COORDINATED NONPREHENSILE MANIPULATION FOR PARTS TRANSFER: MECHANICS, CONTROL, AND PLANNING

by

Qingguo Li

MA.Sc., Northwestern Polytechnical University, Xi'an, P.R.China, 1995

B.A.Sc., Northwestern Polytechnical University, Xi'an, P.R.China, 1992

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Engineering Science

SIMON FRASER UNIVERSITY

Spring 2006

APPROVAL

Name: Qingguo Li

Degree: Doctor of Philosophy

Title of Thesis: Coordinated Nonprehensile Manipulation for Parts Transfer: Mechanics, Control, and Planning

Examining Committee: Dr. K.K. Gupta, Chair
Professor, School of Engineering Science
Simon Fraser University

Dr. S. Payandeh, Senior Supervisor

Dr. M. Saif, Supervisor

Dr. M. Trummer, Supervisor

Dr. W.A. Gruver, Internal Examiner

Dr. K. Goldberg, External Examiner
Professor, University of California at Berkeley

Date Approved: Apr. 18/06

Abstract

This thesis focuses on coordinated nonprehensile manipulation to accomplish a parts transfer task. Parts transfer refers to moving parts from a known initial configuration (position and orientation) to a goal configuration. This task is motivated by automatic manufacturing applications. The problem is studied under quasi-static and dynamic settings. By exploring the task mechanics and geometry, three nonprehensile (grasp-less) manipulation methods are developed. This thesis describes their mechanics, control and planning algorithms.

The first part of the thesis investigates quasi-static cooperative nonprehensile manipulation. Two methods are studied for the parts transfer task. The problems of controllability and planning are studied for these methods. The aim of controllability is to determine whether the goal configuration of the part is reachable by cooperative nonprehensile manipulation, and the objective of planning is to find a cooperative motion (or action) of the agents to bring the part to the goal configuration. The first system demonstrates that a fixed-radius rotational push and a linear normal push are sufficient to manipulate an object in the plane. By using optimal control theory, a planner is developed to find the optimal solution. In the second system, new manipulation primitives such as, equilibrium push and non-equilibrium push, are introduced for manipulating convex parts. We prove the configuration controllability of the object under these pushes. A fully analytical planner is developed to solve the optimal sequence. Simulations and experiments are conducted to demonstrate the proposed manipulation methods.

The second part of the thesis investigates dynamic cooperative manipulation. The motion of the object under manipulation consists of two phases: acceleration by cooperative dynamic pushing, and free sliding with initial velocity. For the free sliding

problem, a free boundary value problem formulation (FBVP) is developed to find the desired release velocity. After transforming the FBVP into a two-point boundary value problem, a set of algorithms is developed to solve the planning problem. A degree of freedom mechanism is implemented to demonstrate the planning method. For the cooperative dynamic pushing problem, a centralized planning method is developed by integration of the backstepping design and quadratic programming under the known pressure distribution assumption. A game theoretic approach is proposed to solve the acceleration problem in the case of uncertain pressure distribution.

Dedication

To the loving memory of my father

Acknowledgments

I would like to thank the following people for contributing so much to the completion of this thesis. With the leadership, example, and support of each of the individuals below, I have managed to learn, enjoy, and grow through what will surely be some of the best years of my life.

I wish first to express my sincere gratitude to my supervisor Dr. Shahram Payandeh, for your guidance, encouragement and support that you have given me over the past six years. Thank you very much for sharing your insight on robotics, and for giving me freedom to explore different things.

I would like to think Dr. Manfred Trummer, for your tireless help and support. Thank you for leading me into the fascinating world of numerical mathematics and computing. More important, teaching me that mathematics do work for engineering problems. Special thanks to Dr. Mehrdad Saif, for teaching me system and control theory. Thank you for giving the hardest assignments and exams in your classes, upon which I built much of my knowledge and problem solving skills. I give thanks to Dr. William A. Gruver for your advice of the style and writing of my thesis, and for providing the robot that made the experiment possible. Special thanks to Dr. Kamal K. Gupta for chairing my thesis defence. Thank you Dr. Max Donelan and Dr. Andy Hoffer for giving me the opportunity to work with you. Thank you also to the staffs in the department. Thank you for helping make confusing paperwork more understandable, and for helping make my experience at SFU wonderful.

Special thanks to Sheena Frisch, Norm Jaffe, Thomas Edward Johnson, Dr. W. Craig Scratchley, Zhengwan Yao for your valuable comments on the manuscript of this thesis.

Finally, thank you so much to my past and present labmates, Wayne Chen, Temei Li, Yi Li, Herbert F. Noriega, Pengpeng Wang, Jian Zhang and Hui Zhang, for always being there with help and encouragement.

My sincere appreciation goes to brothers and sisters from SFU Christian fellowship, Weitian Chen, Fang Liu and Hui Qu, Jinyun Ren and Yifeng Huang, Tong Jin. Thank you for walking with me, otherwise, I will be lonely. From you, I learned life is more than a Ph.D. thesis.

Finally, I am grateful to my family. To my wife Xuan Geng for your unconditional love and support that you have given me. I could not have done it alone. I am indebted to my sons Eric and David, who bring a lot of joys even when I was stressed and burnt out. Eric asks a lot of questions. Specially, always asks me "Daddy, when are you going to graduate?" - a driving force to finish; David, always having smiling face there waiting for me, who makes life fun. To my mom and dad, brother and in-laws, for your supports, patience, and sacrifices.

Table of Contents

Approval ii

Abstract iii

Dedication v

Acknowledgments vi

Table of Contents viii

List of Figures xii

List of Tables xvii

Table of Acronyms xviii

Nomenclature xix

1 Introduction 1

1.1 Overview 2

1.2 Related work 8

1.2.1 Pushing 8

1.2.2 Nonprehensile Manipulation 12

1.2.3 Free Sliding Problem 17

1.3 Thesis Contributions 18
1.4 Thesis Outline 20
1.5 Publication Notes 21

2 Parts Transfer by an Ability-Limited Mobile Robot Push **22**
2.1 The Push System and the Optimal Planning Problem 23
2.2 Analysis of the Optimal Trajectory 30
2.3 Computation of the Optimal Trajectory 36
2.3.1 The Properties of an RPR Trajectory 37
2.3.2 The Properties of a PRP Trajectory 39
2.3.3 Computational Procedure of the Optimal Trajectory 43
2.3.4 Extensions to General R and P Motions 46
2.4 Manipulation Examples and Experimental Results 47
2.5 Discussion 54

3 Parts Transfer by Two-agent Cooperative Push **56**
3.1 Problem Definition 58
3.1.1 Two-agent Cooperative Push 58
3.1.2 Types of Pushes 60
3.2 Locating the Virtual Edges 63
3.2.1 Parametric Representation of a Convex Object 63
3.2.2 Finding the Equilibrium Orientations 66
3.3 Characterizing the Pushes 69
3.3.1 Performing the Equilibrium Push 69
3.3.2 Performing the Non-equilibrium Push 70
3.4 Formulation of Planning Problem and Controllability Analysis 75
3.4.1 Switched System Formulation of Two-agent Cooperative Push 75

3.4.2 Controllability Analysis 77
3.5 Planning Two-agent Cooperative Push 81
3.5.1 The Controller for Non-equilibrium Pushes 81
3.5.2 The Reachable Region for Equilibrium Pushes 83
3.5.3 The Optimal Controller for Equilibrium Pushes 93
3.5.4 Implementation of the Planner 98
3.6 Manipulation Examples and Experimental Results 99
3.7 Discussion 116

4 Planning the Velocity of Free Sliding Objects 117
4.1 Problem Formulation 118
4.1.1 The Model of a Sliding Object 118
4.1.2 Formulation of the Planning Problem 123
4.2 Planning Methods and Implementation 126
4.2.1 Basic Shooting Method 126
4.2.2 Shooting with the Broyden Update 130
4.2.3 Implementation with a Line Search Strategy 131
4.3 Numerical and Experimental Results 137
4.3.1 Numerical Simulations 137
4.3.2 Experimental Setup 141
4.3.3 Experimental Results 145
4.4 Discussion 147

5 Acceleration of the Object by Cooperative Dynamic Pushing 149
5.1 Problem Formulation 151
5.1.1 Model of Cooperative Dynamic Pushing 151
5.1.2 The Cooperative Push Planning Problem 153
5.2 The Two-level Planning Method 154
5.2.1 Backstepping Design for the Generalized Force Controller . . . 154
5.2.2 Coordination as a Quadratic Programming Problem 157
5.3 Simulation Results 157

5.3.1 Coordination under Two-agent Manipulation 158
5.3.2 Coordination under Three-agent Manipulation 162
5.4 Discussion 163

6 Cooperative Dynamic Pushing under Uncertainty **164**
6.1 Problem Formulation 165
6.1.1 Model of Cooperative Dynamic Pushing with Uncertainty 165
6.1.2 The Cooperative Push Planning Problem 168
6.2 Planning the Cooperative Pushing 169
6.2.1 H^∞ Design for the Generalized Force Controller 170
6.2.2 Coordination as a Cooperative Game 175
6.3 Simulation Results 176
6.3.1 Coordination under Three-agent Pushing 177
6.3.2 Coordination under Two-agent Pushing 179
6.4 Discussion 181

7 Conclusions and Future Work **191**
7.1 Conclusions 191
7.2 Future Work 192

A Computation Methods for Polygonal Objects **196**
A.1 Computation of Mass, Moment of Inertia and Friction 196
A.1.1 Triangular Partition of a Polygonal Object 196
A.1.2 Numerical Double Integration Over Triangular Region 198

Bibliography **202**

List of Figures

1.1 Physical fence and virtual fence pushing 5
1.2 Dynamic distributed manipulation 6

2.1 Two push primitives 23
2.2 An example of a pushing trajectory with a known sequence of control signal u_1. u_1 takes $\{1, 0, 1, 0\}$, and switchings occurs at $t = 2$ sec., 5 sec., 8 sec.. The sequence is $RPRP$. The position and orientation of the object are plotted. 27
2.3 Trajectory of rotate-push-rotate(RPR). 28
2.4 Trajectory of PRPR with $H = 0$ 36
2.5 A sequence of PRP 41
2.6 Optimal RPR trajectory. The grey triangular block located at $(0, 0)$ is the initial configuration. The grey triangular block located at $(5, 0)$ is the goal configuration. Starting from the initial configuration, the triangles illustrate the sequence of object positions during RPR pushes. 48
2.7 Optimal PRP trajectory. The grey triangular block located at $(0, 0)$ is the initial configuration. The grey triangular block located at $(5, 0)$ is the goal configuration. Starting from the initial configuration, the triangles illustrate the sequence of object positions during PRP pushes. 49
2.8 Optimal RPR trajectory. The grey triangular block located at $(0, 0)$ is the initial configuration. The grey triangular block located at $(5, 5)$ is the goal configuration. Starting from the initial configuration, the triangles illustrate the sequence of object position during RPR pushes. 50
2.9 Mobile robot for experiments. 51

2.10 Planned optimal RPR trajectory. The grey triangular block located at $(0, 0)$ is the initial configuration. The grey triangular block located at $(-2.3, -2.7)$ is the goal configuration. Starting from the initial configuration, the triangles illustrate the sequence of object positions during RPR pushes. 52

2.11 Optimal trajectory executed by a mobile robot. (a) depicts the snapshot that the robot starts the first R motion, (b) depicts the snapshot that the robot starts the P motion, (c) depicts the snapshot that the robot starts the second R motion, (d) depicts the snapshot that the robot attains the goal. 53

3.1 A sequence of a two-agent cooperative push to transfer a triangular object from an initial configuration to a goal configuration in the world coordinate X_wOY_w. XOY defines the agent frame, and xoy is the object frame. 59

3.2 Complete pushes. (a) Equilibrium push. (b). Non-equilibrium push. The dot indicates the center of mass. 61

3.3 Incomplete pushes. (a) COM outside agents. (b). Pushing at the same edge. The dot indicates the center of mass. 61

3.4 Virtual fence. (a) Two-agent cooperative push. (b). Virtual fence push. 62

3.5 Two agents push a polygonal object 64

3.6 Switching structure between equilibrium and non-equilibrium pushes 76

3.7 Equilibrium push vectors span R^2 space 80

3.8 Reachable cone for equilibrium pushes. The push direction vectors are given by $\mathbf{v}_0, \cdots, \mathbf{v}_3$, and cone ECD gives the reachable goal position. 87

3.9 Dead zone of the goal position for the CW and CCW sequences of equilibrium pushes when $\Delta\theta = \pi$. The initial configuration of the object is labeled with 0. It requires four non-equilibrium pushes to achieve $\Delta\theta = \pi$. The associated push direction vectors $\mathbf{v}_0, \cdots, \mathbf{v}_4$ positively span the CW and CCW reachable zone. The goal position inside the dead zone can not be reached by the sequence of equilibrium pushes. 88

3.10 Enlarged reachable region by a modified CW-CCW sequence. After introducing a CW non-equilibrium push, an additional pushing direction vector $\mathbf{v}_-$ can be used by an equilibrium push. compared with Fig. 3.8, Cone DCF is the enlarged portion of the reachable region. 90

3.11 Classification of non-equilibrium push sequences 91

3.12 Construction of CW-CCW and CCW-CW sequences 92

3.13 Plot of objective function $f(\alpha, \beta)$ with $|\mathbf{b}| = 1$. $f(\alpha, \beta)$ is a monotonic increase function of parameters α and β. 98

3.14 The depth function for the triangular object. The equilibrium orientation occurs at $\theta = 57$deg in the agent frame. 101

3.15 CCW non-equilibrium push. (a). One agent makes contact with the object at the edge BC. (b). After a CCW rotation, the object reaches its equilibrium orientation. 102

3.16 CW non-equilibrium push. (a). One agent makes contact with the object at the edge AB. (b). After a CW rotation, the object reaches its equilibrium orientation. 104

3.17 CCW non-equilibrium push sequence and the reachable region. Two goal positions are labeled as Goal 1 and Goal 2. The push direction vectors of equilibrium pushes are given by $\mathbf{v}_0, \cdots, \mathbf{v}_3$. Vector $\mathbf{b}_1$ is positively spanned by the push direction vectors, and vector $\mathbf{b}_2$ is not positively spanned by the push direction vectors. 105

3.18 Optimal CCW push sequence for goal one. The optimal sequence consists of three non-equilibrium pushes and two equilibrium pushes. These two equilibrium pushes are adjacent to each other. 106

3.19 CW-CCW non-equilibrium push sequence and the reachable region. By introducing a CW non-equilibrium push to the CCW sequence, A new push direction vector $\mathbf{v}_-$ helps enlarge the reachable region of the equilibrium push sequence. The push direction vectors $\mathbf{v}_-, \cdots, \mathbf{v}_3$ span R^2 space. With these pushing directions, any position in the plane can be reached by a sequence of equilibrium pushes. 107

3.20 Optimal CW-CCW push sequence for goal 2. The optimal sequence consists of one CW non-equilibrium push, four CCW non-equilibrium pushes and two equilibrium pushes. These two equilibrium pushes are adjacent to each other. There only exists a CW non-equilibrium push between them. 108
3.21 CW non-equilibrium push sequence and the reachable region. The CW push direction vectors are given by $\mathbf{v}_0, \cdots, \mathbf{v}_6$. These vectors span R^2 space. The position vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are positively spanned by the vectors; thus, Goal 1 and Goal 2 can be reached by a sequence of equilibrium pushes. 109
3.22 Optimal CW push sequence for goal 1. This sequence consists of six non-equilibrium pushes and two equilibrium pushes. The first and the final pushes are equilibrium pushes. 110
3.23 Optimal CW push sequence for goal 2. This sequence consists of six non-equilibrium pushes and two equilibrium pushes. The first and the final pushes are equilibrium pushes. 111
3.24 Experimental setup. Two motors are controlled to adjust the spatial relationship of the agents. The MOTOMAN robot drives the two-fingered gripper to push the object. 113
3.25 Object under a CCW non-equilibrium push. (a). Before contact, (b) one agent makes contact, (c). the object rotates between the agents, (d) the object reaches its equilibrium orientation. 114
3.26 Object under a CW non-equilibrium push. (a). Before contact, (b) one agent makes contact, (c). the object rotates between the agents, (d) the object reaches its equilibrium orientation. 115

4.1 Motion of a planar object 119
4.2 Geometries of parts (unit: cm) 138
4.3 Iteration results for the rectangular object: (a) shows the final configuration after each iteration for a uniform rectangular object; (b)-(d) show the planned initial velocity $\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0)$ after each iteration. 140
4.4 Experiment setup 143

4.5 Description of experiment parameters 144
4.6 Results of qualitative comparison 146

5.1 Cooperative dynamic pushing 151
5.2 Manipulation of triangular object 159
5.3 Force coordination under two-agent manipulation 160
5.4 Force coordination under three-agent manipulation 161
5.5 Force coordination under three-agent manipulation 162

6.1 Dynamic cooperative pushing under uncertainty 168
6.2 Manipulation of triangular object 177
6.3 H^∞ tracking results of positions ($\gamma = 0.1$) 180
6.4 H^∞ tracking results of velocities ($\gamma = 0.1$) 181
6.5 Generalized forces $\overline{\mathbf{u}}(t)(\gamma = 0.1)$ 182
6.6 H^∞ tracking results of positions ($\gamma = 0.05$) 183
6.7 H^∞ tracking results of velocities ($\gamma = 0.05$) 184
6.8 Generalized forces $\overline{\mathbf{u}}(t)$ ($\gamma = 0.05$) 185
6.9 Force distribution under three-agent manipulation ($\gamma = 0.1$) 186
6.10 Force distribution under three-agent manipulation ($\gamma = 0.05$) 187
6.11 H^∞ tracking results of positions(two-agent) ($\gamma = 0.05$) 188
6.12 H^∞ tracking results of velocities(two-agent) ($\gamma = 0.05$) 189
6.13 Force coordination under two-agent manipulation 190

7.1 Coordinated manipulation between agents and environment 193
7.2 Coordinated manipulation with four push primitives 194
7.3 Manipulation by dynamic tilting 195

A.1 Geometry of a polygonal object 197
A.2 Schematic of the transformation 201

List of Tables

4.1 Numerical results for objects with different geometries. The rows marked with G are the initial guess for the velocities. The rows marked with P are the planned velocities. The displacements marked with Go are the desired goal displacements, and the displacements marked with A are the achieved displacements with planned velocities 139

4.2 Convergence results under different initial guess for curved part . . . 142

4.3 Experimental results (The objects gain the initial velocities $\dot{X}_o, \dot{Y}_o, \dot{\theta}$, and the resulting displacements X_f, Y_f, θ_f, which are close to the goal displacements.) 146

Table of Acronyms

1JOC	One-Joint-over-Conveyor
COM	Center of Mass
COR	Center of Rotation
CCW	Counterclockwise
CW	Clockwise
CW-CCW	Clockwise-Counterclockwise
CWW-CW	Counterclockwise-Clockwise
DoF	Degree of Freedom
FBVP	Free Boundary Value Problem
LP	Linear Programming
PD	Proportional Derivative
PRP	Push-Rotation-Push
PRPR	Push-Rotation-Push-Rotation
RPR	Rotation-Push-Rotation
RPRP	Rotation-Push-Rotation-Push
QP	Quadratic Programming
STLC	Small Time Locally Controllable
TPBV	Two Point Boundary Value

Nomenclature

$\bar{A}$	the contact area between the object and the plane.
C_i	the initial configuration of the object
C_g	the goal configuration of the object
$COR(\mathbf{c})$	the center of rotation with the velocities $\mathbf{c}$
COR_w	the center of rotation in the world frame
$Cone(\mathbf{v})$	the conic hull generated by push direction vectors $\mathbf{v}_i$
$\mathbf{c}$	the velocity of the pushed object in the object frame
$DF(\cdot)$	the Jacobian matrix
$\mathbf{d}^{(i)}$	the solution of the linear equation
$\mathbf{d}(t)$	the uncertainties in dynamic cooperative pushing model
$\mathbf{d}^*(t)$	the worst-case disturbance
$\mathbf{F}$	the net friction
F_i^n	the normal component of the pushing force
F_i^t	the tangential component of the pushing force
F_X	the component of net friction force along X-axis
F_Y	the component of net friction force along Y-axis
$f_m(\cdot)$	the *merit function* for line search
$\mathbf{G}(\cdot)$	the vector-valued function for solving TPBV problem
H	the Hamiltonian
$H(\cdot)$	the Hessian matrix
I	the identity matrix
I_0	the mass moment of inertia
I_e	the total number of equilibrium pushes
J	the objective function for optimal control

J_c	the cost function for H^∞ controller design
J_n	the total number of non-equilibrium pushes
$\mathbf{L}_i$	the position vector of the contact point in the local frame
M_n	the minimum push distance for a non-equilibrium push
$M(\mathbf{u})$	the objective function for cooperative dynamic pushing
m	mass of the object
$m_e(k)$	the push distance of the k-th equilibrium push
$m_n(k)$	the push distance of the k-th non-equilibrium push
$\mathbf{n}_i$	the normal vector for agent i
P	the solution of algebraic Riccati-like equation
$\mathbf{p}_i$	the descent direction at step i
Q	a symmetric weighting matrix
$\mathbf{q}_1$	the configuration of an object
$\mathbf{q}_0$	the initial configuration of an object
$\mathbf{q}_d$	the desired configuration of an object
$\mathbf{q}_m$	the intermit configuration of an object
$\mathbf{q}_f$	the goal configuration of an object
$\tilde{\mathbf{q}}_1$	the tracking error
$\tilde{\mathbf{q}}_2$	the tracking error variable on the derivative of the configuration
R	the rotation matrix
R_c	a weighting matrix for H^∞ controller design
r_d	the depth function of a convex object
r_w	the width function of a convex object
$\mathbf{s}$	the solution vector for the free sliding problem
T	the net friction torque
t_f	the ending time of the manipulation
t	time
$\mathbf{t}_i$	the tangential vector for agent i
$\mathbf{u}_0$	the auxiliary control input
$\mathbf{u}$	the state feedback control input

$\mathbf{u}_e$	the control input for equilibrium push
$\mathbf{u}_n$	the control input for non-equilibrium push
$\bar{\mathbf{u}}$	the generalized control input
V	Lyapunov function
$\mathbf{v}_k$	the push direction vector of the k-th equilibrium push
$\mathbf{W}_i$	the wrench matrix for agent i
$\mathbf{X}_i$	the vector field
(X, Y)	the position of the agent frame XOY in the world frame
(x, y)	the position of the object in the plane
(x_r, y_r)	the the reachable position by the sequence of equilibrium pushes
(x_{a2}, y_{a2})	the spatial relationship parameters
$(x_r(0), y_r(0))$	the origin of the conic hull spanned by push direction vectors $\mathbf{v}_k$
$\bar{\mathbf{x}}$	the state variable of the free sliding system
α	the step length in line search
γ	the disturbance attenuation level factor
$(\Delta x_n, \Delta y_n)$	the object position change caused by the sequence of non-equilibrium pushes
$(\Delta x(k), \Delta y(k))$	the position change caused by the k-th non-equilibrium push
$\Delta\theta$	the required reorientation angle for an object
ΔF	numerical approximation of Jacobian matrix $DF(.)$
$\Delta\sigma_j$	a small number used for computing Jacobian matrix
δF_X	the uncertainty associated with the X-axis component of the friction force
δF_Y	the uncertainty associated with the Y-axis component of the friction force
δT	the uncertainty associated with the friction torque
θ	the orientation of the object in the plane
θ_+	the maximum CCW reorientation angle

	for a non-equilibrium push
θ_-	the maximum CW reorientation angle
	for a non-equilibrium push
θ_a	the orientation of the agent frame
$\theta_n(k)$	the reorientation angle for the k-th non-equilibrium push
$\lambda(t)$	the Lagrange multiplier
μ	the friction coefficient
ρ	the mass distribution function
τ	a independent variable
$\psi(.)$	the end point constraint
ω	the angular velocity of the object
$\nabla f_m(\cdot)$	the gradient of the merit function

Chapter 1

Introduction

One of the basic tasks for a robot is to move an object from an initial configuration (position and orientation) to a goal configuration; this is referred to as a parts transfer task. A common solution is to grasp the object rigidly with a gripper and move it, but his method does not work when the object is too heavy or too large to grasp. As an alternative, robotic manipulators can manipulate this class of objects using a nonprehensile manipulation technique: manipulation without grasping. Some examples of nonprehensile manipulation primitives are pushing, throwing, batting and striking. Compared with prehensile manipulation, two of the major advantages of nonprehensile manipulation are as follows: combinations of simple mechanisms, without using an advanced gripper, can accomplish a complex manipulation task, and the work space of a robotic manipulator can be enlarged. In nonprehensile manipulation, the object is not grasped by the robotic manipulator; thus, the manipulated object can exhibit a broad class of motion. The motion of the object will not only depend on the motion of the manipulator, but also depend on the state of the object, the contacts between the object, the manipulator, and the environment. Because of these complexities, an understanding of the mechanics and geometry is critical for performing a

successful nonprehensile manipulation task. A nonprehensile manipulation task can be performed either by a set of distributed physical manipulators, or by a single agent equipped with a set of simple manipulation primitives. In either case, coordination and planning is required.

1.1 Overview

This thesis focuses on the manipulation problem for the parts transfer task. The problem considered in this thesis are planar. The study of this problem is motivated by the development of assembly systems for automatic manufacturing applications. In order to assemble parts correctly, an automatic assembly device requires the receipt of parts in a specified configuration. This assembly requirements lead to the study of the parts transfer and reorienting problem. In this thesis, we focus on developing novel coordinated nonprehensile manipulation methods for the parts transfer task. With this objective, we explore the task mechanics, and study the coordination between multiple manipulation agents or multiple manipulation primitives. The understanding of task mechanics and coordination helps to increase the manipulation capabilities of a given robot or multiple robots, and it provides the following potential benefits: (1) Simpler robots and controller design; (2) Enlarged class of manipulable parts; (3) Improved planning algorithms; (4) Increased workspace.

To demonstrate these ideas and advantages, three nonprehensile manipulation methods for the parts transfer task are studied in this thesis.

1. **Parts transfer task by an ability-limited mobile robot**

   Using an ability-limited robot for the parts transfer task is motivated by the idea of minimalism in robotic manipulation introduced by Canny and Goldberg

[24]. The spirit of minimalism is to explore the capabilities of robots with simple actuators and sensors, which will result in a cheaper, more flexible system. An ability-limited mobile robot is illustrated in the following scenario: Consider a simple two-wheel mobile robot where each wheel is driven by an actuator. One actuator is always supplied with a constant voltage driving the associated wheel at constant speed, and the other actuator is controlled by supplying a full or half voltage; thus, the wheel will run at a full speed and half speed. As a result, the mobile robot works in two modes: In one mode, the two wheels run at the same speed, and the mobile robot moves forward with a constant velocity. In the other mode, one wheel runs at full speed, while the other wheel runs at half speed, and the mobile robot moves around a fixed radius circle with constant angular velocity. Compared to a continuous differential drive vehicle, the controller design is simpler. Moreover, if the robot is tele-controlled or wireless controlled, the usage of simple control signals may reduce the communication cost.

The ability-limited mobile robot provides two manipulation primitives. One is pushing the object with a constant velocity; the other one is to rotate the object around a fixed-radius circle with constant angular velocity. We are interested in using these two manipulation primitives for the parts transfer task. For this purpose, we want to answer the following two questions:

(a) *Controllability:* Under these two manipulation primitives, is the configuration of the object controllable? In other words, given two configurations in the obstacle-free plane, will there always exist a sequence of pushing actions such that the object is transferred from one configuration to the other?

(b) *Coordination:* If the configuration of the object is controllable, what is the

optimal sequence of pushing actions, and the optimal trajectory?

2. **Parts transfer by two-agent cooperative push**

   The research of using two-agent cooperative push instead of a physical fence for the parts transfer task is motivated by our desire to expand the class of manipulable objects with pushing. Physical flat fence based pushing only works for objects having at least one flat edge, such that the pushed object can align with the fence during the push as shown in Fig. 1.1. A one-point contact push is inherently unstable, and the configuration of the pushed object is unpredictable. Therefore, without having feedback, generally it can not be used to perform a stable manipulation task. However, an object may exhibit a desired motion under a two-point contact push. The line connecting the two contact points can function just as a fence, namely, *a virtual fence*, pushing on the object. With the concept of virtual fence, two-agent cooperative push becomes a closed relative to physical fence based push. By coordinating two-point contact pushing actions, objects without a flat edge can also be manipulated by sequence of pushing actions.

   The two-agent cooperative push requires two spatially distributed manipulation agents. The agents can be two mobile robots, a two-fingered adjustable griper, or an array of adjustable pins over a conveyer belt. By setting the spatial relationship between the agents, the contact positions between the agents and the object, plus the direction of push, the object will be trapped by the agents. The trapped object will continue to slide and rotate under the geometrical constraints imposed by the agents, and finally attain a known *equilibrium orientation*. In this situation, the two-agent cooperative push is equivalent to a push performed by a virtual fence that is formulated between the contact points

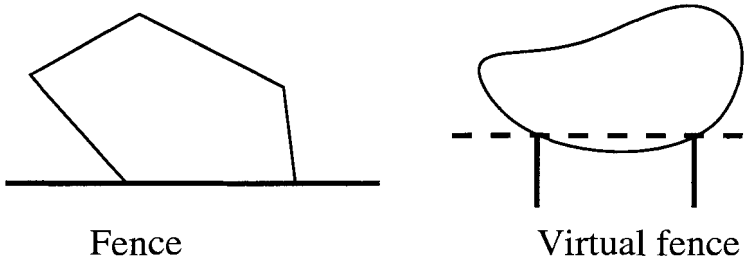


Figure 1.1: Physical fence and virtual fence pushing

(Fig. 1.1). To perform the parts transfer task, two manipulation primitives, *equilibrium push* and *non-equilibrium push*, are introduced. *Equilibrium push* only changes the position of the object, while *non-equilibrium push* changes the position and orientation of the object simultaneously. We are interested in answering the question of configuration controllability of the pushed object and finding a sequence of equilibrium and non-equilibrium pushes to transfer an object between two required configurations.

3. **Dynamic distributed manipulation**

In the ability-limited mobile robot and two-agent cooperative push based manipulation, the object maintains contact with the manipulator during the pushing process. In contrast, a more complex manipulation method is a dynamic distributed manipulation where the object can leave the manipulator and continue its motion until coming to a stop. The workspace of the manipulator is enlarged and extends beyond the physical reach of the manipulators.

A conceptual schematic of the dynamic distributed manipulation is depicted in Fig. 1.2. The manipulation system consists of two robotic manipulators, two actuators and two adjacent platforms. The first platform is fixed with two robotic

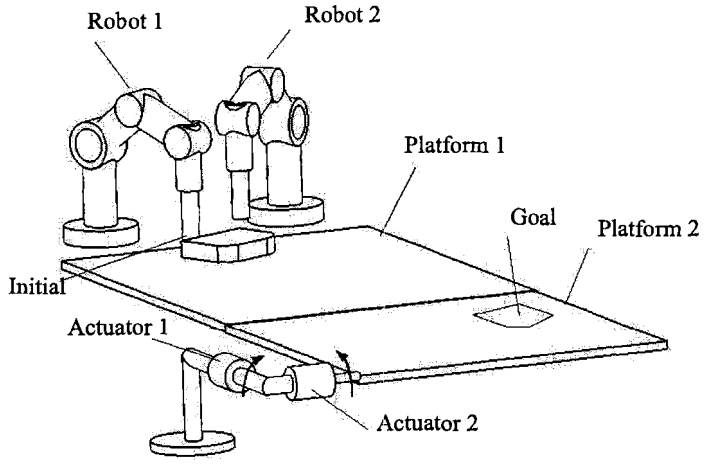


Figure 1.2: Dynamic distributed manipulation

manipulators mounted at one end, and the work space of the manipulators is limited only to the first platform. The second platform is controlled by two actuator agents, and the slope of the platform can be changed to manipulate the object sliding on it. The initial configuration is on the first platform, and the goal configuration is on the second platform. The objective of the manipulation is to transfer an object from the initial configuration to the goal configuration. The goal configuration is outside the workspace of the manipulators, and the manipulators cannot reach the goal configuration. The two manipulators, however, can push and accelerate the object to a certain release velocity. With this velocity, the object may slide to the goal configuration without the help from the manipulators. This is one characteristic of dynamic manipulation.

During the manipulation, the object undergoes three stages of motion: (1) acceleration by the manipulators; (2) free sliding on the first platform; and (3)

controlled sliding on the second platform. In the first stage, two robotic manipulators maintain contact with the object and accelerate the object to a desired position with a required release velocity. In this stage, a cooperative acceleration problem needs to be solved. Because of their workspace limitation, the robotic manipulators will lose contact with the object when the second stage begins. In the second stage, the object slides freely due to its momentum gained during the acceleration phase. The free sliding problem needs to be solved to find the desired release velocity. If the model of the sliding object is accurate, with the planned release velocity, the object will arrive at the goal configuration.

However, there always exists uncertainty in estimating friction between the object and the supporting surface. Therefore, the object may not slide to the goal configuration as expected. This discrepancy leads to the requirement of a third stage called feedback manipulation. In the third stage, two direct-driven actuators are controlled to change the slope of the second platform. By varying the slope, the trajectory of the object are controlled.

In this thesis, the control and planning problems for the first two stages are considered. For the first stage, the objective is to coordinate the agents to generate pushing forces such that the object is accelerated to the desired velocity. For the second stage, the goal of planning is to find the release velocity of a free sliding object for a given displacement.

The first two methods are developed under the quasi-static assumption. The distributed manipulation method is developed in dynamic setting. These two classes of manipulation methods have different mechanics models [67]: For quasi-static manipulation, kinetics, static forces, and quasi-static forces such as friction are considered. Inertial forces are negligible. For dynamic manipulation, kinematics, static forces,

quasi-static forces, and dynamic forces are considered.

The first two methods are open-loop method, no sensory feedback on the position and orientation of the object is required during the course of manipulation. On the other hand, in the dynamic distributed manipulation, the acceleration problem is solved as a closed-loop controller design problem where the position and velocity measurements are required.

1.2 Related work

1.2.1 Pushing

Pushing is a common form of nonprehensile manipulation primitives and has been studied by many researchers. Mason [68] first studied the mechanics of pushing and used pushing as a manipulation primitive. Accordingly, he implemented a numerical routine to find the motion of an object with a known support distribution under a single point-contact push. Following this result, Peshkin and Sanderson [82] and Goyal et al. [35, 36] studied the mechanics of a pushed object. Peshkin and Sanderson [82] studied the locus of centers of rotation of a pushed object for all possible pressure distributions. These centers of rotation provide bounds on the rate of rotation for an object being pushed. By considering the slowest rotation, the minimum push distance guaranteed to align the object with the fence can be estimated. Under a known support pressure distribution, Goyal et al. [35, 36] developed a limit surface description of the relation between the frictional forces and object motion. These results have become the foundation for developing manipulation planners for pushing operations. Using the pushing rules developed in [68], Brost [20] developed a robust grasp planner for a parallel-jaw gripper which can accommodate the object orientation uncertainty.

The push stability diagram was developed to describe the possible motions of an object being pushed by a fence. Goldberg [34] developed a planner to orient polygonal objects using the frictionless gripper and showed that it is possible to reorient any polygon in the plane from an unknown initial orientation to a unique final one by a fixed sequence of normal pushes with a straight fence. Each push is in a direction orthogonal to the face of the fence, and the reorientation of the fence between pushes is independent of the orientation of the part. Mason [69] studied the problem of pushing a block along a wall.

Many researchers have explored single-agent point-contact push. Agarwal et al. [1] explored the path planning problem for a robot pushing an object in the plane with obstacles. Nieuwenhuisen et al. [73] studied the single-robot disk pushing problem in an environment with obstacles, and exploited the boundaries of the environment to increase the possibilities of finding a push path. Okawa and Yokoyama [76] studied the control problem for a mobile robot pushing a box to a goal position. Kurisu and Yoshikawa [47] constructed feedback control strategies to push an object following a planned trajectory.

In order to deal with the instability of the point-contact pushing and generate stable open-loop pushing plans, single-agent line-contact push has been studied in [3, 65], several pushing primitives and open-loop pushing plans have been proposed to manipulate objects in the plane. For example, Akella and Mason [3] studied linear normal pushing for posing parts in an obstacle-free plane. In a linear normal push, a moving fence pushes a part in a direction normal to the fence face. The orientation of the part due to such a push is predicted using the radius function [34], and an open-loop planner is constructed as a linear programming problem that is solved by a commercial linear programming package. As an extension, Akella and Mason [5]

implemented both sensor-based and sensorless planners for the entire variational class of part shapes given a nominal part shape and tolerance bounds.

Lynch and Mason [65] introduced the concept of stable pushing to pose and orient parts in the plane. By using the kinematical constraints on object motion, the object is rigidly attached to the pusher during pushes. With a known pushing friction coefficient, center of mass, and part geometry, a procedure named STABLE was developed to find a set of pushing motions which keep the part fixed to the pusher as it moves. A pushing path planner was developed to transfer an object from an initial configuration to a goal configuration in the presence of obstacles. This planner is a close relative to the nonholonomic motion planner of [12]. Lynch [64] studied locally controllable manipulation by stable pushing, and derived a necessary and sufficient condition for a polygon to be small time locally controllable. As an extension to the previous work of [64, 65], Bernheisel and Lynch [15] studied stable pushing of a stack of parts. The idea of controllability from nonlinear control theory was introduced by Lynch and Mason [65] to study the configuration controllability of an object under a stable push.

Instead of using line contact, two-agent point-contact pushing has also been studied by many researchers. Brown and Jennings [22] introduced a pusher/steerer model to coordinate the actions of two agents, and developed a feedback strategy for the parts transfer task. Rezzoug and Gorce [86] studied two-fingered pushing with fixed contact points, proposed a trajectory tracking algorithm, and solved the optimal force distribution problem using a linear programming method. Balorda and Bajd [11] employed two-fingered pushing to reduce positioning uncertainty. Following this result, a planning method was proposed to reorient an object to a known final resting orientation by a single push [10].

Owing to the industrial needs and the desire to understand human cooperative behavior, there has been much concern on multi-agent manipulation recently. Pushing by three or more agents has also been investigated by many researchers [29, 80, 97, 104]. By analyzing the information requirements of the task, Donald et al. [29] studied the furniture pushing by a team of mobiles and presented several algorithms with varying amounts of global control, communication, and synchronization. Rus [89] developed a coordinated manipulation method for reorienting a polygonal object in a plane. Pereira et al. [80] addressed the problem of transporting a polygonal object from an initial position toward a goal position in $\mathcal{R}^2$ with multiple mobile agents. By integrating the paradigms of pushing and caging, a decentralized control algorithm was developed for this manipulation task. Sudsang et al. [97] studied the problem of manipulating of a polygonal object by three disk-shaped robots, and proposed a feedback control method for these robots. The ideas of object closure [80, 104] and inescapable configuration space [97] are closely related to the concept of caging developed in [87]. Dynamic multi-agent cooperative pushing have been studied in [50, 51, 52]. Li and Payandeh [50] studied the dynamic model for multi-agent manipulation using nonlinear control theory. The local controllability is derived for different cooperation patterns. Li and Payandeh [51] and Li and Payandeh [52] studied the planning problem for the cooperative dynamic manipulation. Feedback control schemes are designed to push the object tracking given trajectories, and optimal force distribution problems are solved using game theory. A detailed review on multi-agent distributed manipulation can be found in [56].

Particularly relevant to the work in Chapter 2 is path planning for car-like mobile robots. The main characteristic of wheeled robots is nonholonomic rolling without a slipping constraint for the wheels on the floor, which forces the vehicle to move

tangentially to its main axis. Controllability results and planners for this class of robots have been reported in [49]. The kinematic model of Dubins' car was first introduced by Dubins [30], who set the problem of characterizing the shortest paths for a particle moving forward in the plane with a constant linear velocity. Reeds and Shepp [85] considered the same problem and introduced the backwards motions. Pontryagin's Maximum Principle (PMP) has been used to derive the optimal paths for Dubins' car and Reeds-Shepp's car. Balkcom and Mason [9] studied the optimal path planning problem for a bounded velocity differential drive vehicle, and derived the conditions required for an optimal trajectory. In the work reported in Chapter 2 and also in [59], a simplified Dubins' car model with only two types of motion is used for the pushing task. Optimal control theory is employed to derive the structure of the candidate optimal trajectories.

1.2.2 Nonprehensile Manipulation

Nonprehensile manipulation refers to as manipulation without grasping. This class of manipulation has different forms including pushing, throwing, striking, batting, rolling, and juggling. Most of the work on pushing reviewed above can be viewed examples of nonprehensile manipulations. Nonprehensile manipulation has been studied for parts feeding and orienting tasks. Peshkin and Sanderson [81] used configuration maps to find sequences of fences to automatically orient a sliding part on a conveyor. It was showed that two dimensional parts can be oriented as they move on a conveyor belt against a sequence of passive fences.

Wiegley et al. [105] proposed the first complete algorithm to design such sequences for a given two dimensional convex polygonal part, and proposed a conjecture that a fence design exists to orient any 2D convex polygonal part defined by a sequence of

rational vertices. Berretty et al. [16] proved that any polygonal part can be oriented by a sequence of fences placed along a conveyor belt, thereby solving the conjecture of [105]. As a result, the authors proposed the first polynomial-time algorithm to compute the shortest sequence.

Akella et al. [6] developed a planar parts feeding system named "1JOC" (one-joint-over-conveyor). By using a series of pushes, the 1JOC can orient a polygonal part from a random initial configuration on a conveyor to a desired configuration. Akella and Mason [4] described the use of partial information sensors along with a sequence of pushing operations to eliminate uncertainty in the orientation of parts. Rusaw et al. [90] used a force/torque sensor for the parts orienting task. For certain part classes, the algorithm finds a shorter-length worst-case plan than those returned by the algorithms of [4]. Berretty et al. [17] considered sensorless orientating of an asymmetric polyhedral part by a sequence of push actions. The method is a three-dimensional generalization of conveyor belts with fences consisting of a sequence of tilted plates with curved tips. Zhang et al. [107] studied parts orienting using a sequence of fixed horizontal pins to topple the parts as they move on a conveyor belt. Zhang and Goldberg [106] Studied part alignment problem using Gripper Point Contacts, and showed that it is possible to align parts during grasping using a standard parallel-jaw gripper. Erdmann and Mason [32] studied the sensorless manipulation by tray-tilting. Aiyama et al. [2] used pivoting to manipulate objects. The manipulation was achieved by rotating the object around the contact point. Harada et al. [38] studied rolling based manipulation method, and proposed a trajectory planning algorithm. Erdmann [31] explored the nonprehensile manipulation of planar convex objects using two flat palms, and based on a static analysis, the configuration space of the object is decomposed into regions of invariant dynamics, and plans are searched

in this simplified space.

Lynch and Mason [66] explored dynamic nonprehensile manipulation, and demonstrated that low degree-of-freedom robots can perform complex tasks through a sequence of dynamic grasp, rolling, and free flight. Srinivasa et al. [93] explored the control synthesis problem for a robot dynamically manipulating an object in the presence of multiple frictional contacts, and derived a constraint on the robot joint accelerations that need to be satisfied to obtain a desired contact mode and a desired dynamic motion of the object. Srinivasa et al. [93] studied the planning and control of dynamic contact manipulation. Blind et al. [19] designed a simple device consisting of a grid of retractable pins mounted on a vertical plate to manipulate polygonal objects, and proposed a novel algorithm for part reorientation. Tabata and Aiyama [98] introduced tossing manipulation; a 1-DOF manipulator swings its arm to roll/slide an object on it, and then tosses it to a goal position. Tossing manipulation is expanded in [99] to catch the tossed object without impact. Black and Lynch [18] studied the batting manipulation. Kriegman [45] studied part dropping, and defined the capture region for curved objects and polyhedra. [108] studied the part dropping for 3D manipulation. Luntz et al. [63] explored distributed manipulation using a planar array of many small stationary elements; through cooperation, they were used to manipulate larger objects. Murphey and Burdick [72] studied feedback control methods for distributed manipulation systems that move objects via rolling and slipping point contacts. Moll et al. [71] studied the parts orienting problem for micro assemblies, and proposed a pair of manipulation primitives and a complete algorithm to uniquely orient any asymmetric part while maintaining contact without sensing.

In one particular form of planar dynamic nonprehensile manipulation, the object does not always keep in contact with the manipulator. The motion of the object

consists of two phases: an acceleration phase and a free sliding phase. In the first phase, the object keeps contact with the manipulator and it is accelerated. In the second phase, the object leaves the manipulator, and slides freely on the supporting surface. Impact or impulse manipulation [37, 41], releasing manipulation [109] and cooperative dynamic push [51, 54, 55] belong to this class. Huang and Mason [41] and Han and Park [37] studied the impulse planar manipulation problem. The problem was decomposed into impact and inverse sliding subproblems. Based on the impact model, Huang [40], and Han and Park [37] developed the control strategies to accelerate the object. Impact is a complex phenomenon which is challenging to model and control. Routh [88] and Wang and Mason [103] studied the dynamics of impact, where they considered the effects of friction in the Routh impact model. Partridge and Spong [77] applied this model to predict the trajectory of an air hockey puck after impact with the table wall, and also presented a planner to strike the puck and follow a desired trajectory. Spong [92] investigated the controllability of the air hockey puck subject to impact from a mallet, and characterized the reachable subset of velocities achievable with a single impact. This result indicated that the object can not achieve an arbitrary velocity with a single impact.

In order to generate a manipulation plan, Huang [40] proposed a multi-tap planning method to direct an object from an initial configuration to a goal configuration. Implicitly this approach also requires that the work space of the tapping device or robot can reach both the initial and final goal configurations. Zhu et al. [109] studied the releasing manipulation where objects are accelerated using one manipulator. Li and Payandeh [58] studied unconstrained dynamic manipulation for parts on a plane using a one joint robot. The object is accelerated by the swing motion of the one joint robot.

In the work reported in Chapter 5 and Chapter 6, we explore the acceleration problem by using multi-agent cooperative pushing. By modeling the motion of the object under acceleration as a nonlinear system, the acceleration problem is solved as a nonlinear control problem. Nonlinear control theory and design has been an active research topic and has attracted a lot of attention [44, 46]. Feedback linearization and the integrator backstepping design are two controller design techniques for nonlinear systems. The idea of feedback linearization is to cancel the nonlinearities using the control input. The cancelation results in a linear system. On the other hand, backstepping is a recursive procedure that interlaces the design of feedback control and the choice of Lyapunov function. It breaks the design problem of the full system into a sequence of design problems for lower order subsystems.

These two methods both work for nonlinear systems that possess a certain structure property with known nonlinearities. The concept of generalized force and backstepping design technique has been used by Toussaint et al. [100] to develop control laws for tracking of a nonlinear under-actuated surface vessel. Due to the pressure distribution uncertainty, the friction force between the object and the supporting surface is not known exactly. As a result, Feedback linearization and integrator backstepping design techniques cannot be applied directly. Alternative methods are required to deal with the uncertainty. Robust control and sliding mode control are the candidate design methods for dealing with the uncertainty in a nonlinear system [44]. H^∞ optimal control [14], a robust control design technique, is motivated by the development of zero-sum differential games. H^∞ optimal control has been widely discussed for its robustness and capability of disturbance attenuation in linear and nonlinear control systems [13]. Motivated by this factor, the H^∞ technique is used to deal with the uncertainty of pressure distribution in the dynamic acceleration problem. Another

alternative is to estimate the uncertainty and cancel it. The approach proposed by Li and Payandeh [54] belongs to this category. A neural network model is introduced to model the nonlinearity associated with the pressure distribution uncertainty. With this model, a feedback controller is designed to cancel the effect of the uncertainty.

1.2.3 Free Sliding Problem

The free sliding problem is an essential part of impact or impulse manipulation, releasing manipulation and the cooperative dynamic push based manipulation. Solving the free sliding problem is to determine the desired initial velocity for a sliding object. With this initial velocity, the object will achieve the required displacement.

Voyenli and Eriksen [102] first studied the dynamics of the sliding motion of disks and rings and investigated the properties of the motion such as stopping time of a sliding object. Owing to the complex dynamics of free sliding objects, it is impossible to determine the desired initial velocity analytically to achieve a specified displacement. Huang [40] mainly addressed the free sliding problem for the classes of axisymmetric objects. Axisymmetric objects are those which have pressure distribution as a function of the radius of the objects only, and they always slide in a straight line. The planning problem is to find the initial linear velocity along the line and the associated rotational velocity. Huang [40] developed a numerical approach to find the desired initial velocity. This method uses the monotonicity property of displacement with respect to initial velocity, and subdivides the 2-D initial velocity space in each iteration. However, the monotonicity property does not hold for non-axisymmetric objects, which limits the application of the impulse manipulation methods. Recently, the impulse manipulation has been extended to polygonal objects by [37]. Another set of qualitative dynamic characteristics of the motion were derived to relate the

initial velocities and the displacement of the polygonal object. Heuristic rules were developed to search for the desired initial velocity in the 3-dimensional initial velocity space.

In methods above, a qualitative relationship between the initial velocity and displacement were derived first, then a search algorithm was developed through bisection of the initial velocity space. Subsequently, simulation and experimental results were presented to verify their planning approaches. Zhu et al. [109] studied the characteristics of the free sliding problem for the releasing manipulation. As a result, a linear model was established to approximate the dynamics, and several learning control schemes were implemented to pose the object. Li and Payandeh [57] studied the free sliding problem for cooperative dynamic pushing, and proposed a new planning method. Instead of using only qualitative information and bisection-based search, quantitative information of the motion was used, and the optimization technique was employed to find the desired initial velocities. This approach works for objects of arbitrary shapes and with more general pressure distributions.

1.3 Thesis Contributions

This thesis focuses on nonprehensile manipulation for the parts transfer task. The problem is studied under quasi-static and dynamic settings, and three nonprehensile manipulation methods are proposed. Control theory and numerical optimization techniques are used as tools for these developments. The novel contributions and significant findings of the thesis are summarized below.

1. **Parts transfer by an ability-limited mobile robot**

- Introduce two stable push primitives, and prove the configuration controllability of the object.
- Using optimal control theory, prove that the optimal trajectories consist of at most, two switchings between these two stable push primitives.
- Implement an analytical planning method to find the optimal solution and demonstrate the parts transfer method using a mobile robot.

2. **Parts transfer via two-agent cooperative push**

   - Introduce two manipulation primitives, *equilibrium push* and *non-equilibrium push* for the parts transfer task; prove the configuration controllability of the object under the two-agent cooperative push.
   - Formulate the pushing process as a switched system, and develop a linear programming formulation to generate a manipulation plan. A fully-analytical expression is derived to solve the planning problem.
   - Implement and experimentally demonstrate the parts transfer method using a two-fingered gripper mounted on a MOTOMAN robot.

3. **Planning of the initial velocities for free sliding objects**

   - Develop a free boundary-value problem formulation for the free sliding problem. No motion characteristics is required in this formulation.
   - Transform the free boundary-value problem into a two-point boundary-value problem, and present three numerical algorithms to solve the planning problem.
   - Implement a one-joint robotic system, and experimentally demonstrate the proposed planning method.

4. **Cooperative dynamic push to accelerate the object**

   - Develop a dynamic model describing the motion of the object under a cooperative dynamic push. Two hierarchical methods are developed to solve the acceleration problem.
   - In the case of known pressure distribution, a centralized planning method is developed by integration of the backstepping design technique and quadratic programming.
   - In the case of the uncertain pressure distribution, a game theoretic approach is proposed to solve the acceleration problem. In the generalized force controller design level, a minimax controller is designed to achieve the tracking performance. In the coordination level, a cooperative game is solved to minimize the worst case interaction force between the agents and the object.

1.4 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 demonstrates that the parts transfer task can be accomplished by an ability-limited mobile robot. Chapter 3 introduces the concept of virtual fence and identifies the equilibrium and non-equilibrium push primitives. A fully analytical algorithm is then developed for the parts transfer task. Chapter 4 studies the free sliding planning problem. The free sliding problem is posed as a free boundary-value problem (FBVP). After transforming the FBVP into a two-point boundary-value (TPBV) problem, three shooting methods are developed to solve the planning problem. Chapter 5 is concerned with the cooperative acceleration problem under a known friction between the object and the

supporting surface, and presents a centralized planning method based on backstepping design technique and quadratic programming. Chapter 6 presents a game-theoretic approach to deal with the uncertainties in the acceleration problem studied in Chapter 5. Finally, conclusions and a discussion of future work is given in Chapter 7.

1.5 Publication Notes

Several of the main ideas in this thesis have been published or submitted for publication during the course of this research. The treatment of the parts transfer task by an ability-limited mobile robot appears in Chapter 2 is submitted in [59]. The concept of the virtual fence and two-agent cooperative push method in Chapter 3 were introduced in [60], and a complete description of the work is submitted in [61]. The planning method for the free sliding problem reported in Chapter 4 was given in part in [53] and [58], and in full in [57]. The material in Chapter 5 was reported in [51]. The game-theoretic planning method for the acceleration problem appears in Chapter 6 was reported in part in [55] and in full in [52].

Chapter 2

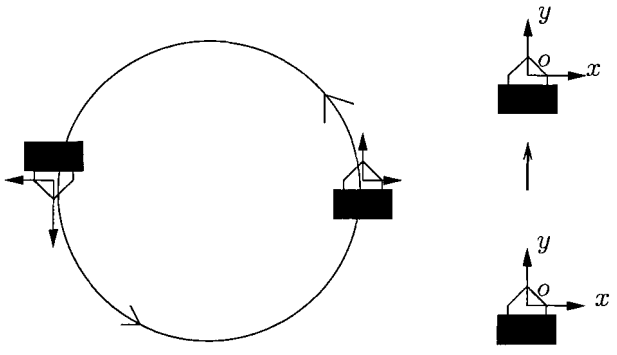
Parts Transfer by an Ability-Limited Mobile Robot Push

In this chapter, we consider an ability-limited mobile robot for the parts transfer task, and study the planning problem. The mobile robot can only perform two push actions on the object that act as the manipulation primitives. We first study the configuration controllability of the object, and the purpose of the controllability analysis is to verify whether these two manipulation primitives are sufficient for a manipulation task. With optimal control theory, we find the optimal sequence of these two push actions to transfer an object from an initial configuration to a goal configuration.

The remainder of chapter is organized as follows. Section 2.1 presents the push system, the controllability analysis, and the planning problem. Section 2.2 derives the structure for candidate optimal trajectories using optimal control theory. In Section 2.3, a computational method is given for the optimal trajectory. Simulation and experimental results are presented in Section 2.4 to demonstrate the proposed planning

method. Section 2.5 discusses the results.

2.1 The Push System and the Optimal Planning Problem



(a) Rotation (b) Linear Normal Push

Figure 2.1: Two push primitives

Considering an object pushed by a mobile robot, the configuration $\mathbf{q} = (x_o, y_o, \theta)$ describes the position and orientation of the object in the world frame. In order to predict the motion of the object, we use a stable push for the manipulation task. The concept of stable pushing was first introduced by Lynch and Mason [65]. Under a stable push, the object remains fixed to the manipulator. As a result, the object will have the same motion as the pushing robot.

The motion of the pushed object is modeled as a nonlinear system [65]

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{v}_x \\ \hat{v}_y \\ \hat{\omega} \end{bmatrix} \tag{2.1}$$

where $\mathbf{c} = (\hat{v}_x, \hat{v}_y, \hat{\omega})^T$ is the velocity of the pushed object in the object frame xoy, and it is the input to the system. Under the stable push, the velocity of the pushed object is the same as the manipulator; therefore, the motion of the object can be controlled by varying the velocity of the manipulator.

We consider the use of an ability-limited mobile robot as the manipulator, which has two modes of motion for pushing an object. In one mode, the robot pushes the object moving along a fixed-radius circle with a constant angular velocity, and we call this type of motion, R motion. In the other mode, the robot pushes the object forward with constant velocity, and we call this type of motion, P motion. These two manipulation primitives are illustrated in Fig. 2.1.

For R motion, we choose the push velocity direction as $\mathbf{c}_1 = (0, 1, 1)^T$ in the object frame. Under R motion, the object rotates in a Counterclockwise (CCW) manner. By setting the magnitude of the angular velocity, this R motion will push an object along a unit-radius circle with angular velocity $\omega = 1\text{rad/s}$. The push velocity direction of the R motion can also be represented by the center of rotation (COR) as $COR(\mathbf{c}_1) = (-1, 0)$ in the object frame. The COR can be represented in the world frame by a simple transformation. At a known configuration $\mathbf{q} = (x_o, y_o, \theta)$, the position of the COR is computed as

$$\begin{aligned} COR_w(\mathbf{c}_1) &= \begin{bmatrix} x_o \\ y_o \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} -1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} x_o - \cos\theta \\ y_o - \sin\theta \end{bmatrix} \end{aligned} \tag{2.2}$$

in the world frame.

For P motion, the pushing velocity direction is chosen in the object frame as $\mathbf{c}_2 = (0, 1, 0)^T$. This velocity direction defines a push along the y-axis in the object frame. The direction of the P motion is always tangential to the R motion circle, and

we choose the speed of the P motion as one unit/s. Here, we assume that the selected pushing velocity directions $\mathbf{c}_1$ and $\mathbf{c}_2$ are inside the region found by the procedure STABLE [65]. The procedure STABLE returns a set of pushing motions which is guaranteed to keep the part fixed to the pusher as it moves.

Therefore, these pushes belong to the stable push family. By increasing the friction coefficient between the agent and the object, this region can be enlarged to assure that the stable push always happens. Since the object will stick with the mobile robot during the stable push, push planning is equivalent to motion planning of the mobile robot.

When applying the inputs $\mathbf{c}_1$ and $\mathbf{c}_2$ to Eq. (2.1), we obtain the following two vector fields associated with R motion and P motion respectively:

$$\mathbf{X}_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 1 \end{bmatrix} \tag{2.3}$$

$$\mathbf{X}_2 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix}. \tag{2.4}$$

With these two vector fields, the motion of the object in the world frame is governed by

$$\dot{\mathbf{q}} = \begin{cases} \mathbf{X}_1 & \text{for an R motion} \\ \mathbf{X}_2 & \text{for a P motion.} \end{cases} \tag{2.5}$$

These two types of motion are used for the parts transfer task.

The manipulation process could be formulated as a switched system. The abstracted switched system consists of a continuous-time subsystem (2.5) and a rule that controls the switching between them. By introducing a switching control signal in a vector form as $u = (u_1, u_2)$, which takes values from a discrete set $U = \{(1,0),(0,1)\}$,

the switched system is written as

$$\dot{\mathbf{q}} = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 1 \end{bmatrix} u_1 + \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} u_2. \tag{2.6}$$

The object will undergo an R motion with $u = (1, 0)$, and a P motion with $u = (0, 1)$. Since the pushed object takes either R motion or P motion, we have $u_1 + u_2 = 1$. By using this fact, the system (2.6) is rewritten as

$$\dot{\mathbf{q}} = \begin{bmatrix} -\sin\theta \cdot (u_1 + u_2) \\ \cos\theta \cdot (u_1 + u_2) \\ u_1 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ u_1 \end{bmatrix}. \tag{2.7}$$

Now the system has only one control input u_1 which takes scalar values of $u_1 = 0$ or $u_1 = 1$. u_1 controls the switching between these two sub-systems. For a given switching sequence of u_1, the trajectory of the object is calculated by integrating the system equation (2.7). An example trajectory is shown in Fig. 2.2. The sequence of u_1 (type of motion) and its switching time instants entirely determine the trajectory of the object. For a given manipulation task, the role of planning is to find the sequence and the switching time instants between R and P motions.

Before we study the planning problem, we need to know whether these two manipulation primitives are sufficient for transferring an object between two configurations. In other words, the controllability of the stable pushing system (2.7) with u_1 takes values as 0 or 1. We first present some notations for a pushing trajectory. A pushing trajectory is divided into adjacent segments of P and R motions. As an example, a rotate-push-rotate (RPR) trajectory is illustrated in Fig. 2.3. The initial position of the object is located at O, and the goal position is located at C with the orientation as shown in the figure. In this case, there are three segments of R and P motions, namely,

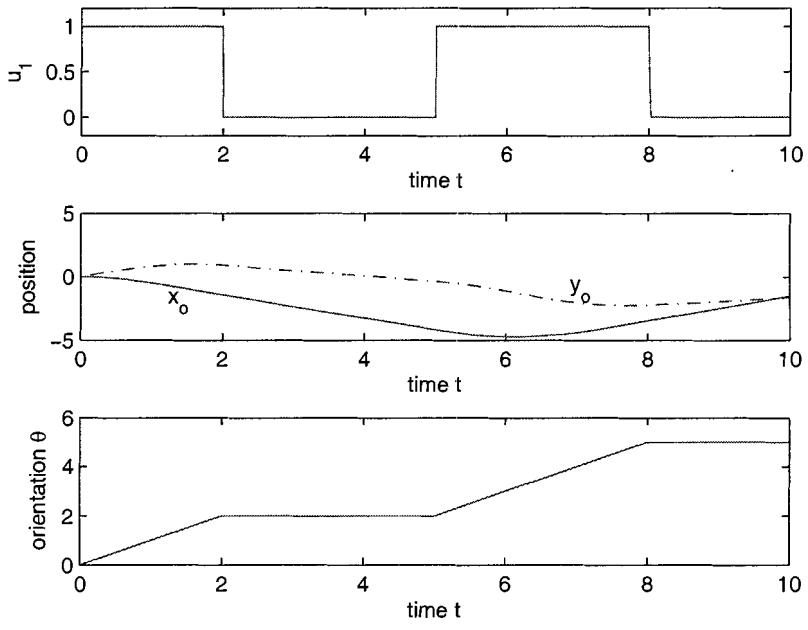


Figure 2.2: An example of a pushing trajectory with a known sequence of control signal u_1. u_1 takes $\{1, 0, 1, 0\}$, and switchings occurs at $t = 2$ sec., 5 sec., 8 sec.. The sequence is $RPRP$. The position and orientation of the object are plotted.

$R(OA), P(AB)$, and $R(BC)$. $R(OA)$ represents the R motion on the arc OA, $P(AB)$ represents the P motion on the line AB, and $R(BC)$ represents the R motion on the arc BC, The CORs of the R motions are $COR_1 = (x_{o1}, y_{o1})$ and $COR_2 = (x_{o2}, y_{o2})$ respectively.

A mobile robot equipped with R motion and P motion has the ability to transfer an object between any pair of configurations in an obstacle-free plane. In other words, the stable pushing system (2.7) with inputs $\mathbf{c}_1$ or $\mathbf{c}_2$ is controllable. The controllability result is summarized in the following theorem.

Theorem 2.1 (Controllability for the stable push system) *The control system (2.7) with input $u_1 \in \{0, 1\}$ is controllable, i.e., for any given pair of initial and goal*

configurations in an obstacle-free plane, there always exists a sequence of R and P motion to transfer the object between them.

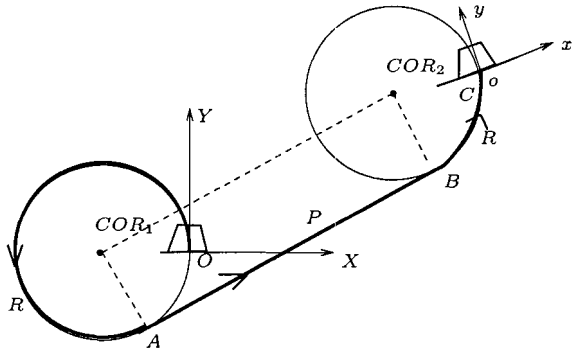


Figure 2.3: Trajectory of rotate-push-rotate(RPR).

Proof: We claim that an object can be pushed to any goal configuration $\mathbf{q}_f$ from any initial configuration $\mathbf{q}_0$ by a sequence of RPR motions. For given initial configuration and goal configurations $\mathbf{q}_0$ and $\mathbf{q}_f$, we can find two R motion circles with the COR located at COR_1 and COR_2. The circle defined by COR_1 gives all configurations that can be reached from $\mathbf{q}_0$ by an R motion, and similarly the circle defined by COR_2 gives all configurations from which $\mathbf{q}_f$ can be reached. If COR_1 and COR_2 are not identical, there will exist common lines tangential to these two circles. With the following RPR sequence, the object will reach the goal configuration $\mathbf{q}_f$. First, the object takes a R motion on the COR_1 circle; then, a P motion on the common tangential line to the circle COR_2. Finally, another R motion is required to attain the goal configuration $\mathbf{q}_f$. If the COR_1 and COR_2 are identical, a single R motion on the COR_1 circle will be sufficient for transferring the object to the goal. Therefore, with a sequence of RPR motion, the mobile robot can push an object between any two configurations in an obstacle-free plane, *i.e.*, the system (2.7) is controllable.

■

We have proven the controllability of the proposed pushing system (2.7) with inputs as $\mathbf{c}_1$ or $\mathbf{c}_2$ using the methodology proposed in [65]. In [65], a condition was proposed to select velocity directions for controllability.

The controllability result of the system (2.7) guarantees that an object can be transferred between any two configurations in an obstacle-free plane. On the other hand, the controllability means that multiple paths exist between any two configurations. For example, pushing an object between two configurations $\mathbf{q}_0$ and $\mathbf{q}_f$ can also be accomplished by a push from $\mathbf{q}_0 \rightarrow \mathbf{q}_m \rightarrow \mathbf{q}_f$. Where $\mathbf{q}_m$ is another configuration that is not on the original path $\mathbf{q}_0 \rightarrow \mathbf{q}_f$. Due to the simplicity of these push primitives, for given two configurations, it is possible to find a sequence of pushing actions and trajectory geometrically, and the RPR sequence used in the controllability proof gives such an example. However, it is difficult to verify whether this trajectory is optimal or not since multiple paths exist between any two configurations. Finding the optimal trajectory from the candidate trajectories is an interesting and challenging problem, which is the main objective of this chapter. Here, we define the optimal trajectory as the one that minimizes the time used for pushing the object to the goal configuration. Optimal control theory is used as a tool for deriving the structure of candidate optimal trajectories.

For the switched system (2.7), the optimal planning is equivalent to finding the optimal switching sequence of $u_1 \in \{0, 1\}$ and switching time instants such that the states of the system (2.7) is controlled from a given initial configuration $\mathbf{q}_0$ to a goal configuration $\mathbf{q}_f$ with minimum time. The objective function is

$$J = \int_0^{t_f} 1 dt \tag{2.8}$$

where t_f is the ending time of the push.

Next we rewrite the optimal control problem in a more general form [39]. Consider the system equation (2.7) with the initial state $\mathbf{q}(0) = \mathbf{q}_0$, and the end point constraints on the final goal configuration is defined as a vector-valued function

$$\psi[\mathbf{q}(t_f), t_f] = \mathbf{q}(t_f) - \mathbf{q}_f = 0. \tag{2.9}$$

The performance index J is defined as an integral of a function $M[\mathbf{q}(t), u_1(t), t]$

$$J = \int_0^{t_f} M[\mathbf{q}(t), u_1(t), t]dt \tag{2.10}$$

and the objective is to minimize the manipulation time, and we set $M[\mathbf{q}(t), u_1(t), t] = 1$.

Solving the optimal control problem requires finding the number, order and time instants of the switching between the P and R motions. Instead of solving the optimal control problem explicitly by a numerical method, we seek to solve the problem analytically in two steps. For this purpose, optimal control theory is first used to determine the number of switchings and the candidate optimal trajectories, then an analytical method is developed to compute the switching time instants for the optimal trajectory.

2.2 Analysis of the Optimal Trajectory

Considering the optimal control problem of system (2.7) with performance index (2.10), we introduce the Lagrange multipliers $\lambda(t)$ to the performance index as follows

$$J = \int_0^{t_f} (M(\mathbf{q}, u_1, t) + \lambda^T(t)(\mathbf{f}(\mathbf{q}, u_1, t) - \dot{\mathbf{q}}))dt \tag{2.11}$$

where $\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, u_1, t)$ is the state equation (2.7), $\lambda(t) = (\lambda_1, \lambda_2, \lambda_3)^T$ are the Lagrange multipliers, and the associated Hamiltonian is defined as

$$H = M(\mathbf{q}(t), u_1, t) + \lambda(t)^T \mathbf{f}(\mathbf{q}, u_1, t). \tag{2.12}$$

For the system (2.7), the Hamiltonian is

$$H = 1 - \lambda_1 \sin\theta + \lambda_2 \cos\theta + \lambda_3 u_1. \tag{2.13}$$

From optimal control theory [39], the Lagrange multipliers satisfy the *adjoint equation*

$$\dot{\lambda} = -\frac{\partial H}{\partial \mathbf{q}}. \tag{2.14}$$

Substituting Eq. (2.13) into the adjoint equation (2.14), we obtain

$$\begin{aligned} \dot{\lambda}_1 &= \tfrac{\partial H}{\partial x} = 0 \\ \dot{\lambda}_2 &= \tfrac{\partial H}{\partial y} = 0 \\ \dot{\lambda}_3 &= \tfrac{\partial H}{\partial \theta} = \lambda_1 \cos\theta + \lambda_2 \sin\theta. \end{aligned} \tag{2.15}$$

Integrating Eq. (2.15) and solving for λ, yields,

$$\begin{aligned} \lambda_1 &= c_1 \\ \lambda_2 &= c_2. \end{aligned} \tag{2.16}$$

By using the fact in Eq. (2.7) that $\dot{x}_o = -\sin\theta$ and $\dot{y}_o = \cos\theta$, we get

$$\lambda_3 = c_1 y_o - c_2 x_o + c_3 \tag{2.17}$$

where (x_o, y_o) are the position of the object in the world frame, and c_1, c_2, and c_3 are integration constants which need to be determined.

After substituting Eq. (2.16) and Eq. (2.17) into the Hamiltonian equation (2.12), the Hamiltonian becomes

$$H = 1 - c_1 \sin\theta + c_2 \cos\theta + (c_1 y_o - c_2 x_o + c_3) u_1. \tag{2.18}$$

At a given configuration $\mathbf{q} = (x_o, y_o, \theta)$, the value of H is uniquely determined by the constants c_1, c_2, c_3 and the control u_1.

The derivative of the Hamiltonian H is computed as

$$\begin{aligned} \dot{H} &= \frac{\partial H}{\partial t} + \frac{\partial H}{\partial u_1}\dot{u}_1 + \frac{\partial H}{\partial \lambda}\dot{\lambda} + \frac{\partial H}{\partial \mathbf{q}}\dot{\mathbf{q}} \\ &= \frac{\partial H}{\partial t} + \frac{\partial H}{\partial u_1}\dot{u}_1 + \mathbf{f}^T\dot{\lambda} - \dot{\lambda}^T\mathbf{f} \\ &= \frac{\partial H}{\partial t}. \end{aligned} \tag{2.19}$$

The Pontryagin's Maximum Principle tells us that the Hamiltonian H is constant along the optimal trajectory if the Hamiltonian is not an explicit function of time [83]. Combined with the transversality condition $H(t_f) = 0$, we have

$$H = 0 \quad \forall \qquad t \in (0, t_f) \tag{2.20}$$

on any optimal trajectories.

From Eq. (2.7), we know that the object will undergo an R motion when $u_1 = 1$. The COR for this R motion is computed according to Eq. (2.2)

$$\begin{aligned} \bar{x} &= x - \cos\theta \\ \bar{y} &= y - \sin\theta \end{aligned} \tag{2.21}$$

where (x, y, θ) is a configuration on the R motion trajectory.

By considering Eq. (2.21), the Hamiltonian (2.18) associated with the R motion has the form

$$H = 1 + c_1\bar{y} - c_2\bar{x} + c_3. \tag{2.22}$$

For a P motion, $u_1 = 0$, and the Hamiltonian (2.18) is

$$H = 1 - c_1 \sin\theta + c_2 \cos\theta \tag{2.23}$$

where θ is the orientation of the object when P motion starts.

The Hamiltonian has three constant parameters c_1, c_2, and c_3. In order to keep $H = 0$ along the optimal trajectories from $t = 0$ to t_f, the number of motion segments (the number of switchings between R motion and P motion) must be finite. The result on the maximum number of motion segments is summarized in the following theorem.

Theorem 2.2 (Maximum number of segments): *Any trajectory containing more than three segments is not optimal.*

Proof: We use the fact that the Hamiltonian $H = 0$ on any optimal trajectory, and prove the theorem by contradiction.

Assume there exists an optimal trajectory with four segments. There are two possible 4-segment sequences, PRPR or RPRP, and each case is evaluated separately. We will prove that the constants c_1, c_2, and c_3 do not exist for $H = 0$.

(1) PRPR trajectory.

Assume there is an optimal PRPR sequence described as: a P motion between the initial configuration (x_0, y_0, θ_0) and (x_1, y_1, θ_0), an R motion between (x_1, y_1, θ_0) and (x_2, y_2, θ_1), another P motion between (x_2, y_2, θ_1) and (x_3, y_3, θ_1), and another R motion between (x_3, y_3, θ_1) and the goal configuration (x_f, y_f, θ_f). None of the motion segments have zero length, otherwise, the sequence will degrade to a sequence with less than four segments.

From the analysis, we know that the Hamiltonian H is equal to zero for the optimal sequence. Depending on the type of motion, the Hamiltonian is computed according to Eq. (2.22) or Eq. (2.23). The Hamiltonian associated with the first P motion is

$$H = 1 - c_1 \sin\theta_0 + c_2 \cos\theta_0 = 0 \tag{2.24}$$

where θ_0 is the initial orientation of the object. The Hamiltonian associated with the first R motion is

$$H = 1 + c_1\bar{y}_1 - c_2\bar{x}_1 + c_3 = 0 \tag{2.25}$$

where $(\bar{x}_1, \bar{y}_1)$ is the COR

$$\begin{aligned} \bar{x}_1 &= x_1 - \cos\theta_0 \\ \bar{y}_1 &= y_1 - \sin\theta_0 \end{aligned} \tag{2.26}$$

Similarly, the Hamiltonian associated with the second P motion is

$$H = 1 - c_1 \sin\theta_1 + c_2 \cos\theta_1 = 0 \tag{2.27}$$

where θ_1 is the orientation of the object after the first R motion. The Hamiltonian associated with the second R motion is

$$H = 1 + c_1\bar{y}_2 - c_2\bar{x}_2 + c_3 = 0 \tag{2.28}$$

where $(\bar{x}_2, \bar{y}_2)$ is the COR

$$\begin{aligned} \bar{x}_2 &= x_3 - \cos\theta_f \\ \bar{y}_2 &= y_3 - \sin\theta_f. \end{aligned} \tag{2.29}$$

In order to have the proposed trajectory PRPR be optimal, there has to exist parameters c_1, c_2, and c_3 such that Eqs. (2.24)-(2.28) are satisfied. The equations are rearranged in a matrix form as

$$\begin{bmatrix} -\sin\theta_0 & \cos\theta_0 & 0 \\ \bar{y}_1 & -\bar{x}_1 & 1 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ \bar{y}_2 & -\bar{x}_2 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}. \tag{2.30}$$

By considering Eqs. (2.26) and (2.29), the matrix on the right hand side of Eq. (2.30) is written as

$$A = \begin{bmatrix} -\sin\theta_0 & \cos\theta_0 & 0 \\ y_1 - \sin\theta_0 & -x_1 + \cos\theta_0 & 1 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ y_3 - \sin\theta_1 & -x_3 + \cos\theta_1 & 1 \end{bmatrix}. \tag{2.31}$$

Equation (2.30) has a solution for c_1, c_2, and c_3 only if the row rank of A is equal or less than three. This requirement on A leads to the following relationship

$$\frac{\sin\theta_1 - \sin\theta_0}{\cos\theta_0 - \cos\theta_1} = \frac{y_1 - y_3 + \sin\theta_1 - \sin\theta_0}{x_3 - x_1 + \cos\theta_0 - \cos\theta_1}. \tag{2.32}$$

After simplification, this relationship leads to the following conditions

$$\{\theta_1 = \theta_0 + 2\pi\} \quad \text{or} \quad \{\bar{x}_1 = \bar{x}_2, \bar{y}_1 = \bar{y}_2\}. \tag{2.33}$$

The first part of this condition indicates that the rotation angle of the object in the first R motion is a multiple of 2π. It implies that the push directions for the P motions P_1 and P_2 are identical. The PRPR sequence cannot be optimal. The reason is that if we eliminate the first R motion, the object can reach the goal configuration with less time. This sequence of PRPR motion is illustrated in Fig. 2.4(a). The first R motion is redundant since with the sequence of $P_1P_2R_2$, the object will reach the goal $\mathbf{q}_f$ with less time. The second part of the condition (2.33) means that these two R motions share the same COR; this implies that the P motion does not exist between these two R motions. This class of sequence is illustrated in Fig. 2.4(b). Two R motions R_1 and R_2 are connected with each other, and the P motion P_2 has zero length. Both cases require one segment having zero length. Thus, we can conclude that c_1, c_2, and c_3 do not exist to satisfy Eq. (2.30), this contradicts the assumption that the Hamiltonian equals zero on the PRPR trajectory. Thus, any PRPR trajectory is not optimal.

The same result can be proven for the RPRP trajectory.

■

The proof indicates that only trajectories with less than four segments can be candidates of the optimal trajectory. When a sequence consists of three segments, the matrix A becomes

$$A = \begin{bmatrix} -\sin\theta_0 & \cos\theta_0 & 0 \\ \bar{y}_1 & -\bar{x}_1 & 1 \\ -\sin\theta_1 & \cos\theta_1 & 0 \end{bmatrix} \tag{2.34}$$

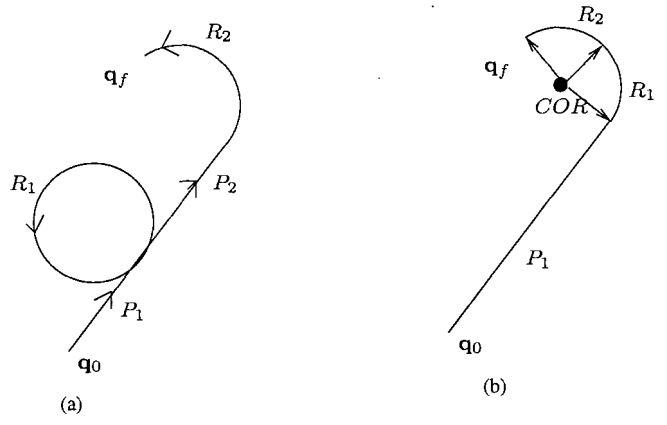


Figure 2.4: Trajectory of PRPR with $H = 0$

for PRP, or

$$A = \begin{bmatrix} \bar{y}_1 & -\bar{x}_1 & 1 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ \bar{y}_2 & -\bar{x}_2 & 1 \end{bmatrix} \tag{2.35}$$

for RPR. By replacing A with Eq. (2.34) or Eq. (2.35), Eq. (2.30) will have a solution. In other words, there will exist c_1, c_2, and c_3 to make the Hamiltonian (2.30) equal to zero. So we can conclude that trajectories with three or less segments could be candidates for the optimal trajectory. In the next section, a procedure is developed to find the optimal trajectory from three-segment candidates.

2.3 Computation of the Optimal Trajectory

We have proven that an optimal trajectory consists of at most three segments, and the three-segment optimal trajectory candidates are PRP, RPR or a sequence with less than three segments. In this section, we first explore some properties of RPR and PRP trajectories, then we propose a procedure to compute the optimal trajectory. Possible extensions to more general P and R motions are discussed in Section 2.3.4.

2.3.1 The Properties of an RPR Trajectory

Without loss of generality, we choose the initial configuration of the object as $\mathbf{q}_0 = (0, 0, 0)$, and an RPR trajectory is described as follows: when $0 \leq t \leq t_1$, the object starts with R motion, and switches to P motion at $t = t_1$; it continues as P motion until $t = t_2$, then switches to R motion at time t_2; the object attains the goal configuration at time $t = t_3$. Corresponding to Fig. 2.3, three segments of the RPR trajectory are OA, AB, and BC. By integrating Eq. (2.5), we get the following equation that describes the RPR motion:

$$\begin{bmatrix} x_o \\ y_o \\ \theta \end{bmatrix} = \begin{cases} \begin{bmatrix} \cos(t) - 1 \\ \sin(t) \\ t \end{bmatrix} & \text{for } 0 \leq t \leq t_1 \\ \begin{bmatrix} \cos(t_1) - 1 - \sin(t_1)(t - t_1) \\ \sin(t_1) + \cos(t_1)(t - t_1) \\ t_1 \end{bmatrix} & \text{for } t_1 \leq t \leq t_2 \\ \begin{bmatrix} -1 - \sin(t_1)(t_2 - t_1) + \cos(t - t_2) \\ \cos(t_1)(t_2 - t_1) + \sin(t - t_2) \\ t - t_2 + t_1 \end{bmatrix} & \text{for } t_2 \leq t \leq t_3. \end{cases} \tag{2.36}$$

Based on Eq. (2.36), we study the property of an RPR sequence. We begin by first presenting a proposition for an R motion.

Proposition 2.3 *If the rotation angle r_θ of an R motion satisfies $r_\theta \geq 2\pi$, the proposed trajectory is not optimal.*

Proof: Assume there exists a R motion on the optimal trajectory with rotation angle $r_\theta \geq 2\pi$. Considering the 2π periodicity of a circle, we can always find an alternative R motion with rotation angle $\bar{r}_\theta = r_\theta - 2\pi$, such that the object attains the same goal

configuration with less time, which contradicts the proposed R motion that $r_\theta \geq 2\pi$ is optimal.

■

Next we prove the uniqueness theorem of the RPR trajectory.

Theorem 2.4 (Uniqueness of the RPR trajectory): *For given initial and goal configurations, there exists a unique RPR trajectory satisfying the proposition 2.3.*

Proof: For the R motion starting from the initial configuration $\mathbf{q}_0 = (0, 0, 0)$, the COR is located at $COR_1 = (-1, 0)$. For the R motion reaching the goal configuration (x_f, y_f, θ_f), the COR is located at $COR_2 = (x_f - \cos(\theta_f), y_f - \sin(\theta_f))$. Associated with each R motion, a unit-radius circle exists with its center at the COR; these circles describe the R motion. These circles are called the *initial circle* and the *goal circle.* Under an R motion, the object moves along these circles in a CCW manner.

Under a P motion, the object is pushed in the direction defined by the vector field $\mathbf{X}_2$, which is tangential to the *initial circle.* From Eq. (2.36), we know that the P motion of the object is parameterized as

$$\begin{bmatrix} x_o(t) \\ y_o(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} \cos(t_1 - 1 - \sin(t_1)(t - t_1) \\ \sin(t_1) + \cos(t_1)(t - t_1) \\ t_1 \end{bmatrix} \tag{2.37}$$

where t_1 is the starting time of the P motion.

Since the angular velocity of the R motion is set as $\omega = 1$ rad/s, the P motion will start at the angle $\theta = t_1$ on the *initial circle*, and the push direction

$$\vec{V} \equiv \begin{bmatrix} -\sin(t_1) \\ \cos(t_1) \end{bmatrix} \tag{2.38}$$

defines the tangential line of the *initial circle.*

Because the orientation of the object does not change during the P motion, the departure angle from the *initial circle* and the arrival angle to the *goal circle* are identical. The push direction is tangential to the *goal circle* too. From analytical geometry, we know that for any two circles with the same radius and distinct centers, there exist two external common tangent lines. If we consider the direction of P motion as defined in Eq. (2.38), only one tangential line gives a valid P motion direction from the *initial circle* to the *goal circle* (The other tangential line is from the *goal circle* to the *initial circle*). This means that there exists only one P motion with the push direction defined as Eq. (2.38) that connects these two circles. Combining the result of Proposition 2.3, we know that the RPR trajectory is unique.

■

In some cases, an RPR trajectory may be reduced to RP or R trajectory. When the initial configuration and goal configuration are all located on the *initial circle*, an R motion is sufficient to transfer the object to the goal configuration. When the goal configuration is located on the push line of the P motion, the trajectory is an RP type.

2.3.2 The Properties of a PRP Trajectory

Since the R motion is always in a CCW manner, a PRP trajectory can not access all configurations in the configuration space, we are going to study the reachable region for a PRP trajectory.

Consider a PRP trajectory as follows: the object takes a P motion from $t = 0$ to $t = t_1$, and an R motion from $t = t_1$ to $t = t_2$; then another P motion from t_2 to $t = t_3$. Two example PRP trajectories with different goal orientations are shown in

Fig. 2.5. By integrating Eq. (2.5) for the P motion and the R motion, we get the following equation describing the PRP trajectory

$$\begin{bmatrix} x_o \\ y_o \\ \theta \end{bmatrix} = \begin{cases} \begin{bmatrix} 0 \\ t \\ 0 \end{bmatrix} & \text{for } 0 \le t \le t_1 \\ \begin{bmatrix} \cos(t-t_1)-1 \\ t_1+\sin(t-t_1) \\ t-t_1 \end{bmatrix} & \text{for } t_1 \le t \le t_2 \\ \begin{bmatrix} \cos(t_2-t_1)-1-\sin(t2-t_1)(t-t_2) \\ t_1+\sin(t2-t_1)+\cos(t_2-t1)(t-t_2) \\ t_2-t_1 \end{bmatrix} & \text{for } t_2 \le t \le t_3. \end{cases} \tag{2.39}$$

The reachable position depends on the goal orientation. In the analysis, the goal orientation space is divided into two regions as $0 \le \theta_f < \pi$ and $\pi \le \theta_f < 2\pi$. The PRP trajectory in Fig. 2.5(a) represents the case of the final orientation $\theta_f \in (0, \pi)$, and the PRP trajectory in Fig. 2.5(b) represents the case of the final orientation $\theta_f \in (\pi, 2\pi)$. Vectors $\overrightarrow{OA}$, and $\overrightarrow{BC}$ are the directions of P motions, and Vector $\overrightarrow{AB}$ is the arc for the R motion. The initial configuration of the object is $\mathbf{q}_0 = (0, 0, 0)$, and the goal configuration is $\mathbf{q}_f = (x_f, y_f, \theta_f)$. For the PRP trajectory, if the first P motion direction vector $\overrightarrow{OA}$ and the second P motion direction vector $\overrightarrow{BC}$ are not parallel to each other, there will exist an intersection point D on the Y axis. The Y-axis coordinate of D is computed as

$$y_D = y_f + \frac{\cos(\theta_f)}{\sin(\theta_f)} x_f. \tag{2.40}$$

For these two intersected lines $\overrightarrow{OA}$ and $\overrightarrow{BC}$, there will exist four unit circles, tangential to these lines which describe the R motion. Since the CCW R motions are used on the PRP trajectory, only the circles on the left hand side of the Y-axis

are used to describe the R motion on the PRP trajectory. As seen from Fig. 2.5, one circle is above the second P motion line $\overrightarrow{BC}$, and we call it the *upper* circle. The other one is below the line of $\overrightarrow{BC}$, and we call it the *lower* circle. (x_{cu}, y_{cu}) is the center of the *upper* circle, and (x_{cl}, y_{cl}) is the center of the *lower* circle. It is clear that $x_{cl} = x_{cu} = -1$. y_{cl} and y_{cu} are computed separately for the cases of $\theta_f < \pi$ and $\pi < \theta_f < 2\pi$. The results are

$$\begin{aligned} y_{cl} &= y_D - \tan(\theta_f/2) \\ y_{cu} &= y_D + \tan(\pi/2 - \theta_f/2) \end{aligned} \quad \text{for} \quad 0 < \theta_f < \pi \tag{2.41}$$

and

$$\begin{aligned} y_{cl} &= y_D - \tan(\theta_f/2 - \pi/2) \\ y_{cu} &= y_D + \tan(\pi - \theta_f/2) \end{aligned} \quad \text{for} \quad \pi < \theta_f < 2\pi. \tag{2.42}$$

The position of these circles depends on the goal configuration of the object. The Y-axis coordinate of the circles y_{cl}, y_{cu} are used to characterize the reachable configurations by a PRP sequence. The following theorem gives the conditions of a reachable configuration.

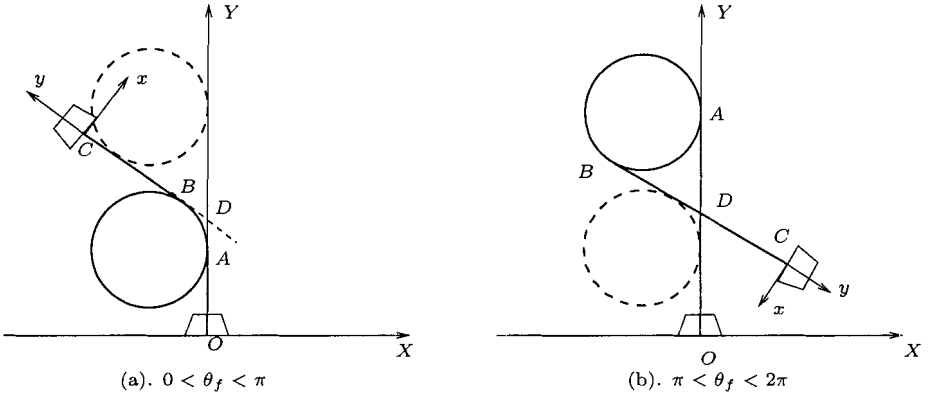


Figure 2.5: A sequence of PRP

Theorem 2.5 (Reachable region for a PRP sequence): *Starting from the initial configuration* $\mathbf{q} = (0, 0, 0)$, *there exists a unique optimal PRP trajectory that transfers the object to the goal configuration* $\mathbf{q}_f = (x_f, y_f, \theta_f)$ *if either of the following conditions are satisfied:*

a. If $\theta_f \in (0, \pi)$, *the center of the lower circle* (x_{cl}, y_{cl}) *satisfies* $y_{cl} \geq 0$ *and* $x_f \leq -1 + \cos(\theta_f)$.

b. If $\theta_f \in (\pi, 2\pi)$, *the center of upper circle* (x_{cu}, y_{cu}) *satisfies* $y_{cu} \geq 0$, *and* $x_f \geq -1 + \cos(\theta_f)$.

c. If $\theta_f = 0, x_f = 0, y_f \geq 0$.

d. If $\theta_f = \pi, x_f = -2$.

Proof:

1. For $\theta_f \in (0, \pi)$, under the condition $y_{cl} \geq 0$, we know that the center of the *lower* circle is above the X-axis, and $|OA| \geq 0$; therefore, there exists a P motion. From the condition $x_f \leq -1 + \cos(\theta_f)$, we know that $|BC| \geq 0$, therefore, there exists another P motion. Since $\theta_f > 0$, there is an R motion; therefore, there is a PRP trajectory under condition (a). The uniqueness of this PRP trajectory comes from the uniqueness of the *lower* circle.

2. For $\theta_f \in (\pi, 2\pi)$, under the condition $y_{cu} \geq 0$, the center of the *upper* circle is above the X-axis, and $|OA| \geq 0$; therefore, there exists a P motion. From $x_f \geq -1 + \cos(\theta_f)$, and $|BC| \geq 0$; therefore, there exists a P motion. Since $\theta_f > 0$, there is an R motion; therefore, there is a PRP trajectory under condition (b). The uniqueness of the PRP trajectory comes from the uniqueness of the *upper* circle.

3. For $\theta_f = 0$, the conditions in (c) gives a unique P motion with $|OA| = y_f$.

4. For $\theta_f = \pi$ and $x_f = -2$ unit, there are three different type of trajectories depending on the sign of y_f.

   In the case of $y_f > 0$, there is a PR trajectory with P motion with $|OA| = y_f$, then an R motion with the length $|AB| = \pi$ unit. There also exists an infinite number of PRP trajectories to attain the goal configuration. For the PRP trajectory, the object will first have a P motion with length $|OA| > y_f$, then an R motion with length $|AB| = \pi$ unit, and finally another P motion with length $|BC| > 0$. However, these classes of PRP trajectories need more time to attain the goal. PR motion gives the unique optimal trajectory.

   In the case of $y_f < 0$, PR gives the unique optimal trajectory. On this trajectory, the R motion has the length of $|AB| = \pi$, the length of P motion is $|BC| = -y_f$. There also exist PRP trajectories with $|OA| > 0$, this class of PRP trajectories are not optimal since it takes a longer time to attain the goal.

   In the case of $y_f = 0$, there is a unique R trajectory and it is optimal. There are also PR trajectories with $|OA| > 0$, and they are not optimal.

■

2.3.3 Computational Procedure of the Optimal Trajectory

For given initial and goal configurations, there will always exist a RPR trajectory. If one of the conditions in Theorem 2.5 is satisfied, a PRP trajectory will also exist. These two classes of trajectories will be the candidates for the optimal trajectory, and the one with the shortest travel time is the optimal trajectory. The RPR and PRP trajectories are computed as follows.

RPR trajectory

Theorem 2.4 tells us that for given initial and goal configurations, there exists only one push direction such that the object is pushed from the *initial circle* to the *goal circle*. The direction is defined by the common external tangent line between these two circles. Since the radii of these two circles are identical, the push direction line is parallel to the line connected $COR_1 = (-1, 0)$ and $COR_2 = (x_f - \cos(\theta_f), y_f - \sin(\theta_f))$. By considering this push direction, the departure angle $\bar{\alpha}$ for the P motion satisfies the following conditions

$$\sin(\bar{\alpha}) = -\frac{x_f - \cos(\theta_f) + 1}{\sqrt{(x_f - \cos(\theta_f))^2 + (y_f - \sin(\theta_f))^2}}$$
$$\cos(\bar{\alpha}) = \frac{y_f - \sin(\theta_f)}{\sqrt{(x_f - \cos(\theta_f))^2 + (y_f - \sin(\theta_f))^2}}.$$

The length for the P motion equals the distance between these two CORs. After finding the departure angle $\bar{\alpha}$, by using the fact that the angular velocity $\omega = 1$ rad/s, the switching time instants are computed as

$$\begin{aligned} t_1 &= \bar{\alpha} \\ t_2 &= t_1 + \sqrt{(x_f - \cos(\theta_f))^2 + (y_f - \sin(\theta_f))^2} \\ t_3 &= t_2 + \theta_f - \bar{\alpha} \end{aligned} \tag{2.43}$$

where t_1 is the switching time between the first R motion and the first P motion, t_2 is the switching time between the P motion and the second R motion, and t_3 is the time that the object attains the goal configuration. The corresponding RPR trajectory is found from Eq. (2.36).

PRP trajectory

For a given goal configuration $\mathbf{q}_f = (x_f, y_f, \theta_f)$, if one of the conditions in Theorem 2.5 is satisfied, there is a PRP trajectory. The switching time instants for the PRP

motion are computed differently for the cases (a)-(d) in Theorem 2.5. In all cases, t_1, t_2 and t_3 are defined as: t_1 is the switching time between the first P motion and the first R motion, t_2 is the switching time between the R motion and the second P motion, and t_3 is the time that the object attains the goal configuration.

1. Case (a)-PRP trajectory with $\theta_f \in (0, \pi)$

$$\begin{aligned} t_1 &= y_{cl} \\ t_2 &= t_1 + \theta_f \\ t_3 &= t_2 + (x_f - \cos\theta_f + 1)^2 + (y_f - t_1 - \sin\theta_f)^2. \end{aligned} \tag{2.44}$$

2. Case (b)-PRP trajectory with $\theta_f \in (\pi, 2\pi)$

$$\begin{aligned} t_1 &= y_{cu} \\ t_2 &= t_1 + \theta_f \\ t_3 &= t_2 + (x_f - \cos\theta_f + 1)^2 + (y_f - t_1 - \sin\theta_f)^2. \end{aligned} \tag{2.45}$$

3. Case (c)-P trajectory with $\theta_f = 0$

$$\begin{aligned} t_1 &= y_f \\ t_2 &= 0 \\ t_3 &= 0. \end{aligned} \tag{2.46}$$

4. Case (d)-PR/R/RP trajectory with $\theta_f = \pi$

   For PR trajectory

$$\begin{aligned} t_1 &= y_f \\ t_2 &= \pi \\ t_3 &= y_f + \pi. \end{aligned} \tag{2.47}$$

   For R trajectory

$$\begin{aligned} t_1 &= 0 \\ t_2 &= \pi \\ t_3 &= 0. \end{aligned} \tag{2.48}$$

For RP trajectory

$$\begin{aligned} t_1 &= 0 \\ t_2 &= \pi \\ t_3 &= \pi - y_f. \end{aligned} \tag{2.49}$$

The overall planning procedure is as follows: For given initial and goal configurations, we first compute the RPR and PRP trajectories. The candidate optimal RPR trajectory is computed according to Eq. (2.36), and the candidate optimal PRP trajectory is computed according to Eq. (2.39). The optimal trajectory is then selected from the candidate PRP and RPR trajectories with the minimal time t_3. Any segments may have zero length.

2.3.4 Extensions to General R and P Motions

We have studied a special class of stable push manipulation planning problems. During the manipulation process, the radius of the R motion is set as $r = 1$ unit, and the angular velocity is $\omega = 1$ rad/s. Moreover, the push speed for the P motion equals to one unit/s. The computation procedure of the optimal trajectory is developed based on these assumptions. The proposed procedure can be used to plan a stable push with more general R motion and P motion segments. As a generalization, the vector fields associated with R motion and P motion can take the following values,

$$\mathbf{X}_1 = \begin{bmatrix} -r\sin\theta \\ r\cos\theta \\ \omega \end{bmatrix} \tag{2.50}$$

$$\mathbf{X}_2 = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} v_l \tag{2.51}$$

where $r > 0$ is a constant radius for the R motion, $\omega > 0$ is the angular velocity used for the R motion, and $v_l > 0$ is the pushing velocity for the P motion.

The corresponding switched system is written as

$$\dot{\mathbf{q}} = \begin{bmatrix} -r\sin\theta \\ r\cos\theta \\ \omega \end{bmatrix} u_1 + \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} v_l(1-u_1). \tag{2.52}$$

When $u_1 = 1$, the object undergoes an R motion with a fixed radius r with the angular velocity ω; when $u_1 = 0$, the object takes a P motion with the velocity v_l.

For the system (2.52), the same procedure presented in Section 2.2 can be used to derive the structure of the optimal trajectory. It will arrive at the same conclusion as $H = 0$ on the optimal trajectory, and the candidate optimal trajectories have at most three motion segments. A similar procedure can be developed to compute the optimal trajectory. However,the resulting optimal trajectory will be different for different parameters r , ω and v_l.

2.4 Manipulation Examples and Experimental Results

In this section, we demonstrate and verify the proposed method by performing the parts transfer task on a triangular object. The task is performed by an ability-limited mobile robot, and we assume that the part does not slide with respect to the robot. The initial configuration is chosen as $\mathbf{q}_0 = (0, 0, 0)$. During the manipulation, the radius of the P motion is set to one unit, and the angular velocity is $\omega = 1$ rad/s, the pushing velocity for P motion is set to one unit/s. Optimal trajectory candidates are computed using the method proposed in the previous section.

In the first example, the goal configuration is chosen as $\mathbf{q}_f = (5, 0, \frac{3\pi}{2})$. The condition (b) of Theorem 2.5 is satisfied, and both PRP and RPR trajectories exist as the candidates for the optimal trajectory. The computed RPR and PRP optimal

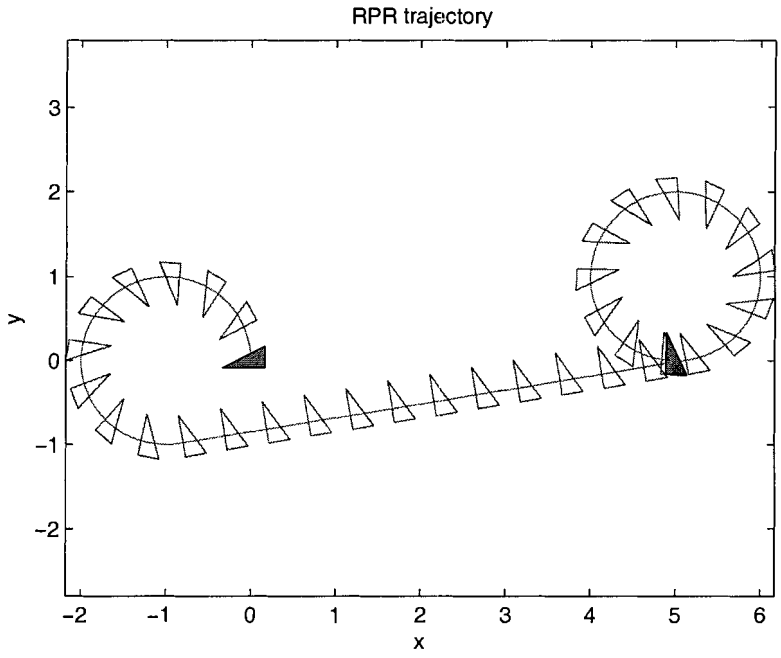


Figure 2.6: Optimal RPR trajectory. The grey triangular block located at $(0, 0)$ is the initial configuration. The grey triangular block located at $(5, 0)$ is the goal configuration. Starting from the initial configuration, the triangles illustrate the sequence of object positions during RPR pushes.

trajectories are shown in Fig. 2.6-2.7. For the RPR candidate trajectory, the total manipulation time is $t_3 = 17.05$ sec.; For the candidate PRP trajectory, the total manipulation time is $t_3 = 11.71$ sec., the length of R motion is $\pi/2$, so we know that the PRP sequence is the optimal trajectory.

In the second example, the goal configuration is located at $\mathbf{q}_f = (5, 5, \frac{\pi}{2})$. Neither condition in Theorem 2.5 holds; a PRP trajectory does not exist. There exists only an RPR trajectory as the optimal candidate. The planned optimal RPR trajectory is shown in Fig. 2.8.

For the experiment, we used a mobile robot developed in the Experimental

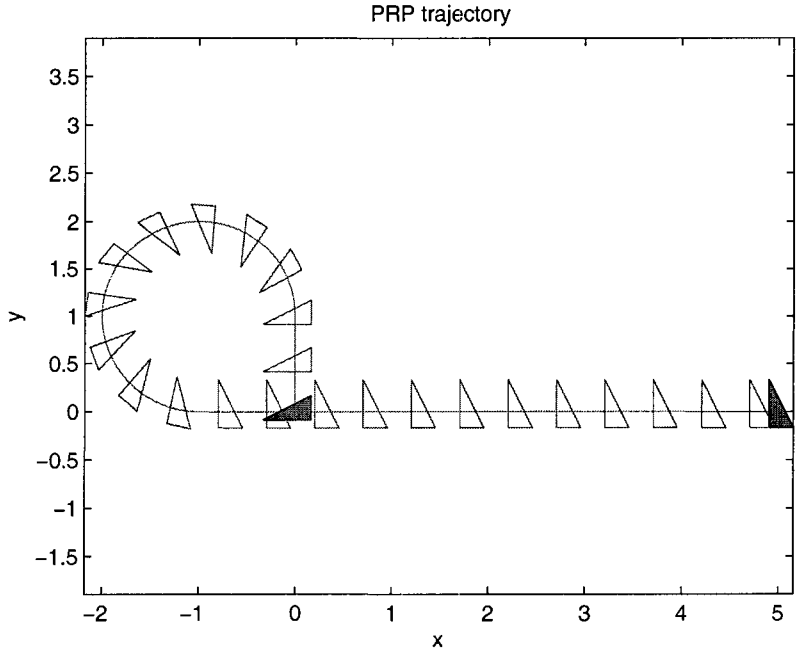


Figure 2.7: Optimal PRP trajectory. The grey triangular block located at $(0, 0)$ is the initial configuration. The grey triangular block located at $(5, 0)$ is the goal configuration. Starting from the initial configuration, the triangles illustrate the sequence of object positions during PRP pushes.

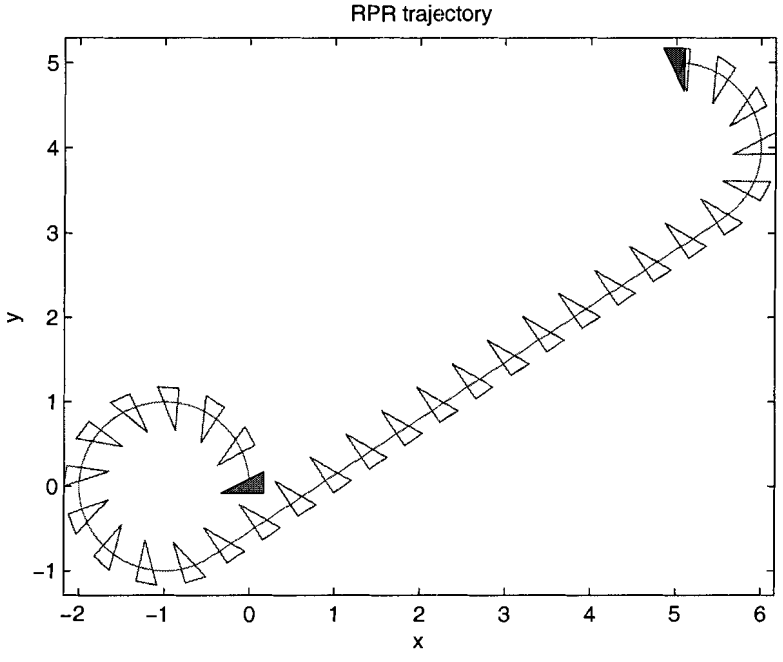


Figure 2.8: Optimal RPR trajectory. The grey triangular block located at $(0, 0)$ is the initial configuration. The grey triangular block located at $(5, 5)$ is the goal configuration. Starting from the initial configuration, the triangles illustrate the sequence of object position during RPR pushes.

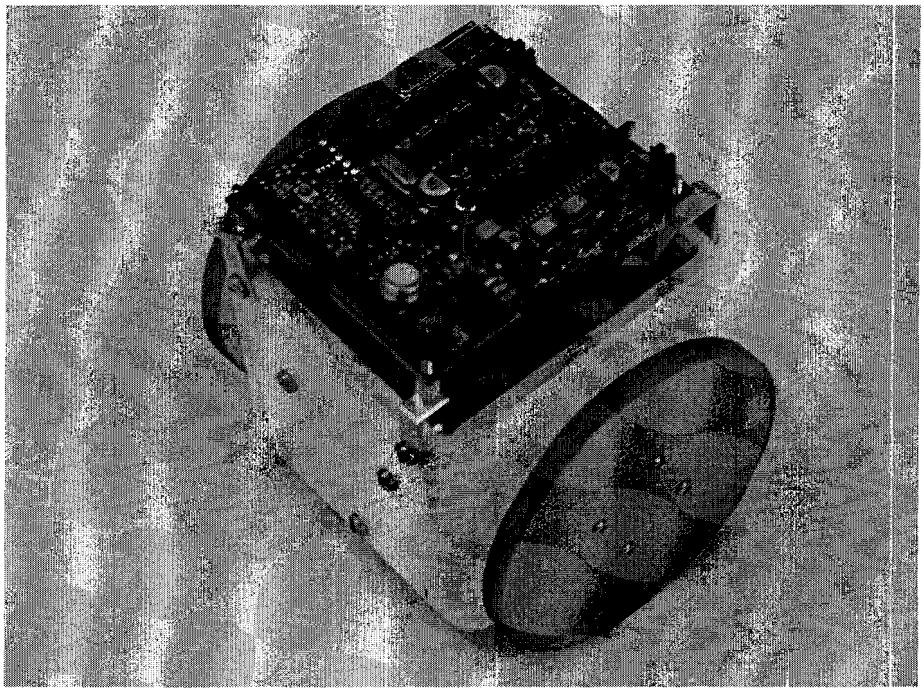


Figure 2.9: Mobile robot for experiments.

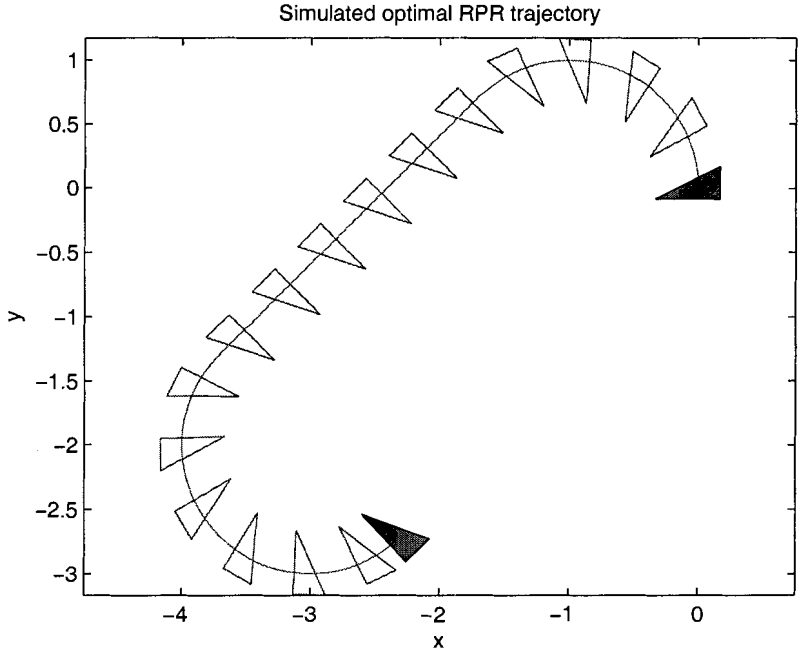


Figure 2.10: Planned optimal RPR trajectory. The grey triangular block located at $(0, 0)$ is the initial configuration. The grey triangular block located at $(-2.3, -2.7)$ is the goal configuration. Starting from the initial configuration, the triangles illustrate the sequence of object positions during RPR pushes.

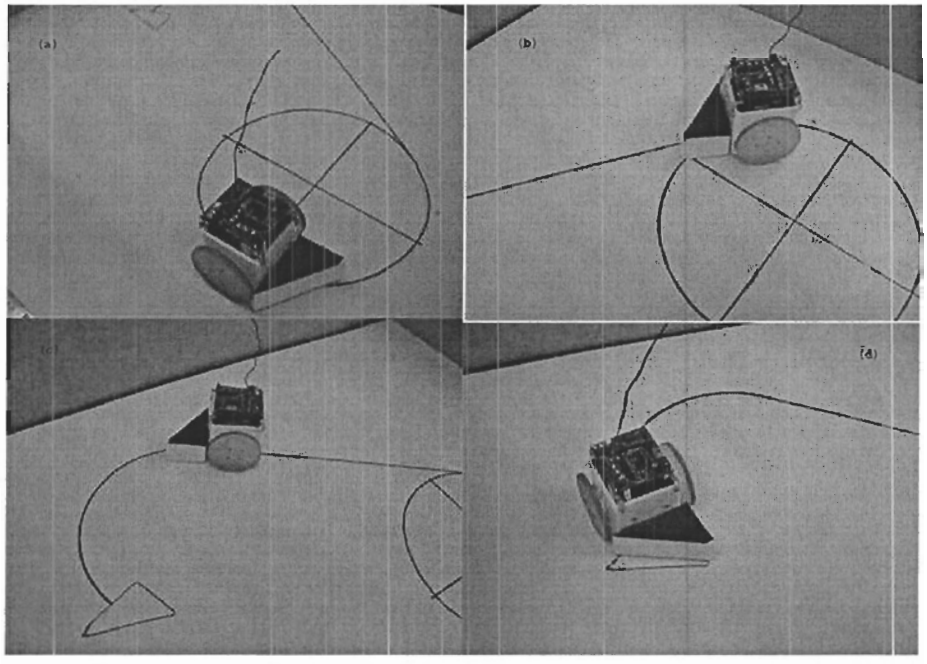


Figure 2.11: Optimal trajectory executed by a mobile robot. (a) depicts the snapshot that the robot starts the first R motion, (b) depicts the snapshot that the robot starts the P motion, (c) depicts the snapshot that the robot starts the second R motion, (d) depicts the snapshot that the robot attains the goal.

Robotics Laboratory at Simon Fraser University, and a picture of the mobile robot is shown in Fig. 2.9. This robot adapts the bi-wheel configuration for robotic hockey applications. This mobile robot uses two DC geared motors to control its two wheels. The mobile robot receives control signals from a station through a receiver. An on-board controller decodes the signals and assigns various voltage levels to the motors. For verification purposes, simple signals between the station and the mobile robot are used to turn the motors on and off. The two motors are given the same voltage for a P motion, and given different voltages for an R motion. With these two modes, the robot pushes a triangular object to track the planned optimal trajectory. In order to have a stable push, we stick a piece of sand paper on the contact edge in order to increase the friction between the object and the mobile robot.

In the experiment, the final goal configuration of the object is chosen as $\mathbf{q}_f = (-2.3, -2.7, 7\pi/4)$. By using the planning method proposed in the previous section, we discovered that there is no PRP trajectory; thus, the RPR sequence gives the optimal trajectory, and the planned optimal trajectory is shown in Fig. 2.10. By varying the speed of the motors, the mobile robot performs an R motion with a radius $r = 15$ cm. We consider $r = 15$ cm as one unit, and scaled the planned optimal trajectory by a factor of 15 as well. The optimal trajectory is painted as a black line. Snapshots of the optimal trajectory performed by the robot are shown in Fig. 2.11. The mobile robot closely tracks the optimal trajectory, and demonstrates that a mobile robot equipped with only simple actions is capable of the parts transfer task.

2.5 Discussion

In this chapter, we explored the parts transfer task by an ability-limited mobile robot. We discovered that by coordinating the simple manipulation primitives, a mobile robot

can transfer an object between any two configurations in an obstacle-free plane. The time optimal trajectory consists of less than four segments of P motion and R motion.

In this study, the parts transfer task is performed by a robot with an R and a P motion primitives. Actually, an ability-limited mobile robot equipped with two R motion primitives can also perform the parts transfer task.

As a limitation, the ability-limited mobile robot cannot push the object tracking an arbitrary trajectory. Therefore, the proposed manipulation method may fail in an environment with obstacles. In an environment with obstacles, if a push sequence exists, the optimal trajectory may consist of more than three segments. In the obstacle-free plane, the dimension and geometric shape of the pushing robot have no effect on the manipulation process since the robot can move in the plane freely. However, in the environment with obstacles, the mobile robot may collide with the obstacles.

The ability-limited assumption on the mobile robot may not be realistic. However, it provides an elegant example to illustrate the spirit of the minimalism in robotic manipulation. The result confirms that, by using the task mechanics and planning, a simple set of manipulation primitives can fulfill a manipulation task generally performed by a complex robot.

Chapter 3

Parts Transfer by Two-agent Cooperative Push

In this chapter, we introduce the concept of virtual fence for two-agent cooperative push, and we study the parts transfer problem for convex objects. This parts transfer problem has been studied by Akella and Mason [3] for polygonal objects. A linear normal push by a physical fence was used as a manipulation primitive. By using the knowledge of the mechanics for a single push action, a planner has been developed using a linear programming technique. Fence based pushing requires the object to have at least one flat edge in order to align with the fence. Moreover, a single normal push can eliminate only the orientation uncertainty, but not the position uncertainty. In order to address these inherent shortcomings of fence based manipulation, and expand the domain of push based manipulation, this chapter explores the possibility of using two-agent cooperative push with two point contacts.

By adjusting the spatial relationship between two pushing agents, the object can be pushed to an equilibrium configuration and sustain at this configuration. This is like the reorient push used in [3], where the object is aligned with the fence with a

known configuration after a push. The catching behavior has also been investigated in [21] by developing a numerical procedure to find linear pushing motions of a polygon such that the polygon attains an equilibrium configuration. By using a two-agent cooperative push, non-polygonal parts can also be manipulated. The position and orientation uncertainties can be simultaneously reduced by the push actions. The two-agent cooperative push can be performed by two mobile robots [78], a two-fingered adjustable griper, or an array of adjustable pins over a conveyer belt [75].

In this chapter, we study the following three issues:

1. Finding virtual edges for a given convex object and spatial relationship between agents.

2. Proving that the configuration of an object is controllable with a two-agent cooperative push. That is, there always exists a sequence of pushing actions to transfer the object between any two configurations.

3. Planning sequences of pushes for simultaneously positioning and orienting objects in an obstacle free plane.

The remainder of this chapter is organized as follows. Section 3.1 proposes the manipulation problem of two-agent cooperative push, and introduces equilibrium and non-equilibrium push primitives. Section 3.2 describes a method for finding the virtual edges of a convex object. Section 3.3 characterizes the net motion of the object under these two classes of pushes. Section 3.4 formulates the planning problem in the framework of a switched system and studies the controllability of the system. Section 3.5 describes a planning method for two-agent cooperative push, and provides a fully analytical solution. Section 3.6 provides some manipulation examples and experimental results. Section 3.7 gives a discussion.

3.1 Problem Definition

In this section, we introduce the method of two-agent cooperative push, and identify push primitives for the parts transfer task.

3.1.1 Two-agent Cooperative Push

In a two-agent cooperative push, the agents make point contacts with the object. A two-agent cooperative push is specified by the spatial relationship between agents, the push direction related to the object, and the push distance. During a single push, the agents are coordinated to move parallel to each other at an identical speed. The spatial relationship does not change during the pushing action. By carefully selecting the spatial relationship between agents, position and direction of push, the object will be trapped by the agents. The trapped object will continue to slide and rotate under the constraint of these two agents. Finally, the object will rest at a known *equilibrium orientation*. *Equilibrium push* and *non-equilibrium push* are used as manipulation primitives for the manipulation task. *Equilibrium push* only changes the position of the object, while *non-equilibrium push* changes position and orientation simultaneously. A manipulation example by two-agent cooperative push is shown in Fig. 3.1. In this example, the configurations are numbered in sequence; the zeroth and the third configurations are the initial and goal configurations respectively. Three pushes are used in this manipulation task. The first two configurations are achieved by non-equilibrium pushes, while the last one is achieved by an equilibrium push. For each pushing action, the positions of agents, the direction and distance of push are controlled.

The following assumptions are made for the analysis:

1. The objects are convex, polygons or objects with a general curved boundary.

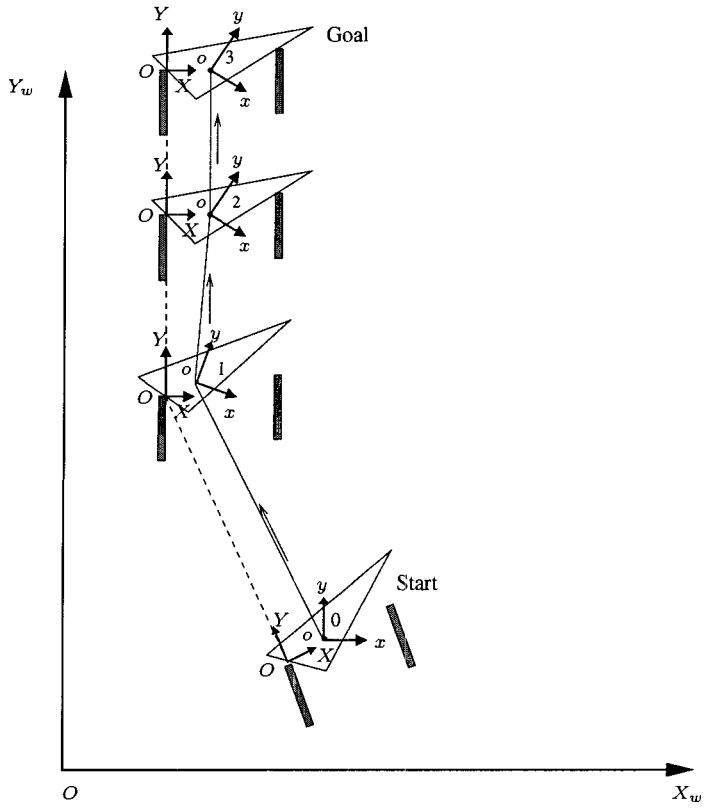


Figure 3.1: A sequence of a two-agent cooperative push to transfer a triangular object from an initial configuration to a goal configuration in the world coordinate X_wOY_w. XOY defines the agent frame, and xoy is the object frame.

2. The geometry and the center of mass (COM) of the object are known.

3. Manipulations are carried out in an obstacle-free horizontal plane. The motion is quasi-static.

4. The point contacts between the agents and the object are frictionless, and the object will slide and rotate between these two agents as far as possible, and it arrives at an orientation with the lowest COM in the agent frame, and remains at this orientation, which we call this orientation as the *equilibrium orientation.*

5. Agents are position controlled and the motion of the agents can be synchronized.

6. Supporting friction between the surface and the object is uniform, and described by Coulomb's law of friction.

Some of these assumptions have been used in [3] to analyze single fence pushing actions. Assumption (4) has been used in [10] to plan the two-fingered pushing action for reducing the orientation uncertainty.

3.1.2 Types of Pushes

The motion of agents and the contact conditions between the agents and the object determine the motion of the pushed object. When two agents both make contact with the object, these contact points, direction of push along with the COM in the agent frame will determine the motion of the pushed object. We identify the following classes of pushes.

1. *complete push:* Two agents move parallel to push the object with point contact along a specified direction for a certain distance, the object remains in contact

with these two agents. The net relative motion between the object and the agents is known before and after the push.(Fig. 3.2)

2. *Incomplete push:* The object loses contact with one agent during the push, the final configuration of the object is unknown.(Fig. 3.3)

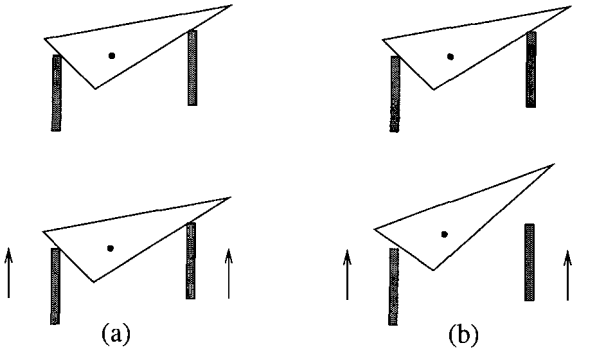


Figure 3.2: Complete pushes. (a) Equilibrium push. (b). Non-equilibrium push. The dot indicates the center of mass.

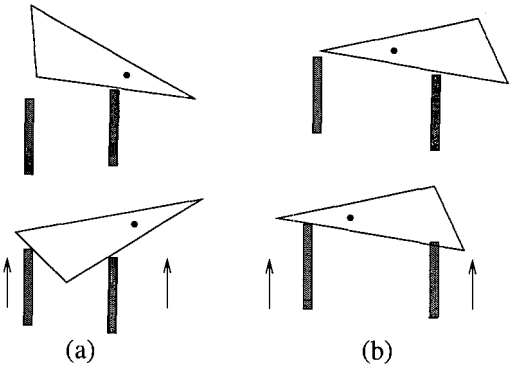


Figure 3.3: Incomplete pushes. (a) COM outside agents. (b). Pushing at the same edge. The dot indicates the center of mass.

In order to avoid uncertainty in the object's configuration after a push, we use only complete push for the parts transfer task. Based on the spatial relationship

between the agents, position and direction of a complete push, the complete pushes are classified into two categories(Fig. 3.2):

1. *equilibrium push:* If a push is performed at an equilibrium orientation, and it does not affect the orientation of the object, this push is called as an equilibrium push. In this situation, the two-agent operative push is equivalent to a push performed at an edge created by the line between two contact points. We call this edge a *virtual stable edge*, and a *virtual fence* is formulated between these two agents. The relationship between a two-agent cooperative push and a virtual fence push is shown in Fig. 3.4. The motion of an ellipsoid object pushed by agents with point contacts is the same as that of a cut ellipsoid object pushed by a fence.

2. *non-equilibrium push:* If a complete push begins at a non-equilibrium orientation, the object will rotate to an equilibrium orientation by the end of the push. This class of pushes changes the orientation and position of the object simultaneously.

Similar push primitives named *translation push* and *reorient push* have been introduced in physical fence based manipulation [3].

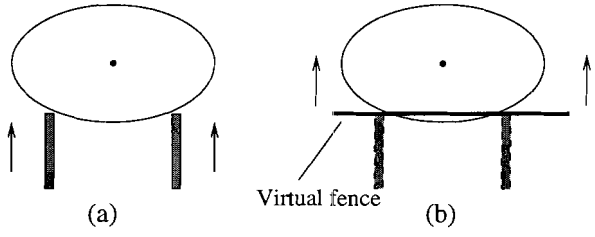


Figure 3.4: Virtual fence. (a) Two-agent cooperative push. (b). Virtual fence push.

3.2 Locating the Virtual Edges

In this section, we are going to study the formation of a virtual fence, and identify *equilibrium push* and *non-equilibrium push* for a convex object with known geometry under a two-agent cooperative push. For this purpose, we first present a parametric representation of a convex object. Then we find the equilibrium orientation and virtual stable edge of the object in the agent frame. Finally, we identify *equilibrium push* and *non-equilibrium push.*

3.2.1 Parametric Representation of a Convex Object

The boundary of a convex object in a plane will be represented by a set of parametric equations. The boundary of a 2-D object can be described by parametric equations as follows

$$\begin{aligned} x(t) &= f(t) \\ y(t) &= g(t) \end{aligned} \qquad 0 \leq t \leq 1 \tag{3.1}$$

where $(x(t), y(t))$ gives the coordinate of the boundary curve at the parameter t, $f(.), g(.)$ are functions representing the shape of the object.

As an example, let us assume the manipulated object is a convex polygon with $N+1$ vertices as $V_0, V_1, \cdots, V_N$ (Fig. 3.5). The coordinates of these vertices are given by $(x_0, y_0), (x_1, y_1), \cdots, (x_i, y_i), \cdots, (x_N, y_N)$ in the local object frame xoy attached at the COM. Each edge can be represented by a line segment between two vertices and the edges are labeled as $E_1, E_2, \cdots, E_{N+1}$, where the edge $E_i (1 \leq i \leq N)$ is parameterized as

$$y = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}) \quad x_{i-1} \leq x \leq x_i \tag{3.2}$$

and the edge E_{N+1} between the vertices V_N and V_0 is represented as

$$y = y_{N+1} + \frac{y_0 - y_{N+1}}{x_0 - x_{N+1}}(x - x_{N+1}) \quad x_{N+1} \leq x \leq x_0 \tag{3.3}$$

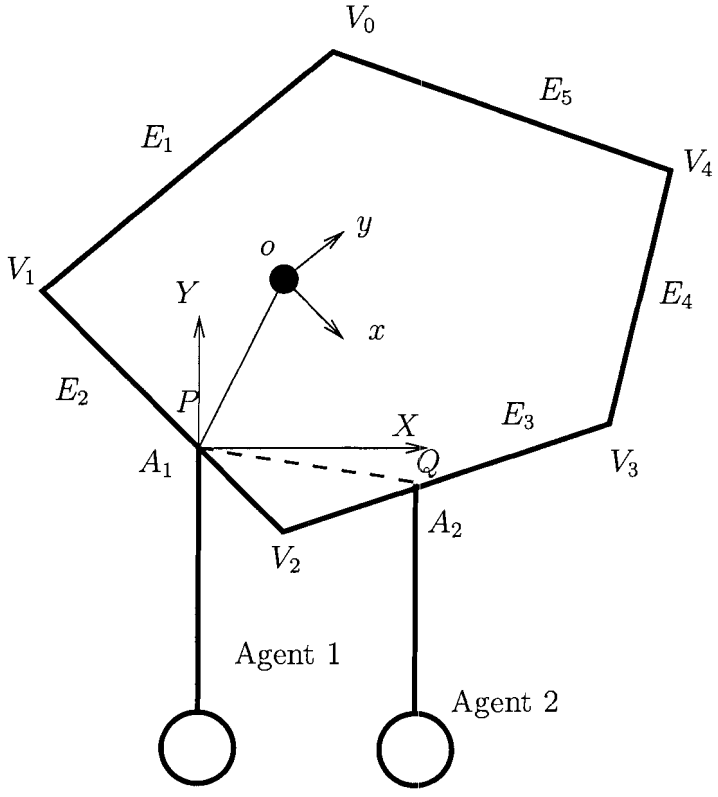


Figure 3.5: Two agents push a polygonal object

where (x, y) gives the position of a point on the edge i of the polygon.

The length of edge E_i is calculated as

$$l_i = \begin{cases} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, & i = 1, \cdots, N \\ \sqrt{(x_0 - x_{N+1})^2 + (y_0 - y_{N+1})^2}, & i = N+1 \end{cases} \tag{3.4}$$

and the perimeter of the polygon is the summation of the length for all edges as

$$L = \sum_{i=0}^{N} l_i.$$

Now we can represent the polygon in parametric form. This parametrization can be interpreted as a particle traveling along the boundary of the polygon with a period of one unit, and it travels at a constant speed $\mu = L/s$. When $t = 0$, the particle is located at vertex V_0, and it travels back to V_0 at time $t = 1$. The traveling velocity on an edge depends on the geometry of the edge, and the velocity is given by

$$v = \begin{bmatrix} \frac{L}{l_i}(x_i - x_{i-1}) \\ \frac{L}{l_i}(y_i - y_{i-1}) \end{bmatrix}. \tag{3.5}$$

The position of the particle on the boundary of the object is uniquely determined by the parameter t. Denote the total traveling time on edge E_i as T_{E_i}

$$T_{E_i} = \frac{l_i}{L} \quad \forall 1 \leq i \leq N+1.$$

Assign T_{V_i} as the time that the particle arrives at vertex V_i, and T_{V_i} is the total time required for the particle to travel from V_0 to V_i, and it is computed as

$$T_{V_i} = \sum_{k=0}^{i} T_{E_i} \quad 1 \leq i \leq N+1$$

and considering the start point, we get $T_{V_0} = 0$.

By comparing the parameter t with T_{V_i}, we can determine the edge on which the particle is located at the instant of parameter t. For example, if $t \in [T_{V_{i-1}} \quad T_{V_i}]$ for

a given i, the particle is on edge E_m. The polygon can be represented by piecewise parametric equations as,

$$\begin{cases} x(t) = x_{i-1} + \frac{1}{T_{E_i}}(x_i - x_{i-1})(t - T_{V_{i-1}}) \\ y(t) = y_{i-1} + \frac{1}{T_{E_i}}(y_i - y_{i-1})(t - T_{V_{i-1}}) \end{cases} \tag{3.6}$$

where $(x(t), y(t))$ gives the position of a point on edge E_i at parameter t, and (x_{i-1}, y_{i-1}) is the position of vertex V_{i-1} in the object frame xoy.

Convex objects with a more general boundary can also be represented by the parametric equations.

3.2.2 Finding the Equilibrium Orientations

In this section, we are going to use the parametric formulation to study the contact condition between the agents and the object. We first find the equilibrium orientations of the object in the agent frame. Then we identify the equilibrium and non-equilibrium pushes. As shown in Fig. 3.5, two agents push the object with their tips A_1, A_2. Assign the agent frame XOY at the tip A_1. Two agents move along the positive direction of the Y-axis of the agent frame with an identical speed. Since the spatial relationship between these agents does not change during one pushing action, the position (x_{a2}, y_{a2}) of A_2 is sufficient to specify the geometrical relationship of these two agents. A_2 is located at the right hand side of A_1 which requires $x_{a2} > 0$. Assume the tip A_1 makes contact with the object at point P represented by (x_p, y_p) in the object frame xoy, A_2 makes contact at Q with coordinate (x_q, y_q). These contact points can be represented in parametric form as stated in Eq. (3.6) with parameters t_p and t_q. If we know the position of contact point P, the position of contact point Q can be uniquely determined by using the spatial relationship and the parametric equation (3.6). The distance for these two contact points $(x_p(t_p), y_p(t_p))$ and $(x_q(t_q), y_q(t_q))$

can be computed as

$$d(t_p, t_q) = \sqrt{(x_q(t_q) - x_p(t_p))^2 + (y_q(t_q) - y_p(t_p))^2}. \tag{3.7}$$

If we know one contact point P, the coordinates of the other contact point must satisfy the following equation

$$d(t_p, t_q) = \sqrt{x_{a2}^2 + y_{a2}^2}. \tag{3.8}$$

By convexity of the boundary, we know that $d(t_p, t_q)$ is a monotonic increasing function in the region close to t_p. Therefore, a one-dimensional search algorithm can be used to determine the parameter t_q such that Eq. (3.8) is satisfied. Substituting t_q into Eq. (3.6), we can find the contact point (x_q, y_q).

After locating the contact points P and Q, the object orientation θ_o in the agent frame XOY can be uniquely determined.

$$\theta_o = \alpha - \beta \tag{3.9}$$

where the angle β is the angle between PQ and the x-axis of the object frame xoy, and α specifies the angle between the line A_1A_2 and the X-axis of the XOY frame.

The position of the object (x_o, y_o) in the agent frame XOY is computed as

$$\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} \cos\theta_o & -\sin\theta_o \\ \sin\theta_o & \cos\theta_o \end{bmatrix} \begin{bmatrix} -x_p \\ -y_p \end{bmatrix}. \tag{3.10}$$

The configuration of the object is specified by (x_o, y_o, θ_o) in the agent frame XOY.

In parametric formulation, for a given contact point with parameter $t_p \in [0 \quad 1]$, the parameter t_q of the other contact point can be computed using Eq. (3.8). The associated configuration of the object in contact is computed by Eq. (3.10) and Eq. (3.9).

The position of COM in the frame XOY is used to determine whether the current push is a complete push or an incomplete push. For this purpose, we introduce *depth and width functions* for a given pair of contact points t_p and t_q. The *depth function* $r_d(t_p)$ is defined as the perpendicular distance from the COM to the X-axis, and $r_d(t_p) = y_o$. The *width function* $r_w(t_p)$ is defined as the perpendicular distance from the COM to the Y-axis, and $r_w(t_p) = x_o$. The width function is used to determine whether a push is stable or not. A complete push happens under the following conditions

$$\begin{array}{ll} \text{(a)} & 0 < r_w(t_p) < x_{a2} \\ \text{(b)} & \frac{dy}{dx}(t_p) \cdot \frac{dy}{dx}(t_q) < 0. \end{array} \tag{3.11}$$

The first condition stipulates that the COM lies between two pushing agents. The second condition indicates that the slopes for the two contact edges have different signs. The convexity of the object implies that the object will be trapped by the agents. This conditions rule out the cases of the incomplete pushes that the COM lies outside the range of the agent and the push on a single edge. These two cases of incomplete pushes are illustrated in Fig. 3.3. When the conditions in Eq. (3.11) are satisfied, and by using the assumption 4, we known a complete push will happen. In the case of a complete push, the depth function $r_d(t_p)$ determines whether the current orientation is an equilibrium orientation or not. If a local minimum of the depth function $r_d(t_p)$ occurs at the contact points t_{pm} with tip A_1, and t_{qm} with tip A_2, the current orientation will be an equilibrium orientation.

The line between these two contact points defines a *virtual stable edge*, and A_1A_2 forms the *virtual fence* for this equilibrium orientation. The virtual fence along with the direction of push specify the equilibrium push. The orientation of the object will not change during an equilibrium push. As a result, the position and orientation of the agent frame will uniquely determine the configuration of the object.

3.3 Characterizing the Pushes

For the virtual edge found in Section 3.2.2, the agents can only perform an equilibrium or a non-equilibrium push at certain contact positions and pushing directions. In order to perform a desired push at a chosen virtual edge, we need to determine and set up the position and orientation of the agent frame XOY respect to the object. For this purpose, we introduce the world frame $X_wO_wY_w$ as shown in Fig. 3.1. The configuration of the object is given by (x, y, θ) in the world frame, and the configuration of the agents is defined by the position and orientation of agent frame XOY, denoted as (X, Y, θ_a). In this section, we first determine the configuration of the agent frame required for performing the equilibrium and non-equilibrium pushes, then we derive separate equations that govern the motion of the object under the equilibrium push and the non-equilibrium push.

3.3.1 Performing the Equilibrium Push

For a given spatial relationship, the agents can only perform an equilibrium push at a configuration where the virtual fence is formed. For a given object configuration (x, y, θ) and a known virtual fence formation, the configuration of the agent frame XOY will be uniquely determined.

Assume that the configuration of the pushed object is $(x(k), y(k), \theta(k))$ before the k-th push, and a virtual edge is formulated between two contact points (x_{pe}, y_{pe}) and (x_{qe}, y_{qe}) on the boundary of the object. The agent frame XOY is placed at point (x_{pe}, y_{pe}) when the agents make contact at these two points. In the world frame, the position of the XOY frame for the k-th push is computed as

$$\begin{bmatrix} X(k) \\ Y(k) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} \cos\theta(k) & -\sin\theta(k) \\ \sin\theta(k) & \cos\theta(k) \end{bmatrix} \begin{bmatrix} x_{pe} \\ y_{pe} \end{bmatrix}. \tag{3.12}$$

The XOY frame's orientation $\theta_a(k)$ is computed as

$$\theta_a(k) = \theta(k) - \theta_o. \tag{3.13}$$

The direction of push $\theta_p(k)$ is along the Y-axis of the XOY frame, *i.e.*,

$$\theta_p(k) = \theta_a(k) + \frac{\pi}{2}. \tag{3.14}$$

If we position the frame XOY at position $(X(k), Y(k))$ with the orientation as $\theta_a(k)$, and push the object along the direction specified as Eq. (3.14), the push performed by the agents will be an equilibrium push.

Under the equilibrium push, the motion of the object is governed by the following equation

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \\ 0 \end{bmatrix} m_e(k) \tag{3.15}$$

where $(x(k+1), y(k+1), \theta(k+1))$ is the configuration of the object after the k-th push, and $m_e(k) \geq 0$ is the distance of the k-th equilibrium push along the direction specified as Eq. (3.14). The push distance $m_e(k) \geq 0$ is the control parameter we need to design.

From Eq. (3.15), we specify the unit push direction vector $\mathbf{v}_k$ of the k-th equilibrium push as

$$\mathbf{v}_k = \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \end{bmatrix} \tag{3.16}$$

and the push direction vector will be used in Section 3.4.2 to analyze the configuration controllability of an object under the two-agent cooperative push.

3.3.2 Performing the Non-equilibrium Push

For a given spatial relationship between the agents, a non-equilibrium push on a virtual edge is specified by the location of the first contact point, the direction and

distance of the push. In this section, we first discuss the method of estimating clockwise (CW) and counterclockwise (CCW) maximum reorientation angles for a non-equilibrium push, and the associated direction and contact point for the push action. Then, we estimate the minimum push distance for a desired reorientation angle. With these parameters, we derive the equation that describes the net motion of the object under a non-equilibrium push.

Finding the Maximum Reorientation Angles

For each virtual edge, we need to find the CW and CCW maximum reorient angles such that the virtual fence can be formed by a non-equilibrium push. During a stable non-equilibrium push, one agent first pushes the object to rotate, then the other agent makes contact with the object. Finally, the object arrives at the equilibrium orientation, and the virtual edge is generated. By using Mason's voting theorem [68], the sense of rotation of the object can be identified. If the agent on the left makes contact first, in order to catch the object, the object must rotate CW. If the agent on the right makes contact first, for it to catch the object, the object must rotate CCW. The voting theorem has been used in [10] to develop the pushing space. The pushing space is introduced to construct the bounded regions such that the contact points inside these regions guarantee the object will be caught between the fingers as the pushing operation proceeds. This result can be used to identify the maximum CW reorientation angle θ_- and CCW angle θ_+. We can pick any of angles less than the maximum reorientation angle, as well as the direction of push and location of the first contact point.

Determining the Minimum Distance of Push

In a non-equilibrium push, the agents must push the object for a minimum distance M_n in order to guarantee that the object arrives at the equilibrium orientation. The minimum distance M_n depends on the magnitude of reorient angle $\Delta\theta$ in current non-equilibrium push; denote this relationship as a function $M_n(|\Delta\theta|)$. This distance can be estimated by using the results in [82]. By using minimum power mechanics, a method has been developed to compute the minimum push distance of an object for a reorienting task. The estimate gives an upper bound of the minimum push distance which occurs in the worst case. The estimate has been used in [3] for developing the pose planner.

In the following development, we assume that the maximum reorientation angles θ_-, θ_+ and the minimum push distance M_n are known.

Net Motion of the Object Under a Non-equilibrium Push

In this section, we are going to determine the net motion of the object under a known non-equilibrium push. The non-equilibrium push is performed on the virtual edge formulated between two contact points (x_{pe}, y_{pe}) and (x_{qe}, y_{qe}) with the equilibrium orientation θ_o. $(x(k), y(k), \theta(k))$ is the configuration of the object before the push.

The reorientation angle for a non-equilibrium push can be chosen as $\theta_n(k) \in [\theta_- \quad \theta_+]$, which determines the direction of push as well as the orientation of the agent frame XOY. The XOY frame's orientation $\theta_a(k)$ are computed as

$$\theta_a(k) = \theta(k) - \theta_o + \theta_n(k). \tag{3.17}$$

Assume that (x_{p1}, y_{p1}) gives the position of the agent frame XOY in the object frame before the k-th push. In the world frame, the position of the XOY frame is

computed as

$$\begin{bmatrix} X(k) \\ Y(k) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} \cos\theta(k) & -\sin\theta(k) \\ \sin\theta(k) & \cos\theta(k) \end{bmatrix} \begin{bmatrix} x_{p1} \\ y_{p1} \end{bmatrix}. \tag{3.18}$$

If the frame XOY is placed at $(X(k), Y(k))$ with the orientation $\theta_a(k)$, the agents can perform a non-equilibrium push in the push direction specified as Eq. (3.14).

During a non-equilibrium push, the object will translate and rotate between the agents. The net change of the object position in the world frame consists of two parts. The first part is associated with the translation of the agents. The other part is generated by the rotation motion of the object during the push. In order to complete the non-equilibrium push, the agents need to push the object for a minimum distance $M_n(\theta_n(k))$. In the world frame $X_wO_wY_w$, the net position change of the object associated with this displacement $M_n(\theta_n(k))$ is computed as

$$\begin{bmatrix} \Delta x_t(k) \\ \Delta y_t(k) \end{bmatrix} = \begin{bmatrix} \cos\theta_a(k) & -\sin\theta_a(k) \\ \sin\theta_a(k) & \cos\theta_a(k) \end{bmatrix} \begin{bmatrix} 0 \\ M_n(\theta_n(k)) \end{bmatrix}. \tag{3.19}$$

In order to find the net change of object position associated with the rotation motion, we need to compute the positions of the object in the agent frame XOY before and after the push. Before the push, the orientation $\theta_o(k)$ of the object in the XOY frame is

$$\theta_o(k) = \theta_n(k) - \theta_o. \tag{3.20}$$

The position of the object $(x_o(k), y_o(k))$ in the agent frame XOY is computed according to Eq. (3.10) as

$$\begin{bmatrix} x_o(k) \\ y_o(k) \end{bmatrix} = \begin{bmatrix} \cos\theta_o(k) & -\sin\theta_o(k) \\ \sin\theta_o(k) & \cos\theta_o(k) \end{bmatrix} \begin{bmatrix} -x_{p1}(k) \\ -y_{p1}(k) \end{bmatrix}. \tag{3.21}$$

After the object being pushed along the direction specified in Eq. (3.14), the object attains the equilibrium orientation θ_o in the agent frame XOY, and the position of

the object $(x_o(k+1), y_o(k+1))$ in the agent frame XOY is computed as

$$\begin{bmatrix} x_o(k+1) \\ y_o(k+1) \end{bmatrix} = \begin{bmatrix} \cos\theta_o & -\sin\theta_o \\ \sin\theta_o & \cos\theta_o \end{bmatrix} \begin{bmatrix} -x_{pe} \\ -y_{pe} \end{bmatrix}. \tag{3.22}$$

In the agent frame XOY, the net change of the object's position associated with the rotation is

$$\begin{bmatrix} \Delta x_o(k) \\ \Delta y_o(k) \end{bmatrix} = \begin{bmatrix} x_o(k+1) - x_o(k) \\ y_o(k+1) - y_o(k) \end{bmatrix}. \tag{3.23}$$

In the world frame $X_wO_wY_w$, the net change of object position associated with the rotation is computed as

$$\begin{bmatrix} \Delta x_r(k) \\ \Delta y_r(k) \end{bmatrix} = \begin{bmatrix} \cos\theta_a(k) & -\sin\theta_a(k) \\ \sin\theta_a(k) & \cos\theta_a(k) \end{bmatrix} \begin{bmatrix} \Delta x_o(k) \\ \Delta y_o(k) \end{bmatrix}. \tag{3.24}$$

In the world frame $X_wO_wY_w$, the total position change of the object in the k-th non-equilibrium push is the summation of position changes in Eq. (3.19) and Eq. (3.24), and it is computed as

$$\begin{aligned} \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \end{bmatrix} &= \begin{bmatrix} \Delta x_t(k) \\ \Delta y_t(k) \end{bmatrix} + \begin{bmatrix} \Delta x_r(k) \\ \Delta y_r(k) \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta_a(k) & -\sin\theta_a(k) \\ \sin\theta_a(k) & \cos\theta_a(k) \end{bmatrix} \begin{bmatrix} \Delta x_o(k) \\ \Delta y_o(k) + M_n(\theta_n(k)) \end{bmatrix}. \end{aligned} \tag{3.25}$$

The motion of the object under the non-equilibrium push is governed by the equation

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \\ \theta_n(k) \end{bmatrix} \tag{3.26}$$

where $(x(k+1), y(k+1), \theta(k+1))$ is the configuration of the object after the k-th non-equilibrium push, and $\theta_n(k)$ is the control input for the non-equilibrium push we need to specify. At a known configuration $(x(k), y(k), \theta(k))$, the position change $(\Delta x(k), \Delta y(k))$ is a nonlinear function of the control input $\theta_n(k)$, and it is uniquely

determined by $\theta_n(k)$. This indicates that a non-equilibrium push is totally determined by the reorientation angle.

3.4 Formulation of Planning Problem and Controllability Analysis

In the previous section, we have identified two manipulation primitives, *equilibrium* push and *non-equilibrium* push, for transferring an object from an initial configuration to a goal configuration in an obstacle-free plane. In this section, we first model the two-agent cooperative cooperative as a switched system, and formulate the planning problem as a control problem, then we propose the controllability analysis.

3.4.1 Switched System Formulation of Two-agent Cooperative Push

A parts transfer task usually requires agents performing a sequence of equilibrium and non-equilibrium pushes. These two classes of pushes will switch from one to the other, and each individual push will become a segment of the manipulation process. The motion of the object under two-agent cooperative push can be modeled as a switched system. The motion is governed by Eq. (3.15) for an equilibrium push, and Eq. (3.26) for a non-equilibrium push. The switching structure is shown in Fig. 3.6. Because of the upper limit of maximum reorientation angle associated with each non-equilibrium push, there may exist cases where a non-equilibrium push is followed by another non-equilibrium push.

The motion of the object under two-agent cooperative push is described by a

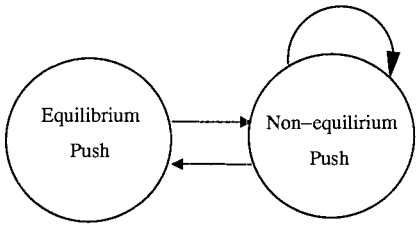


Figure 3.6: Switching structure between equilibrium and non-equilibrium pushes

discrete control system

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \mathbf{u}(k) \tag{3.27}$$

where $(x(k), y(k), \theta(k))^T$ is the current state of the system, which gives the configuration of the object before the push. $(x(k+1), y(k+1), \theta(k+1))^T$ is the state of the system after the push, $\mathbf{u}(k)$ is the control to the system, and the form of $\mathbf{u}(k)$ depends on the type of push performed at the current step. For clarity, we use $\mathbf{u}_e(k)$ as the control input for the equilibrium push, and $\mathbf{u}_n(k)$ as the control input for the non-equilibrium push.

From Eq. (3.15), we know that for an equilibrium push, $\mathbf{u}(k)$ will have the following form

$$\mathbf{u}_e(k) = \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \\ 0 \end{bmatrix} m_e(k). \tag{3.28}$$

If the current push is a non-equilibrium push, $\mathbf{u}(k)$ will take the following form

$$\mathbf{u}_n(k) = \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \\ \theta_n(k) \end{bmatrix}. \tag{3.29}$$

The position change $(\Delta x(k), \Delta y(k))$ is computed according to Eq. (3.25). $\theta_n(k)$ is the reorienting angle for the current non-equilibrium push, and the angle is less than the CW and CCW maximum reorientation angles, *i.e.*, $\theta_n(k) \in (\theta_-, \theta_+)$.

The motion of the object under two-agent cooperative push is controlled by the inputs $\mathbf{u}_e(k)$ and $\mathbf{u}_n(k)$. The planning problem for two-agent cooperative push can be studied as the following control problem:

For a given initial configuration $C_i = (x_i, y_i, \theta_i)$, the objective is to plan a sequence of equilibrium and non-equilibrium pushes (3.15) and (3.26), and design the corresponding inputs $m_e(k)$ for equilibrium pushes and $\theta_n(k)$ for non-equilibrium pushes such that the state of the system (3.27) is transferred to $C_g = (x_g, y_g, \theta_g)$, that is,

$$\sum_{i}^{I_e} \mathbf{u}_e(i) + \sum_{j}^{J_n} \mathbf{u}_n(j) = C_g - C_i \tag{3.30}$$

where I_e is the total number of equilibrium pushes, J_n is the total number of non-equilibrium pushes.

Any sequence of inputs $\mathbf{u}_e$ and $\mathbf{u}_n$ that satisfy Eq. (3.30) provides a solution to the two-agent cooperative push problem. Existence and uniqueness of solutions are questions need to be answered before the study of the planning problem. These questions lead to the controllability analysis.

3.4.2 Controllability Analysis

In this section, we study the controllability for the switched system (3.27) with the inputs (3.28) and (3.29). In the sense of manipulation, controllability means that an object can be pushed from any configuration (position and orientation) to any other configuration in an obstacle-free plane by using a sequence of *equilibrium pushes* and *non-equilibrium pushes*. Here we are going to use the theory of positive bases to study

the controllability for the switched system (3.27). The theory of positive bases was first introduced by Davis [27] to study the properties of nonnegative combination of vectors. The theory of positive bases has been used in [3] to study the completeness of the pose planner. Some definitions and properties of positive bases are outlined as follows:

1. A positive combination of a set of vectors $\{v_i \in R^n : i = 1, \cdots, r\}$ is a linear combination

$$a_1 v_1 + \cdots + a_r v_r$$

with $a_j \geq 0$; if all $a_j > 0$, it is a strictly positive combination.

2. A set of vectors $\{v_i \in R^n : i = 1, \cdots, r\}$ is positively dependent if one of them is a positive combination of the others. Otherwise, the set is positively independent.

3. A positive basis for R^n is that every vector in R^n can be written as a positive combination of the positive basis vectors and no member of the positive basis can be represented as a positive combination of the remaining members of the basis.

Equilibrium and non-equilibrium pushes require the magnitudes $m_e(k) \geq 0$ and $m_n(k) = M_n(\theta_n(k)) \geq 0$, these conditions imply that the object can only be pushed forward along the direction of push. Only positive combinations of equilibrium and non-equilibrium pushes are valid for manipulation tasks. By considering this property, the concept of positive bases is ideal for analyzing the position controllability. The controllability result on the switched control system (3.27) is summarized in the following theorem.

Theorem 3.1 (Configuration controllability for two-agent cooperative push): *The configuration of an object is controllable in an obstacle-free plane under two-agent cooperative push. That is, there always exists a sequence of equilibrium (3.15) and non-equilibrium pushes (3.26) that guarantees to transfer the object between any two configurations.*

Proof. Owing to the different motion of the object under these two classes of pushes, we study the controllability of orientation and position separately. We first consider the controllability of the orientation, then we study the controllability of the position without affecting the orientation controllability.

From Eq. (3.26), we know that the orientation of an object can be controlled CW and CCW by the control input $\theta_n(k) \in (\theta_-, \theta_+)$. Without considering the position of the object, the orientation θ can reach any angle by using a sequence of $\theta_n(k)$ as inputs to the system (3.27). This indicates that the orientation of the object is controllable. The sequence of non-equilibrium pushes will also change the position of the object, and the change associated with each non-equilibrium push can be computed according to Eq. (3.25).

Now we use the concept of positive bases to demonstrate that the object can be transferred between any two given positions by using a sequence of equilibrium pushes. The sequence of equilibrium pushes will not affect the controllability of the orientation. An equilibrium push can be performed after each non-equilibrium push along the direction specified as Eq. (3.14), and the push direction vector $\mathbf{v}_k$ is defined by Eq. (3.16). If we can find a set of vectors $\mathbf{v}_k(k = 0, \cdots, I)$ such that any vector in R^2 can be written as a positive combination of them. Then we can claim that the object can reach any position in the R^2 space by a sequence of equilibrium pushes. From Eq. (3.15), we know that the equilibrium pushing vectors $\mathbf{v}_k$ are determined by

the orientation of the XOY frame before the k-th push. The orientation of the XOY frame is changed after each non-equilibrium push according to the following equation

$$\theta_a(k+1) = \theta_a(k) + \theta_n(k). \tag{3.31}$$

Considering the 2π periodicity of the orientation, we can always find a sequence with r non-equilibrium pushes such that the orientation of the object reaches the goal orientation, and $\theta_a(1), \cdots, \theta_a(r)$ covers over angle π as shown in Fig. 3.7. In this case, the associated set of pushing vectors $\{\mathbf{v}_1, \cdots, \mathbf{v}_r\}$ will positively span the R^2 space. Therefore, any position in the R^2 space can be reached by the sequence of equilibrium pushes defined as Eq. (3.15), *i.e.*, the position of the object is controllable. Since the controllability of the position is proved under the condition that the orientation attains its goal, we conclude that the configuration of the object is controllable.

■

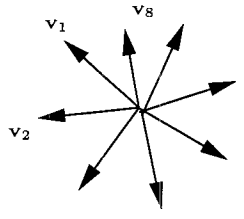


Figure 3.7: Equilibrium push vectors span R^2 space

The property of controllability implies that, for any two given configurations, there exist sequences of equilibrium and non-equilibrium pushes to transfer the object between them. Controllability guarantees the existence of a solution; On the other hand, controllability leads to the non-uniqueness of the solution. In other words, there may exist multiple sequences for a parts transfer task. For a specified performance

index, it is possible to find an optimal solution to the two-agent cooperative push planning problem.

3.5 Planning Two-agent Cooperative Push

In this section, we are going to find an optimal plan for two-agent cooperative push by solving the control problem. An optimal plan is a plan that solves the control problem (3.30) and minimizes a predefined performance index. The control problem can be solved hierarchically in two steps. We first design a controller for the orientation θ according to Eq. (3.26) without considering the position change. In this step, the objective is to find a sequence of non-equilibrium pushes such that the object attains the goal orientation θ_g. In general, the object can not achieve the goal position (x_g, y_g) by the sequence of non-equilibrium pushes. This implies that a sequence of equilibrium pushes is generally required for transferring the object to the goal position. For this purpose, we need to design a controller for equilibrium pushes (3.15) to transfer the object to the goal position (x_g, y_g). The controllability indicates that there will exist multiple push sequences for two given configurations. The purpose of control design is to find an optimal push strategy for a chosen performance criterion.

3.5.1 The Controller for Non-equilibrium Pushes

In order to reorient an object to the goal orientation, it requires a sequence of non-equilibrium pushes. To specify the sequence, we need to find the total number of non-equilibrium pushes and the reorienting angle associated with each push. For each non-equilibrium push, the agents require a period of preparation time to set up the position and push direction in order to achieve the desired reorienting angle. The total preparation time will be proportional to the number of non-equilibrium pushes.

Here, we consider the total preparation time as a criterion for determining the total number of non-equilibrium pushes, and we try to minimize the total preparation time.

The orientation θ is only controlled by $\theta_n(k)$ in Eq. (3.26). For a required orientation change $\Delta\theta = \theta_g - \theta_i$, if a CCW sequence is used to orient the part, the reorienting angle will be

$$\Delta\theta_+ = \Delta\theta. \tag{3.32}$$

If a CW sequence is used, the reorienting angle will be

$$\Delta\theta_- = 2\pi - \Delta\theta. \tag{3.33}$$

For a successful manipulation, the summation of $\theta_n(k)$ must satisfy

$$\sum_{k}^{J_n} \theta_n(k) = \begin{cases} \Delta\theta_+ & \text{for CCW sequence} \\ \Delta\theta_- & \text{for CW sequence} \end{cases} \tag{3.34}$$

where J_n is the total number of non-equilibrium pushes.

Equation (3.34) indicates that the total number of non-equilibrium pushes is determined by the individual reorientation angle $\theta_n(k)$ for each non-equilibrium push in a given pushing sequence. The reorienting angle $\theta_n(k)$ for each non-equilibrium push can be chosen as any value in the range of $[\theta_- \quad \theta_+]$. The minimum number of non-equilibrium pushes J_m is computed differently for CW or CCW sequences as follows

$$J_m = \begin{cases} \lceil \frac{\Delta\theta}{\theta_+} \rceil & \text{CCW sequence} \\ \lceil \frac{2\pi - \Delta\theta}{\theta_-} \rceil & \text{CW sequence} \end{cases} \tag{3.35}$$

where $\lceil a \rceil$ denotes the ceiling for a number a, *i.e.*, the smallest number that is greater than or equal to a.

The minimum number J_m occurs when the non-equilibrium push uses the maximum reorienting angle for the first $J_m - 1$ non-equilibrium pushes as

$$\theta_n(k) = \begin{cases} \theta_+ & \text{for CCW sequence} \\ \theta- & \text{for CW sequence} \end{cases} \tag{3.36}$$

and for the final non-equilibrium push, the reorienting angle is

$$\theta_n(J_m) = \begin{cases} \Delta\theta - (J_m - 1) \cdot \theta_+ & \text{for CCW sequence} \\ 2\pi - \Delta\theta - (J_m - 1) \cdot \theta_- & \text{for CW sequence.} \end{cases} \tag{3.37}$$

In summary, for a part transfer task with net orientation change $\Delta\theta$, the number of non-equilibrium pushes J is determined by Eq. (3.35), and the associated reorientation angles (3.36) or (3.37) are the optimal control inputs to Eq. (3.26) for minimizing the number of non-equilibrium pushes. After performing a sequence of non-equilibrium pushes, the orientation of the object will reach the goal orientation. The position of the object will also be varied by the sequence of non-equilibrium pushes. The total position change of the object is computed as the following

$$\begin{bmatrix} \Delta x_n \\ \Delta y_n \end{bmatrix} = \sum_{k=1}^{J_m} \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \end{bmatrix} \tag{3.38}$$

where $(\Delta x(k), \Delta y(k))$ is the position change of the k-th non-equilibrium push, and it is computed according to Eq. (3.25).

In the next section, after discussing the reachable region for a given push sequence of non-equilibrium pushes, a strategy for selecting the non-equilibrium push sequence will be proposed. With the selected sequence of non-equilibrium pushes, the equilibrium pushes are able to push the object to the goal.

3.5.2 The Reachable Region for Equilibrium Pushes

The Controllability Theorem 3.1 indicates that the configuration of an object is controllable under the condition that the reorienting angle of the sequence of non-equilibrium pushes is larger than π. In this case, the equilibrium push vectors span R^2 space. As illustrated in Eq. (3.34), at most one reorienting angle (CW or CCW) will be larger than π. As a result, the sequence of equilibrium push vectors with a

reorienting angle less than π will not span R^2 space. In this case, the object can only reach a subset of the R^2 space. In this section, we will find the reachable set for a selected CW or CCW sequence with reorienting angle less than π, and determine whether the given goal position (x_g, y_g) is reachable or not.

From Eq. (3.15), we know that the push direction for an equilibrium push is determined by the current orientation of the object $\theta(k)$. For a given sequence of non-equilibrium pushes (3.26), the reachable position (x_r, y_r) of equilibrium pushes is specified as follows:

$$\begin{aligned} \begin{bmatrix} x_r \\ y_r \end{bmatrix} &= \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} \Delta x_n \\ \Delta y_n \end{bmatrix} \\ &+ \sum_{k=0}^{J_m} m_e(k) \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \end{bmatrix}. \end{aligned} \tag{3.39}$$

The first term on the right hand gives the initial configuration, and the second term on the right hand side is the total position change produced by the sequence of non-equilibrium pushes, and it is computed according to Eq. (3.38). The third term describes the equilibrium pushes, where $\theta_a(k)$ is the orientation of the XOY frame before the k-th equilibrium push, $\theta_a(0) = \theta_k - \theta_o$ specifies the initial orientation of the XOY frame for the first equilibrium push, $m_e(k) \geq 0$ is the pushing distance for the k-th equilibrium push that we need to specify. For a specified k, the pushing distance $m_e(k)$ may have zero length, which implies that a non-equilibrium push is immediately followed by another non-equilibrium push.

We rewrite the push direction for the k-th equilibrium push as

$$\mathbf{v}_k = \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \end{bmatrix}. \tag{3.40}$$

Geometrically, the reachable region is spanned by vectors $\mathbf{v}_k(k = 0, \cdots, J_m)$ with

its origin $(x_r(0), y_r(0))$ located at

$$\begin{bmatrix} x_r(0) \\ y_r(0) \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} \Delta x_n \\ \Delta y_n \end{bmatrix} \tag{3.41}$$

where (x_i, y_i) is the initial position of the object, and $(x_r(0), y_r(0))$ specifies the total position change of the object caused by the sequence of non-equilibrium pushes.

By using the results of Eq. (3.40) and Eq. (3.41), the reachable region for the position (x_r, y_r) is represented as

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} x_r(0) \\ y_r(0) \end{bmatrix} + \sum_{k=0}^{J_m} m_e(k) \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \end{bmatrix} \tag{3.42}$$

where $m_e(k) \geq 0$ is the push distance of the k-th equilibrium push.

When the reorientation angle $\Delta\theta \leq \pi$, the vectors $\mathbf{v}_k(k = 0, \cdots, J_m)$ will form a *conic hull*, which is a cone

$$Cone(\mathbf{v}) = \{\sum_{k=0}^{J_m} a_k \mathbf{v}_k | a_k \geq 0, k = 0, \cdots, J_m\}. \tag{3.43}$$

In this case, there will exist sequences of equilibrium pushes to attain the goal if the goal position (x_g, y_g) is inside the conic hull. The result on reachable region for equilibrium pushes is summarized in the following theorem.

Theorem 3.2 (Reachable region for equilibrium pushes): *For a give initial configuration (x_i, y_i, θ_i), and the conic hull formed by vectors $\mathbf{v}_k(k = 0, \cdots, J_m)$ with its origin at $(x_r(0), y_r(0))$, if the position (x_g, y_g) of the configuration is inside the conic hull, then the goal can be reached by a sequence of equilibrium pushes.*

Proof. The proof follows directly from the definition of convex cone. A conic combination of $\mathbf{v}_k(k = 0, \cdots, J_m)$ is defined as

$$v_r = \sum_{k=0}^{J_m} a_k \mathbf{v}_k$$

with $a_k \geq 0$.

A conic hull is defined by the set that contains all conic combinations of points in this set. From this definition, we know that if (x_g, y_g) is a point of this set, this point can be represented as a conic combination of the vectors $\mathbf{v}_k$, which means that a sequence of equilibrium pushes exists with a_k as the pushing distance of the k-th equilibrium pushes. ∎

Now we need to check whether a given goal position (x_g, y_g) is inside the conic hull spanned by the vectors $\mathbf{v}_k (k = 0, \cdots, J_m)$. For this purpose, we introduce a new vector v_g as

$$v_g = \begin{bmatrix} x_g \\ y_g \end{bmatrix} - \begin{bmatrix} x_r(0) \\ y_r(0) \end{bmatrix}. \tag{3.44}$$

If the following conditions are satisfied, then the goal position (x_g, y_g) is inside the conic hull.

$$\begin{cases} (\mathbf{v}_0 \otimes \mathbf{v}_{J_m}) \cdot (\mathbf{v}_0 \otimes v_g) \geq 0 \\ (\mathbf{v}_0 \otimes \mathbf{v}_{J_m}) \cdot (v_g \otimes \mathbf{v}_{J_m}) \geq 0. \end{cases} \tag{3.45}$$

These two inequalities stipulate that vector v_g is inside the cone formed by vectors $\mathbf{v}_0$ and $\mathbf{v}_{J_m}$. An example reachable conic hull for a sequence of CCW equilibrium pushes is illustrated in Fig. 3.8. In this figure, the initial position of the object is given by (x_i, y_i), and the orientation of the object is specified by the xoy frame located at o. In this sequence, the number of non-equilibrium pushes is $J_m = 3$, and the maximum number of equilibrium pushes is $I_m = 4$. $|oA|, |AB|$, and $|BC|$ are the minimum distances required for the non-equilibrium pushes. The reachable conic hull DCE is formed between lines CD and CE, the origin of the conic hull is located at C. Basis vectors $\mathbf{v}_0, \cdots, \mathbf{v}_3$ are specified by the directions of equilibrium pushes. The cone is spanned by vectors $\mathbf{v}_1$ and $\mathbf{v}_3$. Every point inside this cone can be reached by using a sequence of equilibrium pushes along directions specified by the basis vectors

$\mathbf{v}_0, \cdots, \mathbf{v}_3$. For the goal position G_1, the associated position vector $\mathbf{b}$ satisfies the condition in Eq. (3.45), and it is inside the convex hull. For the goal position G_2, the vector $\mathbf{b}_2$ does not satisfy Eq. (3.45). It is outside the reachable cone.

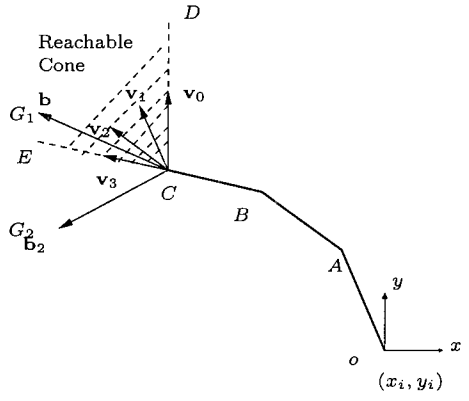


Figure 3.8: Reachable cone for equilibrium pushes. The push direction vectors are given by $\mathbf{v}_0, \cdots, \mathbf{v}_3$, and cone ECD gives the reachable goal position.

Theorem 3.1 and Theorem 3.2 indicate that with the increase of reorientation angles $\Delta\theta = \theta_g - \theta_i$, the reachable region for equilibrium pushes will be enlarged. As long as $\Delta\theta \neq \pi$, there will always exist a CCW or CW sequence that spans the R^2 space. However, when the required orientation change is $\Delta\theta = \pi$, neither the CCW nor the CW sequence covers the R^2 space. As a result, there will exist a dead zone of the goal positions that can not be reached by either CW or CCW sequences of equilibrium pushes.

One example of dead zone is shown in Fig. 3.9. The region inside the solid lines is the dead zone. If the goal position is located in this region, and the required orientation change is π, neither CW or CCW sequences of equilibrium pushes can push the object to the goal position.

In the case of the reorientation angle $\Delta\theta_+ \leq \pi$ or $\Delta\theta_- \leq \pi$, the reachable region

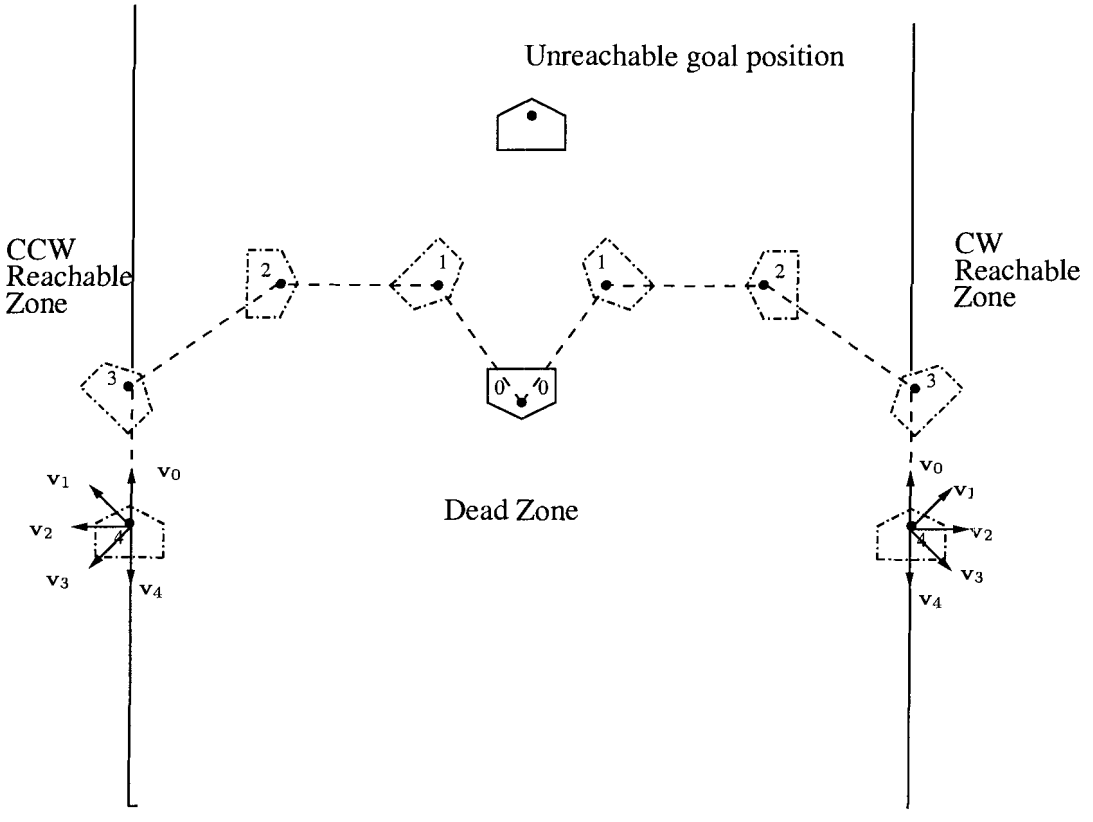


Figure 3.9: Dead zone of the goal position for the CW and CCW sequences of equilibrium pushes when $\Delta\theta = \pi$. The initial configuration of the object is labeled with 0. It requires four non-equilibrium pushes to achieve $\Delta\theta = \pi$. The associated push direction vectors $\mathbf{v}_0, \cdots, \mathbf{v}_4$ positively span the CW and CCW reachable zone. The goal position inside the dead zone can not be reached by the sequence of equilibrium pushes.

can be enlarged by modifying the sequence of non-equilibrium pushes. A modified sequence will consist of a usual sequence of non-equilibrium pushes preceded by non-equilibrium pushes in the opposite direction. By adding additional CW or CCW non-equilibrium pushes to original CCW or CW sequence, the CCW sequence becomes a CW-CCW sequence, and the CW sequence becomes a CCW-CW sequence. Using one additional non-equilibrium push will increase the total number of non-equilibrium pushes by two. The associated reorientation angles $\Delta\theta_+$ or $\Delta\theta_-$ will become bigger. This process will introduce new equilibrium pushing vectors into the associated basis vectors $\mathbf{v}_k$. Consequently, the reachable region for the sequence of equilibrium pushes will be enlarged. When the associated reorientation angles with CW-CCW or CCW-CW sequences are larger than π, the equilibrium push vectors will span the entire R^2 space.

An example of the modified CW-CCW sequence is shown in Fig. 3.10. EOF indicates the enlarged cone, and $\mathbf{v}_-$ is the additional equilibrium push vector. By adding a CW non-equilibrium push to the original CCW sequence (Fig. 3.8), the reachable cone is enlarged. If the goal position is inside the enlarged cone EOF, the goal position is reachable by a sequence of equilibrium pushes.

The required net configuration change $C_g - C_i$ determines the type of all feasible sequences of non-equilibrium pushes. The candidates are CW, CCW, CW-CCW, and CCW-CW sequences. These sequences can be organized according to the reorientation angle $\Delta\theta$. The classification of feasible non-equilibrium push sequences is depicted in Fig. 3.11. When $0 < \Delta\theta < \pi$, a CW sequence always exists since the reorienting angle $\Delta\theta_- > \pi$. There will also exist a CW-CCW sequence. CCW sequence is a special case of CW-CCW sequence with no CW segment. In this case, the goal position of the object is inside the conic hull generated by the equilibrium push vectors in the

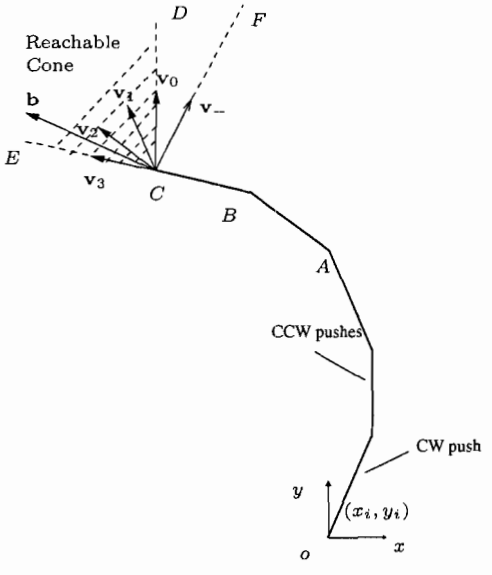


Figure 3.10: Enlarged reachable region by a modified CW-CCW sequence. After introducing a CW non-equilibrium push, an additional pushing direction vector $\mathbf{v}_-$ can be used by an equilibrium push. compared with Fig. 3.8, Cone DCF is the enlarged portion of the reachable region.

CCW sequence.

When $\pi < \Delta\theta < 2\pi$, a CCW sequence always exists since the reorienting angle $\Delta\theta_+ > \pi$. There will also exist a CCW-CW sequence. A CW sequence may exist if the goal position of the object is inside the conic hull generated by the equilibrium push vectors in the CW sequence.

If $\Delta\theta = 0$ and the goal position can not be reached by an equilibrium push, CW-CCW and CCW-CW sequences are required for the task. When $\Delta\theta = \pi$, there will be CW-CCW and CCW-CW sequences.

A procedure for constructing CW-CCW or CCW-CW sequence is shown in Fig. 3.12. The first step of the procedure is to compute the CW and CCW reorienting angles $\Delta\theta_+$ and $\Delta\theta_-$ for a given $\Delta\theta$. If $\Delta\theta_+ > \pi$, there exist both CW and CW-CCW non-equilibrium push sequences, and a CW sequence can be generated according to Section 3.5.1. For the CW-CCW sequence, by adding a CW push, we first generate the CW-CCW sequence, then check if the goal configuration is inside the cone. CCW and CCW-CW sequences are generated in the same manner as the CW and CW-CCW sequences.

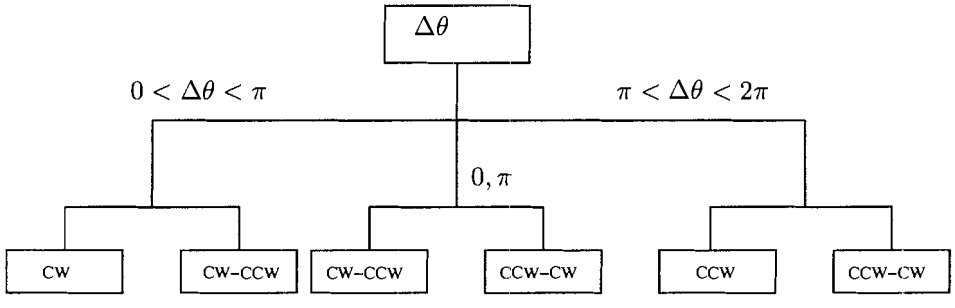


Figure 3.11: Classification of non-equilibrium push sequences

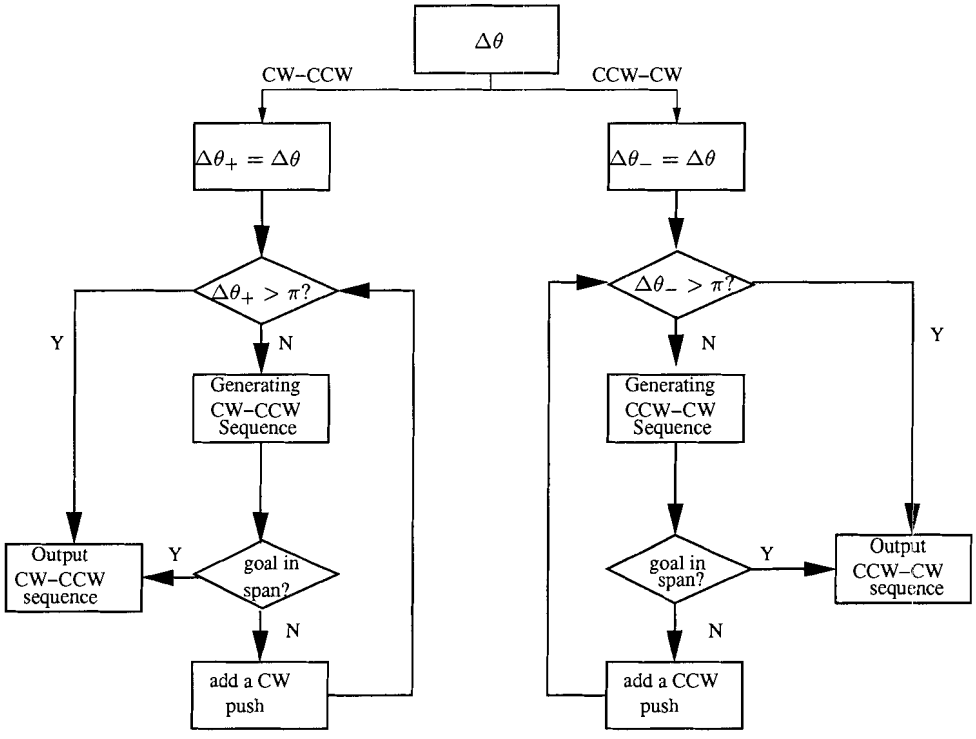


Figure 3.12: Construction of CW-CCW and CCW-CW sequences

3.5.3 The Optimal Controller for Equilibrium Pushes

After obtaining a sequence of non-equilibrium pushes that guarantees the goal position inside the reachable cone, we are going to find an optimal sequence of equilibrium pushes such that the position of the object achieves the goal position (x_g, y_g).

Since the basis vectors $\mathbf{v}_k(k = 0, \cdots, J_m)$ are usually positively dependent, the sequence of equilibrium pushes will not be unique for specified initial and goal configurations. This property implies that there always exists multiple choices of $m_e(k)(k = 0, \cdots, J_m)$ such that Eq. (3.39) is satisfied. This gives us freedom to find an optimal sequence of equilibrium pushes from all feasible sequences. If the objective is to minimize the total push distance $\sum_{k=0}^{J_m} m_e(k)$, the design problem for the controller $m_e(k)$ can be formulated as a linear programming problem:

$$\text{Minimize} \quad \sum_{k=0}^{J_m} m_e(k) \tag{3.46}$$

subject to

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} x_r(0) \\ y_r(0) \end{bmatrix} + \sum_{k=0}^{J_m} m_e(k) \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \end{bmatrix} \tag{3.47}$$

$$m_e(k) \geq 0 \quad \forall \quad k = 1, \cdots, J_m.$$

The constraint equation (3.47) can be rewritten as

$$\mathbf{b} = \sum_{k=0}^{J_m} m_e(k) \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \end{bmatrix} \tag{3.48}$$

where $\mathbf{b} = (x_g - x_r(0), y_g - y_r(0))^T$.

Rewrite the linear programming problem into the primal form [62] as

$$\text{Minimize} \quad \mathbf{c}^T\mathbf{m_e} \tag{3.49}$$

subject to

$$\begin{aligned} \mathbf{A}\mathbf{m_e} &= \mathbf{b} \\ \mathbf{m_e} &\geq 0 \end{aligned} \tag{3.50}$$

where

$$\begin{aligned} A &= \begin{bmatrix} \mathbf{v}_0 & \cdots & \mathbf{v}_{J_m} \end{bmatrix} \\ \mathbf{m_e} &= \begin{bmatrix} m_e(0) & \cdots & m_e(J_m) \end{bmatrix}^T \\ \mathbf{c} &= \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T. \end{aligned}$$

Before we solve the linear programming problem analytically, we first present some definitions and the fundamental theorem for linear programming ([62], p.17-19).

1. *Basic solution*: Given the set of m simultaneous linear equations with n unknowns

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{3.51}$$

Let $\mathbf{B}$ be any nonsingular $m \times m$ submatrix made up of columns of $\mathbf{A}$. Then, if all $n - m$ components of $\mathbf{x}$ not associated with columns of $\mathbf{B}$ are set equal to zero, the solution to the resulting set of equations is said to be a *basic solution* to Eq. (3.51) with respect to the basis $\mathbf{B}$. The components of $\mathbf{x}$ associated with columns of $\mathbf{B}$ are called *basic variables*.

2. *Basic feasible solution*: A vector $\mathbf{x}$ satisfying

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq 0 \end{aligned} \tag{3.52}$$

is said to be *feasible* for the constraints. A feasible solution to the constraints that is also basic is said to be a *feasible basic solution*.

3. *Optimal basic feasible solution*: Corresponding to a linear program in standard

form

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned} \tag{3.53}$$

A feasible solution to the constraints that achieves the minimum value of the objective function subject to the constraints is said to be an *optimal feasible solution.* If this solution is basic, it is an *optimal basic feasible solution.*

The fundamental theorem of linear programming indicates the importance of basic feasible solutions for solving a linear programming problem.

Theorem 3.3 (Fundamental theorem of linear programming [62].p.19): *Given a linear program in standard form (3.53), where* $\mathbf{A}$ *is an* $m \times n$ *matrix of rank* m,

1. *if there is a feasible solution, there is a basic feasible solution.*
2. *if there is an optimal feasible solution, there is an optimal basic feasible solution.*

This theorem indicates that solving a linear programming problem can be achieved by searching over basic feasible solutions. The fundamental theorem of linear programming will be used to design the optimal controller for equilibrium pushes. The result is summarized in the following theorem.

Theorem 3.4 (Properties of the optimal sequence of equilibrium pushes) *For the optimal equilibrium push problem described by Eq. (3.46) and Eq. (3.47), there exists an optimal solution that satisfies:*

1. *The maximum number of equilibrium pushes will be up to two. In other words, at most two components of* $m_e(.)$ *have non-zero values.*

2. *These two push components are adjacent to each other. This means there exists only one non-equilibrium push between these two equilibrium pushes. The optimal pushing directions* $\mathbf{v}_j, \mathbf{v}_{j+1}$ *generate a cone that encompasses the vector* $\mathbf{b}$.

Proof.

(1). For linear program (3.49) and (3.50), $\mathbf{A}$ is a $2 \times J_m$ matrix, the basic solution consists of two basic variables. By Theorem 3.2, there will always exist a feasible solution to the problem. Since $m_e(\cdot) \geq 0$, the lower bound on the objective (3.49) will be zero. This means there will always exist an optimal feasible solution. By the fundamental theorem of linear programming, we induce that there is an optimal basic feasible solution. For the optimal basic feasible solution, at most two components of $m_e(\cdot)$ have non-zero values, *i.e.*, an optimal sequence of two-agent cooperative push will consist of at most two equilibrium pushes. If there exists an equilibrium pushing direction vector along the same direction as vector $\mathbf{b}$, the number of equilibrium pushes will degenerate to one. If $\mathbf{b} = 0$, no equilibrium push is required.

(2). The task of solving linear program (3.49) and (3.50) is reduced to the search over the basic feasible solutions. Any set of two vectors that positively spans $\mathbf{b}$ become a basic solution; setting the indices of the solution vectors to i, j, then the problem can be written as

$$\text{Minimize} \quad m_{ei} + m_{ej} \tag{3.54}$$

subject to

$$\begin{aligned} &\begin{bmatrix} \mathbf{v}_i & \mathbf{v}_j \end{bmatrix} \begin{bmatrix} m_{ei} \\ m_{ej} \end{bmatrix} = \mathbf{b} \\ &m_{ei} \geq 0, \quad m_{ej} \geq 0 \end{aligned} \tag{3.55}$$

where m_{ei} and m_{ej} are the basic variables.

Now, we need to show that for an optimal solution, these two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ are adjacent to each other. Let α be the angle between $\mathbf{v}_i$ and $\mathbf{b}$, and β be the angle between $\mathbf{v}_j$ and $\mathbf{b}$, In order to guarantee m_{ei} and m_{ej} feasible, the equilibrium pushing vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ have to be chosen such that $\mathbf{b}$ lies inside the cone generated by these two vectors. As a result, the angles $\alpha, \beta \in (0, \pi/2)$, and the constraint equation (3.55) can be represented as

$$\begin{bmatrix} \cos\alpha & \cos\beta \\ \sin\alpha & -\sin\beta \end{bmatrix} \begin{bmatrix} m_{ei} \\ m_{ej} \end{bmatrix} = \begin{bmatrix} |\mathbf{b}| \\ 0 \end{bmatrix} \tag{3.56}$$

where $|\mathbf{b}|$ gives the length of the vector $\mathbf{b}$.

By solving this constraint equation, we get the pushing distances associated with the i-th and j-th push as functions of the angles α and β,

$$\begin{aligned} m_{ei} &= \tfrac{\sin\beta}{\sin(\alpha+\beta)}|\mathbf{b}| \\ m_{ej} &= \tfrac{\sin\alpha}{\sin(\alpha+\beta)}|\mathbf{b}|. \end{aligned} \tag{3.57}$$

Furthermore, the objective function $f(\alpha, \beta)$ is

$$f(\alpha, \beta) = m_{ei} + m_{ej} = \frac{\sin\alpha + \sin\beta}{\sin(\alpha+\beta)}|\mathbf{b}|. \tag{3.58}$$

An example plot of objective function with $|\mathbf{b}| = 1$ is shown in Fig. 3.13. It is easy to verify that the function $f(\alpha, \beta)$ is a monotonic increasing function with respect to $\alpha \in (0, \pi/2)$ and $\beta \in (0, \pi/2)$, in order to achieve minimum value for the objective function, we need to choose two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ with α and β as small as possible. Combining with the requirement of $\mathbf{b}$ is inside the convex cone spanned by $\mathbf{v}_i$ and $\mathbf{v}_j$, we can induce that one vector $\mathbf{v}_i$ is the first vector to the right hand side of Vector $\mathbf{b}$, the other vector $\mathbf{v}_j$ is the first vector to the left hand side of Vector $\mathbf{b}$. This proves that these two push vectors are adjacent to each other.

■

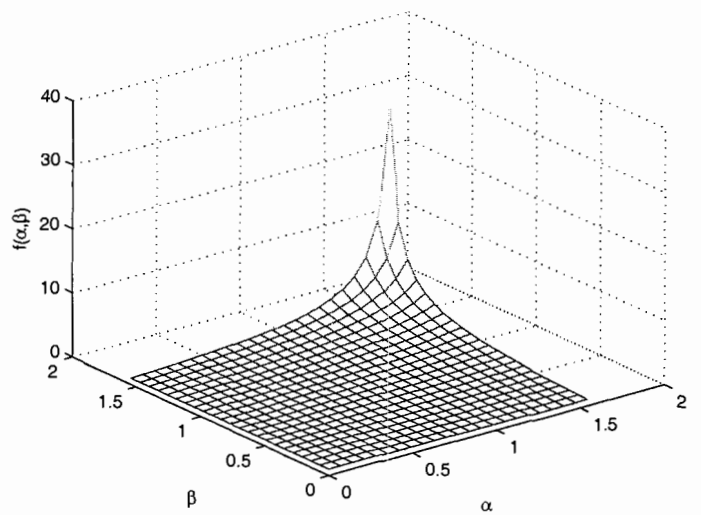


Figure 3.13: Plot of objective function $f(\alpha, \beta)$ with $|\mathbf{b}| = 1$. $f(\alpha, \beta)$ is a monotonic increase function of parameters α and β.

Instead of using a numerical linear programming package solving the optimal planning problem, Theorem 3.4 provides an analytical way to find the optimal equilibrium pushes for given initial and goal configurations. This solution technique can also be applied to [3] for finding the pose planner. The analytical procedure is summarized as follows: First, compute the vector $\mathbf{b}$; Then, find the two pushing direction vectors $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ using the result of Theorem 3.4, and calculate the angles α and β; Finally, compute the optimal distances for the equilibrium pushes according to Eq. (3.57).

3.5.4 Implementation of the Planner

For a manipulation task to transfer an object from an initial configuration (x_i, y_i, θ_i) to the goal configuration (x_g, y_g, θ_g), the planner for two-agent cooperative push is implemented as follows.

1. Adjust the spatial relationship parameters x_{a2} and y_{a2} such that the object can be captured at least at one equilibrium orientation.

2. Determine virtual stable edges using the *depth function* and *width function*, and calculate the equilibrium orientations (x_e, y_e, θ_e) of the object in the agent frame XOY.

3. For each virtual edge, find the maximum CW/CCW reorientation angles θ_- and θ_+ for a non-equilibrium push, then determine the minimum distance of push $M_n(\theta_-)$ and $M_n(\theta_+)$, and pick the first contact point (x_{p1}, y_{p1}).

4. Formulate the equations that govern the motion of the object under equilibrium and non-equilibrium pushes as Eq. (3.15) and Eq. (3.26).

5. Find sequences of CW-CCW and CW-CCW non-equilibrium pushes.

6. Check if there exists a sequence of equilibrium pushes using the result of Section 3.5.2, then find the optimal sequence of equilibrium pushes using linear programming method.

3.6 Manipulation Examples and Experimental Results

A triangular object as shown in Fig. 3.15 is used to demonstrate the proposed manipulation method. For this triangle, $AB = 0.1$ m, $BC = 0.05$ m. The object frame xoy is located at COM of the object. The agent frame XOY is assigned to one agent, and the coordinate of the second agent (x_{a2}, y_{a2}) are chosen as $x_{a2} = 0.06$ m, $y_{a2} = 0$

m, which defines the spatial relationship of these agents. During the manipulation, edges AB and BC are pushed by the agents.

The depth function of the object is depicted in Fig. 3.14. A minimum of the depth function occurs at $\theta_o = 53^o$ with $x_e = 0.021$ m and $y_e = 0.003$ m. When the object is pushed at this orientation, the push will be an equilibrium push. A CCW non-equilibrium push is shown in Fig. 3.15. In Fig. 3.15(a), one agent first makes contact with the object at point D on the edge BC, and $CD = 0.01$ m. The direction of push is indicated by the Y-axis of the agent frame XOY. After the minimum push distance $M_n = 0.15$ m, the object reaches the equilibrium orientation as shown in 3.15(b). The CCW reorienting angle is $\theta_+ = 53^o$. Assume that the configuration of the object is $(x(k), y(k), \theta(k))$ before the k-th non-equilibrium push, the XOY frame's orientation $\theta_a(k)$ is computed according to Eq. (3.17) as

$$\theta_a(k) = \theta(k). \tag{3.59}$$

The net change of the object's position in the frame XOY associated with the rotation is

$$\begin{bmatrix} \Delta x_o(k) \\ \Delta y_o(k) \end{bmatrix} = \begin{bmatrix} -0.014 \\ -0.030 \end{bmatrix} \tag{3.60}$$

and the corresponding position change due to the translation is

$$\begin{bmatrix} \Delta x_t(k) \\ \Delta y_t(k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0.15 \end{bmatrix}. \tag{3.61}$$

In the world frame, the total position change in the k-th non-equilibrium push is computed according to Eq. (3.25) as

$$\begin{bmatrix} \Delta x(k) \\ \Delta y(k) \end{bmatrix} = \begin{bmatrix} \cos\theta_a(k) & -\sin\theta_a(k) \\ \sin\theta_a(k) & \cos\theta_a(k) \end{bmatrix} \begin{bmatrix} -0.014 \\ 0.12 \end{bmatrix} \tag{3.62}$$

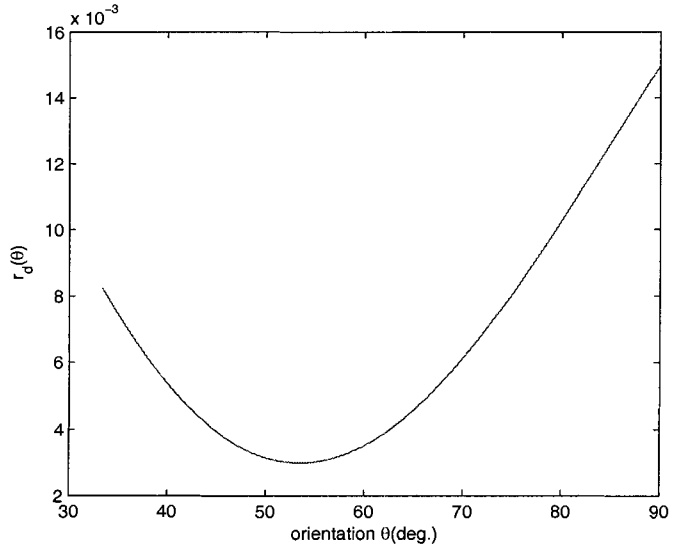


Figure 3.14: The depth function for the triangular object. The equilibrium orientation occurs at $\theta = 57$deg in the agent frame.

The motion of the object under the CCW non-equilibrium push is governed by the following equation

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \\ \theta_n(k) \end{bmatrix} \tag{3.63}$$

where $(x(k+1), y(k+1), \theta(k+1))$ is the configuration of the object after the k-th non-equilibrium push, $\theta_n(k) = \theta_+$.

A CW non-equilibrium push is shown in Fig. 3.16. In Fig. 3.16(a), one agent first makes contact with the object at point E on the edge B, and $BE = 0.05$ m. The direction of push is indicated by the Y-axis of the agent frame XOY. After the minimum distance of pushing $M_n = 0.12$ m, the object reaches the equilibrium orientation as shown in 3.16(b). The CW reorienting angle is $\theta_- = -37^o$. Assume that the configuration of the object is $(x(k), y(k), \theta(k))$ before the k-th non-equilibrium

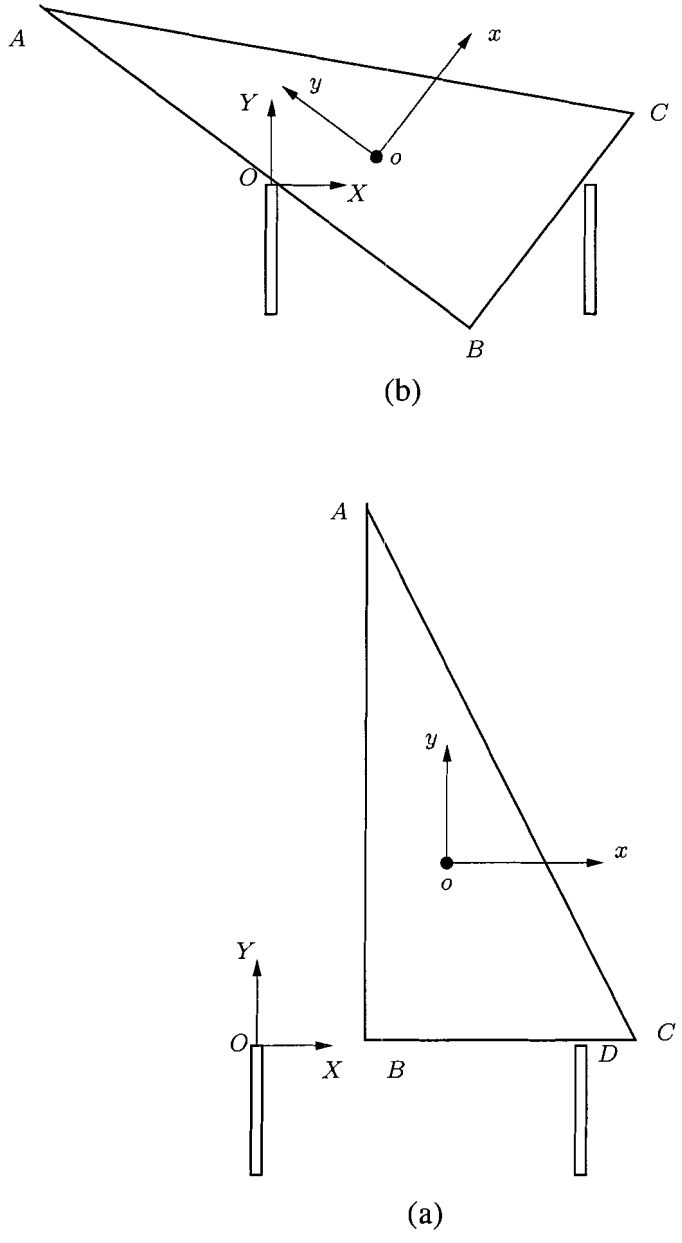


Figure 3.15: CCW non-equilibrium push. (a). One agent makes contact with the object at the edge BC. (b). After a CCW rotation, the object reaches its equilibrium orientation.

push, the XOY frame's orientation $\theta_a(k)$ is computed according to Eq. (3.17)

$$\theta_a(k) = \theta(k) - \pi/2. \tag{3.64}$$

Under the CW non-equilibrium push, the net change of the object's position in the frame XOY associated with the rotation is

$$\begin{bmatrix} \Delta x_o(k) \\ \Delta y_o(k) \end{bmatrix} = \begin{bmatrix} -0.004 \\ -0.012 \end{bmatrix} \tag{3.65}$$

and the corresponding position change due to the translation is

$$\begin{bmatrix} \Delta x_t(k) \\ \Delta y_t(k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0.12 \end{bmatrix}. \tag{3.66}$$

In the world frame, the total position change in the k-th non-equilibrium push is computed according to Eq. (3.25)

$$\begin{bmatrix} \Delta x(k) \\ \Delta y(k) \end{bmatrix} = \begin{bmatrix} \cos\theta_a(k) & -\sin\theta_a(k) \\ \sin\theta_a(k) & \cos\theta_a(k) \end{bmatrix} \begin{bmatrix} 0.004 \\ 0.108 \end{bmatrix}. \tag{3.67}$$

The motion of the object under the CCW non-equilibrium push is governed by the following equation

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \\ \theta_n(k) \end{bmatrix} \tag{3.68}$$

where $(x(k+1), y(k+1), \theta(k+1))$ is the configuration of the object after the k-th non-equilibrium push, $\theta_n(k) = \theta_-$.

For the equilibrium push, the XOY frame's orientation $\theta_a(k)$ is computed as

$$\theta_a(k) = \theta(k) - \theta_o. \tag{3.69}$$

Under the equilibrium push, the motion of the object is governed by the following equation

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} -\sin\theta_a(k) \\ \cos\theta_a(k) \\ 0 \end{bmatrix} m_e(k) \tag{3.70}$$

where $(x(k+1), y(k+1), \theta(k+1))$ is the configuration of the object after the k-th equilibrium push, and $m_e(k) \geq 0$ is the distance of the k-th equilibrium push.

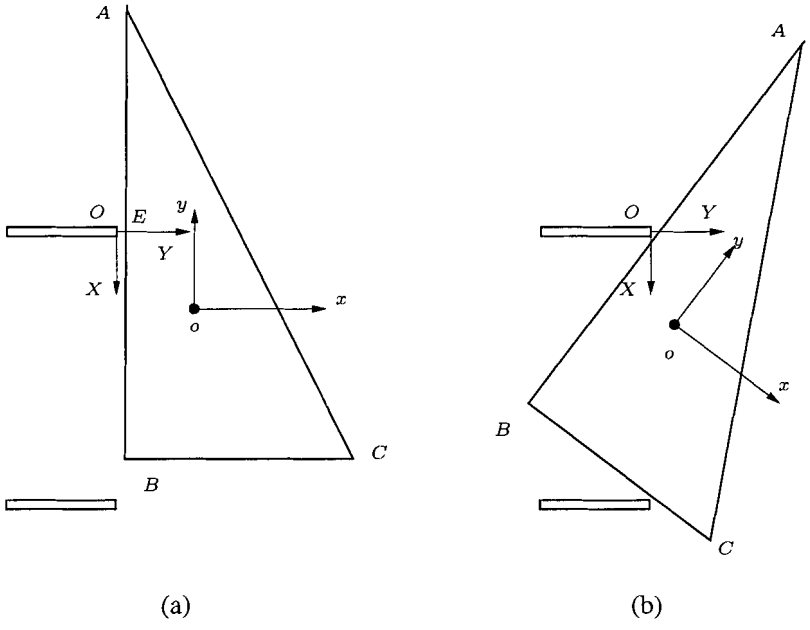


Figure 3.16: CW non-equilibrium push. (a). One agent makes contact with the object at the edge AB. (b). After a CW rotation, the object reaches its equilibrium orientation.

The motion of the object is governed by the equation (3.63) or (3.70) depending on the type of push being used. Assume that the initial configuration $C_i = (x_i, y_i, \theta_i) = (0, 0, 0)$. Here we consider two goal configurations as $C_{g1} = (x_g, y_g, \theta_g) = (-0.2, 0.25, 171^o)$, and $C_{g2} = (x_g, y_g, \theta_g) = (-0.15, 0, 171^o)$. It requires three CCW non-equilibrium pushes to orient the object to the goal orientation. The

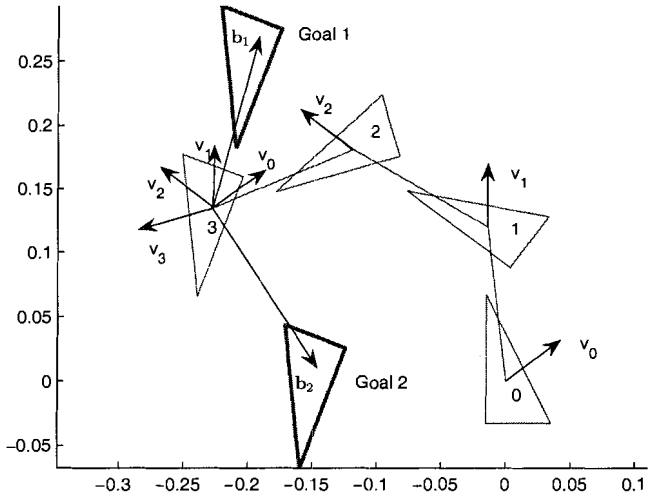


Figure 3.17: CCW non-equilibrium push sequence and the reachable region. Two goal positions are labeled as Goal 1 and Goal 2. The push direction vectors of equilibrium pushes are given by $\mathbf{v}_0, \cdots, \mathbf{v}_3$. Vector $\mathbf{b}_1$ is positively spanned by the push direction vectors, and vector $\mathbf{b}_2$ is not positively spanned by the push direction vectors.

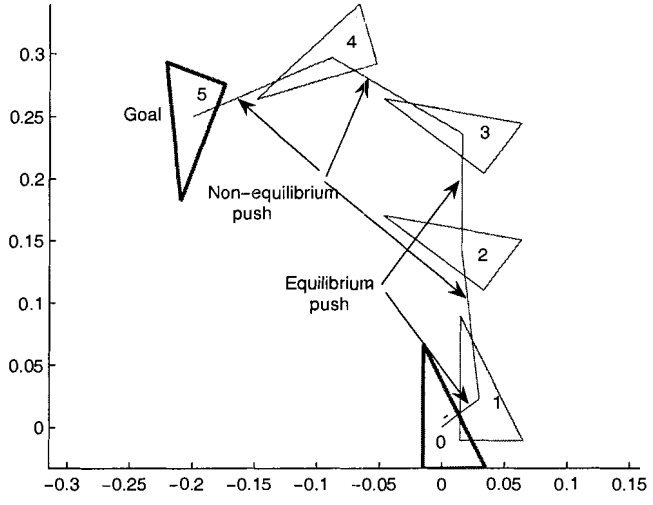


Figure 3.18: Optimal CCW push sequence for goal one. The optimal sequence consists of three non-equilibrium pushes and two equilibrium pushes. These two equilibrium pushes are adjacent to each other.

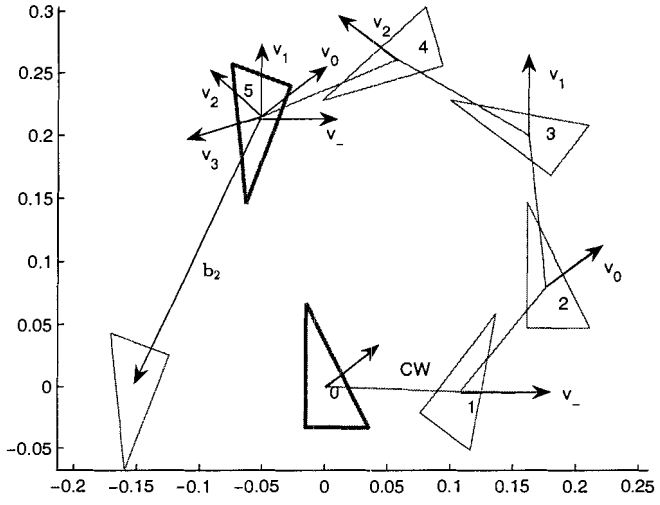


Figure 3.19: CW-CCW non-equilibrium push sequence and the reachable region. By introducing a CW non-equilibrium push to the CCW sequence, A new push direction vector $\mathbf{v}_-$ helps enlarge the reachable region of the equilibrium push sequence. The push direction vectors $\mathbf{v}_-, \cdots, \mathbf{v}_3$ span R^2 space. With these pushing directions, any position in the plane can be reached by a sequence of equilibrium pushes.

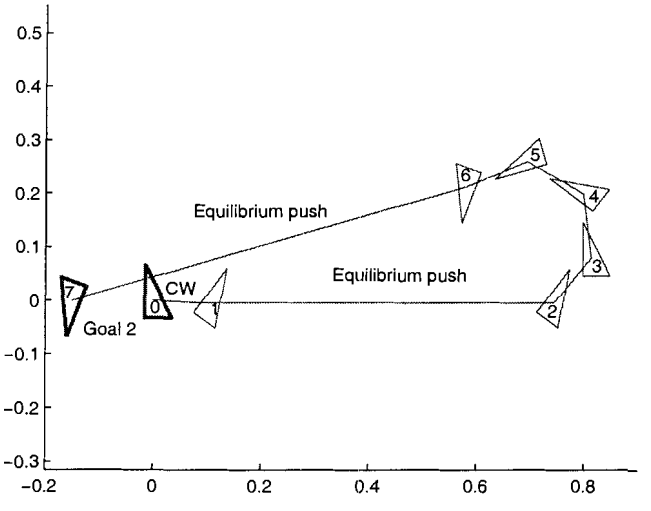


Figure 3.20: Optimal CW-CCW push sequence for goal 2. The optimal sequence consists of one CW non-equilibrium push, four CCW non-equilibrium pushes and two equilibrium pushes. These two equilibrium pushes are adjacent to each other. There only exists a CW non-equilibrium push between them.

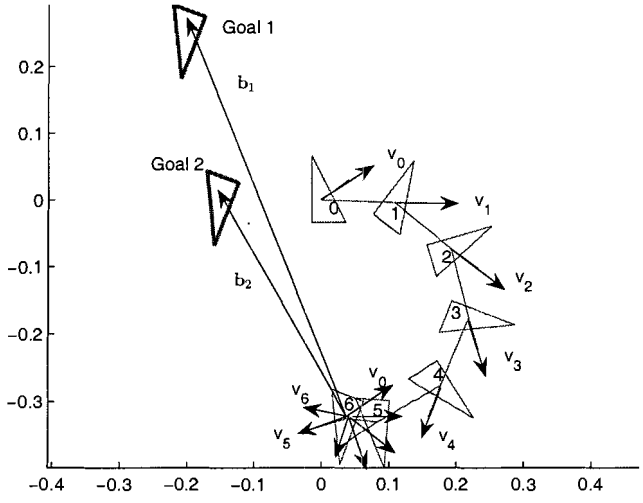


Figure 3.21: CW non-equilibrium push sequence and the reachable region. The CW push direction vectors are given by $\mathbf{v}_0, \cdots, \mathbf{v}_6$. These vectors span R^2 space. The position vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are positively spanned by the vectors; thus, Goal 1 and Goal 2 can be reached by a sequence of equilibrium pushes.

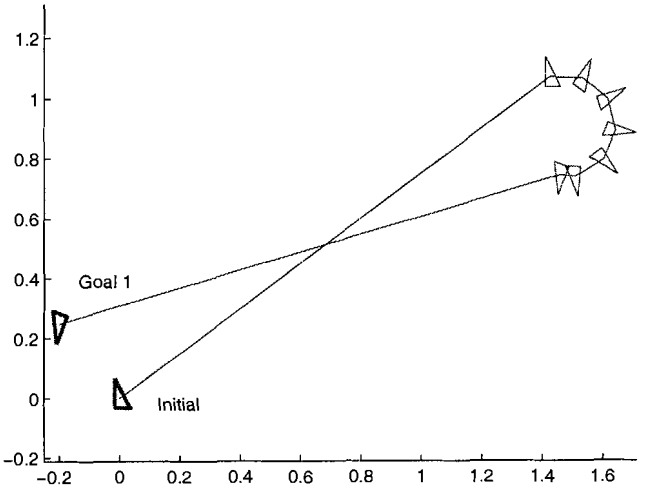


Figure 3.22: Optimal CW push sequence for goal 1. This sequence consists of six non-equilibrium pushes and two equilibrium pushes. The first and the final pushes are equilibrium pushes.

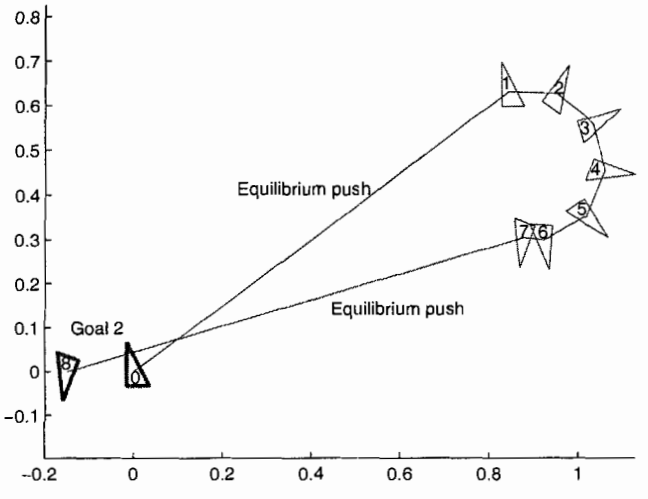


Figure 3.23: Optimal CW push sequence for goal 2. This sequence consists of six non-equilibrium pushes and two equilibrium pushes. The first and the final pushes are equilibrium pushes.

sequence of CCW non-equilibrium pushes and the reachable region is shown in Fig. 3.17. The reachable cone is spanned by vectors v_0, v_1, v_2, and v_3. If the goal position is inside the cone, a CCW sequence is sufficient for the manipulation task. Otherwise, a modified sequence is needed for the task. The goal C_{g1} is inside the cone, and the goal C_{g2} is outside the cone.

For the first goal C_{g1}, it requires a CCW pushing sequence with two equilibrium pushes and three non-equilibrium pushes. The equilibrium pushes along the directions indicated by v_0 and v_1 gives the optimal CCW sequence, and the optimal sequence is shown in Fig. 3.18. The pushing distance for individual equilibrium pushes are $m_e(0) = 0.0371$ m, and $m_e(1) = 0.094$ m.

Since the second goal position is not inside the cone, it requires an additional CW non-equilibrium push to the original CCW non-equilibrium push sequence. The CW-CCW sequence and the reachable region is shown in Fig. 3.19. By adding the pushing vector v_-, the total reorienting angle is larger than π. Therefore, the set of vectors v_-, v_0, v_1, v_2, and v_3 span the entire R^2 space. The position vector $\mathbf{b}_2$ is inside the cone formed by vectors v_- and v_3. The CW-CCW sequence with equilibrium pushes along the direction v_- and v_3 gives the optimal CW-CCW sequence. The corresponding distances for the sequence of equilibrium pushes are $m_e(-) = 0.6366$ m, and $m_e(3) = 0.7634$ m. The resulting CW-CCW sequence is shown in Fig. 3.20. The CW push sequence and the reachable region is shown in Fig. 3.21. For the CW sequence, the total reorienting angle is larger than π. The set of pushing vectors span the entire R^2 space. The optimal push sequence for goal one and two are shown in Fig. 3.22 and Fig. 3.23.

In order to validate the feasibility of these two manipulation primitives, experiments have been conducted by using a two-fingered gripper pushing a triangular

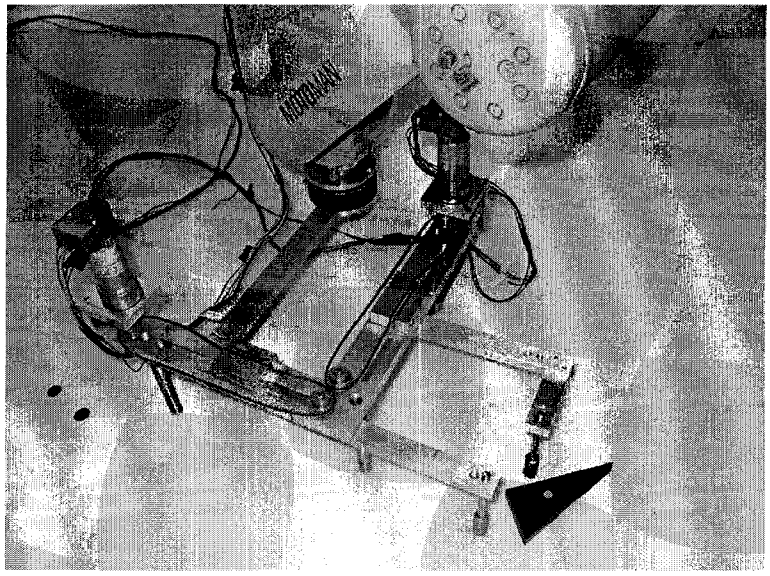


Figure 3.24: Experimental setup. Two motors are controlled to adjust the spatial relationship of the agents. The MOTOMAN robot drives the two-fingered gripper to push the object.

block. The two-fingered gripper is mounted on a MOTOMAN robot where each finger is driven by a motorm, and the motors are controlled by a computer through a motion control card. The experimental setup is shown in Fig. 3.24. The block has the same dimension as the triangle used in the simulations.

In the experiment, after moving the fingers to satisfying a certain spatial relationship, we control the motion of the MOTOMAN robot, and it performs the equilibrium or non-equilibrium push. Sequences of CCW and CW non-equilibrium pushes are shown in Fig. 3.25-3.26. These sequence show the object before and after the CW and CW non-equilibrium pushes. Experiments indicate that objects can be manipulated by using CCW and CW sequences of equilibrium and non-equilibrium pushes.

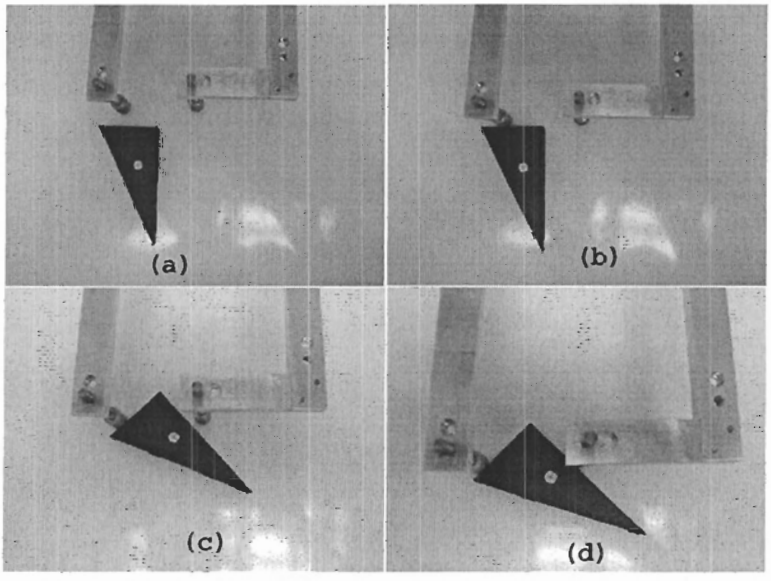


Figure 3.25: Object under a CCW non-equilibrium push. (a). Before contact, (b) one agent makes contact, (c). the object rotates between the agents, (d) the object reaches its equilibrium orientation.

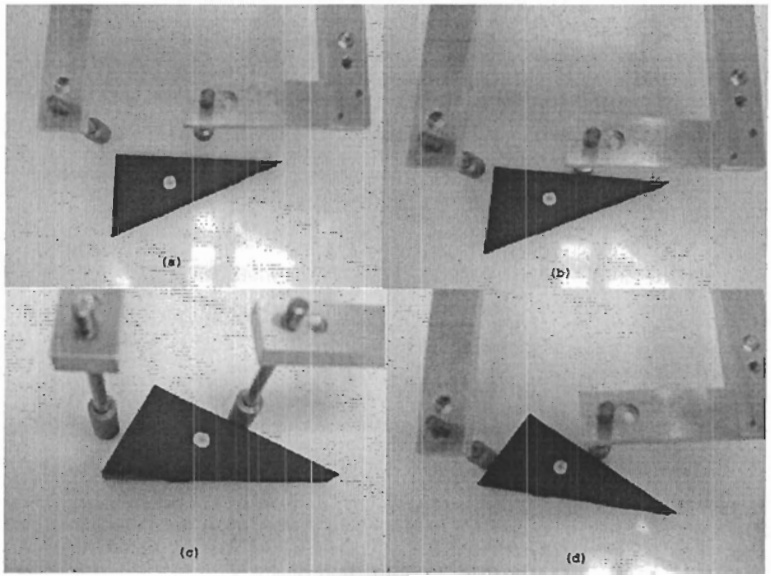


Figure 3.26: Object under a CW non-equilibrium push. (a). Before contact, (b) one agent makes contact, (c). the object rotates between the agents, (d) the object reaches its equilibrium orientation.

3.7 Discussion

In this chapter, we introduced the concept of a virtual fence for the parts transfer task. This method extends the work in [3] and it is capable of manipulating non-polygonal convex objects. We identified two push primitives, namely, equilibrium push and non-equilibrium push. Through the concept of virtual fence, a close relation between fence and point contact based pushing was discovered. After formulating the manipulation process into a switched system, the planning problem becomes a switched control problem. We prove that, in an optimal plan, only two equilibrium push actions are required. A fully analytical planner has been developed for the manipulation task.

In the proposed method, a fixed spatial relationship between agents is considered, and only a single virtual edge is used to perform the pushes. Because equilibrium and non-equilibrium controllers are designed separately, the best plan found is not guaranteed to be globally optimal. Selecting variable spatial relation between agents and designing the equilibrium and non-equilibrium controllers simultaneously may provide more efficient planners.

If we want to deal with toleranced parts, we need to define a proper tolerance model [6], and study the effect on the equilibrium and non-equilibrium push primitives. Alternatively, sensory information and feedback may be used to deal with shape uncertainty.

Chapter 4

Planning the Velocity of Free Sliding Objects

This chapter focuses on solving the planning problem to find the initial velocity of a free sliding object for the given initial and final configurations. In the free sliding problem, the initial and final configurations of the object are known and, if the travel time of the object is also known, the problem can be solved as a two-point boundary value (TPBV) problem. However, the travel time of the object is unspecified. We only know that, at the final goal configuration, the velocity of the object becomes zero. Since we do not know the travel time, the free sliding problem can only be formulated as a free boundary value problem (FBVP), and the assumed zero velocity at the final goal configuration provides another boundary condition for finding the travel time of the object. In order to use well known existing techniques to solve the FBVP [8], the problem is first reduced to a standard TPBV problem, and solved by shooting methods. A recent summary of numerical shooting methods can be found in [7]. Shooting method is one of the standard techniques for solving a TPBV problem, and it is implemented by integrating the initial value solver with an optimization routine.

The quasi-Newton method is used as the optimization routine in this application. To improve the global convergence, a line search strategy is incorporated with the quasi-Newton method [28]. In general shooting methods, the computation of the Jacobian matrix is always time consuming because several initial value problems have to be solved. In order to reduce the cost associated with the computation of the Jacobian matrix, we modify the quasi-Newton method with the Broyden update [74]. In each iteration, the Jacobian matrix is updated iteratively without solving the initial value problem.

The remainder of this chapter is organized as follows. In Section 4.1, we first derive the model for the sliding object, then we formulate the velocity planning problem as a free boundary value problem. In Section 4.2, we present the planning algorithms and implementation. In Section 4.3, we present the simulation and experiment results. Finally, we conclude in Section 4.4 with a discussion.

4.1 Problem Formulation

4.1.1 The Model of a Sliding Object

The motion of a free sliding object on a horizontal plane is governed by the friction force between the object and the plane. With known friction coefficient and pressure distribution, the friction force and torque are calculated by integrating through each infinitesimal element of the object. Consider an object sliding on the plane as shown in Fig. 4.1. XOY is the world frame, and xoy is the local coordinate frame associated with the object with o located at COM. The configuration of the object is defined as (X_o, Y_o, θ), where (X_o, Y_o) gives the position of COM, and θ is the orientation of the object.

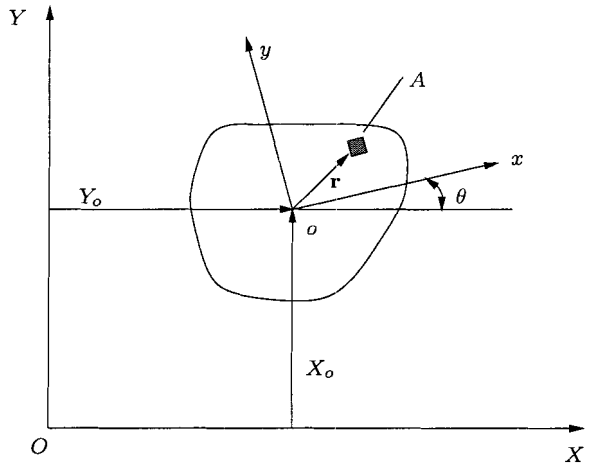


Figure 4.1: Motion of a planar object

The linear velocity of o is defined as $\mathbf{v} = (\dot{X}_o, \dot{Y}_o)$, and we assume that on each infinitesimal element of the object, there exists a friction force. Denote A as an infinitesimal element of the object located at (x, y) in the local frame, and $\mathbf{r}$ is the position vector from o to A. $dm = \rho(x, y)dxdy$ is the mass of element A, where $\rho(x, y)$ is the mass distribution function of the object at position (x, y). $\rho(x, y)g$ is the pressure distribution function, where g is the acceleration of gravity. $\omega = \dot{\theta}$ is the angular velocity of the object. μ is the Coulomb friction coefficient. Under the Coulomb friction assumption, the magnitude of the friction force acting on the element A is independent of the magnitude of the velocity, and it is calculated as

$$\mathbf{dF} = \mu g \cdot dm. \tag{4.1}$$

The direction of friction force is always opposite to the direction of the unit velocity vector $\hat{\mathbf{v}}_A$. The unit velocity vector $\hat{\mathbf{v}}_A$ for the element A is

$$\hat{\mathbf{v}}_A = \frac{\mathbf{v}_A}{|\mathbf{v}_A|} = \frac{\mathbf{v} + \omega \times \mathbf{r}}{|\mathbf{v} + \omega \times \mathbf{r}|}. \tag{4.2}$$

Let $\dot{X}, \dot{Y}$ be the linear velocity components of the element A in the world frame, and they are calculated as

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} \dot{X}_o \\ \dot{Y}_o \end{bmatrix} + R \begin{bmatrix} -\omega \cdot y \\ \omega \cdot x \end{bmatrix} \tag{4.3}$$

where

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4.4}$$

is the rotation matrix associated with the object.

Substituting Eq. (4.4) into Eq. (4.3), we get

$$\begin{aligned} \dot{X} &= \dot{X}_o - \cos\theta \cdot \omega \cdot y - \sin\theta \cdot \omega \cdot x \\ \dot{Y} &= \dot{Y}_o - \sin\theta \cdot \omega \cdot y + \cos\theta \cdot \omega \cdot x. \end{aligned} \tag{4.5}$$

In the world frame XOY, the friction force $dF = (dF_X, dF_Y)^T$ is calculated as

$$dF_X = -\mu g \rho(x,y) dx dy \frac{\dot{X}}{\sqrt{\dot{X}^2 + \dot{Y}^2}} \tag{4.6}$$

$$dF_Y = -\mu g \rho(x,y) dx dy \frac{\dot{Y}}{\sqrt{\dot{X}^2 + \dot{Y}^2}}. \tag{4.7}$$

Substituting Eq. (4.5) into Eqs. (4.6)and (4.7), we get

$$\begin{aligned} dF_X = \; & -\mu \cdot g \cdot \rho(x,y) dx dy \\ & \cdot \frac{\dot{X}_o - \cos\theta\cdot\omega\cdot y - \sin\theta\cdot\omega\cdot x}{\sqrt{(\dot{X}_o - \cos\theta\cdot\omega\cdot y - \sin\theta\cdot\omega\cdot x)^2 + (\dot{Y}_o - \sin\theta\cdot\omega\cdot y + \cos\theta\cdot\omega\cdot x)^2}} \end{aligned} \tag{4.8}$$

$$\begin{aligned} dF_Y = \; & -\mu \cdot g \cdot \rho(x,y) dx dy \\ & \cdot \frac{\dot{Y}_o - \sin\theta\cdot\omega\cdot y + \cos\theta\cdot\omega\cdot x}{\sqrt{(\dot{X}_o - \cos\theta\cdot\omega\cdot y - \sin\theta\cdot\omega\cdot x)^2 + (\dot{Y}_o - \sin\theta\cdot\omega\cdot y + \cos\theta\cdot\omega\cdot x)^2}}. \end{aligned} \tag{4.9}$$

The associated frictional torque with respect to the COM o is

$$\mathbf{dT} = \mathbf{r} \times \mathbf{dF} = -\mu g \cdot \mathbf{r} \cdot dm. \tag{4.10}$$

To calculate the torque $\mathbf{dT}$ generated by the friction force of the element A about the COM, we first transfer the friction force components dF_X and dF_Y into the local frame as $\mathbf{dF^l} = (dF_X^l \quad dF_Y^l)^T$, and they are computed as

$$\begin{bmatrix} dF_X^l \\ dF_Y^l \end{bmatrix} = R^T \begin{bmatrix} dF_X \\ dF_Y \end{bmatrix}. \tag{4.11}$$

Using the friction force in the local frame, the torque dT is calculated as

$$\begin{aligned} \mathbf{dT} &= \mathbf{r} \times \mathbf{dF^l} \\ &= dF_X^l \cdot (-y) + dF_Y^l \cdot x \\ &= -\mu \cdot g \cdot \rho(x,y) \frac{-(\dot{X}_o x + \dot{Y}_o y)\sin\theta - (\dot{X}_o y - \dot{Y}_o x)\cos\theta + \omega \cdot y^2 + \omega \cdot x^2}{\sqrt{(\dot{X}_o - \cos\theta\cdot\omega\cdot y - \sin\theta\cdot\omega\cdot x)^2 + (\dot{Y}_o - \sin\theta\cdot\omega\cdot y + \cos\theta\cdot\omega\cdot x)^2}}. \end{aligned} \tag{4.12}$$

After obtaining the friction force and torque associated with the element A, the net friction force and torque acting on the object are calculated by integrating $\mathbf{dF}$ and $\mathbf{dT}$ over the entire contact area $barA$, *i.e.*,

$$\mathbf{F} = \int_{\bar{A}} \mathbf{dF} = -\mu g \int_{\bar{A}} \hat{\mathbf{v}}_A dm \tag{4.13}$$

$$\mathbf{T} = \int_{\bar{A}} \mathbf{r} \times \mathbf{dF} = -\mu g \int_{\bar{A}} \mathbf{r} \times \hat{\mathbf{v}}_A dm. \tag{4.14}$$

By integrating Eqs. (4.8),(4.9), and (4.12) over the contact area $\bar{A}$, we get the friction force components and torque on the object

$$\begin{aligned} F_X = \; & -\mu \cdot g \int\int \rho(x,y) \\ & \cdot \frac{\dot{X}_o - \cos\theta\cdot\omega\cdot y - \sin\theta\cdot\omega\cdot x}{\sqrt{(\dot{X}_o - \cos\theta\cdot\omega\cdot y - \sin\theta\cdot\omega\cdot x)^2 + (\dot{Y}_o - \sin\theta\cdot\omega\cdot y + \cos\theta\cdot\omega\cdot x)^2}} dxdy \end{aligned} \tag{4.15}$$

$$\begin{aligned} F_Y = \; & -\mu \cdot g \int\int \rho(x,y) \\ & \cdot \frac{\dot{Y}_o - \sin\theta\cdot\omega\cdot y + \cos\theta\cdot\omega\cdot x}{\sqrt{(\dot{X}_o - \cos\theta\cdot\omega\cdot y - \sin\theta\cdot\omega\cdot x)^2 + (\dot{Y}_o - \sin\theta\cdot\omega\cdot y + \cos\theta\cdot\omega\cdot x)^2}} dxdy \end{aligned} \tag{4.16}$$

$$\begin{aligned} T = \; & -\mu \cdot g \int\int \rho(x,y) \\ & \cdot \frac{-(\dot{X}_o x + \dot{Y}_o y)\sin\theta - (\dot{X}_o y - \dot{Y}_o x)\cos\theta + \omega \cdot y^2 + \omega \cdot x^2}{\sqrt{(\dot{X}_o - \cos\theta\cdot\omega\cdot y - \sin\theta\cdot\omega\cdot x)^2 + (\dot{Y}_o - \sin\theta\cdot\omega\cdot y + \cos\theta\cdot\omega\cdot x)^2}} dxdy. \end{aligned} \tag{4.17}$$

The resulting formulations for the forces and torque are in the same form as those in [37], in which they were derived by using the principle of virtual work. Numerical computation procedures on friction forces F_X, F_Y and torque T for general shaped objects can be found in Appendix A.

With known friction forces and torque, the motion of the sliding object is governed by

$$\begin{aligned} \ddot{X}_o &= F_X/m \\ \ddot{Y}_o &= F_Y/m \\ \ddot{\theta} &= T/I_0 \end{aligned} \tag{4.18}$$

where m is the mass of the object, and I_0 is the mass moment of inertia about the COM. By introducing the state variable $\overline{\mathbf{x}} = (x_1, x_2, x_3, x_4, x_5, x_6) = (\dot{X}_o, X_o, \dot{Y}_o, Y_o, \dot{\theta}, \theta)$, Eq. (4.18) is rewritten in state space form as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{F_X}{m} \\ x_4 \\ \frac{F_Y}{m} \\ x_6 \\ \frac{T}{I_0} \end{bmatrix} = \begin{bmatrix} f_1(\overline{\mathbf{x}}) \\ f_2(\overline{\mathbf{x}}) \\ f_3(\overline{\mathbf{x}}) \\ f_4(\overline{\mathbf{x}}) \\ f_5(\overline{\mathbf{x}}) \\ f_6(\overline{\mathbf{x}}) \end{bmatrix}. \tag{4.19}$$

This set of differential equations will be used to find the initial velocity of the free sliding object.

Before studying the planning problem, we first present some existing results on the characteristics of the final motion of a sliding object. For an axis-symmetric object with even mass distribution, it was shown that the translational and rotational velocities will come to a simultaneous stop after sliding [42, 102]. On the other hand, for an object with a general geometrical boundary and mass distribution, it is difficult to predict the exact characteristics of the final motion. Through analyzing the relationship between mass distribution and eigen-direction, Goyal et al. [36] concluded

that the final motion will be a pure translation if the mass is sufficiently central, and the final motion would be a pure rotation if the mass is sufficiently distributed over the contact area. Ishlinskii et al. [42] studied the motion of a two-mass system over a plane with two point masses connected by a weightless bar. Simulation results indicate that the final motion is a pure rotation about one mass point.

In general, it is difficult to predict the exact pattern of the final motion. Instead, we will use the condition

$$v_f(T_f) = \sqrt{\dot{X}^2(T_f) + \dot{Y}^2(T_f) + \dot{\theta}^2(T_f)} = 0 \tag{4.20}$$

as one boundary condition to determine the travel time T_f of the object. Because of the discontinuities of the friction forces (4.15), (4.16), and torque (4.17) at the point $\dot{X}_o = \dot{Y}_o = \dot{\theta} = 0$, it is hard to check the condition $v_f(T_f) = 0$ in practice. For this reason, we will specify a small scalar threshold ϵ, and consider the object stops when

$$v_f(T_f) \leq \epsilon \tag{4.21}$$

which enables us to determine the total travel time of the object.

4.1.2 Formulation of the Planning Problem

Consider an object with known geometry, mass distribution, and friction properties sliding on a supporting plane. At the beginning, the object sits at a known initial configuration (X_i, Y_i, θ_i). After being accelerated to an initial velocity, it starts sliding. It will slow down due to the friction, and eventually come to a stop at the final configuration (X_f, Y_f, θ_f). The objective of the planning is to determine the initial release velocity $(\dot{X}_i, \dot{Y}_i, \dot{\theta}_i)$.

The motion of the sliding object on a plane is governed by Eq. (4.19), which is a system of six first-order differential equations. Due to the nonlinearities of the friction

force and torque, there does not exist an analytical solution of the initial velocity for a given displacement of the configuration. Numerical procedures need to be developed to find the desired initial velocity.

The solution of system (4.19) depends on its initial conditions. If the initial configuration (X_i, Y_i, θ_i), and velocity $(\dot{X}_i, \dot{Y}_i, \dot{\theta}_i)$ are known, the trajectory of the motion will be uniquely determined, and it can be found by numerical integration of the system (4.19). This is known as the initial value problem.

If, on the other hand, the initial configuration (X_i, Y_i, θ_i) and the final goal configuration (X_f, Y_f, θ_f) are known, plus the travel time for Eq. (4.19) is specified, *i.e.*, T_f is known, the problem becomes a standard TPBV problem. However, for the free sliding planning problem, the travel time T_f is unknown. Thus, the planning problem is not a TPBV problem. As a result, we need to determine the travel time T_f using the condition on the velocity (4.21). By introducing this condition as a boundary condition, the system (4.19) will have a solution satisfying the following boundary conditions

$$\begin{cases} X_o(0) = X_i & X_o(T_f) = X_f \\ Y_o(0) = Y_i & Y_o(T_f) = Y_f \\ \theta_o(0) = \theta_i & \theta_o(T_f) = \theta_f \\ v_f(T_f) = \epsilon & (\epsilon > 0 \text{ is a small number}). \end{cases} \tag{4.22}$$

Solving the system (4.19) with boundary conditions (4.22) is referred to as a free boundary value problem.

To solve the FBVP, we first transfer the FBVP into a TPBV problem by introducing a new independent variable [7]. Here, in place of time t, we introduce a new independent variable τ, such that

$$t = \tau T_f \quad 0 \leq \tau \leq 1 \tag{4.23}$$

$$\dot{T}_f = \frac{dT_f}{d\tau} = 0. \tag{4.24}$$

In Eq. (4.23), we know that when the variable τ varies from $\tau = 0$ to $\tau = 1$, the system (4.19) will travel from $t = 0$ to $t = T_f$. From Eq. (4.24), we know that T_f is independent of τ.

After substituting Eq. (4.23) into Eq. (4.19), and augmenting Eq. (4.24) to Eq. (4.19), the motion of the sliding object is governed by the following system of differential equations

$$\begin{cases} \frac{dx_1}{d\tau} &= T_f \cdot f_1(\overline{\mathbf{x}}) \\ \frac{dx_2}{d\tau} &= T_f \cdot f_2(\overline{\mathbf{x}}) \\ &\vdots \\ \frac{dx_6}{d\tau} &= T_f \cdot f_6(\overline{\mathbf{x}}) \\ \frac{dT_f}{d\tau} &= 0. \end{cases} \tag{4.25}$$

The system of differential equations (4.25) is in standard TPBV form with τ varying between known limits of 0 and 1. The associated boundary conditions are

$$\begin{cases} X_o(0) = X_i & X_o(1) = X_f \\ Y_o(0) = Y_i & Y_o(1) = Y_f \\ \theta_o(0) = \theta_i & \theta_o(1) = \theta_f \\ v_f(1) = \epsilon. \end{cases} \tag{4.26}$$

The planning problem is to determine the initial velocity $(\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0))$ and the travel time T_f, such that the solution of system (4.25) satisfies the boundary conditions (4.26).

Numerical methods for solving TPBV problems fall into four categories [7]: (1) the shooting method, (2) the finite difference method, (3) the variational method and (4) collocation method [95]. The shooting method is a direct extension of the solution techniques for initial value problems with integration of optimization techniques. It is conceptually simple, and allows the usage of available solvers for ordinary differential equations. Moreover, the shooting method only stores the initial velocity during the

computation. For this reason, we choose the shooting method to solve the planning problem.

4.2 Planning Methods and Implementation

In this section, we describe the shooting method for the TPBV formulation of Eqs. (4.25) and (4.26). Three quasi-Newton based planning algorithms are developed. In Subsection 4.2.1, a basic quasi-Newton based shooting algorithm is first proposed to solve the TPBV problem. In order to reduce the computation cost on the Jacobian matrix in Newton method, a quasi-Newton based algorithms with the Broyden update is proposed in Subsection 4.2.2, where the Jacobian matrix is updated iteratively. Finally, a line search strategy based algorithm is proposed in Subsection 4.2.3.

4.2.1 Basic Shooting Method

In order to use the shooting method, we need to have the formulation of the initial value problem. The associated initial value problem for the TPBV formulation of Eqs. (4.25) and (4.26) is defined as

$$\begin{cases} \frac{dx_1}{d\tau} &= T_f \cdot f_1(\overline{\mathbf{x}}) \\ \frac{dx_2}{d\tau} &= T_f \cdot f_2(\overline{\mathbf{x}}) \\ & \vdots \\ \frac{dx_6}{d\tau} &= T_f \cdot f_6(\overline{\mathbf{x}}) \\ \frac{dT_f}{d\tau} &= 0 \end{cases} \tag{4.27}$$

with the initial conditions at $\tau = 0$,

$$\begin{cases} X_o(0) = X_i & \dot{X}_o(0) = \dot{X}_i \\ Y_o(0) = Y_i & \dot{Y}_o(0) = \dot{Y}_i \\ \theta_o(0) = \theta_i & \dot{\theta}_o(0) = \dot{\theta}_i \\ T_f(0) = T_{fi} \end{cases} . \tag{4.28}$$

We know from the boundary conditions (4.26) that, when the initial condition $(X_o(0), Y_o(0), \theta_o(0))$ is known, the solution for the configuration $(X_o(1), Y_o(1), \theta_o(1))$ and $v_f(1)$ at $\tau = 1$ are functions of initial velocity and travel time of the object in the initial value problem. Define a vector $\mathbf{s}$ as

$$\mathbf{s} = [\sigma_1, \sigma_2, \sigma_3, \sigma_4)^T = (\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0), T_f(0)]^T. \tag{4.29}$$

Solving the TPBV problem of Eqs. (4.25) and (4.26) is equivalent to finding a suitable vector $\mathbf{s} = (\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0), T_f(0))^T$ for the initial value problem of Eqs.(4.27) and (4.28), such that the solution of the initial value problem satisfies the boundary conditions (4.26) at $\tau = 1$ as

$$\begin{aligned} X_o(1, \mathbf{s}) &= X_f \\ Y_o(1, \mathbf{s}) &= Y_f \\ \theta_o(1, \mathbf{s}) &= \theta_f \\ v_f(1, \mathbf{s}) - \epsilon &= 0. \end{aligned} \tag{4.30}$$

We define Eq. (4.30) as a vector-valued function $\mathbf{G}(\cdot)$,

$$\mathbf{G}(\mathbf{s}) = \left[\begin{array}{cccc} X_o(1, \mathbf{s}) - X_f & Y_o(1, \mathbf{s}) - Y_f & \theta_o(1, \mathbf{s}) - \theta_f & v_f(1, \mathbf{s}) - \epsilon \end{array} \right]^T. \tag{4.31}$$

Now, solving the TPBV problem is equivalent to finding a solution of $\mathbf{s}$ as $\overline{\mathbf{s}} = (\overline{\sigma}_1, \overline{\sigma}_2, \overline{\sigma}_3, \overline{\sigma}_4)$ such that

$$\mathbf{G}(\overline{\mathbf{s}}) = 0. \tag{4.32}$$

The nonlinear equation (4.32) is solved by the Newton method

$$\mathbf{s}^{(i+1)} = \mathbf{s}^{(i)} - DG(\mathbf{s}^{(i)})^{-1} \cdot \mathbf{G}(\mathbf{s}^{(i)}) \tag{4.33}$$

where i indicates the i-th iteration.

In each iteration step, one has to compute $\mathbf{G}(\mathbf{s}^{(i)})$, and the Jacobian matrix

$$DG(\mathbf{s}^{(i)}) = \left[\frac{\partial \mathbf{G}_j}{\partial \mathbf{s}} \right]_{\mathbf{s}=\mathbf{s}^{(i)}} \tag{4.34}$$

and the solution $\mathbf{d}^{(i)} = \mathbf{s}^{(i)} - \mathbf{s}^{(i+1)}$ of the linear system of equations

$$DG(\mathbf{s}^{(i)})\mathbf{d}^{(i)} = \mathbf{G}(\mathbf{s}^{(i)}). \tag{4.35}$$

In order to compute $\mathbf{G}(\mathbf{s}^{(i)})$, one must solve the initial value problem of Eqs. (4.27) and (4.28) with $\mathbf{s} = \mathbf{s}^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}, \sigma_4^{(i)})$. Generally, the Jacobian matrix $DG(\mathbf{s}^{(i)})$ cannot be calculated analytically, it needs to be approximated numerically by the finite difference matrix $\Delta G(\mathbf{s}^{(i)})$ as

$$\Delta G(\mathbf{s}^{(i)}) = \left[\begin{array}{ccc} \mathbf{\Delta G}_1(\mathbf{s}^{(i)}) & \cdots & \mathbf{\Delta G}_4(\mathbf{s}^{(i)}) \end{array} \right] \tag{4.36}$$

where the jth entity of $\Delta G(\mathbf{s}^{(i)})$ is a column vector with four components, and it is computed as

$$\begin{array}{rl} \mathbf{\Delta G}_j(\mathbf{s}^{(i)}) = & \frac{1}{\Delta\sigma_j^{(i)}} \Big(\mathbf{G}(\sigma_1^{(i)}, \cdots, \sigma_j^{(i)} + \Delta\sigma_j^{(i)}, \cdots, \sigma_4^{(i)}) \\ & -\mathbf{G}(\sigma_1^{(i)}, \cdots, \sigma_j^{(i)}, \cdots, \sigma_4^{(i)}) \Big)^T, \quad j = 1, 2, 3, 4 \end{array} \tag{4.37}$$

where $\Delta\sigma_j^{(i)}$ is a small number.

The computation procedure for $\mathbf{G}(\sigma_1^{(i)}, \cdots, \sigma_j^{(i)} + \Delta\sigma_j^{(i)}, \cdots, \sigma_4^{(i)})$ in Eq. (4.37) is the same as the computation of $\mathbf{G}(\mathbf{s}^{(i)})$, and it requires solving the corresponding initial value problems of Eqs. (4.27)and (4.28) with initial conditions $\mathbf{s} = (\sigma_1^{(i)}, \cdots, \sigma_j^{(i)} + \Delta\sigma_j^{(i)}, \cdots, \sigma_4^{(i)})$. By introducing the approximated Jacobian matrix $\Delta G(\mathbf{s}^{(i)})$ into Eq. (4.33), we obtain the following quasi-Newton method,

$$\mathbf{s}^{(i+1)} = \mathbf{s}^{(i)} - \Delta G(\mathbf{s}^{(i)})^{-1} \cdot \mathbf{G}(\mathbf{s}^{(i)}). \tag{4.38}$$

The quasi-Newton method based planning algorithm is summarized as follows.

Choose a small number ϵ, *and starting vector* $\mathbf{s}^{(0)}$.

For $i = 0, 1, 2, \cdots$

Determine $X_o(1,\mathbf{s}^{(i)}), Y_o(1,\mathbf{s}^{(i)}), \theta_o(1,\mathbf{s}^{(i)}), v_f(1,\mathbf{s}^{(i)})$ *by solving the initial value problem of Eqs. (4.27) and (4.28), then compute* $\mathbf{G}(\mathbf{s}^{(i)})$ *according to Eq. (4.31).*

Choose small numbers $\Delta\sigma_j, j = 1,\cdots,4$, *and determine* $X_o(1,\mathbf{s}^{(i)}+\Delta\sigma_j\mathbf{e}_j), Y_o(1,\mathbf{s}^{(i)}+\Delta\sigma_j\mathbf{e}_j), \theta_o(1,\mathbf{s}^{(i)}+\Delta\sigma_j\mathbf{e}_j), v_f(1,\mathbf{s}^{(i)}+\Delta\sigma_j\mathbf{e}_j)$ *by solving four initial value problems of Eqs. (4.27)and (4.28) with*

$$\mathbf{s} = \mathbf{s}^{(i)} + \Delta\sigma_j\mathbf{e}_j = \left[\sigma_1^{(i)},\cdots,\sigma_j^{(i)}+\Delta\sigma_j,\cdots,\sigma_4^{(i)}\right]^T, \quad j = 1,\cdots,4 \tag{4.39}$$

where $\mathbf{e}_j$ *is a four-dimensional column vector with elements having the following form*

$$\mathbf{e}_j(l) = \begin{cases} 1 & l = j \\ 0 & l \neq j \end{cases} \qquad l = 1,\cdots,4. \tag{4.40}$$

First compute $\Delta G(\mathbf{s}^{(i)})$ *by means of Eqs. (4.36) and (4.37), then compute the solution* $\mathbf{d}^{(i)}$ *of the following system of linear equations*

$$\Delta G(\mathbf{s}^{(i)})\mathbf{d}^{(i)} = -\mathbf{G}(\mathbf{s}^{(i)}). \tag{4.41}$$

Update

$$\mathbf{s}^{(i+1)} = \mathbf{s}^{(i)} + \mathbf{d}^{(i)}. \tag{4.42}$$

end(for)

Algorithm 1: Basic shooting method

In each iteration of the algorithm, five initial value problems and a four-dimensional system of linear equations needs to be solved.

4.2.2 Shooting with the Broyden Update

In each iteration of the basic shooting method, four initial value problems are solved to compute the Jacobian matrix. The computation of initial value problems is always time consuming. To reduce the computational cost, we consider the use of the Broyden method [28] to approximate the Jacobian matrix. The local convergence of the Broyden method has been shown in [28]. In the Broyden method, the Jacobian matrix is updated iteratively in each iteration instead of being computed numerically using Eqs. (4.36) and (4.37). The planning algorithm with the Broyden update is as follows.

Choose a small number ϵ, *and starting vector* $\mathbf{s}^{(0)}$.
Determine $X_o(1,\mathbf{s}^{(0)}), Y_o(1,\mathbf{s}^{(0)}), \theta_o(1,\mathbf{s}^{(0)}), v_f(1,\mathbf{s}^{(0)})$ *by solving the initial value problem, then compute* $\mathbf{G}(\mathbf{s}^{(0)})$ *according to Eq. (4.31).*
Choose $\Delta\sigma_j, j = 1,\cdots,4$, *and determine*
$X_o(1,\mathbf{s}^{(0)}+\Delta\sigma_j\mathbf{e}_j), Y_o(1,\mathbf{s}^{(0)}+\Delta\sigma_j\mathbf{e}_j), \theta_o(1,\mathbf{s}^{(0)}+\Delta\sigma_j\mathbf{e}_j), v_f(1,\mathbf{s}^{(0)}+\Delta\sigma_j\mathbf{e}_j)$ *by solving four initial value problems with*

$$\mathbf{s}_j^{(0)} = \mathbf{s}^{(0)} + \Delta\sigma_j\mathbf{e}_j = \left[\sigma_1^{(0)},\cdots,\sigma_j^{(0)}+\Delta\sigma_j,\cdots,\sigma_4^{(0)}\right]^T, \quad j = 1,\cdots,4 \tag{4.43}$$

where $\mathbf{e}_j$ *is a four-dimensional column vector with elements having the following form*

$$\mathbf{e}_j(l) = \begin{cases} 1 & l = j \\ 0 & l \neq j \end{cases} \qquad l = 1,\cdots,4. \tag{4.44}$$

Compute $\Delta G(\mathbf{s}^{(0)})$ *by means of Eqs. (4.36) and (4.37).*
For $i = 1,2,\cdots,$

Compute the solution $\mathbf{d}^{(i)}$ *of the following system of linear equations*

$$\Delta G(\mathbf{s}^{(i)})\mathbf{d}^{(i)} = -\mathbf{G}(\mathbf{s}^{(i)}) \tag{4.45}$$

and update

$$\mathbf{s}^{(i+1)} = \mathbf{s}^{(i)} + \mathbf{d}^{(i)}. \tag{4.46}$$

Determine $X_o(1, \mathbf{s}^{(i+1)}), Y_o(1, \mathbf{s}^{(i+1)}), \theta_o(1, \mathbf{s}^{(i+1)}), v_f(1, \mathbf{s}^{(i+1)})$ *by solving the initial value problem of Eqs. (4.27) and (4.28).*

Compute $\mathbf{G}(\mathbf{s}^{(i+1)})$ *according to Eq. (4.31).*

Compute

$$\mathbf{y}^{(i)} = \mathbf{G}(\mathbf{s}^{(i+1)}) - \mathbf{G}(\mathbf{s}^{(i)}) \tag{4.47}$$

where $\mathbf{y}^{(i)}$ *is the difference of the vector function value* $\mathbf{G}(\cdot)$ *between iterations* $i+1$ *and* i*, which provides the information for the Jacobian update.*

Update the Jacobian matrix as

$$\Delta G(\mathbf{s}^{(i+1)}) = \Delta G(\mathbf{s}^{(i)}) + \frac{(\mathbf{y}^{(i)} - \Delta G(\mathbf{s}^{(i)}))\mathbf{d}^{(i)}(\mathbf{d}^{(i)})^T}{(\mathbf{d}^{(i)})^T\mathbf{d}^{(i)}}. \tag{4.48}$$

end(for)

Algorithm 2: Shooting method with the Broyden update

By using the Broyden method to update the Jacobian matrix, the planning algorithm only needs to solve one initial value problem in each iteration. As a result, the computation cost will be reduced dramatically.

4.2.3 Implementation with a Line Search Strategy

As known from literature [74], neither the quasi-Newton method nor the Broyden method guarantee the convergence to a solution unless the initial guess is close to the

solution. The quasi-Newton and the Broyden methods could be made more robust by using global strategies like line search or trust region techniques. Here, we choose a line search as the global strategy [28].

To implement a line search strategy, we define a *merit function*, which is a scalar-valued function of $\mathbf{s}^{(i)}$ whose value indicates whether a new candidate iterate is better or worse than the current iterate, in the sense of making progress toward the solution of $\bar{\mathbf{s}}$. Here, we choose the sum of squares as the merit function, defined by

$$f_m(\mathbf{s}^{(i)}) = \frac{1}{2}\|\mathbf{G}(\mathbf{s}^{(i)})\|^2 \tag{4.49}$$

where $\|.\|$ represents the norm.

The gradient of Eq. (4.49) is computed as

$$\nabla f_m(\mathbf{s}^{(i)}) = DG(\mathbf{s}^{(i)})^T \mathbf{G}(\mathbf{s}^{(i)}) \tag{4.50}$$

where $DG(s^{(i)})$ is the Jacobian matrix defined by Eq. (4.34).

In the quasi-Newton algorithm, the Jacobian matrix is approximated and updated at each iteration according to Eq. (4.36). Before developing the global strategy, we define the Hessian matrix as

$$H(\mathbf{s}^{(i)}) = DG(\mathbf{s}^{(i)})^T DG(\mathbf{s}^{(i)}). \tag{4.51}$$

In each iteration, a line search method computes a search direction $\mathbf{p}_i$, and then decides how far to move along that direction. The iteration is given by

$$\mathbf{s}^{(i+1)} = \mathbf{s}^{(i)} + \alpha_i \mathbf{p}_i \tag{4.52}$$

where the positive scalar α_i is called the step length. The success of a line search method depends on the effective choices of both the direction $\mathbf{p}_i$ and the step length

α_i. In Newton method, the descent direction is chosen as

$$\mathbf{p}_i = -H(\mathbf{s}^{(i)})^{-1} \nabla \mathbf{f}_m(\mathbf{s}^{(i)}). \tag{4.53}$$

When it is well defined, the Newton step satisfies

$$\frac{-\mathbf{p}_i^T \nabla \mathbf{f}_m(\mathbf{s}^{(i)})}{\|\mathbf{p}_i^T\| \|\nabla \mathbf{f}_m(\mathbf{s}^{(i)})\|} \geq \delta, \text{for some } \delta \in (0,1) \text{ and all } i \text{ sufficiently large.} \tag{4.54}$$

To ensure that the condition (4.54) holds, we may need to modify the Newton descent direction (4.53). One possibility is to add some multiple $\gamma_i I$ of the identity matrix to $H(\mathbf{s}^{(i)})$, and define a new descent direction $\mathbf{p}_i$ to be

$$\mathbf{p}_i = -(H(\mathbf{s}^{(i)}) + \gamma_i I)^{-1} \nabla \mathbf{f}_m(\mathbf{s}^{(i)}). \tag{4.55}$$

For each iteration, we choose γ_i such that the condition (4.54) is satisfied for some given value of $\delta \in (0,1)$. The Choleskey factorization of the matrix $(H(\mathbf{s}^{(i)}) + \gamma_i I)$ is used to find the γ_i [74].

After we have identified the search direction $\mathbf{p}_i$ for Eq. (4.55), we need to choose the step length α_i, which is chosen such that the following Wolfe condition is satisfied,

$$f_m(\mathbf{s}^{(i)} + \alpha_i \mathbf{p}_i) \leq f_m(\mathbf{s}^{(i)}) + c_1 \alpha_i \nabla \mathbf{f}_m(\mathbf{s}^{(i)})^T \mathbf{p}_i. \tag{4.56}$$

This condition stipulates that α_i should give sufficient decrease of the objective function f_m. In other words, the reduction in f_m should be proportional to both step length α_i and the directional derivative $\nabla \mathbf{f}_m(\mathbf{s}^{(i)})^T \mathbf{p}_i$.

The shooting method with a line search is summarized as follows.

Given $\delta \in (0,1)$ and c_1, c_2 with $0 < c_2 < c_2 < \frac{1}{2}$, choose a starting vector $\mathbf{s}^{(0)}$. Determine $X_o(1, \mathbf{s}^{(0)}), Y_o(1, \mathbf{s}^{(0)}), \theta_o(1, \mathbf{s}^{(0)}), v_f(1, \mathbf{s}^{(0)})$ by solving the initial value problem, then compute $\mathbf{G}(\mathbf{s}^{(0)})$ according to Eq. (4.31).

Choose $\Delta\sigma_j, j = 1, \cdots, 4$, *and determine*
$X_o(1, \mathbf{s}^{(0)} + \Delta\sigma_j \mathbf{e}_j), Y_o(1, \mathbf{s}^{(0)} + \Delta\sigma_j \mathbf{e}_j), \theta_o(1, \mathbf{s}^{(0)} + \Delta\sigma_j \mathbf{e}_j), v_f(1, \mathbf{s}^{(0)} + \Delta\sigma_j \mathbf{e}_j)$ *by solving four initial value problems with*

$$\mathbf{s}_j^{(0)} = \mathbf{s}^{(0)} + \Delta\sigma_j \mathbf{e}_j = \left[\sigma_1^{(0)}, \cdots, \sigma_j^{(0)} + \Delta\sigma_j, \cdots, \sigma_4^{(0)}\right]^T, \quad j = 1, \cdots, 4 \tag{4.57}$$

where $\mathbf{e}_j$ *is a four-dimensional column vector with elements having the following form*

$$\mathbf{e}_j(i) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \qquad l = 1, \cdots, 4. \tag{4.58}$$

Compute the approximated Jacobian matrix $\Delta G(s^{(0)})$ *by means of Eqs. (4.36) and (4.37).*

for $i = 1, 2, \cdots,$

Compute the Newton step $\mathbf{p}_i$ *according to Eq. (4.53).*

if *Newton step (4.53) satisfies condition (4.54)*

Set $\mathbf{p}_i$ *to the Newton step.*

else

Obtain $\mathbf{p}_i$ *from Eq. (4.55), choosing* γ_i *to ensure that Eq. (4.54) holds.*

end(if)

if $\alpha = 1$ *satisfies the Wolfe condition (4.56)*

set $\alpha_i = 1$.

else

Perform a line search to find $\alpha_i > 0$ *that satisfies condition (4.56).*

end(if)

Update

$$\mathbf{s}^{(i+1)} = \mathbf{s}^{(i)} + \alpha_i \mathbf{p}_i. \tag{4.59}$$

Determine $X_o(1, \mathbf{s}^{(i+1)}), Y_o(1, \mathbf{s}^{(i+1)}), \theta_o(1, \mathbf{s}^{(i+1)}), v_f(1, \mathbf{s}^{(i+1)})$ *by solving the initial value problem of Eqs. (4.27) and (4.28), then compute* $\mathbf{G}(\mathbf{s}^{(i+1)})$ *according to Eq. (4.31).*

Compute

$$\mathbf{y}^{(i)} = \mathbf{G}(\mathbf{s}^{(i+1)}) - \mathbf{G}(\mathbf{s}^{(i)}) \tag{4.60}$$

where $\mathbf{y}^{(i)}$ *is the difference of the function value* $\mathbf{G}(\cdot)$ *between iterations* $i+1$ *and* i*, which provides the information for the Jacobian update.*

Compute

$$\mathbf{d}^{(i)} = \mathbf{s}^{(i+1)} - \mathbf{s}^{(i)}. \tag{4.61}$$

Update the Jacobian matrix

$$\Delta G(\mathbf{s}^{(i+1)}) = \Delta G(\mathbf{s}^{(i)}) + \frac{(\mathbf{y}^{(i)} - \Delta G(\mathbf{s}^{(i)}))\mathbf{d}^{(i)}(\mathbf{d}^{(i)})^T}{(\mathbf{d}^{(i)})^T \mathbf{d}^{(i)}}. \tag{4.62}$$

end(for)

Algorithm 3: Practical algorithm with line search

The optimization procedures used in the planning algorithms belong to the class of local optimization methods, which exhibit a local convergence. The choice of initial guesses for $\mathbf{s}^{(0)}$ will affect the convergence of the algorithms.

It has been shown that with the same initial velocity, the travel time of an object in a composite motion is larger than that in the cases of pure translation or rotation [102].

The displacement also satisfies this relationship. As a result, for a given displacement, the object in a composite motion needs a smaller initial velocity.

If the object starts with a pure translation, for given displacements X_f, Y_f, the required initial linear velocities $\dot{X}_t(0), \dot{Y}_t(0)$ are calculated as

$$\begin{aligned} \dot{X}_t(0) &= \sqrt{2\mu g X_f} \\ \dot{Y}_t(0) &= \sqrt{2\mu g Y_f}. \end{aligned} \tag{4.63}$$

If the object undergoes a pure rotation, for a given θ_f, the initial rotational velocity $\dot{\theta}_r(0)$ is calculated as

$$\dot{\theta}_r(0) = \sqrt{2\mu g \theta_f \int\int (\sqrt{x^2+y^2}\rho(x,y))dxdy/I_0}. \tag{4.64}$$

These velocities give the bounds of the initial release velocity for a free sliding object. Using these bounds, the initial guesses $\dot{X}_{og}(0), \dot{Y}_{og}(0), \dot{\theta}_g(0)$ for velocity $\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0)$ are chosen as

$$\begin{aligned} 0 < \dot{X}_{og}(0) &\leq \dot{X}_t(0) \\ 0 < \dot{Y}_{og}(0) &\leq \dot{Y}_t(0) \\ 0 < \dot{\theta}_g(0) &\leq \dot{\theta}_r(0). \end{aligned} \tag{4.65}$$

The initial guess for T_f is obtained by solving the initial value problem (4.19) with initial conditions $\dot{X}_{og}(0), \dot{Y}_{og}(0), \dot{\theta}_g(0)$.

Concerning the existence and uniqueness of the solution, there have been some results on the solution uniqueness of TPBV problems [43]. Since the friction forces F_X, F_Y and torque T depend on the geometry of the contact area and pressure distribution, they cannot be integrated explicitly in general. Therefore, we could not apply the results in [43] to verify the uniqueness of the solution. The uniqueness associated with the free boundary value problem itself is a very interesting problem to investigate. In practice, the uniqueness of the solution could be verified by applying

different initial guesses to the planning algorithms and, if all of the planned initial velocities converge, it indicates the solution is a unique one. Moreover, as pointed out in [37], if we consider the periodicity of the orientation of the object, the solution will not be unique. For example, the following set of boundary conditions with orientation $\theta_o(T_f) = \theta_f \pm 2i\pi \quad (i = 1, 2, \cdots)$ will define the same orientation of the object. Therefore, a set of initial velocities could be obtained by using the proposed planning algorithms. This property actually provides us with more flexibility to choose a feasible solution suitable for a specific manipulation device.

4.3 Numerical and Experimental Results

In this section, we present numerical and experimental results that demonstrate the performance of the proposed planning methods.

4.3.1 Numerical Simulations

For simulation purposes, six different parts as illustrated in Fig. 4.2 are selected. The first three parts are polygons with different numbers of vertices (labeled as triangle, rectangle, and pentagon), all having uniform pressure distributions. The fourth part is a curved object (labeled as curved) that is considered in order to verify the applicability of the planning method for non-polygonal objects. The other two parts (labeled as step and slope) are workpieces with nonuniform pressure distributions. The geometric dimensions of the six parts are listed in Fig. 4.2. Later on, in the experimental phase of this study, aluminum parts are utilized with the same geometries. Thus, for consistency, in the simulation, the part properties are calculated using the density of aluminum. The mass density of aluminum is 2700Kg/m^3, and the mass distribution function $\rho(x, y) = 2700 \cdot z$, where z is the thickness of the part at position (x, y).

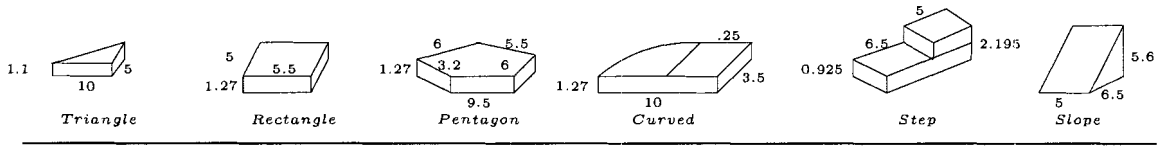


Figure 4.2: Geometries of parts (unit: cm)

In this study, we choose the initial configuration as $(0, 0, 0)$, and we use the practical algorithm with a line search to find the desired release velocity. The performance of the proposed planning schemes are summarized in Table 4.1. In this table, we show the initial guesses for $\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0)$, the planned initial velocities, the goal displacements, and the achieved displacements of the objects with the planned initial velocities. The number of iterations required to find the desired velocities are also listed in the table. The results indicate that with suitable initial guesses, the planning method will find the desired initial velocities for the free sliding objects.

As an example, to show the convergence of the algorithm, the velocities and final configurations of a uniform rectangular object are shown in Fig. 4.3. The initial configuration and the goal configuration are the same as those listed in Table 4.1. In Fig. 4.3(a), the number on each block indicates the position of the object after the number of iterations, and G on the block indicates the goal configuration. It is clear that, after seven iterations, the planned configuration converges to the desired goal configuration. In addition, the corresponding initial velocities $\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0)$ for each iteration found by the planning algorithm are also shown in Fig. 4.3(b)-(d) respectively.

To investigate the influence of initial guesses on the convergence of the planning method, we supply the planning algorithm with different initial guesses

Table 4.1: Numerical results for objects with different geometries. The rows marked with G are the initial guess for the velocities. The rows marked with P are the planned velocities. The displacements marked with Go are the desired goal displacements, and the displacements marked with A are the achieved displacements with planned velocities

Parts Geometry	G & P	Velocities $\dot{X}_o(0)$ (m/s)	$\dot{Y}_o(0)$ (m/s)	$\dot{\theta}(0)$ (rad/s)	Go & A	Displacements X_f (m)	Y_f (m)	θ_f (rad)	No. of Iterations
Triangle	G	0.438	0.33	3.1	Go	0.158	0.09	1.190	19
	P	0.542	0.312	3.448	A	0.159	0.091	1.19	
Rectangle	G	0.525	0.40	4.32	Go	0.228	0.131	1.786	7
	P	0.646	0.372	3.90	A	0.225	0.130	1.781	
Pentagon	G	0.599	0.299	3.95	G	0.206	0.074	1.587	4
	P	0.6268	0.2297	3.9594	A	0.198	0.072	1.586	
Curved	G	0.528	0.263	4.35	Go	0.16	0.057	1.6572	5
	P	0.569	0.208	4.34	A	0.164	0.059	1.65	
Step	G	0.55	0.214	3.5	Go	0.236	0.085	1.736	36
	P	0.685	0.251	3.96	A	0.2357	0.0853	1.7341	
Slope	G	0.45	0.258	2.99	Go	0.162	0.086	1.326	31
	P	0.553	0.294	3.43	A	0.1617	0.0859	1.3258	

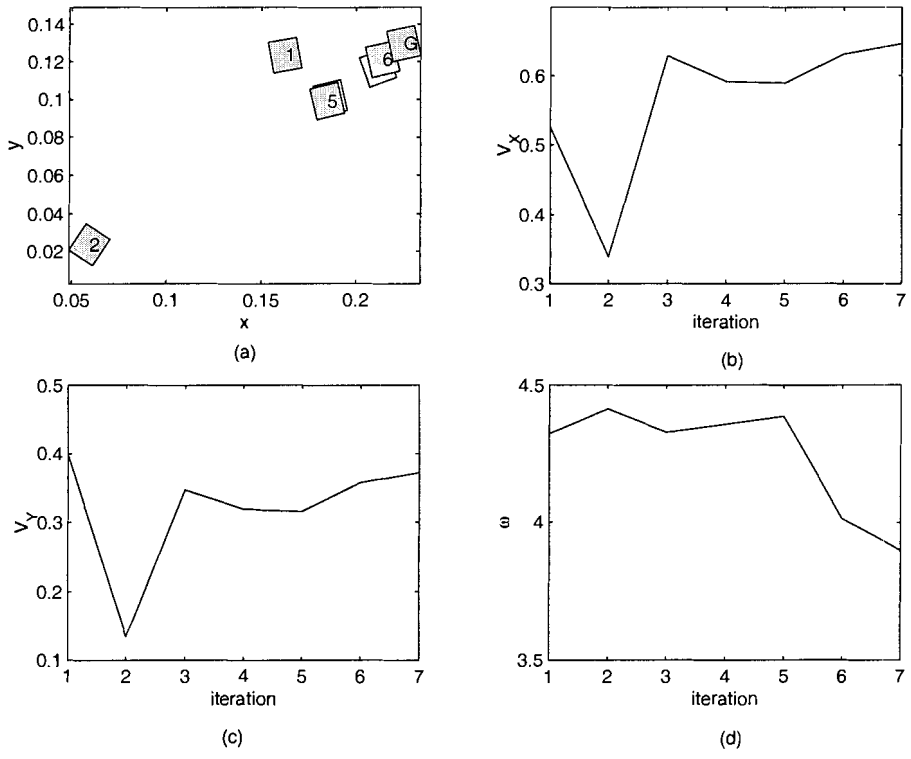


Figure 4.3: Iteration results for the rectangular object: (a) shows the final configuration after each iteration for a uniform rectangular object; (b)-(d) show the planned initial velocity $\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0)$ after each iteration.

$\dot{X}_{og}(0), \dot{Y}_{og}(0), \dot{\theta}_g(0)$ to the curved part (Fig. 4.2). Since the integration scheme proposed in Appendix A is developed for polygonal objects, it cannot be applied to the curved object directly. In order to use the integration method, we introduce middle points on the arc to approximate the curved part by a polygon. The accuracy of the approximation depends on the number of points used. For simplicity, only one point is assigned in the middle of the arc and the object is represented as a pentagon. By using the integration schemes proposed in appendix A, we get $m = 0.111$ Kg, and $I_0 = 8.271e-5$ Kg·m^2, the friction coefficient $\mu = 0.11$, and $g = 9.8$ m/s^2. The goal configuration is assigned as $[X_f \quad Y_f \quad \theta_f]^T = [0.16 \quad 0.057 \quad 1.6572]^T$. From Eq. (4.65), we know that the initial guesses of $\dot{X}_{og}(0), \dot{Y}_{og}(0), \dot{\theta}_g(0)$ should be chosen as

$$\begin{aligned} 0 < \dot{X}_{og}(0) &\leq 0.5873 \\ 0 < \dot{Y}_{og}(0) &\leq 0.3506 \\ 0 < \dot{\theta}_g(0) &\leq 10.885. \end{aligned} \tag{4.66}$$

The convergence results for this curved part are listed in Table 4.2 with different initial guesses, planned velocities, and the resulting displacements. The results indicate that the initial guesses will affect the convergence speed of the algorithm. In all cases, the planning algorithm converges and obtains the desired initial velocity for the free sliding object. However, the number of iterations vary from 3 to 184, depending on the initial conditions. Although cases where the planning methods do not converge may exist, no scenario was found in this exploration.

4.3.2 Experimental Setup

To verify the planning results of the proposed algorithm using a physical setup, we integrated a manipulation setup consisting of a fence connected to a direct drive actuator which creates 1-DoF controlled rotational motion. Similar manipulators

Table 4.2: Convergence results under different initial guess for curved part

No.	Guess			Velocities			Displacements			Iter.
	$\dot{X}_{og}(0)$ (m/s)	$\dot{Y}_{og}(0)$ (m/s)	$\dot{\theta}_g(0)$ (rad/s)	$\dot{X}_o$ (m/s)	$\dot{Y}_o$ (m/s)	$\dot{\theta}$ (rad/s)	X_f (m)	Y_f (m)	θ_f (m)	
1	0.528	0.28	4.8	0.572	0.206	4.332	0.1653	0.059	1.675	9
2	0.528	0.263	4.35	0.569	0.208	4.340	0.164	0.059	1.650	5
3	0.528	0.28	4.5	0.549	0.202	4.503	0.153	0.055	1.655	3
4	0.55	0.3	4	0.57	0.206	4.352	0.1638	0.0581	1.654	9
5	0.4	0.3	4	0.556	0.199	4.477	0.156	0.055	1.662	38
6	0.2	0.35	3	0.558	0.222	4.423	0.159	0.062	1.656	184
7	0.1	0.1	8	0.563	0.203	4.416	0.160	0.057	1.660	83
8	0.2	0.15	8	0.563	0.204	4.410	0.160	0.057	1.657	39

have been used to perform parts feeding over a conveyor [6, 91], where the fences were used to manipulate parts over a conveyer belt for quasi-static manipulation purpose.

A picture of our experimental setup is shown in Fig. 4.4, where a fence of 50 Cm in length is driven by an actuator. The fence is used to accelerate the object to desired velocities, which can then be used to verify the proposed planning methods. A PD controller is implemented for the position and speed control of the actuator, and an encoder with resolution of 2000 counts per revolution is attached to the actuator for measuring the position of the actuator. A Galio $DMC - 18 \times 2$ series motion control card is used to interface between the encoder and the PC as well as between the output control signal and the motor amplifier.

The schematic of the 1-DoF dynamic manipulation of an object is shown in Fig. 4.5. The actuator is placed at O, which is the center of a fixed reference frame, and the fence is fixed to this point and rotates about it. We also introduce a movable stopper, which is a V-shaped block that can be placed anywhere on the fence similar to the fixed stopper used in [6]. The stopper is clipped onto the fence at position

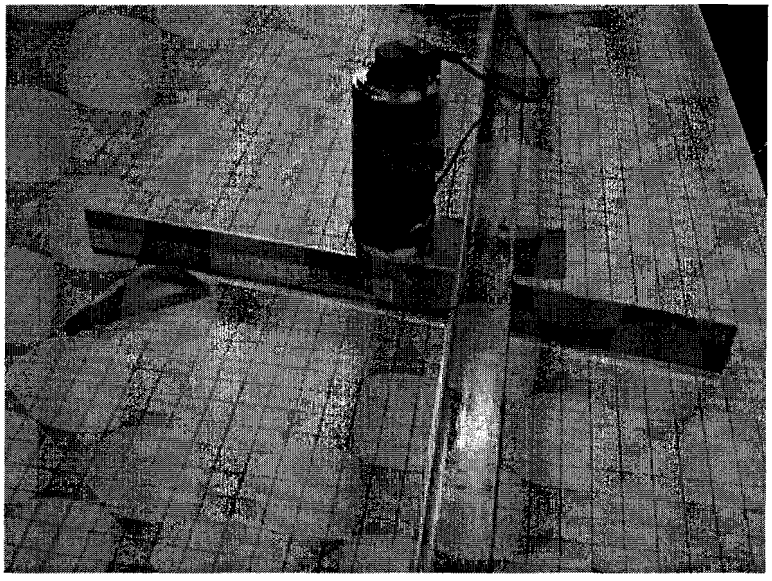


Figure 4.4: Experiment setup

L. There are two purposes for using the stopper. First, the stopper can oppose the centrifugal force acting on the objects during rotation. Thus, it constrains the motion of the object along the fence. Second, by varying the position of the stopper, the linear velocity of the object could be changed independently of the angular velocity.

At the beginning of each experiment, the object is placed against the fence and the stopper. The center of mass of the object is represented by o. The motor drives the fence rotating with constant speed from some initial position. After rotating a predefined angle θ_1, the fence will stop, and the object will be thrown out.

In order to validate the proposed planning method, we need to accelerate the object to the desired velocity $[\dot{X}_o \quad \dot{Y}_o \quad \dot{\theta}]^T$ using the rotational motion of the fence. Since we can regulate the rotation angle and velocity $\dot{\theta}$ separately, we can also implicitly control the linear velocity using the following relation,

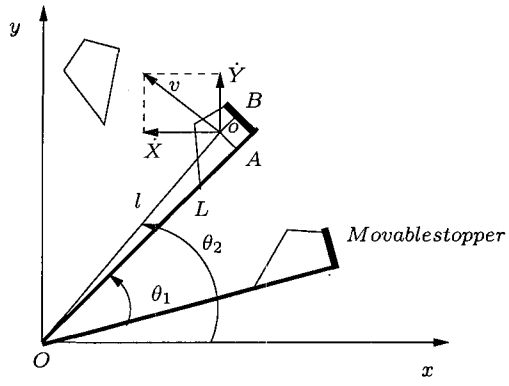


Figure 4.5: Description of experiment parameters

$$v = \dot{\theta} \cdot l \tag{4.67}$$

where l is the distance from the center of mass of the object o to the origin of the world reference frame XOY.

This relationship (4.67) suggests that the linear release velocity can be controlled by varying the position of the stopper relative to the origin O, namely, the variable L, and it is computed as

$$L = \sqrt{l^2 - oA^2} + oB \tag{4.68}$$

where oA is the distance between o and the fence, oB is the distance between o and the stopper.

The components of the linear velocity can then be calculated as

$$\dot{X} = v \cdot \sin\theta_2 \qquad \dot{Y} = v \cdot \cos\theta_2 \tag{4.69}$$

where θ_2 is the release angle.

The purpose of the experiment is to verify the simulation results in the previous section. For a given velocity $(\dot{X}_o(0), \dot{Y}_o(0), \dot{\theta}(0))$ found by the planner, in order to

achieve the same release velocity for the object, the parameters for the experiment are set up according to the following steps. (refer to Fig. 4.5)

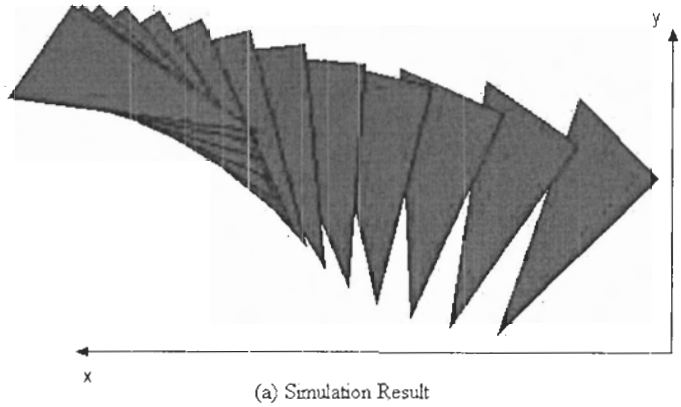
1. Set the speed ω of the actuator to $\dot{\theta}(0)$.
2. Calculate the linear velocity $v = \sqrt{\dot{X}^2(0) + \dot{Y}^2(0)}$, then compute l from the relation (4.67); finally, determine L based on Eq. (4.68).
3. Find the final angle between the fence and the x-axis θ_2 as $\theta_2 = \arcsin(\dot{X}(0)/v)$ from Eq. (4.69).
4. Choose a suitable rotation angle θ_1 for accelerating the object.

4.3.3 Experimental Results

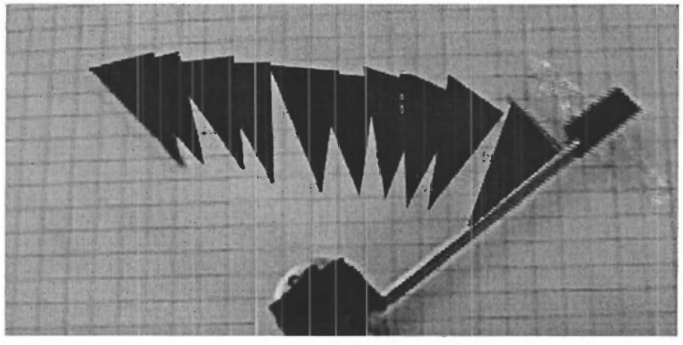
In the experiments, a high-density fiberboard with uniform grids of $2cm$ is used as the supporting plane. The motion of the object is captured by a digital video camera. The experiments are carried out on aluminum parts with geometries as depicted in Fig. 4.2. In the experiments, the initial velocities for the manipulation are the same as those listed in Table 4.1. The parameter ω is set to $\dot{\theta}(0)$, L is calculated according to the procedure provided in Subsection 4.3.2, and $\theta_1 = \pi/2$ is the rotation angle of the fence from start to stop.

Figure 4.6 shows a qualitative comparison of numerical and experimental results for a triangular object. The positions of the moving object are obtained by extracting frames from the captured video. Repeated trials are performed on different parts, and the quantitative experimental results are listed in Table 4.3. These results indicate the effectiveness of the planning and manipulation method.

The experiments using the 1-DoF manipulator give us some reasonable results to verify the proposed planning algorithms. The difference between the numerical and



(a) Simulation Result



(b) Experimental Result

Figure 4.6: Results of qualitative comparison

Table 4.3: Experimental results (The objects gain the initial velocities $\dot{X}_o, \dot{Y}_o, \dot{\theta}$, and the resulting displacements X_f, Y_f, θ_f, which are close to the goal displacements.)

Geometry	Velocities			Goals			Experiments		
	$\dot{X}_o$ (m)	$\dot{Y}_o$ (m)	$\dot{\theta}$ (rad/s)	X_f (m)	Y_f (m)	θ_f (deg)	X_f (m)	Y_f (m)	θ_f (deg)
Triangle	0.542	0.312	3.45	0.159	0.091	68	0.19	0.07	60
Rectangle	0.646	0.372	3.90	0.225	0.130	101	0.238	0.115	88
Pentagon	0.627	0.23	3.96	0.198	0.072	91	0.232	0.065	75
Curved	0.569	0.208	4.34	0.164	0.059	95	0.205	0.066	75
Step	0.685	0.251	3.96	0.236	0.085	99	0.285	0.08	65
Slope	0.544	0.290	3.43	0.156	0.086	76	0.191	0.08	64

experimental results mainly comes from the following two sources, namely, modeling and experimental setup. In the modeling phase, we assume that the object makes a uniform contact with the supporting surface. In the experiments, the surface of the object and the supporting plane may not be flat enough to satisfy this assumption. The object may contact the supporting surface through a number of concentrated points [70]. The other possible reason associated with the model is the Coulomb friction assumption, which makes no distinction between static and dynamic friction coefficients. The friction coefficient used in the model is independent of the part's velocity. In the experiments, the friction coefficient between the sliding object and the supporting surface may depend on the velocity of the object and the temperature.

In the experimental setup, a stopper has been used to cancel the effect of the centrifugal force on the object. However, when the object starts to slip off the tip of the stopper, the centrifugal force will act on the object. As a result, an angular acceleration on the object opposite to the direction of rotation is generated. This negative angular acceleration will reduce the release angular velocity of the object and consequently, the angular displacement of the object. This is one possible reason that, in the experiments, the angular displacements are always smaller than those expected in numerical simulations. Another source of discrepancy is associated with the resulting position overshoot of the fence. The overshoot leads to the variation of the linear velocity distribution in the X and Y axes. As a result, it affects the corresponding displacements.

4.4 Discussion

In this chapter, we studied the planning problem for free sliding objects and proposed an optimization based method to solve the initial velocity for a given displacement.

Quantitative information is used to develop the planning algorithms. Compared to the work of [37] and [41], no motion characteristics, such as monotonicity of force and torque, are required. Thus, this method is suitable for objects with general geometrical shapes and mass distributions. The experiments provide reasonable results to demonstrate the correctness of the model and the planning algorithm. If there exist uncertainties in the friction model and part shape, the planned velocities will not generate the required displacement. The discrepancy between simulation and experimental results inspires us to develop feedback and closed-loop manipulation, which is the third stage of the dynamic cooperative manipulation system depicted in Fig. 1.2.

Chapter 5

Acceleration of the Object by Cooperative Dynamic Pushing

The previous chapter studied the free sliding problem to find the initial release velocity for a required displacement. This chapter studies the acceleration problem to achieve the planed release velocity. Cooperative dynamic pushing is used for this task. During the acceleration process, the robotic agents push the object to follow a predefined trajectory. We are interested in coordinating the pushing actions and force distribution between agents such that the pushed object follows the trajectory.

In the model of cooperative dynamic pushing, the forces exerted by agents need to obey some constraints. First, in order to avoid sliding between the agents and the object, the pushing force vector must lie inside the friction cone of the contact surface. Second, for a pushing action, the pushing force vector can only be directed toward the object. Because of these constraints, it becomes difficult to solve the planning problem directly using nonlinear control design techniques. Instead, by introducing the generalized force(wrench) into the model, we break the overall planning problem

hierarchically into a control design subproblem and a multi-agent coordination subproblem. The concept of generalized force have been used by Toussaint et al. [100] to study the trajectory tracking of a nonlinear under-actuated surface vessel.

For the control design problem, the objective is to find the generalized force(wrench) applied to the object. With the generalized force acting on it, the object will follow the given trajectory, and attain the desired velocity. The generalized force characterizes the required force and torque for a certain motion of the object. It is used for the design purpose. Since the model of the proposed pushing system possesses exactly the structure required for the integrator backstepping design [46], we consider using the backstepping technique to solve the controller design problem and to find the generalized force inputs to the object.

For the multi-agent coordination problem, the objective is to distribute the generalized forces between agents to optimize a predefined performance index. This problem is closely related to the force optimization problem in grasp analysis, where the objective is to solve for optimal contact forces yielding a stable grasp [23]. The formulation of the coordination problem depends on the choice of the performance index. By choosing a quadratic function of the force vectors as the performance index, the coordination between the agents is formulated as a quadratic programming (QP) problem. Through solving the QP problem, the optimal pushing forces applied on the object are found.

The remainder of the chapter is organized as follows: Section 5.1 presents the model of the cooperative dynamic pushing process, and outlines the planning problem. Section 5.2 studies the coordination and planning for the cooperative dynamic pushing. Section 5.3 presents the simulation results. Section 5.4 concludes this chapter with a discussion.

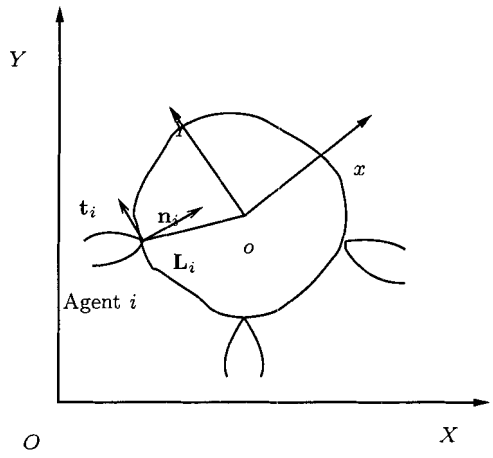


Figure 5.1: Cooperative dynamic pushing

5.1 Problem Formulation

5.1.1 Model of Cooperative Dynamic Pushing

Through the analysis of the motion of the object under pushing, the cooperative dynamic pushing process is formulated as a nonlinear system. Consider an object under cooperative dynamic pushing as shown in Fig. 5.1. Assign XOY as the world frame, xoy is the local coordinate frame associated with the object, and o is located at the center of mass (COM). The configuration of the object is described as (X_o, Y_o, θ), where (X_o, Y_o) is the position of the local coordinate frame, and θ is the orientation of the object. For each contact point, we introduce a contact frame with its $\mathbf{n}$-axis aligned with the contact normal (points inward with respect to the object), and its $\mathbf{t}$-axis aligned with the contact tangent such that the cross product of $\mathbf{n}_i$ and $\mathbf{t}_i$ points out of the plane. The directional vectors $\mathbf{n}_i$ and $\mathbf{t}_i$ are defined in the local coordinate frame. We introduce vectors $\mathbf{q}_1 = (X_o, Y_o, \theta)^T$, and $\mathbf{q}_2 = (\dot{X}_o, \dot{Y}_o.\dot{\theta})^T$ that describe the configuration and velocity of the object. The state of the system are defined as

$\mathbf{q} = (\mathbf{q}_1^T, \mathbf{q}_2^T)^T$. The cooperative dynamic pushing is modeled as a nonlinear system

$$\begin{aligned} \dot{\mathbf{q}}_1 &= \mathbf{q}_2 \\ \dot{\mathbf{q}}_2 &= \mathbf{f}(\mathbf{q}) + M_t \sum_{i=1}^{N} R \cdot W_i \begin{bmatrix} F_i^n \\ F_i^t \end{bmatrix} \end{aligned} \tag{5.1}$$

where $\mathbf{f}(\mathbf{q}) = (F_X/m, F_Y/m, T/I_0)^T$, where F_X, F_Y are the friction forces in the world frame, and T is the torque with respect to the COM generated by the friction force. They are computed in the same manner as that in Chapter 4. m is mass of the object, and I_0 is the mass moment of inertia about the COM.

$$M_t = \begin{bmatrix} \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{I_0} \end{bmatrix}$$

and the rotational transformation matrix is

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$F_i^n, F_i^t (i = 1, \cdots, N)$ are the normal and tangential components of the pushing force applied by agent i in the contact frame, and N is the total number of agents. W_i is the wrench matrix for agent i, which is used to transform the contact force F_i^n, F_i^t to forces and torque in the local coordinate frame xoy. The wrench matrix W_i is computed as [101]

$$W_i = [\mathbf{w}_{in} \quad \mathbf{w}_{it}] = \begin{bmatrix} \mathbf{n}_i & \mathbf{t}_i \\ \mathbf{L}_i \times \mathbf{n}_i & \mathbf{L}_i \times \mathbf{t}_i \end{bmatrix}, \quad i = 1, 2, \cdots, N \tag{5.2}$$

where $\mathbf{L}_i$ is the position vector of the i-th contact point in the local coordinate frame, and $\times$ is the cross product between two vectors.

In order to prevent the slippage between the agents and the object, the applied pushing forces to the object must be constrained to lie inside the friction cones. These constraints are written as

$$|F_i^t| \leq \mu_i F_i^n, \quad i = 1, 2, \cdots, N \tag{5.3}$$

where μ_i is the friction coefficient between the agent i and the object at the i-th contact point.

Since we consider pushing instead of pulling of the object, the contact normal force needs to be nonnegative, *i.e.*

$$F_i^n \geq 0, \quad i = 1, 2, \cdots, N \tag{5.4}$$

5.1.2 The Cooperative Push Planning Problem

With the system (5.1), the cooperative dynamic pushing problem is described as follows: Given an object on a plane at a known initial configuration, find the input forces applied by the agents, such that the object follows a given trajectory to the goal configuration with the specified velocity. The two-level planning problem is formulated as:

Given a desired trajectory $\mathbf{q}_d(t) = (X_{or}(t), Y_{or}(t), \theta_r(t))$, the corresponding velocity $\dot{\mathbf{q}}_d(t) = (\dot{X}_{or}(t), \dot{Y}_{or}(t), \dot{\theta}_r(t))$, and acceleration $\ddot{\mathbf{q}}_d(t) = (\ddot{X}_{or}(t), \dot{Y}_{or}(t), \ddot{\theta}_r(t))$,

design the state feedback control input $\mathbf{u}(t) = (F_1^n, F_1^t, F_2^n, F_2^t, \cdots, F_N^n, F_N^t)^T$ such that the state of the system (5.1) follows the desired trajectory, and minimize the quadratic objective function $M(\mathbf{u})$

$$M(\mathbf{u})(t) = \frac{1}{2}\mathbf{u}^T(t)Q\mathbf{u}(t) \tag{5.5}$$

where $Q > 0$ is a positive definite weight matrix.

In order to minimize the objective function $M(\mathbf{u})$, the pushing agents need to coordinate with each other, and a two-level centralized planner is developed for this purpose.

5.2 The Two-level Planning Method

In this section, we will solve the coordination problem and compute the contact forces F_{in} and F_{it} generated by each agent such that the object follows an arbitrary trajectory and obtains the desired velocity.

The object under pushing has three degrees of freedom, and generally it requires three independent control inputs to fully control the object to follow any given trajectory. By introducing the generalized force control inputs $\overline{\mathbf{u}} = (\overline{u}_1, \overline{u}_2, \overline{u}_3)^T$ to Eq. (5.1), the system is written as

$$\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{q}_2 \\ \mathbf{f}(\mathbf{q}) \end{bmatrix} + M_t \cdot \begin{bmatrix} \mathbf{0} \\ \overline{\mathbf{u}} \end{bmatrix} \tag{5.6}$$

and with these control inputs $\bar{\mathbf{u}}$, the system is fully actuated.

From Eq. (5.1), we know that the generalized force control inputs $\overline{\mathbf{u}}$ and original force inputs satisfy the following relationship

$$\overline{\mathbf{u}} = \sum_{i=1}^{N} R \cdot W_i \cdot \begin{bmatrix} F_i^n \\ F_i^t \end{bmatrix}. \tag{5.7}$$

By introducing $\overline{\mathbf{u}}$, the planning problem is solved hierarchically in two levels. First, the generalized force control inputs for system (5.6) are designed using the backstepping design technique. Second, the coordination is performed between the agents to distribute the generalized force $\overline{\mathbf{u}}$. Quadratic programming is used as a tool to solve the distribution problem.

5.2.1 Backstepping Design for the Generalized Force Controller

We consider using the backstepping design technique to design the generalized force controller. We define the tracking error variable $\tilde{\mathbf{q}}_1$ as the difference between the

actual and the desired configuration vectors

$$\tilde{\mathbf{q}}_1 = \mathbf{q}_1 - \mathbf{q}_d \tag{5.8}$$

and we take the derivative of $\tilde{\mathbf{q}}_1$ to get

$$\dot{\tilde{\mathbf{q}}}_1 = \dot{\mathbf{q}}_1 - \dot{\mathbf{q}}_d \tag{5.9}$$

We use error variable $\tilde{\mathbf{q}}_1$ to define a candidate Lyapunov function V_1 as

$$V_1 = \frac{1}{2}\tilde{\mathbf{q}}_1^T\tilde{\mathbf{q}}_1. \tag{5.10}$$

We want the time derivative of V_1 to be negative, so we examine $\dot{V}_1$

$$\dot{V}_1 = \tilde{\mathbf{q}}_1^T\dot{\tilde{\mathbf{q}}}_1 = \tilde{\mathbf{q}}_1^T(\dot{\mathbf{q}}_1 - \dot{\mathbf{q}}_d). \tag{5.11}$$

Equation (5.6) indicates that the control input $\overline{\mathbf{u}}$ does not appear in the expression $\dot{\mathbf{q}}_1$, and it only appears in the expression $\ddot{\mathbf{q}}_1$. To control the system (5.6) , we introduce another error variable which consists of $\ddot{\mathbf{q}}_1$, and iterate the backstepping procedure. To make $\dot{V}_1$ negative, it is convenient to introduce a new error variable as

$$\tilde{\mathbf{q}}_2 = \dot{\mathbf{q}}_1 - \dot{\mathbf{q}}_d + \lambda_1\tilde{\mathbf{q}}_1 \tag{5.12}$$

where λ_1 is a design parameter to be specified.

Substituting Eq. (5.12) into Eq. (5.11), we get

$$\dot{V}_1 = -\lambda_1\tilde{\mathbf{q}}_1^T\tilde{\mathbf{q}}_1 + \tilde{\mathbf{q}}_2^T\tilde{\mathbf{q}}_1. \tag{5.13}$$

Introducing a new Lyapunov function as

$$V_2 = V_1 + \frac{1}{2}\tilde{\mathbf{q}}_2^T\tilde{\mathbf{q}}_2 \tag{5.14}$$

and the corresponding time derivative is

$$\dot{V}_2 = -\lambda_1 \tilde{\mathbf{q}}_1^T \tilde{\mathbf{q}}_1 + \tilde{\mathbf{q}}_2^T (\tilde{\mathbf{q}}_1 + \dot{\tilde{\mathbf{q}}}_2). \tag{5.15}$$

Computing the time derivative of $\tilde{\mathbf{q}}_2$ from Eq. (5.12), we get

$$\begin{aligned} \dot{\tilde{\mathbf{q}}}_2 &= \ddot{\mathbf{q}}_1 - \ddot{\mathbf{q}}_d + \lambda_1 \dot{\tilde{\mathbf{q}}}_1 \\ &= \begin{bmatrix} \frac{F_X}{m} \\ \frac{F_Y}{m} \\ \frac{\tau}{I_0} \end{bmatrix} + \begin{bmatrix} \frac{\overline{u}_1}{m} \\ \frac{\overline{u}_2}{m} \\ \frac{\overline{u}_3}{I_0} \end{bmatrix} - \ddot{\mathbf{q}}_d + \lambda_1 \dot{\tilde{\mathbf{q}}}_1. \end{aligned} \tag{5.16}$$

By substituting Eq. (5.16) into Eq. (5.15), we rewrite Eq. (5.15) as

$$\dot{V}_2 = -\lambda_1 \tilde{\mathbf{q}}_1^T \tilde{\mathbf{q}}_1 + \tilde{\mathbf{q}}_2^T (\tilde{\mathbf{q}}_1 + \begin{bmatrix} \frac{F_X}{m} \\ \frac{F_Y}{m} \\ \frac{\tau}{I_0} \end{bmatrix} + \begin{bmatrix} \frac{\overline{u}_1}{m} \\ \frac{\overline{u}_2}{m} \\ \frac{\overline{u}_3}{I_0} \end{bmatrix} - \ddot{\mathbf{q}}_d + \lambda_1 \dot{\tilde{\mathbf{q}}}_1). \tag{5.17}$$

In order to make $\dot{V}_2 < 0$, we choose the generalized force control law $\overline{\mathbf{u}}$ as

$$\begin{aligned} \overline{\mathbf{u}} = \begin{bmatrix} \overline{u}_1 \\ \overline{u}_2 \\ \overline{u}_3 \end{bmatrix} &= \begin{bmatrix} m & & \\ & m & \\ & & I_0 \end{bmatrix} \cdot \\ &\left(-\tilde{\mathbf{q}}_1 - \begin{bmatrix} \frac{F_X}{m} \\ \frac{F_Y}{m} \\ \frac{\tau}{I_0} \end{bmatrix} + \ddot{\mathbf{q}}_d - \lambda_1 \dot{\tilde{\mathbf{q}}}_1 - \lambda_2 \tilde{\mathbf{q}}_2 \right) \end{aligned} \tag{5.18}$$

with $\lambda_1 > 0$, and $\lambda_2 > 0$ as the design parameters.

Submitting Eq. (5.18) into Eq. (5.17), we get

$$\dot{V}_2 = -\lambda_1 \tilde{\mathbf{q}}_1^T \tilde{\mathbf{q}}_1 - \lambda_2 \tilde{\mathbf{q}}_2^T \tilde{\mathbf{q}}_2 < 0. \tag{5.19}$$

If we apply the generalized forces $\overline{\mathbf{u}}$ as Eq. (5.18) to the system (5.6), the error variables $\tilde{\mathbf{q}}_1$ and $\tilde{\mathbf{q}}_2$ will approach zero asymptotically. With the generalized force pushing on the object, the object will follow the given trajectory, and attain the desired velocity.

5.2.2 Coordination as a Quadratic Programming Problem

After we obtain the generalized force control law (wrench) (5.18) through the back-stepping design, the next task is to distribute the required generalized force between the agents. The distribution of the generalized force between agents is the objective of the multi-agent coordination. The coordination between agents is formulated as a quadratic programming problem.

By considering the quadratic performance index (5.5), the inequality constraints (5.7) and (5.3) and the equality constraint (5.4), the coordination problem of cooperative dynamic pushing is formulated as a quadratic programming problem

$$\begin{array}{ll} \min \frac{1}{2}\mathbf{u}^T(t)Q\mathbf{u}(t) & \\ \text{Subject to} & \mu_i F_i^n \geq |F_i^t|, \quad i = 1, \cdots, N \\ & F_i^n \geq 0, \quad i = 1, \cdots, N \\ & \overline{\mathbf{u}} = \sum_{i=1}^{N} R \cdot W_i \mathbf{u}_i. \end{array} \tag{5.20}$$

Quadratic programs can always be solved in a finite number of iterations, but the effort required to find a solution depends strongly on the characteristics of the objective function and the number of inequality constraints. If the Hessian matrix Q is positive semi-definite, the problem will be a convex QP. For the coordination problem, the matrix Q is positive definite, and the problem is a convex QP. Here we use the routine **quadprog** in the MATLAB optimization toolbox to solve the QP problem. **quadprog** uses an active set method, detailed description and implementation of this method can be found in [33].

5.3 Simulation Results

In the previous section, we proposed a planning method for the cooperative dynamic manipulation. Simulations will be conducted on a triangular object under two-agent or three-agent pushing. The object is shown in Fig. 5.2 (where $a = 0.2$ m, $b = 0.17$

m,and the angle $\angle A = \pi/6$). We assume uniform distribution of the mass with the mass density $\rho = 200\text{kg/m}^2$. Using the approach proposed in [57], we get the mass of the object $m = 1.7$ Kg, and the mass moment of inertia is $I_0 = 0.0037\text{Kg}\cdot\text{m}^2$. Assume the friction coefficient between the object and the surface is $\mu = 0.5$, and the friction coefficient between the agents and the object is $\mu_i = 0.8$. The local coordinate frame is located at the COM and fixed on the object. We also assign the world frame at the same location, and it is fixed on the supporting plane. As shown in Fig. 5.2, the first agent is pushing at edge AC, the second agent is pushing at BC, and the third agent is pushing at AB. If we assign a contact coordinate frame to each agent, the associated contact normal and tangential vectors are

$$\begin{array}{ll} \mathbf{n}_1 = (\frac{1}{2}, -\frac{\sqrt{3}}{2})^T & \mathbf{t}_1 = (\frac{\sqrt{3}}{2}, \frac{1}{2})^T \\ \mathbf{n}_2 = (-\frac{\sqrt{3}}{2}, -\frac{1}{2})^T & \mathbf{t}_2 = (\frac{1}{2}, -\frac{\sqrt{3}}{2})^T \\ \mathbf{n}_3 = (0, 1)^T & \mathbf{t}_3 = (-1, 0)^T. \end{array} \tag{5.21}$$

The coordinates of the contact points in the object frame are

$$\begin{array}{l} L_1 = (-0.0167 \quad 0.0289) \\ L_2 = (0.0500 \quad 0.00289) \\ L_3 = (-0.0400 \quad -0.0283) \end{array} \tag{5.22}$$

and the wrench matrices are computed according to Eq. (5.2) as

$$\mathbf{W}_1 = \begin{bmatrix} 0.5000 & 0.8660 \\ -0.8660 & 0.5000 \\ 0.0476 & -0.0333 \end{bmatrix} \tag{5.23}$$

$$\mathbf{W}_2 = \begin{bmatrix} -0.8660 & 0.5000 \\ -0.5000 & -0.8660 \\ -0.0278 & 0.0481 \end{bmatrix} \tag{5.24}$$

$$\mathbf{W}_3 = \begin{bmatrix} 0 & -1.000 \\ 1.000 & 0 \\ -0.0400 & -0.0283 \end{bmatrix}. \tag{5.25}$$

5.3.1 Coordination under Two-agent Manipulation

In this simulation, the desired trajectory is a linear translation, without rotation. The desired trajectory is

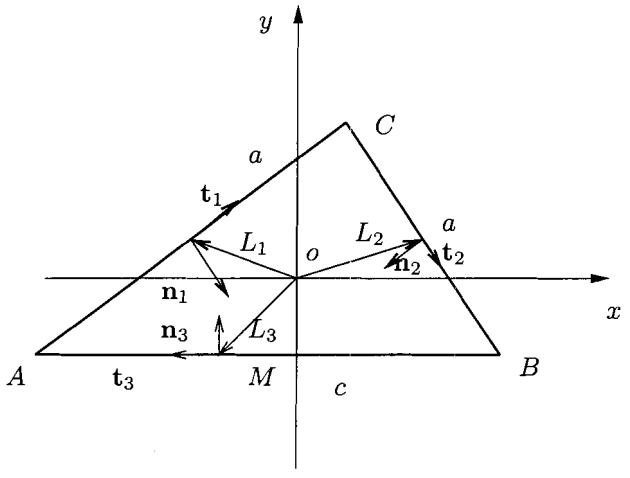


Figure 5.2: Manipulation of triangular object

$$\begin{aligned} X_{or}(t) &= -\tfrac{1}{10}t^2 \\ Y_{or}(t) &= -\tfrac{1}{5}t^2 \\ \theta_{or}(t) &= 0. \end{aligned} \tag{5.26}$$

The required velocity and acceleration trajectories are

$$\begin{aligned} \dot{X}_{or}(t) &= -\tfrac{1}{5}t & \ddot{X}_{or}(t) &= -\tfrac{1}{5} \\ \dot{Y}_{or}(t) &= -\tfrac{2}{5}t & \ddot{Y}_{or}(t) &= -\tfrac{2}{5} \\ \dot{\theta}_{or}(t) &= 0 & \ddot{\theta}_{or}(t) &= 0. \end{aligned} \tag{5.27}$$

Since we consider only translation, the object will have two degrees of freedom, and two agents are enough to control the object to track the given trajectory. The agents located on edges AC, BC perform the manipulation task. We solve the planning problem in two stages. The generalized force control law $\overline{\mathbf{u}}$ was designed according to Eq. (5.18). The coordination is solved as a quadratic programming problem (5.20). The optimal distribution of forces are shown in Fig. 5.3. It is clear that at the beginning, the forces oscillate, and then approach stable and constant values. With the input forces, the object will follow the designed trajectory and obtain the predefined velocity.

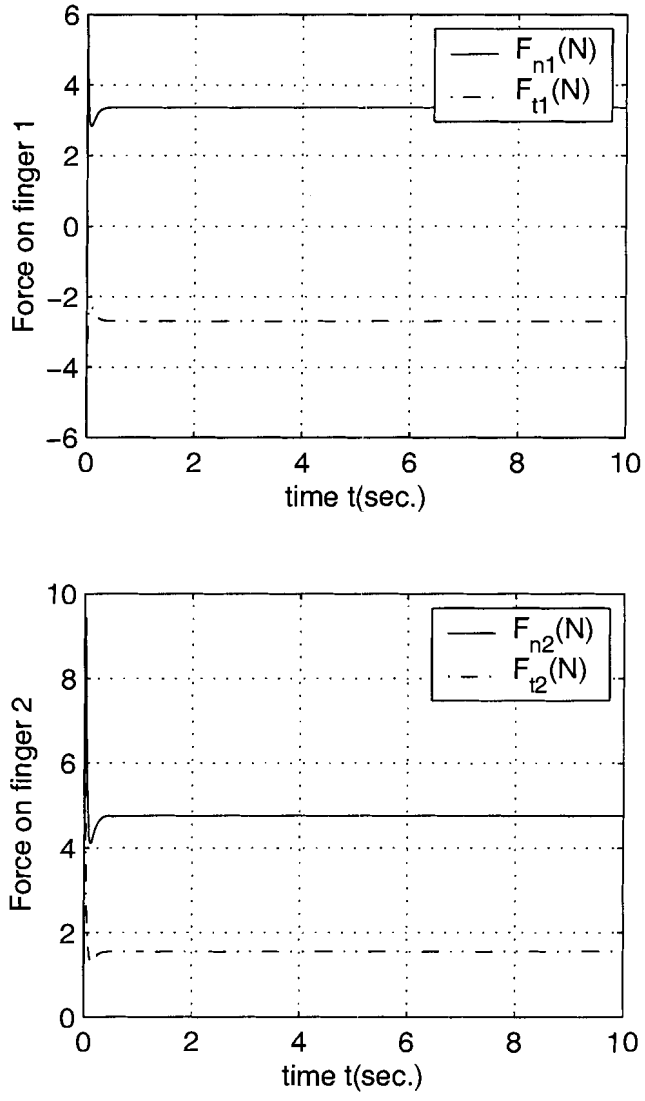


Figure 5.3: Force coordination under two-agent manipulation

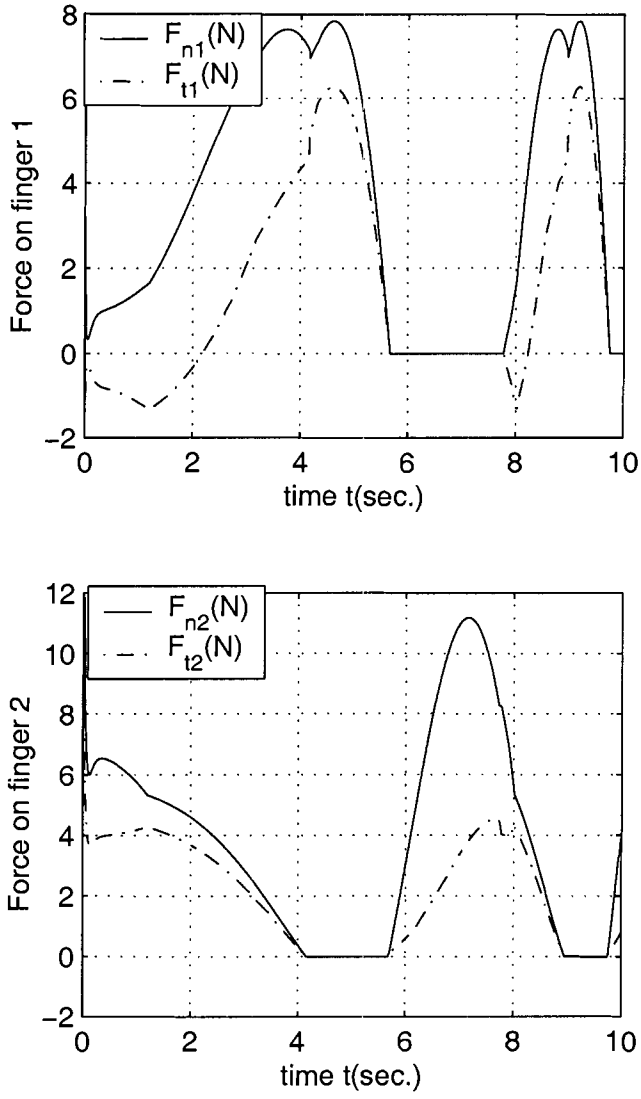


Figure 5.4: Force coordination under three-agent manipulation

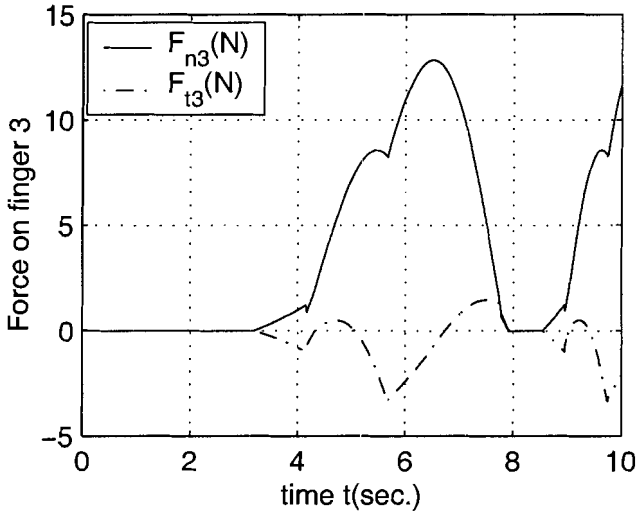


Figure 5.5: Force coordination under three-agent manipulation

5.3.2 Coordination under Three-agent Manipulation

If we consider both translation and rotation of the object, three agents are required to manipulate the object to follow a predefined trajectory. In this example, the trajectories are given as

$$\begin{array}{lll} X_{or}(t) = -\frac{1}{10}t^2 & \dot{X}_{or}(t) = -\frac{1}{5}t & \ddot{X}_{or}(t) = -\frac{1}{5} \\ Y_{or}(t) = -\frac{1}{5}t^2 & \dot{Y}_{or}(t) = -\frac{2}{5}t & \ddot{Y}_{or}(t) = -\frac{2}{5} \\ \theta_{or}(t) = -\frac{1}{10}t^2 & \dot{\theta}_{or}(t) = -\frac{1}{5}t & \ddot{\theta}_{or}(t) = -\frac{1}{5}. \end{array} \tag{5.28}$$

We proceed with the proposed coordination method, and the results of force distribution are shown in Fig. 5.4-Fig. 5.5. With these force inputs, the object can track the given trajectory and obtain the desired velocity, and the forces generated by each agent satisfy the constraints and achieve the predefined performance index.

5.4 Discussion

This chapter studies the planning problem for cooperative dynamic pushing for object acceleration and proposes a centralized planner for the pushing forces. The hierarchical structure of the planner will increase the flexibility of the system. When more agents join the manipulation, redesign of the lower level generalized force control law is not needed. Here, the integrator backstepping design technique is used to find the generalized force applied to the object. Whereas other nonlinear controller design methods such as feedback linearization can also be used for this purpose. With the planned force inputs, force and position control schemes can be used to achieve the interaction force and the tracking trajectory [79].

In the proposed planner, we assume that there exists a trajectory such that the object can achieve a desired releasing velocity, and this trajectory is generated by geometric motion planners [48]. The other assumption is that the friction between the object and the supporting surface is known. The planning problem for cooperative dynamic pushing with uncertainty in friction will be addressed in Chapter 6.

Chapter 6

Cooperative Dynamic Pushing under Uncertainty

In the previous chapter, the planning problem for cooperative dynamic pushing has been investigated under the assumption of known pressure distribution. In other words, the friction force and torque are known. In practice, the pressure distribution of the pushed object is not known exactly, which introduces uncertainties into the cooperative dynamic pushing model. As a result, we cannot apply the backstepping design technique directly to this model and solve the planning problem. This chapter presents a novel planning method for the cooperative dynamic pushing under uncertainty. The main novelty of the proposed approach is the integration of noncooperative game and cooperative game between agents in a hierarchical manner. Based on the dynamic motion model of the pushed object, the coordination problem is solved in two levels.

In the control level, a generalized force control law is designed by using the H^∞ design technique to achieve the minimax tracking performance. The design procedure for the generalized force control law is divided into two steps. First, a linear

nominal control design is obtained via full-state linearization with desired eigenvalue assignment; then, a minimax control scheme is introduced to optimally attenuate the worst-case effect of the uncertainty.

In the coordination level, a game is formulated between agents to distribute the generalized force. The objective of the game is to minimize the worst-case interaction force between the agents and the object. On one hand, each agent makes its own decision and has an intention to minimize its own objective; on the other hand, the agents have a common goal of finishing the manipulation task. By considering these two factors, the coordination problem is formulated as a cooperative game between the pushing agents.

Compared with minimizing a quadratic function of the interaction forces used in Chapter 5, the minimization of the worst-case interaction force ia a more realistic measure on the performance of a manipulation method, especially useful in the case of fragile and delicate object manipulation.

The remainder of the chapter is organized as follows: Section 6.1 presents the model of cooperative dynamic pushing with uncertainty, and formulates the problem. Section 6.2 presents the planning and coordination method. Section 6.3 provides the numerical simulations. Section 6.4 concludes this chapter with a discussion.

6.1 Problem Formulation

6.1.1 Model of Cooperative Dynamic Pushing with Uncertainty

Consider an object under cooperative dynamic pushing as shown in Fig. 6.1. Assign XOY as the world frame, xoy is the local coordinate frame associated with the object,

and o is located at the center of mass (COM). The configuration of the object is described as (X_o, Y_o, θ), where (X_o, Y_o) is the position of the local coordinate frame, and θ is the orientation of the object. For each contact point, we introduce a contact frame with its **n**-axis aligned with the contact normal (points inward with respect to the object), and its **t**-axis aligned with the contact tangent such that the cross product of $\mathbf{n}_i$ and $\mathbf{t}_i$ points out of the plane. The directional vectors $\mathbf{n}_i$ and $\mathbf{t}_i$ are defined in the local coordinate frame. We introduce vectors $\mathbf{q}_1 = (X_o, Y_o, \theta)^T$, and $\mathbf{q}_2 = (\dot{X}_o, \dot{Y}_o.\dot{\theta})^T$ that describe the configuration and velocity of the object. The state of the system are defined as $\mathbf{q} = (\mathbf{q}_1^T, \mathbf{q}_2^T)^T$. The cooperative dynamic pushing under uncertainty is modeled as a nonlinear system

$$\begin{aligned} \dot{\mathbf{q}}_1 &= \mathbf{q}_2 \\ \dot{\mathbf{q}}_2 &= \mathbf{f}(\mathbf{q}) + \mathbf{d} + M_t \textstyle\sum_{i=1}^{N} R \cdot W_i \begin{bmatrix} F_i^n \\ F_i^t \end{bmatrix} \end{aligned} \tag{6.1}$$

where $\mathbf{f}(\mathbf{q}) = (F_X/m, F_Y/m, T/I_0)^T$, F_X, F_Y are the friction forces in the world frame, and T is the torque with respect to the COM generated by the friction force. They are computed in the same manner as that in Chapter 4. m is mass of the object, and I_0 is the mass moment of inertia about the COM.

$$M_t = \begin{bmatrix} \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{I_0} \end{bmatrix}$$

and the rotational transformation matrix is

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$F_i^n, F_i^t (i = 1, \cdots, N)$ are the normal and tangential components of the pushing force applied by agent i in the contact frame, and N is the total number of agents. W_i is the wrench matrix for agent i, which is used to transform the contact force F_i^n, F_i^t to forces and torque in the local coordinate frame xoy. The wrench matrix W_i

is computed as [101]

$$W_i = [\mathbf{w}_{in} \quad \mathbf{w}_{it}] = \begin{bmatrix} \mathbf{n}_i & \mathbf{t}_i \\ \mathbf{L}_i \times \mathbf{n}_i & \mathbf{L}_i \times \mathbf{t}_i \end{bmatrix}, \quad i = 1, 2, \cdots, N \tag{6.2}$$

where $\mathbf{L}_i$ is the position vector of the i-th contact point in the local coordinate frame, and $\times$ is the cross product between two vectors.

In order to prevent the slippage between the agents and the object, the applied pushing forces to the object must be constrained to lie inside the friction cones. These constraints are written as

$$|F_i^t| \leq \mu_i F_i^n, \quad i = 1, 2, \cdots, N \tag{6.3}$$

where μ_i is the friction coefficient between the agent i and the object at the i-th contact point.

Since we consider pushing instead of pulling of the object, the contact normal force needs to be nonnegative, *i.e.*

$$F_i^n \geq 0, \quad i = 1, 2, \cdots, N \tag{6.4}$$

$\mathbf{d} = (\delta F_X/m, \delta F_Y/m, \delta T/I_0)^T \in L_2[0, t_f]$ is the disturbance to the system introduced by the uncertainty of the contact friction between the object and the supporting surface. In detail, $\delta F_X, \delta F_Y$ are the uncertainties on the friction forces owing to uncertain pressure distribution. δT is the uncertainty associated with the torque T.

The major difference between the system (6.1) and the system (5.1) of Chapter 5 is that, in the model (6.1), the uncertainty $\mathbf{d}(t)$ is considered. Because of this uncertainty, the backstepping design method presented in Chapter 5 cannot be applied directly.

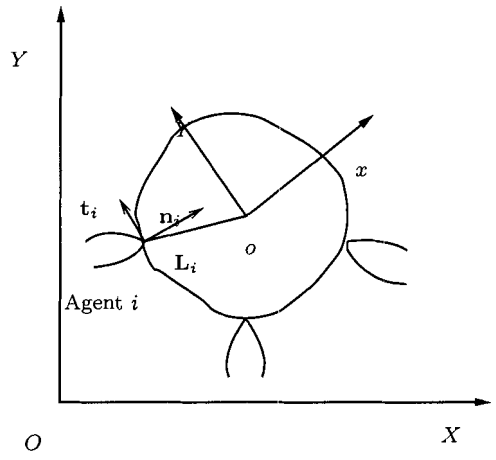


Figure 6.1: Dynamic cooperative pushing under uncertainty

6.1.2 The Cooperative Push Planning Problem

The cooperative dynamic manipulation planning problem is described as: Given an object on a plane with a known initial configuration, find the forces applied by each agent, such that the object moves to a specified goal configuration with the desired velocity. The inertial forces are not negligible compared to quasi-static manipulation. Thus, the planning method must be developed based on the dynamic model of motion described by Eq. (6.1). We assume that we have the collision free path for the object to follow, and we also assume that the states of pushing system are available for measurement. The objective of the planning is to find the forces such that the object follows a given trajectory, and obtains the desired velocity. At the same time, we want to minimize the worst-case interaction force between the agents and the object. The cooperative manipulation planning problem is stated as follows:

Given a continuously differentiable and uniformly bounded trajectory $\mathbf{q}_d(t) = (X_{or}(t), Y_{or}(t), \theta_r(t))$, the corresponding velocity $\dot{\mathbf{q}}_d(t) = (\dot{X}_{or}(t), \dot{Y}_{or}(t), \dot{\theta}_r(t))$ and the acceleration $\ddot{\mathbf{q}}_d(t) = (\ddot{X}_{or}(t), \ddot{Y}_{or}(t), \ddot{\theta}_r(t))$, design the state feedback control inputs

$\mathbf{u}(t) = (F_1^n, F_1^t, F_2^n, F_2^t, \cdots, F_N^n, F_N^t)^T$ such that the state of system (6.1) follows the desired trajectory, the objective function $M(\mathbf{u}(t))$ is defined as

$$M(\mathbf{u}(t)) = \min_u \max_i \{F_i^n, i = 1, \cdots, N\}. \tag{6.5}$$

With this objective function, the worst-case normal interaction force between the agents and the object will be minimized.

The problem will be addressed hierarchically in the next section.

6.2 Planning the Cooperative Pushing

In this section, we will solve the coordination problem and compute the forces F_i^n and F_i^t generated by each agent such that the object follows the given trajectory and obtains the desired velocity.

The object under pushing has three degrees of freedom; generally it will require three independent control inputs to fully control the object to follow any given trajectory. By introducing generalized force control inputs $\overline{\mathbf{u}} = (\overline{u}_1, \overline{u}_2, \overline{u}_3)^T$ to system (6.1), the system is rewritten as

$$\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{q}_2 \\ \mathbf{f}(\mathbf{q}) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{d} \end{bmatrix} + M_t \cdot \begin{bmatrix} \mathbf{0} \\ \overline{\mathbf{u}} \end{bmatrix}. \tag{6.6}$$

From Eqs. (6.1) and (6.6), we know that the generalized force control law $\overline{\mathbf{u}}$ satisfies the following equation

$$\overline{\mathbf{u}} = \sum_{i=1}^{N} R \cdot W_i \cdot \begin{bmatrix} F_i^n \\ F_i^t \end{bmatrix}. \tag{6.7}$$

Based on Eqs. (6.7) and (6.6), the overall planning problem is solved hierarchically in two levels. Namely, design a generalized force control law and the coordination between the agents.

6.2.1 H^∞ Design for the Generalized Force Controller

First, let us define

$$\tilde{\mathbf{q}}_1 = \mathbf{q}_1 - \mathbf{q}_d \tag{6.8}$$

as the tracking error. Since the nonlinearities $\mathbf{d}(t)$ appear together with the control $\overline{\mathbf{u}}$ in Eq. (6.6) and, if the nonlinearities are known, state feedback linearization or backstepping technique could be used to design a control law to cancel the nonlinearities. Whereas, in this application, the nonlinearities are only partially known, these methods cannot be applied directly. For this reason, we break the controller design into two steps: the design of the state feedback controller and the auxiliary minimax controller.

Based on the available information on nonlinearities in system (6.6), we can design the state feedback control law as

$$\overline{\mathbf{u}} = M_t^{-1}(\ddot{\mathbf{q}}_d - K_1\tilde{\mathbf{q}}_1 - K_2\dot{\tilde{\mathbf{q}}}_1 + \mathbf{u}_0 - \mathbf{f}(\mathbf{q})) \tag{6.9}$$

where $\ddot{\mathbf{q}}_d$ denotes the second order time derivative of $\mathbf{q}_d$, and K_1, K_2 are 3×3 matrices to be designed and $\mathbf{u}_0$ is an auxiliary control signal yet to be specified. Substituting Eq. (6.9) and Eq. (6.8) into Eq. (6.6), yields,

$$\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{q}_2 \\ \ddot{\mathbf{q}}_d - K_1\tilde{\mathbf{q}}_1 - K_2\dot{\tilde{\mathbf{q}}}_1 + \mathbf{u}_0 + \mathbf{d} \end{bmatrix}. \tag{6.10}$$

Let

$$\mathbf{e} = \begin{bmatrix} \tilde{\mathbf{q}}_1 \\ \dot{\tilde{\mathbf{q}}}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 - \mathbf{q}_d \\ \dot{\mathbf{q}}_1 - \dot{\mathbf{q}}_d \end{bmatrix} \tag{6.11}$$

be the state error vector. From the Eqs. (6.8), (6.10), and (6.11), we obtain

$$\dot{\mathbf{e}} = \begin{bmatrix} \dot{\tilde{\mathbf{q}}}_1 \\ \ddot{\tilde{\mathbf{q}}}_1 \end{bmatrix} = \begin{bmatrix} \dot{\tilde{\mathbf{q}}}_1 \\ -K_1\tilde{\mathbf{q}}_1 - K_2\dot{\tilde{\mathbf{q}}}_1 + \mathbf{u}_0 + \mathbf{d}(t) \end{bmatrix} \tag{6.12}$$

The system (6.12) represents the dynamics of the tracking error, and it is a nominal linear uncertain system. In this equation, the nonlinear uncertainty is modeled by $\mathbf{d}(t)$. A more convenient form of Eq. (6.12) could be

$$\dot{\mathbf{e}} = A\mathbf{e} + B\mathbf{u}_0 + B\mathbf{d} \tag{6.13}$$

where

$$A = \begin{bmatrix} \mathbf{0} & I \\ -K_1 & -K_2 \end{bmatrix} \text{ and } \quad B = \begin{bmatrix} \mathbf{0} \\ I \end{bmatrix}. \tag{6.14}$$

The control parameters K_1, K_2 are selected such that A is Hurwitz and has desired eigenvalues such that the tracking dynamic equation (6.13) has a desired response if the system is free of uncertainty. However, in the case of uncertainty, the tracking performance will deteriorate, and the system may even be unstable. In order to eliminate the effect of uncertainty, a robust controller must be employed. The H^∞ control technique, as one of the most efficient approaches to attenuate the worst-case effect of uncertainties, is used to design the auxiliary minimax control input $\mathbf{u}_0$. The control signal $\mathbf{u}_0(t)$ should be specified such that the worst-case effect of $\mathbf{d}(t)$ on the tracking error $\mathbf{e}(t)$ must be attenuated below a prescribed level γ. The H^∞ tracking performance index is defined as [96]

$$\min_{\mathbf{u}_0} \max_{\mathbf{d}} \frac{\int_0^{t_f} [\|\mathbf{e}(t)\|_Q^2 + \|\mathbf{u}_0\|_{R_c}^2]\, dt}{\int_0^{t_f} \|\mathbf{d}(t)\|^2\, dt} \leq \gamma^2 \quad \forall \quad \mathbf{d} \in L_2[0, t_f] \tag{6.15}$$

where

$$\|C(t)\|_Q^2 = \mathbf{e}^T(t) Q \mathbf{e}(t)$$
$$\|\mathbf{u}_0\|_{R_c}^2 = \mathbf{e}^T(t) R_c \mathbf{e}(t)$$

where $Q = Q^T > 0$ and $R_c = R_c^T > 0$ are the weighting matrices.

The minimax performance index (6.15) can be viewed as a non-cooperative game between two players, namely, the auxiliary control law $\mathbf{u}_0(t)$ and the disturbance $\mathbf{d}(t)$. The goal of the auxiliary control law is to attenuate the effect of the disturbance on

the tracking error. On the other hand, the objective of the disturbance is try to maximize the tracking error.

In the case $\mathbf{e}(0) \neq 0$, the performance index is transformed as

$$\begin{array}{l} \min_{\mathbf{u}_0(t)} \max_{\mathbf{d}(t)} \int_0^{t_f} (\mathbf{e}^T(t) Q \mathbf{e}(t) \\ + \mathbf{u}_0^T(t) R_c \mathbf{u}_0(t) - \gamma^2 \mathbf{d}^T(t) \mathbf{d}(t)) dt \leq \mathbf{e}^T(0) P \mathbf{e}(0) \end{array} \tag{6.16}$$

where $P = P^T > 0$ is a positive definite weighting matrix to be specified.

Let us define the cost function $J_c(\mathbf{e}, \mathbf{u}_0, \mathbf{d})$ as

$$J_c(\mathbf{e}, \mathbf{u}_0, \mathbf{d}) = \int_0^{t_f} \mathbf{e}^T(t) Q \mathbf{e}(t) + \mathbf{u}_0^T(t) R_c \mathbf{u}_0(t) - \gamma^2 \mathbf{d}(t)^T(t) \mathbf{d}(t) dt. \tag{6.17}$$

After some rearrangements, we obtain,

$$\begin{array}{rcl} J_c(\mathbf{e}, \mathbf{u}_0, \mathbf{d}) & = & \mathbf{e}^T(0) P \mathbf{e}(0) - \mathbf{e}^T(t_f) P \mathbf{e}(t_f) + \int_0^{t_f} \left[\mathbf{e}^T(t) Q \mathbf{e}(t) \right. \\ & & \left. + \mathbf{u}_0^T(t) R_c \mathbf{u}_0(t) - \gamma^2 \mathbf{d}^T(t) \mathbf{d}(t) + \frac{\mathbf{d}}{dt} (\mathbf{e}^T(t) P \mathbf{e}(t)) \right] dt \\ & = & \mathbf{e}^T(0) P \mathbf{e}(0) - \mathbf{e}^T(t_f) P \mathbf{e}(t_f) + \int_0^{t_f} \left[\mathbf{e}^T(t) Q \mathbf{e}(t) \right. \\ & & + \mathbf{u}_0^T(t) R_c \mathbf{u}_0(t) - \gamma^2 \mathbf{d}^T(t) \mathbf{d}(t) \\ & & \left. + \dot{\mathbf{e}}^T(t) P \mathbf{e}(t) + \mathbf{e}^T(t) P \dot{\mathbf{e}}(t) \right] dt. \end{array} \tag{6.18}$$

Substituting Eq. (6.13) into Eq. (6.18), yields,

$$\begin{array}{rcl} J_c(\mathbf{e}, \mathbf{u}_0, \mathbf{d}) & = & \mathbf{e}^T(0) P \mathbf{e}(0) - \mathbf{e}^T(t_f) P \mathbf{e}(t_f) + \int_0^{t_f} [\mathbf{e}^T(t) (A^T P + P A \\ & & + Q) \mathbf{e}(t) + \mathbf{u}_0^T(t) R_c \mathbf{u}_0(t) - \gamma^2 \mathbf{d}^T(t) \mathbf{d}(t) + \mathbf{u}_0^T(t) B^T P \mathbf{e}(t) \\ & & + \mathbf{e}^T(t) P B \mathbf{u}_0(t) + \mathbf{e}^T(t) P B \mathbf{d}(t) + \mathbf{d}^T(t) B^T P \mathbf{e}(t)] dt. \end{array} \tag{6.19}$$

The result on the generalized force control law design is summarized in the following theorem.

Theorem 6.1 *For the system (6.6), let the state feedback control law* $\overline{\mathbf{u}}(t)$ *be chosen as*

$$\overline{\mathbf{u}} = M_t^{-1} (\mathbf{q}_d^{(2)} - K_1 \tilde{\mathbf{q}}_1 - K_2 \dot{\tilde{\mathbf{q}}}_1 + \mathbf{u}_0 - \mathbf{f}(\mathbf{q})) \tag{6.20}$$

with the auxiliary control law $\mathbf{u}_0$ *as*

$$\mathbf{u}_0 = -R_c^{-1} B^T P \mathbf{e}(t) \tag{6.21}$$

where $R_c = R_c^T > 0$ *is a weighting matrix and* $P = P^T > 0$ *is the solution of the following algebraic Riccati-like equation*

$$PA + A^T P + Q - PB(R_c^{-1} - \frac{1}{\gamma^2} I)B^T P = 0. \tag{6.22}$$

Then, the minimax tracking in Eq. (6.16) is guaranteed for a prescribed γ *and the corresponding worst-case disturbance* $\mathbf{d}^*(t)$ *is of the form*

$$\mathbf{d}^*(t) = \frac{1}{\gamma^2} B^T P \mathbf{e}(t). \tag{6.23}$$

Proof: Substituting Eq. (6.22) to Eq. (6.19), we get

$$\begin{aligned} J_c(\mathbf{e}, \mathbf{u}_0, \mathbf{d}) \;=\; & \mathbf{e}^T(0)P\mathbf{e}(0) - \mathbf{e}^T(t_f)P\mathbf{e}(t_f) + \int_0^{t_f} [\mathbf{e}^T(t)(PA + A^T P + Q \\ & -PB(R_c^{-1} - \tfrac{1}{\gamma^2} I)B^T P)\mathbf{e}(t) + \mathbf{u}_0^T(t) R_c \mathbf{u}_0(t) + \mathbf{u}_0^T(t) B^T P \mathbf{e}(t) \\ & +\mathbf{e}^T(t) PB\mathbf{u}_0(t) + \mathbf{e}^T(t) PBR_c^{-1} B^T P \mathbf{e}(t) - \gamma^2 \mathbf{d}^T(t)\mathbf{d}(t) \\ & +\mathbf{d}^T(t) B^T P \mathbf{e}(t) + \mathbf{e}^T(t) PB\mathbf{d}(t) - \tfrac{1}{\gamma^2} \mathbf{e}^T(t) PBB^T P \mathbf{e}(t)] dt. \end{aligned} \tag{6.24}$$

From Eq. (6.22) and using the technique of completion of the squares, we get

$$\begin{aligned} J_c(\mathbf{e}, \mathbf{u}_0, \mathbf{d}) \;=\; & \mathbf{e}^T(0)P\mathbf{e}(0) - \mathbf{e}^T(t_f)P\mathbf{e}(t_f) \\ & + \int_0^{t_f} [R_c\mathbf{u}_0(t) + B^T P\mathbf{e}(t)]^T R_c^{-1} [R_c\mathbf{u}_0(t) + B^T P\mathbf{e}(t)] \\ & -(\gamma\mathbf{d}(t) - \tfrac{1}{\gamma} B^T P\mathbf{e}(t))^T (\gamma\mathbf{d}(t) - \tfrac{1}{\gamma} B^T P\mathbf{e}(t))] dt. \end{aligned} \tag{6.25}$$

From dynamic game theory [13], we know that the control $\mathbf{u}_0(t)$ tries to minimize $J_c(\mathbf{e}_0, \mathbf{u}_0, \mathbf{d})$, and the disturbance $\mathbf{d}(t)$ wants to maximize the performance index $J_c(\mathbf{e}_0, \mathbf{u}_0, \mathbf{d})$. From Eq. (6.25), we know that when the optimal control is chosen as Eq. (6.21), and the worst-case disturbance $\mathbf{d}^*(t)$ takes the value as Eq. (6.23), the second and third terms in Eq. (6.25) become zero. As a result, the following equation is satisfied

$$\begin{aligned} \underset{\mathbf{u}_0(t)}{\min}\underset{\mathbf{d}(t)}{\max}\, J_c(\mathbf{e}, \mathbf{u}_0, \mathbf{d}) \;&=\; \mathbf{e}^T(0)P\mathbf{e}(0) - \mathbf{e}^T(t_f)P\mathbf{e}(t_f) \\ &\leq\; \mathbf{e}^T(0)P\mathbf{e}(0). \end{aligned} \tag{6.26}$$

The above inequality holds when $P = P^T > 0$, and $R_c = R_c^T > 0$.

■

Remark:

In general, we choose $\gamma > 0$ to attenuate $\mathbf{d}(t)$ in order to achieve robust minimax tracking performance. In the case $\gamma \to \infty$, the H^∞ design is reduced to a H_2 optimal tracking control, where no attenuation of $\mathbf{d}(t)$ is considered.

In order to guarantee the positive definite solution of P, the Riccati-like equation (6.22) must satisfy the following condition [96]

$$\gamma^2 I \geq R_c. \tag{6.27}$$

This tells us there is a tradeoff between the attenuation level γ and the control input $\mathbf{u}_0(t)$, *i.e.*, with the smaller γ, a larger $\mathbf{u}_0(t)$ is required.

From the above analysis, a design procedure for the generalized force control law is summarized as follows.

1. Specify the matrices K_1, K_2, and determine A with desired eigenvalues.

2. Choose a desired attenuation level γ, and select positive-definite matrices Q and R_c with $R_c \leq \gamma^2 I$.

3. Solve the positive-definite matrix P from the Riccati-like equation (6.22).

4. Compute the H^∞ auxiliary minimax control law (6.21) and the state feedback control law $\overline{\mathbf{u}}$ as Eq. (6.20).

Until now, we have designed the generalized force control law for the minimax tracking performance, our next task is to distribute the generalized force to the agents optimally and satisfy the condition (6.7).

6.2.2 Coordination as a Cooperative Game

The objective of the multi-agent coordination is to distribute the generalized force obtained in Eq. (6.20) between the pushing agents. The goal is to minimize the maximal interaction force between the agents and the object. Define a vector-valued objective function as

$$\mathbf{F} = [\ F_1^n \quad F_2^n \quad \cdots \quad F_N^n\ ] \tag{6.28}$$

where F_i^n is the objective for agent i to minimize.

For agent i, the goal is to minimize F_i^n through choosing the interaction forces $\mathbf{u}_i = (F_i^n, F_i^t)$. By considering the equality constraint (6.7) and the inequality constraints (6.3) and (6.4), the multi-agent coordination is studied as a cooperative game between agents, and it is formulated as

$$\begin{array}{ll} \min\limits_{\mathbf{u}} \{\mathbf{F}\} & \\ \text{Subject to} & \mu_i F_i^n \geq |F_i^t|, \quad i = 1, \cdots, N \\ & F_i^n \geq 0, \quad i = 1, \cdots, N \\ & \overline{\mathbf{u}} = \sum_{i=1}^{N} R \cdot W_i \mathbf{u}_i. \end{array} \tag{6.29}$$

Since all the constraints in problem (6.29) are linear and convex and, if the problem is strictly feasible, there will be a Pareto solution to the game [84].

The definition of Pareto optimality is as follows: for an objective function $\mathbf{f} = [f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x})]$, a design variable vector $\mathbf{x}^* \in \Omega$ is Pareto optimal if and only if there is no vector $\mathbf{x} \in \Omega$, with the characteristics

$$\begin{array}{ll} f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*) & \text{for all} \quad i = 1, \cdots, m \\ \text{and} \quad f_i(\mathbf{x}) < f_i(\mathbf{x}^*) & \text{for at least one} \quad i, 1 \leq i \leq m. \end{array}$$

From the definition, there may exist a set of Pareto optimal points for the coordination problem (6.29). In order to obtain a single best compromised Pareto solution to the game, we consider the worst-case interaction force between the agents and the object. The cooperative game is transferred into a minimax optimization problem

$$\begin{array}{ll} \min\limits_{\mathbf{u}} \max\limits_{i} \{\mathbf{F}(i)\} & \\ \text{Subject to} & \mu_i F_i^n \geq |F_i^t|, \quad i = 1, \cdots, N \\ & F_i^n \geq 0, \quad i = 1, 2, \cdots, N \\ & \overline{\mathbf{u}} = \sum_{i=1}^{N} R \cdot W_i \begin{bmatrix} F_i^n \\ F_i^t \end{bmatrix}. \end{array} \tag{6.30}$$

To solve the minimax problem, we introduce a slack variable g

$$g \geq F_i^n, \qquad i = 1, \cdots, N. \tag{6.31}$$

The minimax problem (6.30) becomes a linear programming problem

$$\begin{array}{ll} \min\limits_{\mathbf{u}} g & \\ \text{Subject to} & \mu_i F_i^n \geq |F_i^t|, \quad i = 1, \cdots, N \\ & F_i^n \geq 0, \quad i = 1, \cdots, N \\ & g \geq F_i^n, \quad i = 1, \cdots, N \\ & \overline{\mathbf{u}} = \sum_{i=1}^{N} R \cdot W_i \begin{bmatrix} F_i^n \\ F_i^t \end{bmatrix}. \end{array} \tag{6.32}$$

The linear programming problem is solved with the standard Simplex method. By solving this problem, we obtain the set of optimal pushing forces.

6.3 Simulation Results

In this section, simulations are performed on a triangular object under two-agent and three-agent pushing. The object is shown in Fig. 6.2. (where $a = 0.2$ m, $b = 0.17$ m, and $\angle A = \pi/6$). We assume the uniform distribution of mass with the mass density $\rho = 200\text{Kg/m}^2$. The mass of the object is $m = 1.7$ Kg. The mass moment of inertia is $I_0 = 0.0037\text{Kg} \cdot \text{m}^2$. Assume the friction coefficient between the object and the surface is $\mu = 0.5$, and the friction coefficient between the agents and the object is $\mu_i = 0.8$. The local coordinate frame is located at the COM and fixed on the object. Assign the world frame at the same location and it is fixed on the supporting plane.

The agents push on the edges AC, BC, and AB respectively. We assign a contact frame to each agent, and the associated contact normal and tangential vectors are

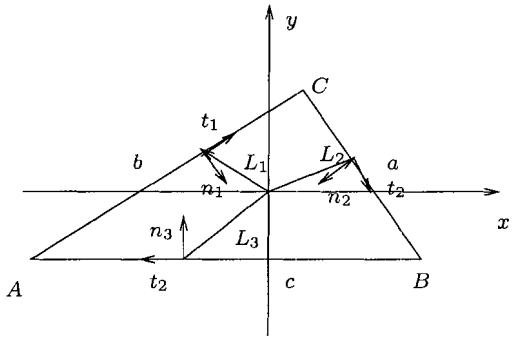


Figure 6.2: Manipulation of triangular object

computed as

$$\begin{array}{ll} \mathbf{n}_1 = (\frac{1}{2}, -\frac{\sqrt{3}}{2})^T & \mathbf{t}_1 = (\frac{\sqrt{3}}{2}, \frac{1}{2})^T \\ \mathbf{n}_2 = (-\frac{\sqrt{3}}{2}, -\frac{1}{2})^T & \mathbf{t}_2 = (\frac{1}{2}, -\frac{\sqrt{3}}{2})^T \\ \mathbf{n}_3 = (0, 1)^T & \mathbf{t}_3 = (-1, 0)^T. \end{array} \tag{6.33}$$

The coordinates of the contact points between the agents and the object are

$$\begin{array}{l} L_1 = (-0.0167 \quad 0.0289) \\ L_2 = (0.0500 \quad 0.00289) \\ L_3 = (-0.0400 \quad -0.0283). \end{array} \tag{6.34}$$

Computing the wrench matrices (6.2) as

$$W_1 = \begin{bmatrix} 0.5000 & 0.8660 \\ -0.8660 & 0.5000 \\ 0.0476 & -0.0333 \end{bmatrix} \tag{6.35}$$

$$W_2 = \begin{bmatrix} -0.8660 & 0.5000 \\ -0.5000 & -0.8660 \\ -0.0278 & 0.0481 \end{bmatrix} \tag{6.36}$$

$$W_3 = \begin{bmatrix} 0 & -1.000 \\ 1.000 & 0 \\ -0.0400 & -0.0283 \end{bmatrix}. \tag{6.37}$$

6.3.1 Coordination under Three-agent Pushing

When we consider the translation and rotation of the object, three agents manipulate the object to follow a predefined trajectory. The desired trajectories on configuration,

velocity, and acceleration are

$$\begin{array}{lll} X_{or}(t) = \sin(t) & \dot{X}_{or}(t) = \cos(t) & \ddot{X}_{or}(t) = -\sin(t) \\ Y_{or}(t) = \cos(t) & \dot{Y}_{or}(t) = -\sin(t) & \ddot{Y}_{or}(t) = -\cos(t) \\ \theta_{or}(t) = \frac{1}{2}\sin(t) & \dot{\theta}_{or}(t) = \frac{1}{2}\cos(t) & \ddot{\theta}_{or}(t) = -\frac{1}{2}\sin(t). \end{array} \tag{6.38}$$

We consider the uncertainties $\delta F_x, \delta F_y, \delta T$ associated with pressure distribution entering the system (6.1) randomly, and the magnitudes are chosen as ten percent of the friction forces and torque. The initial conditions are $X_o(0) = 1, \dot{X}_o = 0, Y_o(0) = 0.5, \dot{Y}_o = 0, \theta_o(0) = 0.2, \dot{\theta}_o = 0$.

Following the procedure in Section 6.2.1, the generalized force control law is designed according to the following steps.

1. Specify
$$K_1 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 8 \end{bmatrix} \qquad K_2 = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$
such that the eigenvalues of the matrix A for the nominal system are $-0.3467, -8.6533, -2.5 + 2.7839i, -2.5 - 2.7839i, -6.7016, -0.2984$.

2. Select two different attenuation levels $\gamma = 0.1$ and $\gamma = 0.05$, respectively. Then, choose the weighting matrices $Q = \text{diag}[100I_3, \quad 100I_3]$, and $R_c = \gamma^2 I$.

3. Solve the Riccati-like equation (6.22) using MATLAB function "are", and we get,
$$P = \begin{bmatrix} 172.2222 & 0 & 0 & 16.6667 & 0 & 0 \\ 0 & 196.4286 & 0 & 0 & 25. & 0 \\ 0 & 0 & 121.25 & 0 & 0 & 6.25 \\ 16.6667 & 0 & 0 & 7.4074 & 0 & 0 \\ 0 & 25 & 0 & 0 & 10.7143 & 0 \\ 0 & 0 & 6.25 & 0 & 0 & 11.25 \end{bmatrix}.$$

4. Compute the generalized force control law as Eqs. (6.20) and (6.21).

Fig. 6.3-6.8 present the simulation results of tracking. In Fig. 6.3-6.4 and Fig. 6.6-6.7, simulation results demonstrate the minimax tracking performance for attenuation levels of $\gamma = 0.05$ and $\gamma = 0.1$, respectively. Fig. 6.5 and Fig. 6.8 show the generalized force under different attenuation levels. According to the simulation results, it is clear that a smaller attenuation level γ yields a better tracking performance than a larger attenuation level γ.

After obtaining the generalized force, a minimax optimization problem is set up as Eq. (6.30) with the wrench matrix as Eq. (6.37). After solving the the optimization problem, we obtain the normal and tangential forces applied to the object by the agents. The results are plotted in Fig. 6.9 and Fig. 6.10 with different attenuation levels of $\gamma = 0.05$ and $\gamma = 0.1$, respectively. The resulting forces satisfy the performance index (6.5).

6.3.2 Coordination under Two-agent Pushing

In this simulation, the motion of the object consists of only a linear translation, without rotation. The desired trajectory is:

$$\begin{array}{l} X_{or}(t) = -\frac{1}{10}t^2 \\ Y_{or}(t) = -\frac{1}{5}t^2 \\ \theta_{or}(t) = 0 \end{array} \tag{6.39}$$

and the corresponding velocities and accelerations are

$$\begin{array}{ll} \dot{X}_{or}(t) = -\frac{1}{5}t & \ddot{X}_{or}(t) = -\frac{1}{5} \\ \dot{Y}_{or}(t) = -\frac{2}{5}t & \ddot{Y}_{or}(t) = -\frac{2}{5} \\ \dot{\theta}_{or}(t) = 0 & \ddot{\theta}_{or}(t) = 0. \end{array} \tag{6.40}$$

Since we consider only translation, the object has two degrees of freedom, hence two agents are enough to control the motion of the object. Two agents push the object

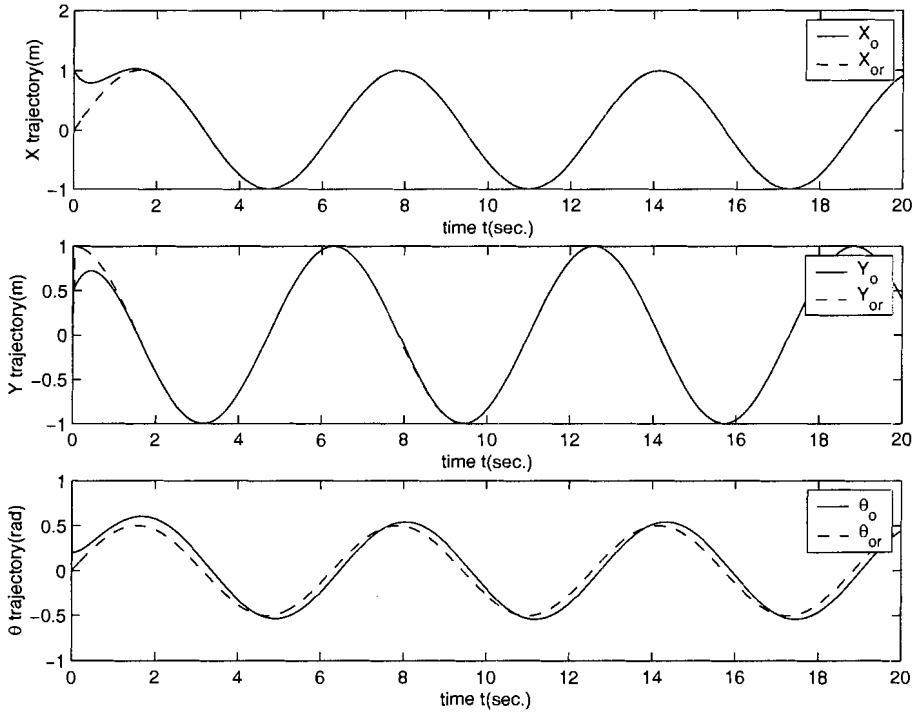


Figure 6.3: H^∞ tracking results of positions ($\gamma = 0.1$)

on edges AC and BC, respectively. In the lower control level, the generalized force control law $\overline{\mathbf{u}}$ is designed in exactly the same way as that in the three-agent case. This is an important feature of hierarchical planning: after the generalized force inputs are designed using H^∞ control, the number of agents and the positions of pushing is rescheduled without redesign of the lower level control law. The coordination is solved using the *Simplex* method on the linear programming problem (6.32). The tracking and optimal distribution of forces are shown in Fig. 6.11-6.13. It is clear that at the beginning, the forces oscillate, and then become stable and close to constant values. This happens because the object is moving along a straight line with a constant acceleration.

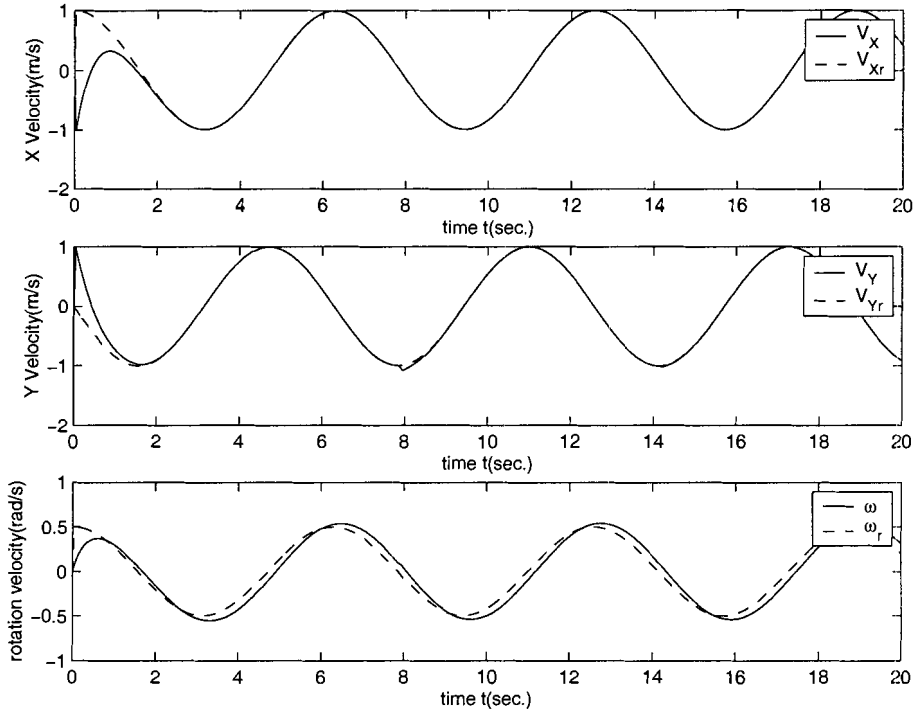


Figure 6.4: H^∞ tracking results of velocities ($\gamma = 0.1$)

6.4 Discussion

This chapter studies the planning problem for the cooperative dynamic pushing with uncertain pressure distribution. The H^∞ control technique has been applied to design the generalized force control law and achieve the tracking performance. A cooperative game is solved to distribute the generalized force between the pushing agents. One major assumption used in this chapter and in Chapter 5 is that the contact positions and the contact friction coefficient between agents and the object are known. Optimally determining the number of agents and locating the contact points dynamically during the manipulation process will improve the performance of the manipulation system.

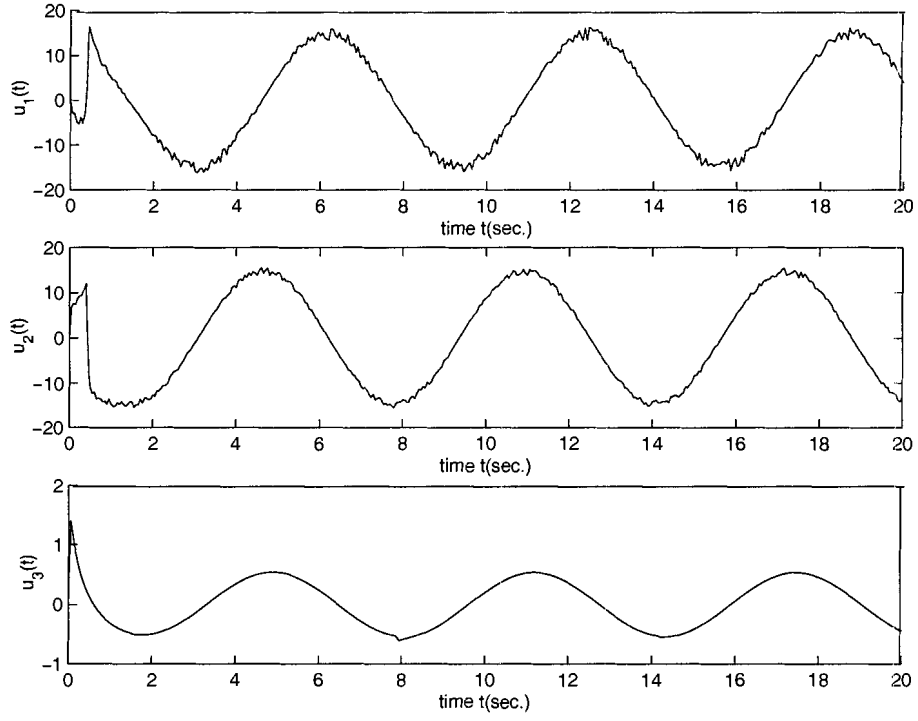


Figure 6.5: Generalized forces $\overline{\mathbf{u}}(t)(\gamma = 0.1)$

In the hierarchical design, redesign of the generalized controller is not required when more pushing agents are introduce to the manipulation process. Because the optimal force distribution problem is solved by static optimization methods performing on each individual set of the generalized force at a fixed time instant, the resulted pushing force may exhibit discontinuities, which may hinder the force/position control implementation. Additionally, because of the usage of the two level planning method, the obtained pushing forces may not be globally optimal. As an alternative, an optimal control approach could be used to solve the tracking problem and coordination problem simultaneously.

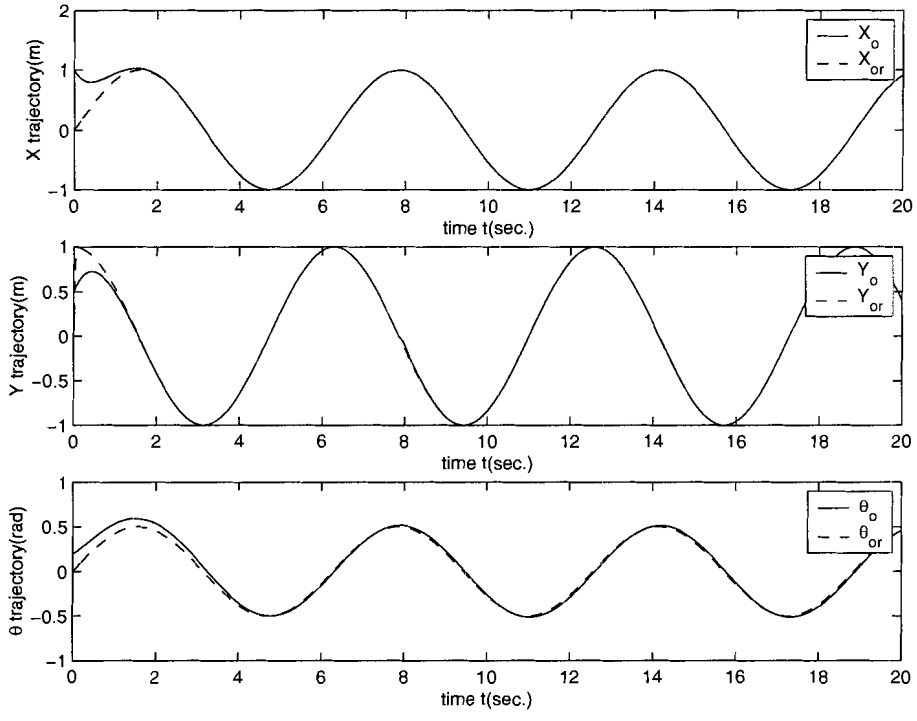


Figure 6.6: H^{∞} tracking results of positions ($\gamma = 0.05$)

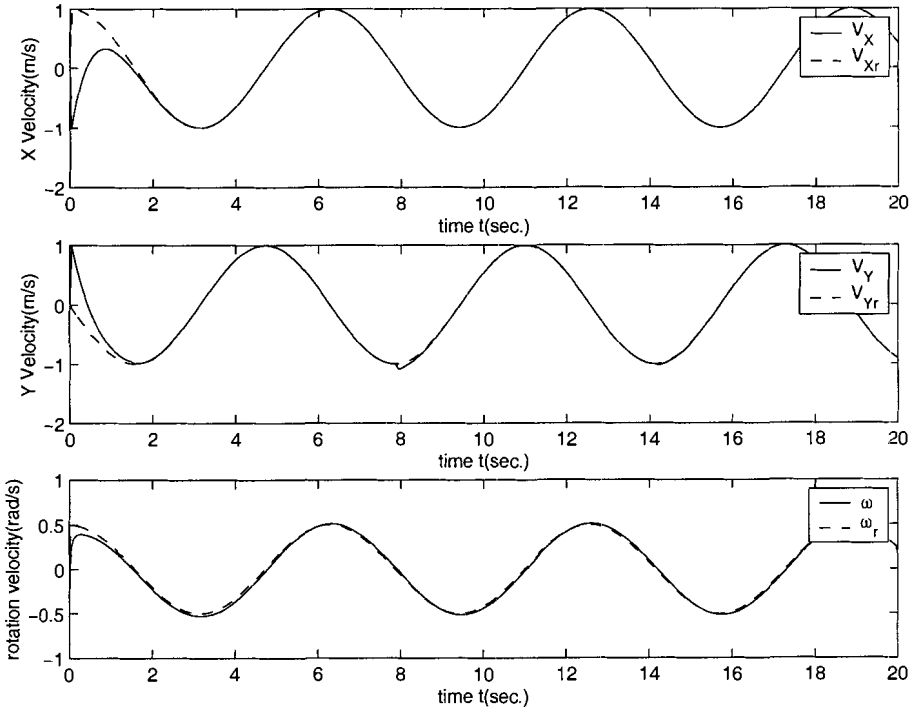


Figure 6.7: H^∞ tracking results of velocities ($\gamma = 0.05$)

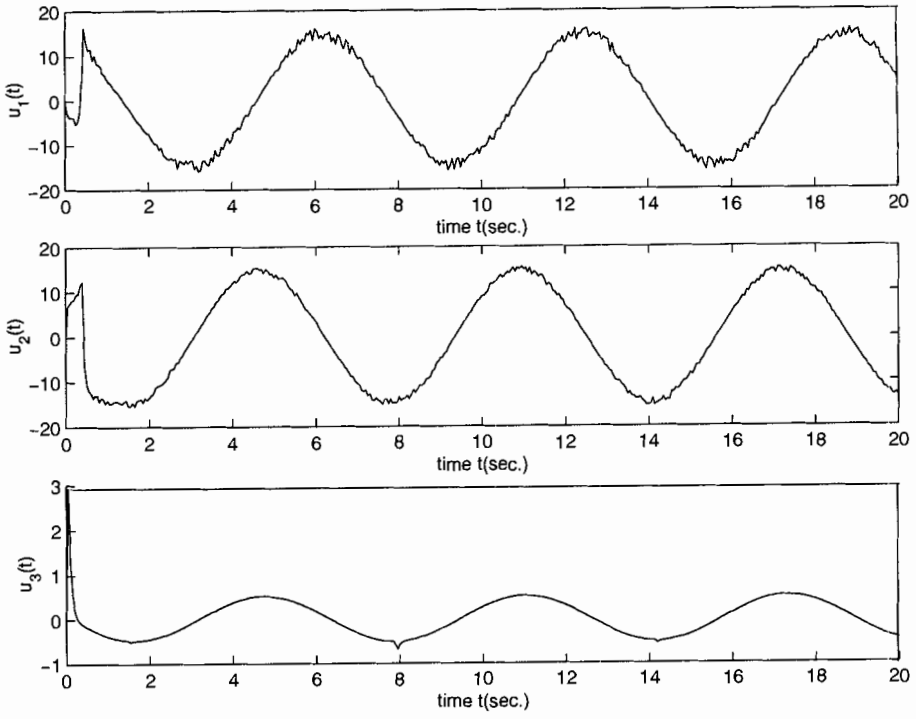


Figure 6.8: Generalized forces $\overline{\mathbf{u}}(t)$ ($\gamma = 0.05$)

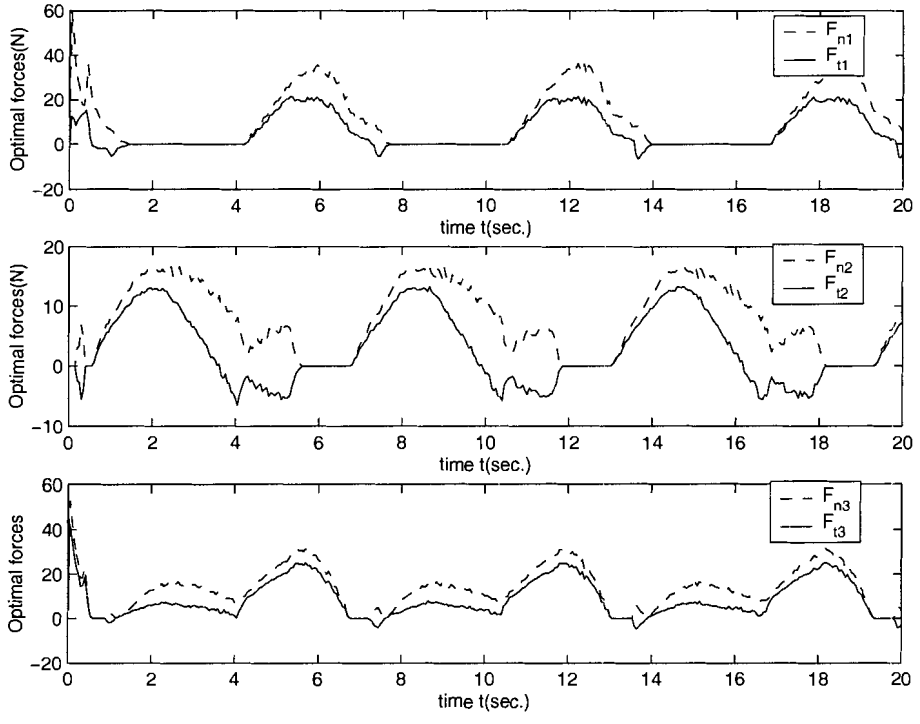


Figure 6.9: Force distribution under three-agent manipulation ($\gamma = 0.1$)

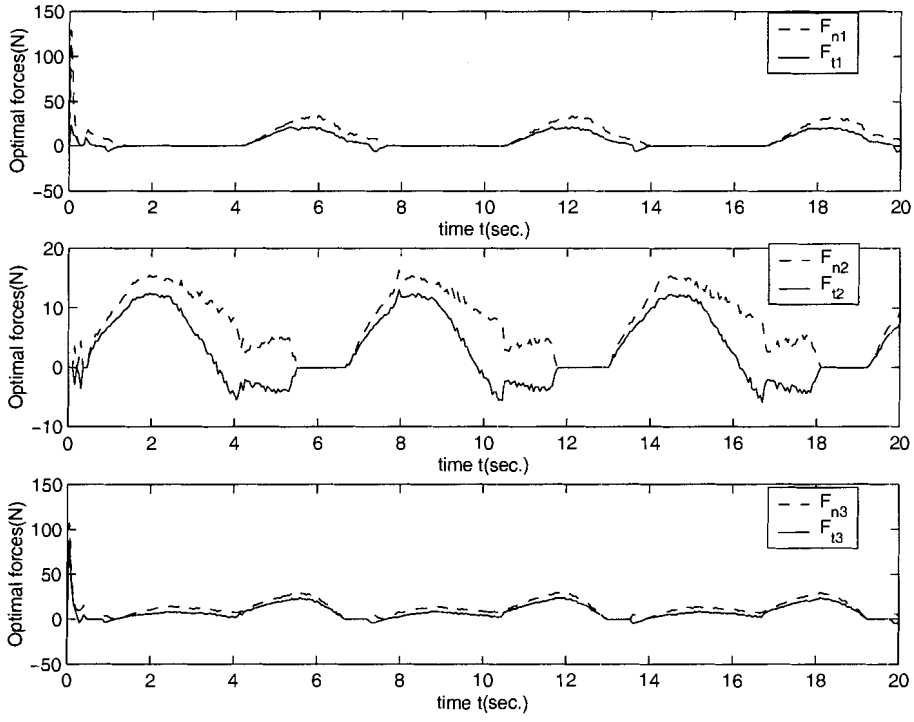


Figure 6.10: Force distribution under three-agent manipulation ($\gamma = 0.05$)

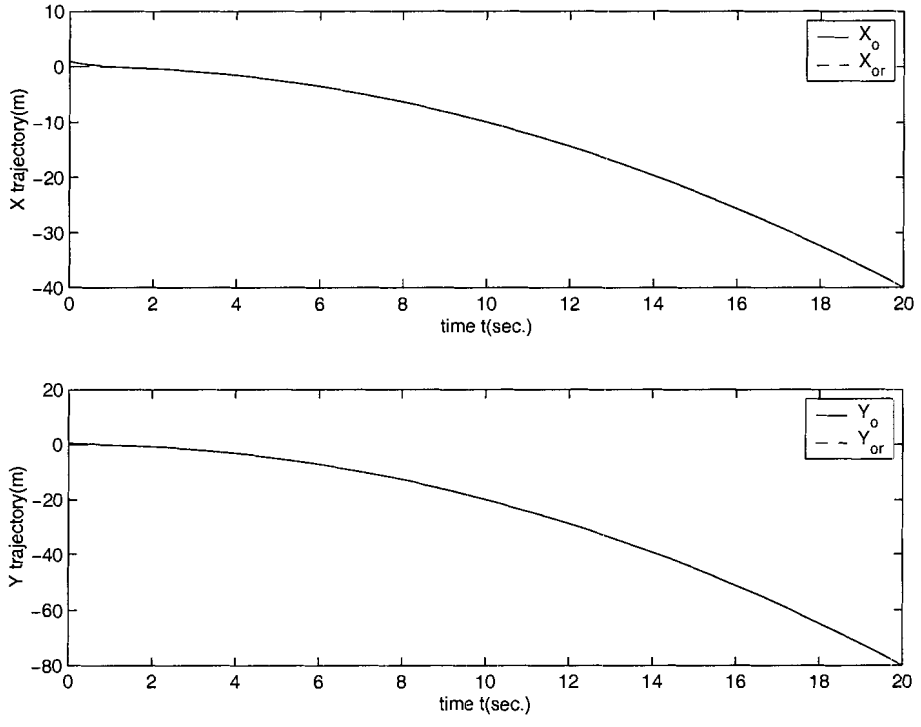


Figure 6.11: H^{∞} tracking results of positions(two-agent) ($\gamma = 0.05$)

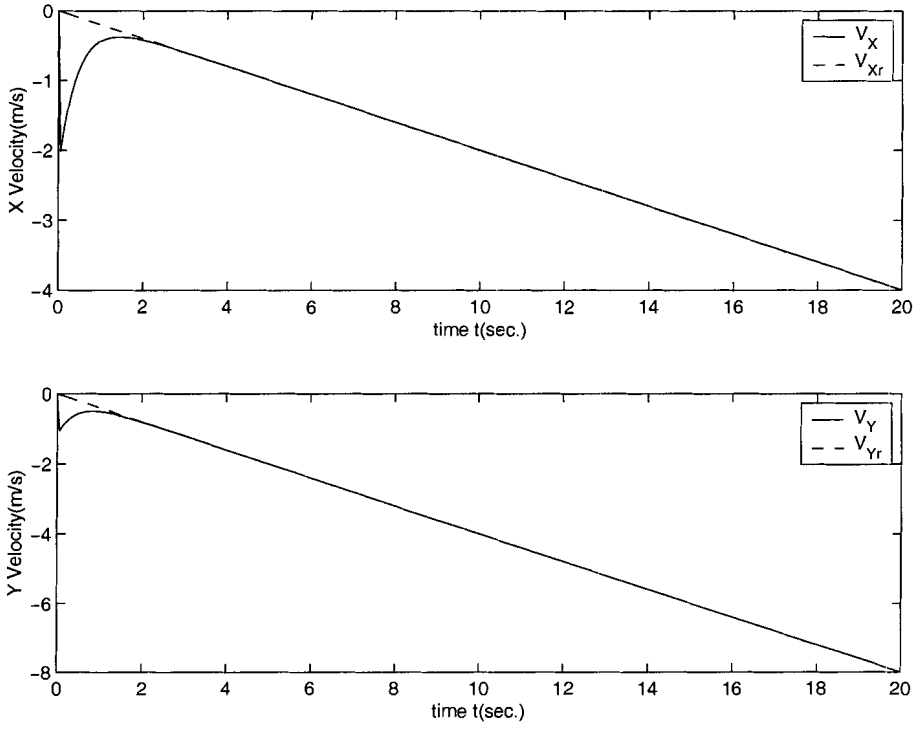


Figure 6.12: H^∞ tracking results of velocities(two-agent) ($\gamma = 0.05$)

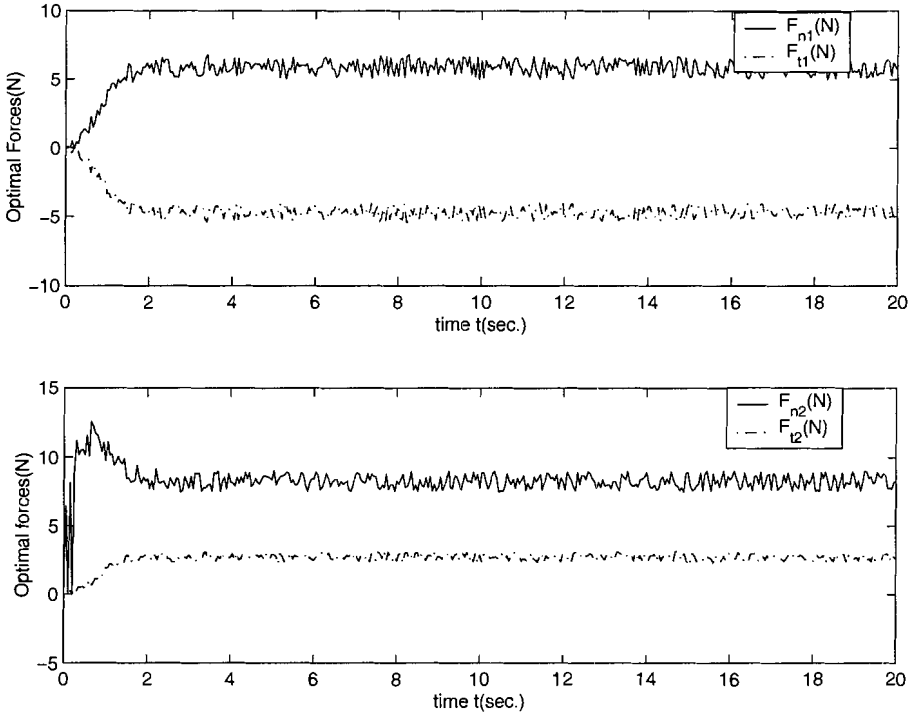


Figure 6.13: Force coordination under two-agent manipulation

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis has focused on the coordinated nonprehensile manipulation for a planar parts transfer task, and has presented the mechanics, control, and planning problems in quasi-static and dynamic settings. Three manipulation methods are proposed, and some of the key results are:

- An pushing agent equipped with two stable pushing primitives can push an object between any two configurations in the plane. The optimal trajectories consist of, at most, two switchings between these two push actions.
- A virtual fence can be formulated during a two-agent distributed push. A virtual fence performs a similar function to a physical fence. A virtual fence can push a convex object between two configurations in the plane, and there exists a fully analytical optimal planner.
- Planning the initial velocities for free sliding objects can be formulated as a free boundary problem. The free boundary value problem can be transformed

into a two point boundary value problem, and solved using nonlinear optimization techniques. This formulation works for objects with general geometrical boundaries.

- Cooperative dynamic pushing can be modeled as a control system. The control problems can be solved by a two-level approach to achieve the tracking performance and minimize the cost.

The results demonstrate that coordination and distribution can increase the flexibility of the nonprehensile manipulation system. By exploring the model of task mechanics, a manipulation task can be achieved by simple robots through cooperation or coordination of simple manipulation primitives. Control theory and optimization play an important role in the study of the planning problem.

7.2 Future Work

This thesis is a first step toward the control and optimization approach for coordinated nonprehensile planar manipulation. The following are some challenging extensions to the problems studied in this thesis.

- **Nonprehensile manipulation in the environment with obstacles:** One straight forward solution is to coordinate agents to perform a manipulation in the free space, and an approach might be: To use boundaries of the environment as passive agents. With the help of passive agents, the class of manipulatable objects increases. Fig. 7.1 illustrates two pushing agents using a wall as a passive agent to orient an object. Recently, Nieuwenhuisen et al. [73] studied the single-robot disc pushing problem in an environment with obstacles. The

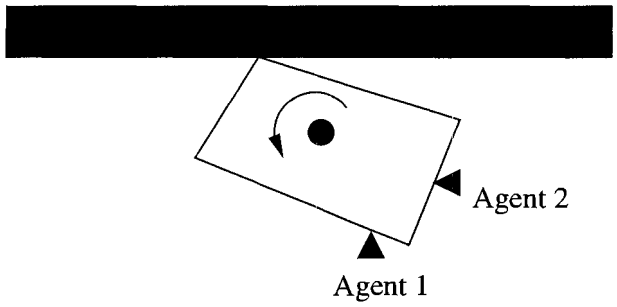


Figure 7.1: Coordinated manipulation between agents and environment

result illustrates that by using the boundaries of the environment, the possibility for the robot to find a push path increases.

- **Incorporate agent dynamics in the acceleration model:** For the acceleration task, instead of considering object dynamics only, a dynamic model of manipulation agents such as robotic hands [26] could be incorporated. The object trajectory planning and control of the manipulator could be achieved simultaneously. Uncertainties on interaction forces between the agents and the object can also be treated in this model. Recently, results on dynamic contact manipulation [93, 94] provide techniques to model the dynamic interaction between the object and the robot, and the object and the environment.

- **Manipulation with more than two primitives:** In Chapter 2, we studied nonprehensile manipulation by an agent equipped with two simple pushing primitives. Exploring the manipulation with more than two primitives (Fig.7.2), and studying the structure of the optimal trajectory would be interesting. It is also possible to study the fault tolerance aspects of the manipulation system. When one or more pushing primitives are disabled, what is the performance of the manipulation system?

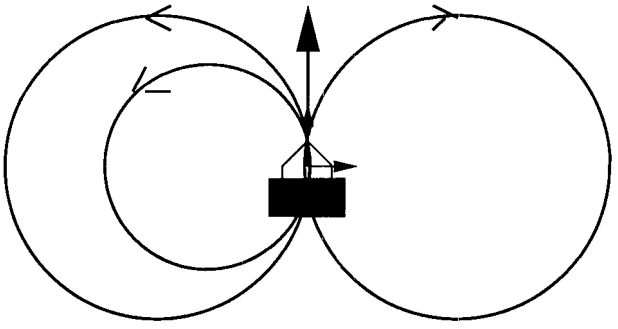


Figure 7.2: Coordinated manipulation with four push primitives

- **Dynamic manipulation by surface tilting:** One challenging problem is whether the configuration of the object is controllable by tilting the surface. For two given configurations, does a sequence of tilting actions exist to connect them? The manipulation process is illustrated in Fig. 7.3. Fig. 7.3(a) shows that an object will land on the tilting surface with a known orientation and velocities; (b) shows the sequence of tilting actions; (c) indicates that the object arrives at the goal configuration after the sequence of tilting. The tilting of the surface can be achieved by a Stewart platform or by using two actuators. Christiansen and Goldberg [25] have considered the surface tilting problem, and consider the case of finite state and action spaces where actions can be modelled as Markov transitions.

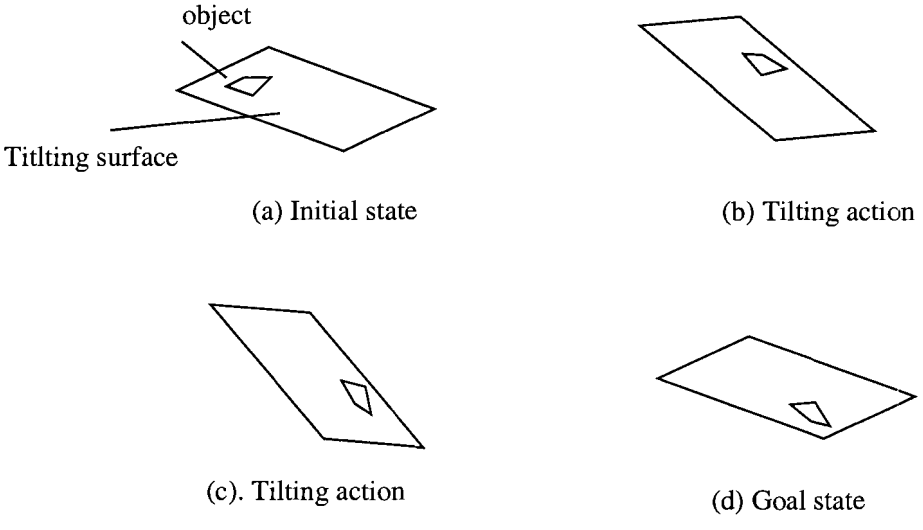


Figure 7.3: Manipulation by dynamic tilting

Appendix A

Computation Methods for Polygonal Objects

A.1 Computation of Mass, Moment of Inertia and Friction

In general, there is no analytical formulation for calculating the mass, mass moment of inertia, friction forces and torque for an object with an arbitrary boundary. One possible solution is to partition the object into a number of simple areas, such as triangles, and compute these quantities over these simple areas first, then add them together. In the next two sections, computation methods are discussed for convex polygonal objects.

A.1.1 Triangular Partition of a Polygonal Object

Consider a convex polygon as shown in Fig. A.1. The vertices are labeled by $V_1, V_2, V_3, \cdots, V_n$, and the coordinates of V_i are given by (X_i, Y_i) in the global frame

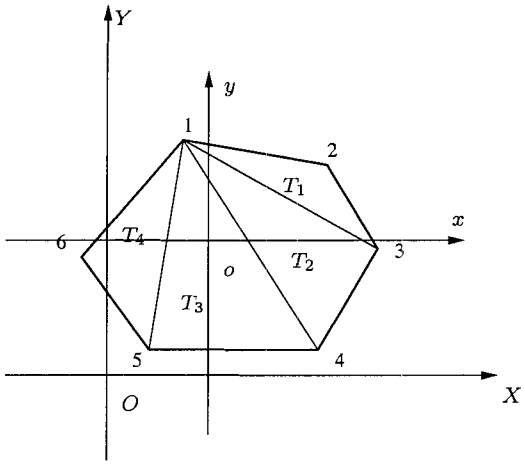


Figure A.1: Geometry of a polygonal object

XOY. The convex polygon is divided into $n-2$ triangles as

$$V_1V_2V_3, V_1V_3V_4, V_1V_4V_5, \cdots, V_1V_kV_{k+1}, \cdots, V_1V_{n-1}V_n$$

which are labeled as $T_1, T_2, \cdots, T_{n-2}$.

To plan the velocities for a free sliding object, it requires the mass m, mass moment of inertia I_0, and center of mass. All these quantities are computed by integrating over the triangular areas of $T_1, T_2, \cdots, T_{n-2}$ individually.

The mass of the object is calculated as

$$\begin{aligned} m &= \int\int_{\bar{A}} \rho_1(X,Y)dXdY \\ &= \sum_{i=1}^{n-2} \int\int_{T_i} \rho_1(X,Y)dXdY \end{aligned} \tag{A.1}$$

where the mass pressure distribution function is represented by $\rho_1(X,Y)$.

The center of mass (X_o, Y_o) is computed as

$$X_o = m_x/m \qquad Y_o = m_y/m \tag{A.2}$$

with m_x and m_y as

$$\begin{aligned} m_x &= \int\int_{\bar{A}} X\rho_1(X,Y)dXdY \\ &= \sum_{i=1}^{n-2} \int\int_{T_i} X\rho_1(X,Y)dXdY \end{aligned} \tag{A.3}$$

$$\begin{aligned} m_y &= \int\int_{\bar{A}} Y\rho_1(X,Y)dXdY \\ &= \sum_{i=1}^{n-2} \int\int_{T_i} Y\rho_1(X,Y)dXdY. \end{aligned} \tag{A.4}$$

After assigning the local frame xoy at the center of mass as (X_0, Y_0), the coordinates of vertices for the polygon are represented in the local frame as

$$x_i = X_i - X_o \quad y_i = Y_i - Y_o \tag{A.5}$$

and the mass distribution function becomes $\rho(x,y) = \rho_1(x + X_o, y + Y_o)$ in the xoy frame. The mass moment of inertia I_0 with respect to the center of mass are computed as

$$I_0 = I_x + I_y \tag{A.6}$$

where

$$\begin{aligned} I_x &= \int\int_{\bar{A}} y^2\rho(x,y)dxdy \\ &= \sum_{i=1}^{n-2} \int\int_{T_i} y^2\rho(x,y)dxdy \end{aligned} \tag{A.7}$$

$$\begin{aligned} I_y &= \int\int_{\bar{A}} x^2\rho(x,y)dxdy \\ &= \sum_{i=1}^{n-2} \int\int_{T_i} x^2\rho(x,y)dxdy. \end{aligned} \tag{A.8}$$

In order to compute the mass, center of mass, and mass moment of inertia for a polygonal object, double integration of Eqs. (A.1), (A.3), (A.4), (A.7), and (A.8) are required. A similar partition strategy also works for curved parts.

The next section will discuss a numerical integration procedure for partitioned objects.

A.1.2 Numerical Double Integration Over Triangular Region

For a known partition as shown in the previous section, integration over the contact area is carried out by integration over each individual triangular area. Almost all

existing double integration routines work only for objects with rectangular boundaries. For objects with non-rectangular boundaries, transformations need to be performed before the integration.

Consider a triangular object with vertices at $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, We perform the first transformation as follows

$$\begin{array}{l} x = x_1(-\frac{1}{2}(\xi+\eta)) + x_2(\frac{1}{2}(1+\xi)) + x_3(\frac{1}{2}(1+\eta)) \\ y = y_1(-\frac{1}{2}(\xi+\eta)) + y_2(\frac{1}{2}(1+\xi)) + y_3(\frac{1}{2}(1+\eta)). \end{array} \tag{A.9}$$

This transformation maps a triangle with vertices (x_i, y_i) into a standard triangle with vertices located at $(-1,-1), (1,-1), (-1,1)$ in the $\xi\eta$ plane. The Jacobian associated with the transformation is calculated as

$$\begin{aligned} J &= \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix} \\ &= \begin{vmatrix} \frac{1}{2}(-x_1+x_2) & \frac{1}{2}(-y_1+y_2) \\ \frac{1}{2}(-x_1+x_3) & \frac{1}{2}(-y_1+y_3). \end{vmatrix} \end{aligned} \tag{A.10}$$

In order to illustrate the transformation, we apply it to the force and torque equations (4.15), (4.16), and (4.17). After substituting Eqs. (A.9) and (A.10) into the integrands become $f_x(\xi,\eta), f_y(\xi,\eta), t(\xi,\eta)$, and we get the following integration formulas

$$\begin{array}{l} F_X = -\mu \cdot g \int_{-1}^{1} \int_{-1}^{-\xi} J \cdot f_x(\xi,\eta) d\eta d\xi \\ F_Y = -\mu \cdot g \int_{-1}^{1} \int_{-1}^{-\xi} J \cdot f_y(\xi,\eta) d\eta d\xi \\ T = -\mu \cdot g \int_{-1}^{1} \int_{-1}^{-\xi} J \cdot t(\xi,\eta) d\eta d\xi. \end{array} \tag{A.11}$$

In this way, integrations over an arbitrary triangular area are transformed into integrations over a standard triangular area. The next step is to transfer the integration over a triangular area into the integration over a rectangular area. For this purpose, we perform another transformation described by the following:

$$\eta = (-\xi + 1)z - 1 \tag{A.12}$$

when $z = 1, \eta = -\xi$ and when $z = 0, \eta = -1$ as required in Eq. (A.11). By differentiating Eq. (A.12), we obtain

$$d\eta = (-\xi + 1)dz. \tag{A.13}$$

Substituting Eq. (A.12) into integrands of $f_x(\xi, \eta)$, $f_y(\xi, \eta)$, and $t(\xi, \eta)$ in Eq. (A.11), we will get a new set of integrands $f_{x1}(\xi, z)$, $f_{y1}(\xi, z)$, and $t_1(\xi, z)$. By considering the Jacobian (A.13), the integration formulas (A.11) become

$$\begin{aligned} F_X &= -\mu \cdot g \int_{-1}^{1} \int_{0}^{1} J(-\xi + 1) \cdot f_{x1}(\xi, z) dz d\xi \\ F_Y &= -\mu \cdot g \int_{-1}^{1} \int_{0}^{1} J(-\xi + 1) \cdot f_{y1}(\xi, z) dz d\xi \\ T &= -\mu \cdot g \int_{-1}^{1} \int_{0}^{1} J(-\xi + 1) \cdot t_1(\xi, z) dz d\xi. \end{aligned} \tag{A.14}$$

After these two transformations, the integrations of F_X, F_Y, and T over a triangular area are transformed into integrations over a rectangular area, and existing double integral routines such as Gaussian quadrature methods could be used to compute this integration. Fig. A.2. illustrates the schematic of the transformations described by Eq. (A.9) and Eq. (A.12).

After we integrate the frictions and torque over each individual triangular area, the overall friction forces and torque acting on the general polygonal object are computed by summation. The integrations of Eqs. (A.1), (A.3), (A.4), (A.7), and (A.8) are performed in the same manner as discussed above.

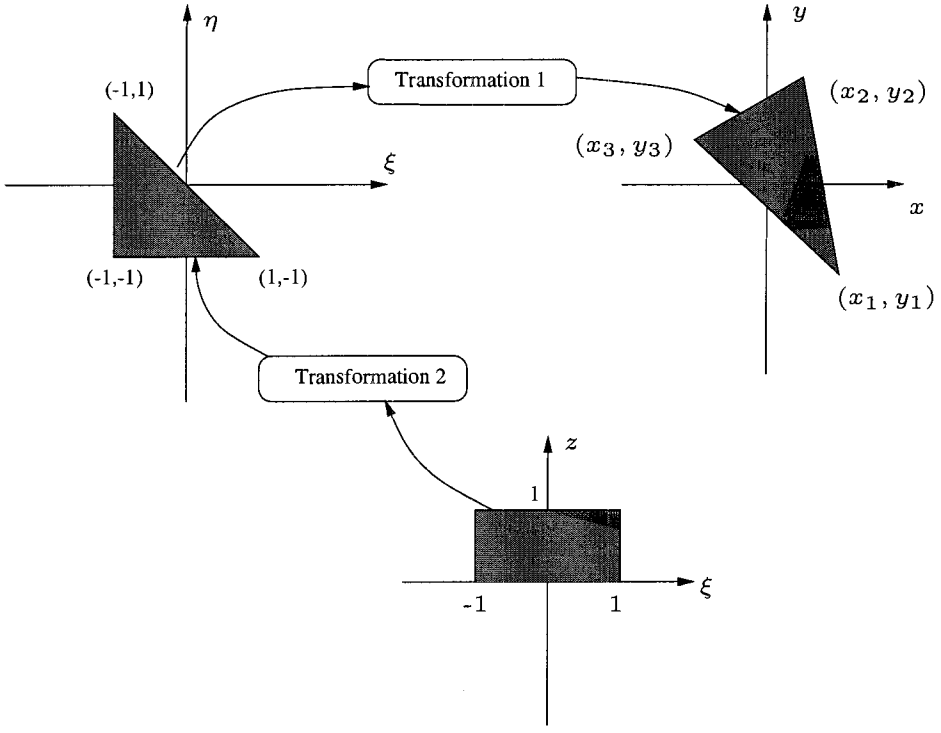


Figure A.2: Schematic of the transformation

Bibliography

[1] P. Agarwal, J. Latombe, R. Motwani, and P. Raghavan. Nonholonomic path planning for pushing a disk among obstacles. In *Proc. 1997 IEEE Intl. Conf. on Robotics and Automation*, pages 3124 – 3129, April 1997.

[2] Y. Aiyama, M. Inaba, and H. Inoue. Pivoting: A new method of graspless manipulation of object by robot fingers. In *Proc. 1993 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 136–143, July 1993.

[3] S. Akella and M.T. Mason. Posing polygonal objects in the plane by pushing. *Intl. J. of Robotics Research*, 17(1):70–88, 1998.

[4] S. Akella and M.T. Mason. Using partial sensor information to orient parts. *Intl. J. of Robotics Research*, 18(10):963–997, 1999.

[5] S. Akella and M.T. Mason. Orienting toleranced polygonal parts. *Intl. J. of Robotics Research*, 19(12):1147–1170, 2000.

[6] S. Akella, W. Huang, K.M. Lynch, and M.T. Mason. Parts feeding on a conveyor with a one-joint robot. *Algorithmica*, 26(3/4):313–314, 2000.

[7] U.M. Ascher and L.R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations.* SIAM, Philadelphia, 1998.

[8] U.M. Ascher, R.D. Russell, and R.M.M Mattheij. *Numerical solution of boundary value problems for ordinary differential equations.* Prentice-Hall, Englewood Cliffs, N.J., 1988.

[9] D.J. Balkcom and M.T. Mason. Time optimal trajectories for bounded velocity differential drive vehicles. *Intl. J. of Robotics Research*, 21(3):199–217, 2002.

[10] Z. Balorda. Automatic planning of robot pushing operations. In *Proc. 1993 IEEE Intl. Conf. on Robotics and Automation*, pages 732–737, May 1993.

[11] Z. Balorda and T. Bajd. Reducing positioning uncertainty of objects by robot pushing. *IEEE Trans. on Robotics and Automation*, 10(4):535–541, 1994.

[12] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and moyion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[13] T. Basar and P. Bernhard. *H^∞-optimal control and related minimax design problems-a dynamic game approach.* Birkhauser, Boston, second edition, 1995.

[14] T. Basar and G.J. Olsder. *Dynamic noncooperative game theory.* Academic Press, New York, 1995.

[15] J.D. Bernheisel and K.M. Lynch. Stable transport of assemblies: pushing stacked parts. *IEEE Transactions on Automation Science and Engineering*, 1(2):163–168, 2004.

[16] RP Berretty, K. Goldberg, MH Overmars, and AF van der Stappen. Computing fence designs for orienting parts. *Computational geometry-theory and applications*, 10(4):249–262, 1998.

[17] RP Berretty, MH Overmars, and AF van der Stappen. Orienting polyhedral parts by pushing. *Computational geometry-theory and applications*, 21(1-2): 21–38, 2002.

[18] C. Black and K. Lynch. Planning and control for planar batting and hopping. In *Proceedings of the 36th Annual Allerton Conference on Communications, Control, and Computing*, pages 724–733, Sept. 1998.

[19] S.J. Blind, C.C. McCullough, S. Akella, and J. Ponce. Manipulating parts with an array of pins: A method and a machine. *Intl. J. of Robotics Research*, 20 (10):808–816, 2001.

[20] R. C. Brost. Automatic grasp planning in the presence of uncertainty. *Intl. J. of Robotics Research*, 2(1):3–17, 1988.

[21] R.C. Brost. Dynamic analysis of planar manipulation tasks. In *Proc. 1992 IEEE Intl. Conf. on Robotics and Automation*, pages 2247–2254, 1992.

[22] R.G. Brown and J.S. Jennings. A pusher/steerer model for strongly cooperative mobile robot manipulation. In *Proc. 1995 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 562–568, 1995.

[23] M. Buss, H. Hashimoto, and J.B. Moore. Dexterous hand grasping force optimization. *IEEE Trans. on Robotics and Automation*, 12(3):406–418, 1996.

[24] J.F. Canny and K.Y. Goldberg. Risc industrial robotics: recent results and open problems. In *Proc. 1994 IEEE Intl. Conf. on Robotics and Automation*, pages 1951–1958, April 1994.

[25] A.D. Christiansen and K.Y. Goldberg. Comparing 2 algorithms for automatic planning by robots in stochastic environments. *ROBOTICA*, 13(6):565–573, 1995.

[26] A.A. Cole, P. Hsu, and S.S. Sastry. Dynamic control of sliding by robot hands for regrasping. *IEEE Trans. on Robotics and Automation*, 8(1):42–52, 1992.

[27] C. Davis. Theory of positive linear dependence. *Am. J. Mathematics*, 76:733–746, 1954.

[28] J.E. Dennis and JR.R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1983.

[29] D.R. Donald, J. Jennings, and D. Rus. Information invariant for distributed manipulation. *Intl. J. of Robotics Research*, 16(5):673–702, 1997.

[30] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[31] M.A. Erdmann. An exploration of nonprehensile two-palm manipulation. *Intl. J. of Robotics Research*, 17:485–503, 1998.

[32] M.A. Erdmann and M.T. Mason. An exploration of sensorless manipulation. *IEEE Trans. on Robotics and Automation*, 4:369–379, 1988.

[33] P. E. GIll, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, London, UK, 1981.

[34] K.Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10 (3):201–225, Aug. 1993.

[35] S. Goyal, A. Ruina, and J.P. Papadopoulos. Planar sliding with dry friction. part 1. limit surface and moment function. *Wear*, 143:307–330, 1991.

[36] S. Goyal, A. Ruina, and J.P. Papadopoulos. Planar sliding with dry friction. part 2. dynamics of motion. *Wear*, 143:352–331, 1991.

[37] I. Han and S. Park. Impulsive motion planning for posing and orienting a polygonal part. *Intl. J. of Robotics Research*, 20(3):249–262, 2001.

[38] K. Harada, T. Kawashima, and M. Kaneko. Rolling based manipulation under neighborhood equilibrium. *Intl. J. of Robotics Research*, 21(5):463–474, 2002.

[39] Y.C. Ho and A.E. Bryson Jr. *Applied optimal control.* Hemisphere-wiley, New York, 1975.

[40] W.H. Huang. *Impulsive manipulation.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1997.

[41] W.H. Huang and M.T. Mason. Mechanics, planning, and control for tapping. *Intl. J. of Robotics Research*, 19(10):883–894, 2000.

[42] A.Yu. Ishlinskii, B.N. Sokolov, and F.L. Chernousko. Motion of plane bodies with dry friction. *Izv. An SSSR. Mekhanika Tverdogo Tela*, 16(4):17–28, 1981.

[43] H.B. Keller. *Numerical methods for two-point boundary-value problems.* Blaisdell Publishing Company, London, 1968.

[44] H.K. Khalil. *Nonlinear systems, 3rd ed.* Prentice Hall, NJ, Saddle River, N.J., 2002.

[45] D.J. Kriegman. Let them fall where they may: Capture regions of curved objects and polyhedra. *Intl. J. of Robotics Research*, 16(4):448–472, 1997.

[46] M. Kristic, I. Kanellakopoulos, and P. Kokotovic. *Nonlinear and adaptive control design.* John Wiley & Sons, Inc., New York, 1995.

[47] M. Kurisu and T. Yoshikawa. Tracking control for an object in pushing operation. *Journal of Robotic Systems*, 14(10):729–739, 1997.

[48] J. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[49] J.P. Laumond(Editor). *Robot Motion Planning and Control.* Springer, Berlin, 1998.

[50] Q. Li and S. Payandeh. Modeling and analysis of dynamic planar multi-agent manipulation. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 200–205, July 2001.

[51] Q. Li and S. Payandeh. Centralized cooperative planning for dynamic multi-agent planar manipulation. In *Proc. 2002 IEEE Intl. Conf. on Decision and Control*, pages 2836–2841, Dec 2002.

[52] Q. Li and S. Payandeh. Planning for dynamic multi-agent planar manipulation with uncertainty: A game theoretic approach. *IEEE Trans. on Systems, Man, and Cybernetics, Special Issue on Collective Intelligence*, 33(5):620–626, 2003.

[53] Q. Li and S. Payandeh. Planning velocities of free sliding objects for dynamic manipulation. In *Proc. 2003 IEEE Intl. Conf. on Robotics and Automation*, pages 3594–3599, September 2003.

[54] Q. Li and S. Payandeh. Multi-agent cooperative manipulation with uncertainty: A neural net-based game theoretic approach. In *Proc. 2003 IEEE Intl. Conf. on Robotics and Automation*, pages 3607–3612, September 2003.

[55] Q. Li and S. Payandeh. Planning for dynamic multi-agent planar manipulation with uncertainty: A game theoretic approach. In *the 2003 American Control Conference*, pages 2193–2198, June 2003.

[56] Q. Li and S. Payandeh. Distributed manipulation systems: A review from multi-agent systems. In *The 11th International Conference on Advanced Robotics(ICAR2003)*, pages 846–851, June 2003.

[57] Q. Li and S. Payandeh. Planning velocities of free sliding objects as free boundary value problem. *International Journal of Robotic Research*, 23(1):69–88, 2004.

[58] Q. Li and S. Payandeh. Unconstrained dynamic planar manipulation with one joint manipulator. In *Proc. 2005 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 132–137, Aug. 2005.

[59] Q. Li and S. Payandeh. Optimal control approach to stable push planning for a class of ability-limited robots. *Robotics and Autonomous Systems(submitted)*, 2005.

[60] Q. Li and S. Payandeh. An approach for object manipulation using cooperative agents. In *Proc. 2006 IEEE Intl. Conf. on Robotics and Automation*, May 2006.

[61] Q. Li and S. Payandeh. Manipulation of convex objects via cooperative push. *International Journal of Robotic Research(submitted)*, 2006.

[62] D.G. Luenberger. *Linear and nonlinear programming.* Addison-Wesley Publishing Co., Reading, MA, 1984.

[63] J.E. Luntz, W. Messner, and H. Choset. Distributed manipulation using discrete actuator arrays. *Intl. J. of Robotics Research*, 20(7):553–583, 2001.

[64] K.M. Lynch. Locally controllable manipulation by stable pushing. *IEEE Trans. on Robotics and Automation*, 15(2):318–327, 1999.

[65] K.M. Lynch and M.T. Mason. Stable pushing: Mechanics, controllability, and planning. *Intl. J. of Robotics Research*, 15(6):533–556, 1996.

[66] K.M. Lynch and M.T. Mason. Dynamic nonprehensile manipulation: Controllability, planning, and experiments. *Intl. J. of Robotics Research*, 18(1):64–92, 1999.

[67] M. T. Mason. Dynamic manipulation. In *Proc. 1993 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 152–159, 1993.

[68] M.T. Mason. Mechanics and planning of manipulator pushing operations. *Intl. J. of Robotics Research*, 5(3):53–71, 1986.

[69] M.T. Mason. How to push a block along a wall. In *NASA Conference on Space Telerobotics*, Jan 1989.

[70] M.T. Mason. *Mechanics of robotic manipulation.* MIT press, Cambridge, MA, 2001.

[71] M. Moll, K. Goldberg, M. Erdmann, and R. Fearing. Aligning parts for micro assemblies. *Assembly Automation*, 22(1):46–54, 2002.

[72] TD Murphey and JW. Burdick. Feedback control methods for distributed manipulation systems that involve mechanical contacts. *Intl. J. of Robotics Research*, 23((7-8)):763–781, 2004.

[73] D. Nieuwenhuisen, A.F. van der Stappen, and M.H. Overmars. Path planning for pushing a disk using compliance. In *Proc. 2005 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 714–720, 2005.

[74] J. Nocedal and S. J. Wright. *Numerical optimization.* Springer-Verlag, New York, 1999.

[75] H. Noriega, S. Payandeh, and K. Gupta. Distributed overhead hybrid pins for part orienting: An exploratory study. In *Proceedings of IEEE International Conference on Advanced Robotics*, pages 1819–1825, July 2003.

[76] Y. Okawa and K. Yokoyama. Control of a mobile robot for the push-a-box operation. In *Proc. 1992 IEEE Intl. Conf. on Robotics and Automation*, pages 761–766, May 1992.

[77] C.B. Partridge and M.W. Spong. Control of planar rigid body sliding with impacts and friction. *Intl. J. of Robotics Research*, 19(4):336–348, 2000.

[78] S. Payandeh. On kinematic geometry for multi-agent contacting system. In *Proceedings of IEEE International Conference on Advanced Robotics*, pages 1831–1837, July 2003.

[79] S. Payandeh and M. Saif. Force and fine-position control of multiple planar robotics mechanism. In *Proceedings of the COMCON 3*, pages 565–575, October 1991.

[80] G.A.S. Pereira, M.F.M. Campos, and V. Kumar. Decentralized algorithms for multi-robot manipulation via caging. *Intl. J. of Robotics Research*, 23(7-8): 783–795, 2004.

[81] M.A. Peshkin and A.C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Trans. on Robotics and Automation*, 4(5):524–531, 1988.

[82] M.A. Peshkin and A.C. Sanderson. The motion of a pushed, sliding workpiece. *IEEE Trans. on Robotics and Automation*, 4(6):569–598, 1988.

[83] E. R. Pinch. *Optimal Control and the Calculus of Variations.* Oxford University Press, Oxford, New York, 1993.

[84] E. Rasmusen. *Games and information : an introduction to game theory.* Blackwell Inc., Malden, Mass., 3rd ed. edition, 2001.

[85] A. Reeds and R.A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[86] N. Rezzoug and P. Gorce. Dynamic control of pushing operations. *Robotica*, 17: 613–620, 1999.

[87] E. Rimon and A. Blake. Caging 2d bodies by 1-parameter two-fingered gripping systems. In *Proc. 1996 IEEE Intl. Conf. on Robotics and Automation*, pages 1458–1464, April 1996.

[88] E.J. Routh. *Dynamics of a system of rigid bodies.* Dover Publications, Inc., New York: Dover, 7th ed. edition, 1960.

[89] D. Rus. Coordinated manipulation of objects. *Algorithmica*, 19(1):129–147, 1997.

[90] S. Rusaw, K. Gupta, and S. Payandeh. Flexible part orienting using rotational direction and force measurements. *Intl. J. of Robotics Research*, 20(6):484–505, 2001.

[91] A. Salvarinov and S. Payandeh. Flexible part feeder: Manipulating parts on conveyer belt by active fence. In *Proc. 1998 IEEE Intl. Conf. on Robotics and Automation*, pages 544–549, May 1998.

[92] M.W. Spong. Impact controllability of an air hockey puck. *Systems and control letters*, 42(5):333–345, 2001.

[93] S. S. Srinivasa, M.A. Erdmann, and M.T. Mason. Control synthesis for dynamic contact manipulation. In *Proc. 2005 IEEE Intl. Conf. on Robotics and Automation*, pages 2523–2528, April 2005.

[94] S. S. Srinivasa, M.A. Erdmann, and M.T. Mason. Using projected dynamics to plan dynamic contact manipulation. In *Proc. 2005 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 3618– 3623, Aug. 2005.

[95] J. Stoer and R. Bulirsch. *Introduction to numerical analysis, Translated by R. Bartels, W. Gautschi, and C. Witzgall.* Springer, Berlin, 1993.

[96] A. Stoorvogel. *The H^∞ control problem: A State Space Approach.* Prentice-Hall, Upper Saddle River, NJ, 1992.

[97] A. Sudsang, F. Rothganger, and J. Ponce. Motion planning for disc-shaped

robots pushing a polygonal object in the plane. *IEEE Trans. on Robotics and Automation*, 18(4):550–562, 2002.

[98] T. Tabata and Y. Aiyama. Tossing manipulation by 1 degree-of-freedom manipulator. In *Proc. 2001 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 132–137, Nov. 2001.

[99] T. Tabata and Y. Aiyama. Passing manipulation by 1 degree-of-freedom manipulator - catching manipulation of tossed object without impact. In *Proc. 2003 IEEE International Symposium on Assembly and Task Planning*, pages 181–186, July 2003.

[100] G.J. Toussaint, T. Basar, and F. Bullo. Tracking for nonlinear underactuated surface vessels with generalized forces. In *IEEE International Conference on Control Applications*, pages 181–186, Sept. 2000.

[101] J.C. Trinkle and D.C. Zeng. Prediction of the quasistatic planar motion of a contacted rigid body. *IEEE Trans. on Robotics and Automation*, 11(2):229–246, 1995.

[102] K. Voyenli and E. Eriksen. On the motion of an ice hockey puck. *American Journal of Physics*, 53(12):1149–1153, 1985.

[103] Y. Wang and M.T. Mason. Two dimensional rigid-body collisions with friction. *Journal of applied mechanics*, 59(3):635–642, 1992.

[104] Z. Wang and V. Kumar. Object closure and manipulation by multiple cooperating mobile robots. In *Proc. 2002 IEEE Intl. Conf. on Robotics and Automation*, pages 394–399, May 2002.

[105] J. Wiegley, K. Goldberg, M. Peshkin, and M. Brokowski. A complete algorithm for designing passive fences to orient parts. In *Proc. 1996 IEEE Intl. Conf. on Robotics and Automation*, pages 1133–1139, May 1996.

[106] M.T. Zhang and K. Goldberg. Gripper point contacts for part alignment. *IEEE Trans. on Robotics and Automation*, 18(6):902–910, 2002.

[107] M.T. Zhang, K. Goldberg, G. Smith, R.P. Berretty, and M. Overmars. Pin design for part feeding. *Robotica*, 19(6):695–702, 2001.

[108] R. Zhang and K. Gupta. Automatic orienting of polyhedra through step devices. In *Proc. 1998 IEEE Intl. Conf. on Robotics and Automation*, pages 550–556, May 1998.

[109] C. Zhu, Y. Aiyama, and T. Arai. Releasing manipulation with learning control. In *Proc. 1999 IEEE Intl. Conf. on Robotics and Automation*, pages 2793–2798, May 1999.