SAPIENZA
UNIVERSITÀ DI ROMA

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea in Ingegneria dei Sistemi

Hybrid Dynamic Nonprehensile Manipulation: Application to a 3-dof Robot

Laureando:
Tommaso Proietti

Matricola n. 1208675

Relatore:
Prof. Giuseppe Oriolo

Co-Relatore:
Prof. Kevin Lynch
(Northwestern University)

Anno Accademico
2012-2013

Alla mia meravigliosa famiglia:
Rita, Stefano, Mari e Buck

Tommaso Proietti:
tommaso.proietti.88@gmail.com

Sommario

La *dynamic nonprehensile manipulation* è la capacità di muovere oggetti nello spazio tramite l'utilizzo di robot manipolatori sprovvisti di pinze, dita o altri componenti atti a fornire e mantenere equilibrio statico tra robot e oggetto durante il movimento. Il vantaggio di utilizzare manipolatori robotici non prensili sta nella possibilità di minimizzare la complessità dell'hardware del sistema robotico e dunque minimizzarne generalmente i costi, andando a colmare eventuali gap concentradosi sulla modellazione delle dinamiche del sistema e utilizzando le forze di contatto per sfruttare effetti dinamici, che potenzialmente possono aumentare il range di compiti risolvibili dal manipolatore stesso. La manipolazione non prensile, che fa uso di primitive di movimento come il *pushing*, il *rolling*, il *throwing*, etc., impone dunque un maggiore sforzo nella realizzazione di un modello matematico accurato, nella generazione di traiettorie e nel controllo. In questa tesi analizziamo un semplice sistema composto da un manipolatore piatto a tre gradi di libertà che movimenta un oggetto rettangolare su di un piano senza attrito sotto l'effetto della gravità. Per questo sistema robotico, definiamo e cataloghiamo tutte le possibili combinazioni di contatti fra robot ed oggetto, modellando ciascuna delle cinque modalità di contatto che si ottengono: *dynamic grasp*, *roll*, *slide*, *slide-roll* e *free-flight*. Sfruttando la teoria degli automi ibridi, descriviamo il nostro sistema come una collezione di *stati*, i quali rappresentano modalità di contatto ciascuna caratterizzata dalla propria dinamica, e *salti*, che rappresentano transizioni tra modalità di contatto. Una volta modellato il sistema, ci occupiamo del problema della pianificazione delle traiettorie, affidata ad un algoritmo gerarchico a due livelli: il primo livello, *sequence planner*, si occupa di trovare la sequenza di stati migliore, ad esempio evitando modalità di contatto difficilmente controllabili come lo *slide*, e di fornire le condizioni di salto per ogni transizione; il livello inferiore dell'algoritmo, *contact mode planner*, si occupa invece di pianificare la traiettoria all'interno della singola modalità di contatto. Concludono la tesi una serie di simulazioni in Matlab per mostrare le performance dell'algoritmo di pianificazione da noi sviluppato.

Abstract

Dynamic nonprehensile manipulation is the ability of transferring objects in the space by using robotic manipulators which are not provided with fingers, grippers, or any kind of tool useful to hold static equilibrium among robot and objects during the full motion. The advantage of using this kind of manipulation is that we can minimize the complexity of robot hardware and so generally obtain less expensive robotic systems, reducing possible gaps by focusing on modeling the dynamics of the system and utilize contact forces to exploit dynamic effects, potentially increasing the set of solvable tasks. Thus nonprehensile manipulation, which uses motion primitives such as *rolling*, *sliding*, *pushing*, *throwing*, etc., raises challenges in fields like planning and controlling. In this thesis we focus on a case-study of nonprehensile manipulation, analyzing a 3-*dof* flat robotic manipulator controlling a rectangular object on a frictionless plane under the effect of gravity. We define and classify every feasible contact combination between object and manipulator, modeling the five contact modes we obtain: *dynamic grasp*, *roll*, *slide*, *slide-roll*, and *free-flight*. By using the theory of hybrid automata, we describe our system as a collection of *states*, which represent contact modes characterized by their own dynamics, and *jumps*, which represent transitions between contact modes. After having modeled our system, we address the problem of trajectory planning. We design a hierarchical two-level planner: the high-level planner, *sequence planner*, that manages to find a feasible sequence of contact modes as well as conditions of transition between modes, and a low-level planner, *contact mode planner*, that takes care of planning trajectory within single contact modes. The thesis ends with some simulations in Matlab in order to show the performance of our solution.

Contents

Notation

General notation

$\mathcal{M}$	the manipulator
$\mathcal{B}$	the object
μ	coefficient of friction
σ	friction angle, $\sigma = tan^{-1}(\mu)$
g	gravity acceleration
m	object mass
w_o, w_m	object and manipulator semi-width
l_o, l_m	object and manipulator semi-length
I_o	object inertia matrix
dof	degrees of freedom

Modeling

B	object body frame
M	manipulator frame
W	world frame
x_o, y_o, θ_o	object linear and angular position w.r.t. M
x_m, y_m, θ_m	manipulator linear and angular position w.r.t. W
q_o	object configuration, $q_o = (x_o, y_o, \theta_o)$
q_m	manipulator configuration, $q_m = (x_m, y_m, \theta_m)$
q	configuration of the system object-manipulator, $q = (q_o, q_m)$
z_o	object state, $z_o = (q_o, \dot{q}_o)$
z_m	manipulator state, $z_m = (q_m, \dot{q}_m)$
z	state of the system object-manipulator, $z = (q, \dot{q})$
u	input vector, $u = (u_x, u_y, u_\theta) = (\ddot{x}_m, \ddot{y}_m, \ddot{\theta}_m)$
$\mathbf{w}$	combined control-state vector, $\mathbf{w} = (z, u)$
F	Fixed contact

L	sliding Left contact
R	sliding Right contact
N	No contact
C_i	flow set for the contact mode i
D_{ij}	jump set from the contact mode i to the contact mode j
$\|\| \circ \|\|$	dimension of the set $\circ$
$F_i(x)$	set-valued mappings for continuous dynamics (flow) of the mode i
$G_{ij}(x)$	set-valued mappings for discrete dynamics (jumps) from the contact mode i to the contact mode j
N	normal force
N_1, N_2	components of the normal force N at the contact points
f	friction force
F_{fic}	fictitious forces (F_{ine} inertial force, F_{cor} Coriolis force, F_{eul} Euler force, F_{cen} centrifugal force)
τ_i	torque about point i
x_M, y_M	linear position of the contact point in rolling and slide-roll modes w.r.t. M

Trajectory planning

z_i	initial state
z_g	goal state
$\mathcal{C}$	the configuration space
$\mathcal{C}_{free}$	the free configuration space, *i.e.*, the space in which the robot can move without any constraint but its own dynamics
$\mathcal{C}_{obs}$	the obstacle space, *i.e.*, $\mathcal{C}_{obs} = \mathcal{C} \setminus \mathcal{C}_{free}$
RRTs	Rapidly-exploring Randomized Trees Algorithm
RG-RRTs	Reachability-Guided RRTs Algorithm
z_{rand}	random state
z_{near}	closest node of the tree to z_{rand}
$R(z)$	reachable set of z
w	wrench

Chapter 1

Introduction

Oh, God help us.
We are in the hands of engineers.

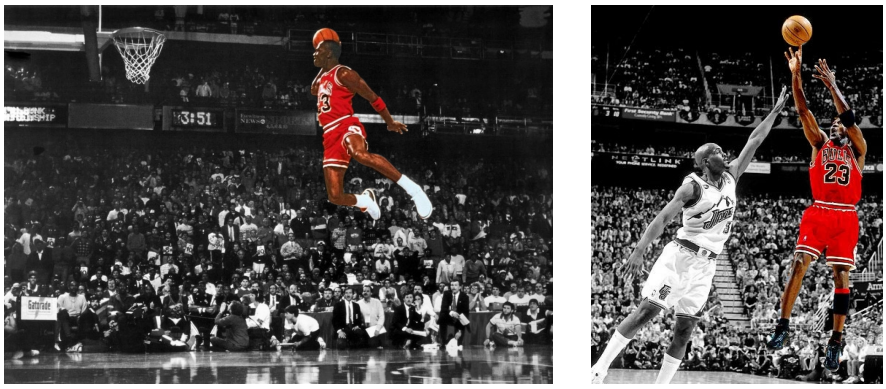
Dr. Ian Malcom - Jurassic Park

Engineers, and more generally scientists, from all over the world are trying to make life easier, safer, and more comfortable. In the last three decades the presence of robots in our everyday life has grown quickly. We are surrounded by embedded devices, "smart" devices, computer-controlled tools which are integrated and exploited in a variety of situations, from industrial productions to home care, from surveillance to transportation, from exploration (terrestrial, undersea, and spatial) to monitoring, from communication to rescuing, from entertainment to, unfortunately, military applications. These devices assist us in achieving more or less complex tasks and most of the time they do work in place of us. As robotic engineers, we are interested in developing robots able to reproduce the wide and huge capabilities of a human being.

This thesis develops in the study of one of these capabilities, that is the possibility of manipulating objects in the environment.

> *Manipulation is the process of using one's hands to rearrange one's environment.*
> Matthew T. Mason, *Mechanics of Robotic Manipulation* [1]

The term *manipulation* refers to a variety of different tasks involving changes of the environment around us: moving objects in the space, joining objects by gluing, welding, or fastening, reshaping an object by cutting, bending, or grinding, and many others. We are interested in studying the process of moving objects in the environment. Many fields in robotics, like navigation and localization capabilities, have improved tremendously in the past few years, but autonomous manipulation still lags. Clearly studying autonomous



(a) Prehensile (b) Nonprehensile

Figure 1.1: Examples of manipulation in sport. Micheal Jordan grasps the basketball to dunk (a), while, taking a jump shot from outside (b), he lets the basketball roll on his hand, then he throws it, pushing it slightly with the tips of his fingers. And of course, no matter what, he always scores.

manipulation is one of the possible keys to enable new capabilities in future robots for any of the above-quoted applications.

1.1 Dynamic nonprehensile manipulation

The ability of moving objects from an initial configuration to a final one depends on how the manipulator can act in the environment and can enter in contact with it. Generally speaking we can describe two types of manipulation processes, namely *prehensile* and *nonprehensile* (figure 1.1). In robotics, prehensile manipulators are usually common solutions to address the problem of object manipulation: these robots are provided either with a gripper or with fingers, used to grab the object and to hold it while moving along the desired trajectory. *Pick-and-place* robots working together with conveyor belts, for example, are common systems exploiting prehensile manipulation. A very less common solution is the use of nonprehensile manipulators.

In nonprehensile manipulation tasks, the system is provided with less dexterous hardware, and, in order to reduce this gap, the efforts are focused on modeling the dynamics of the system and utilizing contact forces to exploit dynamic effects, increasing the set of solvable tasks.

> *A good model of the mechanics of a task is a resource for the robot system, just as actuators, sensors, and computers are resources.*
> Kevin Lynch, *Ph.D. Thesis*, [2]

Nonprehensile manipulation, which uses motion primitives such as *rolling*, *sliding*, *pushing*, and *throwing*, raises challenges in planning and controlling,

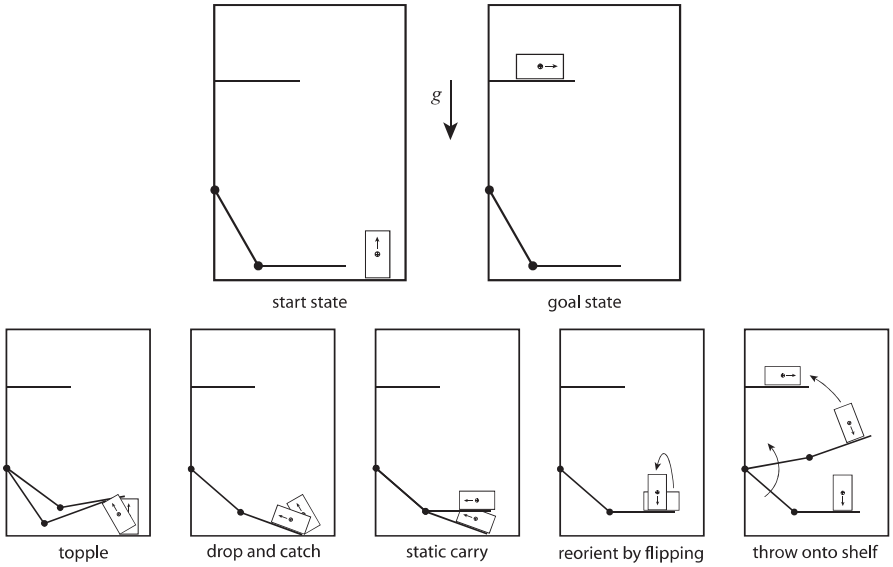


Figure 1.2: An example of nonprehensile manipulation task for a 2-*dof* robot. In this case the dimension of the space in which the robot can manipulate the object is augmented by exploiting dynamic nonprehensile properties.

as the manipulated object does not maintain static equilibrium throughout the process.

In general, the main advantage of nonprehensile manipulation is that it allows "minimalist" robotic installations that minimize the cost and complexity of robot hardware. Besides, by exploiting contacts with the environment if allowed and by exploiting the gravity and friction forces, these kinds of manipulators can increase the set of achievable tasks and success where standard prehensile manipulators may fail, for example in case of either too large or too heavy objects. Furthermore, such manipulators are usually more flexible than prehensile ones, because the latter are often designed with particular shapes and sizes to achieve tasks by exploiting ad-hoc solutions. Last but not least generally the workspace of the robot can be increased in dimension, allowing us to use underactuated robots to reach configuration out of the workspace of the end-effector, for example by throwing the object itself. An example of these advantages is shown in figure 1.2.

It is important to underline that this thesis develops in the field of *dynamic* nonprehensile manipulation, where the "dynamic" adjective means analyzing and taking care of all the possible forces applied among the object, the robot, and the environment, based on the following classification of manipulation models done by Mason and Lynch [3]:

- *Kinematic manipulation*, when kinematics (motion without regards to forces) are considered,

- *Static manipulation*, when kinematics and static forces (such as Coulomb static friction and gravity) are considered,
- *Quasi-static manipulation*, when kinematics, static, and quasi-static forces (such as sliding friction) are considered, and inertial forces are negligible,
- *Dynamic manipulation*, where kinematics, static, quasi-static, and dynamic forces are considered.

1.2 Hybrid System

Manipulation always involves contacts among bodies and very often impacts; a common representation used to describe systems which involve contacts and impacts is the hybrid system representation. We can make a distinction among dynamic systems analyzing the behavior of the state variables; in particular there are three kinds of dynamic systems:

- *Continuous systems*, if the n state variables evolve in the Euclidean space $\mathbb{R}^n$, *e.g.* the motion of a pendulum which is described by a bunch of differential equations of the form $\dot{x} = f(x)$ where x is the state,
- *Discrete systems*, if the n state variables evolve in a countable or finite set $Q \in \mathbb{R}^n$, *e.g.* the growth of an idealized rabbit population, as described by Leonardo Fibonacci in his 1202 book *Liber Abaci*, by using difference equations of the form $x^+ = f(x^-)$,
- *Hybrid systems*, if the n state variables can evolve either continuously or discretely.

A canonical example of hybrid system is a bouncing ball on the ground, in which the state of the system (the height of the ball x and its velocity $\dot{x}$) evolves continuously when the ball is in the air, namely $\ddot{x} = -g$ where g is the gravity acceleration, while at the impact with the ground a discrete *jump* changes instantaneously the velocity of the ball, namely $\dot{x}^+ = -\lambda \dot{x}^-$ where $\lambda \in (0, 1)$ is a dissipation factor, and $\dot{x}^-$ and $\dot{x}^+$ indicate respectively the velocity of the ball before and after the impact.

A common way of describing a hybrid dynamic system is by adopting the following formalism [6]:

$$\begin{aligned} \dot{x} &\in F(x), \quad x \in C, \\ x^+ &\in G(x), \quad x \in D, \end{aligned} \tag{1.1}$$

where $F(x)$ and $G(x)$ denote set-valued mappings for continuous (*flows*) and discrete (*jumps*) dynamics, respectively. The flow set $C \in \mathbb{R}^n$ and the jump set $D \in \mathbb{R}^n$ are subsets of the state space, which represent respectively the set

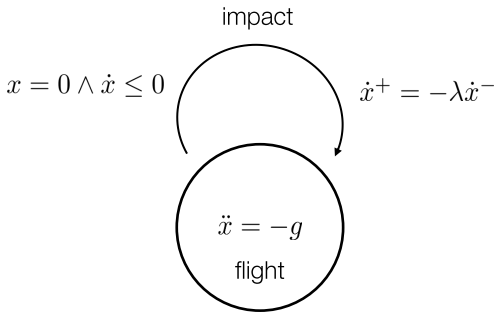


Figure 1.3: The hybrid system graph for the bouncing ball example.

in which flows and jumps occur. A very common way of representing hybrid dynamic systems is given by using a graph in which each *node* represents a state of the system (continuous dynamics), and each *jump*, that is an arrow interfacing nodes, a connection between states (discrete dynamics). In figure 1.3 it is shown the simple graph representing the hybrid system for the bouncing ball example. This basic graph has one only state (flight) which is characterized by the flow dynamics, and a self-ending jump which represents the impact moment and it is characterized by the discontinuous dynamics on the velocity. Labels on jumps and inside states provide the mathematical description of the system, either the dynamics, or the jump condition, or the jump map.

Literature on hybrid system is very broad, and more details about them, for example, can be found in [6], [21], and [33].

1.3 Aim of the thesis

This thesis is motivated by the following central and still partially unsolved general questions in motion planning and control for robotic manipulation:

Given one (or more) object(s), a description of the environment, one (or more) manipulator(s), and the initial state of the system:

- *What states of the system are dynamically reachable, i.e., what tasks are solvable if we allow the robot to make use of dynamic manipulation primitives such as rolling, throwing, catching, etc.?*
- *What is an appropriate form for an automatic manipulation planner to take advantage of manipulation primitives beyond form/force-closure grasping?*
- *How should feedback controllers be designed to stabilize dynamic manipulation primitives?*

These questions are complicated by a number of factors: the state of the system includes not only the state of the object, but that of the manipulator as well, which leads to having high state-space dimension of the system; the coupling of the actuators to the object's state is based on uncertain contact mechanics, and it changes with the manipulator-object contact configuration; sensors for manipulation, such as cameras, laser scanners, tactile sensors, force/torque sensors, etc., tend to give incomplete information on the state of the object; and finally, when the object is not in a firm grasp, gravity sometimes sets a short time scale for manipulation, requiring high-speed sensing and high-bandwidth control of the manipulator to maintain control of the object's motion.

Our long-term goal is to develop a unified framework that automatically plans and executes a sequence of manipulation primitives with each primitive equipped with its own motion planner and stabilizing feedback controller.

To address these kinds of questions, I have been working on a 3-*dof* robot available at *Northwestern University* in Evanston, IL, U.S.A., where I spent one year as a Visiting Predoctoral Fellow. My goal was to study this simple robotic system in order to formulate hypothesis and try to generalize conclusions concerning more general manipulation systems.

1.4 Related research

Particular individual nonprehensile manipulation primitives have been studied in the literature. These include dynamic grasping [34, 35], pushing [36, 37, 38], throwing and catching [39, 40], rolling [41, 42, 43], tapping and sliding [44, 45], and vibratory manipulation [46, 47]. Here, we aim to provide a larger framework that incorporates these previously studied primitives and use them as tools in the full manipulation toolbox.

Hybrid dynamics are used to model mechanical systems with impacts, control systems with switching control modes, biological systems, systems with timers, etc. Goebel, Sanfelice, and Teel [6], studied modeling and controlling of general hybrid systems, while Bullo and Zefran [19] studied hybrid systems in the presence of dynamic constraints and impacts. Stratified configuration spaces were considered by Goodwine and Burdick in [5]. We frame the manipulation problem in this thesis on these works and with the manipulation primitives already mentioned, to create a general framework for hybrid manipulation.

For motion planning for individual manipulation primitives, Trinkle and Hunter [48] outline a method for using sequences of sliding contact primitives to reorient a planar frictionless object. Erdmann demonstrates several open-loop quasi-static manipulation primitives in a single planning framework using two 3-*dof* "palms" moving in a vertical plane [49]. Yashima and Yamaguchi [50] describe a randomized motion planner for whole-arm

prehensile manipulation making use of rolling and sliding contacts. Arai *et al.* [52, 51] describe quasi-static multi-primitive manipulation planning for two fingers that can pivot, topple, and push an object on a support surface. We plan to employ sampling based methods such as the Rapidly-exploring Random Trees developed by LaValle [7]. Branicky *et al.* [9] studied RRTs for nonlinear systems and hybrid systems, and we look to extend their work to manipulation systems with dynamic constraints. To do this, we take advantage of the work done on Reachability-Guided RRTs by Shkolnik *et al.* [13] and provide a reachable set approximation suitable to dynamic manipulation.

1.5 Organization of the thesis

This thesis is composed by two main sections: the first part examines the model of our manipulator and the dynamics of the contacts between robot and environment, while the second part refers to the trajectory planning problem.

Chapter 2 shows the solution of the modeling problem for a simple canonical system in hybrid dynamic nonprehensile manipulation: a bouncing ball on a planar manipulator. Chapter 3 introduces the environmental setup for our 3-*dof* robot as well as the coordinate system defined for it. Chapter 4 gives a wide scenery on the feasible contact modes, and introduce the fundamental theory of the friction cone. Chapter 5 is the core of the first part about modeling, analyzing each contact mode, providing dynamic equations, jump conditions, and jump maps. Chapter 6 provides the topology of the hybrid system in the form of a graph.

Chapter 7 introduces the trajectory planning problem and gets into details about the algorithm we use to complete our tasks. Chapter 8 shows a bunch of simulations we run in order to describe the performance of our solutions. Finally, Chapter 9 ends the thesis providing our conclusions.

There are also multiple appendixes, getting a bit more into details about important but less central topics we analyze during this thesis. We also provide part of the code of our algorithm in one of these appendixes.

Chapter 2

The bouncing ball example

A simple example of hybrid dynamic nonprehensile manipulation system is a bouncing ball on the top of a flat planar manipulator under gravity, in figure 2.1.

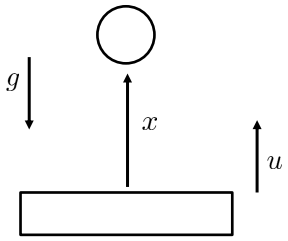


Figure 2.1: The bouncing ball system, where x is the distance between manipulator and ball, u is the acceleration of the manipulator, and g is the gravity acceleration.

Let us consider for simplicity the ball being a point-mass bouncing vertically, and the surface of the manipulator being horizontal. The manipulator is controlled by using vertical acceleration input only, $u \in \mathbb{R}$. The state of the system is the vector $q = (x, \dot{x})^T$ where x is the distance between manipulator surface and the ball. The combined control-state space $\mathbf{w}$ is the three-dimensional space $\mathbf{w} = (x, \dot{x}, u)^T$.

2.1 Hybrid description

We can define two modes for this system: dynamic grasp and free-flight. Dynamic grasp is the name for the condition in which the object is in contact with the manipulator and it is accelerated by it. Free-flight, instead, is the condition in which either there is no contact between the ball and the manipulator or there is contact but the manipulator is accelerating downward with $u < -g$. By adopting the general notations for hybrid systems

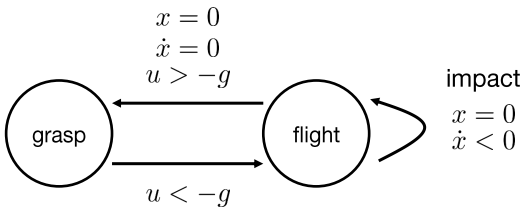


Figure 2.2: Topology of the hybrid system for the bouncing ball.

(1.1), we can define the different sets which describe the ball-manipulator system. In particular, we have the flow set for the dynamic grasp mode

$$C_g = \{\mathbf{w} \in \mathbb{R}^3 : u > -g\}$$

and its jump set to free-flight

$$D_g = \{\mathbf{w} \in \mathbb{R}^3 : u < -g\}.$$

Besides, in dynamic grasp the two set-valued mapping functions are respectively

$$F_g(q) : q = 0 \tag{2.1}$$

for the continuous dynamics, and

$$G_g(q) : q^+ = q^-$$

for the discrete dynamics, which means continuity when there is a transition between grasp and free-flight. For the free-flight mode we have

$$C_f = \{\mathbf{w} \in \mathbb{R}^3 : u < -g\}$$

$$D_f = \{\mathbf{w} \in \mathbb{R}^3 : x = 0, \dot{x} = 0, u > -g\}$$

$$F_f(q) : \ddot{x} = -g$$

and, since there is an impact from free-flight to grasp, and usually impacts lead to discontinuity, we have that

$$G_f(q) : \dot{x}^+ = -\lambda \dot{x}^- + f(q, u)$$

where $\lambda \in (0, 1)$ is a dissipation factor, and $f(q, u)$ is a function that maps the transfer of kinetic energy from the manipulator to the ball. Obviously the condition for having an impact are given by those in figure 2.2.

So far we have not analyzed the behavior of the system when the control is exactly $u = -g$ and position and velocity of the object are, respectively, $x = 0$ and $\dot{x} = 0$. In this state the system has an ambiguous behavior, since the object and the manipulator are in contact, but just knowing position, velocity and acceleration we are not able to determine if the system is in

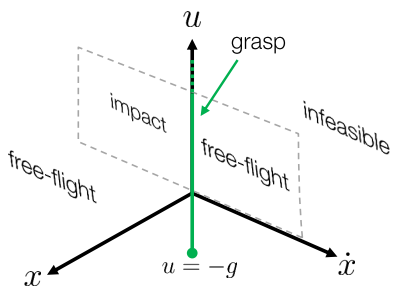


Figure 2.3: Combined control-state space **w** for the bouncing ball system.

dynamic grasp or free-flight. In this ambiguous situation the analysis should be done on the higher orders of derivatives of the state variables (in this case the jerk) and determine by this data the current mode of the system. In particular, if the jerk is greater than zero, we have free-flight, if less than zero dynamic grasp, if exactly zero, we have to move to the snap, and so on. It is important to underline that this technique could not be sufficient in presence of discontinuous inputs, for which the jerk, or generally the higher orders of derivatives, could not be defined.

2.2 State space and control-state space graphs

Generally speaking, given an object and a manipulator, they have infinitely different relative configurations but a finite number of contact modes. All possible modes can be represented in the combined control-state space of the system by carving it into a set of stratified spaces. Broadly speaking, a configuration space is said to be stratified if it contains subspaces upon which the system evolves following different equations of motion [5].

One of the interesting features of this simple example is that we are able to actually draw the control-state space **w**, since it is just three-dimensional, and visualize the stratified spaces. The combined control-state space is shown in figure 2.3. In particular the plane $x = 0$ is drawn dotted in gray, and behind it there is the set of infeasible states, those states which represent the object passing through the manipulator or even below it. On the same plane, adding the constraint $\dot{x} < 0$, we have the impact conditions. Dynamic grasp is represented by the green line along the u axis, which goes to infinity where $u > 0$ and stops in $u < -g$ where the jump condition to free-flight is met. So the dimension of this mode is one and this is confirmed by the fact that, in dynamic grasp, we have two kinematic constraints represented by $F(q)$ equations (2.1). Everything else in this $\mathbb{R}^3$ space represents feasible combinations of states for the free-flight mode.

Another available visualization is the state-space graph, figure 2.4. In

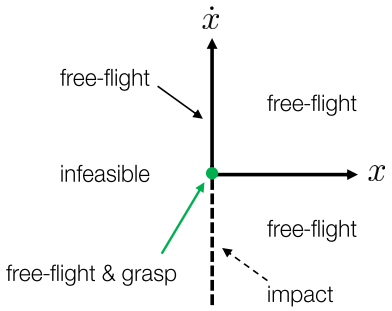


Figure 2.4: State space **q** for the bouncing ball system.

this case we do not consider the presence of the inputs u to graph the stratified space, and, of course, we lose one dimension. Similarly to figure 2.3, in the state-space graph we have all the contact modes represented: infeasible when $x < 0$, impact when $(\dot{x} < 0 \wedge x = 0)$, dynamic grasp when $(\dot{x} = 0 \wedge x = 0)$, and free-flight when $(\dot{x} \geq 0 \wedge x \geq 0)$. Again, the dimension of dynamic grasp is correct since in $\mathbb{R}^2$, constraining two dimension of **q** because of (2.1) we obtain that the grasp mode is a point.

These two graphs have different properties: the combined control-state space graph, incorporating the controls, does not have ambiguity in defining the two different subspaces representing grasp and free-flight when $u > -g$. But in presence of discontinuous controls, transitions from a mode to another does not map to continuous path on the graph. This is shown in figure 2.5 where the jump from grasp to free-flight is represented in both visualizations.

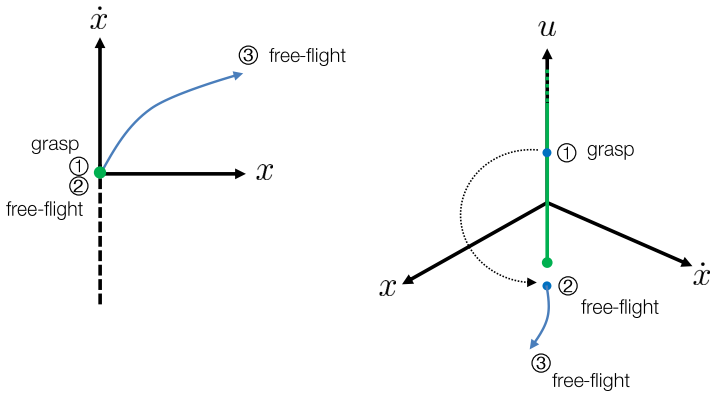


Figure 2.5: Jump from grasp to free-flight visualized in the two space graphs. Step 1: dynamic grasp $(u > 0 \wedge q = 0)$. Step 2: jump to free-flight $(u < -g \wedge x = 0 \wedge \dot{x} = 0)$. Step 3: free-flight $(u < -g \wedge x > 0 \wedge \dot{x} > 0)$.

At step one the object is in dynamic grasp. At step two the control is set to be less than the gravity acceleration: so the system jumps in free-flight

and the value of q starts to increase. At step three the object is completely in free-flight, gaining distance from the manipulator and velocity due to the different acceleration between manipulator and gravity. However, even in the control-state space graph there is still the ambiguity when the state is in $x = 0$, $\dot{x} = 0$, and $u = -g$.

The problem of discontinuous path on the graph does not show up in the state-space graph, even in presence of discontinuous controls. This allows to have interfaces between modes which are only the adjacent interfaces shown in the graph. In the control-state space this is not possible and in fact, in figure 2.5, on the right, step one and two, even if representing an interface between grasp and free-flight, are not adjacent in the graph.

Chapter 3

Our Robot

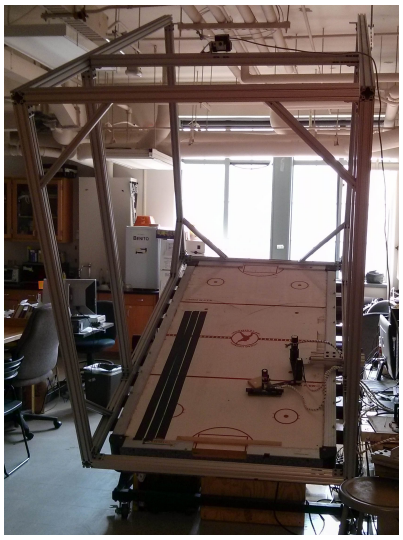
Robots!
Robots on Mars and in oceans, in hospitals and homes, in factories and schools; robots fighting fires, making goods and products, saving time and lives...

B. Siciliano, O. Khatib
Introduction to *Handbook of Robotics* [4]

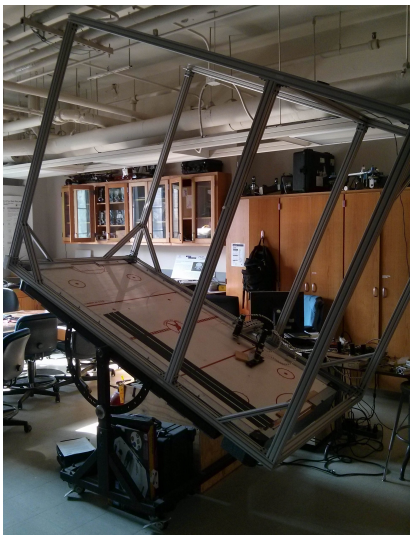
3.1 Experimental setup

A picture of the robot used is shown in figure 3.1: our robot consists of a three link manipulator $\mathcal{M}$ mounted to an air-hockey table. The first two links of the robotic arm are 20 cm each, giving an effective workspace of approximately a 40 cm half circle emanating from the first joint. All manipulation is in the plane of the table. The restriction to a plane simplifies both sensing and mechanism design, allowing us to focus our efforts on the core questions of planning and controlling manipulation sequences.

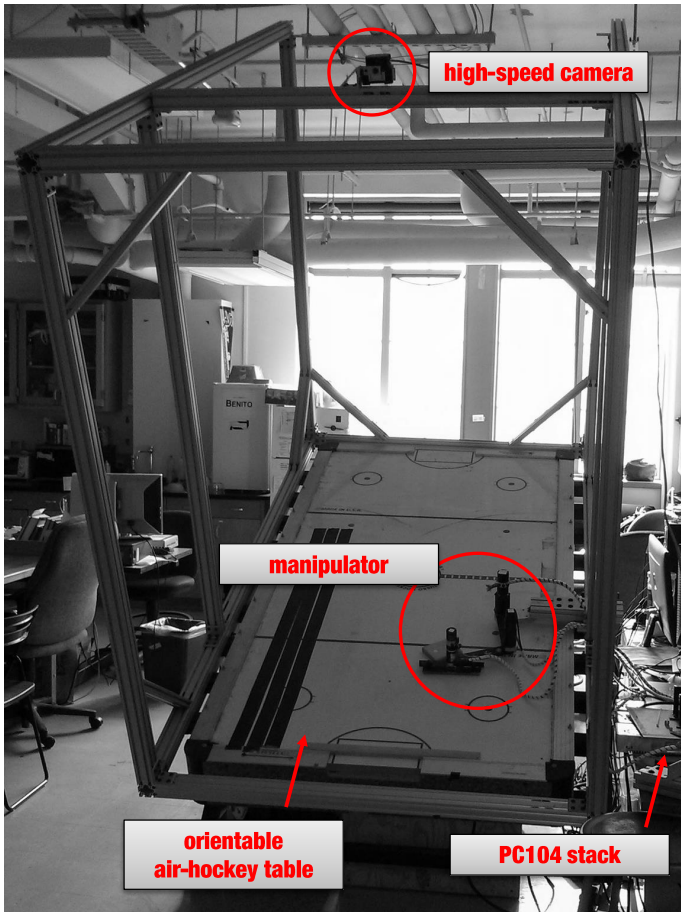
The manipulator is made of aluminum and is adjustably mounted to aluminum extrusions around the edges of the air table or crossing the table. Manipulator actuators are DC motors with high-resolution encoders for position and velocity sensing, and harmonic drive gearing for low backlash and high torque/acceleration. The angle of the table is adjustable from horizontal to vertical, adjusting the gravity vector from perpendicular to the plane to fully in the plane. Air may be turned off, creating support friction between objects and the table, or turned on to create a frictionless surface. The air table is able to float objects of characteristic dimension of several centimeters and weighing up to 1 kg, but for most experiments, objects is



(a) Front



(b) Side



(c) Details

Figure 3.1: The 3-*dof* nonprehensile manipulator.

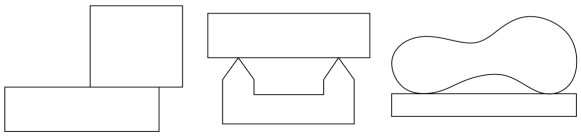


Figure 3.2: Examples of line contact.

made of lightweight plastic. Rubber bands are used on the contact surfaces to increase the friction coefficient.

The primary sensory modality for object state is high-speed vision. We mount an *Optitrack s250e* high-speed IR camera above the air table to image the experiments. We have been able to track markers (e.g., white dots on a dark background) for use in feedback control at rates of 250 Hz. The whole system is controlled by a PC104 stack running the QNX real time operating system with a 1 kHz control loop.

3.2 Assumptions

We focus our analysis on a rectangular planar object $\mathcal{B}$ (figure 3.3). We assume the manipulator and the object are both rigid bodies[1] , all impacts between them are plastic and instantaneous, there are no bounces of the object on the surface of the manipulator, and that contact friction forces follow a Coulomb friction model. In particular, Coulomb's law defines two coefficients, a static friction coefficient μ_s, and a kinetic friction coefficient μ_k, where $\mu_s > \mu_k$. This implies that a larger friction force is available to resist initial motion, but once the object has started to slide, the resisting force decreases. For simplicity, we assume the simplest Coulomb friction model with a single friction coefficient μ, which is a reasonable model for hard, dry materials.

In this thesis, we limit nonprehensile contact to be one of the two basic planar contact types: point contact and line contact. The latter is defined by any contact where all the points of contact are collinear, with contact normals perpendicular to the line. Figure 3.2 shows some examples of line contact. Assuming that the friction coefficient at all the contacts is the same, any line contact can be approximate to two contact points at the ends of the line segment. Point and line contact are the only types of contact which can occur if the manipulator is a straight edge.

[1] A rigid body is a collection of n particles with the property that its shape cannot change, *i.e.*, the distance and the orientation between any two of its constituent particles is fixed. Roughly speaking, a rigid body is a completely "undistortable" body. An arbitrary n particles system requires $3n$ coordinates to specify its configuration in the space $\mathbb{R}^3$, three coordinates x, y, z with respect to a fixed frame for each of these n particles. A rigid body instead requires just six coordinates in $\mathbb{R}^3$, x, y, z to specify the position of the center of mass of the body, and α, β, γ for the body orientation with respect to a fixed frame.

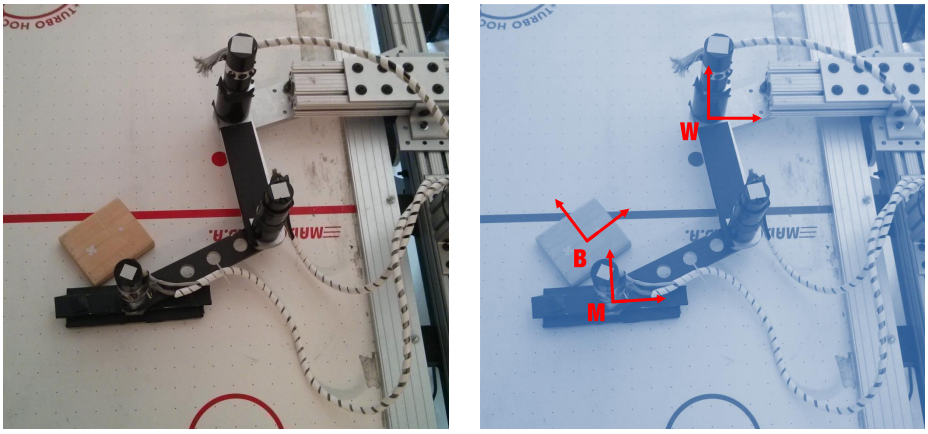


Figure 3.3: Coordinate system. W is a fixed frame, M is a non-inertial frame attached to the manipulator, B is the object body frame. q_o are the coordinates of the center of mass of the object with respect to the manipulator, *i.e.*, in the M frame. q_m are the coordinates of the center of mass of the third link of the manipulator, the "hand", with respect to the world frame W.

3.3 Coordinate system

In our situation, a rigid body in plane, we only need three coordinates to fully describe the configuration of the object $\mathcal{B}$: with respect to the manipulator frame M, we need two coordinates for the position (x_o, y_o) of the center of mass and one (θ_o) for the orientation of the rectangle. In particular θ_o is positive if it is defined by a counterclockwise rotation. Therefore the object pose is described by $q_o = (x_o, y_o, \theta_o)$. Besides these three coordinates, we have to consider other three coordinates to describe the configuration of the manipulator with respect to the world frame W, namely $q_m = (x_m, y_m, \theta_m)$. With these choices $q = (q_o, q_m) \in \mathbb{R}^6$: obviously some configurations cannot be reached because of physical constraints. For example the set $A = \{\forall q : x_o \in [0, 2l_m], y_o < (w_m + w_o)\}$ where l_m is the semi-length of the manipulator surface, and w_m, w_o respectively the manipulator and object semi-width, represents the set of configurations in which the object is passing through the manipulator, that is obviously unfeasible.

Defining the state of $\mathcal{B}$ as $z_o = (q_o, \dot{q}_o)$, and the state of $\mathcal{M}$ as $z_m = (q_m, \dot{q}_m)$, our system is described by its combined control-state vector $\mathbf{w}$, which is 15 dimensional:

$$\mathbf{w} = (z, u) = (q, \dot{q}, u) \tag{3.1}$$

where

$$u = (u_x, u_y, u_\theta) \triangleq (\ddot{x}_m, \ddot{y}_m, \ddot{\theta}_m)$$

is the manipulator acceleration vector, that is our control input. These accelerations are referred to the world frame W.

It is important to notice that the reference frames B and M are *non-inertial* frames. A non-inertial frame is a reference that is undergoing accelerations with respect to an *inertial* frame: laws of motion in non-inertial frames vary from these in inertial frames, depending on the relative acceleration between frames (see Appendix A).

Chapter 4

Contact modes

Given a rigid body $\mathcal{B}$ and a 3-*dof* manipulator $\mathcal{M}$ in $\mathbb{R}^2$, these two objects may assume an infinite number of different relative configurations which potentially may describe many different contact modes. All possible combinations show up in the combined control-state space of the system, which is carved up into a set of stratified spaces. Broadly speaking, a configuration space is said to be stratified if it contains subspaces upon which the system evolves following different equations of motion [5]. In nonprehensile dynamic manipulation systems each stratum represents a contact mode. These spaces may or may not intersect, and if they do, these intersections represent switches among two or more modes. For example, if one stratum represents $\mathcal{B}$ in free-flight over $\mathcal{M}$, and another stratum represents $\mathcal{B}$ in one-point rolling contact with $\mathcal{M}$, the intersections is the set of conditions describing the instant in which $\mathcal{M}$ "catches" $\mathcal{B}$, or in which $\mathcal{M}$ "throws" $\mathcal{B}$, that is the set of states satisfying both free-flight and one-point roll conditions (in one point-contact, but with velocity null). More generally, the interfaces are mathematical descriptions of possible contacts with the environment, with other objects, with the manipulator, and they may be function of the configuration of the system or of the full control-state vector. We are interested in analyzing this complex system in terms of contact modes, *i.e.*, find the mathematical description for these spaces and their intersections.

We want to describe our nonprehensile manipulation system as a hybrid system in the form of [6],

$$\begin{aligned} \dot{x} &= F_i(x,u), \quad x \in C_i, u \in U \\ x^+ &= G_{ij}(x), \quad x \in D_{ij} \end{aligned} \tag{4.1}$$

where $F_i(x,u)$ and $G_{ij}(x)$ are differential and difference equations for continuous (flows) and discrete (jumps) dynamics, respectively. In particular, $F_i(x,u)$ is the flow for the contact mode i, and $G_{ij}(x)$ is the jump from the

contact mode i to the contact mode j. C_i and D_{ij}, the flow set and jump set respectively, are subsets of the combined state-control space $\mathbf{w}$. U is the set of admissible controls. Each contact mode is described by a mode in the hybrid system, and each transition between modes is a jump, so each stratum has a particular flow map F_i. The dynamics of both $\mathcal{M}$ and $\mathcal{B}$ follow this flow map in that particular mode. Since we are assuming instantaneous impacts, forces are not interested in the jump map $G_{ij}(x)$, which is just a function of the state x.

4.1 Friction cone

In order to describe easily all the different contact modes that may occur between object and manipulator, we need to introduce the concept of *friction cone* [1]. Suppose to have a block of mass m at rest on a surface, and let σ

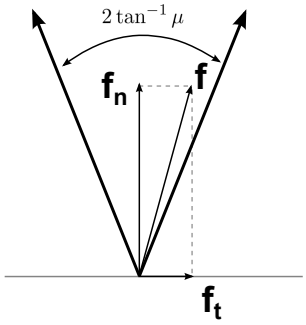


Figure 4.1: Friction cone: the cone described by all the wrenches satisfying Coulomb's law. In this picture, $f = f_n + f_t$ is inside the friction cone, so the contact point will rest on the surface.

be the angle of this surface; under the effect of the gravity we can state:

$$\begin{aligned} f_n &= mg\cos\sigma \\ f_t &= mg\sin\sigma \end{aligned} \tag{4.2}$$

where f_n and f_t are respectively the normal force and the tangential force to the surface. If no other forces are applied, the Coulomb's law describes the behavior of the block. To be at rest, the block must satisfy

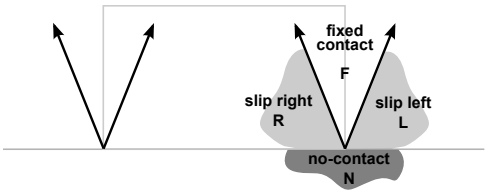
$$|f_t| \leq \mu f_n$$

where μ is the coefficient of friction, while to slide

$$|f_t| = \mu f_n$$

where the direction of slipping depends on the sign of f_t. Substituting (4.2) in the latter case,

$$\begin{aligned} mg\sin\sigma &= \mu mg\cos\sigma \\ \sigma &= \tan^{-1}\mu. \end{aligned}$$



(a) Friction cones

	F	L	R	N
F	✓			✓
L		✓		✓
R			✓	✓
N	✓	✓	✓	✓

(b) Contact combinations

Figure 4.2: Combinations for the two contact points. ✓mark means that the combination is feasible. F = fixed point ($u \in$ friction cone), L = sliding left point ($u \in$ right edge), R = sliding right point ($u \in$ left edge), N = no-contact point ($u \in$ no-contact).

The angle σ is sometimes called *friction angle*, and it is the maximum angle at which the block can remain at rest. This friction angle provides a geometrical approach to Coulomb's law: referring to figure 4.1, it describes a polyhedral convex cone in wrench[1] space, called *friction cone*, always perpendicular to the contact surface with vertex at the contact point, and dihedral angle $2\sigma = 2\tan^{-1}\mu$. Then it is possible to state the Coulomb's law:

$$\begin{array}{lcll} \text{left sliding} & f_n + f_t & \in & \text{right edge of the friction cone} \\ \text{right sliding} & f_n + f_t & \in & \text{left edge of the friction cone} \\ \text{rest} & f_n + f_t & \in & \text{friction cone.} \end{array}$$

4.2 Modes

In our manipulation problem we have two contact points between the object $\mathcal{B}$ and the manipulator $\mathcal{M}$, and so two friction cones. For both contact points we have four different contact types:

$$\begin{array}{lcll} \text{left sliding} & u_n + u_t & \in & \text{right edge of the friction cone} \\ \text{right sliding} & u_n + u_t & \in & \text{left edge of the friction cone} \\ \text{fixed contact} & u_n + u_t & \in & \text{friction cone} \\ \text{no-contact} & u_n < 0, & & \end{array}$$

where u_n and u_t are respectively the acceleration along the normal and the tangential axis of the manipulator. These contact types are represented in figure 4.2(a). In figure 4.2(b) it is analyzed any possible combination for the two contact points between the object and the manipulator. This figure shows that there are 10 different feasible combinations which need to be studied in order to describe the full topology of the system. Trivially we

[1] A generalized force acting on a rigid body, consisting of a linear component f and an angular component τ applied to a point of the body, is a *wrench*. In $\mathbb{R}^6$ we can represent this generalized force as a 6-by-1 vector $w = (f, \tau)^T$, whose values depend on the coordinate frame in which the forces and the moments are expressed.

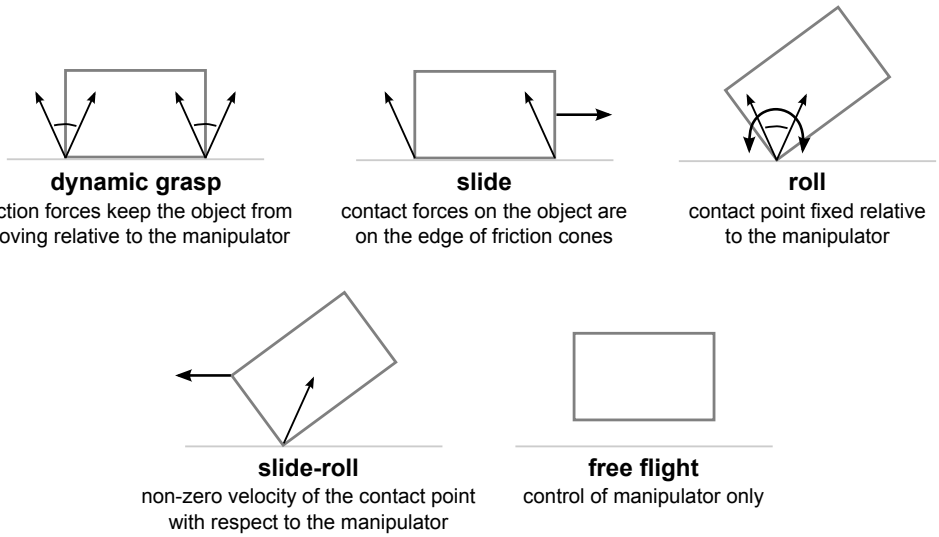


Figure 4.3: General contact modes for the rectangular object $\mathcal{B}$. Possible contact forces are shown by the friction cone at fixed contacts or by the line of action of slipping contact forces.

refer to an AB combination, where A is the left contact point of the object and B is the right one. We can gather these ten contact combinations in five different contact modes, based on the number of contact points and on common dynamics among these combinations (figure 4.3):

- 2-point fixed (FF) $\rightarrow$ *Dynamic Grasp*
- 2-point moving (LL, RR) $\rightarrow$ *Slide*
- 1-point fixed (FN, NF) $\rightarrow$ *Roll*
- 1-point moving (LN, NL, RN, NR) $\rightarrow$ *Slide-Roll*
- 0-point (NN) $\rightarrow$ *Free-Flight*.

Dynamic grasp means that the relative configuration between the object $\mathcal{B}$ and the manipulator $\mathcal{M}$ is fixed, and so the velocity of $\mathcal{B}$ with respect to $\mathcal{M}$ is zero. The "dynamic" term is meant to underline the fact that the relative state between $\mathcal{B}$ and $\mathcal{M}$ is kept while the two object are moving, but, of course, the dynamic grasp mode includes the static situation as well. Sliding modes are those modes in which $\mathcal{B}$ has two point of contact with $\mathcal{M}$, but the velocities are not the same. In particular the velocities along the x-axis are different. One-point rolling is characterized by zero velocity of the contact point, and so fixed position of the contact relative to the manipulator $\mathcal{M}$. As the name suggests, the object can rotate around the contact point. The slide-roll modes are the most complex: in these combinations of contact, the object is allowed to rotate around the contact point even if this is not

fixed relative to the manipulator surface, since it is sliding on it. There are four different combinations of slide-roll modes. Finally, the free-flight mode occurs when each contact between object and manipulator is broken. The control is applied only on the manipulator $\mathcal{M}$, while the object follows a ballistic motion.

Chapter 5

Dynamics of the modes

Contents

In this section we provide a detailed description of each contact mode, analyzing their dynamics and providing the jump conditions reachable from any state in the hybrid graph.

5.1 Dynamic grasp

The dynamic grasp or sticky contact mode occurs when the two contact points are both fixed, FF combination in figure 4.2(b): in this situation the manipulator and the object maintain the same relative configuration while moving. In particular $\mathcal{B}$ velocity in M frame is zero as well as its acceleration.

***FF* dynamics** Refer to figure 5.1. All the forces applied on the object $\mathcal{B}$ are shown in the figure: the weight mg, the normal force N due to the contact with the manipulator edge, and the friction force f. Since it is a two point-contact mode, we can think the normal force N decomposed in the two normal forces N_1 and N_2 relatively to the two points of contact, and that trivially, $N_1 + N_2 = N$. This fact will be useful later to evaluate the jump conditions.

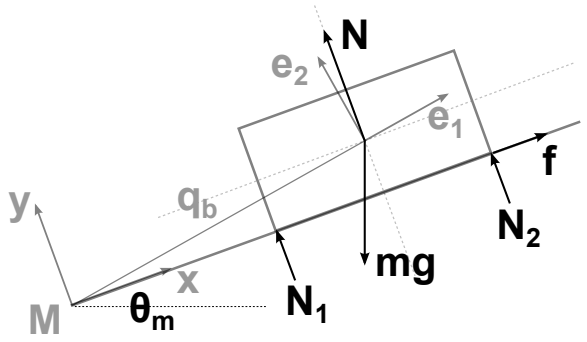


Figure 5.1: Dynamic grasp. x and y represent the M frame of figure 3.3, that is the manipulator frame.

The dynamics of the grasp is simple: the object must stay fixed on the manipulator surface, then we have

$$\begin{aligned} \ddot{x}_o &= 0 \\ \ddot{y}_o &= 0 \\ \ddot{\theta}_o &= 0 \end{aligned} \tag{5.1}$$

that means that the object $\mathcal{B}$ follows the dynamics of the manipulator $\mathcal{M}$.

Control-state space dimension To compute the dimension of the control-state space we need to take into account all the conditions which constrain elements of the control-state vector $\mathbf{w}$. The full dimension, as we said, is 15 and the vector is expressed by (3.1). In dynamic grasp we assume the object fixed on the manipulator surface, so in particular its velocity must be null. Namely

$$\dot{q}_o = (\dot{x}_o, \dot{y}_o, \dot{\theta}_o) = 0.$$

Moreover, we have constraints on the position:

$$\begin{aligned} \dot{x}_o = 0 \rightarrow x_o &= \text{const} \\ y_o &= w_o + w_m \\ \theta_o &= 0 \end{aligned} \tag{5.2}$$

where w_o and w_m are respectively the object and the manipulator semi-width, and θ_o is zero because of the coordinate system chosen at the beginning. It is important to notice that the constraint on the position along the x-axis does not decrease the dimension of the control-state space $\mathbf{w}$: that constraint limits the evolution of the x_o variable once the dynamic grasp mode has been entered in, but it does not force x_o to match any value in advance to get to the dynamic grasp mode. Roughly speaking, we can jump into dynamic grasp with the object $\mathcal{B}$ at any x-position on the surface of the manipulator $\mathcal{M}$, but once we are in this mode, $\mathcal{B}$ cannot change its x_o without changing mode. q_m, $\dot{q}_m$, and u, respectively the manipulator position,

velocity and acceleration vector, are not constrained. In the end we have five constraints, so the dimension of the dynamic grasp mode is 10, that is $\|C_{grasp}\| = 10$.

Jump conditions From the dynamic grasp mode we can jump to any other contact mode, namely to the two-point contact sliding modes, to the rolling modes, to the slide-roll modes, and to the flying mode (figure 5.2). In order to find the jump conditions for any of them, we have to evaluate the force balance and the torque balance in this configuration. Concerning

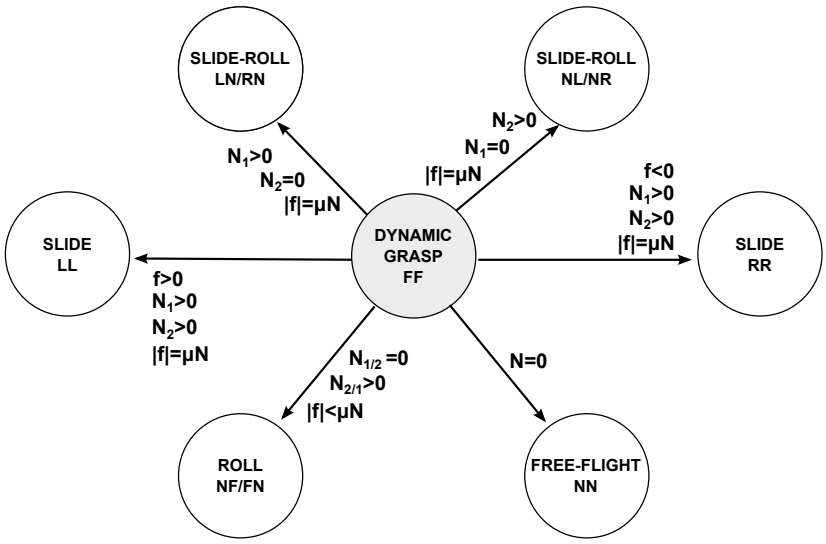


Figure 5.2: Dynamic grasp jump conditions. To be more precise, the jump conditions to the *NF* roll are $(N_1 \leq 0) \wedge (N_2 > 0)$, while $(N_2 \leq 0) \wedge (N_1 > 0)$ represent the conditions to the *FN*. Besides the value of the friction force f determines the direction of slipping in slide-roll modes, which are grouped not to make the figure too complex.

the balance of forces along the two axis we have to take into account the effects of the fictitious forces, since we are considering a non-inertial frame. We can state:

$$F = ma + F_{fic}$$

where a is the acceleration of the object with respect to the non-inertial frame, while the term F_{fic} takes into account the acceleration of the non-inertial frame with respect to the world frame. In particular we know that

$$F_{fic} = F_{ine} + F_{cor} + F_{eul} + F_{cen}$$

where these terms are respectively the so-called *Inertial force*, *Coriolis force*, *Euler force*, and *Centrifugal force*. Along the two axis in the M frame we have:

$$\begin{array}{lrcl} \mathbf{x}: & f - mg\sin\theta_m - F^x_{fic} & = & 0 \\ \mathbf{y}: & N - mg\cos\theta_m - F^y_{fic} & = & 0. \end{array} \tag{5.3}$$

In addition we know that

$$N > 0 \tag{5.4}$$
$$|f| \leq \mu N, \tag{5.5}$$

or, by using the friction cone,

$$f_t + f_n \in \text{cone} \tag{5.6}$$

where f_n and f_t are respectively the sum of the normal forces and the sum of the tangential forces to the manipulator surface, namely those in (5.3). To complete the force balance we need to compute the value of the fictitious forces in this particular contact mode. Using the formulas in the Appendix A, we find that:

$$F_{ine} = \begin{pmatrix} mu_x\vec{x} \\ mu_y\vec{y} \end{pmatrix} \tag{5.7}$$
$$F_{cor} = 0 \tag{5.8}$$
$$F_{eul} = mu_\theta q_b \vec{e_2} \tag{5.9}$$
$$F_{cen} = -mq_b\dot{\theta}_m^2\vec{e_1} \tag{5.10}$$

where $\vec{x}$, $\vec{y}$, $\vec{e_1}$, and $\vec{e_2}$ indicate the direction of the force vectors. Thus their decomposition along the axises is:

$$\begin{aligned} F_{fic}^x &= +mu_x\cos\theta_m + mu_y\sin\theta_m - mu_\theta q_b\sin\beta - mq_b\dot{\theta}_m^2\cos\beta \\ F_{fic}^y &= -mu_x\sin\theta_m + mu_y\cos\theta_m + mu_\theta q_b\cos\beta - mq_b\dot{\theta}_m^2\sin\beta. \end{aligned} \tag{5.11}$$

These equations can be simplified when we calculate the value of q_b (see figure 5.1). In particular we have

$$\beta = \tan^{-1}\left(\frac{y_o}{x_o}\right),$$

$$q_b = \frac{x_o}{\cos\beta},$$

and so

$$\begin{aligned} q_b\cos\beta &= x_o \\ q_b\sin\beta &= y_o. \end{aligned} \tag{5.12}$$

Hence (5.11) becomes:

$$\begin{aligned} F_{fic}^x &= +mu_x\cos\theta_m + mu_y\sin\theta_m - mu_\theta y_o - mx_o\dot{\theta}_m^2 \\ F_{fic}^y &= -mu_x\sin\theta_m + mu_y\cos\theta_m + mu_\theta x_o - my_o\dot{\theta}_m^2. \end{aligned} \tag{5.13}$$

By using (5.3), now we are able to determine:

$$\begin{aligned} f &= f(q,\dot{q},u) \\ N &= N(q,\dot{q},u) \end{aligned} \tag{5.14}$$

that are the friction force and the normal force as functions of the state variables and the inputs in this mode. By evaluating those two parameters we are able to derive all the jump conditions in this contact mode. Jumping into sliding modes depends on the value and the sign of the friction force. In particular, if

$$(f = \mu N) \wedge (f < 0) \tag{5.15}$$

the object is sliding to the right, while with

$$(f = \mu N) \wedge (f > 0) \tag{5.16}$$

the object is sliding to the left. The same can be explained in terms of friction cones: if the total force is on the left edge of the cone, the object $\mathcal{B}$ starts to slide to the right, and vice versa. The jump condition to the flight state is determined by the value of N:

$$N = 0. \tag{5.17}$$

It is interesting to notice that (5.17) is the jump condition to the free-flight state for each mode of the system. The jump conditions to the rolling modes (NF and FN) are found by studying the value of N_1 and N_2, and in particular analyzing the torque balance. Trivially, since we are in a two-point contact mode, we can state that the torque balance with respect to any point of the object $\mathcal{B}$ is

$$\sum\nolimits_i \tau_i = 0$$

and in particular they are zero the balances with respect to the contact points. Let us write those torque balances, where counterclockwise torques are positive (figure 5.3):

$$\begin{aligned} \tau_A &= \Big(+ F^y_{fic} - mg\cos\theta_m \Big) l_o + \Big(mg\sin\theta_m - F^x_{fic} \Big) w_o + 2N_2 l_o = 0 \\ \tau_B &= \Big(- F^y_{fic} + mg\cos\theta_m \Big) l_o + \Big(mg\sin\theta_m - F^x_{fic} \Big) w_o - 2N_1 l_o = 0 \end{aligned} \tag{5.18}$$

where τ_A is the balance with respect to the point A, τ_B to the point B, and $2l_o$ and $2w_o$ are the dimensions of the rectangle. These two equations allow us to compute both the normal forces N_1 and N_2 as functions of the state and the inputs.

Finally, by using these two values, we can obtain the jump conditions to the rolling modes:

$$(N_1 = 0) \wedge (N_2 > 0) \tag{5.19}$$

for a clockwise rotation (NF roll),

$$(N_1 > 0) \wedge (N_2 = 0) \tag{5.20}$$

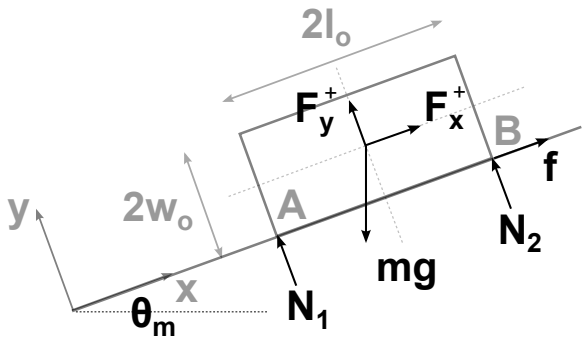


Figure 5.3: Two-point contact. F_x and F_y are respectively the fictitious forces F^x_{fic} and F^y_{fic}. $2l_o$ and $2w_o$ are the object dimensions.

for a counterclockwise rotation (*FN* roll). Lastly, the jump conditions to the slide-roll modes combine (5.15), (5.16), (5.19) and (5.20).

In order to obtain exclusive conditions the final jump sets D_{ij} for the grasp mode are expressed in table 5.1.

jump set D	N_1	N_2	$\lvert f\rvert$	f
slide LL	> 0	> 0	$= \mu N$	> 0
slide RR	> 0	> 0	$= \mu N$	< 0
roll FN	> 0	$= 0$	$< \mu N$	
roll NF	$= 0$	> 0	$< \mu N$	
slide-roll NR	$= 0$	> 0	$= \mu N$	< 0
slide-roll NL	$= 0$	> 0	$= \mu N$	> 0
slide-roll RN	> 0	$= 0$	$= \mu N$	< 0
slide-roll LN	> 0	$= 0$	$= \mu N$	> 0
free-flight NN	$= 0$	$= 0$		

Table 5.1: Jump sets for the dynamic grasp mode.

5.2 Slide

In sliding mode, the object $\mathcal{B}$ has a not null x-velocity with respect to the manipulator $\mathcal{M}$, but it keeps two points of contact with the surface of $\mathcal{M}$. There are 2 different combinations of sliding, depending on the slipping direction. However these two modes share the same dynamics and the same jump conditions, so we analyze them together.

***LL, RR* slide** We can still refer to figure 5.1 for a representation of the forces applied to the object. The analysis of the forces acting on the object is widely similar to what we have in *FF* grasp mode. In particular the

expressions of the fictitious forces are the same, (5.7, 5.9, 5.10), but we have to update the Coriolis term since the object is moving with respect to the manipulator. In particular remembering that

$$F_{cor} = 2m\dot{q}_b \times \Omega$$

we have

$$F_{cor} = 2m\dot{\theta}_m \dot{q}_b \vec{e_2} \tag{5.21}$$

in which $\dot{q}_b = \dot{x}_o$, since the object is sliding on the manipulator surface but it is not allowed to fly. Along the axises,

$$\begin{array}{rcl} F^x_{cor} & = & -F_{cor}\sin\beta \\ F^y_{cor} & = & F_{cor}\cos\beta \end{array}$$

and so, we finally have:

$$\begin{array}{l} F^x_{fic} = +mu_x\cos\theta_m + mu_y\sin\theta_m - mu_\theta y_o - mx_o\dot{\theta}^2_m - 2m\dot{\theta}_m\dot{x}_o\sin\beta \\ F^y_{fic} = -mu_x\sin\theta_m + mu_y\cos\theta_m + mu_\theta x_o - my_o\dot{\theta}^2_m + 2m\dot{\theta}_m\dot{x}_o\cos\beta. \end{array} \tag{5.22}$$

Along the two axis in the M frame we have:

$$\begin{array}{lrcl} \mathbf{x}: & f - mg\sin\theta_m - F^x_{fic} & = & m\ddot{x}_o \\ \mathbf{y}: & N - mg\cos\theta_m - F^y_{fic} & = & 0 \end{array} \tag{5.23}$$

and, in addition, we know that

$$N > 0 \tag{5.24}$$

$$|f| = \mu N \tag{5.25}$$

or

$$f_t + f_n \in \text{edge of the friction cone},$$

where f_n and f_t are respectively the sum of the normal forces and the sum of the tangential forces to the manipulator surface. By using the (5.23) and (5.25), we can derive again:

$$\begin{array}{rcl} f & = & f(q, \dot{q}, u) \\ N & = & N(q, \dot{q}, u) \end{array} \tag{5.26}$$

The only differences with the dynamic grasp case are that here the object has an acceleration with respect to the non-inertial frame (in particular it is sliding on the manipulator and the sign of the terms $\ddot{x}_o$ depends on the direction of sliding), and the friction force is maximum. The dynamics is defined by:

$$\begin{array}{l} \ddot{x}_o = \left(f - mg\sin\theta_m - F^x_{fic}\right)\dfrac{1}{m} \\ \ddot{y}_o = 0 \\ \ddot{\theta}_o = 0. \end{array} \tag{5.27}$$

Control-state space dimension In sliding mode we have constraints on the velocities along the y-axis and the θ-axis. These two velocities must be null. There is not any condition on the x-axis. Again, like in the dynamic grasp mode, similar conditions are true for the position of the object. Namely,

$$\begin{aligned} y_o &= w_m + w_o \\ \theta_o &= 0 \\ \dot{y}_o &= 0 \\ \dot{\theta}_o &= 0 \end{aligned}$$

which make the dimension of the control-state space equal to 11 in this mode, or, in the hybrid system notation, $\|C_{slide}\| = 11$.

Jump conditions For the jump conditions we can refer to figure 5.4. From this mode we can jump to the dynamic grasp mode, to the roll mode, to the flying state, and to the slide-roll modes. Obviously, from the LL slide it is possible to jump only to left sliding modes, while from RR slide to right sliding ones.

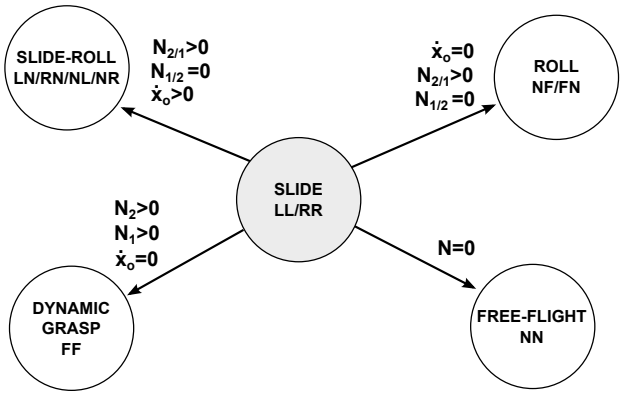


Figure 5.4: Two-point slide contact jump conditions.

The condition to get to the flying state doesn't change with respect to the grasp mode. The condition to the other modes depends on the velocity of the object ($\dot{x}_o$) and, similarly to what we have seen for the grasp jump conditions, on the normal forces N_1 and N_2. By studying the value of N_1 and N_2, and in particular analyzing the torque balance, since we are in a two-point contact mode, we have that

$$\begin{aligned} \tau_A &= \Big(+ F^y_{fic} - mg\cos\theta_m \Big) l_o + \Big(mg\sin\theta_m - F^x_{fic} \Big) w_o + 2N_2 l_o = 0 \\ \tau_B &= \Big(- F^y_{fic} + mg\cos\theta_m \Big) l_o + \Big(mg\sin\theta_m - F^x_{fic} \Big) w_o - 2N_1 l_o = 0 \end{aligned} \tag{5.28}$$

where τ_A is the balance with respect to the point A, τ_B to the point B, and the fictitious forces F^x_{fic} and F^y_{fic} are those in (5.22), updated with the Coriolis terms. The conditions are straightforward (see table 5.2).

jump set D	N_1	N_2	$\dot{x}_o$
dynamic grasp FF	> 0	> 0	$= 0$
roll FN	> 0	$= 0$	$= 0$
roll NF	$= 0$	> 0	$= 0$
slide-roll NR*	$= 0$	> 0	> 0
slide-roll NL*	$= 0$	> 0	< 0
slide-roll RN*	> 0	$= 0$	> 0
slide-roll LN*	> 0	$= 0$	< 0

Table 5.2: Jump conditions for the slide modes. It's not written the condition to the free-flight mode. The * marks remembers that not all these jumps are feasible, depending on the actual direction of slipping.

5.3 Roll

Rolling mode occurs when the object $\mathcal{B}$ rotates around one of its vertexes and this contact point has zero velocity relative to the manipulator $\mathcal{M}$. There are two combinations of rolling, *FN* and *NF*.

***FN* roll** *FN* rolling mode is the name for counterclockwise rotations around the left contact point of $\mathcal{B}$. To get the object rolling, we assume the contact point velocity relative to the manipulator equal to zero, *i.e.*, the contact point velocity "object-side" and "manipulator-side" are the same. With some geometrical considerations, referring to figure 5.5, we can state:

$$p_M = {}^M O_B + R(\theta_o) p_B \tag{5.29}$$

where $R(\theta_o)$ is the rotation matrix around the θ-axis:

$$R(\theta_o) = \begin{pmatrix} \cos\theta_o & -\sin\theta_o \\ \sin\theta_o & \cos\theta_o \end{pmatrix}.$$

Applying (5.29) to the object, we have that

$$\begin{pmatrix} x_M \\ y_M \end{pmatrix} = \begin{pmatrix} x_o \\ y_o \end{pmatrix} + \begin{pmatrix} \cos\theta_o & -\sin\theta_o \\ \sin\theta_o & \cos\theta_o \end{pmatrix} \begin{pmatrix} -l_o \\ -w_o \end{pmatrix}. \tag{5.30}$$

where we defined

$$p_M = \begin{pmatrix} x_M \\ y_M \end{pmatrix}, {}^M O_B = \begin{pmatrix} x_o \\ y_o \end{pmatrix},$$

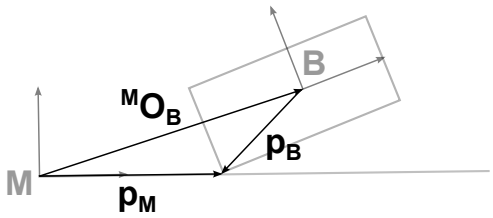


Figure 5.5: Contact point in the manipulator frame.

and

$$p_B = \begin{pmatrix} -l_o \\ -w_o \end{pmatrix}.$$

Double differentiating the (5.30), remembering that to have pure rolling $\dot{p}_M = 0$, $\ddot{p}_M = 0$, we can obtain the two component of the dynamics $\ddot{x}_o$ and $\ddot{y}_o$ as functions of the state variables, the inputs and $\ddot{\theta}_o$:

$$\begin{aligned} \ddot{x}_o &= \ddot{x}_o(q, \dot{q}, u, \ddot{\theta}_o) \\ \ddot{y}_o &= \ddot{y}_o(q, \dot{q}, u, \ddot{\theta}_o) \end{aligned}$$

To describe the full dynamics of the system we need the dynamics of θ_o. This can be obtained by analyzing the torque balance. Generally speaking, we know that the sum of all the torques with respect to the center of mass is equal to the moment of inertia of the center of mass times the angular acceleration of the body, namely

$$\tau_o = \sum\nolimits_i \tau_i^o = I_o \ddot{\theta}_o. \tag{5.31}$$

In our situation, since the contact point is in one of the vertices of the rectangle, we want evaluate the torques with respect to that point, namely

$$\tau_P = \sum\nolimits_i \tau_i^P = I_P \ddot{\theta}_o.$$

By using the Huygens-Steiner theorem, we can state that

$$I_P = I_o + m p_B^2 \tag{5.32}$$

where m is the mass of the body, I_P is the moment of inertia with respect to the contact point P, and p_B is the distance between the contact point and the center of mass. From the literature

$$I_o = \frac{m(l^2 + w^2)}{12}$$

is the moment of inertia for a rectangle, where l is its length and w is its width. With respect to figure 5.6, we can write down the forces which

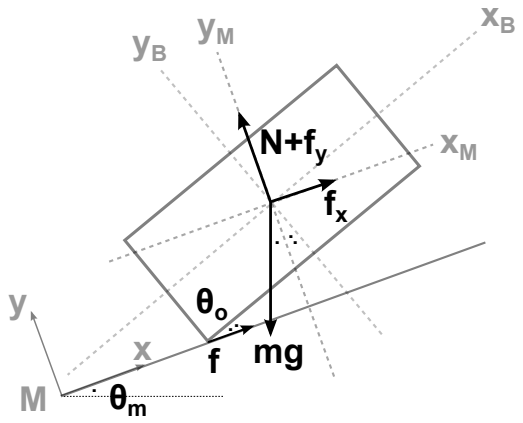


Figure 5.6: *FN* roll mode. f_x and f_y are respectively F^x_{fic} and F^y_{fic}.

produce torques, considering positive the counterclockwise rotations:

$$\begin{aligned}\tau_+ &= l_o\Big(N + F^y_{fic}\Big)\cos\theta_o + mgw_o\sin\Big(\theta_o + \theta_m\Big)\\ \tau_- &= w_o\Big(N + F^y_{fic}\Big)\sin\theta_o + mgl_o\cos\Big(\theta_o + \theta_m\Big)\\ &\quad + F^x_{fic}\Big(w_o\cos\theta_o + l_o\sin\theta_o\Big)\end{aligned} \tag{5.33}$$

where τ_+ is the positive torque, τ_- is the negative torque, l_o and w_o are the two dimensions of the rectangle, and F^x_{fic} and F^y_{fic} are the fictitious forces in rolling mode

$$\begin{aligned}F^x_{fic} &= +mu_x\cos\theta_m + mu_y\sin\theta_m - mu_\theta y_o - mx_o\dot\theta^2_m - 2m\dot\theta_m\dot x_o\sin\beta\\ F^y_{fic} &= -mu_x\sin\theta_m + mu_y\cos\theta_m + mu_\theta x_o - my_o\dot\theta^2_m + 2m\dot\theta_m\dot x_o\cos\beta.\end{aligned} \tag{5.34}$$

The fictitious forces are the same of the sliding mode ones. So we have:

$$\tau_P = \tau_+ - \tau_- = (I_o + p^2_B)\ddot\theta_o \tag{5.35}$$

or, better,

$$\ddot\theta_o = \frac{\tau_P}{I_o + mp^2_B} \tag{5.36}$$

that is the dynamics of the θ_o variable, as

$$\ddot\theta_o = \ddot\theta_o(q, \dot q, u, N).$$

Finally, we can find out the value of the normal force N by solving the second equation of the force balance along the axises:

$$\begin{aligned}\mathbf{x}: \quad f - mg\sin\theta_m - F^x_{fic} &= m\ddot x_o\\ \mathbf{y}: \quad N - mg\cos\theta_m - F^y_{fic} &= m\ddot y_o\end{aligned} \tag{5.37}$$

where the fictitious forces are (5.34). By substituting (5.37) in (5.36) to find θ_o, and then in (5.30) to find x_o and y_o, we obtain the full dynamics of this mode, that is

$$\begin{aligned} \ddot{x}_o &= \ddot{x}_o(q, \dot{q}, u) \\ \ddot{y}_o &= \ddot{y}_o(q, \dot{q}, u) \\ \ddot{\theta}_o &= \ddot{\theta}_o(q, \dot{q}, u). \end{aligned}$$

***NF* roll** *NF* roll is a clockwise rotation around the right contact of the rectangle. In order to analyze the *NF* roll mode, we can study the symmetry with the *FN* case. Figure 5.7 shows the differences between these two

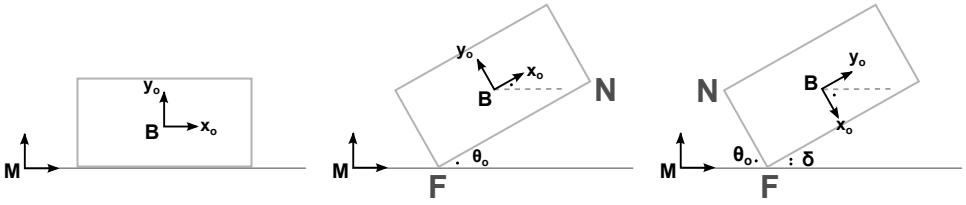


Figure 5.7: Symmetry between *FN* and *NF* rolling modes.

configurations. In the *NF* case, we are dealing with a different angle with respect to the former case: what we used to call θ_o is now represented by δ angle, where δ is θ_o complementary angle

$$\delta = \frac{\pi}{2} - \theta_o.$$

Therefore we can keep the whole analysis we did for the *FN* roll mode taking into account that where it is written θ_o we must substitute δ. Wrapping up the results, we obtain:

$$\begin{pmatrix} x_M \\ y_M \end{pmatrix} = \begin{pmatrix} x_o \\ y_o \end{pmatrix} + \begin{pmatrix} \cos\theta_o & -\sin\theta_o \\ \sin\theta_o & \cos\theta_o \end{pmatrix} \begin{pmatrix} -\frac{a}{2} \\ -\frac{b}{2} \end{pmatrix} \tag{5.38}$$

and by double differentiation,

$$\begin{aligned} \ddot{x}_o &= \ddot{x}_o(q, \dot{q}, u, \ddot{\theta}_o) \\ \ddot{y}_o &= \ddot{y}_o(q, \dot{q}, u, \ddot{\theta}_o). \end{aligned}$$

To find out $\ddot{\theta}_o$,

$$\ddot{\theta}_o = \frac{\tau_P}{I_o + m p_B^2} \tag{5.39}$$

where, now computing the new value of θ_o we have

$$\begin{aligned} \tau_+ &= l_o\Big(N + F^y_{fic}\Big)\sin\theta_o + mgw_o\cos\Big(\theta_o - \theta_m\Big) \\ \tau_- &= w_o\Big(N + F^y_{fic}\Big)\cos\theta_o + mgl_o\sin\Big(\theta_o - \theta_m\Big) + \\ &\quad + F^x_{fic}\Big(w_o\sin\theta_o + l_o\cos\theta_o\Big), \end{aligned} \tag{5.40}$$

and (5.37) to find N as in the former case.

Control-state space dimension As we have seen so far, the control-state space **w** dimension is decreased by kinematic constraints. In rolling mode, there are kinematic constraints on the contact point. In particular we have already said that to make the object $\mathcal{B}$ roll, the contact point velocity must be zero along both x- and y-axis; another constraint is on the position of the contact point P: trivially P must be in contact with the surface of the manipulator, which means $y_P = w_m$, while the x-position must be constant. In the end, **w** is 12 dimensional in this mode, *i.e.*, $\|C_{roll} = 12\|$.

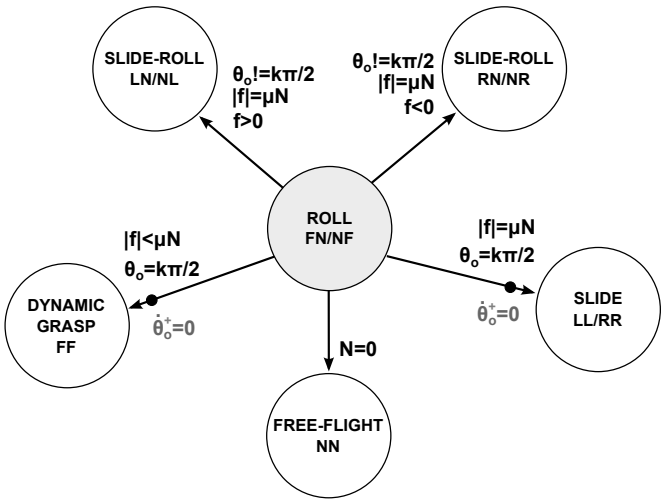


Figure 5.8: Roll mode jump conditions. The black circles indicate impacts in the transitions. From *FN* roll it is possible to reach only *FN*-like slide-roll modes, namely *LN* and *RN*, and vice versa for the *NF* roll. The sign of the friction force f determines the direction of slipping in sliding modes. $\dot{\theta}_o^+ = 0$ is the jump map $G(z)$ when impacts occur.

Jump conditions Even if the two combinations of rolling have different dynamics, although symmetric, they share the same jump conditions. From these modes it is possible to jump to many other modes (figure 5.8). To find the jump conditions, let us start from the force balances:

$$\begin{array}{lrcl} \mathbf{x}: & f - mg\sin\theta_m - F^x_{fic} & = & m\ddot{x}_o \\ \mathbf{y}: & N - mg\cos\theta_m - F^y_{fic} & = & m\ddot{y}_o \end{array} \tag{5.41}$$

where the fictitious forces are (5.34). By using equations (5.41), since we know the dynamics of the system, we can evaluate the friction force f and the normal force N. The easiest condition is the jump equation to the free-flight state, that is $N = 0$. The jump to the slide-roll states is ruled by:

$$(|f| = \mu N) \wedge (f \gtrless 0) \wedge (\theta \neq k\frac{\pi}{2}) \tag{5.42}$$

where the sign of the friction force f determines the direction of slipping. What makes the system jumps into the dynamic grasp mode is determined

by the actual orientation of the object with respect to the manipulator surface: in particular, we can state that the object has got two contacts if $\theta = k\frac{\pi}{2}$, where $k \in \mathbb{Z}$. This condition is not enough to describe the jump conditions because it is necessary also for the slide modes; to create exclusive conditions we need to add a constraint on $|f|$: if the friction force is less than μN we are jumping to the grasp, otherwise we are going to slide. Obviously, as we have already seen many times, the sign of the friction force determines the direction of slipping in case of sliding modes. The conditions are wrapped up in table 5.3.

jump set D	θ_o	$\lvert f\rvert$	f
dynamic grasp FF	$= k\frac{\pi}{2}$	$< \mu N$	
slide LL	$= k\frac{\pi}{2}$	$= \mu N$	> 0
slide RR	$= k\frac{\pi}{2}$	$= \mu N$	< 0
slide-roll RN*	$! = k\frac{\pi}{2}$	$= \mu N$	< 0
slide-roll NR*	$! = k\frac{\pi}{2}$	$= \mu N$	< 0
slide-roll LN*	$! = k\frac{\pi}{2}$	$= \mu N$	> 0
slide-roll NL*	$! = k\frac{\pi}{2}$	$= \mu N$	> 0

Table 5.3: Jump conditions for the rolling modes. It's not written the condition to the free-flight mode. The * marks remembers that not all these jumps are feasible, depending on the actual direction of rolling.

Jump maps The rolling modes introduce an additional necessary analysis concerning the jump map $G(z)$. So far, each mode we have analyzed did not deal with impacts, *i.e.*, either the dynamic grasp or the sliding modes do not create impacts while jumping to other modes. This fact makes trivial the analysis of the jump maps for those modes, since the evolution of the state variables is continuous: the jump maps are just described by

$$G(z) = \{q^+ = q^-, \dot{q}^+ = \dot{q}^-\}.$$

In rolling mode instead, some jumps can cause impacts. Those impacts introduce discontinuity in the angular velocity of the object $\mathcal{B}$. In particular when the system is jumping to dynamic grasp or sliding modes we have:

$$G(z) = \{\dot{\theta}_o^+ = 0\} \tag{5.43}$$

since we assume plastic impact between the object $\mathcal{B}$ and the manipulator $\mathcal{M}$.

5.4 Slide-Roll

The slide-roll mode is a single contact mode. In this mode, the contact point is not constrained to be fixed on the manipulator surface but it has

some x-velocity component (slide) and simultaneously the object can rotate around it (roll). It occurs as a combination of a free contact point (N), with a sliding contact (L or R, depending on the direction of slipping). So we have four different combinations of slide-roll.

***LN, NL, RN, NR* slide-roll** The dynamics of these modes inherits both the rolling and the sliding behaviors. It is possible to apply directly what we have done in order to find $\ddot{\theta}_o$ in the rolling mode: in particular, from the (5.31) we obtain again

$$\ddot{\theta}_o = \frac{\tau_P}{I_P - mp_B^2} \tag{5.44}$$

where $\tau_P = \tau_+ - \tau_-$, τ_+ and τ_- are respectively (5.33) for NL and NR modes, and (5.40) for LN and RN modes.

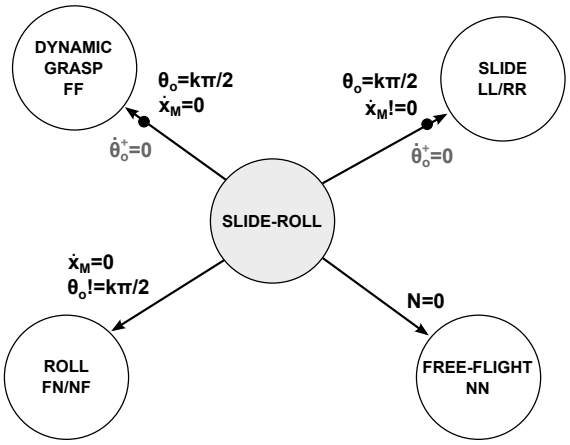


Figure 5.9: Slide-roll mode jump conditions. The black circles indicate impacts in the transitions. Only left slide-roll modes can reach LL slide and only right slide-roll modes can reach RR slide. Only RN and LN slide mode can jump to FN roll, and only NR and NL can jump to NF roll. $\dot{\theta}_o^+ = 0$ is the jump map $G(z)$ when impacts occur.

To evaluate $\ddot{x}_o$ and $\ddot{y}_o$ we still have (5.30) valid, but in this case, since the object is simultaneously rolling and sliding on the manipulator surface, it is not true anymore that $\dot{x}_M^x = 0$. So we cannot complete the dynamics only solving the double derivative of (5.30). Since the object it is sliding, by applying the force balance, we get:

$$\begin{array}{lrcl} \mathbf{x}: & f - mg\sin\theta_m - F_{fic}^x & = & m\ddot{x}_o \\ \mathbf{y}: & N - mg\cos\theta_m - F_{fic}^y & = & m\ddot{y}_o \end{array} \tag{5.45}$$

and, of course,

$$N > 0 \tag{5.46}$$

$$|f| = \mu N. \tag{5.47}$$

It is important to underline that now $\ddot{y}_o \neq 0$. By using the double derivative of

$$\begin{pmatrix} x_M \\ y_M \end{pmatrix} = \begin{pmatrix} x_o \\ y_o \end{pmatrix} + \begin{pmatrix} \cos\theta_o & -\sin\theta_o \\ \sin\theta_o & \cos\theta_o \end{pmatrix} \begin{pmatrix} -l_o \\ -w_o \end{pmatrix} \tag{5.48}$$

we can obtain

$$\begin{aligned} \ddot{x}_o &= \ddot{x}_o(q, \dot{q}, u, \ddot{x}_M) \\ \ddot{y}_o &= \ddot{y}_o(q, \dot{q}, u, \ddot{x}_M). \end{aligned}$$

Then it is possible to solve the problem computing the system of equations (5.44), (5.45), (5.47) plus (5.48), where we have six equations and six unknowns, namely f, N, $\ddot{x}_o$, $\ddot{y}_o$, $\ddot{\theta}_o$, and $\ddot{x}_M$, since $\ddot{y}_M = 0$.

Control-state space dimension The analysis for the control-state space dimension is similar to what we have done for the pure roll case. The big difference is that now $\dot{p}_M \neq 0$, but only its component along the y-axis is null, $\dot{y}_M = 0$. So we lose the condition on the velocity of the x_M variable, and the dimension of the control-state space becomes higher, $\|C_{slide-roll}\| = 13$.

Jump conditions Referring to figure 5.9, we have to define four jump conditions, which are the same whatever of the four modes is actually active. The condition to the free-flight mode is always the same relative to the normal force N; for the other jump conditions, we need to verify the value of the contact point velocity $\dot{x}_M$: as soon as this parameter becomes null we can assume the object not to be anymore in sliding mode. In addition to this, if $\theta_o = k\frac{\pi}{2}$, where $k \in \mathbb{Z}$, we assume the system in grasp *FF* mode, while if $\theta_o \neq k\frac{\pi}{2}$, the object is in pure rolling, where obviously the actual roll mode, either *FN* or *NF*, depends on the former slide-roll mode. Lastly, if the contact point velocity doesn't become null, but the object touches the surface of the manipulator $\mathcal{M}$ with two points, namely when $\theta_o = k\frac{\pi}{2}$, we can think the system in pure sliding where the direction of slipping depends on the direction of the former slide-roll mode. The conditions are wrapped up in table 5.4.

jump set D	θ_o	$\dot{x}_M$
dynamic grasp FF	$= k\frac{\pi}{2}$	$= 0$
slide LL*	$= k\frac{\pi}{2}$	< 0
slide RR*	$= k\frac{\pi}{2}$	> 0
roll FN*	$! = k\frac{\pi}{2}$	$= 0$
roll NF*	$! = k\frac{\pi}{2}$	$= 0$

Table 5.4: Jump conditions for the slide-roll modes. It's not written the condition to the free-flight mode. The * marks remembers that not all these jumps are feasible, depending on the current contact point.

Jump maps Like for the roll mode, also the slide-roll modes can introduce some impacts in the jumps. This happens again while jumping either to dynamic grasp or to two-point sliding modes. The jump map $G(z)$ is still (5.43), that is

$$G(z) = \{\dot{\theta}_o^+ = 0\}.$$

5.5 Free-flight

Last but not least, we need to describe the dynamics and the jump conditions when the system is in free flight. Trivially the object $\mathcal{B}$ is in free-flight when no normal forces are present because of absence of contact points with the manipulator $\mathcal{M}$. This means that, for example, the limit situation of $\mathcal{M}$ accelerating downward with an acceleration equal to the gravity with the object touching the surface of $\mathcal{M}$, since the normal force is zero, it is considered free-flight mode.

***NN* free-flight** The dynamics for this mode is given by

$$\begin{aligned} \ddot{x}_o &= u_x \\ \ddot{y}_o &= u_y - g \\ \ddot{\theta}_o &= u_\theta \end{aligned} \tag{5.49}$$

since, even if the object is flying uncontrolled, the manipulator is still able to move and vary the value of q_o.

Control-state space dimension As soon as the object starts to fly, it looks clear that the only acceleration which applies on the object is the gravity. Trivially the direction of the gravity affects only the y-axis, so the accelerations on the other two axes must be zero. Integrating these two conditions, we obtain two constraints on the velocity of the object, namely one on the linear velocity along the x-axis, and one on the angular velocity around the θ-axis. Those constraints are with respect to the fixed frame W. However those constraints are similar to those met in the dynamic grasp mode (5.2): they do not decrease the dimension of the control-state space $\mathbf{w}$ but they force the evolution of the state variables to follow, in this case, a ballistic motion. As expected, this means that it is not possible to change the velocity terms along x- and θ-axis once the system is entered in free-flight mode. Therefore, the dimension of the system in free-flight is not constrained, *i.e.*, $\|C_{flight}\| = 15$.

Jump conditions From this state it is possible to jump to any mode of the system: it seems clear that some of them are more easily accessible than others, like for example it is generally easier to land in one-point contact modes than in two-point contacts modes. It is important to notice that each

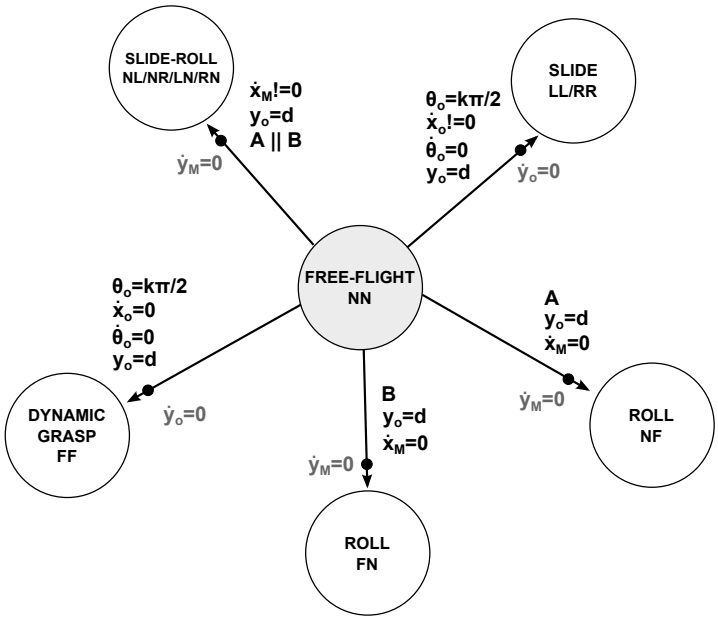


Figure 5.10: Free-flight mode jump conditions. Condition A is $\theta_o \in (k\pi, (k+1)\pi - \gamma)$, while B is $\theta_o \in ((k+1)\pi - \gamma, k\pi)$. The black circles indicate where impacts occur.

of those jumps creates an impact. These jump conditions usually concern the angle θ_o, the value of y_o, and the velocity of the object $\mathcal{B}$. Before analyzing

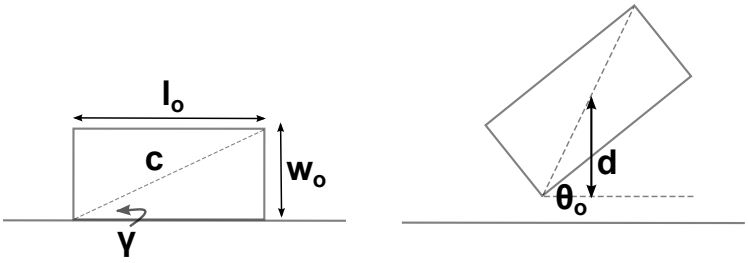


Figure 5.11: Minimum distance d. The left figure shows the initial configuration of the object which is needed to obtain the value of γ.

them, we need to define some new parameters for the system. Let us define c the value of the object diagonal (see figure 5.11); we introduce d as the minimum value of y which guarantees contact between the object and the manipulator surface, that is

$$d = \frac{c}{2}\sin(\gamma + \theta_o)$$

where γ depends on the initial configuration. Looking at the example in figure (5.7) γ is defined as

$$\gamma = \tan^{-1}\left(\frac{w_o}{l_o}\right). \tag{5.50}$$

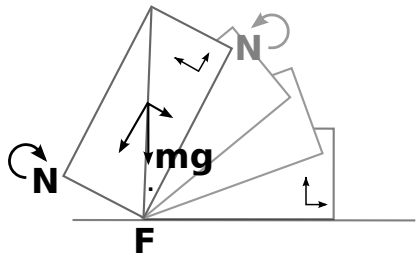


Figure 5.12: Impact angle. The dotted angle is γ.

It seems clear that as soon as the distance from the center of mass of the object is equal to d, the object is somehow touching the surface of the manipulator. It is less clear which mode is active when this condition happens. The condition $y_o = d$ is necessary for all the jump conditions. The value of the angle θ_o characterizes if the contact mode is jumping in a two- or one-point contact mode. In particular, if $\theta = k\frac{\pi}{2}$, where $k \in \mathbb{Z}$, the object $\mathcal{B}$ has two vertexes in touch with the manipulator $\mathcal{M}$, otherwise the contact is single. Jump conditions to dynamic grasp and sliding modes (the three two-point contact modes) share another constraint: the angular velocity must be zero, *i.e.*, $\dot{\theta}_o = 0$. To find differences in the three jump sets D for these modes, we need to check the x-velocity: if $\dot{x}_o = 0$ we can state the system is jumping to dynamic grasp, otherwise to sliding mode, where the sign of $\dot{x}_o$ makes $\mathcal{B}$ jump either to *LL* slide or *RR* slide. Concerning the one-point contact modes, namely rolling modes and slide-rolling modes, it is necessary to study both the object impact angle with the surface of the manipulator, and the velocity of the contact point p_M. The rolling direction, and so the relative mode, depends on the torques generated by the weight mg components. By studying this situation and with the help of figure (5.12), we find out that:

$$\begin{array}{rcccll} k\pi & < & \theta_o & < & (k+1)\pi - \gamma & \rightarrow \quad NF \text{ or } NF\text{-}like \\ (k+1)\pi - \gamma & < & \theta_o & < & k\pi & \rightarrow \quad FN \text{ or } FN\text{-}like \end{array}$$

where γ is still (5.50) and *FN-like* indicates slide-roll modes on the left contact point, like for example the *RN* slide-roll. The value of the velocity of the contact point along the x-axis, namely $\dot{x}_M$, differentiates the jumps to roll modes from those to slide-roll modes. In particular, if $\dot{x}_M = 0$ the system is jumping to roll, otherwise to the slide-roll. The direction of sliding is related to the sign of $\dot{x}_M$. The jump conditions are represented in figure 5.10 and summarized in table 5.5.

Jump maps The free-flight jumps always introduce impacts. The jump map $G(z)$ is different depending on the number of contact points: if the jump is to any one-point contact mode the map is

$$G(z) = \{\dot{y}_M^+ = 0\},$$

jump set D	θ_o	$\dot{x}_M$	$\dot{x}_o$	y_o	$\dot{\theta}_o$
dynamic grasp FF	$= k\frac{\pi}{2}$		$= 0$	$= d$	$= 0$
slide LL	$= k\frac{\pi}{2}$		> 0	$= d$	$= 0$
slide RR	$= k\frac{\pi}{2}$		< 0	$= d$	$= 0$
roll FN	B	$= 0$		$= d$	
roll NF	A	$= 0$		$= d$	
slide-roll RN	B	> 0		$= d$	
slide-roll LN	B	< 0		$= d$	
slide-roll NR	A	> 0		$= d$	
slide-roll NL	A	< 0		$= d$	

Table 5.5: Jump conditions for the free-flight modes.

where $\dot{y}_M$ is the velocity of the contact point, otherwise, for two-point contact modes

$$G(z) = \{\dot{y}_o^+ = 0\}.$$

Trivially, the other state variables evolve without discontinuity.

Chapter 6

Local topology graph

We are interested in describing the topology of the system, *i.e.*, we want to derive a description of the system in terms of hybrid system graph. This graph must show all the ten modes, each of them characterized by its own dynamic equations and represented by a state of the graph, and all the connections among these modes, as edges of the graph. In particular we are interested in understanding through which sequences of modes the system can evolve, in order to finally plan and control the state of the object in the workspace. In table 6.1, the adjacency matrix of the system shows all

	grasp	**roll**		**slide**		**slide-roll**				**flight**
	(FF)	*(NF*	*FN)*	*(LL*	*RR)*	*(NL*	*NR*	*LN*	*RN)*	*(NN)*
FF	1	1	1	1	1	1	1	1	1	1
NF	1	1	0	1	1	1	1	0	0	1
FN	1	0	1	1	1	0	0	1	1	1
LL	1	1	1	1	0	1	0	1	0	1
RR	1	1	1	0	1	0	1	0	1	1
NL	1	1	0	1	0	1	0	0	0	1
NR	1	1	0	0	1	0	1	0	0	1
LN	1	0	1	1	0	0	0	1	0	1
RN	1	0	1	0	1	0	0	0	1	1
NN	1	1	1	1	1	1	1	1	1	1

Table 6.1: System adjacency matrix. The ones mean connection between modes.

the possible connections between different modes. The topological graph is shown in figure 6.1. As we have just said each state has its own dynamics running when that state is active, and for any connection between states there are one or more jump conditions to be satisfied. The graph is not fully connected and this is also understandable by looking at the presence of zeros in the adjacency matrix. Not fully connected means that the system cannot transition between arbitrary states in a single jump; however it is possible

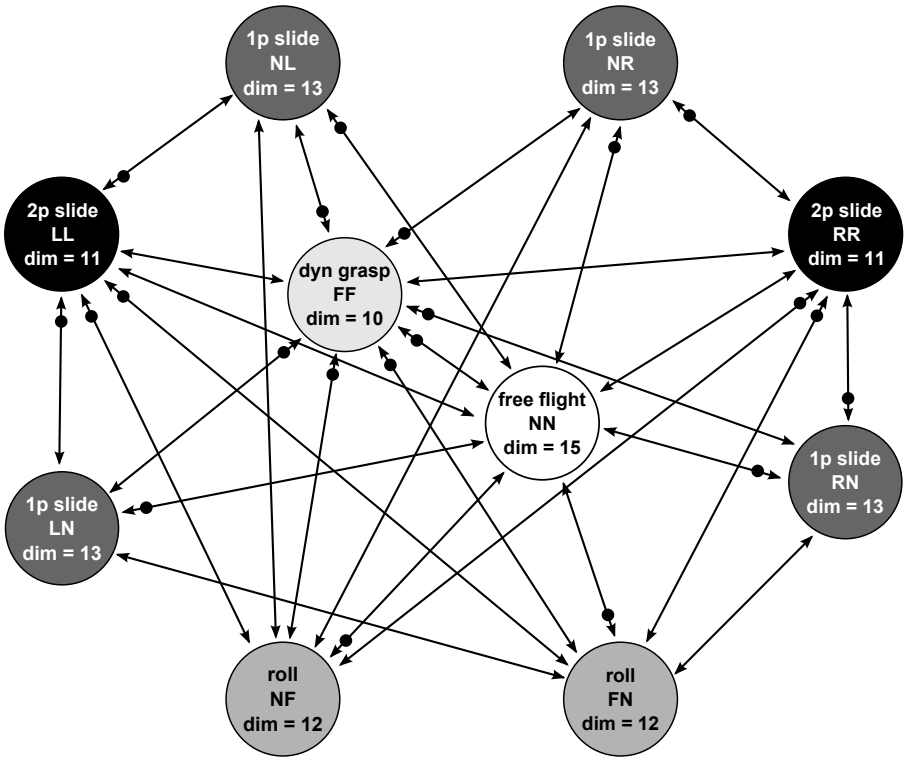


Figure 6.1: Local topological graph. This hybrid system shows the dimension of the control-state space $\mathbf{w}$ in each mode and all the connections between modes. The shades of the circles representing the various modes are associated with the type of contact. The black dots on the ends of the connections indicate where impacts occur. The numbers at the end of the connections represent the increase or decrease in set dimensions by making that jump.

to connect any two modes in two jumps at the most. Each mode dimension, depending on the number of constraints on the control-state space, is written on the third line of each node. Jumps which involve impacts are shown by black circles in figure 6.1. When a jump is characterized by an impact, like any jump from the free-flight mode, the jump maps are discontinuous in the velocity (in particular $\dot{y}_o^+ = 0$ for free-flight because of the hypotheses of plastic impact and no bounces). Another information provided by this graph is the increase or decrease in set dimensions by making any jump: it is shown on the end of each connection. For example, to jump from free-flight to dynamic grasp we decrease the dimension of the control-state space by five, that is, we constrain our system with five new constraints. This information will be useful to plan better sequences of modes since it is trivial to think that the less is the difference in dimension between two modes, the easier will be to take that jump.

Chapter 7

Trajectory planning

Contents

7.1 Introduction

In robotics, the term *trajectory planning*, or similarly *motion planning*, refers to problems such as how to move a piano from a room to another in a house without colliding with the environment, also known as the *piano mover's problem* in classical literature [7]. Our setup makes us focus on two different problems in the motion planning paradigm, both already handled in literature: the presence of kinodynamic constraints, and the specific framework of our system, that is an hybrid system.

Kinodynamic planning, term introduced in [8], refers to the problem of finding a feasible sequence of control inputs which drives the robot from an initial state z_i to a goal state z_g, while obeying to the physically-based dynamic model of the robot and, of course, avoiding any obstacle in the environment. We focus our attention on the analysis of LaValle's RRT algorithm in order to solve the kinodynamic planning.

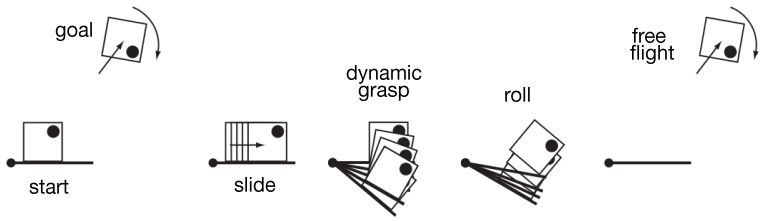


Figure 7.1: Example of manipulation task in five steps: the low-level algorithm plans within these five modes, while the high-level algorithm plans the four transitions between the steps. The latter planner also decides how to get from the start state to the goal one: it plans which sequence of modes the system must pass through, in this case *slide*, *grasp*, *roll*, and *free-flight*.

The problem of motion planning in hybrid systems has already been addressed in literature, for example in [9] where the authors used RRT algorithm as well. Generally speaking, our way to solve it involves engineering a two-layer planner, one of them, the *low-level planner*, which takes care of planning the trajectory for the robot within any single contact mode, within single states of the hybrid system, and the other, the *high-level planner*, which takes care of planning the conditions in which the system jumps from a state of the system to another (in figure 6.1), that is, it plans the transitions between states. And probably more important, it plans which sequence of modes the system must pass through. An example of possible manipulation task is shown in figure 7.1.

7.2 High-level planner: Sequence Planner

The sequence planner selects a contact mode that contains the start condition and a contact mode that contains the end condition and connects them through the hybrid manipulation graph (shown in figure 6.1 for our particular problem). Start and end conditions for the task are generally subsets of the combined state space and may lie in multiple contact modes. If the manipulator state is specified along with the object state, then these sets are each a point, but this point may still intersect multiple modes. The sequence planner samples from the possible start and goal sets to pick initial and final contact modes. A good planner would bias towards contact modes in which starting and ending are "easier" (*e.g.* dynamic grasp and free flight). Other useful attributes would be weighting the planner by controllability of a mode, constraints at contact mode jumps, and uncertainty properties. The only requirement of the planner is completeness: if there is a contact mode sequence that contains a feasible trajectory, the planner must find it.

Transitions are in general easier if the number of changing constraints (difference in dimension of flow sets in figure 6.1) is small. We can then bias

the search to find the smallest sum of the absolute value of the change in active constraints first, followed by the fewest jumps second. We restrict the graph search to enter each contact mode no more than twice to avoid infinite loops. In some contact modes, the flow set is not entirely reachable from each point, thus we allow two uses per contact mode. For example, in the free flight contact mode only points along the ballistic path of the object are reachable so another mode must be entered first to reach a different ballistic path. One problem with a pure graph search based on constraint switching is the fact that modes that rely heavily on friction like sliding modes are harder to transition in and out of accurately, as well as to control. Thus, we can add weights to the graph search into and out of these modes to discourage their selection. Avoiding sliding modes entirely could lead to a lack of completeness, so they must be accessible by the sequence planner.

The transition point selection is the next step before the low-level planners. The only requirement for transition point selection again is completeness: it must try all possible transition points before reporting failure.

7.3 Low-level planner: Contact Mode Planner

The low-level planner, as we have already mentioned, takes care of planning a trajectory for the robot within a single state of the hybrid system, which represents a single contact mode. Staying inside a single contact mode means being able to satisfy specific kinodynamic constraints. The literature about kinodynamic planning, and in general about trajectory planning, is quite recent but already full of solving algorithms, which can be gathered in three main families: roadmap based, probabilistic based, and potential field based algorithms. This partition concerns the different approach by every algorithm to address the problem of planning, and, trivially, it is not the only way of classifying planning algorithms. For example, another important term for comparison of these algorithms is their property of completeness.

In our 15 dimensional control-state space $\mathbf{w}$, and generally speaking in high-dimensional spaces, the main constraint is the so-called *curse of dimensionality*, that is the possibility for the algorithm to cause computational problems. For this reason, in high-dimensional spaces, roadmap-based algorithms, *i.e.*, those usually multiple-query algorithms which try to get a more or less detailed description of the environment, are practically not considered. Artificial potential field algorithms, sometimes called either *deliberative* or *reactive navigation* algorithms, based on the properties of the navigation itself, are more suited for on-line planning and generally they are also not considered in high-dimensional space environments.

Without fear of contradiction, the widest studied and used algorithms in planning problems in high-dimensional spaces are probabilistic methods, in particular referring to those algorithms containing randomized sampling

processes. The very basic idea of a sampling-based algorithm is to build a trajectory towards the goal configuration, avoiding to compute description of the environment but focusing on the capacity to get to the goal as quick as possible, sometimes avoiding the search for optimal or even just better solutions than the first obtained. Those algorithms sample the state space and build a roadmap too: the fact that they do not need a description of the $\mathcal{C}_{free}$ space, *i.e.*, the space in which the robot can move without any constraint but its own dynamics, allows to save memory and computational time. Computing the $\mathcal{C}_{free}$ space has usually exponential cost with the dimension of the space. This improvement in the computational performances makes probabilistic algorithms so powerful and widely used in high-dimensional spaces.

7.3.1 RRT: Rapidly-exploring Randomized Tree

One of the most used algorithm, probably the one with the highest number of different versions, is the Rapidly-exploring Randomized Tree algorithm (RRT) developed by Steven M. LaValle in 1998 [10]. Basically RRT algorithm builds a tree structure rooting it to the initial state, *root node*, and, through a process of random input generation, it spreads the branches of the tree rapidly covering the whole workspace. Eventually it reaches the goal state, *goal node*, providing the sequence of inputs needed to go from the root to the goal node. Each node of the tree is allowed to have multiple children or *leaves*, but a single *parent*. The pseudo-code in algorithm 1 shows the main loop of the RRT.

Algorithm 1 Pseudo-code of the RRT main loop

```
1: while j < J do
2:    z_rand ← RandomState()
3:    z_near ← NearestNode(T, z_rand)
4:    z_new ← ExtendNode(T, z_rand, z_near)
5:    if CollisionTest then
6:       T ← UpdateTree(T, z_new)
7:       j ← j + 1
8:    end if
9: end while
```

After the initialization of the tree T, the algorithm enters in a `while` loop which runs either until a solution is found or a number of samples J is reached. `RandomState` function samples randomly with uniform distribution z_{rand}, a state of the object, towards which the tree will try to evolve. `NearestState` function provides the closest node of the tree T to the sample z_{rand}: trivially the closeness measure depends on the metric of the system, since, for example, the simplest Euclidean metric might provide different

distance compared to an *ad-hoc* weighted metric system. Once the RRT has computed the closest node z_{near} and the sample z_{rand}, it tries to extend the tree in the direction of z_{rand}: the `ExtendNode` function can be considered the heart of the entire algorithm, and different versions of RRT often either make variation of this function or of the `NearestNode` function, trying to derive different and more suited metric systems on the particular problem that the algorithm has to solve. The very basic way of implementing the `ExtendNode` method is to try to connect directly z_{near} and z_{rand} using a straight-line path (fig. 7.2); however, generally speaking, this function tries to extend the nearest node of the tree as close as possible to the random sample. A collision test is then run in order to check possible contacts with the environment: if `CollisionTest` gives positive results, *i.e.*, there are no collisions, the new state z_{new}, that is the state reached from z_{near} in the direction of z_{rand}, is added to the tree T. Otherwise, the sample z_{rand} is thrown away, and a new random state is sampled.

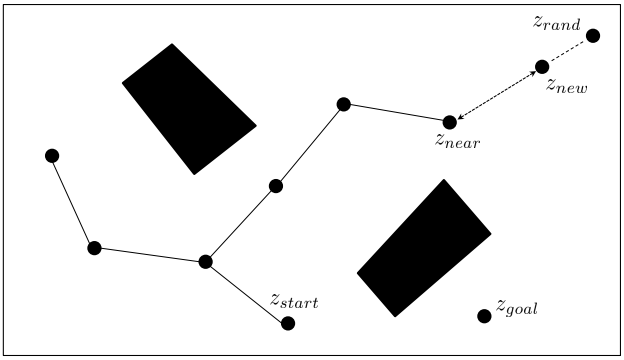


Figure 7.2: RRT algorithm: `ExtendNode` step. z_{new} is the state reached from z_{near} in the direction of z_{rand}, and since there are no collision with obstacles in the path, it is added to the tree as a new node of the tree. The black obstacles are shown even if they are not computed by the algorithm.

RRT algorithm is said to be probabilistic complete. Completeness is the ability by a planning algorithm of either finding a feasible solution or being able to give a failure if solutions do not exist. The latter capacity in particular is not provided by RRT: this algorithm can give a failure only after a threshold J has been reached. RRT is not able to determine if a solution actually does not exist: theoretically letting the RRT run, *i.e.*, making it sample other random states and extend towards them, it might be able to find a feasible path to the goal state eventually. This is the meaning of probabilistic completeness: if the time goes to infinity, that is if the number of nodes of the tree increases to infinity, the probability of finding a solution, if it exists, goes to 1.

Another important feature of RRT is that this algorithm is considered single-query, since a new initial condition would lead to initialize a whole

new tree structure.

7.3.2 RRT in kinodynamic environment

In kinodynamic environment an important role is played by the process of input generation, since not all the possible paths are feasible for the robot. A canonical example of kinodynamic planning is the automatic parking process for a car. Trivially, if the car is parallel to the parking spot, inputs which are supposed to drive the car laterally, are meaningless. LaValle and Kuffner in [11] developed an RRT version for addressing the problem of kinodynamic path planning.

The main improvement with respect to the algorithm 1 is the definition of the `ExtendNode` function. Mainly, once the RRT has provided the nearest node of the tree according to the metric ρ, a motion in the direction of the z_{rand} sample is done by applying an input $u \in U$ for some time increment Δt. In the set U which represents all the feasible inputs (if U is infinite, then a discrete approximation of it is obviously needed), the input u either can be randomly selected, or can be chosen in order to select that one which takes the tree closest to z_{rand}. Δt can be fixed, or randomly selected at each iteration in the range of values $(0, \Delta t_{max}]$.

Even if this implementation of the RRT algorithm works in kinodynamic environment, it has been proved its inefficiency [12], above all for those systems whose dimension is high. The inefficiency is related to the fact that RRT tends to expand nodes at the boundaries of the tree with little progress towards the goal state. To understand this inefficiency, we need to introduce the well-known property of *Voronoi bias* featuring the RRT. In figure 7.3[1] it is shown an example of Voronoi diagram.

Definition 1 (**Voronoi region**). *Given a set of points (seeds or generators), for each seed its Voronoi region is the region consisting of all the points closer to that seed than to any other seed.*

RRT is said to be Voronoi biased because the probability for a generic node of the tree to be expanded is directly proportional to the volume of its Voronoi region. Nodes with larger Voronoi region are more likely to be chosen for expansion and are referred to as *major* nodes. With respect to Euclidean metric, those nodes are often on the outside of the tree, at least during the initial exploration. Conversely, *minor* nodes have smaller Voronoi region and tend to be on the inside of the tree. This property, above all not taking care of kinodynamic constraints, yields towards an initial preference in rapidly exploring the state space. As the number of samples increases, the Voronoi regions become more uniform in volume, and the likelihood of expanding a node tends toward the sampling distribution.

[1] Figure 7.3 from `http://www-cs-students.stanford.edu/~amitp`

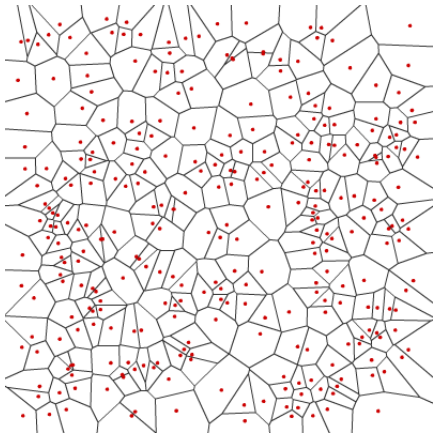


Figure 7.3: Example of Voronoi diagram. Red dots are the *seeds*.

At each iteration of the main loop of the algorithm, the most of the computational cost lies in the expansion step. Evolving the tree means finding a feasible input, integrate the dynamics over Δt, check for collision, and update the tree structure. The efficiency of RRT is highly sensitive to the ability of choosing "good" nodes to be expanded, in order not to decrease the performance of the algorithm, and so, it is highly sensitive to the distance metric.

7.3.3 RG-RRT: Reachability-Guided RRT

A recent version of the RRT, namely the *Reachability-Guided* Rapidly-exploring Randomized Tree algorithm (RG-RRT), provides a powerful tool to choose smartly the nodes to be expanded. RG-RRT was developed by A. Shkolnik, M. Walter, and R. Tedrake in 2009 [13]. This algorithm leverages on the idea that the processes of sampling random states and checking the presence of collisions are less computationally expensive than evolving the tree; adding a new node means the necessity of finding feasible inputs in a kinodynamic constrained environment, and trivially the size of the trees grows up which leads to slow down the whole algorithm. So instead of expanding nodes at the boundaries of the tree like in RRT, RG-RRT tries to make cleverer picks of the nodes to be expanded, by introducing the concept of *reachable set*.

Definition 2 (**Reachable Set**). *Given a state x, its reachable set $R(x)$ is the set of all the states that can be reached by x in a finite time Δt, taking into account the dynamics of the system and the feasibility of the control inputs.*

RG-RRT builds a standard tree structure like in the RRT algorithm, and simultaneously it stores a representation of the R set for each node of

the tree as well. The key of the algorithm is in the different way of deciding which node is a good node to be expanded: this choice is made by the function `RGTest`, figure 7.4.

Algorithm 2 Pseudo-code of the RG-RRT main loop

```
1: while j < J do
2:    while !RGTest(z_near, R(z_near)) and !CollisionTest do
3:       z_rand ← RandomState()
4:       (z_near, R(z_near)) ← NearestNode(T, z_rand)
5:    end while
6:    z_new ← ExtendNode(T, z_rand, z_near, R)
7:    T ← UpdateTree(T, z_new)
8:    R(z_new) ← UpdateRSet(z_new)
9:    j ← j + 1
10: end while
```

`RGTest` function compares the distance between the nearest node of the tree z_{near} and the random state z_{rand}, and the distances between each state of the reachable set $R(z_{near})$ and z_{rand}. If any element of $R(z_{near})$ is closer to z_{rand} than z_{near} itself, the tree is extended towards this element. Otherwise, the z_{rand} state is discarded and a new random state is sampled. This policy of picking particular nodes to be expanded, reflects in a change in the Voronoi biasing for the algorithm. By applying `RGTest` function in RG-RRT algorithm, only those nodes which may expand towards the random state inside of their Voronoi regions, can be considered like suitable nodes for the `ExtendNode` function, regardless of the dimension of their Voronoi regions. Those nodes with large Voronoi regions which are not able to evolve the tree towards the random state are thrown away. This leads to a smaller number of samples needed to reach the goal state, which means, above all in kinodynamic planning, in better performances of the algorithm.

There are small differences in applying the RG-RRT algorithm among the different contact modes. In particular the process of input generation, the dynamics of the system object-manipulator, and the definition of reachable set change.

7.3.4 Input generation

By calling the `ExtendNode` function, the algorithm generates feasible random inputs for the robot and chooses the "best" one, meaning the input who leads the object closest to the new sample z_{rand}. The input generation process varies and depends on the contact mode the system is in. However the inputs provided to the manipulator must be able to guarantee that the motion is kept within the contact mode, without reaching any jump condition, apart from the initial and final states. The general process of input generation is

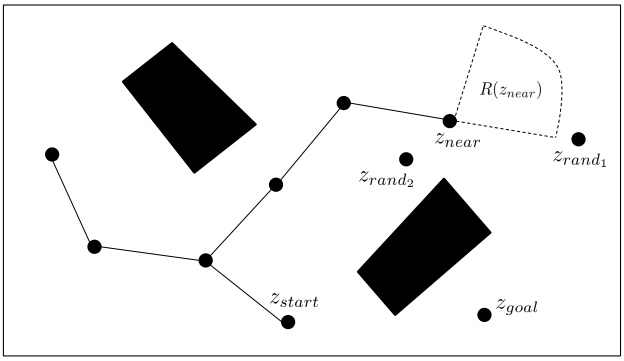


Figure 7.4: `RG-Test` example: the z_{near} node is suitable to expand towards z_{rand_1}, while z_{rand_2} is discarded since any state in the reachable set $R(z_{near})$ is further than z_{near}, that is $\forall z \in R(z_{near}) : |z_{near} - z_{rand_2}| < |z - z_{rand_2}|$, where we use for this simple example the Euclidean metric.

shown in algorithm 3.

Algorithm 3 Input generation process

1: $k_i \leftarrow$ `NumberGenerator`
2: $w_i^B = \sum_i k_i C_{ij}$
3: $w_i^W = A_{g_{BW}}^T w_i^B$
4: $u_i \leftarrow$ `WrenchToAcceleration`$(\mathbf{w}_i^W)$

Dynamic Grasp In order to hold the dynamic grasp contact mode, the inputs must lay inside the friction cone (5.6 and figure 7.5). Once that a coordinate frame has been chosen, for example the body frame B, the forces which describe the friction cones at the contact points can be expressed as wrenches w_i where $i = 1, 2, 3, 4$. These four wrenches describe a composite wrench cone, figure 7.6: every linear combination of w_i, mainly $w = \sum_i k_i w_i$, guarantees that condition (5.6) is satisfied, that is, each w keeps the motion inside the grasp mode (line 2 of the algorithm 3, where C_{ij} is the comoposite wrench cone).

In particular, the composite wrench cone is the same for each contact mode and is expressed by

$$C = \begin{pmatrix} -\sin(\sigma) & \cos(\sigma) & -d\cos(\sigma - \delta) \\ \sin(\sigma) & \cos(\sigma) & -d\cos(\sigma + \delta) \\ -\sin(\sigma) & \cos(\sigma) & d\cos(\sigma + \delta) \\ \sin(\sigma) & \cos(\sigma) & d\cos(\sigma - \delta) \end{pmatrix} \tag{7.1}$$

where $\sigma = \tan^{-1}(\mu)$, μ is the coefficient of friction, $d = (w_o^2 + l_o^2)^{\frac{1}{2}}$ is the semidiagonal of the rectangular object of dimension $(2 \times w_o)$ and $(2 \times l_o)$,

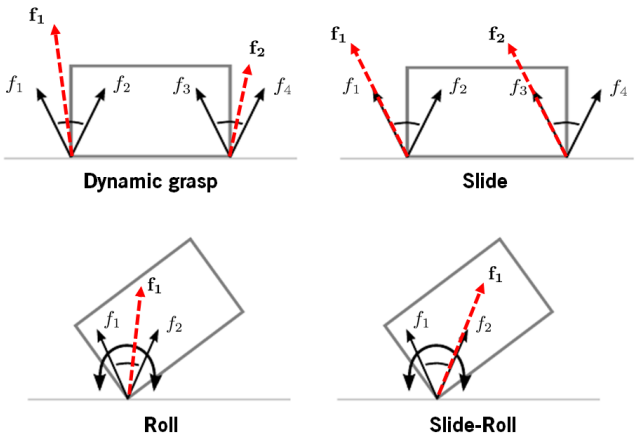


Figure 7.5: Force vector application in the input generation process. In dynamic grasp and roll, the forces (dotted red vectors) must be inside the friction cone, while in slide and slide-roll they must be on the edge of the cone.

and $\delta = \tan^{-1}(\frac{l_o}{w_o})$ is the angle between the diagonal and the basis of the rectangle.

Obviously we still need to provide acceleration inputs in the world frame W, given this wrench w in the body frame B. This step is shown in line 3 of the algorithm, where $A^T_{g_{BW}}$ is the Adjoint matrix (see Appendix B for more details about wrench and change of coordinate system). Once we obtain the wrench in the world frame W, computing the accelerations is straightforward, keeping in mind the presence of the gravity acceleration (line 4 of the algorithm). Trivially, since the RG-RRT algorithm needs random inputs, we can obtain random inputs choosing randomly the coefficients k_i (in this case, `NumberGenerator` in line 1 generates random numbers in a subset of $\mathbb{R}$, where this subset is defined by physical limits on the feasible accelerations).

Sliding mode In order to keep the sliding mode, we need to guarantee (5.24) and (5.25), that is, we need to guarantee that the forces lay on the edges of the friction cones, having the same direction (figure 7.5). What changes in algorithm 3 is the `NumberGenerator` function which generates the k_i coefficients. In this case, we need them to be random but chosen in the way that if $f_1 > 0$ then $f_2 = 0$ and vice versa, and the same behavior for f_3 and f_4. We can write this condition as $|\vec{f_1}||\vec{f_2}| = 0$. Besides we need that if $f_1 > 0$ then also f_3 must be greater than zero, that guarantees that the object is sliding in one direction.

Rolling mode Rolling motion is guaranteed by making the only contact point fixed on the manipulator surface. The condition for this contact point

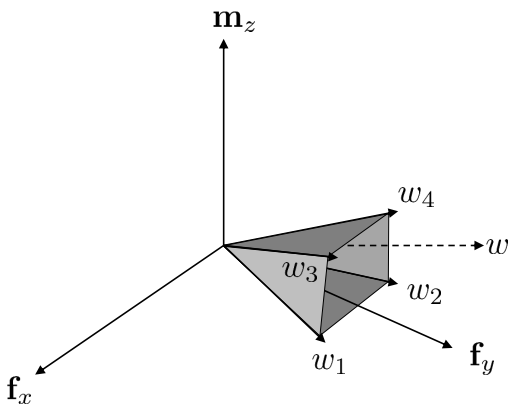


Figure 7.6: An example composite wrench cone resulting from two frictional contacts. w is a linear combination of the four wrenches w_i, where $i = 1, 2, 3, 4$.

is the same for those in dynamic grasp, that is, the forces applied to it must lay within the friction cone (figure 7.5). This guarantees that the velocity of the contact point relatively to the manipulator is zero. Obviously no forces are applied on the other vertex of the object.

Slide-Roll mode The object simultaneously slides and rolls on a single point of contact if the forces applied to this point lay on the edge of the friction cone. Obviously no forces are applied on the other vertex of the object. Figure 7.5 shows this condition.

Free-Flight mode In this mode, we do not apply directly accelerations to the object since it is following a ballistic path and so it is not in contact with the manipulator. The input are chosen in order to obtain a correct catch, at the desired state and contact mode.

7.3.5 Reachable set approximation

Each contact mode (except free flight) has a set of forces that can be applied to the object described either by the full friction cone or the edge of a cone at each contact point. We approximate the reachable set by integrating forward the limits of our force inputs for the mode one time step as well as integrating forward the zero-force input. This provides a set of "maximal" states that the system can reach in that time step, and by taking the convex hull of those points in the state space we have a polyhedral reachable set approximation. Trivially the smaller the time step, the better the approximation.

7.4 Completeness

The completeness of the entire planning algorithm depends on the properties of both level planners. As stated in section 7.2, the sequence planner algorithm must look for a path connecting two nodes of the manipulation hybrid graph. There are an infinite number of feasible paths, but the rule which limits the number of loops guarantees an exhaustive search that the sequence planner is always able to find a feasible sequence to give to the low-level planner. Then, it receives a solution or a reported failure by the low-level planner. If a failure is reported, the next feasible sequence is provided until all have been tested, at this point there is no solution.

The low-level RG-RRT algorithm is probabilistic complete, *i.e.*, if a solution exists then probability of finding a solution goes to one as run time goes to infinity. However this means that the RG-RRT algorithm is not able to determine if a solution does not exist. By imposing a maximum number of samples or iterations, the RG-RRT can report a failure. This prevents the algorithm from looking for a solution in a specific contact mode forever.

Since our high level planner will find a feasible sequence if it exists, and our low-level planner is probabilistic complete, then the overall planner is probabilistic complete.

Chapter 8

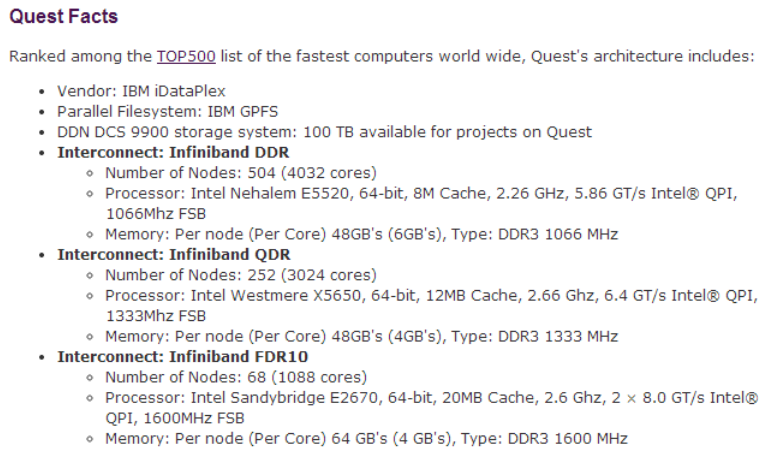
Simulations

In this chapter we want to describe some of the simulations we have done to discuss some useful aspects and interesting results we have obtained. We present three different scenarios: the first scenario is a comparison between the performances of RG-RRT algorithm and its "father" RRT algorithm to solve nonprehensile manipulation tasks; the second simulation is an example of solution for contact mode planning, in particular for a trajectory in dynamic grasp; last but not least the third simulation is a full trajectory at the sequence planner level: we show a transition from dynamic grasp to roll, the two local trajectories within the two contact modes, and the final balance about the equilibrium.

For all the simulations we use the following data:

Parameters	**Value**
Air-hockey table grade	23°
Gravity acceleration	$\sim 3.8\,\mathrm{m/s^2}$
Object mass	$0.075\,\mathrm{kg}$
Coefficient of friction	1.5
Object dimensions	$0.045\,\mathrm{m} \times 0.051\,\mathrm{m}$
Time step	$0.01\,\mathrm{s}$
Workspace dimension	$0.5\,\mathrm{m} \times 0.5\,\mathrm{m}$
High-speed camera field of view	$0.76\,\mathrm{m} \times 1.2\,\mathrm{m}$
Acceptable error	$< 5\%$

The fact that the high-speed camera FOV is larger than the workspace dimension is useful to track the object in free-flight mode, that is the mode which allows us to "increase" the workspace dimension. All these simulations run on QUEST, the Northwestern University's High Performance Computing System: some details about this cluster are available in figure 8.1. The code is written in Matlab and part of it is available in Appendix C.



Quest Facts

Ranked among the TOP500 list of the fastest computers world wide, Quest's architecture includes:

- Vendor: IBM iDataPlex
- Parallel Filesystem: IBM GPFS
- DDN DCS 9900 storage system: 100 TB available for projects on Quest
- **Interconnect: Infiniband DDR**
  - Number of Nodes: 504 (4032 cores)
  - Processor: Intel Nehalem E5520, 64-bit, 8M Cache, 2.26 GHz, 5.86 GT/s Intel® QPI, 1066Mhz FSB
  - Memory: Per node (Per Core) 48GB's (6GB's), Type: DDR3 1066 MHz
- **Interconnect: Infiniband QDR**
  - Number of Nodes: 252 (3024 cores)
  - Processor: Intel Westmere X5650, 64-bit, 12MB Cache, 2.66 Ghz, 6.4 GT/s Intel® QPI, 1333Mhz FSB
  - Memory: Per node (Per Core) 48GB's (4GB's), Type: DDR3 1333 MHz
- **Interconnect: Infiniband FDR10**
  - Number of Nodes: 68 (1088 cores)
  - Processor: Intel Sandybridge E2670, 64-bit, 20MB Cache, 2.6 Ghz, 2 × 8.0 GT/s Intel® QPI, 1600MHz FSB
  - Memory: Per node (Per Core) 64 GB's (4 GB's), Type: DDR3 1600 MHz

Figure 8.1: Northwestern University's High Performance Computing System details. More information available at `http://www.it.northwestern.edu/research/adv-research/hpc/quest/`

8.1 RG-RRT vs RRT

This simulation is somehow similar to what is shown in [26], by the authors of the RG-RRT algorithm, since the object of this simulation is to show that RG-RRT overcomes RRT in performance for manipulation tasks, that is, for environments with kinodynamic constraints. For this simulation we run the two algorithms 1000 times with a threshold of 10 000 samples, and the simple task is to lead the object as close as possible to the goal state in dynamic grasp. The start state is the origin of the reference system at rest, while the goal state is the object horizontally 20 cm to the left and 10 cm up of the start state, reaching this configuration with zero velocity. We choose the best input, generating every time 20 random feasible inputs (line 6 of Algorithm 2). The metric is Euclidean with weights on the velocity distances to give more importance to distances in configuration. The distance is not absolute but relative to the maximum range of variation of the single state component, *e.g.* both the x and y distances are divided by the workspace dimensions along these two components. Both the algorithm are biased towards the final state with a bias parameter set to 30%, that is, three times out of ten the algorithms select the goal state as the random sample to expand towards.

The results of this simulation are shown in figure 8.2. In particular we represent the differences in configuration of the two algorithms after 10 000 samples. We do not show differences about θ configuration, as well as differences in velocity: the reason is that both the algorithms obtain acceptable results on these components of the state, with no interesting differences to describe. The result is that RG-RRT is able to arrive 50.9%

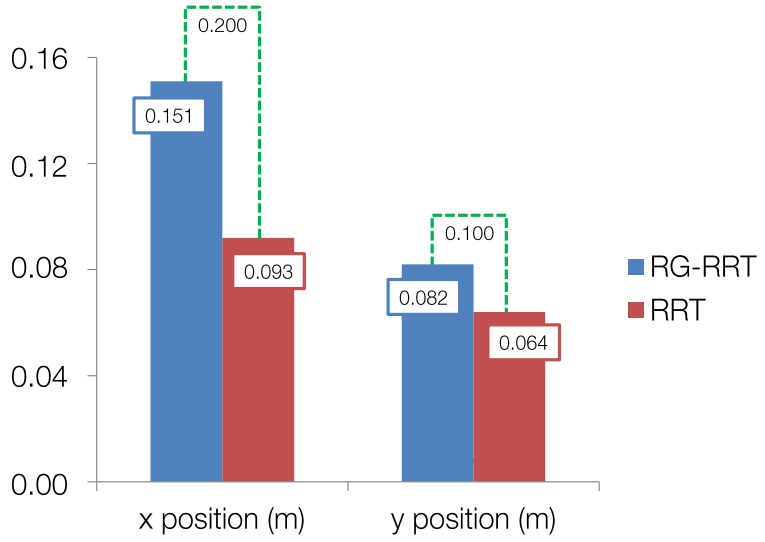
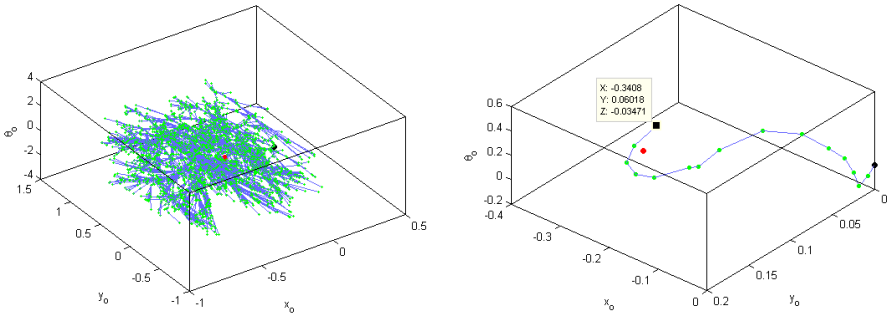


Figure 8.2: Comparison between RG-RRT and RRT algorithms out of 10 000 samples. The target position is $x = 0.2\,\text{m}$, $y = 0.1\,\text{m}$, and it is shown by the dashed line in green on the plot. No relevant differences are found neither along the θ axis, nor in velocity values.

closer to the goal state than RRT, on the average of the 1000 runs for this simulation. This is what we were expecting, meaning that the choice of using reachable set theory in kinodynamic environment generally leads to better performances.

8.2 Contact mode planning

This second simulation wants to show the solution of a case of contact mode planning. In particular we want to design a trajectory in dynamic grasp to bring the object from a start state (object at rest at the origin of the reference frame, that is $q_o = 0\,\text{m}$ and $\dot{q}_o = 0\,\text{m/s}$) to a goal state ($x_o = -0.3\,\text{m}$, $y_o = 0.1\,\text{m}$, $\theta = 0\,\text{rad}$, and zero velocity $\dot{q}_o = 0\,\text{m/s}$). The simulation time is 1.3 s and the solution is found after 5287 samples. In figure 8.3a we show the full configuration tree, plotted directly from Matlab, made of 5287 samples. We are not showing the corresponding tree for the velocity. In figure 8.3b it is shown the solution for this manipulation task, that is a sequence of 16 consecutive nodes which brings the object from the initial state close enough to the final state to consider that a solution has been found. Thus the trajectory takes 1.6 s. Obviously, the range around the solution to consider a state a valid solution, can be modified by the user in the algorithm, and decreasing this range leads to an increase in the number of samples needed to find a solution. We run this simulation 1000 times, and the average number of samples needed to solve it, is about 30 000. This



(a) Configuration tree (b) Solution trajectory

Figure 8.3: The full configuration tree after 5287 samples (a), and the trajectory solving the dynamic grasp task (b). The final state is $x_o = -0.3408\,\text{m}$, $y_o = 0.06018\,\text{m}$, $\theta_o = -0.03471\,\text{rad}$. The green dots are the states reached after each forward integration of the dynamics of the system, while the red dot and the black one are respectively the goal and the start state.

simulation, which just needed 5000 samples, is one of the quickest obtained.

8.3 Sequence planning

The third and last simulation we want to show in this thesis, is an example of full nonprehensile manipulation task including a transition between two different contact modes. In particular we describe the task of balancing on one-point rolling the object, starting from a rest position in static grasp. For this simulation the sequence planner provides the transition state, which represents respectively the goal state for the dynamic grasp mode, and the initial state for the roll. We found a transition state that is close enough to the final equilibrium configuration. Trivially in this case the sequence planner does not have to provide a sequence of contact modes too, since we are choosing to obtain one single transition from grasp to roll. In order to obtain a smoother and generally better trajectory we run the RG-RRT algorithm up to almost 100 000 samples to obtain the two trajectories used in this simulation. It is important to underline that this simulation implied the use of feedback controllers to obtain the final result. Although the design of a control law is not part of this thesis, we want just to introduce what was used in this simulation. For the first part of the simulation, namely the dynamic grasp trajectory, a controller consisting of a feedforward term plus a PD like feedback controller with inputs as object accelerations was used. For the second part of the trajectory, the rolling mode, we were able to address the object balancing problem in the manner of an inverted

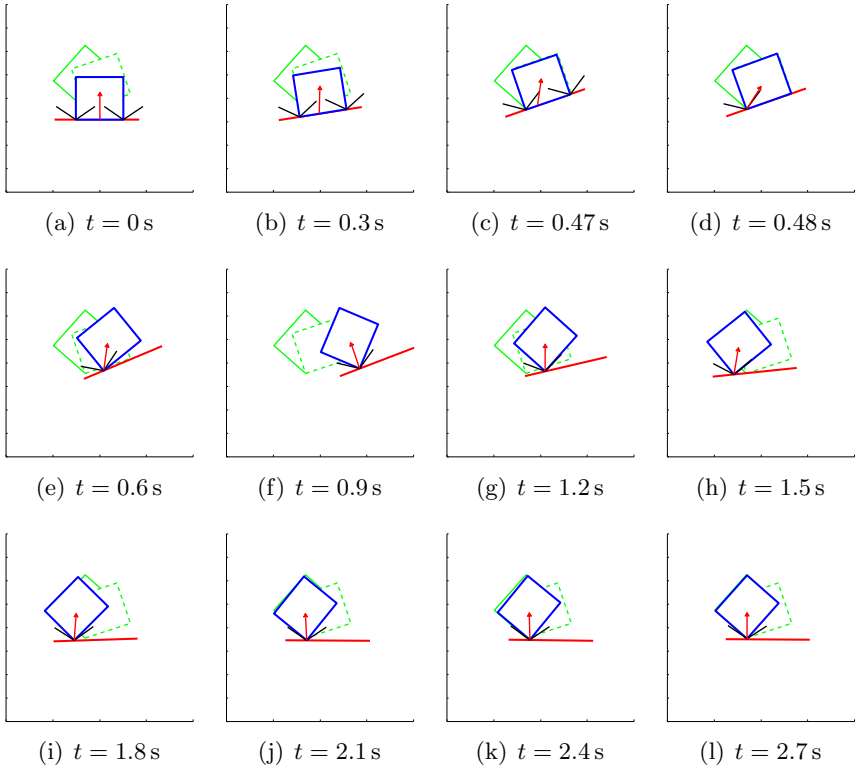


Figure 8.4: Simulation of a transition from dynamic grasp (figures a to c) to one-point roll (d to l). The black lines at the contact points represent the bounds of the friction cone, while the red vector is the (normalized) contact force. While it is very close to breaking the friction constraint at the transition to one-point rolling, the contact force remains inside the cone. The dashed green outline is the desired transition position of the object, and the solid outline is the desired final position of the object.

pendulum system, so the LQR controller used in literature[1] to solve the inverted pendulum problem was used here as well.

Figure 8.4 shows the resulting trajectory for the manipulation task. It is a sequence of twelve snapshots taken from an animation in Matlab of the trajectory. In the first snapshots, namely figures a to c, the system is in dynamic grasp, while in the last snapshots, from figure d to the final figure l, the system is in one-point rolling. The dashed green outline, in each figure, is the desired transition position of the object, and the solid outline is the desired final position of the object. The animation shows the bounds of the friction cones (two in grasp, and one in roll) by using two black lines at the contact points. Besides, it shows also the normalized contact force by using a red vector. The whole trajectory happens quite fast (the final time

[1]Details can be found, for example, in `http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlStateSpace#6`.

is 2.7 seconds) and, above all, the transition is just after $t = 0.47\,\mathrm{s}$. This reflects in what we said in the introduction of this thesis about general need in nonprehensile manipulation for high-speed sensing and high-bandwidth control of the manipulator.

Chapter 9

Conclusions

This thesis addressed the general problem of hybrid nonprehensile dynamic manipulation, applying the analysis to the specific case of a rectangular object on the top of a flat manipulator. We were interested in providing a model of the system and in designing a trajectory planner for it.

In the first part of the thesis, we classified and modeled the interactions between the 3-*dof* manipulator and the object, and derived a hybrid description of the local topology of the system. We gathered the 10 combinations of contacts in five contact modes: *grasp*, *roll*, *slide*, *slide-roll*, and *free-flight*. For each mode, we provided the dynamics of the contact, the jump conditions for the feasible jumps to other modes, and the control-state space dimension. The resulting hybrid graph, which shows dynamics of the system and possible connections among them, is figure 6.1.

In the second part of this work, we designed a trajectory planning algorithm in order to plan trajectories within the hybrid system. We chose to divide the planner in two hierarchical levels: a high-level planner, the *sequence planner*, which takes care of both planning feasible sequences of modes and finding transition conditions in the jump sets between the modes, and a low-level planner, the *contact mode planner*, which takes care of planning within single contact modes. We utilized RG-RRT algorithm for the contact mode planner, and an *ad-hoc* solution for the sequence mode planner for which we provided some guidelines.

Final simulations showed performances of our solution, in particular successfully solving the task of manipulating the object from dynamic grasp to one-point rolling.

9.1 Future work

Many interesting problems have to be solved yet, and definitely this thesis is just a specific case-study of a small subset of the properties of nonprehen-

sile manipulators. In the next future we will keep on analyzing this specific problem as well as trying to extend the properties we have found to the general case of generic multi-bodies object and generic multi-bodies manipulator. We are interested in analyzing the differences in using non-polygonal convex manipulator, as well as non-convex objects. The final goal is to extend the analysis of hybrid dynamic nonprehensile manipulation to tasks in the 3D space, not only constrained to the plane of our air-hockey table.

For the planning part, we will try to improve our algorithms, as well as to study new techniques which may enhance performances of the full planner. To improve RG-RRT we are thinking about changing the nearest searching function by implementing the k-nearest neighbor algorithm (k-NN) to speed up the searching loop in the contact mode algorithm which is the heaviest process in the entire algorithm. We are interested in providing some properties of optimization to our algorithm, so in the future we will focus more our attention on algorithms like RRT* [22] or CHOMP [53] to try to include some of their features to our solution. Another issue with RRT-like algorithms is that our solution often is fragmented and highly discontinuous, so we are interested in finding a way to smooth the trajectory provided by RG-RRT in order to provide solutions which our controllers can track and stabilize more easily.

Definitely a question still open is how to realize controllers for each contact mode, in particular improving the *ad-hoc* solutions showed for the simulations in this thesis, and developing controllers for the other contact modes, like sliding.

Appendix A

Mechanics in non-inertial frames

We want introduce some basic concepts of mechanics in non-inertial frames we used in this paper. To get more into details you can refer to [14]. As we know, Newton's Laws govern the mechanics in the inertial world. Consider a particle point of mass m and position q_w with respect to the world frame W, figure A.1. We can easily state that

$$F = m\ddot{q}_w$$

because W is the inertial frame.

To keep this law still valid into a non-inertial reference frame we have to take into account that our reference is accelerating with respect to the world. This simple consideration implies that, to use Newton's Laws even in the non-inertial world, we must add to our analysis the so called *fictitious forces*: these forces may arise due to either linear relative accelerations or angular relative accelerations between the frames. We can derive the mathematical description of the fictitious forces starting from the simple property of vector addition:

$$q_w = q_{wb} + q_b$$

and differentiating it

$$\dot{q}_w = \dot{q}_{wb} + \left.\frac{d\, q_b}{dt}\right|_w .$$

To solve the derivative of a vector Q as measured in the inertial frame W to the corresponding derivative in the non-inertial frame B, we can use this important identity:

$$\left.\frac{d\, Q}{dt}\right|_w = \left.\frac{d\, Q}{dt}\right|_b + \Omega \times Q \tag{A.1}$$

where Ω is the angular relative velocity between the two frames. Therefore

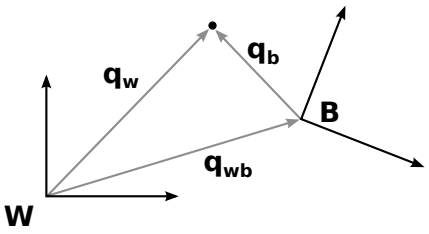


Figure A.1: W is the inertial world frame, B is the non-inertial body frame.

substituting,

$$\begin{aligned}\dot{q}_w &= \dot{q}_{wb} + \left.\frac{d\,q_b}{dt}\right|_b + \Omega \times q_b \\ &= \dot{q}_{wb} + \dot{q}_b + \Omega \times q_b\end{aligned}$$

Again, differentiating the velocities we obtain

$$\begin{aligned}\ddot{q}_w &= \ddot{q}_{wb} + \left.\frac{d\,\dot{q}_b}{dt}\right|_w + \left.\frac{d}{dt}(\Omega \times q_b)\right|_w = \\ &= \ddot{q}_{wb} + \left.\frac{d\,\dot{q}_b}{dt}\right|_b + \Omega \times \dot{q}_b + \left.\frac{d}{dt}(\Omega \times q_b)\right|_b + \Omega \times (\Omega \times q_b) = \\ &= \ddot{q}_{wb} + \ddot{q}_b + \Omega \times \dot{q}_b + \dot{\Omega} \times q_b + \Omega \times \dot{q}_b + \Omega \times (\Omega \times q_b) = \\ &= \ddot{q}_{wb} + \ddot{q}_b + 2(\Omega \times \dot{q}_b) + \dot{\Omega} \times q_b + \Omega \times (\Omega \times q_b).\end{aligned}$$

Finally, multiplying by m we find that

$$F = m\ddot{q}_b + F_{fictitious} \tag{A.2}$$

where

$$F_{fictitious} = m\ddot{q}_{wb} + 2m\dot{q}_b \times \Omega + mq_b \times \dot{\Omega} + m(\Omega \times q_b) \times \Omega. \tag{A.3}$$

All these components of the fictitious force have a proper physical meaning which we are not going to explain. We just want to name all these forces to complete this basic analysis.

The first of these extra-terms is called *inertial force*

$$F_{ine} = m\ddot{q}_{wb} \tag{A.4}$$

and it is strictly related to the linear relative acceleration between the two frames. The second is the so-called *Coriolis force*, after the French physicist G.G. de Coriolis who was the first to explain it,

$$F_{cor} = 2m\dot{q}_b \times \Omega. \tag{A.5}$$

The third is the *Euler force*, named for the Swiss mathematician and physicist Leonhard Euler,

$$F_{eul} = mq_b \times \dot{\Omega}. \tag{A.6}$$

The last is the famous *centrifugal force*,

$$F_{cen} = m(\Omega \times q_b) \times \Omega. \tag{A.7}$$

Appendix B

Wrenches and change of coordinates

A generalized force acting on a rigid body consists of a linear component and an angular component applied to a point of the body. In $\mathbb{R}^6$ we can represent this generalized force as a 6-by-1 vector:

$$F = \begin{bmatrix} f \\ \tau \end{bmatrix} \qquad \begin{matrix} f \in \mathbb{R}^3 & \text{linear component} \\ \tau \in \mathbb{R}^3 & \text{angular component} \end{matrix}$$

This force/moment pair is called *wrench*. The values of the wrench vector F depend on the coordinate frame in which the forces and the moments are expressed. If B is a coordinate frame attached to a rigid body, $F_b = (f_b, \tau_b)^T$ indicates a wrench applied at the origin of B.

We may be interested in describing F_b in a different reference frame, for example the world frame W. In order to find the description of F_b in W we need to introduce the concept of *equivalence* between two wrenches. Two wrenches are said to be equivalent if they generate the same work for every possible rigid body motion. This fact can be exploited to rewrite a given wrench in terms of a wrench applied at a different point and with respect to a different coordinate frame. Let $g_{bw} = (p_{bw}, R_{bw})$ be the configuration of the frame W relative to the body frame B, where p_{bw} is the vector of the relative coordinates between the origins of the two frames, and R_{bw} is the rotational matrix which describes the rotation between the two frames. By equating the instantaneous work done by the wrench F_b and the wrench F_w over an arbitrary interval of time, skipping some easy simplifications, we obtain that

$$F_w = A^T_{g_{bw}} F_b. \tag{B.1}$$

Equation (B.1) says us how to transform a wrench applied in a frame, in this case B the body frame, in a different frame, in this case the world frame W. In particular, expanding the adjoint matrix $A^T_{g_{bw}}$, we have

$$\begin{bmatrix} f_w \\ \tau_w \end{bmatrix} = \begin{bmatrix} R^T_{bw} & 0 \\ -R^T_{bw}\hat{p}_{bw} & R^T_{bw} \end{bmatrix} \begin{bmatrix} f_b \\ \tau_b \end{bmatrix}, \tag{B.2}$$

where the term $\hat{p}_{bw}$ stands for the *skew-symmetric* matrix of p_{bw}. The adjoint matrix rotates the force and torques vectors from the B frame into the W frame and includes an additional torque of the form $-p_{bw} \times f_b$, which is the torque generated by applying a force f_b at a distance $-p_{bw}$. More details about wrench and changes of coordinates can be found in [15].

Appendix C

Trajectory planning algorithm

What follows is the *Matlab* code for the RG-RRT algorihtm.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RG-RRT trajectory planner
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear; close all;
parameters_RGRRT % .m file with robot parameters
rrt = 0; % run the RRT version

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% User Inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% object initial state
pos_i = [0 0 0]; % position [m], [m], [rad]
vel_i = [0 0 0]; % velocity [m/s], [m/s], [rad/s]

% object goal state
pos_g = [-0.2 0.1 -pi/12]; % position [m], [m], [rad]
vel_g = [0 0 0]; % velocity [m/s], [m/s], [rad/s]

threshold = 10000; % max number of samples
delta_t = 0.01; % constant input holding time [s]
weight = [1  1  1/2  1/6  1/6  1/6]; % weight in nearest search function
bias = 0.3; % biasing the algorithm towards the goal state [min 0, max 1]
num_inputs = 10; % number of control inputs
coef = 0.3; % coefficient for rand() function for k multipliers
r = [0.05,0.05,10*(2*pi/360),0.1,0.1,10*(2*pi/360)]; % admitted region ...
                    % around the goal state (r(1-3) = conf - r(4-6) = vel)

% post processing
% images
print = 0; % plot the tree at the end of the simulation
both = 0; % 1: plot both trees, 0: plot just the configuration tree
```

```
store = 0; % save the images --> 0 no, 1 .bmp, 2. png
picname1 = 'simulation/graph/conf'; % save the figure - configuration tree
picname2 = 'simulation/graph/vel'; % save the figure - velocity tree
dotfig = 1; % save the .fig image
if(store == 1)
    ext = 'bmp';
else
    ext = 'png';
end
% video
animation = 1; % final animation of the path
fps = 0.01; % seconds between frames in the animation (0 = "manual" animation)
record = 0; % record a video .avi
filename = 'simulation/video/animation.avi'; % name for the output

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RG-RRT Algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Tq,Tdq,A,Tu,R,u,tmp_U,tmp_evol] = Initialize(pos_i,vel_i,5,threshold,...
                                    num_inputs); % necessary initializations
solved = 0; % stop the algorithm if a solution is found
counter = 1; % current number of samples

tic;
while(~(solved) && counter<threshold)
   clc;
   if(rrt)
        disp(['RRT: ',num2str(counter),'/',num2str(threshold),' samples'])
   else
        disp(['RG-RRT: ',num2str(counter),'/',num2str(threshold),...
                ' samples'])
   end
    % 1. update reachable set
    if(~(rrt))
        R(:,counter) = ReachableSet(Tq,Tdq,counter,delta_t,wo,lo,mu,m,...
                                                            g,Io,coef);
    end
    % 2. random sampling
    accepted = 0;
    while(~(accepted))
        %%% 2.a sampling
        [q_rand,dq_rand] = RandomState(wspace_bounds,bias,pos_g,vel_g);
        %%% 2.b nearest state
        [dis,index] = NearestState([q_rand,dq_rand],Tq,Tdq,counter,...
                                                    weight,wspace_bounds);
        %%% 2.c reachable set test
        if(~(rrt))
            accepted = ReachableSetTest(R,[q_rand,dq_rand],index,dis,...
                                                    weight,wspace_bounds);
        else
            accepted = 1;
        end
    end
```

```
88      % 3. best input
89      %%% 3.a generate random inputs
90      for i=1:num_inputs
91         tmp_U(:,i) = RandomInput(coef,wo,lo,mu,Tq(3,counter));
92         tmp_evol(:,i) = Dynamics([Tq(:,index) Tdq(:,index)],...
93                                          tmp_U(:,i),delta_t,Io,m,g);
94      end
95      counter = counter+1;
96      %%% 3.b choose best input
97      [~,index_tmp] = NearestState([q_rand,dq_rand],tmp_evol(1:3,:),...
98                           tmp_evol(4:6,:),num_inputs,weight,wspace_bounds);
99      % 4. update trees
100     new_node = tmp_evol(:,index_tmp);
101     new_input = tmp_U(:,index_tmp);
102     [Tq,Tdq,A,Tu] = UpdateTrees(Tq,Tdq,A,Tu,new_node,index,...
103                                                  new_input,counter);
104     % 5. check for solutions
105     solved = Solution(Tq,Tdq,[pos_g vel_g],r,counter);
106 end
107
108 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
109 % Post processing and screen outputs
110 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111 clc;
112 toc;
113
114 [min_dist,min_dist_index,sequence] = PrintStat(solved,counter,Tq,...
115                              Tdq,[pos_g vel_g],r,A,weight,wspace_bounds);
116
117 if(print)
118     printRRT(A,Tq,pos_i,pos_g,counter,1)  % position tree
119     if(store)
120         saveas(gcf,picname1,ext)
121         if(dotfig)
122             saveas(gcf,picname1,'fig')
123         end
124     end
125     if(both)
126         printRRT(A,Tdq,vel_i,vel_g,counter,2) % velocity tree
127         if(store)
128             saveas(gcf,picname2,ext)
129             if(dotfig)
130                 saveas(gcf,picname2,'fig')
131             end
132         end
133     end
134 end
135
136 if(animation)
137     AnimateRRT(Tq,Tdq,sequence,[pos_g vel_g],wo,wm,lo,lm,...
138                         wspace_bounds,weight,fps,counter,filename,record);
139 end
```

Code for the friction cone evaluation in dynamic grasp mode:

```
function f = FrictionCone(wo,lo,mu,theta)
% FrictionCone computes the friction composite cone components in the world
% frame
% ---
% wo: object width [m]
% lo: object lenght [m]
% mu: coefficient of friction
% theta: current theta angle
% ---
% f: friction composite cone
% ---

    % semidiagonal length
    diag = sqrt(wo^2+lo^2);
    % angle between diagonal and one side of the object
    beta = atan2(wo,lo);
    f(1,:) = [-sin(atan(mu)+theta) cos(atan(mu)+theta) ...
                        -diag*cos(atan(mu)-beta)]; % left cone, left force
    f(2,:) = [sin(atan(mu)+theta) cos(atan(mu)+theta) ...
                       -diag*cos(atan(mu)+beta)]; % left cone, right force
    f(3,:) = [-sin(atan(mu)+theta) cos(atan(mu)+theta) ...
                        diag*cos(atan(mu)+beta)]; % right cone, left force
    f(4,:) = [sin(atan(mu)+theta) cos(atan(mu)+theta) ...
                       diag*cos(atan(mu)-beta)]; % right cone, right force

end
```

Input generation function in dynamic grasp mode:

```
function [w] = RandomInput(coef,wo,lo,mu,theta)
% Generate a random input for the RRT algorithm
% Contact mode: DYNAMIC GRASP
% ---
% coef: multiplier coefficients for rand() function
% wo, lo: object semi-width and semi-length
% mu: friction coefficient
% theta: current object orientation
% ---
% w: wx,wy,wt -> wrench in world frame
% ---

    w = zeros(1,3);
    k(1) = coef*rand(1);
    k(2) = coef*rand(1);
    k(3) = coef*rand(1);
    k(4) = coef*rand(1);
    f = FrictionCone(wo,lo,mu,theta);
    for i=1:3
        for j=1:4
            w(i) = w(i)+k(j)*f(j,i);
        end
    end

```

```
25 end
```

Nearest state search function:

```
1  function [dis,index] = NearestState(random_state,Tq,Tdq,counter,...
2                                                          weight,wspace_bounds)
3  % NearestState looks for the most suitable node to be extended
4  % ---
5  % random_state: random sample
6  % Tq,Tdq: trees
7  % counter: number of samples
8  % weight: weight vector for position and velocity distances
9  % wspace_bounds: workspace boundaries
10 % ---
11 % dis: mimimum distance value [m]
12 % index: index of the node in the tree
13 % ---
14
15     % ranges to normalize position along axises
16     range(1) = abs(wspace_bounds(2)-wspace_bounds(1));
17     range(2) = abs(wspace_bounds(4)-wspace_bounds(3));
18     range(3) = 2*pi;
19     range(4) = wspace_bounds(5);
20     range(5) = range(4);
21     range(6) = wspace_bounds(6);
22     distance = zeros(1,counter); % initialize distance vector
23     for i=1:counter
24         for j=1:3
25             % position distance
26             distance(i) = distance(i)+weight(j)*...
27                                 abs(Tq(j,i)-random_state(j))/range(j);
28             % velocity distance - normalized to vel limit and weighted
29             distance(i) = distance(i)+weight(j+3)*...
30                                 abs(Tdq(j,i)-random_state(j+3))/range(j+3);
31         end
32     end
33     [dis,index] = min(distance);
34
35 end
```

Bibliography

[1] M. T. Mason, *Mechanics of Robotic Manipulation*, A Bradford Book, August 2001

[2] K. M. Lynch, *Nonprehensile Robotic Manipulation: Controllability and Planning*, Ph.D. thesis, March 1996

[3] M. T. Mason, K. M. Lynch, *Dynamic manipulation*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 152–159, Yokohama, Japan, 1993

[4] B. Siciliano, O. Khatib, *Springer Handbook of Robotics*, Springer, 2008, ISBN: 978-3-540-23957-4

[5] B. Goodwine, J. Burdick, *Controllability of Kinematic Control Systems on Stratified Configuration Spaces*, IEEE Transactions on Automatic Control, 46(3), pp. 358–368, 2001

[6] R. Goebel, R. Sanfelice, A.R. Teel, *Hybrid dynamical systems. Robust stability and control for systems that combine continuous-time and discrete-time dynamics*, IEEE Control Systems Magazine, 2009 ,29(2), pp. 28–93

[7] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006

[8] B. Donald, P. Xavier, J. Canny, J. Reif, *Kinodynamic motion plannig*, Journal of the ACM, 40(5), pp. 1048–1066, November 1993

[9] M. S. Branicky, M. M. Curtis, J. A. Levine, S. B. Morgan, *RRTs for nonlinear, discrete, and hybrid planning and control*, IEEE Conference on Decision and Control, 2003

[10] S. M. LaValle, *Rapidly-exploring random trees: A new tool for path planning*, TR 98-11, Computer Science Dept., Iowa State University, October 1998

[11] S. M. LaValle, J.J. Kuffner, *Randomized Kinodynamic Planning*, In Proceedings of the 1999 IEEE International Conference on Robotics and Automation, Vol.1, Inst. of Electrical and Electronics Engineers, NewYork, 1999, pp. 473–479

[12] P. Cheng, *Sampling-based motion planning with differential constraints*, Ph.D. dissertation, University of Illinois, Urbana-Champaign, Urbana, IL, August 2005

[13] A. Shkolnik, M. Walter, R. Tedrake, *Reachability-Guided Sampling for Planning Under Differential Constraints*, In Proceedings of the IEEE/RAS International Conference on Robotics and Automation ICRA, 2009, Kobe, Japan, pp. 2859–2865

[14] J. R. Taylor, *Classical Mechanics*, Sausalito CA: University Science Books, 2004

[15] R. M. Murray, Z. Li, S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994

[16] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics: Modelling, Planning and Control*, 3rd Edition, Springer, 2009

[17] K. M. Lynch, M. T. Mason, *Dynamic underactuated nonprehensile manipulation*, 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems pp. 889-896, Osaka, Japan, November 1996

[18] K. M. Lynch, M. T. Mason, *Dynamic manipulation with a one joint robot*, 1997 IEEE International Conference on Robotics and Automation ICRA

[19] F. Bullo, M. Zefran, *Modeling and Controllability for a Class of Hybrid Mechanical Systems*, IEEE Transactions on Robotics and Automation, 18(4), pp. 563–573, 2002

[20] B. Goodwine, J. Burdick, *Motion Planning for Kinematic Stratified Systems with application to Quasi-Static Legged Locomotion and Finger Gaiting*, Proceedings of the Workshop Algorithmic Foundations of Robotics, 2000

[21] R. Goebel, J. Hespanha , A.R. Teel, C. Cai, R. Sanfelice, *Hybrid systems: Generalized solutions and robust stability*, IFAC Symposium on Nonliear Control Systems, 2004, pp. 1–12

[22] S. Karaman, E. Frazzoli, *Incremental sampling-based algorithms for optimal motion planning*, Int. Journal of Robotics Research, 30, pp. 846–894, June 2011

[23] E. Frazzoli, M. A. Dahleh, E. Ferron *Real-time motion planning for agile autonomous vehicles*, Journal of Guidance, Control, and Dynamics, 25(1), pp. 116–129, 2002

[24] P. Cheng, S. M. LaValle, *Reducing metric sensitivity in randomized trajectory design*, In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 43–48, 2001

[25] A. Yershova, S. M. LaValle, *Improving motion planning algorithms by efficient nearest-neighbor searching*, IEEE Transactions on Robotics, 23(1), pp. 151–157, February 2007

[26] A. Shkolnik, R. Tedrake, *Path planning in 1000+ dimensions using a task-space Voronoi bias*, In Proceedings of the IEEE/RAS International Conference on Robotics and Automation ICRA, 2009

[27] E. Glassman, R. Tedrake, *A quadratic regulator-based heuristic for rapidly exploring state space*, IEEE International Conference on Robotics and Automation ICRA, 2010, pp. 5021–5028

[28] A. Yershova, L. Jaillet, T. Siméon, S. LaValle, *Dynamic-Domain RRTs: efficient exploration by controlling the sampling domain*, In Proceedings of the IEEE/RAS International Conference on Robotics and Automation ICRA, pp. 3856–3861, 2005

[29] A. Atramentov, S. LaValle, *Efficient nearest neighbor searching for motion planning*, In Proceedings of the IEEE/RAS International Conference on Robotics and Automation ICRA, pp. 632–637, 2002

[30] S. R. Lindemann, S. LaValle, *Incrementally reducing dispersion by increasing Voronoi bias in RRTs*, In Proceeding IEEE International Conference on Robotics and Automation, 2004

[31] Y. Li, K. E. Bekris, *Learning approximate cost-to-go metrics to improve sampling-based motion planning*, In IEEE International Conference on Robotics and Automation ICRA, Shanghai, China, 2011

[32] P. Cheng, *Sampling-based motion planning with differential constraints*, Ph.D. dissertation, University of Illinois, Urbana-Champaign, Urbana, IL, August 2005

[33] C. G. Cassandras, S. Lafortune, *Introduction to Discrete Event Systems*, Springer, Second Edition, 2008

[34] K. Lynch, M. T. Mason, *Dynamic nonprehensile manipulation: Controllability, planning, and experiments*, The International Journal of Robotics Research, 1999, 18(1), pp. 64–92

[35] T.D. Murphey, K. Lynch, *Case Studies in Planar Part Feeding and Assembly Based on Design of Limit Sets*, The International Journal of Robotics Research, 2008, 27(6), pp. 693–708, SAGE Publications

[36] S. Akella, M. T. Mason, *Posing polygonal objects in the plane by pushing*, The International Journal of Robotics Research, 1998, 17(1), pp. 70–88, SAGE Publications

[37] O. Ben-Shahar, E. Rivlin, *Practical pushing planning for rearrangement tasks*, IEEE Transactions on Robotics, 14(4), pp. 549–565

[38] J. D. Bernheisel, K. M. Lynch, *Stable transport of assemblies: pushing stacked parts*, IEEE Transactions on Automation Science and Engineering, 2004, 1(2), pp. 163–168

[39] B. Hove, J.-J. E. Slotine, *Experiments in robotic catching*, American Control Conference, pp. 380–385, 1991

[40] Y. Imai, A. Namiki, K. Hashimoto, M. Ishikawa, *Dynamic active catching using a high-speed multifingered hand and a high-speed vision system*, In Proceeding IEEE International Conference on Robotics and Automation, 2004, pp. 1849–1854

[41] J.-C. Ryu, F. Ruggiero, K. M. Lynch, *Control of Nonprehensile Rolling Manipulation: Balancing a Disk on a Disk*, In Proceeding IEEE International Conference on Robotics and Automation, 2012, pp. 3232–3237

[42] K. M. Lynch, N. Shiroma, H. Arai, K. Tanie, *The roles of shape and motion in dynamic manipulation: The butterfly example*, In Proceeding IEEE International Conference on Robotics and Automation, 1998, pp. 1958–1963

[43] Z. Li, J. Canny, *Motion of Two Rigid Bodies with Rolling Constraint*, In Proceeding IEEE Transactions on Robotics and Automation, 6(1), 1990, pp. 62–72

[44] , T. Higuchi, *Application of electromagnetic impulsive force to precise positioning tools in robot systems*, International Symposium on Robotics Research, pp. 281–285, 1985

[45] W. H. Huang, M. T. Mason, *Mechanics, planning, and control for tapping*, The International Journal of Robotics Research, 19(10), 2000, pp. 883–894, SAGE Publications

[46] D. Reznik, J. Canny, K. Goldberg, *Analysis of part motion on a longitudinally vibrating plate*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 421–427, 1997

[47] T. H. Vose, P. Umbanhowar, K. M. Lynch, *Friction-induced velocity fields for parts sliding on a rigid oscillated plate*, The International Journal of Robotics Research, 2009, 28(8), pp. 1020–1039

[48] J. C. Trinkle, J. J. Hunter, *A framework for planning dexterous manipulation*, IEEE International Conference on Robotics and Automation, pp. 245–251, 1991

[49] M. A. Erdmann, *An exploration of nonprehensile two-palm manipulation*, The International Journal of Robotics Research, 17(5), 1998, pp. 485–503

[50] M. Yashima, H. Yamaguchi, *Dynamic motion planning whole arm grasp systems based on switching contact modes*, IEEE International Conference on Robotics and Automation, pp. 245–251, 2002

[51] K. Miyazawa, Y. Maeda, T. Arai, *Planning of graspless manipulation based on rapidly-exploring random trees*, International Symposium on Assembly and Task Planning, 2005

[52] Y. Maeda, H. Kijimoto, Y. Aiyama, T. Arai, *Planning of graspless manipulation by multiple robot fingers*, IEEE International Conference on Robotics and Automation, 2001

[53] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, S. Srinivasa, *CHOMP: Covariant Hamiltonian Optimization and Motion Planning*, International Journal of Robotics Research, accepted for publication in April 2013

Acknowledgments

During the last six years, three of Bachelor's and two plus one of Master's, I have met so many people I will forget somebody for sure. It is a tough task to thank everyone who helped me during these years, but I am sure I would not be the person I am, if I did not meet and share something with everyone I met during my academic studies. Thanks to everyone.

First of all, I would like to thank both my advisors, Professor Giuseppe Oriolo and Professor Kevin Lynch, for the amazing opportunities they have given to me to realize my "American dream", to experience life in a dynamic and very gratifying environment like the Northwestern University, surrounded by amazing people who really helped me in growing as a student, as a scientist, and above all as a man. And, of course, my university La Sapienza has not disfigured at all in the comparison with one of the most famous and important universities in the world. La Sapienza gave me all the tools I need to become a good engineer and it is still in my opinion the best environment where growing up.

I want to thank all the people I met in Evanston, starting from my dear French friend Michael and my dear Taiwanese friend Yu-Wei, which helped me to create a small "foreign" corner in the NxR lab! How to forget about Professor Ji-Chul! He was one of the most helpful and inspiring person I met here at the Northwestern. I wish you the best in your career. Adam, I guess you still have nightmares of me walking towards you and asking for the craziest questions you have ever heard of... thanks for everything. We will submit that paper eventually! Paul you helped us way more than you can think. Your passion and your attention during the weekly meeting were an example for me. Thanks! Nick, Matt, Craig, Jian, Nelson, Steven, Alex, Jarvis, Joe, Lauren, Lucie, Vlad, David, and Eric: thanks guys! It was fun to meet you, "drink beer" on Friday happy hours, chat about sport or nerd things at lunch time, hang out at night. I will always remember these days and I hope to see you soon in my home town!

E ora passiamo in italiano che mi trovo sempre più a mio agio... nonostante un anno avanti a hot-dog e burgers. Questo sicuramente non mi

mancherà mai!

Voglio ringraziare i miei amici conosciuti durante questi anni universitari. Daniele "Zozzetta" e Daniele "Caricatooore" e le mangiate a Ceccano, PornoPaolo, Diego, Nick e le partite a minigolf mentre Paolo smadonnava al pc, Andrea e tutte le risate che ci siamo fatti tra i banchi (il DOC! e il taglialegna dal Canada!), il mitico "Trigger" Salvatore, Rob, Decio, Jessica, Raffo, Valerio, Caterina, Chiara, Riccardo, Maurizio, Iovel, Giovanni, Emilio, Mauro, Daniele, Peppe e tutta la combriccola di amici informatici (sicuramente mi sto dimenticando di qualcuno, ma voi siete informatici, chissene...!), ma soprattutto la briscola a cinque che ci ha deliziato per anni e ci ha aiutato a stemperare l'ansia pre-esame! Un saluto anche ai primissimi amici di ingegneria aerospaziale... chissà che sarebbe successo se non avessi mai cambiato corso di laurea!

Come non citare Jacopo e Don Paoli per la fantastica vacanza durante questi mesi qui a Evanston. E Goro, Savigio e l'Uturniolo per il supporto a distanza! Dario, Ada e Moro che ancora stanno aspettando il loro batterista di fiducia... Bode e Vinz, amici unici, che di magheggi al fantacalcio sono i maestri, meriterebbero una laurea.

E poi ovviamente devo ringraziare tutte le persone conosciute extrauniversità qui ad Evanston, a partire da Riccardo, che mi ha accompagnato per tutto l'anno nella fredda Chicago. Un vero amico! Padre Luca...ho ancora delle bottiglie di Vitamin Water sul mio scaffale solo in memoria tua! Antonella: che dire?! Senza di te sarebbe stato ovviamente tutto molto più deprimente qui ad Evanston. Hai acceso la luce in tutti noi, l'hai accesa in me! Cinzia, Valentina, Corinna e Andrea, Chiara, Juanito, Alex, Luiza, Benedicte, Amod, Atsuro, Guenael, Max e Christie, Antonio: che spettacolo avervi conosciuto. Non dimenticherò mai nessuno e sono sicuro ci rincontreremo prima o poi!

Però permettetemi di dire, con molto orgoglio, che le persone che più devo ringraziare sono la mia famiglia. Mia sorella Maria Chiara, il mio babbo Stefano e la mia mamma Rita. Senza di voi nulla, ma nel vero senso della parola nulla, sarebbe potuto succedere. Non vi ringrazierò mai abbastanza e spero solo di avervi reso oggi un pochino più orgogliosi del vostro figlio maschio preferito e del tuo fratellino perferito!

Infine, sarà una cosa matta, ma non mi frega niente, un ultimo pensiero ad un amico vero di lunghissima data che è venuto a mancare durante questo periodo e che purtroppo nessuno mi ripoterà mai indietro. Ciao piccolo Buck. Ti porterò sempre nel mio cuore!

Grazie a tutti!

Roma, 23 Ottobre 2013