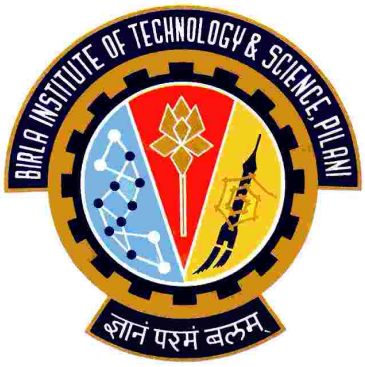
In-Hand Manipulation Using Extrinsic Dexterity

By

Ajinkya Arun Bhole



Department of Mechanical Engineering
Birla Institute of Technology and Science, Pilani, Rajasthan

A report submitted to Birla Institute of Technology and Science, Pilani, Rajasthan in accordance with the requirements of the degree of B.E. Hons in Mechanical Engineering.

May 2016

Certificate

It is certified that the work contained in this report, titled **"In-Hand Manipulation Using Extrinsic Dexterity"** by **Ajinkya Bhole**, has been carried out under my supervision and is to my satisfaction.

Date Dr. B.K. Rout

In-Hand Manipulation Using Extrinsic Dexterity

Abstract

This work is studies the task of achieving In-Hand manipulation with **simple robotic hands** by exploiting Extrinsic Dexterity. In-Hand manipulation is the action of changing the pose of a grasped object with respect to the hand and Extrinsic Dexterity as defined by Matthew Mason et. al. in [3], is an In-Hand manipulation strategy to achieve dexterity by using resources extrinsic to the robot.

Extrinsic Dexterity can take three forms, namely, Quasi-static actions, Passive Dynamics actions and Active Dynamic actions. This work specifically studies the form of in-hand manipulation of a grasped object by pushing it against an external pusher (Quasi-static Action). This work majorly involved the study of modeling the friction contact between the object and the robotic hand and the external contact as a **Mixed-Linear Complementarity Problem**. The model predicts the minimum force required from the external pusher to break the grasp equilibrium of the object in **desired directions**.

The model was evaluated using a primitive prehensile pushing action of straight sliding using different types of external contacts, namely point and line contact.

Acknowledgements

I would like to take this opportunity to express profound gratitude and deep regards to my advisor **Dr. B.K. Rout** for his exemplary guidance, monitoring and constant encouragement throughout the course of this work. The help and guidance given by him time to time shall carry me a long way in the journey of life in which I am about to embark. Working on this project was an extremely challenging and interesting adventure.

Contents

List of Figures

Chapter 1

Introduction

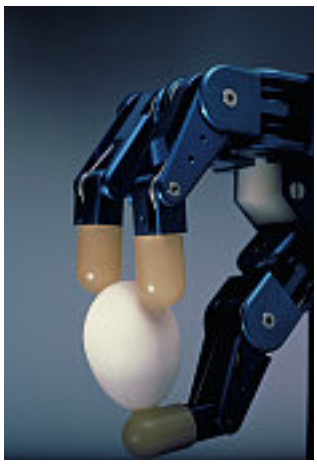


Figure 1.1: The Salisbury Hand, an anthropomorphic robotic hand which uses nine actuators [7]

.

We often try to design robotic hands that can achieve variegated manipulation tasks. A common example being robotic anthropomorphic hands (Figure 1.1). These designs no doubt are able to achieve numerous complex manipulation tasks but they also come with a complex mechanical design, fairly complex control of the actuation system and at a very high cost. What we often forget during the design phase is to look at the environment of the robot. Whether there are any resources extrinsic to the robot which can be used to our benefit, thereby simplifying the mechanical design, control and reducing the cost.

> *"..the dexterity of a hand is mostly about the brain, not the hand.."*
>
> – Matthew T. Mason (MLab, CMU)

Matthew T. Mason, the founder of Manipulation Lab at Carnegie Mellon University, took this approach for In-Hand Manipulation. He called this approach,

Extrinsic Dexterity [3]. A larger goal of this approach is to shoulder simple robotic hands with complex capabilities like In-Hand Manipulation.

Manipulation using Extrinsic Dexterity stands in contrast to the Dexterous Manipulation approach [6], which manipulates the object through finely controlled fingertip contacts. Extrinsic dexterity complements the intrinsic capabilities of the hand.

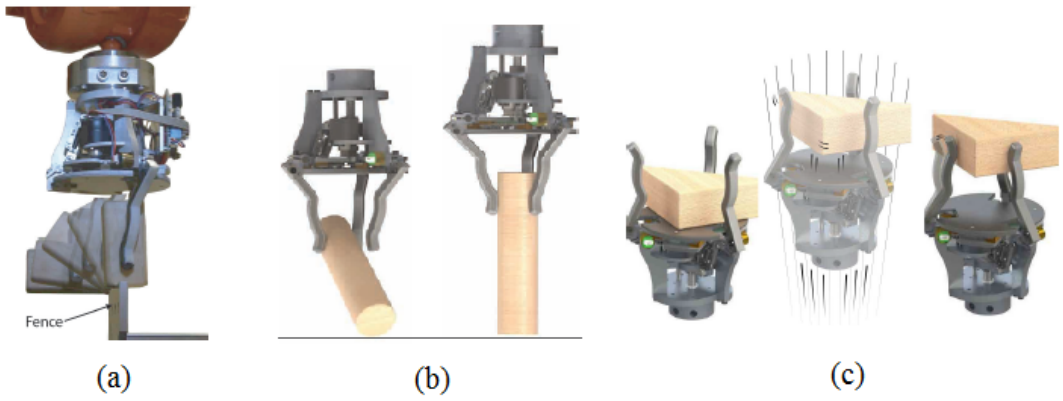


Figure 1.2: Extrinsic Dexterity forms (a) Quasi-static Action (b) Passive Dynamic Action and (c) Active Dynamic Action [3]

.

> *"Other useful sources of control forces include gravity, the frictional kinematic constraints (floor, walls, obstacles) making up the robots environment, and dynamic forces. If the robot can reason about these forces, it can use a richer set of manipulation primitives, including pushing, throwing, and striking."*
>
> – Lynch and Mason in [5]

Extrinsic Dexterity can take on of the following forms: Quasi-static Actions (using external contacts), Passive Dynamic Actions (gravity) and Active Dynamic Actions (inertia, dynamic motions) (Figure 1.2). This work studies in detail the problem of Prehensile Pushing using Quasi-static Actions [2].

The quasi-dynamic motion of the of the object held by point and line contacts and pushed by an external pusher is described. The friction contacts in this In-Hand Manipulation task are modeled as a Linear Complementarity Problem (LCP). Additional equality and non-equality constraints arising from the object dynamics, object acceleration, etc., converts the problem into a Mixed-Linear Complementarity Problem. The model predicts the minimum force required from the external pusher to break the grasp equilibrium in the user desired directions.

2

Prehensile Pushing: Problem Formulation

The main goal of this study is to predict the motion of a grasped object when pushed against a fixed external pusher through a controlled movement of the manipulator. If we are sitting on the robot hand frame, an external pusher in the environment can be seen as a highly dexterous virtual finger whose motion is the reflection of the motion of the arm. For the simplicity of exposition, an equivalent problem is formulated, by keeping the hand fixed and the object being pushed by a moving environment. The arm moving the object is assumed to have full cartesian dexterity, giving the external pusher/environment the freedom to push in any direction and orientation.

2.1 Known Information

The following information is assumed to be known:

- the shape and mass of the object
- the location, geometries, and frictional properties of all contacts between the object and the gripper
- the kinematics of the gripper, in the form of the jacobian from actuators to contact velocities
- magnitude of the gripping force
- location, geometry, frictional properties and motion of the external pusher

2.2 Problem Space

The problem is formulated in the space of:

- $\vec{f_i}$, $\vec{f_{ext}}$: forces at all contact points between the gripper, the object, and the external pusher
- $\vec{a_i}$, $\vec{a_{ext}}$: relative accelerations at each contact in the contact frame (defined in the following chapter)
- $\vec{a_{obj}}$: acceleration of the object in the object frame

2.3 Problem Constraints

The problem is subjected to the following constraints:

- **Newtonian Mechanics:** The acceleration of the object results from the total wrench applied on the object by the internal grasp forces, the forces at the external pusher, and gravity
- **Rigid body constraints:** Accelerations at contacts should be in accordance with that of the object
- **Unilateral contact constraints:** Contacts can only push, and only if the contact is maintained.
- **Frictional force constraints:** The normal force, tangential frictional force, and tangential acceleration at each contact should follow Coulomb's friction law and the principle of maximum energy dissipation
- **Constraints due to complex contacts:** We model contacts with non trivial geometry as a discrete set of rigidly attached frictional points. The individual interactions of each constituent point with the object must respect the rigidity of the patch
- **Constraints due to the motion of the pusher:** When the grasp breaks, the object must follow the pusher motion. In particular, their relative normal accelerations at contact must comply.

3

Contact Modeling

This chapter briefly describes conventional point contact models and the standard linearization of Coulomb friction cone for point contacts.

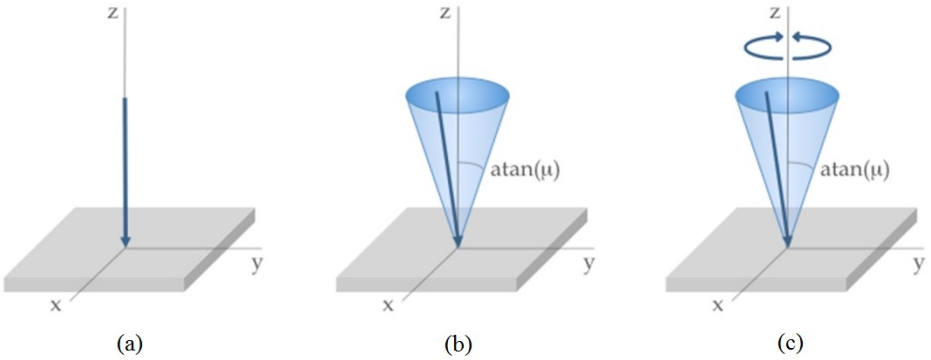


Figure 3.1: Point Contact Models: (a) Frictionless point contact (b) Frictional hard point contact (c) Frictional soft point contact

.

3.1 Point Contact Models

Friction interaction is quite difficult to model accurately. Contact models, however, are compact and computationally convenient, and to some degree indicative. **At the most basic level, contact models encode the directions along which a contact can transmit forces and torques**. Following are the three most common:

- **Frictionless point contact**: It transmits force only along the normal to the surface (Figure 3.1 a).
- **Frictional hard point contact**: It transmits forces along three directions, one normal and two tangential, but no torques. It approximates interaction

with a small contact area (Figure 3.1 b).

- **Frictional soft point contact**: It transmits forces along three directions, one normal and two tangential, and torque about the contact normal. It approximates interaction with a larger contact area (Figure 3.1 c).

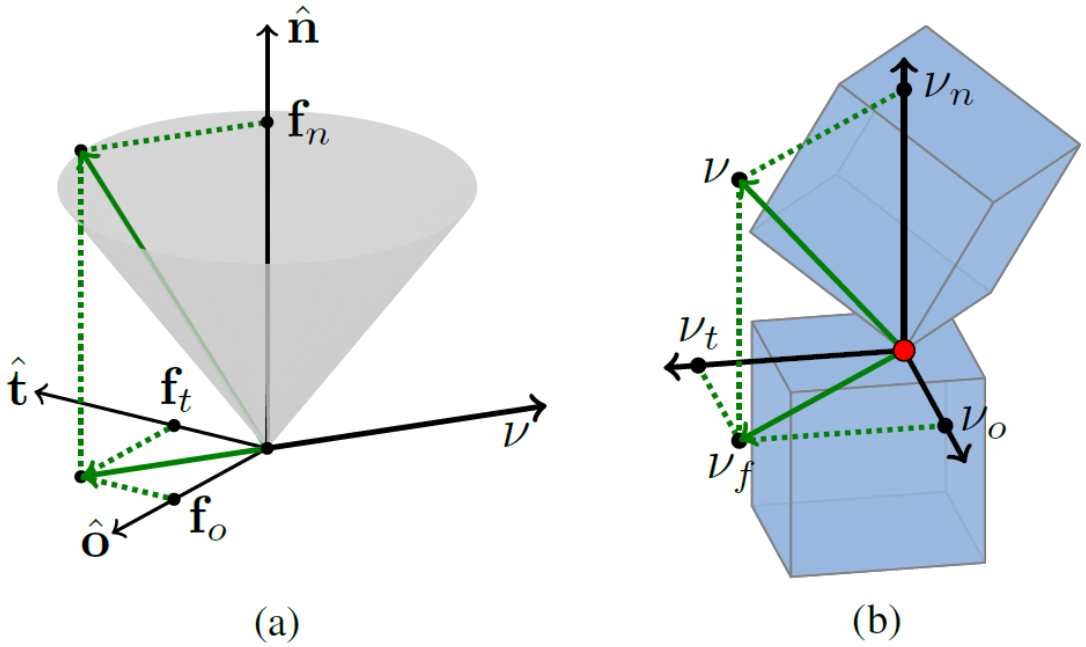


Figure 3.2: (a) The friction cone of a contact and (b) the decomposition of the relative contact velocity [1]

.

In the presence of friction, the tangential and normal transmitted forces are related by Coulomb's law, and the coefficient of friction μ. Consider a contact between a finger and an object, and a frame at contact composed by unit vectors $\hat{n}$ normal to the surface, and $\hat{o}$ and $\hat{t}$ both orthogonal and spanning the tangent plane. Let λ_n, λ_t, and λ_o be the magnitudes of the normal force, and frictional forces along $\hat{t}$ and $\hat{o}$ respectively. We express the total contact force in the local coordinates $\langle \hat{n}, \hat{t}, \hat{o} \rangle$ as $\Lambda = [\lambda_n, \lambda_t, \lambda_o]^T$. Coulomb's friction law is satisfied if the total contact force is inside the following set:

$$FC = \{\lambda_n \hat{n} + \lambda_t \hat{t} + \lambda_o \hat{o} | \lambda_n \geq 0, \lambda_t^2 + \lambda_o^2 \leq \mu \lambda_n^2\} \tag{3.1}$$

known as friction cone (Figure 3.2 (a)), **where the last inequality becomes a strict equality only when there is relative motion at contact**.

3.2 Linear Approximation of Friction Cone

As can be seen in Equation 3.1, the FC brings non-linearity in the dynamics of the system. This can increase the computational cost for finding a solution to the problem. The convention is to replace the friction cone FC with a pyramidal approximation, where each face of the approximated cone represents a linear constraint. The concept behind this approximation is beautifully explicated in [1].

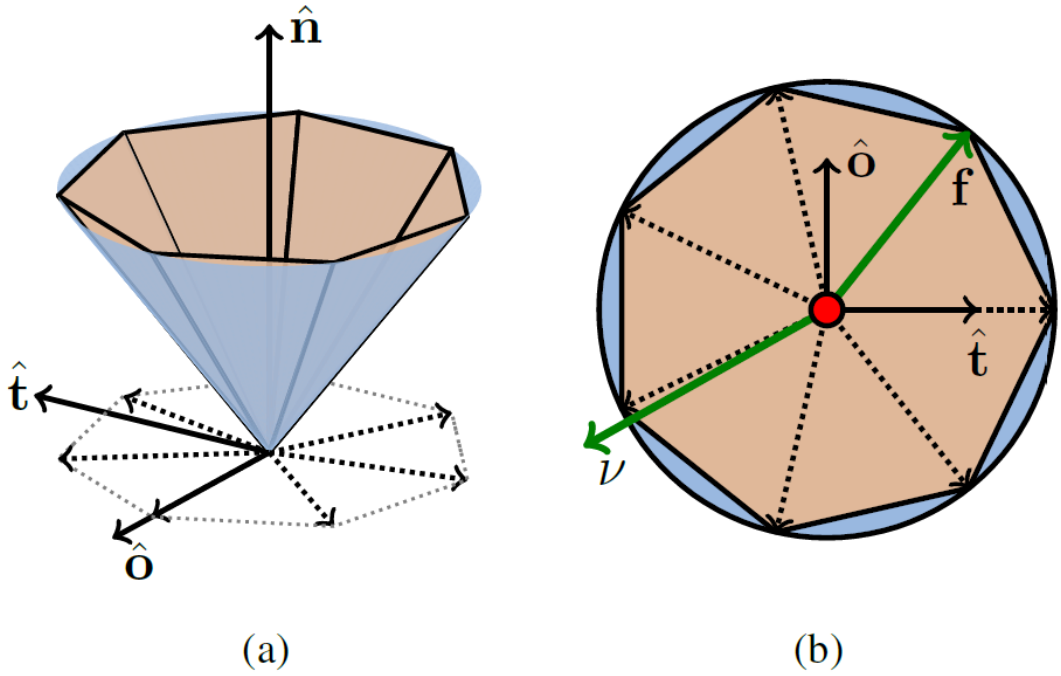


Figure 3.3: (a) The pyramidally approximated friction cone of a contact and (b) the decomposition of the relative contact velocity using spanning vectors [1]

.

The process of linearization for the FC is illustrated in Figure 3.3. The circular friction limit set is a circle of radius $\mu\lambda_n$ (shown in Figure 3.3(b)). This circle is approximated by a convex polygon whose vertices are defined by n unit vectors $\hat{d}_i$ that positively span the friction plane. Consequently, we can always express the frictional force as a linear combination of the spanning vectors $\hat{d}_i$ and non-negative barycentric coordinates β_i as $\sum \beta_i \hat{d}_i$. Therefore, the total contact force can now be expressed in local contact coordinates as $\overline{\Lambda} = [\lambda_n, \beta_1, \beta_2, .., \beta_i, .., \beta_n]^T$.

Collect all generators $\hat{d}_i$ in a matrix D and the magnitudes along those generators in a vector β, we then approximate the friction cone as:

$$\overline{FC} = \{\lambda_n \hat{n} + D.\beta | \lambda_n \geq 0, \beta \geq 0, e^T \beta \leq \mu\lambda_n\} \tag{3.2}$$

where $e^T = [1...1]$.

For the developments in the next few paragraphs, it is important to see that

if $\beta_i = \mu\lambda_n$, then the frictional force is simply $\mu\lambda_n\hat{d}_i$, which is a vertex of the polygon. If $\beta_i + \beta_j = \mu\lambda_n$, where β_i and β_j are adjacent direction vectors, then the frictional force is on an edge of the polygon. Importantly, these are the only ways to represent friction impulses on the boundary of the limit set using barycentric coordinates. All other coordinate combinations define a friction impulse on the interior of the polygon. We also need to firstly briefly understand the definition of Linear Complementarity Problem (LCP), before understanding how friction contact model can be formulated as a LCP.

3.3 Understanding LCP

In physics-based animation Linear Complementarity Problems (LCPs) have historically been used as models of contact forces between rigid bodies [4]. Recently LCPs are being deployed for other types of animation like deformable models, fluids, and granular material. Thus, LCPs are becoming a general important fundamental model. This section briefly describes a LCP.

Let $b, x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ so $y = Ax + b$ For the n-dimensional LCP we have for all $i \in [1...n]$,

$$x_i \geq 0, \tag{3.3}$$

$$(Ax + b)_i \geq 0, \tag{3.4}$$

$$x_i(Ax + b)_i = 0. \tag{3.5}$$

We can write this compactly in matrix-vector notation as

$$x \geq 0, \tag{3.6}$$

$$y \geq 0, \tag{3.7}$$

$$x^T y = 0. \tag{3.8}$$

Even more compactly,

$$0 \leq x \perp y \geq 0 \tag{3.9}$$

3.4 Posing Frictional Contact as a LCP

For a contact, let a_r be the relative acceleration between the object and the contact, and a_r^n be its component in the direction $\hat{n}$ of the local contact coordinate. Then

we can write:

$$0 \leq a_r^n \perp \lambda_n \geq 0 \tag{3.10}$$

This means if there is a separation $a_r^n > 0$ then there can be no normal contact force $\lambda_n = 0$. On the other hand if there is a normal contact force $\lambda_n > 0$, this is non-sticking and there must be a resting contact $a_r^n = 0$.

Coulomb's law and Maximum Dissipation Principle remain to be enforced. Coulomb's law also specifies that the friction force should be on top of the friction cone if and only if there is relative motion at contact. The maximum dissipation principle relates frictional forces and motions at each individual contact. It states that friction is always along the direction that maximizes energy dissipation, which intuitively means that friction will oppose, as much as possible, the direction of motion. The LCP formulation for these two conditions is taken care of collectively. A non-negative slack variable ξ is introduced which can be seen as an indicator of sliding.

$\xi = 0$ indicates rolling while $\xi > 0$ indicated sliding. When $\xi = 0$, the frictional force may be anywhere in the interior of the polygon or on its boundary, but when $\xi > 0$ it must be on the boundary. These requirements suggest the following complementarity condition.

$$0 \leq (\mu\lambda_n - e^T\beta) \perp \xi \geq 0 \tag{3.11}$$

To enforce the maximum dissipation, one must introduce another condition that allows only one or two consecutive barycentric coordinates to be nonzero. One way to accomplish this is by the introduction of another complementarity constraint that maps the relative acceeration a_r onto the spanning vectors D (this can be accomplished by $D^T a_r$) and finally an i such that $\hat{d}_i$ is most directly opposite to a_r. The following complementarity condition **in conjucation with** Equation 3.11 identifies the correct $\hat{d}_i$:

$$0 \leq (\xi e + D^T a_r) \perp \beta \geq 0 \tag{3.12}$$

4

Grasp Modeling

4.1 Grasp Matrix

The grasp matrix G defines the span of all possible wrenches transmitted by all contacts to the object, in the object reference frame. Following the notation in Chapter 3, for any given contact point i, the matrix $G_i = [\hat{n}_i, \hat{t}_i, \hat{o}_i]$ linearly spans the set of forces that contact i is capable of transmitting to the object. Then the contact force transmitted by contact i to the object is $G_i.\Lambda_i$. The grasp matrix G is built by concatenating matrices G_i for all the contacts:

$$G = [G_1 G_2 .... G_j G_{ext1} ... G_{extk}] \tag{4.1}$$

We can collect Λ_i's for all the contacts into a big vector Λ which allows us to compute the total wrench on the object as $G.\Lambda$.

We can write equivalent expressions for the polyhedral approximation of friction cone, where now $\overline{G_i} = [\hat{n}_i \hat{d}_1 .... \hat{d}_n]$ spans the set of forces that contact i can transmit to the object, and $\overline{\Lambda}_i$ is the column vector $[\lambda_{ni}, \beta_{1i}, ..., \beta_{ni}]^T$ with their corresponding magnitudes. Analogously, we build $\overline{G}$ and $\overline{\Lambda}$ by concatenating them for all contacts. Then, the contact force transmitted by point i is $\overline{G}_i.\overline{\Lambda}_i$, and the total wrench applied on the object $\overline{G}.\overline{\Lambda}$.

With this notation, we can finally write down the condition for the stability of the grasp (force balance on the object) as:

$$\overline{G}.\overline{\Lambda} + \vec{w} = M.\vec{a}_{obj} \tag{4.2}$$

where $\vec{w}$ is the gravitational wrench applied on the object, M is the generalized inertia matrix and $\vec{a}_{obj}$ is the acceleration of the object in the object frame.

4.2 Hand Jacobian Matrix

The hand jacobian matrix J encodes the motion of actuator joints into local motions at contact points. We construct it as $J^T = [J_1^T ....J_m^T]$, where J_i has one column for each hand actuator and expresses the induced local velocity at point contact i in the local frame $\langle \hat{n}_i, \hat{t}_i, \hat{o}_i \rangle$. We extend the hand jacobian to include the effect of the external pusher:

$$J^T = [J_1^T ....J_j^T J_{ext1}^T ....J_{extk}^T] \tag{4.3}$$

4.3 Contact Acceleration

The local accelerations at contacts are related to the accelerations of the object, hand, and pusher. We can look at the motion of a contact point from two perspective:

- From the object point of view, the grasp matrix relates the acceleration of the object to the acceleration at all contacts as $G^T.\vec{a}_{obj}$.
- From the hand point of view, the hand jacobian matrix relates the accelerations of the actuators $\ddot{\theta}$ to the accelerations at all contacts as $J.\ddot{\theta}$

Note that $\ddot{\theta}$ includes both the motion of the actuators of the gripper, which (although not necessary) we will assume to be zero, and the "virtual" actuators of the pusher, for which the Jacobian is the identity. The relation formulates then as:

$$a_r = G^T.\vec{a}_{obj} - J.\ddot{\theta} \tag{4.4}$$

4.4 User-Desired Constraints

It can be required in an in-hand manipulation task for the object to follow a desired directions. This can be written as follows:

$$a_{obj} = b \tag{4.5}$$

where, b is a constant column vector.

5

Solving Mixed LCP

The collection of all constraints define a solution space for contact forces, contact accelerations, and object acceleration. The problem has the form of a linear complementarity problem (LCP) with the addition of extra linear constraints. This is commonly referred to as mixed LCP where some of the variables are subject to complementarity constraints and others are not.

Let x be the collection of all the variables, in the space of which the problem is formulated. Then, Equations 3.9 - 3.11 and Equations 4.2,4.4 and 5.1 can now be written as:

$$0 \leq (M.x - q) \perp x \geq 0 \tag{5.1}$$

$$I.x = J \tag{5.2}$$

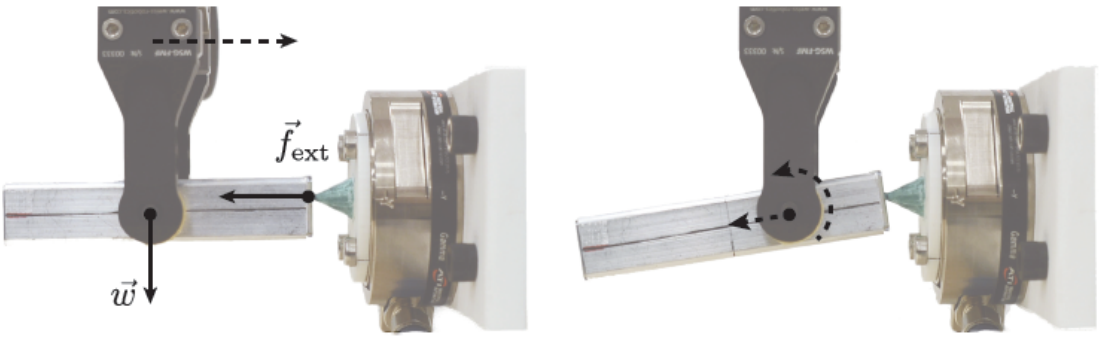
The above equations can be solved by reformulating them as a quadratic optimization problem. A LCP can be seen as a solution to the Karush-Kunn-Tucker (KKT) constraints of the quadratic problem instance [4]. A quadratic problem can in general be formulated as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2} x^T H x + f^T x \\ \text{subject to} \quad & A.x \leq b \\ & A_{eq}.x = b_{eq} \\ & lb \leq x \leq ub \end{aligned} \tag{5.3}$$

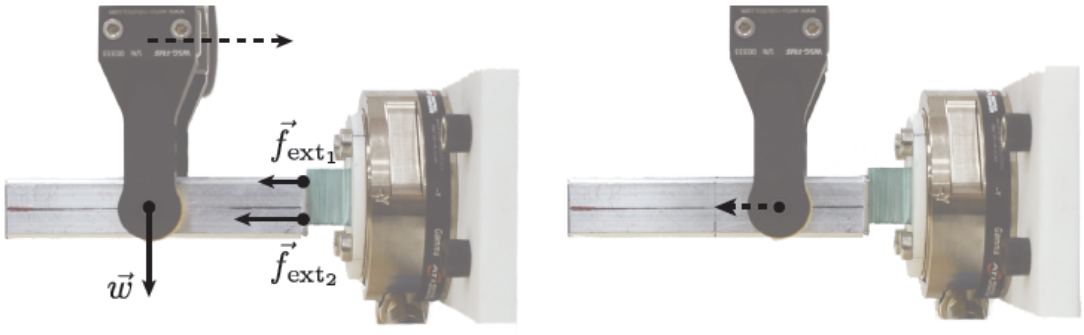
Equations 5.1-5.2 can be reformulated into the form of Equations 5.3 and can be solved using a Quadratic Program Solver. This work used MATLAB's `quadprog` to solve the formulated quadratic optimization problem.

6

Validating the formulation on Primitive Prehensile Pushing Actions



(a) Sliding with point contact.



(b) Sliding with line contact.

Figure 6.1: Sliding with point and line contacts [2]

.

The problem formulation was validated for a sliding action as shown in Figure 6.1. Point as well as linear contact pusher was considered for the problem of straight

sliding. It was observed that if the pusher makes a point contact, under the effect of gravity, it is difficult to push the object straight and the object starts rotating. Straight sliding is possible only for a very specific location below the center of gravity of the object. Instead, when pushed with a line contact, the resulting motion of the object is always straight. A line pusher modeled as two rigidly connected points at its ends, a pair of coupled forces both pushing straight but with different magnitudes are responsible for compensating for gravity.

6.1 MATLAB Code

This section provides the MATLAB Code for the sliding problem considered in Figure 6.1(b). As can be seen intuitively, forces of varying magnitudes are required at each of the external point contacts. Any different required problem can be solved by changing the positions of the contact points inside the code.

```
1
2
3  factor=1; % To change mass of the object by a factor
4  sides=16; % Number of vertices used for the pyramidal
       approximation
5 m=factor*1; %Mass of the object
6  g=10; %Gravity
7  accdes=-5; % Desired linear acceleration along y direction
       in object frame
8  e=ones(sides,1); % Required in Complementarity conditions
9 mu=1; % Coefficient of friction
10  fg=5*sqrt(2); % Gripper Force
11
12
13 % Contact Positions of the gripper (r1 and r2) and external
       pusher (r3 and r4)
14  r1=[0.01;-0.01;0];r2=[-0.01;-0.01;0];
15  r3=[0;0.05;0.01];r4=[0;0.05;-0.01];
16
17
18 %Inertia Matrix
19 M=[m 0 0 0 0 0;
20     0 m 0 0 0 0;
```

```
21      0  0 m 0  0  0;
22      0  0  0 m*8.6667e-04  0  0;
23      0  0  0  0 m*6.6667e-05  0;
24      0  0  0  0  0 m*8.6667e-04];
25
26
27  % Symbolic Variables for all the variables in the space of
        which the problem is formulated
28  xs = sym(sym('x%d',[(sides+1)*4+4 1]),'real');
29
30
31  % Grasp Matrix
32   c1v_norm=[-1 0 0;0 -1 0;0 0 1];c2v_norm=[1 0 0;0 1 0;0 0
         1];c3v_norm=[0 1 0;-1 0 0;0 0 1];c4v_norm=[0 1 0;-1 0
         0;0 0 1];
33   c1v=c1v_norm*contact_vectors([1;0;0],sides);
34   c2v=c2v_norm*contact_vectors([1;0;0],sides);
35   R_z=[0 1 0;-1 0 0;0 0 1];
36   c3v=R_z*contact_vectors([1;0;0],sides);
37   c4v=R_z*contact_vectors([1;0;0],sides);
38   d1=c1v(:,2:sides+1);d2=c2v(:,2:sides+1);d3=c3v(:,2:sides+1)
         ;d4=c4v(:,2:sides+1);
39   Grasp=[c1v c2v c3v c4v];
40
41
42   % Contact Force Weights
43   Lambda1=xs(1:sides+1,1);Lambda2=xs(sides+2:2*sides+2,1);
         Lambda3=xs(2*sides+3:3*sides+3,1);Lambda4=xs(3*sides
         +4:4*sides+4,1);
44   fn1=xs(1,1);fn2=xs(sides+2,1);fn3=xs(2*sides+3,1);fn4=xs(3*
         sides+4,1);
45   c1w=xs(2:sides+1,1);c2w=xs(sides+3:2*sides+2,1);c3w=xs(2*
         sides+4:3*sides+3,1);c4w=xs(3*sides+5:4*sides+4,1);
46   Lambda=[Lambda1;Lambda2;Lambda3;Lambda4];
47
48
49   %Contact Forces
```

```
50  F_c1=c1v*Lambda1; F_c2=c2v*Lambda2; F_c3=c3v*Lambda3; F_c4=c4v
        *Lambda4;
51  F_contact=F_c1+F_c2+F_c3+F_c4;
52
53
54  %Contact Torques
55  F_tor=cross(r1,F_c1)+cross(r2,F_c2)+cross(r3,F_c3)+cross(r4
        ,F_c4);
56
57  %Contact Wrench
58  F_wrench=[F_contact;F_tor];
59
60  %Gravity Wrench
61  g_wrench=[0;0;-m*g;0;0;0];
62
63  % System Dynamics
64  acc_b = M\(g_wrench + F_wrench);
65
66  % Relative Accelerations at contacts in local frames
67  trans1=[c1v_norm zeros(3,3);skew(r1)*c1v_norm c1v_norm];
68  trans2=[c2v_norm zeros(3,3);skew(r2)*c2v_norm c2v_norm];
69  trans3=[c3v_norm zeros(3,3);skew(r3)*c3v_norm c3v_norm];
70  trans4=[c4v_norm zeros(3,3);skew(r4)*c4v_norm c4v_norm];
71  acc_1=trans1*acc_b; acc_2=trans2*acc_b;
72  % Assumed that there is no relative motion between pusher
        and object
73  acc_3=zeros(6,1); acc_4=zeros(6,1);
74
75  % Auxillary variables required in complementarity
        conditions
76  a1=xs((sides+1)*4+1,1); a2=xs((sides+1)*4+2,1); a3=xs((sides
        +1)*4+3,1); a4=xs((sides+1)*4+4,1);
77
78
79
80 % Gather Quadratic Problem Matrices
81
```

```
82 % For Cost Function: H and f
83   eq9=(mu*fn1 - e.'*c1w);
84   eq10=(mu*fn2 - e.'*c2w);
85   eq11=(mu*fn3 - e.'*c3w);
86   eq12=(mu*fn4 - e.'*c4w);
87
88   eq2=a1*e + d1.'*acc_1(1:3,1);
89   eq4=a2*e + d2.'*acc_2(1:3,1);
90   eq6=a3*e + d3.'*acc_3(1:3,1);
91   eq8=a4*e + d4.'*acc_4(1:3,1);
92
93   eq1=acc_1(1,1);
94   eq3=acc_2(1,1);
95   eq5=acc_3(1,1);
96   eq7=acc_4(1,1);
97
98 eqn=[eq1;eq2;eq3;eq4;eq5;eq6;eq7;eq8;eq9;eq10;eq11;eq12];
99 [H1,h1]= equationsToMatrix([eqn], [xs]);
100 H=2*double(H1.');f=-1*double(h1);
101
102
103
104 %InEquality Constraints: A and b
105
106 ieq1=[fn1;fn2;fn3;fn4];
107 ieq2=[acc_1(1,1);acc_2(1,1);acc_3(1,1);acc_4(1,1)];
108 ieq3=[a1;a2;a3;a4];
109 ieq4=[eq11;eq12];
110 ieq5=[c1w;c2w;c3w;c4w];
111 ieq6=[eq2;eq4;eq6;eq8];
112 ieq7=acc_b(4,1)+1e-1;ieq8=1e-1-acc_b(4,1);
113 ieq9=acc_b(3,1)+1e-1;ieq10=1e-1-acc_b(3,1);
114 ieq=[ieq1;ieq2;ieq3;ieq4;ieq5;ieq6;ieq7;ieq8;ieq9;ieq10];
115 [A1,b1]=equationsToMatrix([ieq], [xs]);
116 A=-1*double(A1);b=-1*double(b1);
117
118
```

```
119
120 %Equality Constraints: Aeq and beq
121 eeq1=fn1-(fg);
122 eeq2=fn2-(fg);
123 eeq3=acc_b(2,1)-accdes;eeq4=acc_b(1,1);
124 eeq5=F_tor(2:3,1) -[0;0];
125 eeq6=cross(r1,F_c1)+cross(r2,F_c2);eeq7=cross(r3,F_c3)-cross
    (r4,F_c4);
126 eeq=[eeq1;eeq2;eeq5;eeq3;eeq4;eq9;eq10];
127 [Aeq1,beq1]=equationsToMatrix([eeq], [xs]);
128 Aeq=double(Aeq1); beq=double(beq1);
129
130
131
132 % Solve the Quadratic Problem Using quadprog
133 options = optimoptions('quadprog','Algorithm','active-set','
    MaxIter',1000,'Display','none');
134 [soln,fval]=quadprog(H,f,A,b,Aeq,beq,[],[],[],options);
135
136 % Contact Force Weights
137 Lambda11=soln(1:sides+1,1);Lambda22=soln(sides+2:2*sides
    +2,1);Lambda33=soln(2*sides+3:3*sides+3,1);Lambda44=soln
    (3*sides+4:4*sides+4,1);
138 fn11=soln(1,1);fn22=soln(sides+2,1);fn33=soln(2*sides+3,1);
    fn44=soln(3*sides+4,1);
139 c1ww=soln(2:sides+1,1);c2ww=soln(sides+3:2*sides+2,1);c3ww=
    soln(2*sides+4:3*sides+3,1);c4ww=soln(3*sides+5:4*sides
    +4,1);
140 Lambdaa=[Lambda11;Lambda22;Lambda33;Lambda44];
141
142 %Contact Forces
143 F_c11=c1v*Lambda11;F_c22=c2v*Lambda22;F_c33=c3v*Lambda33;
    F_c44=c4v*Lambda44;
144 F_contactt=F_c11+F_c22+F_c33+F_c44;
145
146 %Contact Torques
```

```
147  F_torr=cross(r1,F_c11)+cross(r2,F_c22)+cross(r3,F_c33)+
         cross(r4,F_c44);
148
149  %Contact Wrench
150  F_wrenchh=[F_contactt;F_torr];
151
152  % Display Results
153  disp1='Fext';disp(disp1)
154  soln(2*sides+3,1)
155  soln(3*sides+4,1)
156
157
158  % To check correctness of the solution
159
160  acc_bb = M\(g_wrench + F_wrenchh);
161
162  ej9=(mu*fn11 - e.'*c1ww);
163  ej10=(mu*fn22 - e.'*c2ww);
164  ej11=(mu*fn33 - e.'*c3ww);
165  ej12=(mu*fn44 - e.'*c4ww);
166
167  a11=soln((sides+1)*4+1,1);a22=soln((sides+1)*4+2,1);a33=
         soln((sides+1)*4+3,1);a44=soln((sides+1)*4+4,1);
168  acc_11=trans1*acc_bb;acc_22=trans2*acc_bb;acc_33=zeros(6,1)
         ;acc_44=zeros(6,1);
169
170  ej2=a11*e + d1.'*acc_11(1:3,1);
171  ej4=a22*e + d2.'*acc_22(1:3,1);
172  ej6=a33*e + d3.'*acc_33(1:3,1);
173  ej8=a44*e + d4.'*acc_44(1:3,1);
174
175  ej1=acc_11(1,1);
176  ej3=acc_22(1,1);
177  ej5=acc_33(1,1);
178  ej7=acc_44(1,1);
179
180  ej=[ej1;ej2;ej3;ej4;ej5;ej6;ej7;ej8;ej9;ej10;ej11;ej12];
```

Function Required:

```
% Generation of spanning vectors
% Takes as input j=normal vector, u=number of sides

function y = contact_vectors(j,u)
y=zeros(3,u+1);
y(:,1)=j;
for i=1:u
    y(2:3,i+1)=[cos((i-1)*(2*pi/u));sin((i-1)*(2*pi/u))];
end
```

Output:

```
>> Code
Fext

ans =

  14.0999

ans =

   0.5667
```

7

Discussion and Future Work

This work studied in-hand manipulation of a grasped object using extrinsic dexterity. Given a grasp configuration, gripping forces, and the location and motion of a pusher, the problem formulation estimates the minimum force required from an external pusher to break the grasp equilibrium in the desired directions. To achieve this, the friction contact model was posed as a LCP. The friction contact formulation along with the imposed equality and inequality constraints results in a Mixed-LCP. The Mixed-LCP was solved by re-formulating it as a quadratic optimization problem.

The formulation was validated by simulating a primitive sliding action. The results corroborate with the ones that can be found in [2].

One major problem faced during this work was that the MATLAB's `quadprog` solver allows some tolerances in the constraints and objective function. This many times resulted in breaking of the Coulomb's Law of friction, thereby providing erroneous results. Future work can involve usage of more accurate solvers which can provide solution with even lower tolerances. Also, the plot of external force versus position of the contact points as they are translate along specific directions can be analyzed for any symmetricities or for finding positions which require minimum external force for breaking the grasp equilibrium.

Bibliography

[1] Jan Bender, Kenny Erleben, and Jeff Trinkle. "Interactive simulation of rigid body dynamics in computer graphics". In: *Computer Graphics Forum*. Vol. 33. 1. Wiley Online Library. 2014, pp. 246–270.

[2] N Chavan Dafle and A Rodriguez. "Prehensile Pushing: In-hand Manipulation with Push-Primitives". In: *IEEE/RSJ Intl. Conf on Intelligent Robots and Systems*. 2015.

[3] Nikhil Chavan Dafle et al. "Extrinsic dexterity: In-hand manipulation with external forces". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 1578–1585.

[4] Kenny Erleben. "Numerical methods for linear complementarity problems in physics-based animation". In: *Acm Siggraph 2013 Courses*. ACM. 2013, p. 8.

[5] Kevin M Lynch and Matthew T Mason. "Stable pushing: Mechanics, controllability, and planning". In: *The International Journal of Robotics Research* 15.6 (1996), pp. 533–556.

[6] J Kenneth Salisbury and B Roth. "Kinematic and force analysis of articulated mechanical hands". In: *Journal of Mechanisms, Transmissions, and Automation in Design* 105.1 (1983), pp. 35–41.

[7] Gregory P Starr. "Cartesian stiffness control of the jpl/stanford/salisbury hand". In: *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*. IEEE. 1988, pp. 636–637.