

Dynamic Underactuated Nonprehensile Manipulation

Kevin M. Lynch¹
Biorobotics Division
Mechanical Engineering Laboratory
Namiki 1-2, Tsukuba, 305 Japan

Matthew T. Mason
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Abstract

By exploiting centrifugal and Coriolis forces, simple, low-degree-of-freedom robots can control objects with more degrees-of-freedom. For example, by allowing the object to roll and slip, a one-degree-of-freedom revolute robot can take a planar object to a full-dimensional subset of its state space. We present a dynamic manipulation planner that finds manipulator trajectories to move an object from one state to another without grasping it. The trajectories have been successfully implemented on a one-degree-of-freedom direct-drive arm to perform dynamic tasks such as snatching an object from a table, rolling an object on the surface of the arm, and throwing and catching.

1 Introduction

The utility of a manipulator is measured by the set of tasks it is capable of accomplishing. To determine the capability of a robot, we must consider both properties of the robot and the mechanics laws mapping the robot's actions to changes in the task state.

For parts transfer problems, a particularly useful simplifying assumption is that the part is grasped and follows the end-effector as it moves. With this assumption about the mechanics, we can focus on properties of the robot such as its payload capacity, accuracy, speed, and the geometry of its workspace. In particular, a general positioning manipulator requires at least six degrees-of-freedom, and much work has been invested in the design and construction of sophisticated positioning devices.

Surprisingly, however, little effort has been made to understand the manipulation capabilities of even the simplest robots under more complete mechanics models. By exploiting mechanics, a simple robot may be able to solve the same tasks as a more complex robot. For example, by allowing the manipulated object to roll and slip, a low-degree-of-freedom robot can control more degrees-of-freedom of an object. Because the robot has fewer freedoms, the manipulation is *underactuated*; because the object is not firmly grasped, the contact is *nonprehensile*. With underactuated nonprehensile manipulation, some of the complexity of the robot system is transferred from hardware (joints and actuators) to planning and control.

Our previous work on underactuated nonprehensile manipulation has focused on quasistatic pushing. By analyzing the mechanics of pushing, we have shown

- A two-degree-of-freedom robot (a point translating in the plane) can push an object arbitrarily closely along any path in the object's three-dimensional configuration space, unless the object is a frictionless disk centered at its center of mass (Lynch and Mason [18]).
- A single revolute joint, operating above a fixed-speed conveyor, can move any polygon from any initial configuration on the conveyor to a single goal configuration by using pushing and conveyor drift (Akella *et al.* [3]).

In this paper we study underactuated nonprehensile manipulation using a dynamic model. In Section 4 we show how a one-degree-of-freedom revolute robot, with just a two-dimensional state space, can take a planar object to a six-dimensional subset of its six-dimensional state space. Section 5 studies the *dynamic pick-and-place* problem of planning a manipulator trajectory to take an object from one state to another using nonprehensile contact and friction and dynamic forces. Finally, Section 6 presents experiments with a one-degree-of-freedom direct-drive arm. The arm automatically executes dynamic tasks such as snatching an object from a table, rolling an object on the surface of the arm, and throwing and catching.

This paper summarizes work presented in (Lynch [17]).

2 Related Work

This research has been inspired by industrial parts feeders using dynamic nonprehensile manipulation, such as bowl feeders and the APOS system (Hitakawa [14]). Related research includes dynamic parts orienting on a vibrating plate (Böhringer *et al.* [7], Swanson *et al.* [24]). Other forms of nonprehensile manipulation are parts orienting by tray-tilting (Erdmann and Mason [11]), tumbling (Sawasaki *et al.* [21]), pivoting (Aiyama *et al.* [2]), tapping (Higuchi [13], Huang *et al.* [15]), two pin manipulation (Abell and Erdmann [1]), and two palm manipulation (Erdmann [10], Zumel and Erdmann [27]).

Dynamic underactuated manipulation is similar to the control of underactuated manipulators, except the unactuated freedoms are controlled through unilateral frictional contacts. Research on underactuated manipulators includes that of Oriolo and Nakamura [20] and Arai and Tachi [5]. In work related to ours, Arai and Khatib [4] demonstrated rolling of a cube on a paddle held by a PUMA. Their motion strategy was hand-crafted with the assumption of infinite friction at the rolling contact. In this paper, we automatically find motion strategies which account for finite friction. Our approach to motion planning uses nonlinear optimization, which has also

¹This research was conducted while the first author was a Ph.D. student at the Robotics Institute, Carnegie Mellon University.

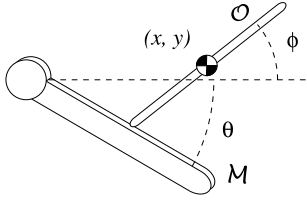


Figure 1: A one-degree-of-freedom frictionless revolute robot manipulating a rod. The robot can control the rod to a six-dimensional subset of its state space. A frictionless *prismatic* joint cannot.

been used by Witkin and Kass [25], Yen and Nagurka [26], and Chen [8] to find motion strategies for fully-actuated systems. Gradient descent approaches to motion planning for underactuated systems have also been proposed by Divilbiss and Wen [9], Fernandes *et al.* [12], and Sussmann [23].

See (Lynch [17]) for other related work.

3 Assumptions

All problems considered in this paper are planar, and the non-prehensile contact between the object and the manipulator is always either a point or line contact. Objects are polygonal. Coulomb friction acts between the robot and the object.

4 Accessibility

The state of a one-degree-of-freedom revolute robot (Figure 1) is given by its angle and angular velocity $(\theta, \dot{\theta})$. The state space of the robot link is a two-dimensional subset of the six-dimensional planar state space, where the four state equality constraints are that the pivot’s position cannot change and its linear velocity must be zero. The natural question is whether these constraints on the state of the robot translate to equality constraints on the state $(x, y, \phi, \dot{x}, \dot{y}, \dot{\phi})$ of the object it is manipulating. In most cases, the answer is no. In other words, by exploiting dynamics, the robot can take the object to a six-dimensional subset of its six-dimensional planar state space.

To show this, we can examine the Lie algebra of vector fields describing the possible motions of the object as a function of the current state and the manipulator control (see Lynch [17]). A one-degree-of-freedom revolute robot can exploit Coriolis and centrifugal forces to achieve control of the object’s six state variables, even with frictionless contact. A one-degree-of-freedom frictionless *prismatic* robot cannot.

In this paper we describe an intuitive approach to controlling the degrees-of-freedom of the object using *algorithmic* control, which is the basis of the planner described in Section 5. The idea behind algorithmic control is to control only a subset of the state variables at any given time, but to switch between subsets such that the goal state is reached. For example, a unicycle can be driven to the zero state by first reorienting it so that it points toward the origin, then driving it to the origin, and finally reorienting it to the goal angle. In general, we must account for the possibility of drift in the variables that are not being directly controlled.

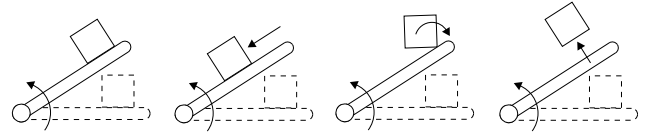


Figure 2: Manipulation phases: dynamic grasp, slip, roll, and free flight.

In the context of dynamic nonprehensile manipulation, we can define the following control phases (Figure 2):

1. *Dynamic grasp* (Mason and Lynch [19]). An object is in a dynamic grasp if it makes line contact with the manipulator and the manipulator accelerates such that the object remains fixed against it. With a dynamic grasp, up to $\min(2n, 6)$ of the object’s state variables can be directly controlled by an n -joint manipulator.
2. *Slip*. Controlled slip provides control of two state variables, the slipping distance and the slipping velocity.
3. *Roll*. Rolling provides control of two state variables, the rolling angle and angular velocity.
4. *Slip and roll*. Slip and roll occur simultaneously, giving control of up to four state variables.
5. *Free flight*. After the object is released, it follows a one-dimensional path through its state space, parameterized by its time of flight.

A control algorithm can sequence these phases. The dimension of the accessible state space is the sum of the independent freedoms of each phase, up to a maximum of six.

Example. An n -degree-of-freedom manipulator carries an object with a dynamic grasp, allows it to begin rolling, and then releases it. The dimension of the accessible state space of the object is upper-bounded by $\min(2n + 2 + 1, 6)$. Roughly speaking, the “controls” are the state of the object at the onset of rolling, the roll angle and velocity at release, and the time of flight (assuming the arrival time at the goal state is unimportant).

5 Planning

For a given initial state of the object and the manipulator, the planning problem is to find a manipulator trajectory to take the object to the goal state using frictional, gravitational, and dynamic forces. We are especially interested in the following dynamic tasks:

1. *Snatch*: transfer an object initially at rest on a table to rest on the manipulator. The manipulator accelerates into the object, transferring control of the object from the table to the manipulator.
2. *Throw*: throw the object to a desired goal state. The object is carried with a dynamic grasp and released instantaneously (no slipping or rolling) at a point where the free-flight dynamics will take the object to the goal state (possibly a catch).

3. *Roll*: roll a polygonal object sitting on the manipulator from one statically stable edge to another statically stable edge.
4. *Rolling throw*: allow the object to begin rolling before throwing it. By controlling the roll angle and velocity before the release, the dimension of the object’s accessible state space is increased by two.

The manipulation phases in these tasks are dynamic grasp, roll, and free flight. Slipping contact is not used.

To solve these problems we cast trajectory planning as a constrained nonlinear optimization problem, where the system’s initial state and goal state (or state manifold) are specified as constraints to the optimization. The trajectory is also subject to a set of nonlinear equality and inequality constraints arising from constraints on the manipulator motion and the dynamics governing the object’s motion relative to the manipulator. Because dynamic nonprehensile manipulation relies on friction between the object and the manipulator, and friction coefficients are often uncertain and varying, the optimization is usually asked to minimize the required friction coefficient for successful manipulation. Unlike other work on optimizing the time or energy of a robot’s motion, we are more concerned with making the manipulation maximally robust to variations in the friction coefficient.

5.1 Problem Specification

Every task is assumed to consist of a sequence of manipulation phases made up of one or more of the following: a (dynamic) grasp phase g , a roll phase r , and a flight phase f . With this notation, a throw (as defined above) is denoted gf , a roll is denoted grg , and a rolling throw is denoted grf . A snatch can be either g or rg . We assume that there is no rebound from the impact at the end of a roll (the transition from r to g). The instant the new edge contacts, if the dynamic grasp conditions are met, then the object is assumed to be in a dynamic grasp.

The times of the manipulation phases are t_{g1} for the first dynamic grasp phase g , t_{roll} for the rolling phase r , t_{g2} for the second g phase, and t_{flight} for the flight phase f . If a phase is omitted, its corresponding time duration is zero. We also define the running times $T_{g1} = t_{g1}$, $T_{roll} = T_{g1} + t_{roll}$, $T_{g2} = T_{roll} + t_{g2}$, $T_{flight} = T_{g2} + t_{flight}$. Finally, we define $T = T_{g2}$, where T is the total time the manipulator is in contact with the object during the manipulation.

The primitive manipulations—the snatch, the roll, the throw, and the rolling throw—can be composed or “glued together” to form more complex manipulations. The glue between primitives is a static equilibrium carry of the object.

In the next three subsections we describe the three fundamental elements of the nonlinear program: the design variables, the constraints, and the objective function.

5.1.1 Design Variables

The design variables consist of the variables \mathbf{x} specifying the trajectory of the manipulator over the interval $[0, T]$; the times of each phase of the manipulation, t_{g1} , t_{roll} , t_{g2} , and t_{flight} (some subset of these will be applicable based on the

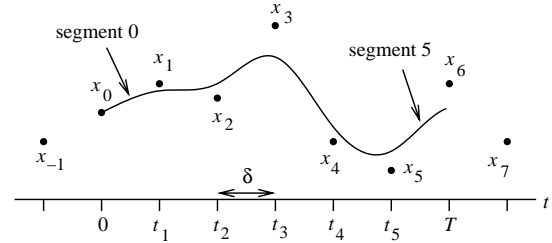


Figure 3: A cubic B-spline joint trajectory with nine knot points and six segments.

problem specification); and the required friction coefficient μ between the manipulator and the object. These variables are not independent; the trajectory of the manipulator implicitly defines the time of each phase. However, the problem formulation is much simpler if we make each of these variables explicit and constrain them to be dynamically consistent. Although we cannot control the friction coefficient μ , it is convenient to represent μ as a design variable and explicitly enforce the resulting friction constraints.

Many different finite parameterizations of manipulator trajectories have been explored, including polynomials, Fourier bases, summed Fourier and polynomial functions, splines, and piecewise constant acceleration segments. After some experimentation, we decided to represent trajectories as uniform cubic B-splines (Bartels [6], Chen [8]).

For an n -joint robot, $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n)$, where \mathbf{x}^i is the vector of knot points for the cubic B-spline position history of joint i . The time of each knot point \mathbf{x}_j^i is given by t_j , and the knot points are evenly spaced in time. The position of the joint passes near the knot points; the actual position at each time is obtained by taking a weighted sum of the four knot points which are closest in time. The weighting basis functions are cubic polynomials of time. Therefore, the position is C^2 and piecewise cubic, the velocity is C^1 and piecewise quadratic, and the acceleration is C^0 and piecewise linear (constant jerk segments). See Figure 3.

5.1.2 Constraints

Constraints arise from limitations on the motion of the manipulator and constraints on the motion of the object. The latter are determined by the inequality constraints of Coulomb friction and the equality constraints of Newton’s laws. To simplify the notation, the dependencies of the constraints on the design variables is omitted.

Manipulator constraints

1. Position constraints $t \in [0, T]$

$$\mathbf{p}(\Theta(t)) \leq \mathbf{0},$$

where Θ is the arm configuration and \mathbf{p} is a vector-valued function representing joint limits and obstacles.

2. Joint velocity constraints $t \in [0, T]$

$$\dot{\Theta}_{min} \leq \dot{\Theta}(t) \leq \dot{\Theta}_{max}$$

3. Joint torque constraints $t \in [0, T]$

$$\tau_{min} \leq \tau(t) \leq \tau_{max},$$

where $\tau(t)$ is the torque to move the arm and object along the trajectory.

4. Initial state constraints

$$\Theta(0) = \Theta_0, \dot{\Theta}(0) = \dot{\Theta}_0$$

Object constraints During a dynamic grasp or rolling phase, contact friction constraints must be enforced to prevent slipping or breaking contact. These force constraints encode the unilateral nature of contact (forces can only be applied into the object) and the finite friction coefficient μ .

To maintain a dynamic grasp, the sum of the negated gravitational vector $-\mathbf{g}$ and the manipulator's acceleration, measured in the object frame, must fall inside an acceleration cone \mathcal{A} . The cone \mathcal{A} is determined by the line contact and the friction coefficient μ , and it lives in the three-dimensional space of body-centered accelerations (two linear components and one angular component). The cone is bounded by four edges, $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$, numbered so that the interior of the cone lies to the left as we move from \mathbf{a}_1 to \mathbf{a}_2 , etc. (These edges correspond to the accelerations of the object from forces through the endpoints of the line contact and on the boundaries of the friction cone.) The magnitude of these vectors is not important, but for simplicity assume they are unit vectors. We now form the 3×4 matrix

$$\mathbf{A}_g = (\mathbf{a}_2 \times \mathbf{a}_1 \mid \mathbf{a}_3 \times \mathbf{a}_2 \mid \mathbf{a}_4 \times \mathbf{a}_3 \mid \mathbf{a}_1 \times \mathbf{a}_4),$$

where each column of \mathbf{A}_g is an outward-pointing normal to a face of the acceleration cone \mathcal{A} . The acceleration of the manipulator must have a nonpositive dot product with each column vector of \mathbf{A}_g .

1. Dynamic grasp constraints (dynamic grasp phase)

$$\mathbf{A}_{gi}^T (\dot{\mathbf{J}}_{gi}(\Theta(t)) \dot{\Theta}(t) + \mathbf{J}_{gi}(\Theta(t)) \ddot{\Theta}(t) - \mathbf{g}) \leq 0$$

where i is 1 or 2, depending on the current dynamic grasp phase. During a dynamic grasp phase, \mathbf{J}_{gi} and $\dot{\mathbf{J}}_{gi}$, the manipulator Jacobian and its time derivative, are measured at the center of mass of the object, so that all accelerations can be represented in the object frame.

During rolling contact, a single point of the object is in contact with the manipulator, so \mathcal{A} is a planar acceleration cone (now measured in the world frame), bounded by the two edges \mathbf{a}_l and \mathbf{a}_r (from forces on the left and right edges of the friction cone, respectively). Define the vector \mathbf{a}_\perp normal to the plane of this cone: $\mathbf{a}_\perp = \mathbf{a}_r \times \mathbf{a}_l$. We form the matrix $\mathbf{A}_r = (\mathbf{a}_\perp \times \mathbf{a}_l \mid \mathbf{a}_r \times \mathbf{a}_\perp)$, where each column of \mathbf{A}_r lies in the $(\mathbf{a}_l, \mathbf{a}_r)$ plane and is an outward-pointing normal to an edge of the acceleration cone. The acceleration of the object must have a nonpositive dot product with each column vector of \mathbf{A}_r .

The object acceleration $\mathbf{a}_{roll} = (a_{roll,x}, a_{roll,y}, \alpha_{roll})^T$ (measured at its center of mass) required to maintain the rolling contact is determined by assuming a pin joint at the contact and finding the object acceleration consistent with the motion of the manipulator. The acceleration $\mathbf{a} = (a_x, a_y, \alpha)^T$ of the contact point on the robot (including negated gravity) is given by

$$\mathbf{a} = \dot{\mathbf{J}}_r(\Theta(t)) \dot{\Theta}(t) + \mathbf{J}_r(\Theta(t)) \ddot{\Theta}(t) - \mathbf{g},$$

where \mathbf{J}_r and $\dot{\mathbf{J}}_r$ are measured at the contact point on the manipulator. The constraint that the linear acceleration (a_x, a_y) matches the acceleration of the contact point on the object is expressed

$$(a_x, a_y) = -\omega^2 \mathbf{r} + (-r_y \alpha_{roll}, r_x \alpha_{roll}) + (a_{roll,x}, a_{roll,y}),$$

where ω is the angular velocity of the object, $\mathbf{r} = (r_x, r_y)$ is the vector from the CM of the object to the contact point in the world frame. This gives us two constraints on the object acceleration \mathbf{a}_{roll} ; the third is that \mathbf{a}_{roll} must result from a force through the contact point:

$$\alpha_{roll} = \frac{1}{\rho^2} (r_x a_{roll,y} - r_y a_{roll,x}),$$

where ρ is the object's radius of gyration. After a little manipulation, we get

$$\begin{aligned} a_{roll,x} &= \frac{a_x(\rho^2 + r_x^2) + a_y r_x r_y + r_x \omega^2(\rho^2 + r_x^2 + r_y^2)}{\rho^2 + r_x^2 + r_y^2} \\ a_{roll,y} &= \frac{a_y(\rho^2 + r_y^2) + a_x r_x r_y + r_y \omega^2(\rho^2 + r_x^2 + r_y^2)}{\rho^2 + r_x^2 + r_y^2} \\ \alpha_{roll} &= \frac{r_x a_{roll,y} - r_y a_{roll,x}}{\rho^2}. \end{aligned}$$

Now we can write the rolling friction constraints.

2. Rolling friction constraints (rolling phase only)

$$\mathbf{A}_r^T \mathbf{a}_{roll} \leq 0$$

3. Roll angle constraints (rolling phase only)

$$\psi_{min} \leq \psi(t) \leq \psi_{max},$$

where ψ is the angle of the object relative to the manipulator. This constraint prevents the object from penetrating the manipulator during the rolling phase.

4. Roll completed constraint (for rolls only)

$$\psi(T_{roll}) = \psi_{goal}$$

5. Release state constraints (for throws only)

$$\mathbf{s}(\mathbf{q}, \dot{\mathbf{q}}, t_{flight}) = 0$$

These constraints specify that the object reaches the goal submanifold by free flight, where $(\mathbf{q}, \dot{\mathbf{q}})$ is the release state and t_{flight} is the time of flight. The goal submanifold is usually specified by goal values of some subset of the state variables.

In principle, the manipulator constraints (1)–(3) and object constraints (1)–(3) should be satisfied at all times during their domain of applicability. In practice, the optimization can only handle a finite number of constraints. For this reason, the constraints are only enforced at p uniformly-sampled points during each manipulation phase. In the examples here, p is chosen between 20 and 50.

5.1.3 Objective Function

In most of our problems, we minimize the required friction coefficient μ between the object and manipulator.

5.2 Sequential Quadratic Programming

Sequential quadratic programming (SQP) is used to solve the nonlinear program. SQP is a generalization of Newton’s method for unconstrained optimization in that it finds a step away from the current iterate by minimizing a quadratic model of the problem. At each iteration, SQP determines the direction to step by solving a quadratic subprogram, where the objective function is a quadratic approximation at the current point and nonlinear constraints are linearized.

The constrained nonlinear program can be written

$$\begin{aligned} \min f(x) \\ \text{subject to } c_i(x) &\leq 0, \quad i \in \mathcal{I} \\ c_i(x) &= 0, \quad i \in \mathcal{E} \end{aligned}$$

where $x \in \mathbf{R}^m$ is the iterate, f is the objective function, each c_i is a constraint mapping \mathbf{R}^m to \mathbf{R} , and \mathcal{I} and \mathcal{E} are index sets for inequality and equality constraints, respectively. The Lagrangian function is defined

$$L(x, \lambda) = f(x) + \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i c_i(x)$$

where the λ_i are the Lagrange multipliers. At each iterate, the direction of the step is computed by a quadratic programming subproblem of the form

$$\begin{aligned} \min \nabla f(x_k)^T d + \frac{1}{2} d^T H_k d \\ \text{subject to } c_i(x_k) + \nabla c_i(x_k)^T d &\leq 0, \quad i \in \mathcal{I} \\ c_i(x_k) + \nabla c_i(x_k)^T d &= 0, \quad i \in \mathcal{E} \end{aligned}$$

where x_k is the current iterate, the constraints c_i are local linear approximations, and H_k is a positive definite estimate of the Hessian of the Lagrangian ($\nabla_{xx}^2 L(x_k, \lambda_k)$). The solution d_k to this subproblem defines the direction of descent. The distance moved in the step direction is determined by a line search to minimize a merit function consisting of the objective function and a penalty function based on the violation of the constraints.

To solve SQP problems, we used CFSQP (C code for Feasible Sequential Quadratic Programming, Lawrence *et al.* [16]) using QLD (Schittkowski [22]) to solve each quadratic programming subproblem. CFSQP is a variant of the general approach described above, and it is unique in

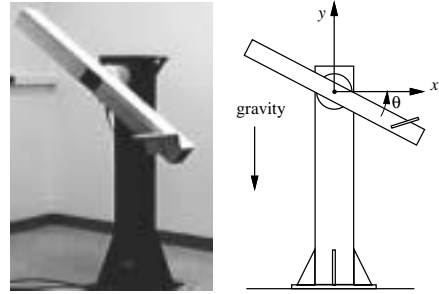


Figure 4: The NSK direct-drive arm.

that it maintains “feasible” iterates during the optimization—once an iterate is found that satisfies all linear constraints and nonlinear inequality constraints, all subsequent iterates will also satisfy these constraints. For problems without an rg subsequence, this property assures that each iterate in the solution process corresponds to a physically valid motion, though the goal state may not be achieved.

As with all iterative gradient-based optimization routines, SQP finds a local optimum which is not necessarily the global optimum. In addition, the finite-dimensional parameterization of the manipulator trajectory artificially limits the space of possible trajectories. The particular local optimum achieved by SQP depends on the shape of the feasible space and the initial guess. This problem can be alleviated by solving with many different initial guesses and choosing the best solution.

5.3 Putting It Together

A dynamic task is specified by a geometric description of the polygonal object, along with its center of mass and radius of gyration ρ ; the initial state and the desired goal state; and the sequence of manipulation phases to use (e.g., g , rg , gf , grg , or grf). A guess is also required to initialize the search. The motion of the object during rolling phases is simulated using fourth-order Runge-Kutta. SQP requires gradients of the constraints with respect to the design variables, and these are calculated using finite differences. All functions are implemented in C on a Sun SPARC 20.

6 Experiments

We have built a one-degree-of-freedom arm powered by an NSK direct-drive motor (Figure 4). The hollow aluminum arm is centrally-mounted and 122 cm long with a 10 cm square cross-section. It also has a “palm” mounted at a 45 degree angle. The top surface of the arm and the palm are used as manipulation surfaces. To increase friction and damping on these surfaces, they have been covered with a soft, 5 mm thick foam.

Because dynamic manipulation requires precise trajectory following, we have carefully modeled the response of the NSK motor. As a result, we obtain good open-loop tracking. (See (Lynch [17]) for details.) Small feedback corrections are also used.

Trajectories specified by the planner are directly implemented on the robot, without modification. In some cases,

however, effects that are not modeled in the planner cause the plans to fail when implemented on the robot. For example, the planner assumes that if the dynamic grasp constraints are satisfied at the end of a roll (an *rg* sequence), then the object is immediately in a dynamic grasp. Impact is not modeled. Also, the planner's rigid-body assumption is violated by the soft foam, which has the effect of slightly rounding a rolling vertex. When these unmodeled effects cause a plan to fail, the problem specification can be modified to compensate.

To execute a throw, the arm is maximally decelerated at the release point. This causes the object to be released nearly instantaneously. If the arm is also to catch the object, it follows a bang-bang trajectory to reach the catching configuration. Catches are made robust by appropriately choosing the object's impact state with the immobile arm.

We describe a snatch, throw, roll, and rolling throw in Figures 5–13. The test objects are lightweight and made of wood. More details can be found in (Lynch [17]).

7 Conclusion and Future Work

Dynamic underactuated nonprehensile manipulation exploits dynamic effects to achieve interesting behaviors with simple robots. This paper has presented an approach to generating open-loop controls that have been successfully implemented on a real robot system to perform a variety of dynamic tasks.

Future work should address feedback stabilization of dynamic manipulation trajectories, possibly using vision feedback; a more detailed analysis of the geometry of accessible states; using repeated contacts (batting) to increase the accessible state space; and tractable approaches to finding globally optimal manipulation plans.

Acknowledgments

This work was funded by NSF under grant IRI-9114208. Thanks to Pradeep Goel and NSK for providing the motor used in the experiments and André Tits and Craig Lawrence for providing the CFSQP software.

References

- [1] T. Abell and M. A. Erdmann. Stably supported rotations of a planar polygon with two frictionless contacts. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.
- [2] Y. Aiyama, M. Inaba, and H. Inoue. Pivoting: A new method of graspless manipulation of object by robot fingers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 136–143, Yokohama, Japan, 1993.
- [3] S. Akella, W. Huang, K. M. Lynch, and M. T. Mason. Planar manipulation on a conveyor with a one joint robot. In *International Symposium on Robotics Research*, 1995.
- [4] H. Arai and O. Khatib. Experiments with dynamic skills. In *1994 Japan–USA Symposium on Flexible Automation*, pages 81–84, 1994.
- [5] H. Arai and S. Tachi. Position control system of a two degree of freedom manipulator with a passive joint. *IEEE Transactions on Industrial Electronics*, 38(1):15–20, Feb. 1991.
- [6] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.
- [7] K. F. Böhringer, V. Bhatt, and K. Y. Goldberg. Sensorless manipulation using transverse vibrations of a plate. In *IEEE International Conference on Robotics and Automation*, pages 1989–1996, 1995.
- [8] Y.-C. Chen. Solving robot trajectory planning problems with uniform cubic B-splines. *Optimal Control Applications and Methods*, 12:247–262, 1991.
- [9] A. W. Divelbiss and J. Wen. Nonholonomic path planning with inequality constraints. In *IEEE International Conference on Decision and Control*, pages 2712–2717, 1993.
- [10] M. A. Erdmann. An exploration of nonprehensile two-palm manipulation: Planning and execution. In *International Symposium on Robotics Research*, 1995.
- [11] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics and Automation*, 4(4):369–379, Aug. 1988.
- [12] C. Fernandes, L. Gurvits, and Z. Li. Near-optimal nonholonomic motion planning for a system of coupled rigid bodies. *IEEE Transactions on Automatic Control*, 30(3):450–463, Mar. 1994.
- [13] T. Higuchi. Application of electromagnetic impulsive force to precise positioning tools in robot systems. In *International Symposium on Robotics Research*, pages 281–285. Cambridge, MA: MIT Press, 1985.
- [14] H. Hitakawa. Advanced parts orientation system has wide application. *Assembly Automation*, 8(3):147–150, 1988.
- [15] W. Huang, E. P. Krotkov, and M. T. Mason. Impulsive manipulation. In *IEEE International Conference on Robotics and Automation*, pages 120–125, 1995.
- [16] C. Lawrence, J. L. Zhou, and A. L. Tits. User's guide for CFSQP version 2.3. Institute for Systems Research 94-16, University of Maryland, 1994.
- [17] K. M. Lynch. *Nonprehensile Robotic Manipulation: Controllability and Planning*. PhD thesis, Carnegie Mellon University, The Robotics Institute, Mar. 1996. Available as CMU-RI-TR-96-05 and at <http://www.cs.cmu.edu/~mlab>.
- [18] K. M. Lynch and M. T. Mason. Controllability of pushing. In *IEEE International Conference on Robotics and Automation*, pages 112–119, Nagoya, Japan, 1995.
- [19] M. T. Mason and K. M. Lynch. Dynamic manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 152–159, Yokohama, Japan, 1993.
- [20] G. Oriolo and Y. Nakamura. Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators. In *Conference on Decision and Control*, pages 2398–2403, 1991.
- [21] N. Sawasaki, M. Inaba, and H. Inoue. Tumbling objects using a multi-fingered robot. In *Proceedings of the 20th International Symposium on Industrial Robots and Robot Exhibition*, pages 609–616, Tokyo, Japan, 1989.
- [22] K. Schittkowski. *QLD: A Fortran Code for Quadratic Programming, User's Guide*. Mathematisches Institut, Universität Bayreuth, Germany, 1986.
- [23] H. Sussmann. A continuation method for nonholonomic path-finding problems. In *IEEE International Conference on Decision and Control*, pages 2718–2723, 1993.
- [24] P. J. Swanson, R. R. Burrige, and D. E. Koditschek. Global asymptotic stability of a passive juggler: A parts feeding strategy. In *IEEE International Conference on Robotics and Automation*, pages 1983–1988, 1995.
- [25] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, 1988.
- [26] V. Yen and M. L. Nagurka. A suboptimal trajectory planning problem for robotic manipulators. *ISA Transactions*, 27(1):51–59, 1988.
- [27] N. B. Zumel and M. A. Erdmann. Nonprehensile two palm manipulation with non-equilibrium transitions between stable states. In *IEEE International Conference on Robotics and Automation*, pages 3317–3323, 1996.

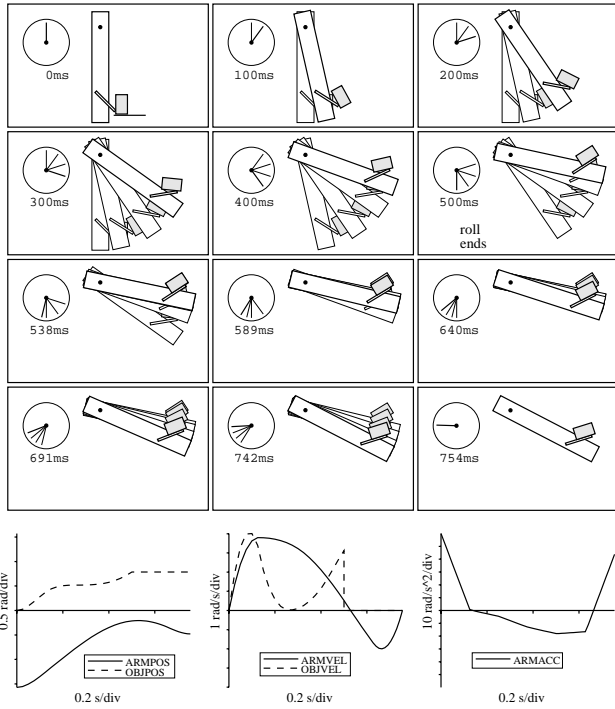


Figure 5: Snatch: the trajectory found by the optimization. Friction is 0.609. The plots show the angle of the arm (ARMPOSD) and the angle of the object (OBJPOSD) relative to it. The time between frames is not constant, so an equal number of frames of both phases can be seen. The clock indicates the time of each frame, and the previous three frames create a motion blur.

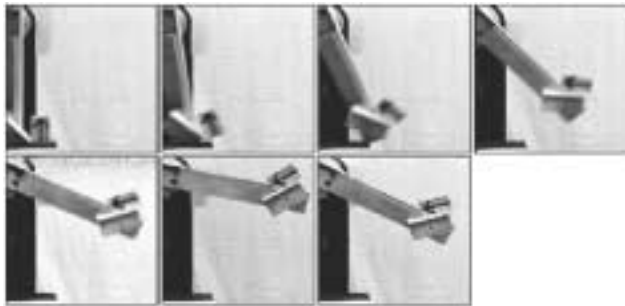


Figure 6: Snatch: implementation on the robot.

Snatch. Using the phase sequence rg , the optimization finds the snatching trajectory shown in Figure 5 after 200 iterations and 120 seconds. The trajectory consists of nine knot points, and the required friction coefficient μ is 0.609. The initial guess is for the arm to remain motionless, but the constraints of the optimization pull it toward a solution where the palm accelerates into the wooden block. The goal is any statically stable configuration after the roll has been completed.

The implementation on the arm works consistently (Figure 6). If the same trajectory is slowed down too much, the block is simply pushed off the table, and if it is sped up too much, the block is thrown.

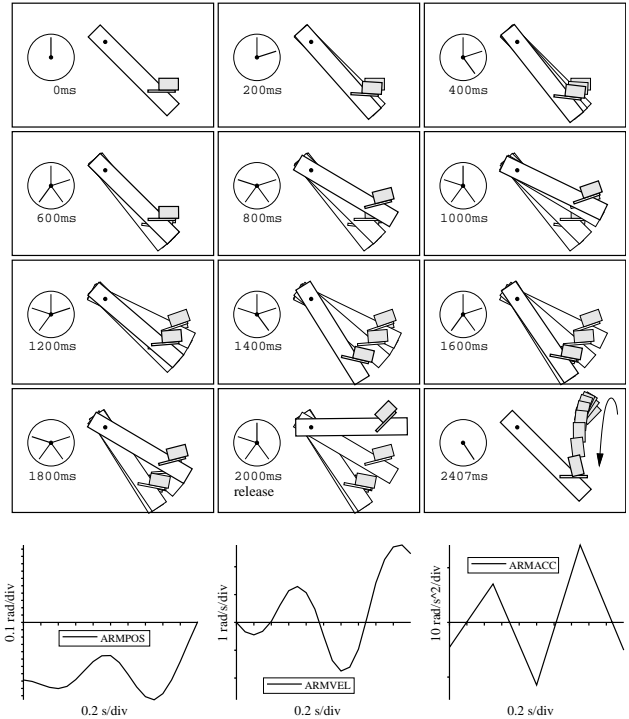


Figure 7: Throw: required friction is 0.156.

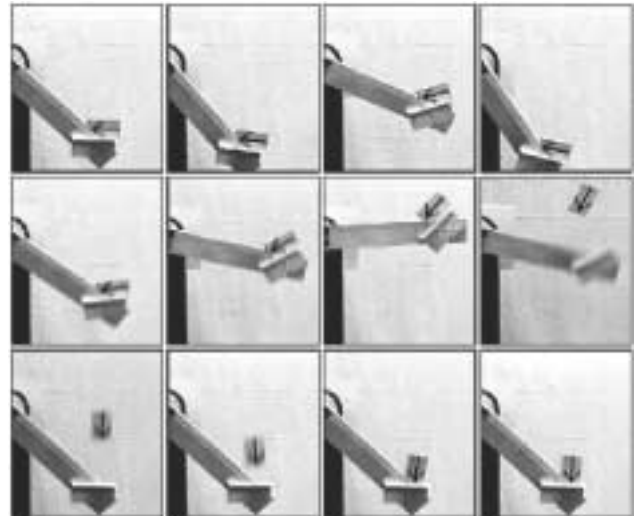


Figure 8: Throw: implementation on the robot.

Throw. After snatching the wooden block, the arm reorients it on the palm by throwing and catching it. Using the phase sequence gf , the optimization finds the trajectory shown in Figures 7 and 8 after 98 iterations and 13 seconds. The trajectory consists of seven knot points and the required friction coefficient is 0.156. The goal is a catching configuration where the block is overrotated to counteract its angular velocity.

Interestingly, the solution is to “double pump” before throwing. Path reversals are often necessary to minimize the required friction.

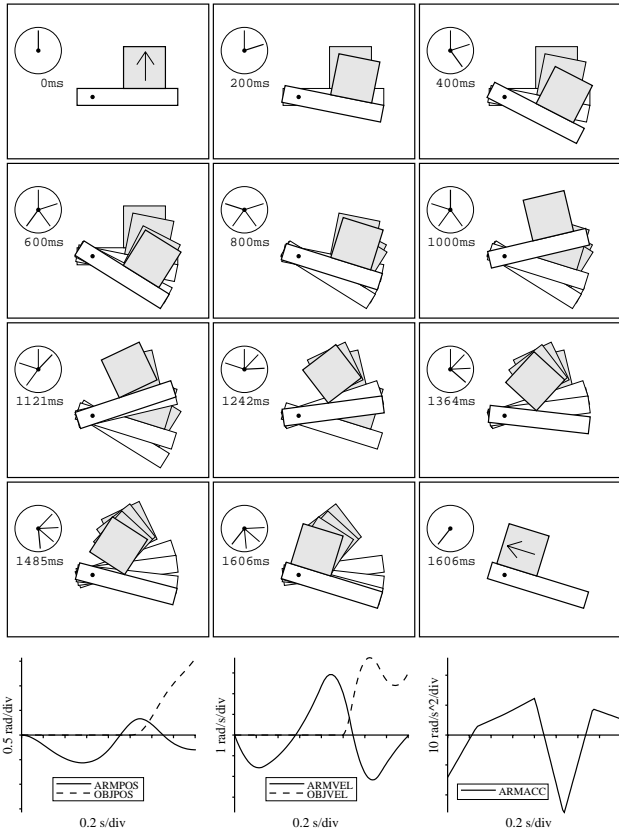


Figure 9: Roll: the trajectory found by the optimization.

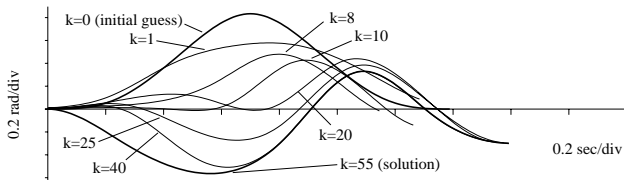


Figure 10: Roll: initial guess, solution, and intermediate iterates.

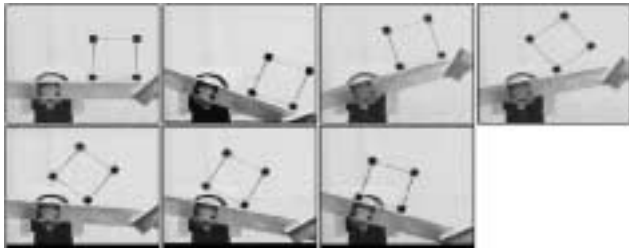


Figure 11: Roll: implementation on the robot.

Roll. The nine-knot rolling trajectory of Figures 9, 10, and 11 takes 32 seconds and 55 iterations to find. The object is a 27 cm square frame. Note the windup before the roll. In this example, the contact friction μ is set to 1.5 and the objective is to minimize the (squared) impact velocity at the end of the roll (Figure 9). We also limit the end angle of the arm to make the roll experimentally robust to impact. Otherwise, the solution is to end the roll with zero impact velocity and the arm at -45 degrees, which is not robust.

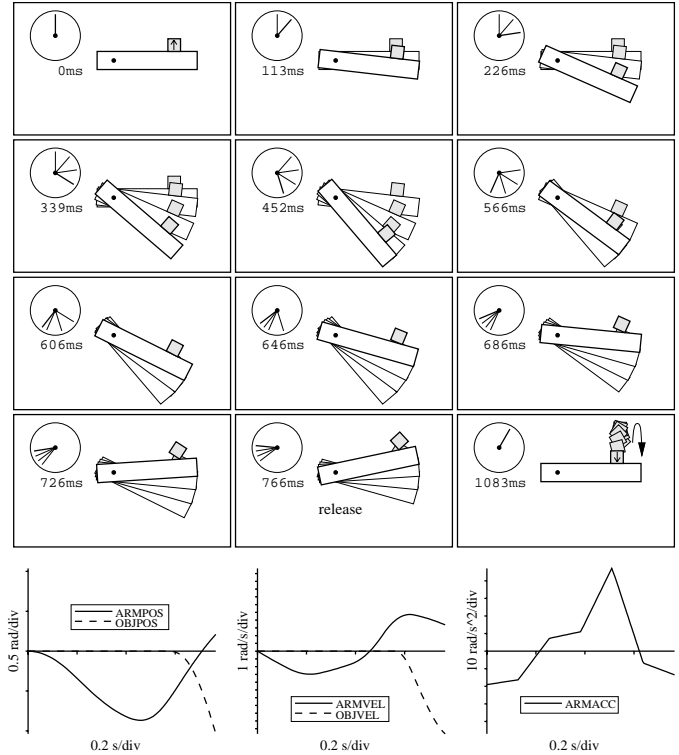


Figure 12: Rolling throw: the trajectory found by the optimization.

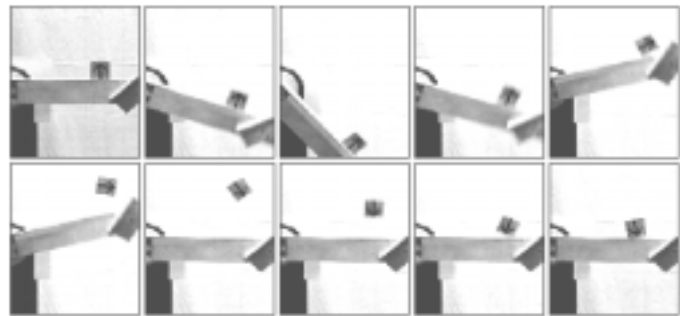


Figure 13: Rolling throw: implementation on the robot.

Rolling throw. In this example, a wooden cube (7.6 cm on each side) is thrown with an angular velocity different from that of the arm at release. This is only possible with a rolling throw. Here the block rotates half a revolution clockwise before landing on the arm in the same position. Notice that the block does not begin to roll when the arm is at its nadir; centrifugal and gravitational forces combine to begin the roll. The nine-knot trajectory in Figure 12 takes 54 iterations and 35 seconds to find. The required friction coefficient is 1.011. Examining the geometry of the object, we see that the friction coefficient must be greater than 1.0 to apply a clockwise torque to the object through the rolling vertex, indicating that the solution is nearly optimal.

The rolling throw is the dynamic task most sensitive to trajectory error, as any error in the roll is propagated to the flight phase. The implementation is shown in Figure 13.