Automated Design of Manipulators For In-Hand Tasks

Christopher Hazard, Nancy Pollard and Stelian Coros

***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 create a pipeline that automatically generates custom hands designed for specific manipulation tasks based on high level user input. Our framework comprises of a sequence of trajectory optimizations chained together to translate a sequence of objective poses into an optimized hand mechanism along with a physically feasible motion plan involving both the constructed hand and the object. We demonstrate the feasibility of this approach by synthesizing a series of hand designs optimized to perform specified in-hand manipulation tasks of varying difficulty.**

I. 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[11]. Dexterous manipulators are able to accomplish motions more efficiently and operate in limited workspace environments more easily[17]. Additionally, we want to develop manipulators that work in intuitive and human-like ways, particularly if they are meant to work alongside humans.

One line of research in dexterous manipulation focuses on the design of manipulators to mirror the kinematics of the human hand[16][15]. These hands have shown impressive capabilities with regards to dexterous manipulation [25] tasks, however the problem of dexterous manipulation remains unsolved [4]. One reason for this is that we cannot yet fully replicate the capabilities of the human hands and choices made to simplify the design may end up limiting capabilities of the hand. We have experienced this in our own research when the thumb of a dexterous hand does not have sufficient range of motion or the geometry of the hand's inner surfaces impedes rather than aids performing a manipulation. Progress in this domain is further burdened by the fact that these hands are prohibitively costly.

Rather than trying to approach manipulation from the perspective of human hand kinematics and dynamics, we focus on accomplishing some critical dexterous human hand functions and optimizing mechanisms to perform specific in-hand manipulation tasks. Our vision is to create an optimization pipeline for generating low cost hands that are well tuned for specific tasks or families of tasks. The possibility of creating useful low-cost hands has been well demonstrated, as in [18][6][7]. 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 takes high level user specifications such as a sequence of goal poses for a manipulated object and builds a mechanism specifically designed for the given task with no additional parameter tuning required on the part of the user. In this work, we limit ourselves to the class of in-hand manipulations that can be wholly described as reorientation of the object with respect to the palm, however the pipeline we have developed is extensible to other classes of in-hand manipulations. We show that our pipeline is able to synthesize a wide variety of useful specialized manipulators for various tasks.

II. Related Work

A large body of work revolves around classifying human manipulation behaviors and replicating them with robotic manipulators inspired by the human hand. Works such as [1] and [27] 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 [23] with the intention of mimicking these motions on robotic hands. Platforms such as the NASA Robonaut hand [16], GIFU III [21], and Shadow Dexterous [15] have become standard models on which manipulation algorithms and controllers have been implemented to mimic these types of behaviors. These hands are meant to be generic manipulators that should be able to carry out virtually any manipulation task given an appropriate control policy.

Low DOF hands have the advantage of being easier to build and maintain, easier to control, less expensive, and less prone to mechanical failure since they have fewer moving parts[9][8]. Due to the fact that they are cheaper and easier to build, specifics of the design can be optimized to tune or specialize a given hand. Various works [3][5][2] have optimized continuous parameters such as component lengths, tendon stiffness, and pulley radii to address kinematic concerns such as reachability constraints, avoidance of Jacobian singularities within the workspace, limits on individual joint torques, etc. [26] addresses the problem of discrete optimization of gripper design by chaining together individual modules to build fingers until a desired grasp quality is reached.

We build on previous work by optimizing both discrete and continuous characteristics of hand design to suit specific tasks. A significant portion of our design process consists of trajectory optimizations to test the competence of our

hands in performing different tasks. Trajectory optimization methods have shown remarkable ability to synthesize complex motions in both robot locomotion and manipulation, allowing the user to create complicated physically feasible motions from high level goal specifications. [13] and [12] 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 constructs physically feasible motion plans to synthesize target manipulations. Other work in trajectory optimization for manipulation captures demonstrated manipulations and finds contact forces that explain the motion[28].

Recent work in trajectory optimization has explored the use of discontinuous contacts in locomotion and manipulation tasks [19][24]. Mordatch et. al. [19] introduced the concept of contact invariance, in which contact is treated as a continuous variable, to facilitate optimization with changing contacts. Our work draws inspiration from [20], which applies the contact invariant method to the domain of manipulation.

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

III. Optimization Pipeline Description

We focus on precision in-hand manipulation where the hand manipulates an object with the fingertips in order to change the object's configuration with respect to the base of the hand. The object may be partially supported by the environment. This type of manipulation is fundamental to acquiring and placing objects, and moving from one grasp to another [22]. Our examples demonstrate 2 and 3-fingered hands, however our approach can be applied to accommodate hands with more fingers.

Our optimization pipeline has three parts, as shown in Figure 1. The input to our system is a sequence of objective poses for the object, a trajectory for the base, and an initial placement (subject to change) of the contact points on the object. The output of our system is an optimized mechanism, contact points, and forces that meet our objectives in a physically valid motion. The sections below discuss the components of this pipeline. Results are given in Section IV, and implementation details are provided in Appendix A.

A. Floating Contact Optimization

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

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

be the state at time t of the object, with $\mathbf{x_O}$ denoting the object's position and orientation in the world frame, and $\dot{\mathbf{x}}_\mathbf{O}$ being the derivative at time t of position and 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 [19] that is constrained to lie in the interval [0,1], with 0 representing an inactive contact and 1 being fully active.

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

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

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

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

- Physics terms:

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

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

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

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

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

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

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

- Task objectives:

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

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

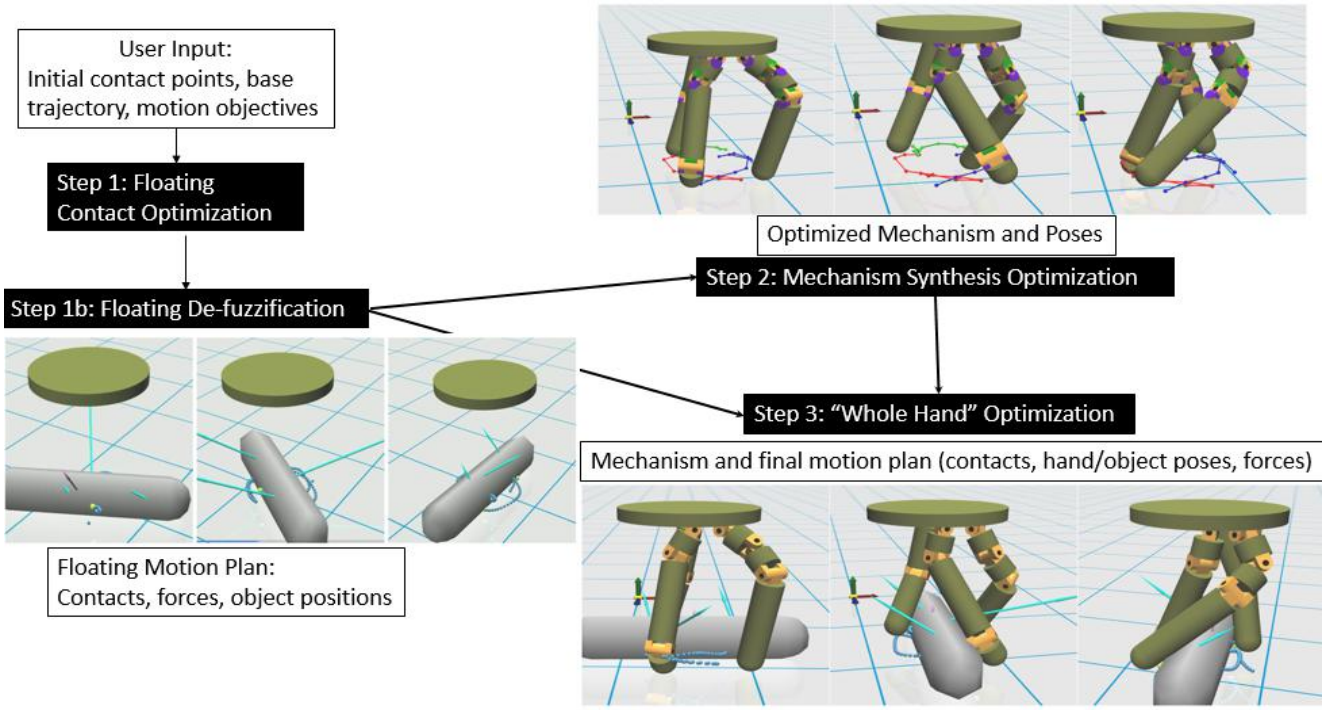


Fig. 1. Given the required user input, our "floating" optimization generates a physically feasible motion plan using disembodied contacts points. We then synthesize a mechanism with fingertips designed to follow these trajectories and provide the required forces. Finally, we combine the floating motion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

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

- Contact Invariant Costs:

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

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

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

$$L_{ci\_object\_slippage}(t) = \sum_i ||c_i f_i||^2 * ||(\dot{e}_{object}(i,t))||^2 \tag{12}$$

The L_{ci_object} cost dictates that the contact points lie on the object surface, while the $L_{ci\_object\_slippage}$ term discourages (but does not prevent) contact point slippage on the object for active contacts.

- Additional regularization terms:

$$L_{floatingContactAccel}(t) = \sum_i ||\ddot{r}_i(t)||^2 \tag{13}$$

$$L_{objectAccelReg}(t) = \sum_i \ddot{x}^2 \tag{14}$$

$$L_{angularAccelReg}(t) = \sum_i ((\omega \times (I_{world}\omega) + I_{world}\dot{\omega})/t_{step})^2 \tag{15}$$

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

The motions output by a first pass through the floating optimization have contact invariant values c_i between 0 and 1, and typically cluster around higher values (above .7) and low values (.3 and below), indicating the importance of the contact point in the optimization. After this first pass, we "de-fuzzify" our c_i values by setting each c_i to either 0 or 1 based on a threshold of either .1, .2, or .3. We pick our threshold by testing each one and re-optimizing our floating motion with the contact values held fixed, ultimately picking the "de-fuzzed" motion with the best objective value to pass to the synthesis optimization. Future steps in the pipeline hold these contact values fixed.

B. Mechanism Synthesis Continuous Optimization

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

We optimize a set of morphological parameters $\mathbf{M} = \{\mathbf{L}\ \mathbf{A}\ \mathbf{B}\}$, a set of joint angle poses $\mathbf{Q} = \{\mathbf{Q_1}, \mathbf{Q_2}, ..., \mathbf{Q}_{N_{keyframes}}\}$, and a set of contact

points $\mathbf{P}$ on the constructed fingertips such that:

$$\mathbf{M}, \mathbf{Q}, \mathbf{P} = \underset{\mathbf{M,Q,P}}{argmin}\ \Sigma_k \Sigma_i\ w_i * L_i(k) \tag{16}$$

$$\text{for } k \in \{1, 2, ..., N_{keyframes}\} \tag{17}$$

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

- Contact point costs:

$$L_{eeTarget}(k) = \sum_i c_i * ||p_i - p_{target}||^2 \tag{18}$$

$$L_{contactDistSurface}(k) = \sum_i ||p_{proj} - p_i||^2 \tag{19}$$

  where c_i represents the binarized contact invariant term (either 0 or 1) for fingertip i at keyframe k. $L_{eeTarget}$ is the distance between the contact point p_i on fingertip i and the corresponding point on the given trajectory for that contact, encouraging our selected contact points to line up with the trajectories from the floating motion plan. $L_{contactDistSurface}$ is the distance between the contact point and its projection onto the surface of the fingertip it is attached to: this is paired with a high coefficient to force contact points to lie on the surfaces of the fingertips.
- Collision
  For collision penalty calculations, we use a second order smooth piecewise cubic spline that interpolates between the functions $f(x) = 0$ for $x < 0$ and $f(x) = x^2$ for $x > 0$ as follows: $g(x) = \begin{cases} 0 & x \leq -\epsilon \\ \frac{x^3}{6\epsilon} + \frac{x^2}{2} + \frac{\epsilon x}{2} + \frac{\epsilon^2}{6} & -\epsilon \leq x \leq \epsilon \\ x^2 + \frac{\epsilon^2}{3} & \epsilon \leq x \end{cases}$

$$L_{collision}(k) = \sum_{i,j \in bodies} g(pen(body_i, body_j)) \tag{20}$$

  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 ($\epsilon = 10^{-6}$)
- Finger length costs:

$$L_{fingerLengthRegularization} = \sum_i (l_i)^2 \tag{21}$$

$$L_{fingerMinLengthCost} = \sum_i g(l_{min} - l_i) \tag{22}$$

  where i ranges over all the capsules present in the hand, l_i denotes the length of the principle axis of capsule i, and g denotes the piecewise cubic spline defined above. These two costs are meant to keep the finger lengths within a reasonable range of values.
- Controllability related costs:
  We require that our mechanisms be able to actively supply the necessary forces needed to accomplish the target motion. This is done through two terms: one to penalize the component of the applied force that lies along the null space of our mechanism's Jacobian (requiring our mechanism to actively supply the necessary force) and another to regularize the torque applied at the joints (encouraging efficient mechanisms).

$$L_{jacNull} = \sum_i c_i * \sqrt{\sum_k (f \cdot e_k)^2} \tag{23}$$

  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.

$$L_{torque} = \sum_i \|\vec{\alpha}\|^2 \tag{24}$$

  where $\vec{\alpha}$ is the vector of torque magnitudes that must be supplied by the mechanism to actively provide the desired force. Note that this torque penalty does not take into account the torque required to compensate for gravity, nor does it account for a hard maximum on the allowed torque to be supplied by a given motor (although this could be added if necessary).
- Additional costs:

$$L_{fingerPositions} = \sum_i ||proj_{base}(b_i) - b_i||)^2 \tag{25}$$

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

$$L_{fingerAcceleration}(k) = \sum_i (1 - c_i) * \ddot{x_i}^2 \tag{26}$$

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

$$L_{jtLimit}(k) = \sum_i \sum_j g(j(i) - j_{max}) + g(j_{min} - j(i)) \tag{27}$$

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

C. Mechanism Synthesis Discrete Optimization

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

After optimizing each finger independently for multiple trials, we enter a recombination step in which we combine the top performing fingers into a complete hand design. Upon recombination, we must re-optimize our hand due to the fact that we may incur self-collision among the recombined fingers (since they were optimized independently, their motions can easily overlap). The first recombination trial always takes the top performing fingers from each set of fingers meant to track the end effector points. Additional recombination trials randomly select fingers from each set according to a weighting that is inversely proportional to their combined $L_{eeTarget}$ and $L_{jacNull}$ scores (so that fingers with lower costs have higher chances of being selected). We then take our best performing hand, and if the combined $L_{eeTarget}$ and $L_{jacNull}$ scores for each finger fall below our threshold we return this design as our constructed hand. Otherwise, we add an additional joint to each of the fingers with scores still above this threshold and repeat the loop again until either that finger hits the maximum number of joints allowed per finger or it meets our objective criteria.

D. Whole Hand Optimization

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

- $L_{jointLimits}$: this is equation 27, taken from the synthesis step
- $L_{fingerAcceleration}$: equation 26 taken from the synthesis step
- $L_{collision}$: equation 20 from synthesis
- $L_{jacNull}$ and L_{torque}: equations 23 and 24 from the synthesis step
- Finger contact invariant terms: L_{ci_finger} and $L_{ci\_finger\_slippage}$ mirror the contact invariant terms introduced in the floating contact optimization, but applied to the finger instead of the object. L_{ci_finger} dictates that the hand's fingers remain in contact with the object if exerting a force, while $L_{ci\_finger\_slippage}$ discourages slippage of the contact point on the fingertip.

$$L_{ci\_finger}(t) = \sum_i c_i ||r_{proj} - r_i||^2 \tag{28}$$

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

$$L_{ci\_finger\_slippage}(t) = \sum_i ||c_i f_i||^2 * ||(\dot{e}_{finger}(i,t))||^2$$
$$\text{where } e_{finger}(i,t) = r_{i,proj\_finger}(t) - r_i(t) \tag{29}$$

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

$$\begin{aligned} L_{frictionConeHand}(t) = \sum_i c_i * \exp(\alpha(||f_i \\ -n*(f_i \cdot n_i)|| - \mu f_i \cdot n_i)) \end{aligned} \tag{30}$$

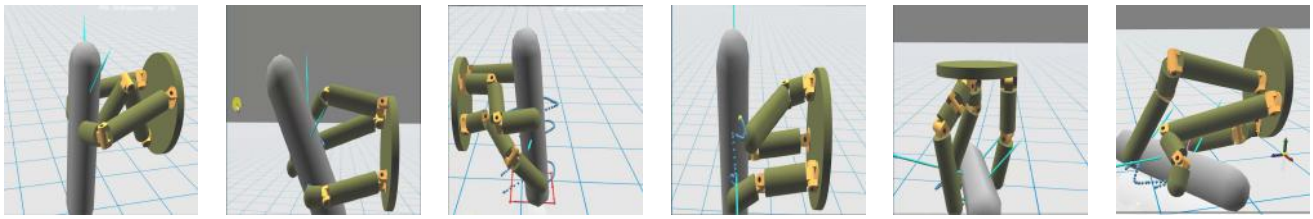
where n is the surface normal to the fingertip (world frame), μ is the coefficient of friction, and α is a sharpening factor for the exponent

IV. Results and Discussion

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

Our pipeline is robust in the sense that we use the same fixed set of optimization weights for each step regardless of the motion. In the vast majority of cases, the final result of our "whole hand" optimization yields very low physics error penalties and near zero Jacobian null space projection penalties, indicating that the pipeline generates a motion plan that fits very well the designed mechanism, ensuring that the mechanism is capable of performing the motion in a physically realistic way.

Our optimization program is able to discover quirks in our motions that lead to non-trivial mechanical designs. For example, in Figure 2(b), our optimization was able to suggest a mechanism in which we use one of our upper fingers to push out our capsule to bow it out while rotating it 90 degrees perpendicular to our palm surface, using the other two fingers to pivot the object. Our mechanism originally



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

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

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

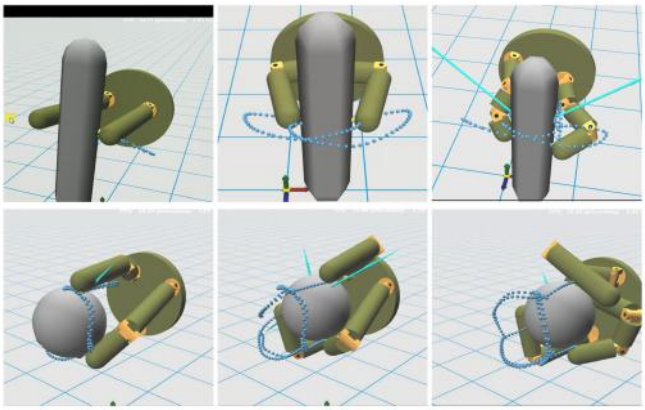


Fig. 4. Here we demonstrate 2 sequences of progressively more complex motions (refer to accompanying video for complete motions): (above) we demonstrate a horizontal side to side motion (without gravity, so that we don't require our mechanism to actively provide counter-gravitational force), a circular planar motion (no gravity), and a hemispherical motion (with gravity). (Below) we demonstrate a line of motions (all with gravity enabled) beginning with a sphere rotation 180 degrees either way, followed by a sphere rotation with the ability to translate in and out, and finally a sphere rotation with the ability to translate in the plane.

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 2(c), we replace our usual pose objective with an objective that attempts to track the shown goal points with the tip of the gripped "pencil". In this motion, our optimization discovered a cyclic manipulation in which the finger on the bottom left of the pencil automatically resets itself while still maintaining contact on the object. In the middle motion shown in Figure 4(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 2(e) and 2(f) in which we place our contacts in the same positions but placed our base differently: the result is that our floating motion plans are identical, but we get two completely different mechanisms out of the initial conditions. Similarly, we can get distinct mechanisms from placing our initial contacts differently. The fact that our pipeline gives different results for different initial conditions means that the user can select their ideal mechanism by trying out different initial conditions, as well as gain intuition about how the base and contact initialization affect the optimal mechanism design in general.

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

V. Limitations and Future Work

There is a disconnect between the floating optimization and synthesis optimization in our pipeline due to the simple fact that the floating optimization has no concept of what a finger is or how kinematic constraints can limit motion

capability. It is therefore not possible for our system to generate manipulations like finger gaiting motions or meaningful grasp transitions organically, although we can force this behavior to occur if we manually set the initial contacts at every frame.

This disconnect is the cause of most of the failure cases we have observed. Sometimes the floating contact optimization gives a non-collision free trajectory: in general this can be addressed by providing multiple initial seeds. Additionally, poor selection of base location can give awkward looking mechanisms. In future work, we may try to combine the design optimization with the contact planner to resolve these issues.

Our current pipeline is restricted to placing contacts on the distal finger segments, the palm, and objects in the environment. In future work, we plan to add in the capability to use intermediate finger segments as contact surfaces, allowing us to model motions like power grasps.

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. In particular, our generated hands are optimized with respect to a single known starting configuration as well as known object size and weight. In future work, we plan to design mechanisms that are robust to sensor noise, variations in object geometry, friction, mass, etc.

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.

VI. Appendix A: Implementation Details

Although our optimization has very few parameters for a user to tune, there are internal weights and other details required to duplicate these results. Weights were set the same for all examples. We observe that these weights lead to qualitatively similar motions as long as they are set to within an order of magnitude of the listed values. Most examples in the paper completed between 30 and 60 minutes on Dell Inspiron 5520 laptop.

A. Floating Contact Optimization

When calculating our objective in the floating contact optimization, we interpolate between the keyframes and calculate the sum at each time according to a time step $t_{step} = .1$ seconds. Weights for the objective terms are given in Table I. 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). 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

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

TABLE I

Table of common weights for floating contact optimization

1-3 additional keyframes between each of our objective keyframes, spaced uniformly over time. The object position component of $\mathbf{x_O}$ and contact positions $\mathbf{r_j}$ are interpolated via catmull-rom splines, object orientations are computed via the exponential map[10], 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.

In our optimization, we allow for one point contact on each fingertip on our hand and a single point contact to allow the object to interact with the ground. We optimize our objective with a standard L-BFGS solver [14].

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

B. Mechanism Synthesis Continuous Optimization

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

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

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

TABLE II

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

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 IV

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

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

TABLE III

Common weights for mechanism synthesis continuous optimization: note that $w_{eeTarget}$ and $w_{collision}$ vary with the annealing schedule

C. Mechanism Synthesis Discrete Optimization

On a given iteration of our outer loop, we typically optimize 10 different randomly seeded fingers, generate 5 recombined hands, and limit ourselves to a maximum of 3 joints per finger. In generating our initial seeds for the fingers, we randomly reseed the finger pose, finger segment lengths, the initial joint angles, joint axes, and the points at which the fingers connect to the base of the hand as well as the contact points on the fingertips (expressed in the local coordinate frame of the fingertip). We propagate the same random initial pose for the finger across all of the keyframes in our optimization.

D. "Whole Hand" Optimization

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

REFERENCES

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

[2] 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*. 2005, pp. 6–9.

[3] Marco Ceccarelli and Chiara Lanni. "A multi-objective optimum design of general 3R manipulators for prescribed workspace limits". In: *Mechanism and Machine Theory* 39.2 (2004), pp. 119–132.

[4] Henrik I Christensen et al. "A roadmap for us robotics: from internet to robotics". In: *Computing Community Consortium* (2009).

[5] J-F Collard, Paul Fisette, and Pierre Duysinx. "Contribution to the optimization of closed-loop multibody systems: Application to parallel manipulators". In: *Multibody System Dynamics* 13.1 (2005), pp. 69–84.

[6] Raphael Deimel and Oliver Brock. "A novel type of compliant and underactuated robotic hand for dexterous grasping". In: *The International Journal of Robotics Research* 35.1-3 (2016), pp. 161–185.

[7] Aaron M Dollar and Robert D Howe. "A robust compliant grasper via shape deposition manufacturing". In: *IEEE/ASME transactions on mechatronics* 11.2 (2006), pp. 154–161.

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

[9] Aaron M Dollar and Robert D Howe. "Towards grasping in unstructured environments: Grasper compliance and configuration optimization". In: *Advanced Robotics* 19.5 (2005), pp. 523–543.

[10] F Sebastian Grassia. "Practical parameterization of rotations using the exponential map". In: *Journal of graphics tools* 3.3 (1998), pp. 29–48.

[11] John M Hollerbach. *Workshop on the Design and Control of Dexterous Hands*. Tech. rep. MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1982.

[12] C Karen Liu. "Dextrous manipulation from a grasping pose". In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 3. ACM. 2009, p. 59.

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

[14] Dong C Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[15] Hong Liu et al. "Multisensory five-finger dexterous hand: The DLR/HIT Hand II". In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE. 2008, pp. 3692–3697.

[16] 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*. Vol. 2. IEEE. 1999, pp. 907–912.

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

[18] Raymond R. Ma, Lael Odhner, and Aaron M. Dollar. "A modular, open-source 3D printed underactuated hand". In: *2013 IEEE International Conference on Robotics and Automation* (2013), pp. 2737–2743.

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

[20] Igor Mordatch, Emanuel Todorov, and Zoran Popović. "Discovery of complex behaviors through contact-invariant optimization". In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), p. 43.

[21] Tetsuya Mouri et al. "Anthropomorphic robot hand: Gifu hand III". In: *Proc. Int. Conf. ICCAS*. 2002, pp. 1288–1293.

[22] Yuzuko C Nakamura et al. "The complexities of grasping in the wild". In: *Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on*. IEEE. 2017, pp. 233–240.

[23] 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*. Vol. 1. IEEE. 2000, pp. 255–262.

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

[25] Frank Rothling et al. "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*. IEEE. 2007, pp. 2951–2956.

[26] Filippo Sanfilippo et al. "Efficient modular grasping: an iterative approach". In: *Proc. IEEE Int. Conf. on Biomedical Robotics and Biomechatronics,(Rome, Italy)*. 2012, pp. 1281–1286.

[27] P Wright, J Demmel, and M Nagurka. "The dexterity of manufacturing hands". In: *Robotics Research, DSC* 14 (1989), pp. 157–163.

[28] Yuting Ye and C Karen Liu. "Synthesis of detailed hand manipulations using contact sampling". In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), p. 41.