Noname manuscript No.
(will be inserted by the editor)

Optimal Shape and Motion Planning for Dynamic Planar Manipulation

Orion Taylor · Alberto Rodriguez

Received: date / Accepted: date

Abstract This paper presents a framework for optimizing both the shape and the motion of a planar rigid end-effector to satisfy a desired manipulation task. Both shape and motion play key roles in determining contact interaction, and while both are commonly seen as design/control freedoms, their synergies are rarely formally explored. In this paper we study quasistatic problems like cam design, or dynamic problems like ball throwing, where both the shape and motion of the surfaces at contact are relevant.

We frame this design problem as a nonlinear optimization program, where shape and motion are decision variables represented as splines. The task is represented as a series of constraints, along with a fitness cost, which force the solution to be compatible with the dynamics of frictional hard contact while satisfying the task.

We illustrate the approach with the example problem of moving a disk along a desired path or trajectory, and we verify it by applying it to three classical design problems: the rolling brachistochrone, the design of teeth of involute gears, and the pitch curve of rolling cams. We conclude with a case study involving the optimization and real implementation of the shape and motion of a dynamic throwing arm.

Keywords Manipulation · Optimization · Robot End Effector Design

1 Introduction

Jai alai players use a *cesta* to catch and throw a ball at high speeds and with high accuracy (Figure 1). The cesta is an

Orion Taylor and Alberto Rodriguez
Department of Mechanical Engineering, Massachusetts Institute of Technology
E-mail: tayloro@mit.edu, albertor@mit.edu
Video Link

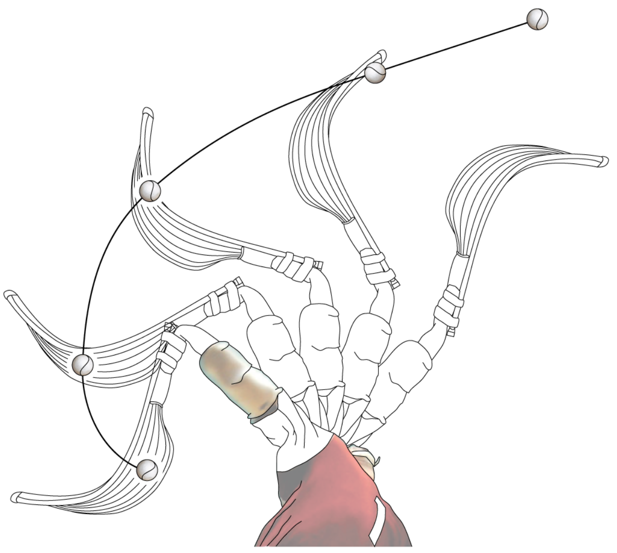


Fig. 1 Jai alai player throwing a ball. The cesta allows players to transfer a large amount of energy to the ball while controlling its rolling trajectory.

evocative example of the interplay between shape and motion. Their coordination allows players to transfer a large amount of energy to a ball while controlling its trajectory.

Just like jai alai, many common mechanisms rely on synergies between shape and motion to encode and maximize functionality. This is particularly true in mechanisms designed for high performance on specialized tasks, for example how the geometry of screw threads encodes the relationship between applied torque and output force.

We are motivated by the observation that both motion and shape play key roles in determining contact interaction [22]. Both are commonly exploited as design or control freedoms, but their synergies are rarely formally explored in robotics, where contact tends to be for finger-tips and pointy-feet. In this paper we study the problem of simultaneously optimizing them for planar manipulation tasks, and illustrate that by doing so, we unlock functionality impossible otherwise.

The main contribution of this paper is a general framework to design shapes and to plan motions that work together to accomplish kinematic or dynamic tasks such as reaching a goal state, follow a path or a trajectory, or optimize a fitness function. A significant part of the contribution is in the representations for motion, shape, and tasks, that enable the optimization. The long term goal of our work is a better understanding, and the development of tools to effectively use shape and actuation in manipulation.

The proposed approach to design shape and motion, and the structure of this paper, is as follows:

- The **problem** has the form of a standard nonlinear program where shape and motion are decision variables. The dynamics-kinematics of planar contact are represented as constraints, similar to previous works on trajectory optimization through contact [14, 16].
- The **representation** of shape, motion, interaction, and task, is in terms of splines and dynamic-kinematic constraints at collocation points. Section 3 describes the system, Section 4 the interaction and task constraints, and Section 5 their spline parametrization.
- We **illustrate** the approach in Section 6 with the toy example task of moving a disk along a desired path or trajectory. We show the differences between optimizing either the manipulator's shape, or its trajectory, or both. A key observation is that often both shape and motion can be used to satisfy the same task, and that there is an inherent nullspace in their combined design space.
- We **verify** the approach in Section 7 by formulating classical problems with known solutions: gear tooth profiles, rolling cams of variable transmission, and the rolling brachistochrone. The optimization approach yields correct solutions, and offers flexibility in studying variations.
- We **implement** the problem of planar dynamic throwing in Section 8. In the optimized solutions, the shape and throwing trajectory cooperate with gravity to maximize reach and respect the frictional limits of the interaction between a throwing palm and a ball.

We finish with a discussion of the main challenges involved in simultaneously optimizing shape and motion and promising directions for future work.

2 Related Work

2.1 Shape optimization for contact

The field of shape optimization for contact interactions is a broad subset of mechanism design for automation. Examples include the design of part feeders, traps, fences, finger shapes, gear teeth, and cams. Caine [3] develops a framework and set of computational tools for designing the shapes of features in a vibratory bowl feeder. Caine's work models contact interaction as a set of constraints in space of possible object motions. A set of shaped fences in the part feeder is used to block trajectories of the object that result in undesired orientations. Just as in Caine's work, we seek to design the manipulator shape to take advantage of the constraints that contact places on the motion of an object being manipulated. Unlike in Caine's work, we use constraints explicitly to produce a particular trajectory, rather than to prevent them.

Similarly, Brokowski et al [2] optimize the shape of a curved fence used to reorient parts traveling along a conveyor belt. A model of the interaction between part and fence, and the desired output orientation of the part, place a set constraints on the fence shape. These are converted into a set of differential equations which are then integrated to generate the fence shape that produces the desired part reorientation. In the case of Caine and Brokowski, motion is assumed to be quasistatic. Moreover, knowledge of the exact motion of part is not necessary. Rather, the effector shape is used as a funnel, capturing a broad set of initial states while letting through a small set of output states. Our approach on the other hand, in the absence of any explicit feedback, requires precise knowledge of the state of both the part and the end effector at all time. In that respect, it can be seen closer to a planning problem.

Rodriguez and Mason [21, 22, 20] build a framework for computing end-effector shapes for 1 DOF actuators and desired contact interactions based on sets of contact normals. In this case, each instance of contact presents a local constraint on the shape of the end effector, which can be converted into a differential equation, which is then integrated into an end effector shape. Motion is still assumed to be quasistatic. However, unlike Caine or Brokowski, but similar to our work, the state of both the part and end effector are prescribed at all times. In all three cases, there is no notion of optimization, merely constraint satisfaction.

Gear design is a relatively large field with extensive recent work [5, 12, 11, 25, 26] on methods to design shapes and pitch curves of circular and non-circular gears. This work is primarily analytical in nature and targeted toward the specific task of pitch curve design. In contrast, we are able to apply our general numerical optimization based framework developed in this paper to tackle gear tooth and pitch curve design problems, as seen in sections 7.2 and 7.3.

2.2 Trajectory optimization through contact

The manipulation and locomotion communities have been especially interested in motion optimization involving frictional contact. Lynch and Mason. [13] introduced a control system for a one joint nonprehensile manipulator, enabling it to preform various dynamic tasks such as throwing and catching with a flat palm/arm. This work is very similar to the motion planning aspects of our work, given that both focus on motion planning for a one joint nonprehensile manipulator. Moreover both works frame this task as a constrained nonlinear optimization problem. However our work differs in a few key aspects. Lynch uses a shooting method, where the trajectory of the object is obtained as the forward integration of control decision variables. This is well suited for trajectories that go through multiple modes of contact, but makes it more difficult to impose path constraints along the trajectory of the object. On the other hand, we use a collocation method, that gives us more control to constrain the trajectory of the object.

Ryu et al [23] and Lippiello et al [10] both create control frameworks for stabilizing and driving planar rolling systems. Both of these papers deal with special symmetric examples of a rolling contact manipulation system. Because of this, they are able to exploit the symmetry in the problem to derive elegant control systems to accomplish the desired task. Our work deals with a more general form of the problem, and is focused on deriving original shape/trajectory pairs, rather than controlling around them.

Posa et al [16] propose one of the very few frameworks for trajectory optimization for systems that undergo intermittent frictional interaction based on a complementarity formulation for contact resolution. Unlike in our applications, that focus on rolling contact, Posa's framework allows to discover trajectories that undergo contact mode changes. Like in our work, Posa also makes use of direct trajectory optimization methods.

Becker and Bretl [1] design a set of control inputs for a sphere rolling on a table such that the cumulative rotation of the sphere is invariant with respect to its size- a unique example of motion planning that takes shape uncertainty into account. A key feature of this analysis that is shared with our framework is that it takes into account how variations in both the geometry of the system and the motion of the system affect the transition between the initial and final states of the system, and synthesizes the two to get a desired result.

2.3 Combined shape and motion optimization

Despite the abundance of work on either shape or motion optimization for contact interactions, there is relatively little work that approach both simultaneously. Reist and D'Andrea. [17, 18] optimize the motion and concavity of a paddle juggler capable of stably bouncing a ball without feedback. The approach is limited to the particular application, and the contact manipulation is limited to periodic instantaneous impacts. Reist is able to pose this problem such that the task of shape and motion optimization is reduced to the design of two key scalar parameters (paddle acceleration at time of impact and paddle curvature). This stands in contrast with our manipulation task, which requires the optimization of a full trajectory and a full effector shape profile.

Lynch [14] explores the design space (shape and motion) of a contact juggler for the specific task of butterfly juggling in a planar rolling system. Both motion and shape are represented as a 2-dimensional parametrized families. This system is the closest work to this paper, and serves as primary inspiration for the proposed approach. In our work, we more generally use spline parametrizations of shape and motion, and our formulation optimizes simultaneously both for motion trajectory and shape profile in the same nonlinear program.

Coincidentally, Chen [4] optimizes both the shape and control input for an underactuated throwing arm. Chen focuses on a problem that is nearly identical to the dynamic throwing example explored in this paper. Morever, they also frame this as a nonlinear optimization where both shape and effector motion are design variables that are optimized simultaneously. However, this approach uses a very different parameterization for shape and motion, as well as an indirect trajectory optimization algorithm. A direct trajectory optimization method leads us to a significantly sparser set of constraints, which allows us to explore higher dimensional parametrizations.

More recently, Ha et al [8] presents a formulation for simultaneously optimizing shape and motion parameters for the design of a quadruped robot. This is a recent application of simultaneous shape-motion design to a problem that is quite different from the one explored in this paper.

3 A Planar Manipulation System

This paper focuses on a type of planar contact manipulation system consisting of two rigid bodies: a hand H and an object B, which share a single contact point, as illustrated in Figure 2. This section describes the notation and coordinates that we will use to describe their shapes, motions, and interactions. Throughout the paper we will use subscripts h and b referring to hand and object respectively.

In the paper we will make use of the following notation:

- $(\mathbf{p}_h, \theta_h) = (p_{h_x}, p_{h_y}, \theta_h)$ and $(\mathbf{p}_b, \theta_b) = (p_{b_x}, p_{b_y}, \theta_b)$ describe the planar poses of hand and object.
- $\mathbf{c}_h(s), \mathbf{c}_b(s) : [0,1] \to \mathbb{R}^2$ parametrize the shape profiles of hand and object in their respective frames.
- s_h and s_b are the values of the parameter s at contact, thus $\mathbf{c}_h(s_h)$ and $\mathbf{c}_b(s_b)$ are the contact point in the hand and object reference frames.
- $\mathbf{v}(s) = \frac{d}{ds}\mathbf{c}(s)$ is the tangent vector to a shape at point $\mathbf{c}(s)$ in the hand and object reference frames.

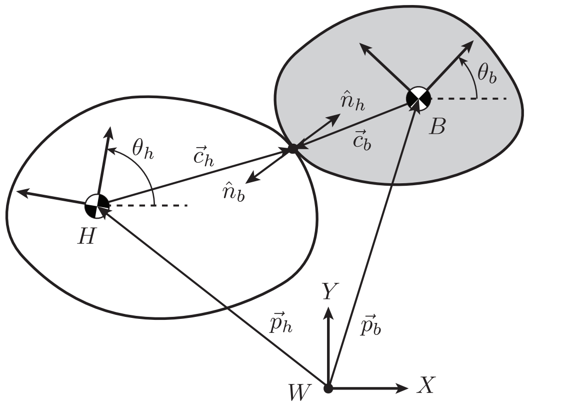


Fig. 2 Planar manipulation system consisting of a hand H manipulating an object B. We make use of an inertial reference frame W, and moving frames attached to the hand and ball. Their configurations are given by their position vectors and orientations $(\mathbf{p}_h, \theta_h)$ and $(\mathbf{p}_b, \theta_b)$. Hand and object interact at contact point located at $\mathbf{c}_h$ or $\mathbf{c}_b$ and with normals $\hat{n}_h$ and $\hat{n}_b$, all defined in the hand and object reference frames.

- $\hat{n}(s) = R(\frac{\pi}{2}) \cdot \frac{\mathbf{v}(s)}{|\mathbf{v}(s)|}$ is the normalized outward facing surface normal to a shape at point $\mathbf{c}(s)$ in the hand and object reference frames.

Note that we will make frequent use of the rotation matrix about axis $\hat{z}$ by θ radians, noted by $R(\theta)$.

The **motion** of the type of planar system we consider in this paper is then parametrized by the time-dependent functions:

$$\mathbf{p}_h(t), \theta_h(t), \mathbf{p}_b(t), \theta_b(t)$$

its **shape** is parametrized by the functions:

$$\mathbf{c}_h(s), \mathbf{c}_b(s)$$

and its **interaction** by the evolution of the contact point:

$$\mathbf{c}_h(s_h), \mathbf{c}_b(s_b)$$

which evolve in time with the parameters $s_h(t), s_b(t)$.

Note that the variables describing the system can be split into two categories: **design variables** $\mathbf{p}_h, \theta_h, \mathbf{c}_h, \mathbf{c}_b$ which describe parts of the system that can be directly controlled, i.e., the shape and motion of the hand and the shape of the ball; and **descriptor variables** $\mathbf{p}_b, \theta_b, s_h, s_b$ which describe underactuated degrees of freedom determined by the evolution of the design variables, i.e, the motion of the ball and the evolution of the contact point.

The system is also affected by the following constants, which we assume to be known: the mass of the object m, its moment of inertia I, gravity $\mathbf{g}$, and the coefficient of friction between hand and object μ.

4 A Planar Manipulation Task

We study two types of planar manipulation tasks framed as constrained satisfaction/optimization problems:

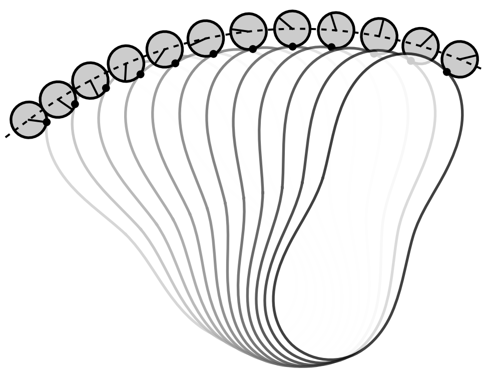


Fig. 3 Example of a simple planar manipulation task. Under gravity, the hand (white) moves an object (grey ball) along a trajectory. Note that the orientation of the object is linked to its displacement along the path.

1. Produce a desired motion of the object, in the form of either a goal state, a path, or a trajectory to follow, e.g. move along a curve, as shown in Figure 3.
2. Optimize a behavior of the object defined by a fitness function, e.g., throw fast.

To simplify interactions, and for the sake of optimization complexity, we restrict the search for solutions to where hand and object interact with sticking or rolling contact, but do not slip with respect to each other.

The task then takes the form of a nonlinear optimization program with shape and trajectory as decision variables, subject to dynamic, kinematic, and task constraints. A big part of the work, which we describe in the two following subsections, is in finding a tractable way to formulate these constraints.

4.1 Kinematic and Dynamic Constraints

Before tailoring the system to any particular task, we need to make sure that the interactions it produces adhere to the laws of physics. To do so, we impose a series of kinematic and dynamic constraints that guarantee that contact is maintained with no penetration, that frictional forces are such that objects do not slide with respect to each other, and that the acceleration of the system is along the resultant of forces.

The expression of the kinematics of contact in the form of constraints was already described by Montana [15]. The algebraic representation we use in this paper is similar to the one proposed by Lynch et al [14] to describe contact juggling. For their expression, we will make use of the notation introduced in Section 3 to describe the shape and motion of the system. For simplicity of notation, we suppress the dependencies of $\mathbf{c}_h$, $\mathbf{c}_b$ and their derivatives on s_h and s_b respectively.

Contact Constraint The contact points in the hand and object must be the coincident in the world reference frame:

$$\mathbf{p}_h + R(\theta_h) \cdot \mathbf{c}_h = \mathbf{p}_b + R(\theta_b) \cdot \mathbf{c}_b \quad (1)$$

Tangency Constraint At the point of contact, the vectors tangent to the hand and object must be opposing each other (note that $\angle$ measures the orientation angle of a vector):

$$\theta_h + \angle \mathbf{v}_h = \theta_b + \angle \mathbf{v}_b - \pi \pmod{2\pi} \quad (2)$$

Rolling Constraint The speed of the contact point in the hand and object must be opposite:

$$|\mathbf{v}_h| \dot{s}_h = -|\mathbf{v}_b| \dot{s}_b \quad (3)$$

Inertia Constraint The angular acceleration of the object must be consistent with the sum of torques ($\sum r \times f = I\alpha$):

$$(R(\theta_b) \cdot \mathbf{c}_b) \times m\,(\ddot{\mathbf{p}}_b - \mathbf{g}) = I\ddot{\theta}_b \quad (4)$$

Friction Cone Constraint The contact force exerted by the hand on the object must be inside the friction cone. If $\mathbf{f_h} = R(-\theta_h)m(\ddot{\mathbf{p}}_b - \mathbf{g})$ is the contact force applied from the hand to the object, in the hand reference frame, then:

$$\pm \frac{\mathbf{v}_h}{|\mathbf{v}_h|} \cdot \mathbf{f_h} \leq \mu \hat{n}_h \cdot \mathbf{f_h} \quad (5)$$

We refer to the first three constraints as *kinematic constraints* which we impose to all problems in this paper, and the last two as *dynamic constraints* which we impose only when appropriate.

These constraints are only a local approximation to the physics of interaction. They do not explicitly prevent, for example, the hand and object from intersecting at some point other than the studied contact point due to their global shapes, or due to their local curvatures. The global problem, while important and interesting, is significantly more difficult to formalize. The local constraints have proven useful and sufficient for the problems analyzed in the paper.

4.2 Manipulation Task Constraints

The previous constraints narrow the set of possible manipulation systems to those that are physically sound. Now we explore additional constraints and the use of fitness functions to represent manipulation tasks.

Decision variable constraints It is common to reduce the dimension of the problem by directly restricting the range of acceptable values of decision variables α.

$$\alpha = k \quad (6)$$

$$k_1 \leq \alpha \leq k_2 \quad (7)$$

Common examples are to constrain the hand to rotate about a pivot $\mathbf{p}_h = (0, 0)$, or to fix the shape of the object, for example to be a circumference of radius r:
$\mathbf{c}_b(s) = (r\cos(s), r\sin(s))$.

Initial and endpoint constraints We often want to constrain the hand or object to start from or reach a configuration:

$$\mathbf{p}_h(t_0|t_f) = \mathbf{k_1} \text{ and/or } \theta_h(t_0|t_f) = k_1 \quad (8)$$

$$\mathbf{p}_b(t_0|t_f) = \mathbf{k_2} \text{ and/or } \theta_b(t_0|t_f) = k_2 \quad (9)$$

to start from rest:

$$\dot{\mathbf{p}}_b(t_0) = \mathbf{0} \text{ and } \dot{\theta}_b(t_0) = \dot{s}_b(t_0) = 0 \quad (10)$$

$$\dot{\mathbf{p}}_h(t_0) = \mathbf{0} \text{ and } \dot{\theta}_h(t_0) = \dot{s}_h(t_0) = 0 \quad (11)$$

or to (additionally) start from a static equilibrium:

$$\ddot{\mathbf{p}}_b(t_0) = \mathbf{0} \text{ and } \ddot{\theta}_b(t_0) = \ddot{s}_b(t_0) = 0 \quad (12)$$

$$\ddot{\mathbf{p}}_h(t_0) = \mathbf{0} \text{ and } \ddot{\theta}_h(t_0) = \ddot{s}_h(t_0) = 0 \quad (13)$$

Implicit motion constraints In some cases, constraints only implicitly affect the decision variables. The most frequent use is to constrain the object or hand to move along a path, rather than a trajectory. These are formulated as general implicit non-linear constraints:

$$F(\mathbf{p}_b, \mathbf{p}_h) = 0 \quad (14)$$

Regularization constraints Occasionally, we incorporate extra constraints to guide the solver to find or avoid a particular type of solution. The two most frequently used regularization constraints are fixing the x component of the hand shape to a given function:

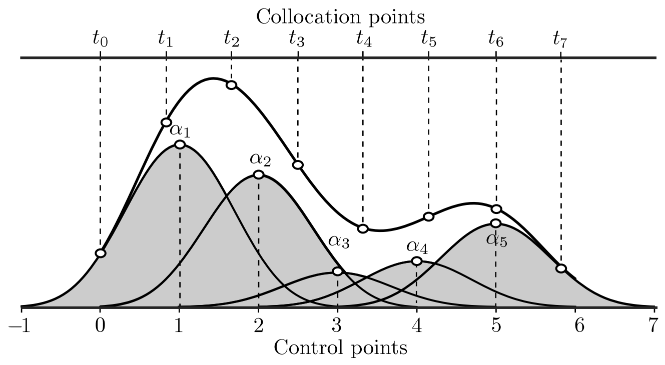
$$c_{h_x}(s) = k(s) \quad (15)$$

and constraining each point of the hand shape to a line that varies with s:

$$k_1(s)c_{h_x}(s) + k_2(s)c_{h_y}(s) = k_3(s) \quad (16)$$

We often use (15) to enforce $c_{h_y} = f(c_{h_x})$ i.e. the hand shape passes the vertical line test. Similarly, (16) is used to enforce $|\mathbf{c}_h| = f(\angle \mathbf{c}_h)$. Both are effective at preventing self-intersecting hand shapes, which are undesirable.

Fitness cost Often, we want to optimize a behavior with respect to a performance metric, rather than satisfy a particular constraint. These become part of the cost function of the optimization problem. Two performance metrics we will study in this paper are throwing distance and travel time.



Collocation points
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7
α_1 α_2 α_3 α_4 α_5
–1 0 1 2 3 4 5 6 7
Control points

Fig. 4 Spline construction of the continuous decision variable $\Phi(x)$ with the discretized parameters $\alpha_1 \ldots \alpha_N$, with $N = 5$, and with $M + 1 = 8$ collocation points where we will impose all constraints. Note that the domain of each basis is $(-2, 2)$ which gives sparsity to the spline representation. We'd like to emphasize that the number of collocation points and basis functions used in implementation depends on the problem and number of constraints. This is discussed in more detail in Section 9.

5 Discretization

The shape and motion of the system are functions of space and time. The formulation in Section 4 is continuous, but for optimization purposes, we use a discrete representation. We want a representation that supports smoothness C^2 and that is sparse, i.e., each decision variable has a limited domain of influence both in space and time.

To do so, we describe shape and motion as linear combinations of shifted basis functions, as illustrated in Figure 4. Let $\xi(x)$ be any given decision variable, dependent on x, which could be either time t or space s. Then we construct:

$$\xi(x) = \sum_{i=1}^{N} \alpha_i \Phi\left(L(x) - i + k\right) \tag{17}$$

where:

- Φ is a cubic B-spline basis function with uniformly spaced knot points, which is twice differentiable (smoothness) and only non-zero in the interval $(-2, 2)$ (sparsity).
- $\alpha_1 \ldots \alpha_N$ are coefficients that weight the basis functions. Note that these become the discrete decision variables that discretize the continuous decision variable ξ.
- $L(\cdot)$ is a factor that non-dimensionalizes x as appropriate. For shape variables, s is already dimensionless, so $L(s) = s$. For motion variables, we transform time as $L(t) = \gamma \frac{t}{\tau}$, where τ is a global time constant equal to the duration of motion ($\frac{1}{\tau}$ is a decision variable in the optimization program, allowing trajectories of varying length), and γ is a normalizing constant that ensures each component of the trajectory has the same nominal duration regardless of the number of basis functions used in its representation (which allows varying resolution).
- k is an offset (usually $k = 2$) used to correctly place the basis functions relative to the domain of ξ.

The use of basis functions allows us to compute in closed form the derivatives of the decision variables, which are necessary for many of the constraints described in Section 4:

$$\dot{\xi}(x) = \sum_{i=1}^{N} \alpha_i \dot{\Phi}\left(L(x) - i - k\right) \dot{L}(x) \tag{18}$$

$$\ddot{\xi}(x) = \sum_{i=1}^{N} \alpha_i \ddot{\Phi}\left(L(x) - i - k\right) \dot{L}(x)^2 \tag{19}$$

where we used that $\ddot{L}(x) = 0$.

Ideally, the constraints of motion would hold true at all times. However, for resolution purposes, we impose the motion constraints at $M + 1$ evenly distributed points along the trajectory, playing the role of *collocation* points in trajectory optimization [9]. In particular, we replace each of the continuous motion constraints in a problem

$$G(t, \alpha_1, ..., \alpha_N, \frac{1}{\tau}) = 0\big|_{t\in[0,\tau]} \tag{20}$$

with a set of $M + 1$ discrete constraints:

$$G(t_j = \frac{\tau}{M} j, \alpha_1, ..., \alpha_N, \frac{1}{\tau}) = 0\big|_{j=0\ldots M} \tag{21}$$

We extend this discretization to a periodic domain to represent closed shapes and periodic motions. The number of collocation points and basis functions we use varies with the problem and number of constraints. A more detailed description of implementation specifics can be found in Appendix A.

6 Illustrative Toy Problem

In this section we describe the method to optimize shape and motion with the simple problem of moving a ball under gravity along a desired path $\mathbf{p}_b$, with a rotational paddle.

We start by exploring these two problems: (Prob. 1) For a given fixed hand motion, is there a *hand shape* that forces the ball to travel along the desired path? and (Prob. 2) For a given fixed hand shape, is there a *hand motion* that forces the ball to travel along the desired path?

We can formulate both problems as a shape-motion pair that satisfies the following constraints:

- **Kinematic:** contact, tangency and rolling.
- **Dynamic:** inertia and friction.
- **Fixed decision variables:**
  - The object is a ball $\mathbf{c}_b(s) = (l\cos(s), l\sin(s))$.
  - The hand pivots about the origin: $\mathbf{p}_h(t) = \mathbf{0}$.
  - (Prob. 1) Fixed hand motion $\theta_h(t) = k \cdot t$, for example to constant velocity.
  - (Prob. 2) Fixed hand shape $\mathbf{c}_h(s)$, for example, to a straight line.

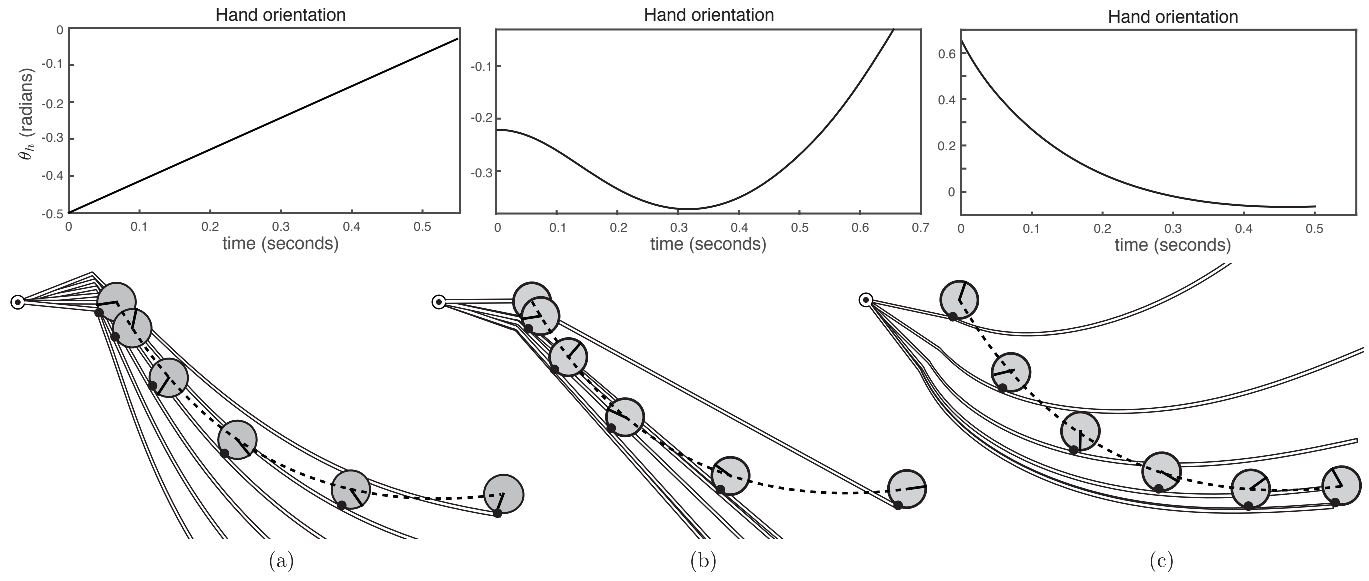


Fig. 5 Toy problem of moving a ball under gravity along a given path, drawn with a dotted line. There are infinite solutions in the shape-motion nullspace. Here we show two interesting cases: (a) Solution when the hand is forced to moved along a fixed trajectory, e.g., constant angular velocity, and only the hand shape is a design freedom. (b) Solution when the hand shape is fixed, e.g., a straight line, and only the hand trajectory is a design freedom. Note that the ball follows exactly the same path in both cases, although at different velocities. (c) Solution when both hand shape and trajectories are design freedoms. Note that in this case we can impose the trajectory of the ball, not just its path, illustrated by the fact that the ball moves at a constant speed along the curved path.

- **Task:** The ball moves along a desired path given as a level set $G(\mathbf{p}_b(t)) = 0$. In this example we use a parabola.

Figure 5a and Figure 5b show the outcome of the optimization. Both solutions satisfy all constraints and succeed in transporting the ball along the desired path, while rolling under the effect of gravity on the moving hand. This is an illustrative example of the nullspace that exists in the shape-motion design space. It is ultimately the combination of both that produces the desired object manipulation, but in many cases we can reproduce the effect of a motion in a shape, as well as the effect of a shape in a motion.

Note in the previous examples that the ball traverses the path at different speeds. Freeing both shape and motion in the optimization problem, and exploiting their nullspace, give us enough design freedom to control the trajectory $\mathbf{p}_b(t)$ along which the ball will move, not just its path. The problem has a very similar formulation to the previous one, but we instead replace the task constraint with a stricter constraint on the ball motion $\mathbf{p}_b(t) = \mathbf{p}_b^*(t)$, and remove the constraints on the shape or motion of the hand. The solution, illustrated in Figure 5c, succeeds in moving the ball along the desired path, but now, for example, with constant speed. Controlling the timing of the motion of the ball is something that would have been impossible when only optimizing shape or motion, because there are not enough design freedoms.

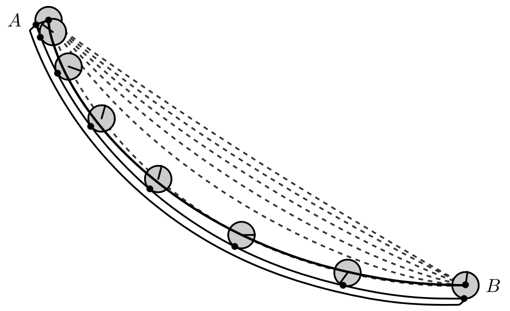


Fig. 6 Brachistochrone for a rolling cylinder. The analytical solution of the trajectory of the center of the cylinder converges (dotted line) to a cycloid (continuous line), when reducing the time to traverse from A to B.

7 Classical Problems

7.1 Rolling Brachistochrone

A brachistochrone curve is the path that allows an object to travel from A to B in the shortest amount of time, when starting from rest at A and accelerated by gravity $\mathbf{g}$. A classical result in mechanics is that when the object is a frictionless bead, the brachistochrone is a section of a cycloid [24]. Rodgers [19] showed that in the case of a rolling disk the brachistochrone is also a cycloid.

We represent the problem of the rolling brachistochrone with the proposed framework, where the path is a non-moving hand and the object is a disk, whose rolling trajectory down the hand $\mathbf{p}_b(t)$ becomes the brachistochrone when it achieves minimum travel time. We impose the following constraints:

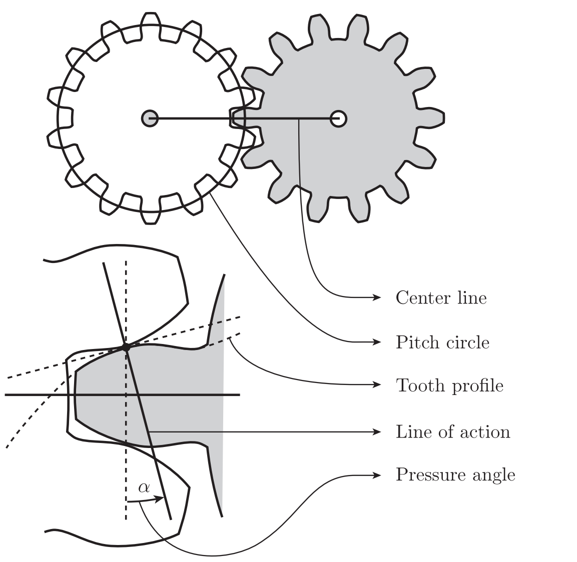


Fig. 7 Anatomy of a gear. The interaction between gears is largely determined by their tooth profile. The *center line* is the line connecting the two rotation centers, the *line of action* is orthogonal to the contact tangent between the two gears, hence is the direction along which force is transferred from one gear to another. The *pressure angle*, complementary to the angle between the line of action and the center line, is key in the design of gears.

- **Kinematic:** contact, tangency and rolling.
- **Dynamic:** inertia. Since the disk rolls without slipping (infinite friction), we omit the friction cone constraint.
- **Fixed decision variables:**
  - The object is a disk $\mathbf{c}_b(s) = (l\cos(s), l\sin(s))$.
  - The hand is static $\mathbf{p}_h(t) = \mathbf{0}$ and $\theta_h(t) = 0$.
- **Initial and endpoint constraints:**
  - The disk starts at rest: $\dot{\mathbf{p}}_b(t_0) = \mathbf{0}$, $\dot{\theta}_b(t_0) = \dot{s}_b(t_0) = 0$.
  - The disk starts at $\mathbf{p}_b(t_0) = A$.
  - The disk ends at $\mathbf{p}_b(t_f) = B$.
- **Task:** The duration of the trajectory is $\tau = T$.

We have approached the formulation of this problem in two different ways. In the first approach, we set the objective function to be proportional to the duration of the trajectory T. We have found that this approach results in the optimizer getting trapped in local minima. An alternative method is to set the value of T as a constraint, then iteratively decrease it outside the optimizer. The process starts with T equal to the time it takes for the roller to traverse a straight line from A to B, and gradually asks the program to find paths with smaller and smaller values of T until a solution cannot be found. Though this method requires human supervision to choose the values of T to iterate through, we have found that it is more robust. Figure 6 shows the sequence of solutions converging to a cycloid.

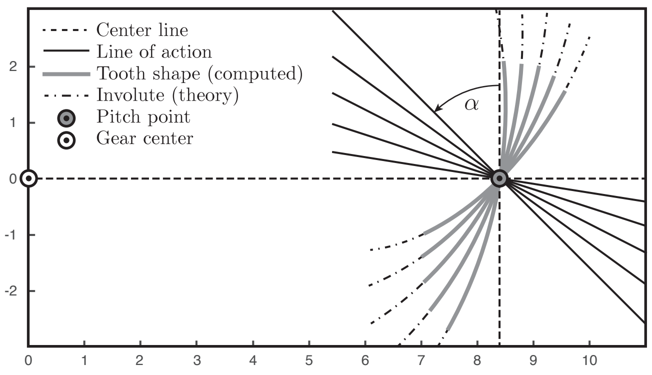


Fig. 8 Gear tooth profiles obtained for five different pressure angles α. The figure shows the corresponding line of action for each desired pressure angle (orthogonal to the gear tooth profile at the pinch point) and the recovered gear tooth profiles. These correspond very accurately to involute curves, known to provide constant pressure angle.

7.2 Involute Gears

Gear design is a classic shape design problem. A pair of gears should mesh while maintaining a time-invariant gear ratio. The fundamental law of gearing [7] states that these two properties are equivalent to constraining the *line of action* to pass through the *pitch point* at all times. Figure 7 illustrates these concepts.

Involute gears, i.e., gears with teeth shaped as involute curves, are a popular solution that satisfy the above properties. One unique (and useful) property of the meshing between involute gears is that the line of action is constant throughout contact. As a consequence, the *pressure angle* α, which determines the amount of power that can be transmitted through the gear train, is constant throughout the meshing.

We now formulate the problem of finding gear shapes that satisfy the above properties with the proposed optimization approach. More concretely: for a given center distance between gears l, gear ratio r, and pressure angle α, find gear teeth that mesh adequately. We will indeed recover involute gears. In this case both hand and object are a pair of meshing gear teeth, and we impose the following constraints:

- **Kinematic:** contact, tangency and rolling.
- **Fixed decision variables:**
  - The driving gear rotates about the origin $\mathbf{p}_h(t) = \mathbf{0}$.
  - The driven gear rotates about the point $\mathbf{p}_b(t) = (l, 0)$.
- **Task:**
  - Constant gear ratio r. We achieve this by fixing the trajectory of the gears $\theta_h(t) = \omega t$, $\theta_b(t) = -r\omega t$.
  - Constant pressure angle α. That is: $\theta_h + \angle \mathbf{v}_h(s_h) = \alpha + \pi \pmod{2\pi}$

Note that in this case we do not impose dynamic constraints, since the meshing between gears can be seen as a purely kinematic/geometric problem. Another distinction from other

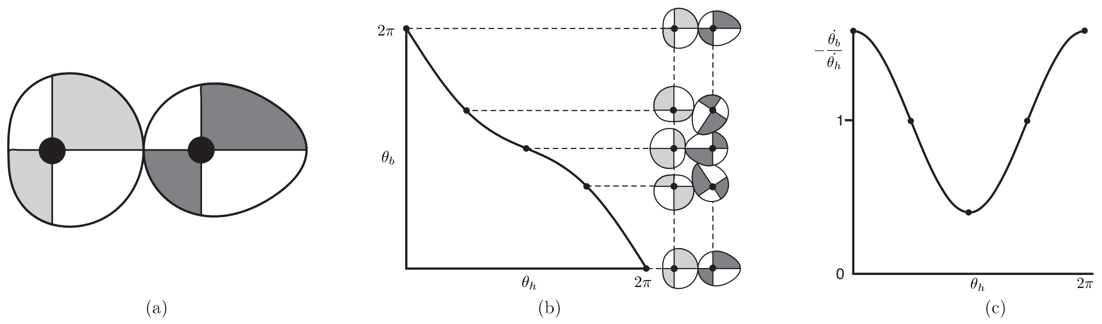


Fig. 9 (a) Pair of pitch curves with a desired transmission profile (b), and transmission ratio $h(\theta_1) = \frac{\dot{\theta}_2}{\dot{\theta}_1}$ (c).

problems, is that in this case we are looking for both the shape of hand and object (both gears).

Figure 8 shows the obtained shapes corresponding to the section of a single geartooth for pressure angles $\alpha = (.5, .6, .7, .8, .9) \cdot \frac{\pi}{2}$ and gear ratio $r = 1.5$. These curves can then be assembled into entire gear profiles. The resulting profiles are closely aligned with the expected analytical result corresponding to involute gears, also depicted in Figure 8.

It should be noted that the rolling constraint is only valid when the contact and pitch points coincide, and is approximately true in the vicinity of this condition. Involute gears with higher pressure angles are less eccentric, so a larger portion of the contact interaction is spent nearby the pitch point. In this case, the rolling contact constraint is approximately true, and can be used in the optimization. However, as the pressure angle decreases, the involute becomes more eccentric, resulting in a larger portion of the contact interaction being spent further away from the pitch point. In this case, the approximation of rolling contact no longer holds, causing the optimization to break down. As a result, the optimization becomes unstable when trying to generate involute profiles with pressure angles below $45°$. For reference, the pressure angles used in real gears are closer to $20°$. To generate such a profile, we would need to change the constraints of the optimization, and relax the rolling with no slip constraint.

7.3 Pitch Curve of non-Circular Gears

Every planar gear is characterized by a *pitch curve*, an imaginary smooth curve that defines its perimeter. The pitch curves of two meshing gears are in rolling contact as the gears rotate. For a circular gear, the pitch curve is a circle.

A classic problem in noncircular gear design is that of finding a pair of pitch curves $R_1(\theta_1), R_2(\theta_2)$ for meshing gears with a given transfer function $h(\theta_1) = \frac{\dot{\theta}_2}{\dot{\theta}_1}$, and center distance l. The typical approach [25] is to limit the problem to pitch lines that contact along the center line, in which case:

$$R_1(\theta_1) + R_2(\theta_2(\theta_1)) = l, \;\; h(\theta_1) = \frac{R_1(\theta_1)}{R_2(\theta_2(\theta_1))}$$

due to contact and rolling constraints. The solution to these equations is then:

$$R_1(\theta_1) = \frac{Lh(\theta_1)}{1 + h(\theta_1)}, \;\; R_2(\theta_2(\theta_1)) = \frac{L}{1 + h(\theta_1)}$$

Alternatively, we formulate the problem with the proposed framework, and, if desired, remove the above limitation. In this case, the hand and object shapes are the pitch curves. If we want to design a pair of pitch curves with transfer function $h(\cdot)$, the solution should satisfy:

- **Kinematic:** contact, tangency and rolling.
- **Fixed decision variables:**
  - The driving gear rotates about the origin $\mathbf{p}_h(t) = \mathbf{0}$.
  - The driven gear rotates about the point $\mathbf{p}_b(t) = (l, 0)$.
  - The orientations of both gears are $\theta_h(t) = \omega t$ and $\theta_b(t) = -H(\omega t)$, where $H'(\theta) = h(\theta)$.
- **Periodic boundary constraints:** We impose that the contact point resets after a full rotation of the gear $s_h(0) \equiv s_h(t_f) \text{mod } N_h$ and $s_b(0) \equiv s_b(t_f) \text{mod } N_b$.

Figure 9 shows an example with transfer function $h(\theta) = 1 + \frac{1}{1.707} cos(\theta)$ (the same as an example in [11]). The resulting pitch curves align closely with the analytical result.

8 Dynamic Throwing

Inspired by the jai alai cesta in Figure 1, we set to design and implement a planar 1DOF thrower. Optimal throwing, in the context of a rotating paddle, requires an agreement between shape and trajectory. To do so, we look for a combination of shape and throw trajectory that maximizes the distance travelled by the ball before hitting the ground. To evaluate this distance, we assume that the ball is released from the hand at the end of the throw, and then follows a ballistic motion (without drag) until it hits the ground. Thus, the throwing distance is a function of the position and velocity of the ball at the end of the throw trajectory.

There are several potential degenerate solutions to this optimization problem. Intuitively, the throwing distance should increase with the speed of the throwing motion and the length of the throwing arm. Thus, a naively constrained optimization should result in an exceedingly long throwing arm paired with an exceedingly fast throwing motion, solutions that are far beyond the capabilities of the hardware. To avoid such solutions, we impose some regularization constraints:

- To prevent arbitrarily large throwing speeds, the angular acceleration of the hand is bounded by $\bar{\alpha}$.
- To bound the size of the throwing arm, we fix the ball to start at rest at a given location A, while constraining the hand to release the ball within radius l of the origin.
- The friction constraint helps to regularize the solution, since faster throws are more likely to result in slipping.

In summary, we define the problem of optimizing the distance the ball travels before hitting the ground using the following constraints:

- **Kinematic:** contact, tangency and rolling.
- **Dynamic:** inertia and friction.
- **Fixed decision variables:**
  - The object is a ball $\mathbf{c}_b(s) = (l\cos(s), l\sin(s))$.
  - The hand pivots about the origin: $\mathbf{p}_h(t) = \mathbf{0}$.
  - Bounded angular acceleration: $|\ddot{\theta}_h(t)| \leq \bar{\alpha}$.
- **Initial and endpoint constraints:**
  - The system starts at static equilibrium as in Equation 10-Equation 13.
  - The ball starts at $\mathbf{p}_b(t_0) = A$.
  - The ball leaves the hand within a radius $|\mathbf{p}_b(t_f)| = l$.

We have found that in general, for the solver to generate a solution that satisfies the constraints of motion, it must be initialized with a guess that also satisfies these constraints. Here, this is accomplished by simulating the system with a trivial arm shape (a straight line) for some throwing motion, then converting the simulation results into an initial guess. By choosing the initial guess for the shape and motion of the throwing arm, we can prod the optimizer to produce either an overhand or underhand throw.

For the overhand throw, the result of the optimizer, as seen in Figure 11, is a hand shape that is concave up near the center, and concave down near the tip. For the underhand throw, the resulting shape, seen in Figure 12, is concave down near the center, and concave up near the tip. In both cases, the throw consists of an initial downward dip to get the ball rolling using gravity, followed by an upward flick to fling the ball forward.

Experiments

The experimental setup consists of a motor attached to a rigid base, as seen in Figure 10. The computed hand shapes for underhand and overhand throwing are fused into a two-ended lasercut hand profile. The motor controller (Galil DMC 4020) has position/velocity tracking functionality, allowing the system to execute the computed trajectories. The resulting motion was captured with a high speed camera. We also use a Vicon motion tracking system to measure $\mathbf{p}_b(t)$. For our experiments, we had the system execute 30 underhand throws and 30 overhand throws. It should be noted that there is no feedback and the initial positions were set manually. We see that the predicted motion of the ball generated by the optimization aligns closely with the motion of the ball measured by the tracking system, as seen in Figure 11 and Figure 12. This suggests that the kinematic and dynamic constraints used in the optimization are valid.

9 Discussion

Solver

Our implementation uses SNOPT [6] for solving the presented nonlinear programs. Out of a handful solvers we have tried, this is the one that we have found to consistently converge to a feasible solution for the presented optimization problems.

Multiple contact modes

This work assumes rolling/sticking contact, and does not currently permit other contact modes (sliding, impact etc.). This is a limitation that would be great to alleviate in the fu-

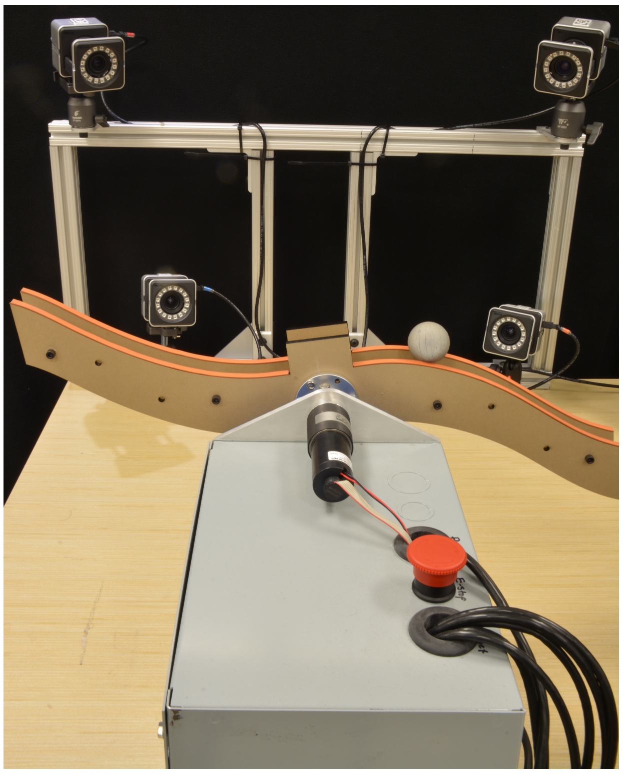


Fig. 10 The hardware setup. The palm is attached to a motor which is mounted to the control box. Four vicon cameras are used to track the motion of the ball.

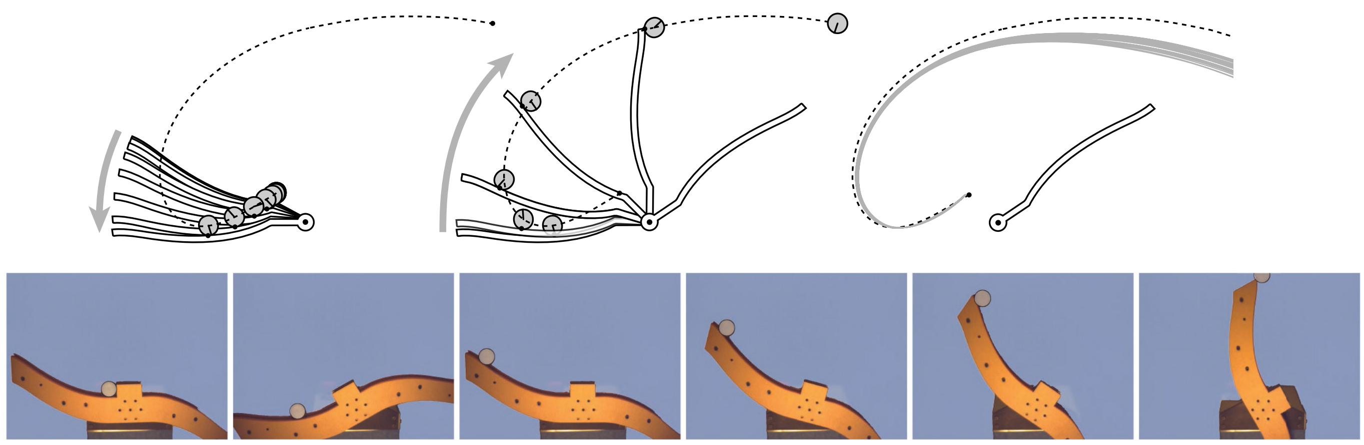


Fig. 11 The optimal overhand throw obtained is composed of two phases: a first gentle inclination where gravity accelerates the ball, followed by a fast upward stroke. The stream of pictures shows a real throw and the right most figure shows the ball trajectory spread over 30 throws.

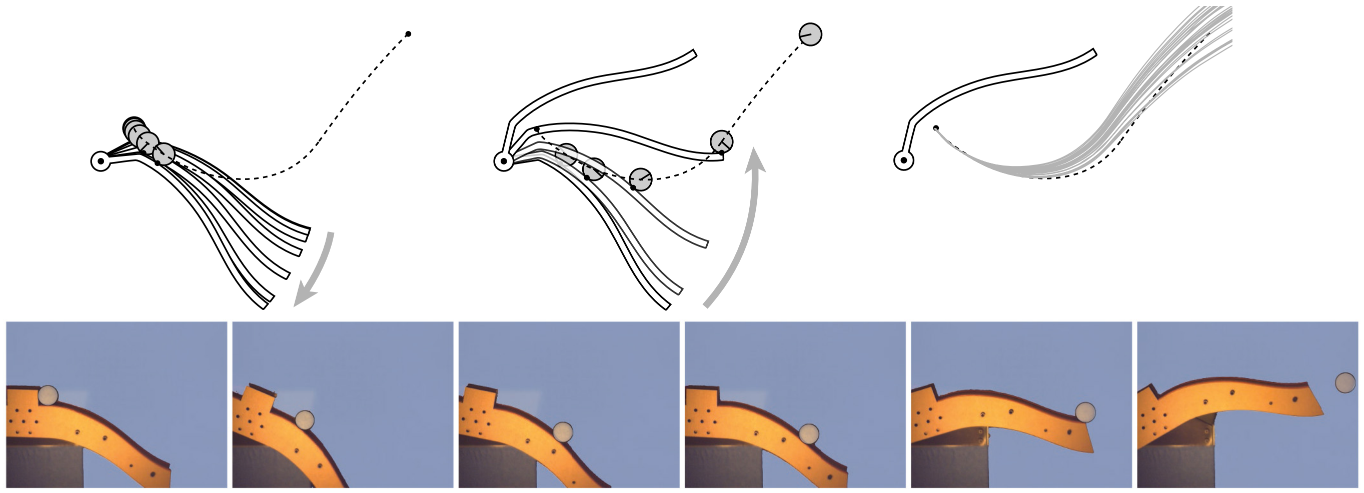


Fig. 12 The optimal underhand throw we obtain is composed of two phases: a first gentle inclination where gravity accelerates the ball, followed by a fast upward stroke. The stream of pictures shows a real throw and the right most figure shows the ball trajectory spread over 30 throws.

ture by exploring complementarity formulations. The transition between contact modes often corresponds to discontinuities in the system state, which our current formulation cannot handle, because we represent the entire trajectory of each state variable using a single spline, which is necessarily smooth. To remedy this, we would need to change our representation to one that is piecewise smooth.

Extension to three dimensions

This approach could be generalized to 3d, but would require extra regularization. Our framework generates shapes and motions by applying many local motion constraints along a trajectory. The result is that the geometry of the end-effector is constrained along the path of the contact point between it and the object. In 2d, where the shape of the end-effector is a contour, this sufficiently constrains its shape. However, in 3d, where the shape of the end-effector is a surface, this means that the majority of the end-effector geometry has no effect on its interaction with the object, and is therefore indeterminate. This could be remedied by either optimizing for several trajectories simultaneously or including a set of well designed regularization constraints while still only optimizing for a single trajectory.

Fitness objective vs. constraint satisfaction

Posing some design objectives (i.e. travel time) as an optimization cost often results in local minima. We fix this by instead imposing the design objective as a constraint that iteratively increases or decreases. In the present implementation, this requires human supervision to determine the sequence of constraint values.

Collocation

Our implementation of collocation differs from the norm in a few key ways. First, the explicit equations of motion $\ddot{x} = f(x, \dot{x}, t)$ do not appear in the collocation constraints. In fact, the accelerations of several motion variables do not appear in any of the optimization constraints. Instead, most of the equations of motion are expressed in an implicit, integrated form (i.e. the kinematic constraints of motion), a property specific but not limited to rolling contact problems.

As a result, the optimization constraints are more compact, and are therefore easier to implement.

Furthermore, the number of basis functions used for each decision variable and the number of collocation points can be chosen independently of one another. This means that resolution of each decision variable can be tuned independently. The number of collocation points can then be tuned so that the optimization does not become too over/under-constrained. In a typical collocation method, each component of the trajectory has the same number of basis functions, which allows the collocation points to be placed to maximize the integration accuracy of the method. We have dropped these integration accuracy guarantees in favor of the flexibility and ease of implementation our method has to offer. However, we plan to explore more exact collocation methods in the future.

It should also be noted that since the same set of motion constraints are applied at each of the collocation points, adding or removing a collocation point results in the addition or removal of more than one constraint from the optimization. Many of the design problems we present are constraint satisfaction problems. In these cases, it would be ideal for the number of constraints to match the number of decision variables. However, because each collocation point corresponds to multiple constraints, it is generally not possible to exactly match the number of constraints with the number of decision variables by adding or removing collocation points. In practice, we deal with this by using enough collocation points to slightly over-constrain the problem, and include a small tolerance on equality constraints so the solver can find a feasible solution.

Optimization Initialization

As mentioned in Section 8, for some problems, it is necessary to initialize the optimization with a guess that satisfies both the kinematic and dynamic constraints of motion. i.e., that is feasible. To generate a valid initial guess, we simulate the system for some trivial geometry and control input. In the case of the toy problems and brachistochrone, this was simply a ball rolling down a flat slope. In the case of the noncircular pitch curve design problem, the initial guess was a pair of identical circular gears rotating at constant angular velocity. In the case of dynamic throwing, the initial guess is the simulation of a straight throwing arm executing a simple throwing motion.

Problem precision and solution sensitivity

We use intuition and trial and error to determine the number of control points and collocation points. This is far from idea. Some regions of the solution space require higher resolutions than others, and high resolution discretizations require more accurate initial guesses for solution convergence. The solution is also sensitive to the number of collocation and control points, the relative scaling of costs vs. constraints, and the initial guess. A potential solution is to use multiscale optimization techniques that automatically increase resolution where necessary.

Shape regularization

There are two main reasons why shape regularization is necessary. First, the shape function has many extra degrees of freedom. For instance, $\mathbf{c}_h(s)$ and $\mathbf{c}_h(f(s))$ describe the same contour for all monotonically increasing functions f. This could be resolved if s corresponded to arc-length, however this is difficult to implement in practice. Without regularization, the control points representing shape tend to spread out unevenly, clustering too closely together in some regions, resulting in poor solutions. Second, any sufficiently general representation of shape is capable of self-intersections. Our formulation, which relies on local constraints, cannot prevent self-intersections, which are a global feature. The regularization constraints described by Equation 15 and Equation 16 are designed to address these issues.

We implement Equation 15 by constraining the control points $(\alpha_{xi}, \alpha_{yi})$ of the shape contour to live along specific vertical lines that are chosen beforehand, as seen in Figure 13. This fixes the control points of the x-spline, α_{xi}, while the y-spline of the shape is free to vary, halving the total degrees of freedom of the shape in the optimization. In this case, the parametric curve can be thought of as a function of x, and thus has no self-intersections. We found that this constraint was effective both for the toy problems and dynamic throwing.

We implement Equation 16 by constraining the control points $(\alpha_{xi}, \alpha_{yi})$ of the shape contour to live along specific lines that radiate from the origin:

$$0 = -\sin(\beta_i)\alpha_{xi} + \cos(\beta_i)\alpha_{yi} \tag{22}$$

as seen in Figure 14. Like the previous constraint, this also halves the total degrees of freedom of the shape in the optimization. Here, the parametric curve can be thought of as polar function, and thus has no self-intersections. We found that this constraint was effective for computing both the involute profile and the noncircular pitch curves.

Though these regularization constraints may be effective, they require human supervision. We plan to explore alternative shape representations that reduce dimensionality, and to develop heuristics for finding more natural regularization constraints of a given task. One specific shape representation that we'd like to explore is to represent the end-effector shape as a single function in polar coordinates $(r_h(\theta), \theta)$ instead of as a parametric curve in cartesian coordinates $(x_h(s), y_h(s))$. This would eliminate extra degrees of freedom in the representation and reduce the number of self-intersections of the curve.

Human Supervision

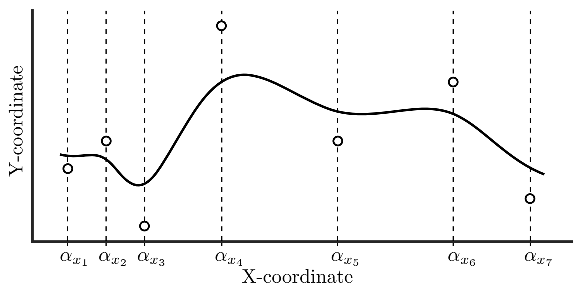


Fig. 13 An example of shape regularization constraint (15). Here, the control points of the shape contour $(\alpha_{xi}, \alpha_{yi})$ are constrained to specific vertical lines.

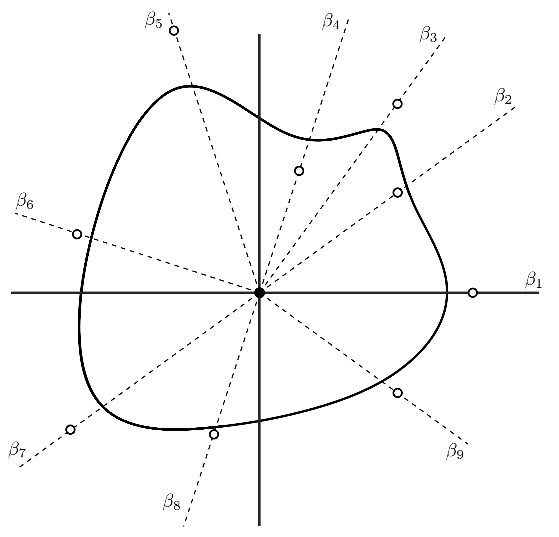


Fig. 14 An example of shape regularization constraint (16). Here, the control points of the shape contour $(\alpha_{xi}, \alpha_{yi})$ are constrained to lines passing through the origin with angle β_i: $-\sin(\beta_i)\alpha_{xi} + \cos(\beta_i)\alpha_{yi} = 0$

Currently, our method requires a fair amount of human supervision to be used effectively. Specifically, the person implementing the method must:

- Choose the number basis functions used to represent each trajectory component.
- Choose the number of collocation points.
- Design the regularization constraints.
- Choose an initial guess for the solver.

We have found that the final result is sensitive to these choices, and that these "settings" vary from problem to problem. In the future, we will study how to automate this process, which will help reduce the dependence of domain specific knowledge to implement our method. For instance, when choosing the number of basis functions, it would be useful to incorporate a method that slowly increases the resolution of each spline until it is sufficient for describing the desired behavior in a given trajectory.

Future work

This work is focused on generating a shape/motion pair for a given set of design constraints. Our framework generates only one motion for a corresponding shape, and that motion corresponds to an open loop control input. In the future, we would like to extend our framework to design shape/motion pairs that facilitate the integration of a closed loop control system. Specifically, end-effector shapes and control inputs that might increase either the controllability or stability of the desired manipulation task. One application would be the development of a throwing arm that could consistently and accurately hit a target. Another possible direction along these lines is to design shape/motion pairs that are robust to variations in the system parameters like the mass and radius of the ball or the coefficient of friction between the ball and the hand.

We would also like to extend our framework so that it generates end-effector shapes with a multitude of motions in mind. For instance, the design of a catching arm should take multiple catching motions into account, because the incoming ball could approach from a wide range of projectile arcs, and could have a variety of initial angular velocities.

References

1. Becker A, Bretl T (2012) Approximate steering of a plate-ball system under bounded model perturbation using ensemble control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 5353–5359
2. Brokowski M, Peshkin M, Goldberg K (1995) Optimal curved fences for part alignment on a belt. Transactions-America Society of Mechanical Engineers Journal of Mechanical Design 117:27–35
3. Caine M (1994) The design of shape interactions using motion constraints. In: IEEE International Conference on Robotics and Automation (ICRA), pp 366–371
4. Chen PT (2010) Simulation and optimization of a two-wheeled, ball-flinging robot. Masters thesis, University of California, San Diego
5. Cristescu A, Cristescu B, Laurenia A (2014) Generalization of Multispeed Gear Pitch Curves Design. Applied Mechanics and Materials 659:559–564
6. Gill PE, Murray W, Saunders MA (2005) SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM review 47(1):99–131
7. Goss VGA (2013) Application of analytical geometry to the form of gear teeth. Resonance 18(9):817–831
8. Ha S, Coros S, Alspach A, Kim J, Yamane K (2017) Joint Optimization of Robot Design and Motion Parameters using the Implicit Function Theorem. In: Robotics: Science and Systems
9. Hargraves CR, Paris SW (1987) Direct trajectory optimization using nonlinear programming and collocation. Journal of Guidance, Control, and Dynamics 10(4):338–342

10. Lippiello V, Ruggiero F, Siciliano B (2016) The effect of shapes in input-state linearization for stabilization of nonprehensile planar rolling dynamic manipulation. IEEE Robotics and Automation Letters 1(1):492–499
11. Litvin FL, Fuentes-Aznar A, Gonzalez-Perez I, Hayasaka K (2009) Noncircular Gears: Design and Generation. Cambridge University Press, New York, New York
12. Liu JY, Chen YC (2008) A design for the pitch curve of noncircular gears with function generation. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol 2
13. Lynch KM, Mason MT (1999) Dynamic nonprehensile manipulation: Controllability, planning, and experiments. The International Journal of Robotics Research 18(1):64–92
14. Lynch KM, Shiroma N, Arai H, Tanie K (1998) The roles of shape and motion in dynamic manipulation: The butterfly example. In: IEEE International Conference on Robotics and Automation (ICRA), pp 1958–1963
15. Montana DJ (1988) The kinematics of contact and grasp. The International Journal of Robotics Research 7(3):17–32
16. Posa M, Cantu C, Tedrake R (2014) A direct method for trajectory optimization of rigid bodies through contact. The International Journal of Robotics Research 33(1):69–81
17. Reist P, D'Andrea R (2009) Bouncing an unconstrained ball in three dimensions with a blind juggling robot. In: IEEE International Conference on Robotics and Automation (ICRA), pp 1774–1781
18. Reist P, D'Andrea R (2011) Design of the pendulum juggler. In: IEEE International Conference on Robotics and Automation (ICRA), pp 5154–5159
19. Rodgers E (1946) Brachistochrone and tautochrone curves for rolling bodies. American Journal of Physics 14(4):249–252
20. Rodriguez A (2013) Shape for Contact. Phd thesis, CMU-RI-TR-13-21, Carnegie Mellon University
21. Rodriguez A, Mason MT (2012) Grasp Invariance. The International Journal of Robotics Research 31(2):237–249
22. Rodriguez A, Mason MT (2013) Effector Form Design for 1DOF Planar Actuation. In: IEEE International Conference on Robotics and Automation (ICRA), pp 349–356
23. Ryu JC, Ruggiero F, Lynch KM (2012) Control of nonprehensile rolling manipulation: Balancing a disk on a disk. In: IEEE International Conference on Robotics and Automation (ICRA), pp 3232–3237
24. Sussmann HJ, Willems JC (1997) 300 years of optimal control: from the brachystochrone to the maximum principle. IEEE Control Systems Magazine 17(3):32–44
25. Yang DC, Tong SH (1998) Generation of identical noncircular pitch curves. Journal of mechanical design 120:337–341
26. Zarbski I, Saaciski T (2008) Designing of non-circular gears. Archive of Mechanical Engineering 55(3):275–292

A Collocation Details

A.1 Motion Variables

Let's consider a motion variable $\xi(t)$ represented using N evenly spaced basis function, in a nonlinear program that makes use of $M+1$ evenly spaced collocation points. If the time length of the trajectory is τ, then the jth collocation point corresponds to time $t_j = \frac{\tau}{M}j$. As described in Section 5, $\xi(t)$ equals:

$$\xi(t) = \sum_{i=1}^{N} \alpha_i \Phi\left(L(t) - i + 2\right) \tag{23}$$

where:

$$L(t) = (N-3)\frac{t}{\tau} \tag{24}$$

$$\dot{L}(t) = \frac{N-3}{\tau} \tag{25}$$

$$\ddot{L}(t) = 0 \tag{26}$$

For the sake of completeness, we list the definition of $\Phi(x)$ and its derivatives:

$$\Phi(x) = \begin{cases} 0| & -\infty \le x \le 0 \\ \frac{1}{6}x^3 + x^2 + 2x + \frac{4}{3}| & -2 \le x \le -1 \\ -\frac{1}{2}x^3 - x^2 + \frac{2}{3}| & -1 \le x \le 0 \\ \frac{1}{2}x^3 - x^2 + \frac{2}{3}| & 0 \le x \le 1 \\ -\frac{1}{6}x^3 + x^2 - 2x + \frac{4}{3}| & 1 \le x \le 2 \\ 0| & 2 \le x \le \infty \end{cases} \tag{27}$$

$$\dot{\Phi}(x) = \begin{cases} 0| & -\infty \le x \le 0 \\ \frac{1}{2}x^2 + 2x + 2| & -2 \le x \le -1 \\ -\frac{3}{2}x^2 - 2x| & -1 \le x \le 0 \\ \frac{3}{2}x^2 - 2x| & 0 \le x \le 1 \\ -\frac{1}{2}x^2 + 2x - 2| & 1 \le x \le 2 \\ 0| & 2 \le x \le \infty \end{cases} \tag{28}$$

$$\ddot{\Phi}(x) = \begin{cases} 0| & -\infty \le x \le 0 \\ x + 2| & -2 \le x \le -1 \\ -3x - 2| & -1 \le x \le 0 \\ 3x - 2| & 0 \le x \le 1 \\ -x + 2| & 1 \le x \le 2 \\ 0| & 2 \le x \le \infty \end{cases} \tag{29}$$

As a quick reminder, this choice of $\Phi(x)$ is special because:

- It is twice differentiable
- It vanishes outside the interval $[-2, 2]$
- It is constructed out of 3rd degree polynomials (which is the minimum degree required for it to be twice differentiable)
- It has evenly spaced knot points.

Differentiating, we find expressions for $\dot{\xi}(t)$ and $\ddot{\xi}(t)$:

$$\dot{\xi}(t) = \sum_{i=1}^{N} \alpha_i \dot{\Phi}\left(L(t) - i + 2\right)\left(\frac{N-3}{\tau}\right) \tag{30}$$

$$\ddot{\xi}(t) = \sum_{i=1}^{N} \alpha_i \ddot{\Phi}\left(L(t) - i + 2\right)\left(\frac{N-3}{\tau}\right)^2 \tag{31}$$

In the optimization, $\xi(t)$ and its derivatives are only evaluated at the collocation points $t_j = \frac{\tau}{M} j$. Substituting this in for t, we find that:

$$\xi_j = \xi(t_j) = \sum_{i=1}^{N} \alpha_i \Phi\left(\frac{N-3}{M} j - i + 2\right) \tag{32}$$

$$\xi'_j = \dot{\xi}(t_j) = \sum_{i=1}^{N} \alpha_i \dot{\Phi}\left(\frac{N-3}{M} j - i + 2\right)\left(\frac{N-3}{\tau}\right) \tag{33}$$

$$\xi''_j = \ddot{\xi}(t_j) = \sum_{i=1}^{N} \alpha_i \ddot{\Phi}\left(\frac{N-3}{M} j - i + 2\right)\left(\frac{N-3}{\tau}\right)^2 \tag{34}$$

It is important to remember that the actual decision variables in the optimization are the weights of the basis functions $\alpha_1 ... \alpha_N$ used to describe each of the continuous decision variables $\xi(x)$, and the inverse of the time length of the trajectory $\frac{1}{\tau}$. Thus, it would be useful to be able to quickly map $\frac{1}{\tau}$ and the vector of basis function weights,

$$A = [\alpha_1, ... \alpha_i, ... \alpha_N]^T \tag{35}$$

to the vectors encoding the values of ξ and its derivatives evaluated at the collocation points:

$$V = [\xi_0, \xi_1, ... \xi_j, ... \xi_M]^T \tag{36}$$

$$V' = [\xi'_0, \xi'_1, ... \xi'_j, ... \xi'_M]^T \tag{37}$$

$$V'' = [\xi''_0, \xi''_1, ... \xi''_j, ... \xi''_M]^T \tag{38}$$

To do this, we define transformation matrices D^0, D^1, D^2:

$$D^0_{ij} = \Phi\left(\frac{N-3}{M} j - i + 2\right) \tag{39}$$

$$D^1_{ij} = \dot{\Phi}\left(\frac{N-3}{M} j - i + 2\right)(N-3) \tag{40}$$

$$D^2_{ij} = \ddot{\Phi}\left(\frac{N-3}{M} j - i + 2\right)(N-3)^2 \tag{41}$$

These matrices are constant for a given choice of (M, N), meaning that they only need to be evaluated once at the start of the optimization. The matrices are also sparse, since $\Phi(x)$,$\dot{\Phi}(x)$,$\ddot{\Phi}(x)$ vanish outside of the interval $[-2, 2]$. It follows that:

$$V = D^0 A, \qquad V' = \frac{1}{\tau} D^1 A, \qquad V'' = \frac{1}{\tau^2} D^2 A \tag{42}$$

We construct transformation matrices for each motion variable, since each motion variable can be represented with a different number of basis functions. This framework gives us the flexibility to tune the resolution of each spline independently, while managing the necessary re-scaling operations to ensure that each motion variable is evaluated across the same time interval.

A.2 Shape Variables

We represent shape as a parametric curve: $(X(s), Y(s))$. Like the motion variables, $X(s)$ and $Y(s)$ are described as a sum of basis functions,

$$X(s) = \sum_{i=1}^{N} \alpha_{xi} \Phi(s - i - k) \tag{43}$$

$$Y(s) = \sum_{i=1}^{N} \alpha_{yi} \Phi(s - i - k) \tag{44}$$

where the basis function Φ is described in Section A.1. Unlike the motion variables, which can have varying resolutions, we use the same number of basis functions to describe the X and Y splines of a given shape. Since the contact parameter $s(t)$ is one of the motion variables in the optimization, it follows that $\Phi(s(t_j) - i - k)$ is not constant during the optimization. Thus, there is no benefit in constructing a set of transformation matrices for X and Y as we did in Section A.1. Instead, we must re-evaluate $\Phi, \dot{\Phi}$ using (27) and (28) each time we compute X, Y and their derivatives.

B Toy Problem Implementation

In this section, we will describe the specifics details of converting the toy problem into a nonlinear program.

B.1 Variable Definitions

For the sake of simplicity, we will assume that each motion variable uses the same number of basis functions. In this case, if the motion variables use N basis functions and the shape variables use L basis functions, then the decision variables in the optimization are the vectors of basis function weights for each continuous decision variable:

$$\bar{\theta}_h = [\bar{\theta}_{h1}, ... \bar{\theta}_{hi}, ... \bar{\theta}_{hN}]^T \tag{45}$$

$$\bar{s}_h = [\bar{s}_{h1}, ... \bar{s}_{hi}, ... \bar{s}_{hN}]^T \tag{46}$$

$$\bar{\theta}_b = [\bar{\theta}_{b1}, ... \bar{\theta}_{bi}, ... \bar{\theta}_{bN}]^T \tag{47}$$

$$\bar{s}_b = [\bar{s}_{b1}, ... \bar{s}_{bi}, ... \bar{s}_{bN}]^T \tag{48}$$

$$\bar{p}_{bx} = [\bar{p}_{bx1}, ... \bar{p}_{bxi}, ... \bar{p}_{bxN}]^T \tag{49}$$

$$\bar{p}_{by} = [\bar{p}_{by1}, ... \bar{p}_{byi}, ... \bar{p}_{byN}]^T \tag{50}$$

$$\bar{c}_{hx} = [\bar{c}_{ch1}, ... \bar{c}_{hxi}, ... \bar{c}_{hxL}]^T \tag{51}$$

$$\bar{c}_{hy} = [\bar{c}_{hy1}, ... \bar{c}_{hyi}, ... \bar{c}_{hyL}]^T \tag{52}$$

We include an additional decision variable $\nu = \frac{1}{\tau}$, where τ is the time length of the trajectory. For the toy problem, we assume that the ball is a circle of radius r, and the hand is constrained to rotate about the origin, meaning that the decision variables $\bar{p}_{hx}, \bar{p}_{hy}, \bar{c}_{bx}, \bar{c}_{by}$ are absent from the optimization.

If ξ_j, ξ'_j, ξ''_j denote the value of a continuous decision variable and its derivatives evaluated at the jth collocation point, then for M+1 collocation points this can be represented in vector form as:

$$\xi = [\xi_0, \xi_1, ..., \xi_j, ... \xi_M]^T \tag{53}$$

$$\xi' = [\xi'_0, \xi'_1, ..., \xi'_j, ... \xi'_M]^T \tag{54}$$

$$\xi'' = [\xi''_0, \xi''_1, ..., \xi''_j, ... \xi''_M]^T \tag{55}$$

$$\bar{\xi} = [\bar{\xi}_1, ..., \bar{\xi}_i, ... \bar{\xi}_N]^T \tag{56}$$

As shown in Section A.1, we see that if $\xi(t)$ is a motion variable, then we can use the transformation matrices D^0, D^1, D^2 to quickly map $\bar{\xi}$ and ν to ξ_j, ξ'_j, ξ''_j:

$$\xi = D^0\bar{\xi}, \qquad \xi' = \nu D^1\bar{\xi}, \qquad \xi'' = \nu^2 D^2\bar{\xi} \tag{57}$$

To evaluate the hand shape and its tangency vector at the jth collocation point, we use the definitions:

$$\begin{bmatrix} c_{hxj} \\ c_{hyj} \end{bmatrix} = \sum_{i=1}^{L} \begin{bmatrix} \bar{c}_{hxi} \\ \bar{c}_{hyi} \end{bmatrix} \Phi(s_{hj} - i - k) \tag{58}$$

$$\begin{bmatrix} c'_{hxj} \\ c'_{hyj} \end{bmatrix} = \sum_{i=1}^{L} \begin{bmatrix} \bar{c}_{hxi} \\ \bar{c}_{hyi} \end{bmatrix} \dot{\Phi}(s_{hj} - i - k) \tag{59}$$

Since the ball is a circle of radius r, the ball shape and its tangecny vector at the jth collocation point are given by:

$$\begin{bmatrix} c_{bxj} \\ c_{byj} \end{bmatrix} = r \begin{bmatrix} \cos(s_{bj}) \\ \sin(s_{bj}) \end{bmatrix} \tag{60}$$

$$\begin{bmatrix} c'_{bxj} \\ c'_{byj} \end{bmatrix} = r \begin{bmatrix} -\sin(s_{bj}) \\ \cos(s_{bj}) \end{bmatrix} \tag{61}$$

B.2 Function Definitions

We define R as a function that maps angle θ to the rotation matrix corresponding to a counterclockwise rotation by θ:

$$R(\theta) = \begin{bmatrix} \cos(\theta), & -\sin(\theta) \\ \cos(\theta) & \sin(\theta), \end{bmatrix} \tag{62}$$

We define $\angle$ as the function that maps the vector $[x, y]^T$ to its polar angle:

$$\angle[x, y]^T = Im\left(\ln(x + yi)\right) \tag{63}$$

B.3 Constraints

At the jth collocation point, we apply the kinematic and dynamic constraints of motion, as described in Section 4.1.

Contact Constraint

$$R(\theta_{hj}) \begin{bmatrix} c_{hxj} \\ c_{hyj} \end{bmatrix} - R(\theta_{bj}) \begin{bmatrix} c_{bxj} \\ c_{byj} \end{bmatrix} - \begin{bmatrix} p_{bxj} \\ p_{byj} \end{bmatrix} = 0 \tag{64}$$

Tangency Constraint

$$\theta_{hj} + \angle R\left(\frac{\pi}{2}\right) \begin{bmatrix} c'_{hxj} \\ c'_{hyj} \end{bmatrix} - \theta_{bi} - \angle R\left(\frac{\pi}{2}\right) \begin{bmatrix} c'_{bxj} \\ c'_{byj} \end{bmatrix} + (2k_j + 1)\pi = 0 \tag{65}$$

Here, k_j is the integer used to deal with the fact that the original tangency constraint is true mod 2π. It is chosen at each constraint evaluation to minimize the constraint error.

Rolling Constraint

$$s'_{hj}\sqrt{\left(c'_{hxj}\right)^2 + \left(c'_{hyj}\right)^2} + s'_{bj}\sqrt{\left(c'_{bxj}\right)^2 + \left(c'_{byj}\right)^2} = 0 \tag{66}$$

Inertia Constraint

$$\left(R(\theta_{bj}) \begin{bmatrix} c_{bxj} \\ c_{byj} \end{bmatrix}\right) \times m \left(\begin{bmatrix} p''_{bxj} \\ p''_{byj} \end{bmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix}\right) - I\theta''_{bj} = 0 \tag{67}$$

Friction Cone Constraint

$$\left(\left(\mu R\left(\frac{\pi}{2}\right) \pm \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \frac{[c'_{hxj}, c'_{hyj}]^T}{\sqrt{\left(c'_{hxj}\right)^2 + \left(c'_{hyj}\right)^2}}\right) \cdot \mathbf{f_{hj}} \geq 0 \tag{68}$$

where:

$$\mathbf{f_{hj}} = R(-\theta_{hj}) m \left(\begin{bmatrix} p''_{bxj} \\ p''_{byj} \end{bmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix}\right) \tag{69}$$

Task Constraints

To constrain the ball to move along a specific path, we use the constraint:

$$G(p_{bxj}, p_{byj}) = 0 \tag{70}$$

If we want to control both the spatial path, and the speed of the ball as it travels, we instead fix the decision variables for the position of the ball and the length of the trajectory:

$$[\bar{p}_{bxi}, \bar{p}_{byi}] = [f_{px}(i), f_{py}(i)] \tag{71}$$

$$\nu = C \tag{72}$$

We can fix the motion of the hand using the constraints:

$$\bar{\theta}_{hi} = f_\theta(i) \tag{73}$$

$$\nu = C \tag{74}$$

If instead we'd like to fix the shape of the hand, we use the constraint:

$$[\bar{c}_{hxi}, \bar{c}_{hyi}] = [f_{cx}(i), f_{cy}(i)] \tag{75}$$