Carnegie Mellon University
Research Showcase @ CMU

Robotics Institute | School of Computer Science

8-2005

Using Projected Dynamics to Plan Dynamic Contact Manipulation

Siddhartha S. Srinivasa
Carnegie Mellon University, ss5@andrew.cmu.edu

Michael A. Erdmann
Carnegie Mellon University, me@cmu.edu

Matthew T. Mason
Carnegie Mellon University, matt.mason@cs.cmu.edu

Follow this and additional works at: http://repository.cmu.edu/robotics

Part of the Robotics Commons

Recommended Citation
Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 3618- 3623.

This Conference Proceeding is brought to you for free and open access by the School of Computer Science at Research Showcase @ CMU. It has been accepted for inclusion in Robotics Institute by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Using projected dynamics to plan dynamic contact manipulation*

Siddhartha S. Srinivasa, Michael A. Erdmann and Matthew T. Mason
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA - 15213

***Abstract*— This paper addresses the planning and control of dynamic contact manipulation. In an earlier paper [13], we derived a constraint on the robot joint accelerations that needed to be satisfied to obtain a desired contact mode and a desired dynamic motion of the object. We proposed a technique for trajectory planning which involved planning a path in the system configuration space followed by time-scaling the path to satisfy dynamic constraints. This paper tackles a problem where only a small set of paths can be time-scaled to satisfy the constraints. We note that the dynamic constraints depend only on a subspace of the system state space. Projecting the dynamics and the constraints onto the subspace allows us to compute an analytical solution for the trajectory generation problem. We generate controllable simulations by allowing the user to control the system in the space orthogonal to the projection. We also demonstrate the construction of feedback controllers using dynamic programming.**

I. Introduction

Manipulation is the art of moving things. At the core of the manipulation problem, an object needs to be moved from a start to a desired goal by a robot that manipulates the object. As the system evolves, contacts occur both between the object and the robot, and between the object and the environment. Contacts are important because they act as the coupling between the various subsystems — they transmit forces and impose motion constraints on the object and the robot. For example, when a waiter manipulates a glass on a tray, he exerts a force on the glass through the tray. He also imposes a motion constraint that prevents the glass from sliding, tipping or losing contact.

A solution to the manipulation problem is a set of controls that can be applied by the robot on the object that will cause the desired motion of the object and satisfy the desired motion constraints.

By *dynamic* manipulation we mean "acting on the object during its dynamic phase". There are many reasons for using dynamics in manipulation. We refer the reader to [13] for a detailed list. The motivating reason in this paper is that *there is no quasi-static solution* for the problem described.

The problem we will be focusing on in this paper is shown in Fig.1(i). A block rests on a flat palm. The goal is to make the block stand up. If we denote the angle made by the block with the palm by θ, the goal is to manipulate the block from $\theta = 0$ to $\theta = \frac{\pi}{2}$. The palm cannot rotate. We control the horizontal and vertical acceleration of the palm ($\ddot{\boldsymbol{q}}_1$,$\ddot{\boldsymbol{q}}_2$). Furthermore, we need to maintain a rolling contact between the block and the palm. As per the laws of Coulomb friction this means that the contact friction force $\boldsymbol{f}$ must lie within the friction cone at the contact $\mathcal{F}$. As a practical consideration, we also bound the magnitude of the horizontal and vertical acceleration of the palm.

This problem, henceforth identified as the *tipping problem*, was first proposed by Erdmann[7]. There is a solution to the problem in Erdmann's paper, but one that was derived by hand-tuning the controller and the system parameters. One of our goals in this paper is to automate the trajectory planning. Furthermore, we would also like to answer questions like:

> What is the set of all feasible palm motions that move the block from $\theta = 0$ to $\theta = \frac{\pi}{2}$ in, say, 2.5 seconds, and have the exact same angular motion $\theta(t)$ of the block?

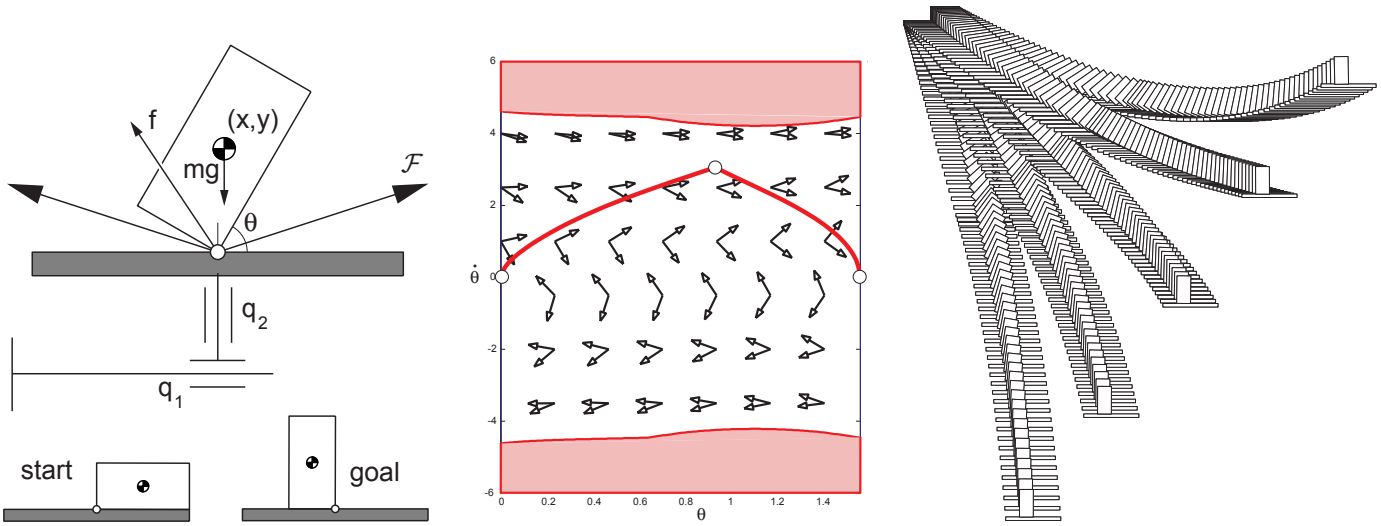
The answer to the question is illustrated in Fig.1(iii).

The problem solved in this paper is also closely related to the work of Lynch and Mason [10]. Using a robot arm with a single revolute degree of freedom, they demonstrated dynamic tasks such as snatching, rolling, throwing and catching an object.

In [13], we derived a constraint on the feasible joint accelerations for a given system state, a desired motion of the object and a desired set of contact conditions. A feasible trajectory is one that satisfies the joint acceleration constraint at every point. In [13], to obtain a feasible trajectory, we decoupled the problem into a path planning problem in the system configuration space followed by a time-scaling of the candidate path to satisfy the acceleration constraints. Note that a path has no notion of time. When we time-scale, we are in effect coming up with a time parametrization of the candidate path; we are deciding how fast or slow we wish to travel along the path. We deem a time-scaling successful if it is possible to find a time parametrization of the path that does not violate the acceleration constraints. Decoupling reduces the dimensionality of the problem from $2n_q$ to $(n_q + 2)$, where n_q is the dimension of the configuration space.

There is a drawback to the decoupling approach. It is possible that only a small subset of all feasible paths in the system configuration space can be successfully time-scaled. Applying the decoupling technique then amounts

*This work was supported in part by NSF grants IIS-9820180, IIS-9900322, IIS-0082339 and IIS-0222875.



(i) (ii) (iii)

Fig. 1. (i) : The tipping problem — a block rests on a flat palm. The goal is to tip the block *i.e.* go from $\theta = 0$ to $\theta = \frac{\pi}{2}$ while maintaining rolling contact, (ii) : projecting the dynamics and the constraints onto the space of task freedoms, (iii) : a sampling of the set of all feasible palm motions that move the block from $\theta = 0$ to $\theta = \frac{\pi}{2}$ in 2.5 seconds and have the exact same angular motion $\theta(t)$ of the block

to repeatedly trying candidate paths until one that could be successfully time-scaled is found. Searching for a small sliver in an infinite dimensional function space of configuration space paths can be very hard.

We propose a new technique for solving the trajectory generation problem. We first derive the contact acceleration constraint for the tipping problem using a result from [13]. We identify the subset of the state variables that the constraint depends on. We call these state variables the *task freedoms* of the system. For the block tipping, the task freedom is the rotation of the block relative to the palm, the angle θ. We then project the dynamics and the constraints onto the space of task freedoms. For the block tipping, the state space comprises of the configuration of the block and its velocity, a 6 dimensional space. We project the dynamics and the constraints onto the 2 dimensional space of $(\theta, \dot{\theta})$, as shown in Fig.1(ii). We plan a feasible trajectory in the lower dimensional task freedom space and control the nullspace of the projection to control the reminder of the state variables. A feasible trajectory for the block tipping is shown in Fig.1(ii). The trajectory takes the block from $\theta = 0$ to $\theta = \frac{\pi}{2}$ in 2.5 seconds while maintaining a rolling contact between the block and the palm. By varying the control in the nullspace, we are able to produce various palm motions that do not affect the task freedom trajectory. The family of trajectories generated is shown in Fig.1(iii).

We are also interested in developing feedback controllers for dynamic contact manipulation. For a given cost function, a feedback controller outputs the optimal control that needs to be applied at each system state which will result in a trajectory that minimizes the cost function. In this paper, we develop a feedback controller for the task freedoms using backward dynamic programming. We discuss the implementation of the controller in §V.

The rest of the paper is arranged as follows. In §II, we introduce the terminology used in the rest of the paper and give a clear definition of the problem we are interested in solving. §III describes relevant background work on manipulation and dynamic programming. §IV describes the solution of the planning problem. §V describes the feedback controller. In §VI, we discuss our contributions and describe future work.

II. Problem Statement

We describe the configuration of the block by its pose $\boldsymbol{q_o} = (x, y, \theta)^T$ and the configuration of the robot by its joint variables $\boldsymbol{q_r} = (q_1, q_2)^T$. We denote the configuration of the system by $\boldsymbol{q} = (\boldsymbol{q_o}, \boldsymbol{q_r})^T$.

A rolling contact constrains the relative velocity between the object and the palm at the contact point:

$$\mathsf{G}(\boldsymbol{q_o})^{\mathsf{T}} \dot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{J}(\boldsymbol{q_r}) \dot{\boldsymbol{q}}_{\boldsymbol{r}} \tag{1}$$

where $\mathsf{G}(\boldsymbol{q_o}) \in \mathbb{R}^{3\times 2}$ is the grasp map which relates contact forces to wrenches on the object and $\mathsf{J}(\boldsymbol{q_r}) \in \mathbb{R}^{2\times 2}$ is the Jacobian of the robot.

Coulomb friction imposes a constraint on the contact force $\boldsymbol{f}$:

$$\boldsymbol{f} \in \mathcal{F}(\mu) \tag{2}$$

where $\mathcal{F}(\mu)$ is a convex cone and μ is the coefficient of friction between the palm and the object, a material property.

The motion of the object is governed by its dynamics:

$$\mathsf{M}\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{G}\boldsymbol{f} + \boldsymbol{n_o} \tag{3}$$

where M is the inertia matrix of the object and $\boldsymbol{n_o}$ is the wrench on the object due to gravity.

The manipulation problem can be stated as:

> Given a start $\boldsymbol{q_s}$ and a goal $\boldsymbol{q_g}$ configuration for the system, find the robot joint acceleration

$\ddot{\boldsymbol{q}}_{\boldsymbol{r}}(t)$ that will move the system from the start to the goal without violating the contact velocity constraint (Eqn.1) or the contact force constraint (Eqn.2).

An industrial robot typically accepts desired joint torque as an input. The mapping from robot joint acceleration to joint torque involves the dynamics of the robot and is a well studied problem [6].

III. Background Work

Early work on dynamic manipulation focused on *dexterous manipulation* where a robot hand manipulated an object with multiple frictional fingers. In Cole *et al.* [4], [5], some fingers were designated to slide on the object's surface while others were designated to roll. The authors provided a control law that achieved simultaneous tracking of a pre-planned object trajectory together with the desired motion at the fingertips using the location of the contacts and the relative velocity at the contacts as feedback.

Brook *et al.* [3] studied the manipulation of objects in equilibrium grasps. They showed that most equilibrium grasps were locally controllable, and stabilizable under suitable feedback control. They showed that manipulation from one equilibrium grasp to another was possible if there was a continuity of equilibrium grasps between them.

Trinkle and Hunter [14] provided a framework for manipulation planning. They defined a contact formation (CF) as a qualitative description of a grasp based on contacts between the vertices, edges and surfaces of the robot and the object. To plan a motion, they constructed CF-trees with the start and goal CFs as the parent nodes, the child nodes being the CFs the parents could transition to, and the arcs being controls that caused the transitions. A plan existed if the the start and goal CF-trees had a common node.

Erdmann [7] explored the nonprehensile manipulation of planar convex objects using two flat palms. He restricted the actions that could be performed by the palms to four primitives which caused the object to tilt, rotate, slide, or be released. Based on a static analysis, he decomposed the configuration space of the object into regions of invariant dynamics, and searched for plans in this simplified space.

Sarkar *et al.* [11] studied the control of a robot in a cooperative manipulation task ordered to compliantly follow the motion of an object. They designed a nonlinear feedback controller that maintained rolling contact.

Yashima, Shiina and Yamaguchi [15] studied the dynamic manipulation of an object by a hand where the contacts underwent both rolling and sliding. They used a rapidly exploring random tree (RRT) to search for a feasible path for the object.

The optimal control problem deals with the generation of trajectories and controls for a system that minimize a given cost function. The work of Bellman [1] pioneered the use of dynamic programming to solve optimal control problems. Bellman proposed a numerical technique for finding the optimal cost from any point in a state space to a goal. The gradient of the optimal cost function can be used to generate the corresponding optimal feedback controller. One drawback of dynamic programming is that the running time grows exponentially with the number of dimensions. Lavalle and Konkimalla [8] proposed several techniques for improving the speed of dynamic programming.

IV. Planning

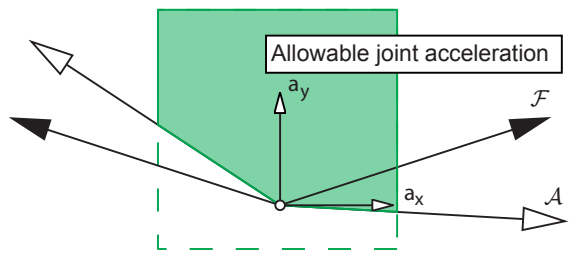


Fig. 2. Constraints in contact acceleration space: the dark polygon denotes the set of feasible contact accelerations. The friction cone $\mathcal{F}$ is mapped to the acceleration cone $\mathcal{A}$.

In this section, we map the constraints given by Eqn.1 and Eqn.2 to a common space and describe an analytical solution to our problem.

A. The contact acceleration constraint

In [13], we derived a constraint that the robot joint acceleration has to satisfy in order to obtain both a desired motion of the object and a desired contact condition. We state the theorem in the following paragraph and refer the reader to [13] for a detailed proof.

Theorem 1. *For a given configuration $\boldsymbol{q} = [\boldsymbol{q_o}, \boldsymbol{q_r}]^T$ and velocity $\dot{\boldsymbol{q}} = [\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}]^T$, the allowable joint acceleration of the robot $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}$ that satisfies both the velocity and the force constraint is constrained to lie within the cone:*

$$\boxed{\mathsf{J}\ddot{\boldsymbol{q}}_{\boldsymbol{r}} + \mathsf{V}(\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n_o}) \in \mathcal{A}} \tag{4}$$

$$\mathsf{V}(\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n_o}) = \dot{\mathsf{J}}\dot{\boldsymbol{q}}_{\boldsymbol{r}} - \dot{\mathsf{G}}^{\mathsf{T}}\dot{\boldsymbol{q}}_{\boldsymbol{o}} - \mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\boldsymbol{n_o}$$

and

$$\mathcal{A} = \mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{G}(\mathcal{F})$$

is the contact acceleration cone.

The theorem gives us a series of mappings that can be used to combine the contact force constraint and the contact velocity constraint in a common space. The contact friction cone $\mathcal{F}$ is mapped to a contact acceleration cone $\mathcal{A}$ and the contact velocity constraint is mapped to a contact acceleration constraint.

When we apply Thm.1 to the tipping problem, we obtain the following constraints on the allowable joint acceleration:

$$\begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} - d \begin{pmatrix} \cos(\theta+\beta) \\ \sin(\theta+\beta) \end{pmatrix} \dot{\theta}^2 + \begin{pmatrix} 0 \\ g \end{pmatrix} \in \mathcal{A}(\theta) \tag{5}$$

where $\beta = \arctan(w/l)$, w and l are the width and length of the block respectively, d is the distance from the center of mass of the block to the contact and g is the acceleration due to gravity.

Note that Eqn.5 only depends on θ and $\dot{\theta}$, and not on (x, y) or $(\dot{x}, \dot{y})$. This suggests that, for the purpose of satisfying the contact velocity and contact force constraints, it is sufficient to analyze the evolution of the dynamics in the $(\theta, \dot{\theta})$ subspace. We denote this subspace as the space of *task freedoms*.

For the tipping problem, the following map takes us from the space of contact forces to the space of contact accelerations:

$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} = (\mathsf{G}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G})f \tag{6}$$

where $(a_x, a_y)^T$ is the contact acceleration.

For a given state of the system, an illustration of Eqn.5 in the contact acceleration space is shown in Fig.2. The rays with solid arrows represent the friction cone at the contact and the rays with outlined arrows represent the acceleration cone at the contact. The bounds that we have imposed on the joint accelerations of the robot appear as a rectangle in the contact acceleration space. Feasible contact accelerations lie in the intersection of the contact acceleration cone and the joint acceleration limits.

It can be shown that:

$$\mathcal{N}(G) = \{\mathbf{0}\} \implies \mathcal{N}(\mathsf{G}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G}) = \{\mathbf{0}\} \tag{7}$$

This means that in the absence of internal forces the map from contact forces to contact accelerations is invertible. This makes intuitive sense since an internal force is a contact force that produces no contact acceleration and by definition lies in the nullspace of the map from contact forces to contact accelerations. If no internal forces exist, the map has no nullspace and is invertible.

We invert the map to obtain:

$$\boldsymbol{f} = (\mathsf{G}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G})^{-1}\begin{pmatrix} a_x \\ a_y \end{pmatrix} \tag{8}$$

We can write the dynamics of the block as:

$$\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{M}^{-1}(\mathsf{G}\boldsymbol{f_c} + \boldsymbol{n_o}) \tag{9}$$

We can combine Eqn.8 and Eqn.9 as:

$$\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{A}\begin{pmatrix} a_x \\ a_y \end{pmatrix} + \boldsymbol{b} \tag{10}$$

Hence the evolution of the system is given by Eqn.10 subject to the constraint given by Eqn.5.

B. Projection onto the space of task freedoms

In this subsection we will project the contact acceleration constraint onto the space of task freedoms. The projection is illustrated in Fig.3. As mentioned in the previous subsection, the dark polygon in Fig.3 corresponds to set the feasible contact accelerations. Each point in this set can be mapped to a feasible $\ddot{\theta}$ using Eqn.10. In Fig.3, this mapping is illustrated by a projection onto the unit vector $\boldsymbol{a_\theta}$. Limits

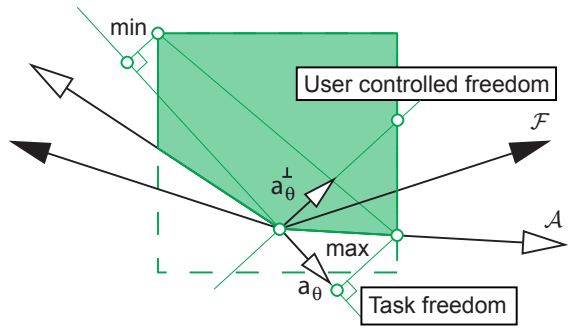


Fig. 3. Projecting the contact acceleration constraint onto and orthogonal to the space of task freedoms

on the feasible $\ddot{\theta}$ can be easily computed by projecting the vertices of the feasible polygon onto $\boldsymbol{a_\theta}$, as shown in the figure.

To obtain $\boldsymbol{a_\theta}$, we rewrite Eqn.10 as:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{pmatrix} = \begin{bmatrix} \mathsf{A}_{\mathsf{xy}} \\ \mathsf{A}_\theta \end{bmatrix}\begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{bmatrix} \boldsymbol{b_{xy}} \\ b_\theta \end{bmatrix} \tag{11}$$

We then perform a transformation of the input by rotating it along and orthogonal to $\boldsymbol{a_\theta}$:

$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} = \boldsymbol{a_\theta}\alpha + \boldsymbol{a_\theta^\perp}\beta \tag{12}$$

We compute the orthogonal unit vectors $\boldsymbol{a_\theta}$ and $\boldsymbol{a_\theta^\perp}$ using the additional relation $\mathsf{A}_\theta \boldsymbol{a_\theta^\perp} = 0$.

We can now decouple the evolution of the task freedoms and the rest of the state variables as:

$$\ddot{\theta} = \mathsf{A}_\theta \boldsymbol{a_\theta^T}\alpha + b_\theta \tag{13}$$

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \mathsf{A}_1\alpha + \mathsf{A}_2\beta + \boldsymbol{b_{xy}} \tag{14}$$

We compute limits on α by projecting the vertices of the feasible polygon onto $\boldsymbol{a_\theta}$. We use the limits on α to compute limits on $\ddot{\theta}$ using Eqn.13.

We now focus our attention on the evolution of the task freedoms given by Eqn.13. Limits on $\ddot{\theta}$ constrain the tangent space of the task freedom $(\theta, \dot{\theta})$, as shown in Fig.1(ii). At every point of a feasible trajectory in the task freedom space, the tangent must lie within the cone of allowable tangents at that point.

There is substantial literature on computing analytical solutions for trajectory generation in 2 dimensions [2], [12]. We choose a simple technique for finding feasible trajectories. We pick a feasible vector field and follow it from the start (here, $\theta = 0$). We pick another feasible vector field and back project it from the goal (here, $\theta = \frac{\pi}{2}$). At the point where the two trajectories intersect, we switch vector fields.

Once a feasible trajectory is computed, we can compute the control $\alpha(t)$ that is required to move the system along that trajectory. We can then use the computed $\alpha(t)$ in Eqn.14 and vary $\beta(t)$ to move the palm wherever we

wished. Note that varying $\beta(t)$ does not affect the motion of the task freedoms in Eqn.13 because $\beta(t)$ lies in an orthogonal space to the task freedoms, by construction (as shown in Eqn.12).

A few of the trajectories generated by varying the $\beta(t)$ are shown in Fig.1(iii). Following any of these trajectories gives us the same rotational motion of the block. Furthermore, we can also transition from one trajectory to another without changing the motion of the block.

The decomposition provides us a controllable simulation. For example, in the tipping problem, if a user were handed two joysticks, one for each degree of freedom of the palm and told to tip the block, he would invariably fail — the block would very easily slide, or lose contact with the palm. Instead, we can give the user control of $\beta(t)$ with the assurance that any value of $\beta(t)$ will satisfy the contact constraints. We have solved the hard problem (tipping the block) and given the control of the freedoms of secondary importance (the motion of the palm) to the user.

C. Sliding

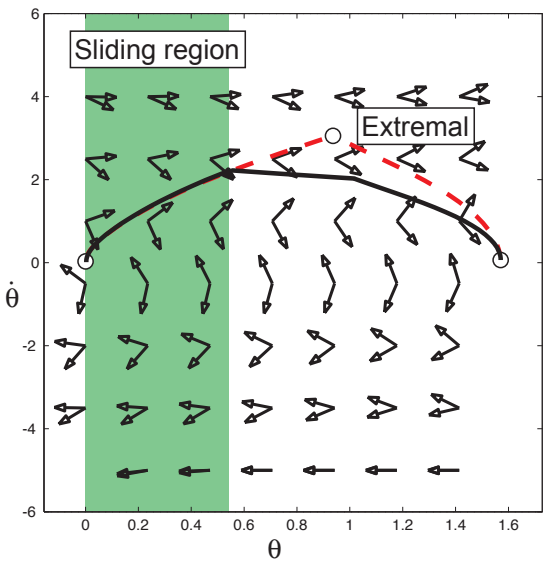


Fig. 4. A trajectory that involves both sliding and rolling at the contact. The block slides in the sliding region and reverts back to rolling out of the region.

Until now we have looked at plans that only involve rolling at the contact between the block and the palm. For the tipping problem, it is relatively easy to incorporate the sliding of the palm as well. This is predominantly because the contact between the block and the palm is of *Type A* (a contact between a vertex of the block and a side of the palm [9]). As a result, the moment applied by any contact force on the block by the palm remains the same regardless of the position of the block relative to the palm. This is illustrated in Fig.5. Hence sliding the palm relative to the block does not affect the motion of the task freedoms.

An instance of a plan involving sliding and rolling is shown in Fig.4. The dotted line denotes the extremal

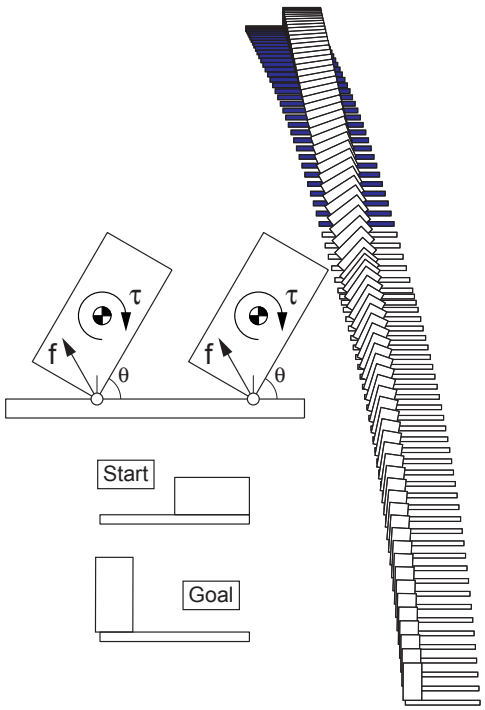


Fig. 5. An illustration of a Type A contact and a palm motion that results in both sliding (shown by the dark palms) and rolling at the contact

trajectory — one where sliding of the palm must occur. The extremal is computed by following the limits of the allowable tangents at each point of of the state space. The solid line denotes a trajectory where both rolling and sliding occur. When the solid line coincides with the extremal, we are allowed to slide the palm relative to the block (the allowable region is colored in the plot). However we need to ensure that by the time the trajectory reaches the end of the allowable region, the sliding must cease — the relative velocity of the palm with respect to the block must be brought to zero. At the end of the allowable region, the solid line leaves the extremal and rolling contact is initiated. The block then maintains rolling contact until it reaches the goal.

The motion of the palm for the trajectory given in Fig.4 is shown in Fig.5. The times where the palm is dark denote sliding. For this problem, we used a simple constant acceleration followed by a constant deceleration for the motion of the palm relative to the block during the sliding phase.

V. Feedback Control

A feedback controller provides the optimal control to be applied at any state of the system that will result in a trajectory that minimizes a given cost function. We used backward dynamic programming to construct a feedback controller for the block tipping problem. The cost function we optimized was time-optimality, *i.e.* the feedback controller gave us paths that took the least amount of time to get to the goal.

The numerical technique we used is described in [1]. We discretize the $(\theta, \dot{\theta})$ space with a uniform grid and discretize the controls, as shown in Fig.6. The output of the dynamic program is an optimal cost function that gives the optimal cost (here, minimum time) to get from any point in the state space to the goal. The optimal control is one that moves the trajectory in a direction that is closest to the gradient of the optimal cost function. The level sets of the optimal cost function and a time-optimal trajectory from $\theta = 0$ to $\theta = \frac{\pi}{2}$ are shown in Fig.7.

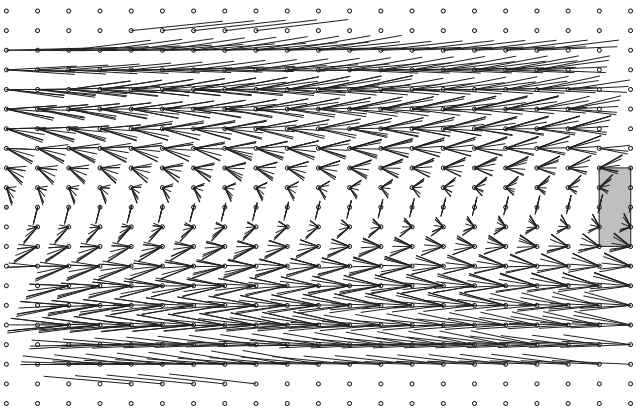


Fig. 6. Discretization for dynamic programming: the circles denote the discrete states. At each discrete state there are 10 discrete feasible controls. The curves denote the orbits of each control for the given time-step. The goal region is denoted by the dark rectangle.

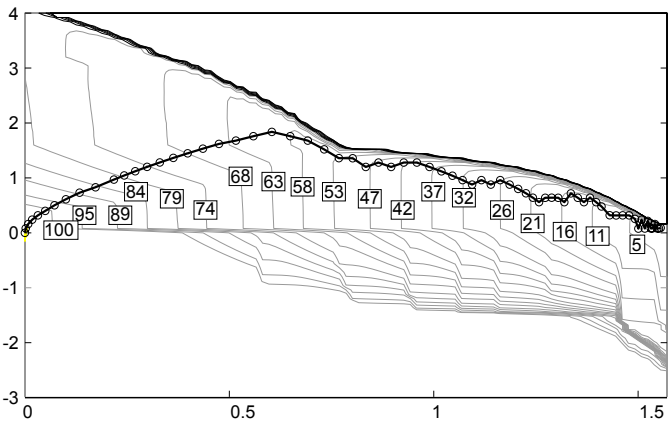


Fig. 7. Feedback control: isocontours of the time-optimal cost-to-go function and their values in milliseconds are shown. A time-optimal trajectory from $\theta = 0$ is shown as a thick curve along with the points of evaluation of the gradient

VI. Conclusions

In this paper, we have demonstrated a new technique for solving the trajectory generation problem for dynamic contact manipulation. We have shown how the projected dynamics can be used to generate controllable simulations — where the user can control a set of the freedoms of the system without violating the dynamic constraints. We have also presented the construction of feedback controllers for manipulation systems using dynamic programming. In the future, we will work on exploring the reachability and controllability of manipulation systems. We are also working on an implementation of the block tipping problem using an Adept 550 industrial arm.

VII. Acknowledgment

We would like to thank Steven LaValle for his insight on using dynamic programming for writing feedback controllers.

References

[1] R.E. Bellman. Dynamic programming. Princeton University Press, 1957.

[2] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. In *International Journal of Robotics Research*, volume 4, pages 3–17, Fall 1985.

[3] N. Brook, M. Shoham, and J. Dayan. Controllability of grasps and manipulations in multi-fingered hands. In *IEEE Transactions on robotics and automation*, volume 14, pages 185–192, 1998.

[4] A.A. Cole, J.E. Hauser, and S.S. Sastry. Kinematics and control of multifingered hands with rolling contact. In *IEEE Transactions on robotics and automation*, volume 34, 1989.

[5] A.A. Cole, P. Hsu, and S.S. Sastry. Dynamic control of sliding by robot hands for regrasping. In *IEEE Transactions on robotics and automation*, volume 8, 1992.

[6] J.J. Craig. *Introduction to Robotics*. Addison Wesley, 1989.

[7] M.A. Erdmann. An exploration of nonprehensile two-palm manipulation. In *Int. Journal of Robotics Research*, volume 17, 1998.

[8] S.M. LaValle and P. Konkimalla. Algorithms for computing numerical optimal feedback strategies. In *International Journal of Robotics Research*, volume 20, pages 729–752, September 2001.

[9] T. Lozano-Pérez. Spatial planning: a configuration space approach. In *IEEE Transactions on Computers*, volume 32, pages 108–120, 1983.

[10] K.M. Lynch and M.T. Mason. Dynamic nonprehensile manipulation: Controllability, planning, and experiments. In *International Journal of Robotics Research*, volume 18, 1999.

[11] N. Sarkar, X. Yun, and V. Kumar. Control of contact interactions with acatastatic nonholonomic constraints. In *International Journal of Robotics Research*, volume 16, 1997.

[12] K. Shin and N. McKay. Minimum-time control of robotic manipulators with geometric path constraints. In *IEEE Transactions on Automatic Control*, volume 30, pages 531–541, June 1985.

[13] S.S. Srinivasa, M.A. Erdmann, and M.T. Mason. Control synthesis for dynamic contact manipulation. In *IEEE International Conference on Robotics and Automation*, 2005.

[14] J.C. Trinkle and J. J. Hunter. A framework for planning dexterous manipulation. In *IEEE International Conference on Robotics and Automation*, pages 1245–1251, 1991.

[15] M. Yashima, Y. Shiina, and H. Yamaguchi. Randomized manipulation planning for a multi-fingered hand by switching contact modes. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2689 – 2694, September 2003.