Automated Design of Manipulators For In-Hand Tasks

by

Christopher Hazard

CMU-RI-TR-18-36

Submitted in partial fulfillment of the requirements for the degree of Masters in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

July 2018

Abstract

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 goal is to ultimately design a program that is able to automatically design robotic hands that can perform a set of target tasks by leveraging their physical design to encourage robust manipulations. 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 mechanical hand designs optimized to perform specified in-hand manipulation tasks of varying difficulty. We also briefly explore the feasibility of constructing multi-purpose hands from scratch that are meant to perform multiple primitive tasks in sequence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Related Work

1.1 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. In the immediate future, dexterous manipulation has the potential to create more capable manipulators in factory settings with the benefit of increasing precision in fine tasks and improving the efficiency of factory work[1]. Dexterous manipulators are able to accomplish motions more efficiently and operate in limited workspace environments more easily [2]. However, achieving these types of manipulations remains a challenge for various reasons[3].

One line of research in dexterous manipulation focuses on the design of manipulators to mirror the kinematics of the human hand[4] [5]. These hands have shown impressive capabilities with regards to dexterous manipulation [6] tasks, however the problem of dexterous manipulation remains unsolved [7]. 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 to perform a manipulation 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.

Low DOF hands that are optimized for specific manipulation tasks tend to be more robust to these difficulties due to the fact that they rely less on high fidelity sensing, precise state estimation, or sophisticated control strategies[8][9]. By restricting ourselves to specific types of motions we intend to accomplish for a given hand, we can build in robustness to the mechanical design, thereby offloading some of the burden on designing control systems or working with expensive sensors. Simple hands are easier to build and maintain, easier to control, less expensive, and are less prone to mechanical failure since they have fewer moving parts.

We believe that an approach completely opposite to that of designing sophisticated anthropomorphic hands is a better route for solving dexterous manipulation problems in general. Our goal is to design a tool that can be used by non-expert users to design custom manipulators that are specifically built to accomplish a set of high level tasks specified by the user. This tool should have the ability to design optimal hands given minimal task specification that are as mechanically simple as possible, inexpensive, and relatively easy to build via 3d printing technology [10][11]. A user-friendly tool of this sort would allow novice users with little or no mechanical background to easily design a variety of hands for their intended use cases with the additional benefit that these hands would have robustness built in to their mechanical design, meaning that specialized control policies would not have to be designed to accommodate the use of these hands, nor would they require high fidelity sensing.

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 [10][12][13]. In several cases, optimization has been used to tune some of the design parameters for these types of hands [9]. We go beyond 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 breaks this mechanism synthesis problem into a sequence of self contained optimization procedures. Our system 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. Our system functions end-to-end to produce workable designs given the user's initial input, and can be used in an interactive manner to allow the user to explore the space of suitable designs for their desired task. 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 readily extensible with regards to making it more capable of performing other classes of in-hand manipulations. The extensibility of our optimization program is due to its modular nature: each component is self contained and reliably produces an output that is useful to the next component, thus we can extend the capabilities of our system via the inclusion of additional self-contained design and planning modules to produce a more sophisticated design tool. We discuss several additional modules that may be included in future work to further expand our design capabilities.

1.2 Related Work

1.2.1 Anthropomorphic Hands vs. Simple Manipulators

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 [14] and [15] 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 [16] with the intention of mimicking these motions on robotic hands. These types of hierarchies are useful not only for grouping together observed human manipulations into classes of manipulations that might be implemented by similar control strategies, but may also be used to select from a list of specialized grippers in a factory environment, as in [15]. These types of hierarchies are also relevant in developing grasp synthesis algorithms [17] [18] for robotic hands.

A common viewpoint with respect to classifying and modelling individual manipulation behaviors is that if the robotics community is able to adequately define a set of motion primitives, these primitives can be implemented on a single anthropomorphic hand that can chain simple manipulations together [19] to accomplish

sophisticated tasks. Much effort has thus been spent in developing increasingly sophisticated anthropomorphic hands to more closely resemble the capabilities of the human hand. Platforms such as the Utah-MIT hand [20], NASA Robonaut hand [4], DLR-Hand [21], GIFU III [22], ACT hand [23], and Shadow Dexterous [5], among others, have become standard models on which manipulation algorithms and controllers have been implemented and tested. These hands are meant to be generic manipulators that should be able to carry out virtually any manipulation task given an appropriate control policy. To allow for feedback control, these hands are built with a wide array of force sensing capabilities. This means that these hands can be somewhat difficult to manufacture, often have many moving parts that can be targets of mechanical failure, and are relatively expensive when compared to simple, less-capable manipulators like parallel jaw grippers.

For the most part, the designs of the anthropomorphic testbed manipulators for dexterous manipulation research remain fixed regardless of their intended use. This is understandable due to the complexity and cost of these manipulators; it is only practical to design a one-size fits all manipulator. One exception to this is the work of [24], which briefly delves into the optimization of several continuous design parameters (e.g. pulley radii at joints, tendon stiffness) with respect to the Stanford-JPL hand to specialize its physical capabilities for performing things like manipulative (precision) grasps, power grasps, and optimizing for working volume of the grasp.

A competing viewpoint [8] is that relatively simple manipulators are better for individual tasks for which they will be used repetitively: therefore robotic manipulation research should focus on developing as many types of specialized simple manipulators as necessary. In practice, simple manipulators are more commonly used since they are more straightforward to control, are less likely to encounter mechanical failures, and tend to be more robust for the tasks for which they are designed. This of course comes at the cost of having less capability to accomplish a wide range of motions, meaning that a robot could possibly require multiple specialized manipulators in order to be general purpose.

Robots that are specifically designed to meet manipulation task requirements have been designed for many applications. One notable application is agriculture, in which manipulators are designed to pick specific fruits like cucumbers [25], eggplants [26], kiwis [27], and apples[28]. The advantage of these specialized fruit

pickers is that their design can be optimized through repeated physical simulation to make them particularly robust for their intended purpose.

With respect to grasping in particular, a well known grasp planner introduced in [29] applies principle components analysis to a range of grasps synthesized by human hands and robotic hands alike to create "eigengrasps". The authors use simulated annealing in combination with a simple grasp quality metric, evaluated via the GraspIt! simulator [30], to optimize for the coefficients of the top several eigengrasps in order to produce stable grasps for a wide variety of objects. A related experiment, described in [31] applies simulated annealing to simplify the morphology of a given high DOF hand design, and shows that effective DOF's can be drastically reduced while only experiencing moderate decline in grasp quality. These works showed that the majority of human generated grasps can be described in a low dimensional space, indicating that simplified manipulators could potentially rival the capabilities of anthropomorphic hands.

1.2.2 Design Parameter Optimization for Manipulators

Even if the discrete parameters of a manipulator are held fixed (such as number of joints, number of motors, etc.), continuous design parameters such as component lengths, tendon stiffness, pulley radii, etc. can still be adjusted to make specialized manipulators. Various works [32][33] have sought to optimize continuous parameters in parallel [34][35] and serial [36] manipulators to address kinematic concerns such as reachability constraints, avoidance of Jacobian singularities within the workspace, limits on individual joint torques, etc. The problem of addressing several of these objectives at once for a single manipulator design typically lends itself to methods in multi-objective optimization [33][37].

An interesting sub-domain of design parameter optimization is that of evolutionary design, in which evolutionary/genetic algorithms are applied to jointly optimize the morphology and control policies of robots for specific tasks [38][39]. Due to the high computational costs associated with evolutionary algorithms, the structures that we plan to evolve must be carefully designed and made amenable to the evolution process. This usually means that these methods are limited to rather simple control policies [40][41], with the main focus being the optimization of the robot design. An important benefit of evolutionary mechanism design is that one can build a library of evolved mechanisms to use as seeds for similar related

tasks, meaning that future mechanisms do not necessarily have to be designed from scratch. More recently, evolutionary optimization has been applied to the domain of soft robotics [42], which has its own set of unique design challenges.

1.2.3 Trajectory Optimization for Manipulation

Trajectory optimization (with a given, fixed hand design) has shown remarkable ability to synthesize complex motions in both robot locomotion and manipulation. Works such as [43] have sought to construct anatomically correct models of human hands in simulation to simulate hand pose control via simulation of muscle contractions. [44] and [45] develop 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 is able to construct physically feasible motion plans to synthesize the target manipulations. Treating manipulation synthesis as a trajectory optimization problem allows the user to create complicated motions from high level goals, and create controllers that are able to withstand unexpected external forces.

Recent work in trajectory optimization has explored the use of discontinuous contacts in locomotion and manipulation tasks [46][47]. Mordatch et. al. [47] introduced the concept of contact invariance, in which contact between two bodies, which is normally a binary variable, is treated as a continuous variable ranging from 0 to 1, in which 0 represents no contact and 1 represents full contact: intermediate values can be seen as an in-between phase in deciding whether or not two bodies should be in contact. Representing contact as a continuous variable allows contact variables to be jointly optimized alongside other continuous variables in the trajectory optimization with a gradient based optimizer. Our work draws inspiration from the contact invariant method introduced in [48], which applies the contact invariant method to the domain of manipulation. In this work, contacts between the hand and object are treated as point contacts and hand dynamics are not explicitly accounted for, as motions are assumed to be sufficiently slow to allow for this approximation.

Other work in trajectory optimization for manipulation involves the usage of motion capture data in conjunction with simulation methods to capture and synthesize human-like motion [49][50][51]. These methods supplement motion capture

data in which humans demonstrate target manipulations and supplement the captured motions with physics-based simulation or optimization to construct feasible motions.

1.2.4 Interactive and User Driven Mechanical Design

Gradient-based optimization techniques have been applied to the problem of mechanism synthesis for linkage based characters to design mechanical automata for a variety of purposes such toy design [52][53], design of walking robots [54], and for use in real-life linkage mechanisms such as those found in robot manipulators. Works like [55] and [56] strongly emphasize user interaction and create tools through which a non-expert user can more easily design linkages that satisfy their design requirements. The result is that users are able to provide high level input with regards to their needs, while the machine takes care of handling kinematic constraints and designing the pieces of the mechanism. Our work shares a similar goal in that we wish to ultimately design a tool in which a user can input high level objectives for manipulations without having to worry about the particulars of the design process. Other examples of user driven mechanical design programs include interactive design of walking robots[57] and [58], in which the user specifies a desired direction in the space of design parameters to change the current design, while an optimizer applies the Implicit Function Theorem to re-balance the mechanical constraints.

1.2.5 Design of Compliant Manipulators and Mechanisms

Thus far, we have mainly discussed attempts on the algorithmic side to optimize for functional and robust mechanical design. Efforts to produce robust specialized manipulators on the mechanical engineering side have primarily focused on the design of compliant hand mechanisms and their physical fabrication [13][59]. Compliant manipulators[60] have the advantage of offsetting some of the burden on control and state estimation procedures since they are physically designed to function under environmental uncertainty.

Compliant mechanisms use elastic components to help the gripper grasp objects of uncertain size, pose, or geometry[9] without unexpected slippage occurring. These mechanisms naturally conform to the object they are attempting to grasp,

thus reducing the need for specialized control algorithms and/or the need for high fidelity sensing. Manufacturing techniques have been developed to fabricate compliant hands with embedded sensors[61], so as not to preclude the use of feedback in designing control policies. Under-actuated compliant mechanisms [62][10] have the further advantage of using fewer motors by coupling elastic joints such they are controlled by the same motor: this leads to the fabrication of simpler, lower-cost, and more easily controllable hands that are robust to environmental uncertainty. The recent work of [63] applies the concept of interactive linkage design for rigid bodies to the design of compliant mechanisms, using simulation and design optimization to create more complicated but expressive mechanisms.

Chapter 2

Optimization Pipeline

2.1 Introduction

We propose a solution to the problem of optimizing specialized hand topology designs for individually specified manipulation tasks via a pipeline of simpler optimizations meant to build up a motion plan and a corresponding manipulator in steps. Our pipeline consists of 3 main components: initial motion plan generation with floating contact points, mechanism synthesis, and "whole hand" motion planning in which the motion plan is further adapted to the designed hand. The initial input to our system consists of an object along with several points placed on the object representing contact point locations. These "floating" contact points are each representative of an individual contact with a fingertip in a future hand design, although at this step these points are not bound to a particular hand design. A physically feasible motion plan is then generated in terms of contact forces and locations to move the object between a sequence of specified goal poses (and fulfill any other specified task requirements). In the second step of our pipeline, a mechanism is constructed and optimized to fit the end effector positions corresponding to the floating contacts as well as possible. In the third step, differences between the initial motion plan and mechanism synthesis are reconciled to generate a physically realistic manipulation plan.

2.2 Optimization Pipeline Description

In this section we provide a full description of each segment of the optimization pipeline, including each of the objective terms and their significance. We also provide a list of hyper-parameter values for each optimization that we have found to have worked well across our experiments.

2.2.1 Floating Contact Optimization

Let

$$\mathbf{S_t} = [\mathbf{x_O}\ \mathbf{f_j}\ \mathbf{r_j}\ c_j] \tag{2.1}$$

be the state at time t of the object, with $\mathbf{x_O}$ denoting the object's location and orientation in the world frame (location is defined w.r.t. the center or mass of the object), and $\dot{\mathbf{x}}_\mathbf{O}$ being the derivative at time t of position and orientation. $\mathbf{f_j}$ denotes the force vector at contact point j for $j \in \{1, 2, ..., N_{contacts}\}$ expressed w.r.t. 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 [47] that is constrained to lie in the interval [0,1]. The contact invariant term can be interpreted as a fuzzy set membership value that dictates whether or not a given contact point is active at time t. The force term is scaled linearly by c_j, so if $c_j = 0$ the contact is inactive and exerts no force, whereas the full force $\mathbf{f_j}$ is applied if $c_j = 1$ on the object at the contact location $\mathbf{r_j}$. The presence of the contact invariant term allows fingers to break contact with the object and re-establish contact later on, thereby turning a discontinuous constraint into a continuous one.

In the floating contact optimization step, the user specifies a set of initial contacts on the object prior to optimization, with each contact point representing contact with a different fingertip in our eventual hand mechanism. At this step in the optimization, we have no concept of a hand kinematic design: each floating contact point is simply a disembodied point able to exert force on the object subject to friction cone constraints.

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

$$\mathbf{S} = \underset{\mathbf{S}}{\text{argmin}}\ \Sigma_t \Sigma_i\ w_i * L_i(t) \tag{2.2}$$

$$\text{s.t. } c_j \in [0, 1]\ for\ 0 \leq t \leq T \tag{2.3}$$

where each cost L_i pertains to the individual cost terms listed below

When calculating the above summation, we interpolate between the keyframes and calculate the sum at each time according to a time step t_{step}. Depending on the number of steps involved, we set our motions to last between 4 and 12 seconds for most of our example manipulations (more complex manipulations require longer time horizons for smooth motions), and we evaluate our objective function every $t_{step} = .1$ seconds by splining between the keyframes for our motion. 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. 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[64], and $\dot{\mathbf{x}}_\mathbf{O}$ are calculated via finite differences. The contact forces $\mathbf{f_j}$ are linearly interpolated and the $\mathbf{c_j}$ are evaluated as piecewise-constant terms.

The individual loss terms $L_i(t)$ are defined as follows:

- Physics terms:

$$L_{physics}(t) = L_{linMom}(t) + L_{angMom}(t) \tag{2.4}$$

$$L_{angMom}(t) = ||\Sigma_i \; c_i(t) * (r_i \times f_{i,local}) - (\omega \times (I^{local}_{object}\omega) + I\dot{\omega})||^2 \tag{2.5}$$

$$L_{linMom}(t) = \sum_i c_i(t) f_i - m\ddot{x} \tag{2.6}$$

  where I_{object} is the moment of inertia of the object in its own local frame, ω is angular velocity, and m is the object mass

$$L_{forceReg}(t) = \sum_i ||c_i(t) f_i||^2 \tag{2.7}$$

$$L_{frictionCone}(t) = \sum_i c_i * \exp(\alpha(||f_{i,local} - n * (f_i \cdot n_i)|| - \mu f_i \cdot n_i)) \tag{2.8}$$

  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 but are still valid

- Task objectives:

$$L_{task} = \frac{1}{k}\sum_{k} ||p(k) - p_{goal}(k)||^2 + quatdist(o(k), o_{goal}(k))^2 \tag{2.9}$$

where k is the set of keyframes for which we define goal poses p_k and o_k, which represent desired object positions and orientations respectively. In the above equation, the function quatdist(q_1,q_2) refers to quaternion distance formula, which is essentially the angle required to rotate from one frame of reference to the other. Task objectives can be specified several ways depending on the desired behavior we want to see from our system. In our experiments, we have limited the scope of our objectives to achieving intended goal poses for our object. Other goals, such as tracing out a desired path with an end effector point on the manipulated object, can easily be translated into objective function terms.

- Contact Invariant Costs:

$$L_{ci\_object}(t) = \sum_{i} ||F_i||c_i||r_{proj} - r_i||^2 \tag{2.10}$$

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

$$projErr(i,t) = r_{i,proj}(t) - r_i(t) \tag{2.11}$$

$$L_{ci\_object\_slippage}(t) = \sum_{i} ||c_i F_i||^2 * ||(projErr(i,t) - projErr(i,t - t_{step}))/t_{step}||^2 \tag{2.12}$$

- Additional regularization terms:

$$L_{floatingContactAccel}(t) = \sum_{i} ||((r_i(t + t_{step}) - 2 * r_i(t) + r_i(t - t_{step}))/t_{step}^2||^2 \tag{2.13}$$

$$L_{accelerationRegularization}(t) = \sum_{i} \ddot{x}^2 \tag{2.14}$$

where x is object position

$$L_{angularAccelerationRegularization}(t) = \sum_{i} ((\omega \times (I_{world}\omega) + I_{world}\dot{\omega})/t_{step})^2 \tag{2.15}$$

Except for the L_{task} term above, we normalize each of the terms mentioned above by the number of keyframes in our motion. Given a reasonable starting grasp pose for our floating contact points, and a sequence of object goal poses, we automatically construct keyframes for the object and contacts in which the object's position and orientation are smoothly interpolated between objective poses and contact positions are held constant with respect to the object's local frame. This gives the optimization a good initial seed with respect to the motion that it is expected to produce while requiring minimal input on behalf of the user. Note that this initialization will give 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 rather than try to find a similar yet more easily accomplished motion that readily compromises our task objective. The result is that we tend to see an optimization procedure in which the L_{task} cost remains small in comparison to other terms, while the $L_{physics}$ term dominates the optimization, eventually leading to a solution with only slight deviation from the original task objective and an acceptably low physics penalty, indicating that we have found a feasible motion.

We optimize our objective with a standard L-BFGS solver [65]. The motions output by a first pass through this optimization typically have contact invariant values c_i that are between 0 and 1, and typically cluster around higher values (above .7) and low values (.3 and below), indicating the importance of the contact point in the optimization. After this first pass, we run an additional optimization, called the "defuzzification step", in order to set each c_i to either 0 or 1, so that future steps in our pipeline may treat contact as a given binary value (that is held fixed from hereon out). To accomplish this defuzzification, threshold our c_i values with a threshold of either .1, .2, or .3, setting every c_i value above this threshold to 1 (indicating active contact) and every value below it to 0 (indicating that the contact is lifted). Holding these binarized c_i values fixed, we re-optimize our motion using the previous optimization result as an initial seed. We select our threshold by simply testing each of our candidate thresholds and select the threshold that yields the best optimization objective.

As an additional objective in various test cases, we jointly optimize our motion plan with another motion in which we aim to accomplish the same task objective, but under the presence of specified perturbing forces and/or torques about the object's center of mass. This type of additional objective is meant to create more robust

$w_{physics}$	10	w_{ci_object}	100
w_{task}	50	$w_{ci\_object\_slippage}$	100
$w_{forceReg}$	.01	$w_{floatingContactAccel}$	.01
$w_{objectAccelRegularization}$	.1	$w_{objectAngAccelRegularization}$	1
$w_{frictionCone}$	.1	$\alpha(frictionsharpen)$	5

TABLE 2.1: Table of common weights for floating contact optimization

grasping policies for circumstances in which the user believes additional perturbing forces may be present. These perturbing forces are specified by the user prior to the optimization and can take place at any time and for any duration during the motion. To accommodate this objective (i.e. optimize a contact placement scheme that can provide the required forces both with and without the presence of external perturbations), we jointly optimize for an additional scenario in which contact forces are allowed to be different in each keyframe than in our original motion plan, but contact invariant (c_i) and contact position terms are shared between the motion plans. At the cost of adding additional variables to our optimization (thereby increasing computational costs), the result of this additional objective is often improved contact placement with regards to stable grasps and higher c_i values (indicating greater use of contacts that might otherwise be largely unused in the optimization). See the Results section for further details.

2.2.2 Mechanism Synthesis Continuous Optimization

After generating an initial motion plan with the trajectory optimization described above, we next design a mechanism with fingertips that are roughly able to follow the generated contact point trajectories. This step involves both the optimization of discrete structures as well as the optimization of continuous parameters governing the design of the generated mechanism. Below we describe the continuous optimization, which is used by the discrete optimization procedure described in the next section. The optimization below assumes that we have all discrete parameters (i.e. the kinematic structure) fixed, and we are optimizing for continuous design variables.

We wish to find a set of morphological parameters $\mathbf{M} = \{\mathbf{L}\ \mathbf{A}\ \mathbf{B}\}$, a set of joint angle poses $\mathbf{Q} = \{\mathbf{Q_1}, \mathbf{Q_2}, ..., \mathbf{Q}_{N_{keyframes}}\}$, and a set of contact points $\mathbf{P}$ on the

constructed fingertips. **L**, **A**, and **B** respectively represent the finger segment lengths, joint axis orientations, and positioning of fingers on the base/palm of the hand. We select our contact points as variables in the optimization. Additionally, we make use of the fact that each of the contact invariant terms c_i is known for each fingertip i for each keyframe: these terms tell us whether or not a given contact is active. We restrict our contact points p such that active contact points must remain fixed with respect to the fingertip's local coordinates for any string of keyframes where that contact point is active. If that contact becomes inactive, we are allowed to change the position of that contact point with respect to the fingertip for the next string of frames in which the contact is active. We do this in order to promote the selection of stable contact regions such that our final motion involves as little slipping with respect to the fingertip as possible.

We optimize the parameters such that:

$$\mathbf{M} = \underset{\mathbf{M,Q,P}}{\operatorname{argmin}} \ \Sigma_k \Sigma_i \ w_i * L_i(k) \tag{2.16}$$

$$\text{for } k \in \{1, 2, ..., N_{keyframes}\} \tag{2.17}$$

Where L_i represent the cost terms listed below:

- Contact point costs:

$$L_{eeTarget}(k) = \sum_i c_i * ||p_i - p_{target}||^2 \tag{2.18}$$

$$L_{fingerContactDistSurface}(k) = \sum_i ||p_{proj} - p_i||^2 \tag{2.19}$$

  where c_i represents the defuzzed contact invariant term (either 0 or 1) given by the floating contact optimization above for fingertip i at keyframe k. $L_{eeTarget}$ is the distance between the contact point p_i on fingertip i and the corresponding end effector point on the given trajectory for that contact. $L_{fingerContactDistSurface}$ 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 relatively high coefficient to force contact points to lie on the surfaces of the fingertips.

- Collision

  For collision penalty calculations, we make use a piecewise cubic interpolation spline that smoothly interpolates (to the second order) between

the functions $f(x) = 0$ for $x < 0$ and $f(x) = x^2$ for $x > 0$ as follows:

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

$$L_{collision}(k) = \sum_{i,j \in bodies} g(penetration(body_i, body_j)) \tag{2.20}$$

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 (we use $\epsilon = 10^{-6}$)

- Finger length costs:

$$L_{fingerLengthRegularization} = \sum_i (l_i)^2 \tag{2.21}$$

where i ranges over all the capsules present in the hand and l_i denotes the length of the principle axis of capsule i.

$$L_{fingerMinLengthCost} = \sum_i g(l_{min} - l_i) \tag{2.22}$$

where g denotes the piecewise cubic spline defined above. We include the minimum finger length cost since it prevents our optimization from finding unrealistically small finger length capsules.

- Controllability related costs:
  Our synthesized mechanisms are required to fulfill two main objectives, the first being the ability to track the end effector points generated by the floating contact policy from the first step in our optimization pipeline, and the second being the ability to supply the necessary forces to accomplish the target motion. To accomplish the later, we introduce two new terms to penalize the component of the force that lies along the null space of our mechanism and to regularize the torque necessary to provide the motion. In this section we simply list these terms as $L_{jacNull}$ and L_{torque}. See Appendix A for a more in depth analysis of these terms as well as their derivation.

$$L_{jacNull} = \sum_i c_i * \sqrt{\sum_k (f \cdot e_k)^2} \tag{2.23}$$

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 (so that we don't penalize inactive contacts).

$$L_{torque} = \sum_i \|\vec{\alpha}\|^2 \tag{2.24}$$

where is the vector of torque magnitudes that must be supplied by the mechanism to actively provide the desired force (see Appendix A for the derivation)

- Additional costs:

$$L_{fingerPositions} = \sum_i ||proj_{base}(b_i) - b_i||)^2 \tag{2.25}$$

where i ranges over all the bases of the fingers and we find the closest projected point onto the base: we assign a rather stiff penalty to this term to ensure that the fingers are always connected to the surface of the palm. This term can be applied to a variety of base shapes, as long as a smooth projection formula exists for the surface representing the base. In this work we focus on flat palms (flat bases) since they are the most conventional type of palm to work with and the projection function onto a disc in a plane is sufficiently smooth for our optimization. More generally, base shapes could consist of a collection of spheres and/or capsule objects as these individual primitives have convenient projection formulae.

$$L_{fingerAcceleration}(k) = \sum_i (1 - c_i) * \ddot{x_i}^2 \tag{2.26}$$

where $\ddot{x_i}$ is the second derivative (estimated via finite differences) of the position of the center of mass of fingertip i and c_i is the contact invariant term for that fingertip in frame k. Note that the finger acceleration penalty only applies to fingertip contacts that are inactive in order to encourage smooth transit when repositioning a finger on the object.

$$L_{jointLimits}(k) = \sum_{i \in keyframes} \sum_{a \in joints} g(a(i) - a_{max}) + g(a_{min} - a(i)) \tag{2.27}$$

In our implementation, the terms a_{max} and a_{min} are constants set to $\pi/2$ and $-\pi/2$ respectively to limit the range of motion of each of our joints. We select these particular values due to the fact that these are the maximum affordances allowable for the joints in our synthesized mechanisms such finger segments wont collide with their parent and child segments. This term also tends to steer our mechanisms away from degenerate configurations.

Additionally, to check for collisions between keyframes, we evaluate collision costs for intermediate poses in which the object position and orientation are catmull-rom splined and orientations represented via the exponential map are catmull-rom splined between frames, and joint angles are linearly interpolated. For our test motions, we have fixed the position and orientation of the base to remain the same for a given manipulation to prevent base movement from trivially reorienting the object into its desired pose with no finger movement beyond finding a suitable grasp. We that we make use of multiple randomly generated initial conditions for optimization as well as an annealing schedule in which we run the optimization multiple times with increasingly large collision penalties. See the "Results and Discussion" section below for a thorough treatment of this.

$w_{eeTarget}$	varies	$w_{collision}$	varies
$w_{fingerLengthRegularization}$	.1	$w_{fingerContactDistSurface}$	1000
$w_{fingertipAcceleration}$	.001	$w_{jacNull}$	1
w_{torque}	.05	$w_{jointLimits}$	1
$w_{fingerPositions}$	1	$w_{fingertipMinLength}$	1

TABLE 2.2: Table of common weights for mechanism synthesis continuous optimization: note that $w_{eeTarget}$ and $w_{collision}$ vary with the annealing schedule

2.2.3 Mechanism Synthesis Discrete Optimization

The process by which we optimize the discrete structure of our hand designs is rather simple: we treat fingers independently and keep adding additional finger segments (and joints) to each of our fingers until they satisfy their individual objectives well enough (as determined by a pre-defined threshold) or until we reach an upper limit on the number of joints we are willing to allocate per finger. After

designing individual fingers for each contact point trajectory, we take combinations of these fingers to produce completed hands.

For a given discrete structure, the continuous parameter optimization tends to be highly non-convex and thus susceptible to many undesirable local minima. To combat this, we must use multiple randomly initialized seeds to optimize our hand designs. This introduces a computational bottleneck, which we address by optimizing for each of our fingers separately and then recombining our top performing fingers at the end. We are able to do this since we treat each of our finger joints as being independently controlled, thus we don't have to worry about coupling between joints on separate fingers. In each iteration, we optimize each of the fingers whose combined $L_{eeTarget}$ and $L_{jacNull}$ scores from the previous iteration lie above a predefined threshold (user defined) separately according to the continuous optimization described above.

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). During each of our recombination trials, we apply the same annealing schedule to discourage unstable gradient updates caused by collision penalties. 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 typically set MAX_FINGER_TRIALS to 10, $MAX\_RECOMBINATIONS$ to 5, and MAX_JOINTS to 3. The call to optimizeContinuousParameters(finger) is a continuous optimization in which we optimize a single finger ignoring the presence of any other fingers on the hand. The call to optimizeContinuousParameters(hand) then uses the fingers selected by the recombination function as initial conditions and corrects for any self collision in the robot hand. 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 initialize object poses

Algorithm 1 Discrete Structure Optimization

```
1: procedure OptimizeDiscreteStructure
2:     fingers = ∅, ∅, ... , ∅
3:     allFingersSaturated = False
4:     while ¬ allFingersSaturated do
5:         for all fingerSet ∈ fingers do
6:             if ¬ finger.saturated then
7:                 addJointToFinger(finger)
8:                 for i=0; i < MAX_FINGER_TRIALS; i++ do
9:                     finger = reseed(finger)
10:                    for annealingParams ∈ annealingSchedule do
11:                        optimizeContinuousParameters(finger, annealingParams)
12:                    end for
13:                    fingerSet ← finger
14:                end for
15:            else
16:                finger.saturated = True
17:            end if
18:        end for
19:
20:        hands = ∅
21:        for trial=0; trial < MAX_RECOMBINATIONS; trial++ do
22:            recombinedHand = recombineFingers(trial, fingers)
23:            for annealingParams ∈ annealingSchedule do
24:                optimizeContinuousParameters(recombinedHand,annealingParams)
25:            end for
26:            hands ← recombinedHand
27:        end for
28:        bestHand = select hand from hands with minimum hand.objective
29:        allFingersSaturated = True
30:        for all finger ∈ bestHand.fingers do
31:            finger.saturated = finger.numJoints == MAX_JOINTS ∨
32:                finger.eeTarget + finger.jacNull < SCORE_THRESHOLD
33:            allFingersSaturated = allFingersSaturated ∧ finger.saturated
34:        end for
35:    end while
36: end procedure
37:
38: procedure recombineFingers(trial, fingers)
39:    hand = ∅
40:    for all fingerSet ∈ fingers do
41:        finger.score = finger.eeTarget + finger.jacNull
42:        if trial == 0 then
43:            finger = select finger from fingerSet with minimum finger.score
44:        else
45:            finger = sample finger from fingerSet ∝ 1/(finger.score)
46:        end if
47:        hand ← finger
48:    end for
49:    return hand
50: end procedure
```

with the poses calculated from the floating optimization, and propagate the same random initial pose for the finger across all of the keyframes in this continuous optimization.

At the end of our recombination step we select the best hand that we have optimized so far (in terms of objective score) and determine whether we are done adding joints to each of our fingers. As mentioned in the above section, we use an annealing schedule for each of our continuous optimizations in which we gradually increase the cost for collision to help explore the space of available motions.

Variable	Step 1	Step 2	Step 3
$w_{eeTarget}$	50	10	50
$w_{collision}$	.1	5	100

TABLE 2.3: Table of weights for the annealing schedule: the complete process involves 3 steps. The first step is mainly meant to establish a finger capable of tracking the end effector, while the second step relaxes the importance of end effector tracking and moderately increases the collision penalty so as not to induce unstable gradients in the optimization. The third step enforces the final set of weights we put on collision avoidance and end effector tracking

2.2.4 Whole Hand Optimization

As the final stage in our motion optimization pipeline, we take the generated motion plan for the object and the constructed hand mechanism and perform a trajectory optimization similar to the one used in the floating contact optimization (step 1 of the pipeline). Along with the terms already present in step 1, we introduce several additional terms to the objective function to facilitate the inclusion of the hand mechanism. These additional optimization terms are meant to create a valid trajectory optimization procedure in which the hand constructed in the previous step establishes appropriate contacts with the objects. The result of this step is a smooth motion plan in which the motion plan is better adapted to the specified robot hand design.

We also add the joint angles at each keyframe to the list of variables that we intend to optimize and we do not restrict contact points to be stationary with respect to the fingertips as we did in the synthesis step (thus allowing us to perform rolling and slipping motions in this step, similar to the floating contact optimization).

The result of this step is a physically realistic motion plan involving both the hand and object. Note that in this optimization (referred to as the "Whole Hand Optimization" or step 3), we do not optimize for the morphology of the hand any further.

We initialize this optimization with the object poses determined by the floating contact optimization (step 1) and the hand design and poses determined by the synthesis optimization (step 2). Additionally, we use the contact forces calculated in step 1, but we reinitialize each contact location to be the closest point on the object to its assigned fingertip. This provides us with a better seed than the previously calculated contact positions, which are hand-agnostic.

For the purposes of this work, we assume that each joint in the hand is independently controlled at each time step (i.e. linkage dependencies do not exist between joints), however this limitation can easily be addressed with the inclusion of explicit linkage constraints that reduce degrees of freedom (see the section Limitations and Future Work). 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:

- Finger contact invariant terms: L_{ci_finger} and $L_{ci\_finger\_slippage}$
  These are simply the contact invariant terms introduced in the floating contact optimization, but applied to the finger instead of the object (although we still keep the terms applied to the object in this optimization). These costs have the same form as L_{ci_object} and $L_{ci\_object\_slippage}$, introduced in the floating contact optimization.

- $L_{kinematic}$: this term enforces joint limit constraints on each of our joints. By default we set joint limits of $-\pi/2$ and $\pi/2$ for each joint and penalize joint angle positions that fall outside this range by the squared error.

- $L_{fingerAcceleration}$: $L_{fingerAcceleration}(k) = \sum_i \ddot{x}_i{}^2$ taken from the synthesis step encourages smooth finger transitions

- $L_{Collision}$: collision penalties are identical to those in the synthesis step

- $L_{jacNull}$ and L_{torque}: these are the same terms introduced in the synthesis step of our optimization pipeline, reproduced below:

$$L_{jacNull} = \sum_i c_i * \sqrt{\sum_k (f \cdot e_k)^2} \tag{2.28}$$

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

$$L_{torque} = \sum_i \|\vec{\alpha}\|^2 \tag{2.29}$$

where α is the vector of torque magnitudes supplied at the joints

- $L_{frictionConeHand}$: Ideally our fingertips will be perfectly tangent to the object they make contact with by the end of the optimization, however there is always some small error present. Rather than over-constrain our optimization to enforce that fingertips remain perfectly tangent to the object, we introduce an additional friction cone term that measures how well the applied force fits into the friction cone with respect to the fingertip normal at the contact point.

$$L_{frictionConeHand}(t) = \sum_i c_i * \exp(\alpha(||f_{i,local} - n * (f_i \cdot n_i)|| - \mu f_i \cdot n_i)) \tag{2.30}$$

where n is the local surface normal (in the fingertip's local frame), μ is the coefficient of friction, and α is a sharpening factor for the exponential cost: the values for μ and α are kept the same as in the term for friction with respect to the object (which is also included in this step of the optimization pipeline)

w_{ci_finger}	100	$w_{ci\_finger\_slippage}$	10
w_{ci_object}	100	$w_{ci\_object\_slippage}$	10
$w_{fingerAcceleration}$	.001	$w_{collision}$	100
w_{task}	50	$w_{physics}$	10
$w_{jacNull}$	1.0	w_{torque}	.05
$w_{frictionCone}$	.1	$w_{frictionConeHand}$	.1
$\alpha(frictionsharpen)$	5	$w_{kinematic}$	1

TABLE 2.4: Table of weights for the whole hand optimization: weights for terms not mentioned above are kept the same as in the floating contact optimization.

Chapter 3

Results and Future Work

3.1 Results

We have demonstrated the ability of our pipeline to generate feasible mechanisms on a variety of in-hand manipulation tasks involving translation and rotation of the manipulated object with respect to the palm primarily using the motion of individual fingers. Our set of example motions ranges from simple translations and rotations to more complex multi-step motions as well as examples of motions that may be of interest in a factory setting. Videos for the individual example motions presented in the paper can be found at https://github.com/chazard/hand-videos.

Our work is primarily concerned with in-hand manipulation tasks. We require the user to set a preliminary base motion for each of our examples (usually this means the base is held fixed at a given point and orientation). In appropriate examples, we allow our optimizer to adjust the motion of the base in the third ("whole hand") step of the pipeline.

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 (e.g. when the target object is lying on a table). Whether or not these contacts are active is determined by the floating contact optimization in the first step of the pipeline, namely the "de-fuzzification" step, in which we threshold the contact invariant terms to binary values for use in later steps. Additionally, our current implementation requires that the simple objects being manipulated be composed

of capsule and sphere primitives due to the fact that these surfaces lend themselves to smooth/differentiable projection and collision formulas used by our gradient-based optimization. We also represent the underlying collision detection primitives of our fingers as capsules with a fixed radius, and the flat palm of our hand as a cylinder with a predefined height and radius.

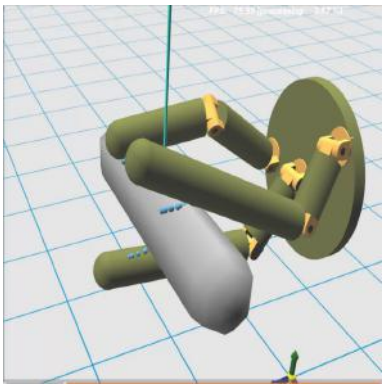
Figure 3.1 demonstrates several examples of simple manipulations that we have used to evaluate our pipeline. Figures 3.2 and 3.3 demonstrate several motions that may be of interest in a real-world setting such as a factory assembly line or on a mobile robot. Figure 3.4 demonstrates two different drawing motions, in which the regular position-based task objective is replaced with an objective that attempts to track the shown objective points with a specific end effector point located on the bottom of the "pen" object it is holding: the result is a motion plan and a mechanism that draws lines between the end effector targets on the ground plane.

We briefly touch upon the topic of designing hands meant for multiple tasks in Figure **??**, in which we demonstrate two sequences in which we build up progressively more complex motions from a set of simple primitive motions. From these examples we can get a sense of how our optimization procedure handles the need for additional degrees of freedom, as well as what combinations of motion primitives necessitate the inclusion of additional DOF's.

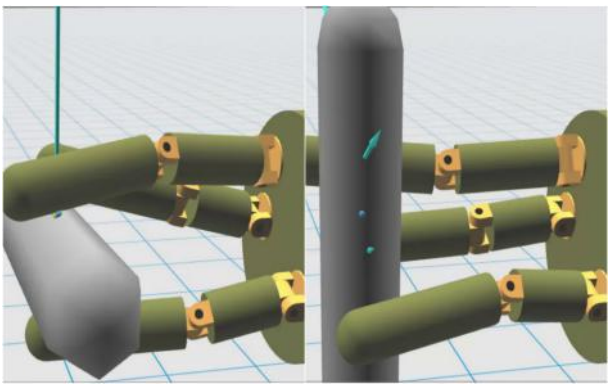
Our optimization program is able to discover quirks in our motions that lead to non-trivial mechanical designs. For example, in Figure 3.1(d), our optimization was able to suggest a mechanism in which we use one of our upper fingers to push out our capsule to bow it out while rotating it 90 degrees perpendicular to our palm surface, using the other two fingers to pivot the object. Our mechanism originally bows out the mechanism beyond the 45 degree target, then slides the pushing finger upward and reduces the force it exerts to achieve the desired position. In Figure 3.4(a), 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 Figure 3.7(b) we dictate that our mechanism is to rotate the sphere 180 degrees 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 3 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.

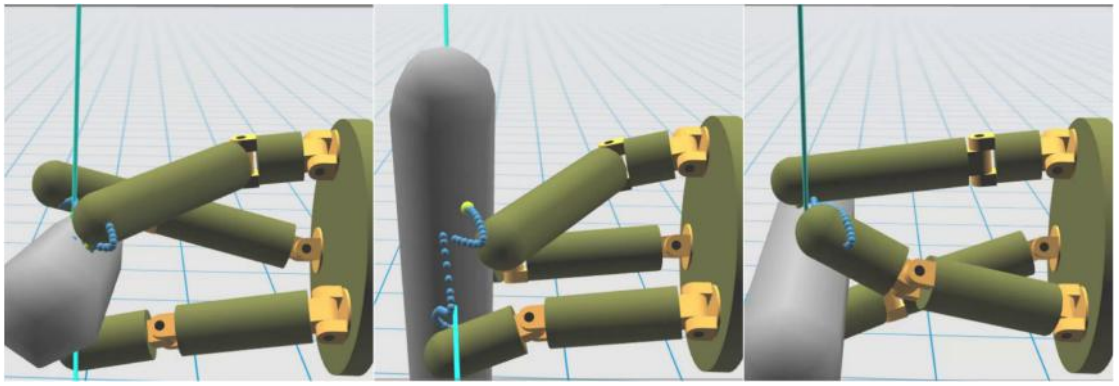
We are often able to create distinctly different mechanisms for the same motion simply by varying the initial contacts placed on the object or by varying the initial position of the base. We demonstrate this in Figures **??** and **??** in which we place our contacts in the same positions but placed 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.



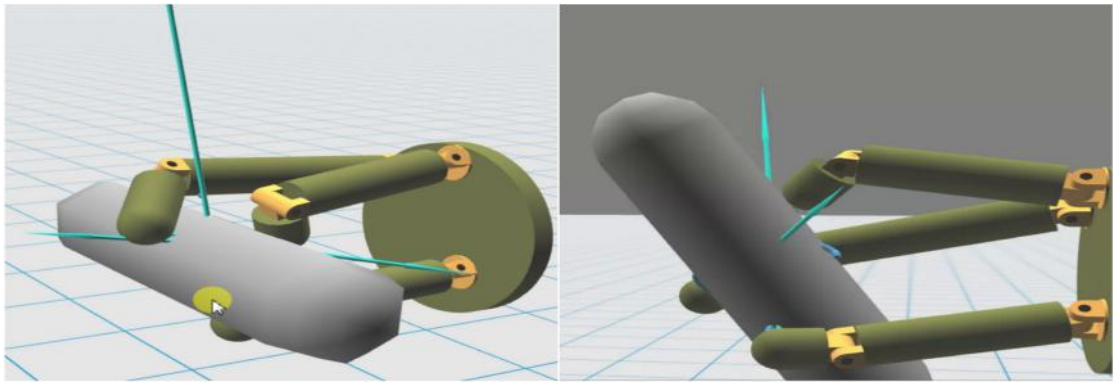
(a) Horizontal Translation



(b) Vertical Rotation

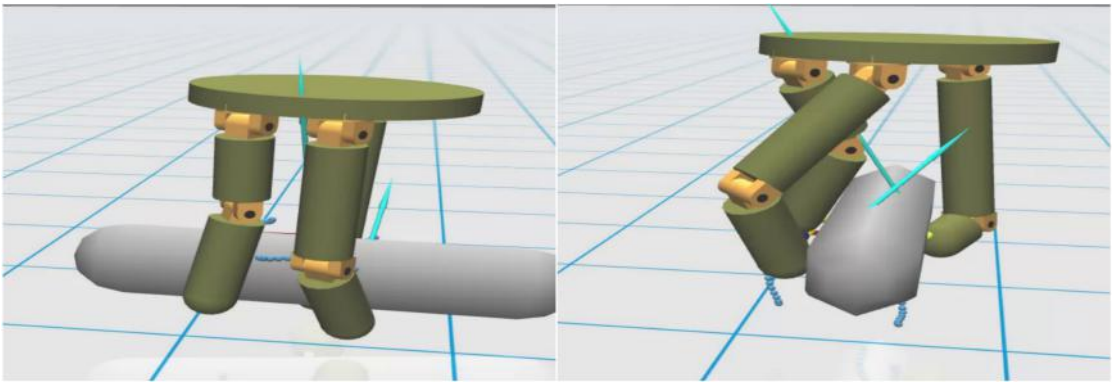


(c) 180 Rotation

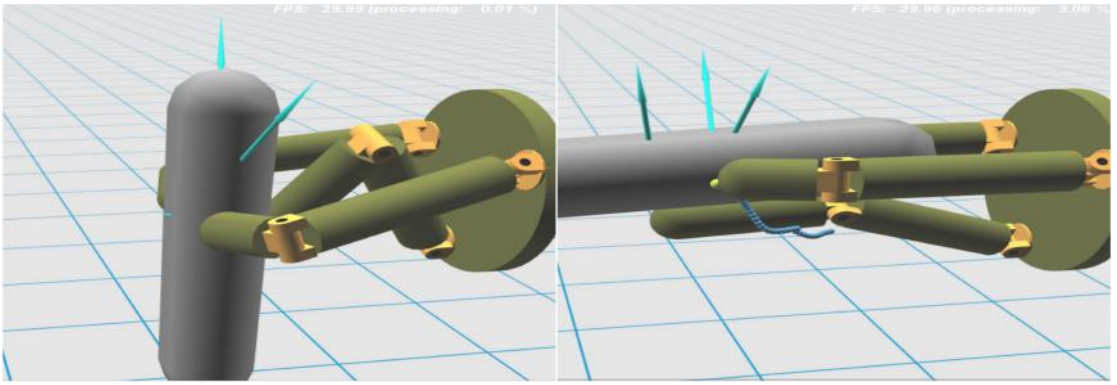


(d) Rotate and Bow Out

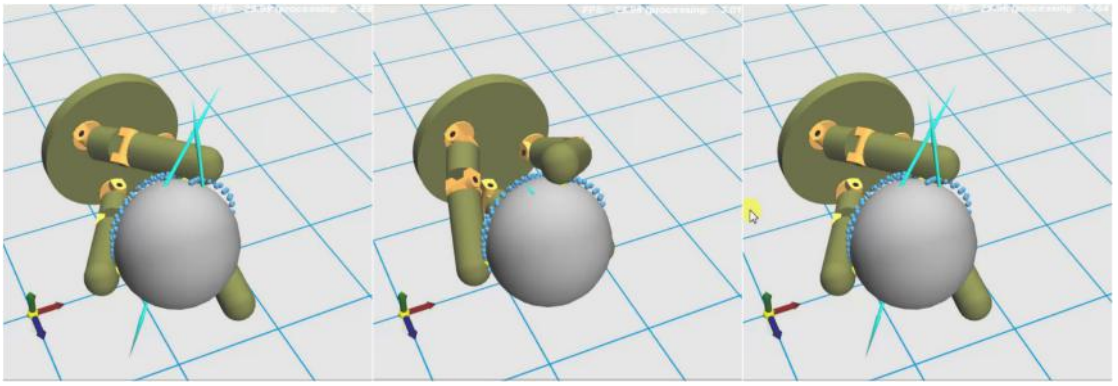
FIGURE 3.1:
(a) This is a simple translation motion inward and outward from the palm.
(b) A vertical rotation motion.
(c) 180 degree rotation, where the top finger moves from one side of the capsule to the other. This is a rare case in which the floating contact optimization breaks contact in a meaningful way and re-positions it.
(d) This rotation motion moves the object vertically (as in the vertical rotation motion) and bows out the object using the uppermost finger to push outward.



(a) Pick Up and Rotate

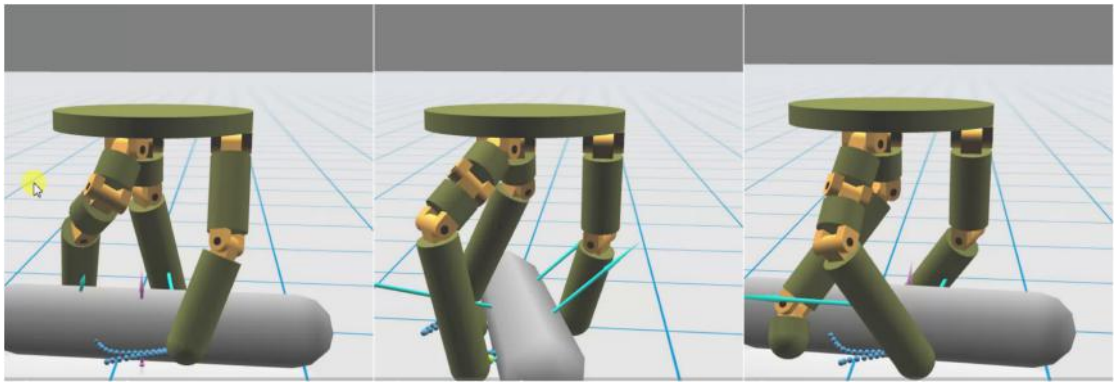


(b) Vertical Flip

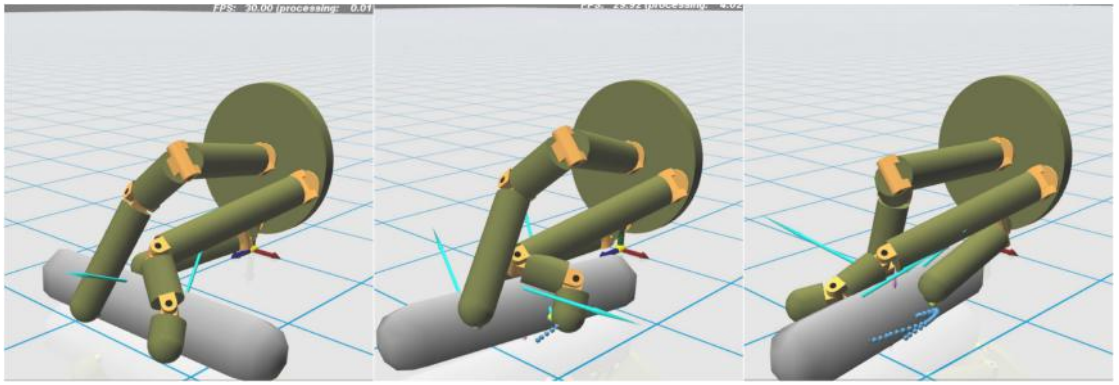


(c) Sphere Rotation

FIGURE 3.2:
(a) This is a pick and place type of motion in which an object is picked up from a table surface and then rotated and held parallel to the surface (as if to set it back down). In this motion we allow the optimizer in the "whole hand" step (step 3) to modify the base degrees of freedom (using the initially provided base motion as a seed), with a slight regularization on acceleration of the hand's base.
(b) This is a vertical flipping motion that could be potentially applied to a part feeding mechanism.
(c) Rotation of a sphere 180 degrees in either direction about an axis perpendicular to the palm. This type of motion could be used for an inspection sort of task.

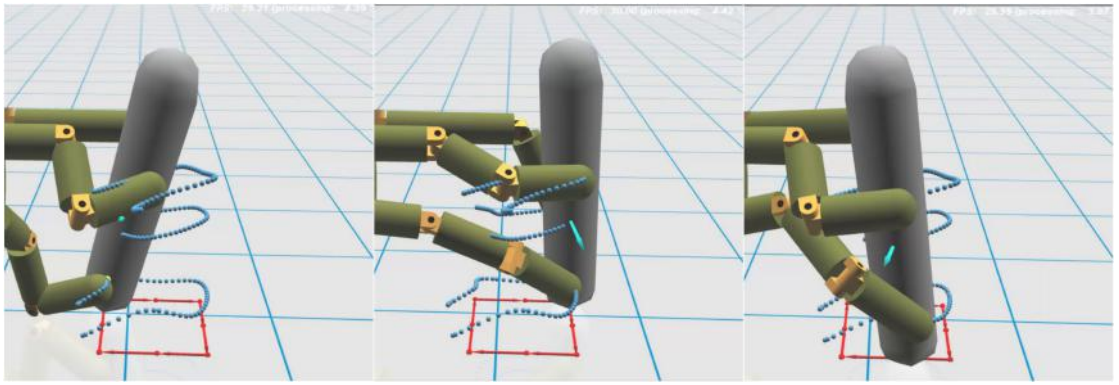


(a) Overhead Tabletop Rotation

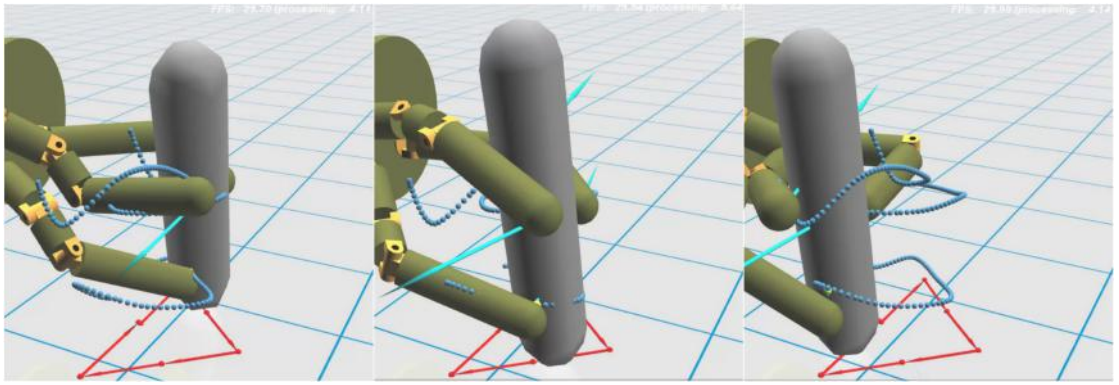


(b) Tabletop Rotation from the Side

FIGURE 3.3:
(a) The target object is rotated around while remaining level on the tabletop surface. The purple force drawn in this scenario represents the force supplied at the ground contact point, which is determined via the optimization.
(b) In this case, we accomplish the same motion, but with the base placed on the side of the object. The case is part (a) is a more natural base placement, while this example illustrates that our system is able to adapt to non-conventional base placements and create a feasible mechanism.

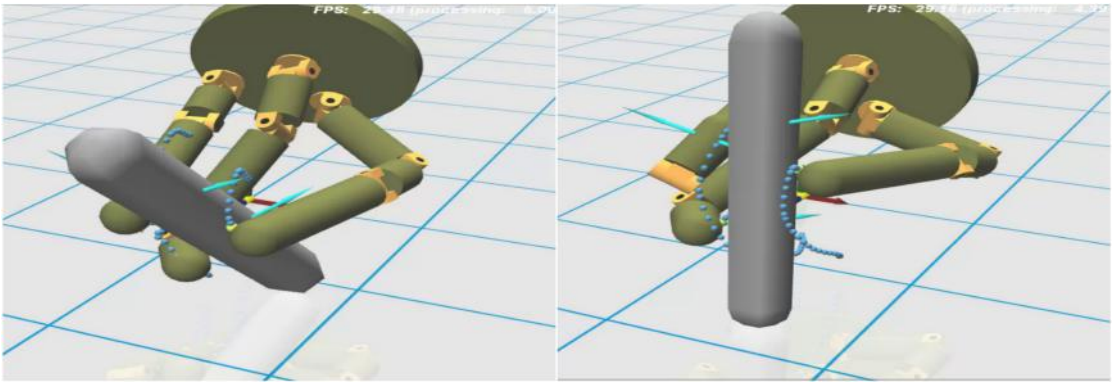


(a) Drawing a Box

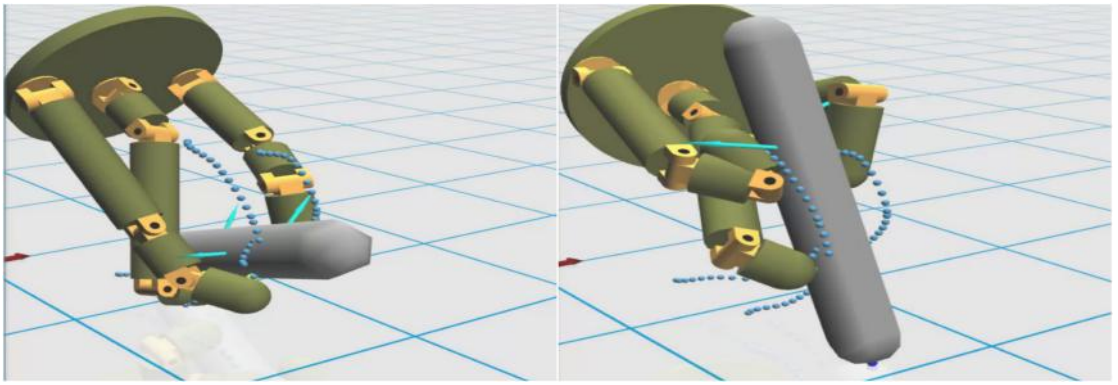


(b) Drawing a Triangle

FIGURE 3.4: In (a) and (b) we demonstrate the construction of mechanisms meant to draw shapes on the ground plane using a "pen" object. This pen object has a designated end effector point on its tip, and we use a different task objective function (instead of our usual position and orientation objective for the object) in which we replace the L_{task} term with the squared distance between the pen tip and the intended target points (in red) for the pen tip.

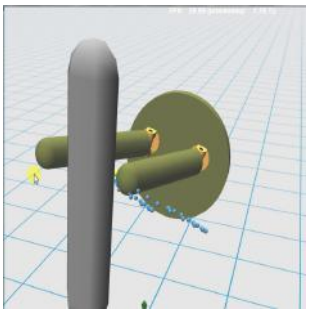


(a) Pencil Pickup Motion

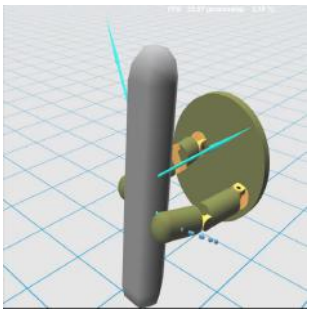


(b) Pencil Pickup Motion with Excessive Slippage

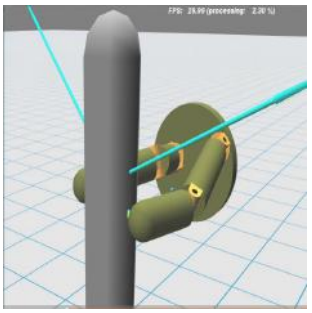
FIGURE 3.5: Each of the motions we have discussed thus far has displayed only minor slippage behavior primarily due to the fact that we regularize out most of the slippage by the time we have completed the "whole hand" optimization. However, in some cases in which slippage occurs at the very tips of the fingers, even if the actual distance slipped is relatively small, we can encounter some unrealistic slipping behavior. Examples (a) and (b) show a motion in which a pencil is picked up off of a table and put in a position as if to write with it. The design in example (a) shows an acceptable amount of slippage for reference, whereas example (b), although a similar design, has the contact point on its upper left finger migrate from the front of the finger to the back of the finger during the motion, which seems somewhat infeasible. In future work, explicit modelling of slippage and rolling constraints could resolve this issue.



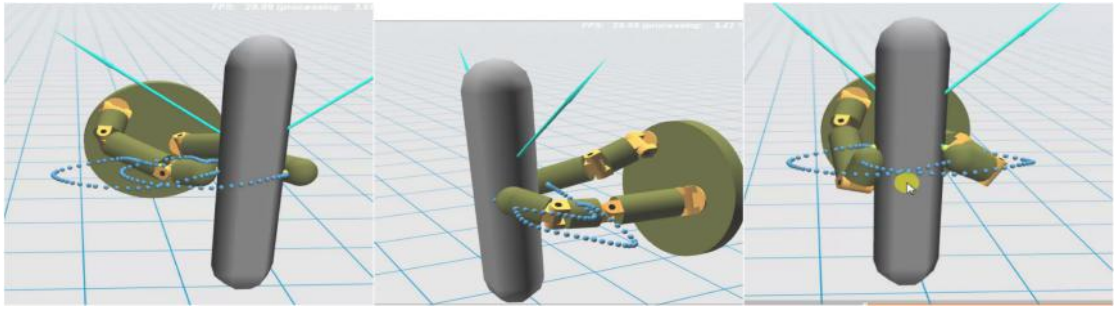
(a) Horizontal Movement (Gravity Disabled)



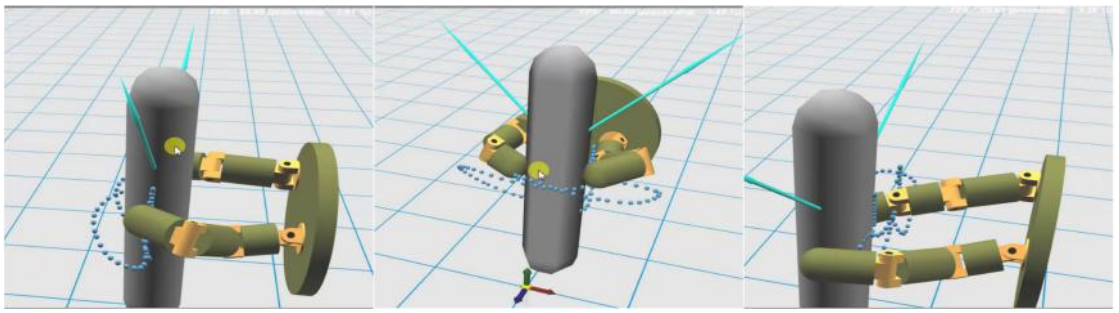
(b) Horizontal Movement (with Gravity)



(c) Outward Translate



(d) Circular Rotation



(e) Hemispherical Rotation

FIGURE 3.6: In this set of examples, we build up a 2 finger gripper by specifying an increasingly complex set of objectives, creating an increasingly complex set of floating motion plans. The optimization builds more capable versions of these manipulators as needed to follow the required trajectories and actively supply the necessary force. We can also view the later hand designs in this sequence as multi-objective hands, since they can accomplish multiple distinct primitive motions (which happen to be chained together).

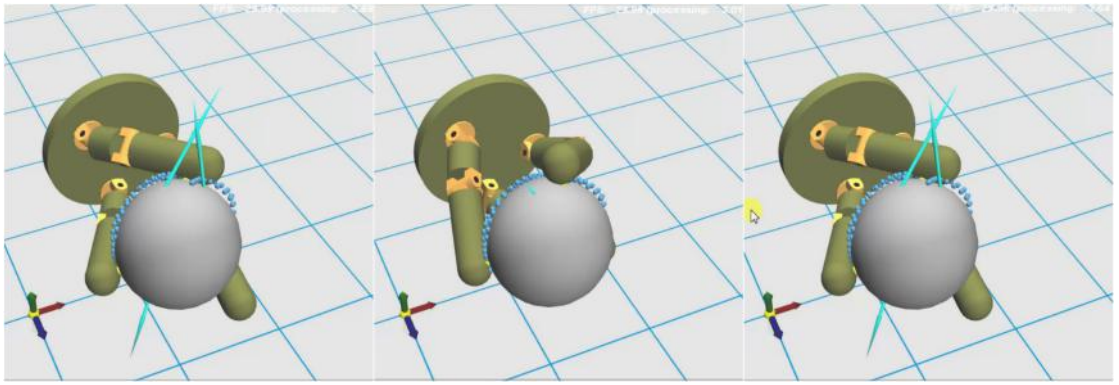
(a) This is a simple horizontal translation that can be achieved by a one link finger mechanism: we disable gravity since inclusion of gravity would necessitate the use of a second joint on each finger due to the fact that the fingers are required to actively supply the force applied to the object (as opposed to passively applying force to counteract gravity through static friction).

(b) Same motion as in (a), but with gravity enabled, causing our optimization to produce fingers with two links.

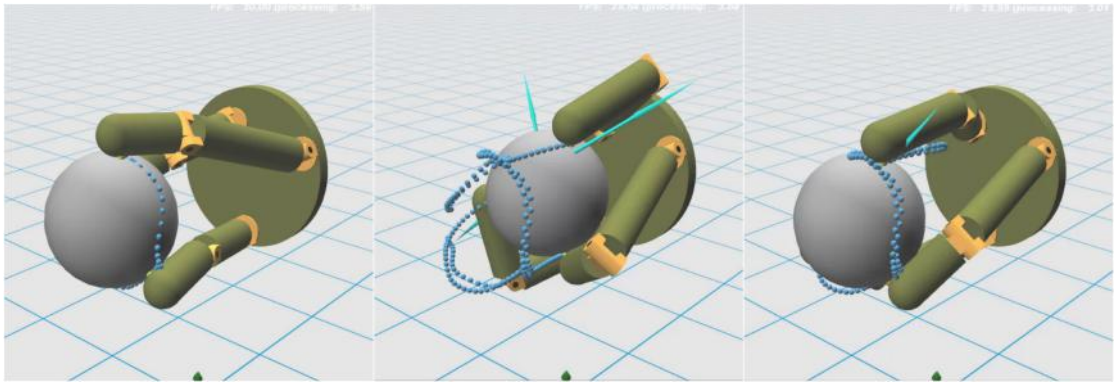
(c) This is a simple outward translation motion, and is one of the primitives we use in building up our larger motion.

(d) Rotation of the capsule in a circle parallel to the ground (no up and down motion). This combines the motions from (b) and (c).

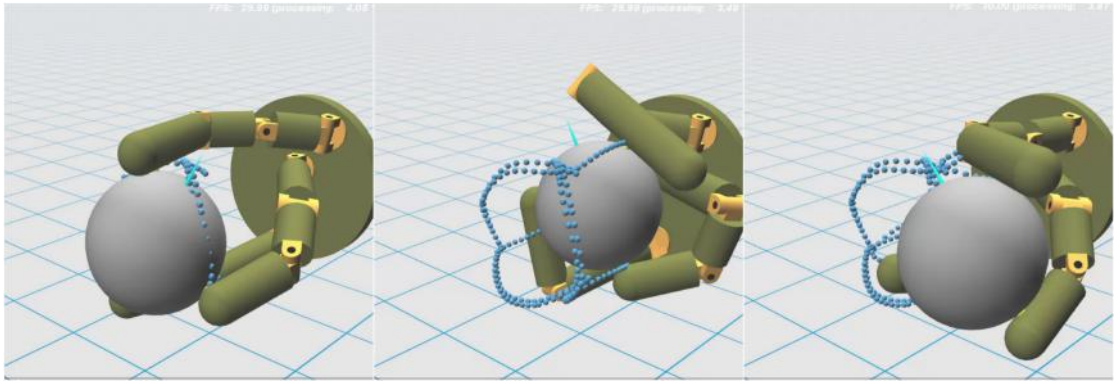
(e) A hemispherical rotation motion in which we add an up/down component to the motion in (d).



(a) Sphere Rotation



(b) Sphere Rotation and Translation



(c) Sphere Rotation with Bi-directional Translation

FIGURE 3.7: In a vein similar to that of Figure 3.6, here we build up a progressive sequence of motions based on the sphere rotation introduced in Figure 3.2.

(a) Is a reproduction of the previously introduced sphere rotation example.

(b) This is the sphere rotation motion with an additional translation toward the palm. Interestingly enough, this motion does not seem to require additional degrees of freedom, however the translation motion can only be achieved from the initial configuration of the manipulator.

(c) Finally we add an additional translation to translate the sphere from side to side, causing our system to add another link to each of our fingers.

3.2 Discussion

3.2.1 Significance of the Contact Invariant Term

In the floating contact optimization step, it is important to note the significance of the contact invariant (c_i) term. In most cases, contact is never broken and, if contacts need to be re-positioned to accomplish the motion, the optimizer tends to prefer a reasonable degree of sliding along the object while still actively providing force (which does not preclude the successful synthesis of a working mechanism). When contact is broken, it is due to the fact that the contact in question was under-utilized for a sequence of timesteps during the motion (i.e. it was not required to actively supply force for a significant period of time), as indicated by a low c_i value for the contact prior to "defuzzification". In other words, contacts are not broken due to any perceived kinematic constraints on the part of a would be hand (for the simple reason that the floating contact optimization has no concept of a hand in the first place). This, along with several other factors is responsible for the apparent disconnect between the floating contact and mechanism synthesis steps of our optimization (see Limitations and Future Work for a more extensive discussion). In most of our motions however, the contacts are adequately utilized throughout the optimization to warrant never breaking contact.

If we desired to force our optimized motion to break contact in some fashion to purposefully re-position the grasp on the object (in which case we would already have an idea of a hand design in mind), we could do so with this optimization by manually setting the contact invariant weights. However, in this paper we refrain from doing this since our goal is to be able to synthesize hand designs from scratch, and doing so would bias the design of our manipulators. As a result, we observe few motions in which floating contacts break naturally.

In the absence of perturbing forces, we have observed cases in which contacts have been ignored altogether (depending on the initial location at which the contact is placed) in our floating optimization due to the simple fact that these contacts weren't strictly necessary to accomplish the motion, as in the sphere translation part of the example shown in Figure 3.7(b). In this particular example, the contact on the top of the sphere is rendered inactive in the absence of external perturbations. When we specify a perturbing force (as an additional objective) in the same direction as the translation part of the motion for the duration of that segment

(i.e. a force perpendicular to the plane of the palm), we observe that the contact becomes active. This is due to the fact that this contact is necessary to hedge against unexpected acceleration (on a physical robot) when the object is moved toward the palm.

The addition of perturbing forces in the objective typically forces the unused contacts to function as supports against the perturbing force; in other words, they still do not provide active force in the unperturbed case but their c_i values increase well above the threshold value, rendering them active and making for a more robust manipulation.

3.2.2 Annealing Schedule in the Mechanism Synthesis Optimization

The mechanism synthesis step is usually very sensitive to initial conditions regarding regarding the initial hand morphology and the initial pose of the hand (which is propogated across all of the keyframes in the motion). It is essential that this step be run with multiple random seeds over the initial hand pose, finger segment lengths, joint axes, and positions on the palm in order to generate fingers that are able to both track the desired end effector positions and supply the required forces. We have not found any overt correspondence between initial guesses for hand topologies and objective motions, thus limiting us to selecting seeds randomly to overcome the many local minima associated with this optimization.

We have also found it necessary to include an annealing schedule in which we progressively increase the penalty for collision between the hand and object through multiple optimization iterations. This schedule is necessary for searching through the space of synthesized motions because it allows us to begin by exploring infeasible motions that involve a fair amount of object collision and progressively clean up our motions to make them acceptable with regards to object collision.

Use of this annealing schedule allows us to overcome many situations in which the optimization would normally get stuck, such as the common case in which fingers are trapped on the wrong side of the object and are unable to change positions mid-optimization because "phasing through" the object will incur large penalties due to prohibitive weights on object collision. Starting off with a weighting scheme in which collisions carry a minor cost, we are able to establish fingers that track

the end effector trajectories well. As we progressively increase the relative cost of collision, we see that the fingers are able to find a compromise between collision and end effector tracking while avoiding the unstable gradients that would result if we were to immediately enforce the full collision penalty after only the first step of the annealing schedule (hence the need for an intermediate annealing step). This results in a final motion plan with very little collision as well as minimal tracking and controllability errors.

As another example of how our annealing schedule improves optimization results, consider the case of a finger that must be lifted off one side of a vertically placed capsule object to move around to the opposite side of the object to place a contact as in Figure 3.1(c). Depending on the seed given to the optimization, it is likely that shortening the finger length immediately rather than changing the joint angles for that finger while in transit will be the fastest way to get to a local minimum if we were to use the full collision weighting right off the bat. Although reasonable, this sort of design decision can lead to a local minimum in which the finger length can not be extended to reach the intended contact location. Instead, the annealing schedule would first allow the finger to phase through the object, then gradually push the finger outside of the object while allowing it to generate a smooth transition from one target point to another.

3.2.3 Initial Contact Point Seeds in the Floating Optimization

It is important to note that the motion plans generated by the contact planner in the first step of the optimization pipeline are not bound to any given kinematic design. While this allows for non-biased exploration of mechanisms/motion plans, it does have the potential to generate motion plans that are infeasible with respect to any reasonable hand design or yield extremely brittle motion plans. For this reason, we currently require that the user input a set of reasonable (though not necessarily optimal) initial contact points on the object for the first frame only: these contact points are then propagated across all of the keyframes in the optimization such that they remain stationary in the local coordinate frame of the object being manipulated. In most cases, this gives the floating contact optimization a good seed from which it can optimize for the forces and contact positions needed to accomplish the task at hand.

We have explored the possibility of using many randomly initialized contact points as seeds to our floating optimization. As it stands, we not been able to devise a scheme that ensures the construction of a reasonable hand given these contact seeds. This is due to the fact that even if the randomly seeded floating contact optimization generates a result that has objective score competitive with that of a manually initialized seed, the resulting motion can easily be infeasible or awkward given the position of the palm. Although the additional specification of perturbing forces seems to aid in finding good contact positions, we have not yet found a reliable solution to this problem. This limitation arises from the fact that the floating contact optimization has no concept of what a finger is and simply treats the palm of the hand as an external object that it needs to avoid collision with. Since the contact points in the floating optimization are treated as independent and no spatial constraints exist with respect to the base position, we are limited in our ability to synthesize more complex multi-step motions without explicit human guidance with regards to contact placement on the object. Future work should concern itself with improving this aspect of the floating contact planner: several suggestions are listed in the following section.

3.3 Limitations and Future Work

3.3.1 Future Additions to the Floating Contact Optimization

Our method is able to produce physically valid motion plans and automatically constructed hand topologies for an array of in-hand manipulation tasks. In this work, we have limited ourselves to in-hand manipulations whose goal is to reorient the object with respect to the palm. This class of manipulations does not include manipulations whose main purpose is to shift the grasp on the object using finger manipulations. An example of this type of motion would be re-positioning the fingers on a pencil object to go from a grasp concentrated on the lower portion of the pencil (as if to write with it) to the upper portion of the pencil as if to turn it by pivoting along the eraser end of the pencil. In this example, if the goal were to simply rotate the object, our floating contacting planner would come out with a very different answer than the more complex behavior of walking the hand's fingers up the length of the object to re-position the grasp. This is mainly due

to the fact that the floating contact optimization has no sense of what a finger is or how the contact points are related to the base of the hand. For this particular reason, we currently require that the user supply a reasonable guess at initial contact placement on the object in the floating optimization.

Ideally, our pipeline should be able to figure out proper contact placement such that there is no disconnect between the synthesis and floating steps of our optimization, however we leave this to future work. In the future, we may build in the ability for the contact planner to strategically re-position contacts with the concept of fingers and finger collisions in mind: this, combined with some additional object terms to express the underlying purpose of grasp-changing manipulations, could allow us to automatically replicate motions such as walking fingers up an object while maintaining control over it or gaiting fingers along the object to reposition it such that the fingers will not self collide. One possible way to address this issue is to create a planner that selects the initial contact positions as a seed for the optimization at each keyframe (rather than simply propagating a user-provided initial seed throughout all the keyframes as it is done now). This would hopefully give a rough sketch of the motion we plan to accomplish to the floating contact optimization, which would then fine tune the contact placement and determine the necessary forces to create a physically valid motion plan that displays the desired complex behavior. In this framework, the floating contact optimization would remain largely unchanged and would not be directly responsible for directing the more high-level purpose driven behavior of the manipulation. Successful inclusion of an additional planning step would likely open up the pipeline to automatically creating hands capable of much more sophisticated manipulations.

3.3.2 Extension To Multi-objective Hands

Since we are currently restricting our design pipeline to build hands for individual motions, most of our generated designs involve simple, yet valid constructions that are capable of carrying out the desired manipulations. Understandably, these hands are only suited for specific individual manipulation tasks: therefore a main focus of future work will be to extend this methodology to generating simplified hands that are able to accomplish a variety of related manipulation tasks. We refer to this extension as creating "multi-objective hands", in which the hand being designed is meant to accomplish two or more distinct motions. In this work, we

have partially addressed this problem by creating sequences of increasingly complex motions that essentially chain together simpler manipulations and optimizing hand designs for these motions. In general, a multi-objective hand need not be created such that the motions it is designed to accomplish trivially chain together as in the examples presented in this paper. Furthermore, we would seek to solve the more general problem of designing a procedure that could scale to more than just two or three simple manipulations for a given hand.

This extension is non-trivial given our current framework because it would require that we solve the problem of assigning fingertips to each of the contact point trajectories produced by the floating contact optimization. In the case of designing a hand for a single task this assignment problem is trivial since we are assigning each finger to only one trajectory. A naive implementation would simply try every combination of mapping fingers to trajectories in each floating motion, however this would quickly lead to a combinatorial explosion for even a small number of motions. We would thus seek to design a good heuristic for generating priors over (discrete) distributions for fingertip assignments as well as an efficient search procedure over these assignments to generate a multi-objective hand.

Phrasing the multi-objective problem as a set of parallel tasks involving the same hand has the additional benefit of avoiding speed bottlenecks introduced by a growing number of parameters introduced to the BFGS optimization algorithm [65], which scales quadratically with the number of parameters and effectively limits how many manipulations we could reasonably expect to chain together to get a multiobjective hand (as is done in this work). Application of the Alternating Direction Method of Multipliers (ADMM) algorithm to the synthesis step would likely be useful in this step to ensure convergence of the design.

3.3.3 Additional Future Improvements

Similar to [47], the final step of our pipeline does not explicitly model slipping and rolling interactions between the fingertips and the object [66]. This leaves the potential for it to generate unrealistic slipping/rolling interactions, however in most cases this is only a minor issue since we implicitly discourage this behavior when it is not necessary. We have also limited ourselves to considering point contacts between the fingertips and the object, rather than surface contact relationships.

Issues with building robust manipulators can 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, perhaps with the addition of unexpected perturbing forces or other uncertainties to encourage robust behavior. We could apply a gradient free optimization to jointly optimize the robot and the control policy given the warm start provided by our current third step.

Our synthesized hand mechanisms currently assume independent actuation of joints. We could potentially interleave our "whole hand" optimization with a dimensionality reduction step meant to couple joints together via linkage relationships. Using an implementation of an automatic linkage designer [55], we could iteratively update linkage relationships and re-plan with our mechanism until we reach a sufficiently low number of DOF's.

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.

Appendix A

Appendix A: Additional Notes on the Optimization Terms

A.1 Synthesis Optimization

A.1.1 Controllability Constraints

In this section we provide a more thorough examination of the Jacobian null space penalty and torque regularization terms as well as a complete derivation of these terms.

Null Space of Jacobian: First we compute the Jacobian J_i for finger i for i ranging over all fingers (we may treat these Jacobians seperately since we are assuming that joints are independently controlled) for the set of contact points currently selected by our optimization [see MLS chapter 4 for details on Jacobian calculation] [66]

From here we collect an orthonormal basis for the for the null space of J (i.e. a basis for the set of unit vectors x satisfying $Jx = 0$) consisting of vectors $E = e_0, ..., e_k$, where there are at most two elements of set E given that every finger must have at least one joint. Then

$$L_{jacNull} = \sum_i c_i * \sqrt{\sum_k (f \cdot e_k)^2} \tag{A.1}$$

Torque Regularization: It is not enough to simply penalize the component of the force lying in the null space of the Jacobian as demonstrated in the figure below, in which two joint axes that are slightly misaligned can eliminate the null space in the single finger case shown. The Jacobian null space term must be paired with a regularization on the active torques that must be applied at each joint to create the resultant force. This encourages the optimization to produce designs that can efficiently provide the needed force. We derive this term 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

$$F = T \times r_{perp} / \|\vec{r_{perp}}\|^2 + k * r_{perp} \tag{A.2}$$

where $r_{perp} = r - (r \cdot a) * a$ 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 $k * r_{perp}$ represents the passive force applied to this joint (i.e. force that the joint can not actively generate): we set $k = 0$ since we want our joint to actively provide all the force. The torque about the joint with unit axis a can be expressed as $T = \alpha_i * a$ where α_i is the scalar torque for joint i. From here, the force at the end effector is equal to the sum of the supplied forces from each joint, i.e. $F = \sum_{j \in joints} F_j$. Expressed in matrix form:

$$F = X * \alpha \tag{A.3}$$

where X is the matrix consisting of column vectors $T \times r_{perp} / \|\vec{r_{perp}}\|^2$ concatenated for each joint and α is the vector of torque magnitudes actively applied at the joints. Thus we want to minimize the term

$$L_{torque} = \|\vec{\alpha}\|^2 \tag{A.4}$$

where

$$\vec{\alpha} = (X^T X + \lambda^2 I)^{-1} X_T F \tag{A.5}$$

is the singularity robust psuedo-inverse solution to equation A.3 above with lambda being a small constant (we use $\lambda = .001$)

Bibliography

[1] John M Hollerbach. Workshop on the design and control of dexterous hands. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1982.

[2] Raymond R Ma and Aaron M Dollar. On dexterity and dexterous manipulation. In *Advanced Robotics (ICAR), 2011 15th International Conference on*, pages 1–7. IEEE, 2011.

[3] Allison M Okamura, Niels Smaby, and Mark R Cutkosky. An overview of dexterous manipulation. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 255–262. IEEE, 2000.

[4] CS Lovchik and Myron A Diftler. The robonaut hand: A dexterous robot hand for space. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 907–912. IEEE, 1999.

[5] Hong Liu, Ke Wu, Peter Meusel, Nikolaus Seitz, Gerd Hirzinger, MH Jin, YW Liu, SW Fan, T Lan, and ZP Chen. Multisensory five-finger dexterous hand: The dlr/hit hand ii. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3692–3697. IEEE, 2008.

[6] Frank Rothling, Robert Haschke, Jochen J Steil, and Helge Ritter. Platform portable anthropomorphic grasping with the bielefeld 20-dof shadow and 9-dof tum hand. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2951–2956. IEEE, 2007.

[7] Henrik I Christensen, T Batzinger, K Bekris, K Bohringer, J Bordogna, G Bradski, O Brock, J Burnstein, T Fuhlbrigge, R Eastman, et al. A roadmap for us robotics: from internet to robotics. *Computing Community Consortium*, 2009.

[8] Antonio Bicchi. Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE Transactions on robotics and automation*, 16(6):652–662, 2000.

[9] Aaron M Dollar and Robert D Howe. Towards grasping in unstructured environments: Grasper compliance and configuration optimization. *Advanced Robotics*, 19(5):523–543, 2005.

[10] Raymond R. Ma, Lael Odhner, and Aaron M. Dollar. A modular, open-source 3d printed underactuated hand. *2013 IEEE International Conference on Robotics and Automation*, pages 2737–2743, 2013.

[11] Hod Lipson and Melba Kurman. *Fabricated: The new world of 3D printing.* John Wiley & Sons, 2013.

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

[13] Aaron M Dollar and Robert D Howe. A robust compliant grasper via shape deposition manufacturing. *IEEE/ASME transactions on mechatronics*, 11(2): 154–161, 2006.

[14] Ian M Bullock and Aaron M Dollar. Classifying human manipulation behavior. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[15] P Wright, J Demmel, and M Nagurka. The dexterity of manufacturing hands. *Robotics Research, DSC*, 14:157–163, 1989.

[16] Zexiang Li, John F Canny, and Shankar S Sastry. On motion planning for dexterous manipulation. i. the problem formulation. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 775–780. IEEE, 1989.

[17] Karun B Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230–266, 1996.

[18] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 348–353. IEEE, 2000.

[19] Thomas H Speeter. Primitive based control of the utah/mit dextrous hand. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 866–877. IEEE, 1991.

[20] Steve Jacobsen, Edwin Iversen, D Knutti, R Johnson, and K Biggers. Design of the utah/mit dextrous hand. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1520–1532. IEEE, 1986.

[21] Jörg Butterfaß, Markus Grebenstein, Hong Liu, and Gerd Hirzinger. Dlr-hand ii: Next generation of a dextrous robot hand. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 109–114. IEEE, 2001.

[22] Tetsuya Mouri, Haruhisa Kawasaki, Keisuke Yoshikawa, Jun Takai, and Satoshi Ito. Anthropomorphic robot hand: Gifu hand iii. In *Proc. Int. Conf. ICCAS*, pages 1288–1293, 2002.

[23] M Vande Weghe, Matthew Rogers, Michael Weissert, and Yoky Matsuoka. The act hand: Design of the skeletal structure. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3375–3379. IEEE, 2004.

[24] J Kenneth Salisbury and John J Craig. Articulated hands: Force control and kinematic issues. *The International journal of Robotics research*, 1(1):4–17, 1982.

[25] EJ Van Henten, DA Vant Slot, CWJ Hol, and LG Van Willigenburg. Optimal manipulator design for a cucumber harvesting robot. *Computers and electronics in agriculture*, 65(2):247–257, 2009.

[26] Song Han, Sun Xueyan, Zhang Tiezhong, Zhang Bin, and Xu Liming. Design optimisation and simulation of structure parameters of an eggplant picking robot. *New Zealand Journal of Agricultural Research*, 50(5):959–964, 2007.

[27] Alistair J Scarfe, Rory C Flemmer, HH Bakker, and Claire L Flemmer. Development of an autonomous kiwifruit picking robot. In *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, pages 380–384. IEEE, 2009.

[28] Johan Baeten, Kevin Donné, Sven Boedrij, Wim Beckers, and Eric Claesen. Autonomous fruit picking machine: A robotic apple harvester. In *Field and service robotics*, pages 531–539. Springer, 2008.

[29] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dimensionality reduction for hand-independent dexterous robotic grasping. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3270–3275. IEEE, 2007.

[30] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[31] Frank L Hammond, Jonathan Weisz, A Andrés, Peter K Allen, and Robert D Howe. Towards a design optimization method for reducing the mechanical complexity of underactuated robotic hands. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2843–2850. IEEE, 2012.

[32] Chris Paredis and Pradeep K Khosla. An approach for mapping kinematic task specifications into a manipulator design. 1991.

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

[34] J-F Collard, Paul Fisette, and Pierre Duysinx. Contribution to the optimization of closed-loop multibody systems: Application to parallel manipulators. *Multibody System Dynamics*, 13(1):69–84, 2005.

[35] Sung-Gaun Kim and Jeha Ryu. New dimensionally homogeneous jacobian matrix formulation by three end-effector points for optimal design of parallel manipulators. *IEEE Transactions on Robotics and Automation*, 19(4):731–736, 2003.

[36] Marco Ceccarelli, Giuseppe Carbone, and Erika Ottaviano. An optimization problem approach for designing both serial and parallel manipulators. In *Proc. of MUSME 2005, the Int. Sym. on Multibody Systems and Mechatronics*, pages 6–9, 2005.

[37] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6): 369–395, 2004.

[38] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.

[39] Chris Leger et al. *Automated synthesis and optimization of robot configurations: an evolutionary approach*. Carnegie Mellon University USA, 1999.

[40] Hod Lipson and Jordan B Pollack. Towards continuously reconfigurable self-designing robotics. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1761–1766. IEEE, 2000.

[41] Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974, 2000.

[42] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 167–174. ACM, 2013.

[43] Irene Albrecht, Jörg Haber, and Hans-Peter Seidel. Construction and animation of anatomically based human hand models. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 98–109. Eurographics Association, 2003.

[44] C Karen Liu. Synthesis of interactive hand manipulation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 163–171. Eurographics Association, 2008.

[45] C Karen Liu. Dextrous manipulation from a grasping pose. In *ACM Transactions on Graphics (TOG)*, volume 28, page 59. ACM, 2009.

[46] Michael Posa and Russ Tedrake. Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic foundations of robotics X*, pages 527–542. Springer, 2013.

[47] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 137–144. Eurographics Association, 2012.

[48] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):43, 2012.

[49] Nancy S Pollard and Victor Brian Zordan. Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 311–318. ACM, 2005.

[50] Paul G Kry and Dinesh K Pai. Interaction capture and synthesis. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 872–880. ACM, 2006.

[51] Yuting Ye and C Karen Liu. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (TOG)*, 31(4):41, 2012.

[52] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)*, 32(4):83, 2013.

[53] Duygu Ceylan, Wilmot Li, Niloy J Mitra, Maneesh Agrawala, and Mark Pauly. Designing and fabricating mechanical automata from mocap sequences. *ACM Transactions on Graphics (TOG)*, 32(6):186, 2013.

[54] Gaurav Bharaj, Stelian Coros, Bernhard Thomaszewski, James Tompkin, Bernd Bickel, and Hanspeter Pfister. Computational design of walking automata. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 93–100. ACM, 2015.

[55] Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. Computational design of linkage-based characters. *ACM Transactions on Graphics (TOG)*, 33(4):64, 2014.

[56] Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. Linkedit: interactive linkage editing using symbolic kinematics. *ACM Transactions on Graphics (TOG)*, 34(4):99, 2015.

[57] Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. Interactive design of 3d-printable robotic creatures. *ACM Transactions on Graphics (TOG)*, 34(6):216, 2015.

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

[59] 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. A compliant, underactuated hand for robust manipulation. *The International Journal of Robotics Research*, 33(5):736–752, 2014.

[60] Aaron M Dollar and Robert D Howe. Simple, robust autonomous grasping in unstructured environments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4693–4700. IEEE, 2007.

[61] Aaron M Dollar and Robert D Howe. The highly adaptive sdm hand: Design and performance evaluation. *The international journal of robotics research*, 29(5):585–597, 2010.

[62] Aaron M Dollar and Robert D Howe. Joint coupling design of underactuated grippers. In *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 903–911. American Society of Mechanical Engineers, 2006.

[63] Vittorio Megaro, Jonas Zehnder, Moritz Bächer, Stelian Coros, Markus Gross, and Bernhard Thomaszewski. A computational design tool for compliant mechanisms. *ACM Trans. Graph*, 36(4):82, 2017.

[64] F Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of graphics tools*, 3(3):29–48, 1998.

[65] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[66] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.