manipulated object.

- Contact Invariant Costs:

$$L_{\text{ci\_object}}(t) = \sum_i c_i ||r_{\text{proj}} - r_i||^2, \tag{10}$$

where r_{proj} is the projection (in local coordinates) of the contact point r_i onto the object

$$e_{\text{object}}(i,t) = r_{i,\text{proj\_object}}(t) - r_i(t). \tag{11}$$

The $L_{\text{ci\_object}}$ cost dictates that the contact points lie on the object surface.

- Additional regularization terms:

$$L_{\text{floatingContactAccel}}(t) = \sum_i || \ddot{r}_i(t)||^2, \tag{12}$$

$$L_{\text{objectAccelReg}}(t) = \sum_i \ddot{x}^2, \tag{13}$$

$$L_{\text{angularAccelReg}}(t) = \sum_i ((\omega \times (I_{\text{world}}\omega) + I_{\text{world}}\dot{\omega})/t_{\text{step}})^2, \tag{14}$$

where x is the manipulated object's position. $L_{\text{floatingContactAccel}}$ regularizes the movement of the floating contact points preventing them from teleporting on the object while $L_{\text{objectAccelReg}}$ and $L_{\text{angularAccelReg}}$ encourage smooth movement of the object.

The motions output by a first pass through the floating optimization have contact invariant values c_i between 0 and 1, and typically cluster around higher values (above 0.7) and low values (0.3 and below), indicating the importance of the contact point in the optimization. After this first pass, we "de-fuzzify" our c_i values by setting each c_i to either 0 or 1 based on a threshold of either 0.1, 0.2, or 0.3. We pick our threshold by testing each one and re-optimizing our floating motion with the contact values held fixed, ultimately picking the "de-fuzzed" motion with the best objective value to pass the synthesis optimization. Future steps in the pipeline hold these contact values fixed.

3.2. *Mechanism synthesis continuous optimization*

The synthesis step involves both the optimization of the discrete structure of the hand (the number of joints per finger) as well as the optimization of continuous parameters governing the mechanism design. In this section, we describe the continuous optimization, which is used by the discrete optimization procedure described in Sec. 3.3. The optimization below assumes that we have all discrete parameters (i.e., the kinematic structure) fixed.

We optimize a set of morphological parameters $\mathbf{M} = \{\mathbf{L}\ \mathbf{A}\ \mathbf{B}\}$, a set of joint angle poses $\mathbf{Q} = \{\mathbf{Q_1}, \mathbf{Q_2}, \ldots, \mathbf{Q}_{N_{\text{keyframes}}}\}$, and a set of contact points $\mathbf{P}$ on the constructed fingertips such that

$$\mathbf{M},\mathbf{Q},\mathbf{P} = \underset{\mathbf{M},\mathbf{Q},\mathbf{P}}{\operatorname{argmin}} \Sigma_k \Sigma_i\ w_i * L_i(k), \tag{15}$$

$$\text{for } k \in \{1, 2, \ldots, N_{\text{keyframes}}\}, \tag{16}$$

where $\mathbf{L}$, $\mathbf{A}$, and $\mathbf{B}$, respectively represent the finger segment lengths, joint axis orientations, and positioning of fingers on the base/palm of the hand and L_i are the costs in the set $\{L_{\text{eeTarget}}, L_{\text{contactDistSurface}}, L_{\text{collision}}, L_{\text{fingerLengthRegularization}}, L_{\text{fingerMinLength}}, L_{\text{jacNull}}, L_{\text{torque}}, L_{\text{fingerPositions}}, L_{\text{fingerAcceleration}}, L_{\text{jointLimits}}\}$. To discourage slipping on the fingertips, we restrict contact points p to remain fixed in the fingertip's local frame when that contact is active. The term "fingertip" used throughout this paper refers to the surface of the last segment on a given finger, not the actual tip of that finger segment. Our contact points are therefore able to lie anywhere on the surface of these finger segments.

- Contact point costs:

$$L_{\text{eeTarget}}(k) = \sum_i c_i * ||p_i - p_{\text{target}}||^2, \tag{17}$$

$$L_{\text{contactDistSurface}}(k) = \sum_i ||p_{\text{proj}} - p_i||^2, \tag{18}$$

  where c_i represents the binarized contact invariant term (either 0 or 1) for fingertip i at keyframe k. L_{eeTarget} is the distance between the contact point p_i on fingertip i and the corresponding point on the given trajectory for that contact, encouraging our selected contact points to line up with the trajectories from the floating motion plan. $L_{\text{contactDistSurface}}$ is the distance between the contact point and its projection onto the surface of the fingertip it is attached to: this is paired with a high coefficient to force contact points to lie on the surfaces of the fingertips.

- Collision
  For collision penalty calculations, we use a second-order smooth piecewise cubic spline that interpolates between the functions $f(x) = 0$ for $x < 0$ and $f(x) = x^2$ for $x > 0$ as follows:

$$g(x) = \begin{cases} 0 & x \leq -\epsilon \\ \dfrac{x^3}{6\epsilon} + \dfrac{x^2}{2} + \dfrac{\epsilon x}{2} + \dfrac{\epsilon^2}{6} & -\epsilon \leq x \leq \epsilon \\ x^2 + \dfrac{\epsilon^2}{3} & \epsilon \leq x, \end{cases}$$

$$L_{\text{collision}}(k) = \sum_{i,j \in \text{bodies}} g(\text{pen}(\text{body}_i, \text{body}_j)), \tag{19}$$

where penetration distances are calculated such that non-penetrating bodies have negative penetration distance (hence no collision cost) and ϵ is simply a small arbitrary constant ($\epsilon = 10^{-6}$).

- Finger length costs:

$$L_{\text{fingerLengthRegularization}} = \sum_i (l_i)^2, \tag{20}$$

$$L_{\text{fingerMinLengthCost}} = \sum_i g(l_{\min} - l_i), \tag{21}$$

where i ranges over all the capsules present in the hand, l_i denotes the length of the principle axis of capsule i, and g denotes the piecewise cubic spline defined above. These two costs are meant to keep the finger lengths within a reasonable range of values.

- Controllability related costs:
We require our mechanisms to be able to actively supply the necessary forces needed to accomplish the target motion. This is done through two terms: one to penalize the component of the applied force that lies along the null space of our mechanism's Jacobian (requiring our mechanism to actively supply the necessary force) and another to regularize the torque applied at the joints (encouraging efficient mechanisms).

$$L_{\text{jacNull}} = \sum_i c_i \times \sqrt{\sum_k (f \cdot e_k)^2}, \tag{22}$$

where the vectors e_k consist of an orthonormal basis of the null space of the manipulator Jacobian for each given finger/contact point pair i, f being the force required for the finger to provide at the end effector, and c_i being the contact invariant weight (either 0 or 1) for the given contact.

$$L_{\text{torque}} = \sum_i \|\boldsymbol{\alpha}\|^2, \tag{23}$$

where $\boldsymbol{\alpha}$ is the vector of torque magnitudes that must be supplied by the mechanism to actively provide the desired force. Note that this torque penalty does not take into account the torque required to compensate for gravity, nor does it account for a hard maximum on the allowed torque to be supplied by a given motor (although this could be added if necessary). We calculate this as follows: for any given finger, we have $T = r \times F$, where T is the torque applied with respect to a given joint, F is the force at the selected contact point (the end effector), and r is the lever arm. We can decompose this into $F = T \times r_{\text{perp}}/\|\mathbf{r}_{\text{perp}}\|^2 + k * r_{\text{perp}}$ where $r_{\text{perp}} = r - (r \cdot a) * a$ is the component of r perpendicular to the unit vector a aligned with the rotation axis of the joint in question. In the above equation, k is a constant and $k * r_{\text{perp}}$ represents the passive force applied to this joint: setting $k = 0$, we have $F = X * \alpha$ where X is the matrix consisting of column vectors

$T \times r_{\text{perp}}/\|\mathbf{r}_{\text{perp}}\|^2$ concatenated for each joint and α is the vector of torque magnitudes actively applied at the joints. Then, $\boldsymbol{\alpha} = (X^T X + \lambda^2 I)^{-1} X_T F$ is the singularity robust psuedo-inverse[2] solution with lambda being a small constant ($\lambda = 0.001$). Note that this torque penalty does not take into account the torque required to compensate for gravity nor does it account for a hard maximum on the allowed torque to be supplied by a given motor (although this could be added if necessary).

- Additional costs:

$$L_{\text{fingerPositions}} = \sum_i ||\text{proj}_{\text{base}}(b_i) - b_i||^2, \tag{24}$$

where i ranges over all the bases of the fingers and we find the closest projected point onto the base. This term ensures that our fingers are attached to the surface of the base and can be applied to a variety of base shapes as long as a smooth projection formula exists for the surface. In this work, we limit ourselves exclusively to circular bases although this can be readily extended.

$$L_{\text{fingerAcceleration}}(k) = \sum_i (1 - c_i) * \ddot{x}_i^2, \tag{25}$$

where $\ddot{x_i}$ is acceleration of fingertip i and c_i is the contact invariant term for that fingertip in frame k. Fingertip acceleration only applies to contacts that are inactive in order to encourage smooth transitions for lifted fingers.

$$L_{\text{jtLimit}}(k) = \sum_i \sum_j g(j(i) - j_{\max}) + g(j_{\min} - j(i)), \tag{26}$$

where j runs over our set of joints. The terms $j_{\max}$ and $j_{\min}$ are constant joint limits set to $\pi/2$ and $-\pi/2$, respectively and g is our piecewise cubic spline introduced earlier for smooth interpolation.

3.3. *Mechanism synthesis discrete optimization*

The process by which we optimize the discrete structure of our hand designs is relatively simple. We optimize fingers independently for computational efficiency and treat joints as being independently controlled. We keep adding additional finger segments (and joints) to each of our fingers until the combined L_{eeTarget} and L_{jacNull} scores for our finger fall below a predefined threshold or until we reach an upper limit on the number of joints allowed per finger.

After optimizing each finger independently for multiple trials, we enter a recombination step in which we combine the top performing fingers into a complete hand design. Upon recombination, we must re-optimize our hand due to the fact that we may incur self-collision among the recombined fingers (since they were optimized independently, their motions can easily overlap). The first recombination trial always takes the top performing fingers from each set of fingers meant to track the

end effector points. Additional recombination trials randomly select fingers from each set according to a weighting that is inversely proportional to their combined L_{eeTarget} and L_{jacNull} scores (so that fingers with lower costs have higher chances of being selected). We then take our best performing hand, and if the combined L_{eeTarget} and L_{jacNull} scores for each finger fall below our threshold, we return this design as our constructed hand. Otherwise, we add an additional joint to each of the fingers with scores still above this threshold and repeat the loop again until either that finger hits the maximum number of joints allowed per finger or it meets our objective criteria.

3.4. *Whole hand optimization*

As the final stage in our motion optimization pipeline, we take the generated motion plan for the object and the hand mechanism constructed for it to go about a trajectory optimization similar to the one used in step 1. We introduce several additional terms to the objective function to create a physically realistic motion with respect to the hand and we add the joint angles at each keyframe to the list of variables that we intend to optimize. We do not restrict contact points to be stationary with respect to the fingertips as we did in the synthesis step, thereby allowing us to perform some small degree of slipping and rolling. We do not explicitly model slipping or rolling on the fingertips, though we discourage these via the imposition of soft constraints. Below, we detail the additional terms added to the objective function:

- $L_{\text{jointLimits}}$: This is Eq. (26), taken from the synthesis step.
- $L_{\text{fingerAcceleration}}$: Eq. (25) taken from the synthesis step.
- $L_{\text{collision}}$: Eq. (19) from synthesis.
- L_{jacNull} and L_{torque}: Eqs. (22) and (23) from the synthesis step.
- Finger contact invariant term: $L_{\text{ci\_finger}}$ mirrors the contact invariant term introduced in the floating contact optimization, but applied to the finger instead of the object to encourage the contact point to lie on the finger. This works alongside $L_{\text{ci\_object}}$ to encourage the finger to be in contact with the object without overconstraining the optimization problem.

$$L_{\text{ci\_finger}}(t) = \sum_i c_i ||r_{\text{proj}} - r_i||^2, \tag{27}$$

  where r_{proj} is the projection (in world coordinates) of the contact point r_i (world coordinates) onto the finger
- Slippage constraints: We do not explicitly penalize contact slippage on either the object or the fingertip since this would be overly restrictive and prevent our motions from naturally exhibiting interesting slipping and rolling behaviors. Instead, we adapt a pair of soft constraints introduced in Ref. 22 to our pipeline that works to implicitly regularize slippage. These constraints essentially require that the distance given a contact slip with respect to the object and the fingertip is equal with respect to the world frame.

$$L_{\text{ci\_finger\_slippage}}(t) = \sum_i ||c_i f_i||^2 \times ||(\dot{e}_{\text{finger}}(i,t))||^2, \tag{28}$$

where $e_{\text{finger}}(i,t) = r_{i,\text{proj\_finger}}(t) - r_i(t)$,

$$L_{\text{ci\_object\_slippage}}(t) = \sum_i ||c_i f_i||^2 \times ||(\dot{e}_{\text{object}}(i,t))||^2. \tag{29}$$

- $L_{\text{frictionConeHand}}$: Since our finger may not be perfectly tangent to the object it makes contact with, we introduce an additional friction cone term that mirrors the friction cone with respect to the object, but using the outgoing normal from the fingertip at the contact point instead (similar to Eq. (8)). This prevents us from exerting unrealistic forces with respect to the fingertip surface.

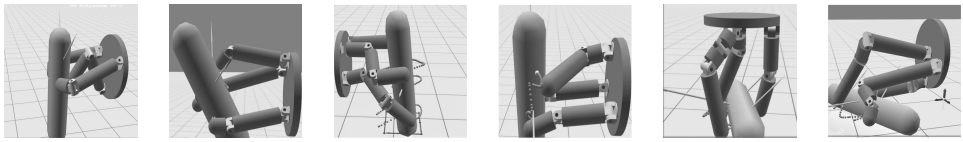
$$L_{\text{frictionConeHand}}(t) = \sum_i c_i \times \exp(\alpha(||f_i - n \times (f_i \cdot n_i)|| - \mu f_i \cdot n_i)), \tag{30}$$

where n is the surface normal to the fingertip (world frame), μ is the coefficient of friction, and α is a sharpening factor for the exponent.

4. Results for Single Objective Hands

In the first part of our accompanying video, we demonstrate a set of example motions ranging from simple object re-orientations to multi-step motions. In Fig. 2, we demonstrate the ability of our pipeline to generate feasible mechanisms on a variety of in-hand manipulation tasks. Figure 4 demonstrates two sequences in which we build up progressively more complex motions from a set of simple primitive motions. From these examples, we can see a variety of different mechanisms arise to meet our task specifications rather than a generic one-size-fits-all design, and we note that the complexity of these mechanisms scales directly with the complexity of the given task.

Our pipeline is robust in the sense that we use the same fixed set of optimization weights for each step regardless of the motion. In the vast majority of cases, the final result of our "whole hand" optimization yields very low physics error penalties and near zero Jacobian null space projection penalties, indicating that the pipeline



(a) (b) (c) (d) (e) (f)

Fig. 2. Here we reproduce still-frames of several example hands/motions generated by our system (refer to accompanying video for complete motions): (a) A vertical flipping motion, as if feeding a part. (b) Rotate from horizontal to vertical and bow out the capsule. (c) Drawing a box with a pen on the ground. (d) Rotating a capsule 180°. (e) Tabletop rotation with hand above the object. (f) Tabletop rotation with the hand on the side of the object.

Fig. 3. To demonstrate the feasibility of our designs, we fabricated a hand with 3D printed parts scaled with regard to the embedded motors. This particular design is meant to rotate a 4 inch diameter ball 180° either way.

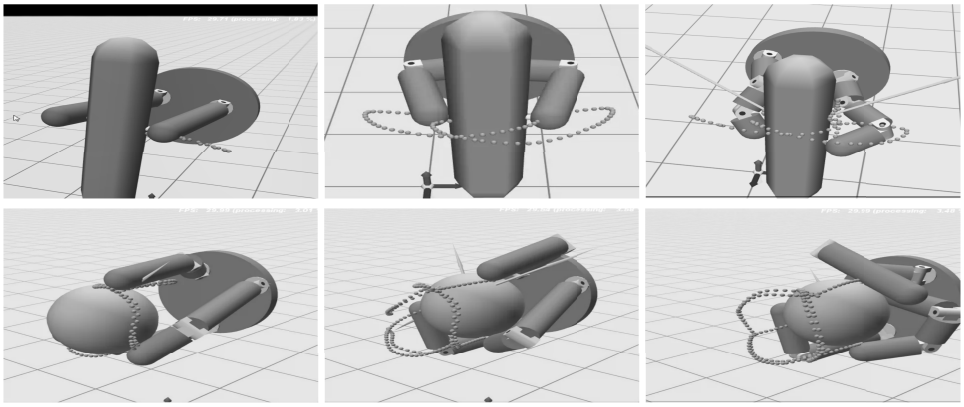


Fig. 4. Here we demonstrate two sequences of progressively more complex motions (refer to accompanying video for complete motions): (above) We demonstrate a horizontal side-to-side motion (without gravity, so that we don't require our mechanism to actively provide counter-gravitational force), a circular planar motion (no gravity), and a hemispherical motion (with gravity). (Below) We demonstrate a line of motions (all with gravity enabled) beginning with a sphere rotation 180° either way, followed by a sphere rotation with the ability to translate in and out, and finally a sphere rotation with the ability to translate in the plane.

generates a motion plan that fits very well the designed mechanism, ensuring that the mechanism is capable of performing the motion in a physically realistic way.

Our optimization program is able to discover interesting aspects of our motions that lead to non-trivial mechanical designs. For example, in Fig. 2(b), our optimization was able to suggest a mechanism in which we use one of our upper fingers to push out our capsule to bow it out while rotating it 90° perpendicular to our palm surface using the other two fingers to pivot the object. Our mechanism originally bows out the mechanism beyond the 45° target, then slides the pushing finger upward and reduces the force it exerts to achieve the desired position. In Fig. 2(c), we replace our usual pose objective with an objective that attempts to track the shown goal points with the tip of the gripped "pencil". In this motion, our optimization discovered a cyclic manipulation in which the finger on the bottom left of the pencil automatically resets itself while still maintaining contact on the object. In the middle motion shown in Fig. 4(b), we dictate that our mechanism is to rotate the sphere 180° either way followed by a translation in and out from the palm. Surprisingly, our

optimization found a way to do this with only two degrees of freedom per finger by discovering that our hand can "lock" in our object by folding the distal joints. Normally, one would expect such a manipulation to require at least three DOF's per finger as in the succeeding motion, in which we require that the sphere also be able to translate side-to-side as well as in and out.

We are often able to create distinctly different mechanisms for the same motion simply by varying the initial contacts placed on the object or by varying the initial position of the base. We demonstrate this in Figs. 2(e) and 2(f) in which we place our contacts in the same positions but placed our base differently: the result is that our floating motion plans are identical, but we get two completely different mechanisms out of the initial conditions. Similarly, we can get distinct mechanisms from placing our initial contacts differently. The fact that our pipeline gives different results for different initial conditions means that the user can select their ideal mechanism by trying out different initial conditions, as well as gain intuition about how the base and contact initialization affect the optimal mechanism design in general.

To demonstrate the feasibility of our designs, we fabricated a physical prototype for a hand that is meant to rotate a sphere 180° forward and backward (shown in Fig. 3). Given a design generated by our pipeline, we programatically generate a set of individual parts represented via constructive solid geometry which are then converted to a set of 3D printable CAD files. Due to the fact that we have embedded the motors in the fingers, we have scaled our design according to the size of the individual motors. Our prototype hand (equipped with the computed set of poses) is capable of reliably rotating a 4 inch Styrofoam ball from a variety of initial palm orientations. This motion involves a significant amount of rolling between the fingertips and the object which is made possible by its unique design.

5. Extension to Multi-Objective Hands

In this paper, we extend upon our previous work in Ref. 12 by making our pipeline capable of designing hands meant for multiple separate motions. To do this, we have added a set of terms designed to couple contact trajectories to be similar to one another in both location and shape. We have also added a term that iterates over the contact points and deactivates the contact force, replacing it with a perturbed version of the computed contact force that acts along the same direction as in the unperturbed case, causing the other contact forces to re-balance to reject the disturbance.

In Fig. 5, we demonstrate the need for our trajectory coupling terms for multi-objective motions on a combination of the sphere rotation and capsule rotate and bow out motions. With our coupling terms, our pipeline generates a design that is similar to our single objective sphere rotation motion with two dofs per finger. Without the coupling terms, it either generates an under-actuated hand that can't provide the required force to move the object as intended, or it provides an overly complicated hand that yields a brittle motion.

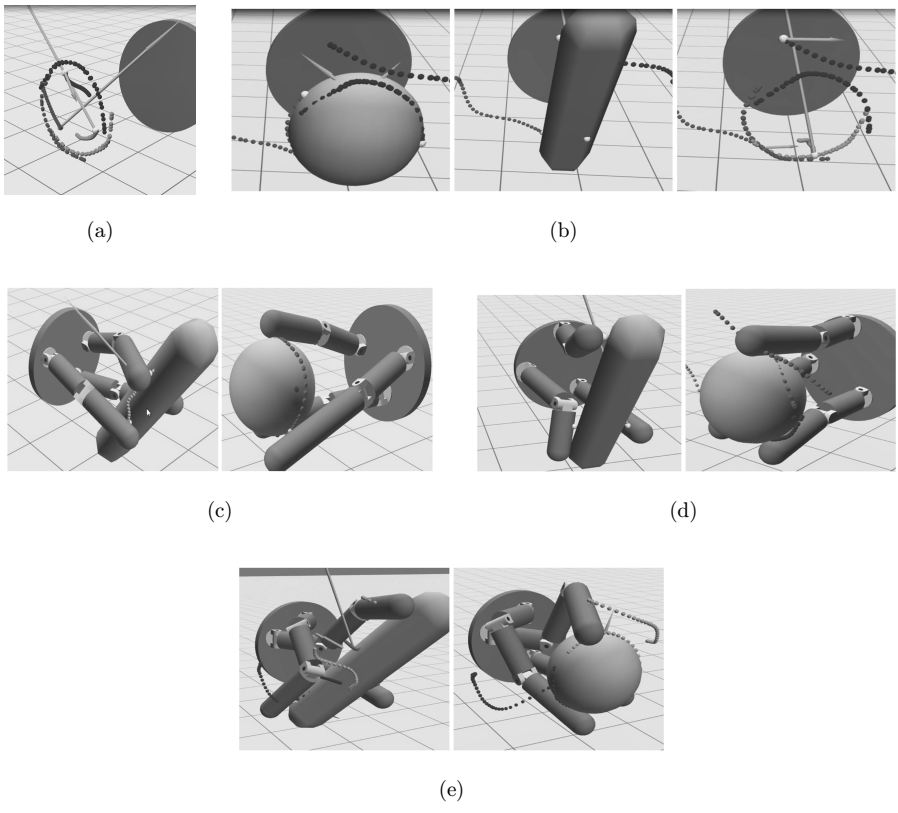


Fig. 5. In this example, we combine the sphere rotation motion with the rotate and bow out motion from earlier figures. (a) shows the result of our multiobjective floating optimization with the coupling terms enabled as well as the overlaid contact trajectories. (b) demonstrates the same optimization with the coupling terms removed (initial conditions were kept the same as in (a)), resulting in dissimilar sets of contact trajectories. (c) shows the results of the coupled trajectories at the end of the "whole hand" optimization. The resulting hand that is capable of both motions is nearly identical to our earlier single objective example for sphere rotation. (d) demonstrates the best scoring proposed hand design from the second-level of the synthesis step for the trajectories in (b) after passing through the "whole hand" optimization: this design is physically infeasible since 22% of the magnitude of the total force applied by the fingers lies in the Jacobian null space. (e) shows the next level synthesis design for the uncoupled trajectories after passing through the final optimization step. This design is physically feasible according to our objective function, however it is clearly an awkward and brittle mechanism.

To accommodate multi-objective motions, we introduce a set of trajectory coupling terms and perturbing force objectives into our floating contact optimization. These coupled trajectories are then sent to the synthesis step in which we optimize the structure of our hand to minimize the sum of the individual costs incurred for each target motion. In other words, our synthesis step runs a single optimization for each motion with the constraint that all morphological variables are shared among the individual synthesis optimization problems. This allows efficient parallel processing. Once the synthesis step has found a hand design that is optimal in the

combined case, we run our "whole hand" optimization on each motion individually, where the hand design is fixed and each motion is independent. This means that we no longer use the trajectory coupling objective nor do we use the perturbing force objective in the "whole hand" optimization.

We will now derive our trajectory coupling terms. First we must group together the floating contact trajectories for each fingertip between motions: that is, we must find which trajectory in Motion 2 matches finger trajectory 1 in Motion 1. Suppose we have m motions $\{M_1, M_2, \ldots, M_m\}$ and each motion has contact trajectories $\{T_1, T_2, \ldots, T_f\}$ where we have f total fingertip contacts. We will denote trajectory j from motion i as $M_{i,j}$, each of which has a mean value of $a_{i,j}$. Our coordinate frame for these contact points is always defined with respect to the palm of the hand. A trajectory grouping is defined as a set of mappings from the set of trajectories from M_1 to each of the other motions $\{M_2, \ldots, M_m\}$. To find our initial grouping, we enumerate all possible trajectory groupings to find the one that minimizes the objective $\sum_{t\in\{T_1,\ldots,T_f\}}\sum_{i\in M}\sum_{j\in M|i\neq j}(a_{i,\mathrm{im}_i(t)} - a_{j,\mathrm{im}_j(t)})^2$ where $\mathrm{im}_i(t)$ denotes the image of trajectory t from M_1 to motion M_i under mapping dictated by our trajectory grouping. Put simply, we minimize the squared distance of the means of each trajectory pair that we match up from each motion.

We now calculate three optimization terms given our grouping (which stays the same from here on out) in our multiobjective floating contact optimization, namely L_{variance}, L_{mean}, and $L_{\text{similarity}}$. For each trajectory $M_{i,j}$, we perform principle components analysis to find eigenvectors $\{v_{i,j,1}, v_{i,j,2}, v_{i,j,3}\}$ and eigenvalues $\lambda_{i,j,1}, \lambda_{i,j,2}$, $\lambda_{i,j,3}$ as well as normalized eigenvalues $\overline{\lambda_{i,j,1}}, \overline{\lambda_{i,j,2}}, \overline{\lambda_{i,j,3}}$ where we divide by the sum of the eigenvalues for the trajectory in question. Now given a trajectory grouping $\{g_1, \ldots, g_m\}$ where g_i denotes the mapping from trajectories in motion 1 to motion i, we compute the terms:

- $L_{\text{variance}} = \sum_i \sum_j \sum_k \lambda_{i,j,k}$,
- $L_{\text{similarity}} = \sum_{T_i}\sum_{g_j}\sum_{g_k}\sum_m\sum_n \overline{\lambda_{j,g_j(T_i),m}} \cdot \overline{\lambda_{k,g_k(T_i),n}}(v_{j,g_j(T_i),m} \cdot v_{k,g_k(T_i),n})^2$,
- $L_{\text{mean}} = \sum_{T_i}\sum_{g_j}\sum_{g_k}(a_{j,g_j(T_i)} - a_{k,g_k(T_i)})^2$,

where $g_k(T_i)$ denotes the mapping from trajectory i in motion 1 to its corresponding trajectory in motion k under our trajectory grouping g. These three additional optimization terms are added to the linear combination objective function used for our floating contact optimization, thus our total objective is the sum of the individual floating objectives plus the coupling terms. $L_{\text{similarity}}$ encourages contact trajectories to take the same shape and is optimal when each trajectory lies along the same principle component, while L_{variance} encourages contact points to remain close to their mean if possible, and L_{mean} encourages contacts to be centered in the same region relative to the palm.

Due to the fact that our coupling terms tend to pull contacts away from the object to make trajectories more similar, our coupling term can end up making motions more brittle. To prevent this, we need to add a measure for robustness of our

motions. One way to do this is to calculate a different set of forces (keeping contact locations the same) under an additional scenario in which a particular perturbing force is applied to the object throughout its motion. Though we have successfully experimented with this approach, it does require that we manually specify sets of perturbing forces for each motion since no single set of perturbing forces works in all cases.

Since we want to require as little user input as possible, we instead introduce an additional contact-wise perturbing force objective and an additional set of optimization variables to determine the applied forces in each additional perturbing scenario. We iterate through each contact point (each of these is considered a different perturbing force scenario) and apply a small perturbing force in the same direction as the contact force at that point in the unperturbed case: now we recalculate the applied force at each of the other contact points (which are included as additional optimization variables) in order to balance this disturbing force such that our net force and torque on the object remains unchanged. If we have a total of N contacts,

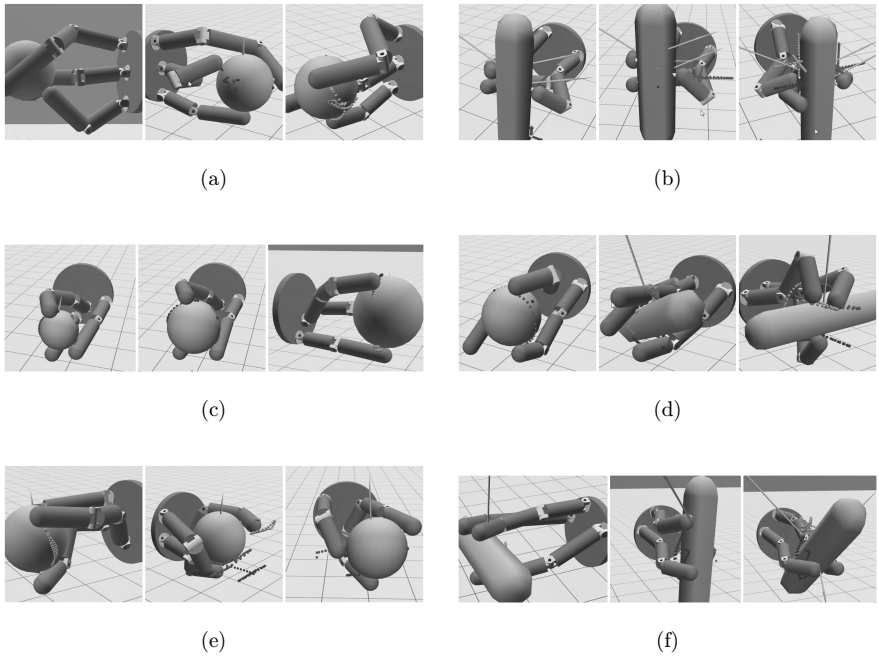
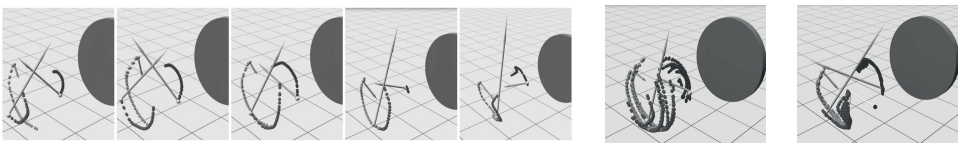


Fig. 6. Here, we demonstrate several examples of motion combinations passed through our multi-objective pipeline. Each row shows a single-hand design that has been optimized for three different target motions. We refer the reader to our accompanying video for the full motions. (a) Is optimized for a case in which we want to rotate a sphere around three orthogonal axes. (b) Translation of a capsule along three orthogonal axes. (c) Rotation of several different sized spheres (i.e., multi-scale sphere rotation). (d) Accomplishes three dissimilar tasks, namely sphere rotation, capsule translation, and rotating and bowing out a capsule. (e) Is optimized for the individual motions comprising our progressive sphere motion sequence from Fig. 4. Note that this hand requires fewer degrees of freedom since the motions are no longer required to execute in sequence. Finally (f) Shows a capsule rotated around three orthogonal axes.



(a) (b) (c)

Fig. 7. (a) We illustrate five different valid floating motions that are capable of executing the sphere rotation example from earlier. (b) We see the overlaid independent trajectories, where contacts from separate motions have been matched up on a nearest neighbor criteria (shown with the same color). This is equivalent to a multi-objective floating contact result in which the trajectory coupling terms introduced in this section are disabled. (c) This is the set of trajectories produced by the multi-objective optimization with coupling terms enabled. Since the desired motions are equivalent in this scenario, the contact trajectories for each random seed end up converging to the same set of trajectories to minimize the coupling score.

then we add $3 \times (N-1)$ additional optimization variables per contact per keyframe to give us our resulting contact forces for a total additional $3N(N-1)$ variables per keyframe. For each of the N new sets of forces acting on the object, we calculate $L_{\text{frictionCone}}$ and L_{physics} (Eqs. (4) and (8)) and add them to our objective with the same weighting as in the unperturbed case and normalize the sum by $N+1$. The point-wise perturbing forces encourage placement of opposing contacts to make motions more robust and the L_{physics} and $L_{\text{frictionCone}}$ calculated for these perturbing cases encourage contacts to remain active via the contact invariant weighting c_i.

Figure 7 demonstrates the ability of our coupling terms to reduce different trajectories to similar trajectories. In this example, we take five different physically valid floating motions each created from a separate initial contact seed for the sphere rotation task. Running these combined motions through our multi-objective floating optimization causes similar contacts between motions to converge to the same trajectory. This demonstrates that the coupling terms have the potential to remove unnecessary complexity from the floating contact optimization, thereby simplifying the synthesis optimization problem.

6. Analysis of Individual Optimization Terms

In this section, we demonstrate the necessity of each group of terms in our optimization framework and show examples of failures caused by the removal of each of these terms. While terms such as L_{task} and L_{physics} in our first and third optimization steps are obviously necessary for our pipeline to produce meaningful physically feasible designs and motions, the effects of several of our optimization terms are not quite as obvious. Due to the large number of terms in our optimization pipeline, we will restrict this discussion to only the non-trivial optimization terms with non-obvious utility and/or interesting behavior. Figures for each of the examples discussed in this section are shown in Appendix B.

In the preceding section, we established the need for our coupling terms in Fig. 5. As mentioned earlier, the perturbing forces heuristic helps to give us more robust motions by counteracting the tendency of our coupling terms to lift contacts unnecessarily: we present an example of this in Fig. B.1. In the case where we do not include perturbing forces, some contacts become inactive during the floating optimization step. Since contacts are binarized after the floating step, it rare for the "whole hand" optimization to re-establish inactive contacts unless they are absolutely necessary.

The other non-trivial terms we would like to highlight are the Jacobian null space penalty (Eq. (22)), the friction cone with respect to the hand (30), applied torque regularization (23), and the contact slippage constraints ((28) and (29)).

Figure B.2 shows that if we remove the Jacobian null space term, our optimization shows no regard for whether or not the hand can actually provide required forces for the motion. Figure B.3 shows the need for the $L_{\text{frictionConeHand}}$ term in cases where minor collision occurs between rigid bodies. If contacting rigid bodies were always perfectly tangent, $L_{\text{frictionConeHand}}$ would just duplicate $L_{\text{frictionCone}}$, but this cannot be guaranteed since we penalize collision as a soft cost. Figure B.5 shows a case in which the bottom contact point in a capsule translation motion teleports on the finger due to the lack of a large enough contact surface. Increasing our slippage penalty coefficients causes our finger to change its configuration to provide a larger surface for the contact to realistically slip on (note that the weight change is unique to this example for the purposes of demonstration). The torque regularization $L_{\text{torqueReg}}$ term helps the pipeline generate mechanisms that can efficiently accomplish the task. Figure B.4 demonstrates the ability of our optimization to build mechanisms that require as little joint torque as possible for actuation.

7. Discussion of Pipeline Limitations

There is a disconnect between the floating optimization and synthesis optimization in our pipeline due to the simple fact that the floating optimization has no concept of what a finger is or how kinematic constraints can limit motion capability. It is therefore not possible for our system to generate manipulations like finger gaiting motions or meaningful grasp transitions organically, although we can force this behavior to occur if we manually set the initial contacts at every frame. This disconnect is the cause of most of the failure cases we have observed. Sometimes, the floating contact optimization gives a non-collision free trajectory: In general, this can be addressed by providing multiple initial seeds for contacts. Additionally, poor selection of base location can give awkward looking mechanisms.

In future work, we may try to combine the design optimization with the contact planner to resolve these issues. This may allow our pipeline to correct for poor motion planning or design choices made earlier on by the pipeline. Although the final optimization has the capacity to re-plan contact trajectories, it is not always able to

successfully re-plan a given motion to fix a flawed contact trajectory/mechanism pair, as demonstrated in Fig. B.7. We envision a feedback loop in which we iteratively design a mechanism and re-plan its motion until we reach a satisfactory solution. Our pipeline is limited to placing contacts on the distal finger segments, the palm, and objects in the environment, however a planning/design feedback loop may allow us to use intermediate finger segments as contact surfaces allowing us to model motions like power grasps.

Another important limitation of our pipeline is that we do not explicitly model rolling constraints on the fingertips. In most cases, our choice to not model rolling does not introduce any noticeable physical infeasibility. In some cases, such as the sphere rotation motions, feasible rolling behavior seems to emerge anyway due to our other optimization terms. As Fig. B.6 demonstrates, our method has difficulty in planning for rolling motions where rolling is the central focus of the particular manipulation. Our pipeline is able to generate reasonable floating contact trajectories, but our synthesis step is designed around the premise that a single contact point on the fingertip should be able to track the contact trajectory matched to that finger (this is done to make the synthesis optimization tractable). It is therefore up to the "whole hand" optimization to adjust contact positions between frames to account for any slipping or rolling that may occur which is not always possible given the mechanism design.

Future work should also improve upon the way we measure and optimize for robust motions. Although we have observed that sets of manually specified perturbing forces can encourage robustness, this solution is not generally enough. This led us to develop our point-wise perturbing force heuristic introduced in Sec. 5. This heuristic encourages robust contact placement, however, it is not a perfect solution either since it can add unnecessary constraints on contact placement. While the heuristic is necessary for multi-objective hands to counter the negative effects of our coupling terms, we do not strictly need it and therefore do not include it in our single objective examples. Our heuristic tends to make contact points line up such that they oppose one another, however, this can be too restrictive for some of our more complex single objective motions like our pen drawing motions.

8. Concluding Remarks and Additional Directions for Future Work

In this work, we have presented a methodology for generating task specific manipulators for both single motions and multiple motions. We believe that the pipeline introduced in this paper can serve as the basis for development of a scalable and increasingly sophisticated design tool that is intuitive, user-friendly, and allows users to generate designs to suit their particular needs. One of the benefits to our approach is that our pipeline can be further outfitted with self-contained modules that can be separately developed before being incorporated into our final pipeline. A particular module that may prove useful is an automatic linkage designer[29] to reduce degrees of freedom after the "whole hand" optimization step.

The main shortcoming of our methodology is that it does not adequately plan for robustness under environmental uncertainty. Issues with building robust manipulators can possibly be addressed by adding a simulation-based optimization to the end of our pipeline in which we take the trajectory optimization-based design and put it in a Physics engine to carry out the intended motion in various scenarios. Our generated hands are optimized with respect to a single known starting configuration as well as known object size and weight. In future work, we plan to design mechanisms that are robust to sensor noise, variations in object geometry, friction, mass, etc.

Appendix A. Implementation Details

Although our optimization has very few parameters for a user to tune, there are internal weights and other details required to duplicate these results. Weights were set the same for all examples. We observe that these weights lead to qualitatively similar motions as long as they are set to within an order of magnitude of the listed values. Our implementation for each segment of the pipeline is multithreaded wherever possible. Most examples in the paper completed between 15 and 30 min each through the pipeline from start to finish on a quadcore Intel I7 laptop.

A.1. *Floating contact optimization*

When calculating our objective in the floating contact optimization, we interpolate between the keyframes and calculate the sum at each time according to a time step $t_{\text{step}} = 0.1\,\text{s}$. Weights for the objective terms are given in Table A.1. Depending on the number of steps involved, we set our motions to last between 4 and 12 s for most of our example manipulations (more complex manipulations require longer time horizons for smooth motions). The number of keyframes we use for a given motion is dependent upon the number of objective keyframes specified (which is also a measure of the motion's complexity): we typically use 1–3 additional keyframes between each of our objective keyframes, spaced uniformly over time. The object position component of $\mathbf{x_O}$ and contact positions $\mathbf{r_j}$ are interpolated via catmull-rom splines, object orientations are computed via the exponential map[11] and $\dot{\mathbf{x}}_\mathbf{O}$ are calculated via finite

Table A.1. Table of common weights for floating contact optimization.

w_{physics}	10	$w_{\text{ci\_object}}$	100
w_{task}	50	w_{forceReg}	0.01
$w_{\text{floatingContactAccel}}$	0.01	$w_{\text{objectAccelReg}}$	0.1
$w_{\text{objectAngAccelReg}}$	1	$w_{\text{frictionCone}}$	0.1
$\alpha_{\text{(friction sharpen)}}$	5	$w_{\text{coupling\_similarity}}$	0.25
$w_{\text{coupling\_mean}}$	10	$w_{\text{coupling\_variance}}$	0.01

differences. The contact forces $\mathbf{f_j}$ are linearly interpolated and the $\mathbf{c_j}$ are evaluated as piecewise-constant terms.

In our optimization, we allow for one point contact on each fingertip on our hand and a single point contact to allow the object to interact with the ground. We optimize our objective with a standard L-BFGS solver.[16]

Except for the L_{task} term, we normalize each of our cost terms by the number of keyframes in our motion. We initialize the object poses in our keyframes by interpolating between our objective goal positions and determine the contact positions by keeping the same initial contacts with respect to the objects of the local frame (i.e., initial contacts are just translated and rotated with the object). This initialization gives zero L_{task} cost on the first step of the optimization prompting the optimizer to focus on solving for the forces and contact positions needed to satisfy the L_{physics} term. The result is that we tend to see an optimization procedure in which L_{task} remains small while the L_{physics} term dominates the optimization eventually leading to a solution with only slight deviation from the original task objective and low physics penalty.

A.2. *Mechanism synthesis continuous optimization*

In our continuous mechanism optimization (whether we are optimizing entire hands or individual fingers), we apply a three step annealing schedule (shown in Table A.2) in order to deal with errant collisions such that our gradient updates do not become unstable. Our annealing schedule initially assigns a low penalty for collisions, and gradually increases it to the full value to generate a mechanism that tracks trajectories as well as possible while avoiding collision.

Additionally, to check for collisions between keyframes, we evaluate collision costs for intermediate poses in which the object position and orientation are splined between keyframes. We fix the position and orientation of the base to the trajectory specified by the user: this prevents accomplishing the motion by trivially reorienting the base while holding a static grasp.

A.3. *Mechanism synthesis discrete optimization*

On a given iteration of our outer loop, we typically optimize 10 different randomly seeded fingers, generate five recombined hands, and limit ourselves to a maximum of

Table A.2. Weights for the annealing schedule: we gradually increase the trade-off between end effector tracking and collision to encourage stable gradients.

Variable	Step 1	Step 2	Step 3
w_{eeTarget}	50	10	50
$w_{\text{collision}}$	0.1	5	100

Table A.3. Common weights for mechanism synthesis continuous optimization: note that w_{eeTarget} and $w_{\text{collision}}$ vary with the annealing schedule.

w_{eeTarget}	Varies	$w_{\text{collision}}$	Varies
$w_{\text{fingerLengthRegularization}}$	0.1	$w_{\text{contactDistSurface}}$	1000
$w_{\text{fingertipAcceleration}}$	0.001	w_{jacNull}	1
w_{torque}	0.05	$w_{\text{jointLimits}}$	1
$w_{\text{fingerPositions}}$	1	$w_{\text{fingertipMinLength}}$	1

Table A.4. Table of weights for the whole hand optimization: Weights for terms not mentioned above are kept the same as in the floating contact optimization.

$w_{\text{ci\_finger}}$	100	$w_{\text{ci\_finger\_slippage}}$	10
$w_{\text{ci\_object}}$	100	$w_{\text{ci\_object\_slippage}}$	10
$w_{\text{fingerAcceleration}}$	0.001	$w_{\text{collision}}$	100
w_{task}	50	w_{physics}	10
w_{jacNull}	1.0	w_{torque}	0.05
$w_{\text{frictionCone}}$	0.1	$w_{\text{frictionConeHand}}$	0.1
$\alpha_{\text{(friction sharpen)}}$	5	$w_{\text{kinematic}}$	1

three joints per finger. In generating our initial seeds for the fingers, we randomly reseed the finger pose, finger segment lengths, the initial joint angles, joint axes, and the points at which the fingers connect to the base of the hand as well as the contact points on the fingertips (expressed in the local coordinate frame of the fingertip). We propagate the same random initial pose for the finger across all of the keyframes in our optimization.

A.4. "*Whole hand*" *optimization*

A table of common weights is shown in Table A.4. Prior to optimization, we reinitialize each contact location to be the closest point on the object to its assigned fingertip. By default, the user specified base trajectory is held fixed, however in various examples where it is appropriate, we allow the "whole hand" step of our optimization to adjust the base motion. This helps to avoid collision with the ground in several cases.

Appendix B. Leave-One-Out Examples and Failure Cases

In this appendix, we present the figures referred to in Secs. 6 and 7 of the paper, in which we discussed the effects of removing each non-trivial term from our pipeline and discussed several limitations of our existing framework.

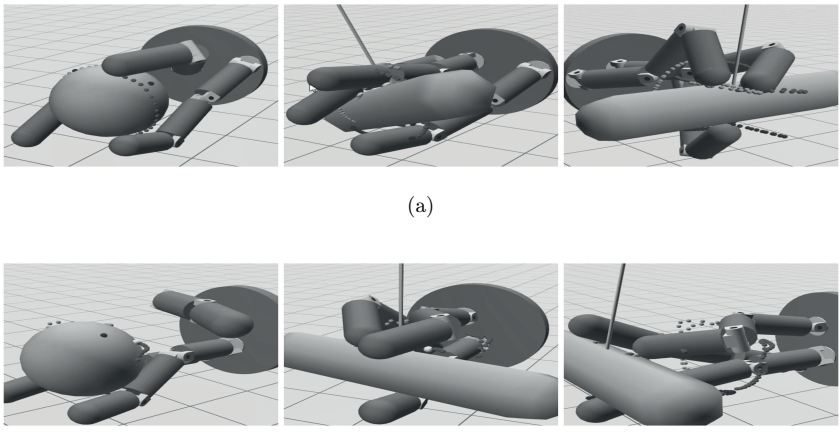


Fig. B.1. (a) shows the multiobjective sphere rotation, capsule rotation, and translate motion presented earlier after going through our normal pipeline. In (b), we run the same example through but without using perturbing forces in our multiobjective floating optimization. As a result, contacts that are not strictly necessary are lifted in the floating optimization to increase the coupling similarity resulting in unnecessarily brittle motions.

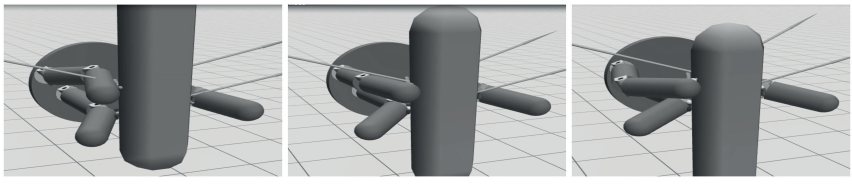


Fig. B.2. The above example shows one of the motions comprising the capsule translation multi-objective example presented in Fig. 6 with the Jacobian null space penalty L_{jacNull} removed from the pipeline. With this constraint removed, the synthesis step is content with finding a hand capable of tracking the end effector trajectories from our floating step for each component motion. As a result, the "whole hand" optimization (from which L_{jacNull} is also removed) produces a motion that allows the object to unrealistically float up and down with no way for the hand to actuate the motion.

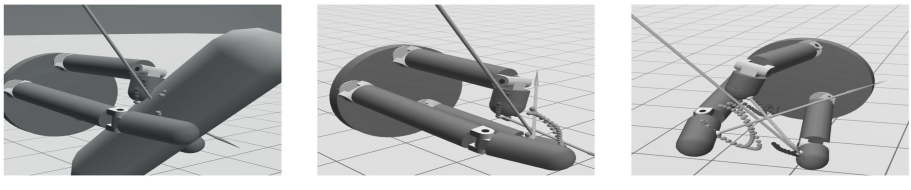
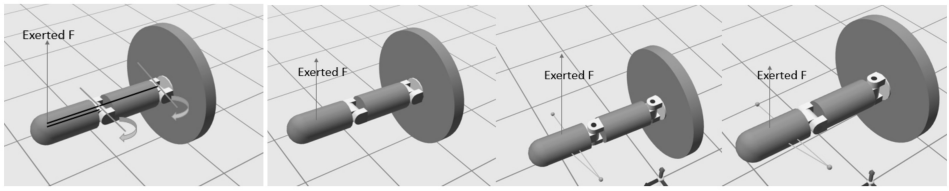
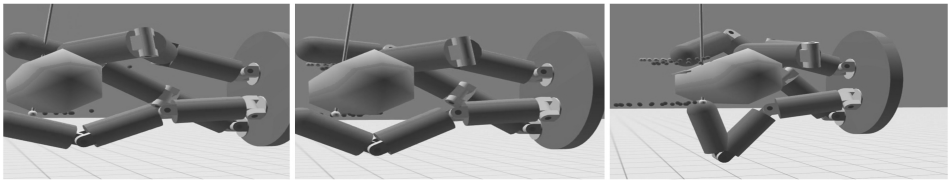


Fig. B.3. As Figs. (a) and (b) show (on our rotate and bow out single objective motion), with $L_{\text{frictionConeHand}}$ removed from our whole hand optimization, even slight collision between the fingertip and object can cause very unrealistic contact forces with respect to the fingertip's frame. Figure (c) shows the same optimization with $L_{\text{frictionConeHand}}$ included, indicating that it prevents this type of inconsistency from occurring.

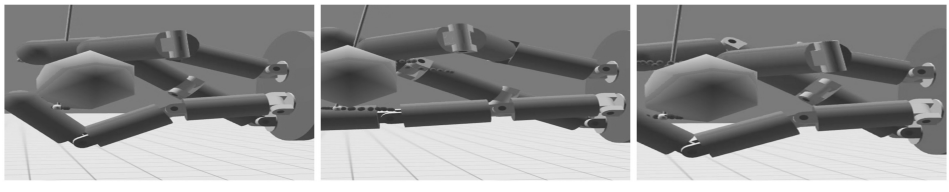


(a) (b) (c) (d)

Fig. B.4. In (a), we have a scenario in which a single finger in the synthesis stage must exert an upward force originating from the center of the fingertip. In this example, we disabled all parameters except joint axis directions and provided a manually constructed initial design. In (b), we fit our hand to a stationary contact trajectory: the optimization aligns both axes perpendicular to the exerted force to balance the torque load on the joints. In (c) and (d), we fit our finger to a motion in the plane perpendicular to the force, requiring one joint for actuation and one for movement. In (c), we set our distal joint to be slightly off axis from the direction of the exerted force while the first joint is set parallel to the force: this configuration incurs a very high torque penalty and no Jacobian null space penalty. (d) shows the global optimum design that our optimization finds in which the distal axis is perpendicular to the force while the base joint is parallel to it allowing the hand to actuate with minimal torque and match the desired trajectory. If $L_{\text{torqueReg}}$ were excluded from the synthesis step, (c) and (d) would have identical objective values.



(a)



(b)

Fig. B.5. To demonstrate the effectiveness of the slipping constraints $L_{\text{ci\_finger\_slippage}}$ and $L_{\text{ci\_object\_slippage}}$ in our "whole hand" optimization we build on the example shown in part (b) of Fig. B.1. The bottom contact in the translation motion (reproduced above as (a)) for this particular example displays a very large slippage error. The contact point essentially teleports to another location on the fingertip since there is only a small contact surface available. For demonstration purposes, in (b) we increased the normal weighting (in Appendix A) of both slippage terms by one order of magnitude causing the optimization to find an acceptable solution with a much lower slippage penalty.

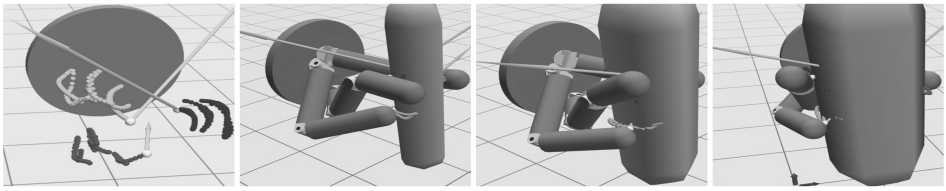
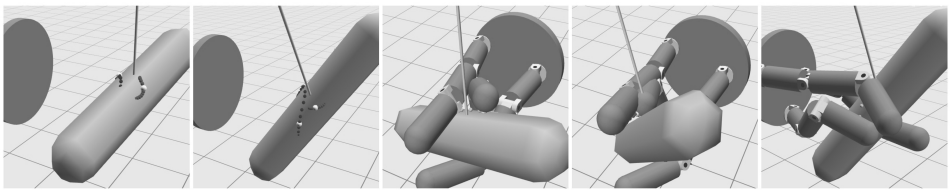


Fig. B.6. In this example, we optimized a hand for the task of rolling capsules of various sizes around their principle axes. Since we are requiring a single hand design to accomplish these motions, this is only possible if the pipeline plans for or learns the rolling behavior. Our pipeline is able to plan reasonable contact trajectories, but the "whole hand" optimization is unable to adjust the contact positions to generate the necessary rolling motion since the mechanism is ill-suited for this type of motion. This is due to the fact that our mechanism was constructed with the premise that a single contact point on the fingertip should be responsible for tracking each contact trajectory. We refer the reader to our accompanying video for the complete motions.



(a) (b) (c) (d) (e)

Fig. B.7. (a) Displays the floating contacts for the acceptable sphere rotation, capsule rotation and translation motion example, while (b) displays a different version of this motion obtained via a different set of initial contact points. The contacts in (a) do not cause the attached fingers to self-collide, while it is impossible to construct fingers to follow the trajectories in (b) without self collision. (c) displays the non-colliding example, while (d) displays the colliding example after the "whole hand" optimization stage. The "whole hand" planner was unable to find an alternate solution in (d) with the mechanism given to it that it did not involve finger collision, and as a result (e) the solution it found involves the fingers passing through one another.

References

1. I. M. Bullock and A. M. Dollar, Classifying human manipulation behavior, in *2011 IEEE Int. Conf. Rehabilitation Robotics (ICORR)* (IEEE, 2011), pp. 1–6.
2. S. R. Buss, Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods, *IEEE J. Robot. Autom.* **17**(1–19) (2004) 16.
3. M. Ceccarelli, G. Carbone and E. Ottaviano, An optimization problem approach for designing both serial and parallel manipulators, in *Proc. Int. Sym. Multibody Systems and Mechatronics (MUSME 2005)*, pp. 6–9.
4. M. Ceccarelli and C. Lanni, A multi-objective optimum design of general 3R manipulators for prescribed workspace limits, *Mechanism Mach. Theory* **39**(2) (2004) 119–132.
5. J. Hollerbach, M. Mason and H. Christensen, A roadmap for us robotics: From internet to robotics, in *Computing Community Consortium (CCC)*, Tech. Rep. (2009).
6. J.-F. Collard, P. Fisette and P. Duysinx, Contribution to the optimization of closed-loop multibody systems: Application to parallel manipulators, *Multibody Syst. Dyn.* **13**(1) (2005) 69–84.

7. R. Deimel and O. Brock, A novel type of compliant and underactuated robotic hand for dexterous grasping, *Int. J. Robot. Res.* **35**(1–3) (2016) 161–185.
8. A. M. Dollar and R. D. Howe, A robust compliant grasper via shape deposition manufacturing, *IEEE/ASME Trans. Mechatronics* **11**(2) (2006) 154–161.
9. A. M. Dollar and R. D. Howe, Simple, robust autonomous grasping in unstructured environments, *2007 IEEE Int. Conf. Robotics and Automation* (IEEE, 2007), pp. 4693–4700.
10. A. M. Dollar and R. D. Howe, Towards grasping in unstructured environments: Grasper compliance and configuration optimization, *Adv. Robot.* **19**(5) (2005) 523–543.
11. F. Sebastian Grassia, Practical parameterization of rotations using the exponential map, *J. Graph. Tools* **3**(3) (1998) 29–48.
12. C. Hazard, N. Pollard and S. Coros, Automated design of manipulators for in-hand tasks, thesis, Carnegie Mellon University (2018), pp. 1–8.
13. J. M. Hollerbach, Workshop on the design and control of dexterous hands, Technical report, Massachusetts Inst of Technology Cambridge Artificial Intelligence Lab (1982).
14. C. K. Liu, Dextrous manipulation from a grasping pose, *ACM Trans. Graph.* **28**(3) (2009) 59.
15. C. K. Liu, Synthesis of interactive hand manipulation, in *Proc. 2008 ACM SIGGRAPH/ Eurographics Symp. Computer Animation* (Eurographics Association, 2008), pp. 163–171.
16. D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Program.* **45**(1) (1989) 503–528.
17. H. Liu *et al.*, Multisensory five-finger dexterous hand: The DLR/HIT hand II, in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2008 (IROS 2008)* (IEEE, 2008), pp. 3692–3697.
18. C. S. Lovchik and M. A. Diftler, The robonaut hand: A dexterous robot hand for space, in *Proc. 1999 IEEE Int. Conf. Robotics and Automation*, Vol. 2 (IEEE, 1999), pp. 907–912.
19. R. R. Ma and A. M. Dollar, On dexterity and dexterous manipulation, in *2011 15th Int. Conf. Advanced Robotics (ICAR)* (IEEE, 2011), pp. 1–7.
20. R. R. Ma, L. U. Odhner and A. M. Dollar, A modular, open-source 3D printed underactuated hand, in *2013 IEEE Int. Conf. Robotics and Automation* (IEEE, 2013), pp. 2737–2743.
21. I. Mordatch, Z. Popović and E. Todorov, Contact-invariant optimization for hand manipulation, in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation* (Eurographics Association, 2012), pp. 137–144.
22. I. Mordatch, E. Todorov and Z. Popović, Discovery of complex behaviors through contact-invariant optimization, *ACM Trans. Graph.* **31**(4) (2012) 43.
23. T. Mouri *et al.*, Anthropomorphic robot hand: Gifu hand III, in *Proc. Int. Conf. ICCAS* (2002), pp. 1288–1293.
24. Y. C. Nakamura *et al.*, The complexities of grasping in the wild, in *2017 IEEE-RAS 17th Int. Conf. Humanoid Robotics (Humanoids)* (IEEE, 2017), pp. 233–240.
25. A. M. Okamura, N. Smaby and M. R. Cutkosky, An overview of dexterous manipulation, in *Proc. IEEE Int. Conf. (ICRA'00) Robotics and Automation, 2000*, Vol. 1 (IEEE, 2000), pp. 255–262.
26. M. Posa and R. Tedrake, Direct trajectory optimization of rigid body dynamical systems through contact, in *Algorithmic Foundations of Robotics X* (Springer, 2013), pp. 527–542.
27. F. Rothling *et al.*, Platform portable anthropomorphic grasping with the bielefeld 20-dof shadow and 9-dof tum hand, in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2007 (IROS 2007)* (IEEE, 2007), pp. 2951–2956.
28. F. Sanfilippo *et al.*, Efficient modular grasping: An iterative approach, in *Proc. IEEE Int. Conf. of Biomedical Robotics and Biomechatronics (BioRob)* (IEEE, 2012), pp. 1281–1286.

29. B. Thomaszewski *et al.*, Computational design of linkage-based characters, *ACM Trans. Graph.* **33**(4) (2014) 64.
30. P. Wright, J. Demmel and M. Nagurka, The dexterity of manufacturing hands, *Robot. Res.* **14** (1989) 157–163.
31. Y. Ye and C. K. Liu, Synthesis of detailed hand manipulations using contact sampling, *ACM Trans. Graph. (TOG)* **31**(4) (2012) 41.
32. C. Hazard, N. Pollard and S. Coros, Automated design of manipulators for in-hand tasks, *18th Int. Conf. Humanoid Robots* (2018), pp. 1–8, doi:10.1109.HUMANOIDS.2018.8624932

Christopher Hazard received his M.S. in Robotics from Carnegie Mellon University in 2018. He is currently a Robotics Engineer at the National Robotics Engineering Center (NREC), which is owned by CMU. His research interests include machine learning, robotic manipulation, and motion planning.

Nancy Pollard is a Professor in the Robotics Institute and the Computer Science Department at Carnegie Mellon University. She received her Ph.D. in Electrical Engineering and Computer Science from the MIT Artificial Intelligence Laboratory in 1994, where she performed research on Grasp Planning for Articulated Robot Hands. Before joining CMU, Nancy was an Assistant Professor and part of the Computer Graphics Group at Brown University. She received the NSF CAREER award in 2001 for research on 'Quantifying Humanlike Enveloping Grasps' and the Okawa Research Grant in 2006 for "Studies of Dexterity for Computer Graphics and Robotics."

Stelian Coros is an Assistant Professor in the Department of Computer Science at ETH Zurich, where he leads the Computational Robotics Lab (CRL). He is also an Adjunct Professor in the Robotics Institute at Carnegie Mellon University. He received his Ph.D. in Computer Science from the University of British Columbia in 2011. His work bridges the fields of Visual Computing, Robotics and Computational Fabrication.