Robot Motion Planning with Contact from Global Pseudo-inverse Map

by

Changrak Choi

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

Author .... Signature redacted
Department of Mechanical Engineering
July 6, 2018

Certified by.............................. Signature redacted
Emilio Frazzoli
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by............................ Signature redacted
Sangbae Kim
Associate Professor of Mechanical Engineering
Chair, Thesis Committee

Accepted by........................... Signature redacted
Rohan Abeyaratne
Chairman, Department Committee on Graduate Theses

Robot Motion Planning with Contact from Global Pseudo-inverse Map

by

Changrak Choi

Submitted to the Department of Mechanical Engineering
on July 6, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In the robot motion planning problems, environment and its objects are often treated as obstacles to be avoided. However, there are situations where contacting with the environment is not costly. Moreover, in many cases, making contact can actually help a robot to maneuver around to reach a goal state which would not have been possible otherwise.

This thesis presents a framework for motion planner that utilizes multiple contacts with the environment and its objects. The planner is targeted to autonomously generate motion, where robot has to make multiple contact with different part of its body in order to achieve a task objective. It is motivated by and has significance in developing a robust humanoid planner that is capable of recovering from a fall down. The recent DRC has been marked with compilation of humanoid robots falling down, but only one robot managed to recover to a standing up position. In a real disaster scenario, the inability to stand up would mean end of the rescue mission for what is extremely expensive machinery. A robust planner capable of recovery is must and this work contributes towards it. The developed planner autonomously generates standing up motion from fall down in the presence of torque limits.

The proposed multi-contact motion planner leverages upon following two key components. Existing multi-contact planners require good initial seeds to successfully generate a motion. These are hard to find and often manually encoded. Here, we utilize pre-computed global pseudo-inverse map (inverse kinematic map for each contact-state that has property of global resolution, connected by connectivity functions) to generate multi-contact motion from current configuration to the goal without need for an initial seed. Nevertheless, constructing the global pseudo-inverse map is computationally expensive. In an effort to facilitate the construction, we utilize singular configurations as a heuristic to reduce the search space and justify its use based on the physical analysis. Although computationally expensive, once pre-computed, the global map can be used to generate plans fast online in a multi-query manner.

Thesis Supervisor: Emilio Frazzoli
Title: Professor of Aeronautics and Astronautics

Acknowledgments

To begin with, I would like to thank my supervisor, Professor Emilio Frazzoli, for his valuable and kind advice, support, guidance and patience during my PhD. Not only has he given me academic advices regarding the work, but also provided kind supports and encouragements to which I am truly grateful.

I wish to thank my lab mates, especially Valerio Varricchio, Brian Paden, Prince Singh, and Sze Zheng Yong for all the help and support they have given me in and out of the lab. In addition, I would like to express special thank to Prof. Kris Hauser for the kind advices and meetings over the time I was visiting at Duke.

As a member of KGSAME, I would like to thank all its members, my wonderful friends and colleagues at MechE whose interaction was a great joy during my graduate years. Also, I thank Kwanjeong Educational Foundation for supporting me with the scholarship during the PhD program.

Finally, my deepest gratitude goes to my family for their endless love and support throughout my life; this thesis would not have been possible without them. Especially, words cannot describe all the support and happiness that my wife, Seoha Min, has given me during the years of PhD. She is a recognized scholar in the field of apparel design whom I deeply respect, both as a person and professional. I owe this Thesis to her.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Motion planning has traditionally focused on generating trajectory that views environment as obstacles. This is largely because the motivation for motion planning problems derived from situations where collision with the environment is highly undesirable. For example, an assembly robot in a car manufacturing factory must plan a motion through a rather complex car body without hitting it. Failure to do so will result in damage to both the car and the manipulator. A quadrotor navigating inside a crowded building must plan a motion that avoids contacting with the walls and other native objects. If a quadrotor hits an object, it can lose stability and in the worst case scenario, fall to the ground and break down. An autonomous car must plan a motion that avoids collision at all cost. If not, accidents can happen which may hurt riders inside.

However, there are situations where contacting with the environment is not costly. Moreover, in many cases, contact can actually help a system to maneuver around to reach a goal state which would not have been possible otherwise. For example, an astronaut inside the international space station often needs to move from one compartment to another through narrow corridors. Here, planning a motion to avoid any contact whatsoever is not only difficult but unnecessary - in contrary, astronauts actively seek to make contact and push off with his or her legs and arms to navigate through.

In fact, I would argue that for a general robotic system without thrusters, contacting with the environment is the only means of navigating through. Newton writes in his book Principia, *"Every body under the sole action of its innate force moves uniformly in a straight line indefinitely unless something extraneous hinders it* [60]." This law of motion is universal and a robot is no exception. Motion of a robotic system - generalized velocity of its center of mass - can only be changed through external forces acting on the system. Here lies the fundamental limitation of a robot. All the joint torques which it controls are internal forces, hence the robot has no ability to directly generate external forces needed to change its motion. For example, a humanoid floating in the space will forever drift away with its initial velocity no matter how hard it actuates its joint torques. That is, until the humanoid comes into contact with an external object. A robotic system has control authority over its internal joints torque which can only be translated to external forces when the robot makes a contact. Environment, in the eyes of a robot, should then be viewed as a source of external force generators to be utilized through contact.

Hence the importance of planning a motion that incorporates contact with environment and utilizes forces from it. Motion Planning with Contact as will be referred henceforth is the work presented in this thesis. In this work, I explore how to plan a motion for a dynamical system that takes advantage of the contact forces and develop planning algorithm framework that can efficiently handle contacts. The dynamic system of interest is primarily a general n-linkage robot (e.g. bipedal robot, quadruple runner, or multi-armed climber) with no restriction placed on its form.

1.2 Problem Formulation

Motion Planning with Contact Problem

Consider a dynamical control system S with hybrid dynamics given by

$$\frac{dx}{dt} = f_i(x, u)$$

where x is the state, u is the control, i is the mode.

Given

- State Space X
- Obstacle Set $X_{obs} \subset X$
- Initial State $x(0) = x_{init}$, and a Goal Set $X_{goal} \subset X$
- Constraints on control input $u_{\min} \leq u \leq u_{\max}$

Automatically find, if it exists, a control input $u(t)$ such that the system S

- Satisfies hybrid dynamics and input constraint for all time
- Reaches the goal state, i.e., $x(t_f) \in X_{goal}$
- Avoids penetrating the obstacles, i.e., $x(t) \notin X^o_{obs}$ for $^\forall t \in [0, t_f]$
- Switches mode only when in contact with the obstacle, i.e., $x \in \overline{X}_{obs} \backslash X^o_{obs}$ with impulsive change given by impact map $x^+ = \Delta(x^-)$

Return failure if no such control signal exists

The dynamical control system S of interest in this work is a generalized n-linkage robot. The state $x = [q, \dot{q}] \in R^{2n}$, where $q \in R^n$ is joint angles and control $u \in R^m$ is torque applied to each active joints. The system S may be a bipedal robot, a quadruple runner or multi-armed climber - no restriction is placed on its form.

The dynamics of the system S is hybrid in nature, because each contact with the environment results in a change in its dynamics. To be more precise on when the

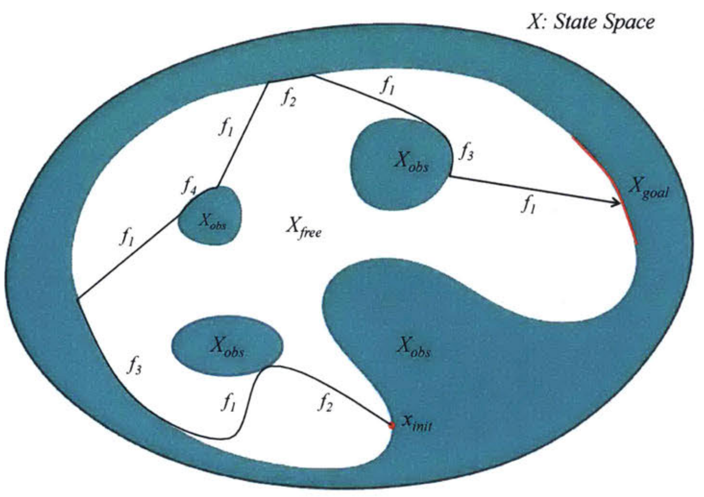


Figure 1-1: Given system S with dynamics $\dot{x} = f_i(x, u)$, obstacle O, State Space X, and $x_{init}, X_{goal} \subset X \backslash X^0$, Automatically compute $u(t)$ s.t. $u(0) = x_{init}$ and $u(t_f) \in X_{goal}$. The system can switch its mode i when in contact with the obstacles but is not allowed to penetrate the obstacles.

switching in mode and impulsive change takes place, let $A(x) \subset R^3$ be set of points in the physical world occupied by the system S at a state x. Let $O \subset R^3$ be set of points occupied by the obstacles. Let $c(x)$ be the cardinality of set $A(x) \cap O$, which equals the number of points in contact with the obstacles at a state x. Then mode switches when $c(x^+) \neq c(x^-)$ and impulsive change defined by the impact map occurs when $c(x^+) > c(x^-)$.

The tasks of interest which the system S must perform are those that requires multiple contact with the environment. Ideally, the system S would be operating near or at control limits in its motion when performing the tasks that makes forces from contact inevitable. Nevertheless, the planning algorithm should not be task-specific and work for general tasks.

The planning algorithm would preferably be an anytime in nature, where initial solution is returned quickly and gradually improved upon over time. Computational efficiency of the algorithm play a key role in whether it can be planned online or restricted to pre-planning a feed-forward motion. Ideally, it should target for the former.

1.3 Motivational Examples of Interest

1.3.1 Recovery from Fall Down

Figure 1-2: Darpa Robotics Challenge had numerous instance of robots falling down. Recovering from the fall down is a non-trivial task that requires making multiple contact with different part of the robot with the environment(ground). Image retrieved from https://spectrum.ieee.org/ automaton/robotics/humanoids.

Fixed-base manipulators (e.g. industrial robots) have fixed contact point at the base and moving contact point at the end-effector which cannot be changed. On the other hand, floating-base robots (e.g. humanoids) do not have a fixed contact point, but contacts are generally restricted to feet or hands. But for many motions, contacting with environment on points besides feet and hand are desirable if not imperative.

An great example of this is recovery motion from fall down position. A humanoid robot fallen down may only be able to recover using several contact points in addition to feet and hand if torque isn't sufficiently large (and in fact, the torques are very limited compare to the weights of these robots). In fact, the recent DRC has been marked with compilation of humanoid robots falling down, but only one robot managed to recover to a standing up position. In a real disaster scenario, the inability to stand up would mean end of the rescue mission for what is extremely expensive machinery. A robust planner capable of recovery is must and this work contributes towards it.

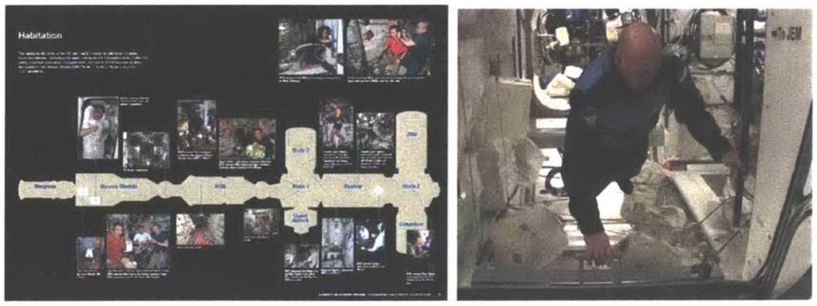


Figure 1-3: Astronauts need to constantly navigate through narrow and crowded modules inside the International Space Station as shown on the left. Rather than seeking to avoid the obstacles, astronauts actively make contact with the environment objects to help them navigate as seen in the right. Image courtesy of NASA retrieved from https://www.nasa.gov.

1.3.2 Navigating inside the International Space Station

An astronaut inside the International Space Station (ISS) need to constantly move from one module to another. Figure 1-3 shows different modules inside the ISS and how accomodations - sleep compartment, toilet, exercise facilities and many more - are widely spread out throughout the ISS. These modules are crowded with equipments and connected by narrow corridors.

Navigating through the ISS in a collision-free manner is very hard, if not impossible. Even if an astronaut is wearing a spacesuit with full-axis thrusters, this is an daunting task given the crowdedness and narrowness of the ISS. However, astronauts navigate around the ISS with ease without even needing to turn on the thrusters by making use of the contacts. Rather than trying to avoid environment objects, astronauts make contact with them and utilizes the reaction forces to help pass through narrow corridors.

For astronauts, an environment object are not an adversary to be avoided but an ally to be utilized. In fact, it is because the ISS is crowded and narrow that astronauts can easily navigate around through constantly making contacts. An astronaut navigating inside the ISS motivates following problems which are abstraction of the what has been described.

First problem is navigation of 2-link robot inside the ISS as shown in Figure 1-4.

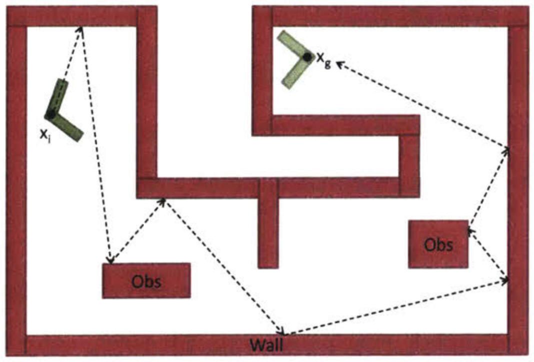


Figure 1-4: Navigation of 2-Link Robot inside the ISS Problem. A two link robot has to travel from x_i to x_g and it can only control its joint torque. The robot can change its velocity while in contact by applying appropriate joint torque. Nevertheless, it has no ability to change its velocity in flight.

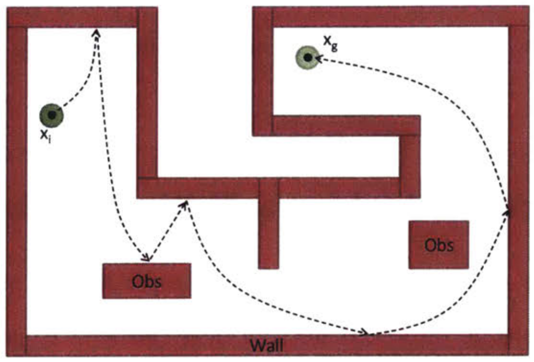


Figure 1-5: Navigation of a Point Mass with one-axis thruster inside the ISS Problem. A point mass must reach x_g starting from x_i and it can accelerate only in one direction. The point mass bounces off the contact such that the angle of incidence equals that of reflection. While in flight, it can alter velocity of one axis by controlling its thruster.

An astronaut is simplified as a 2-link robot and torque on the joint acts as the control $u(t)$. The 2-link robot(astronaut) starts at a initial state $x_i = (q_i, \dot{q}_i), \dot{q}_i \neq 0$ where q is the position of the center of mass. Finding the control input $u(t)$ that will steer the robot towards the goal state x_g is the problem of interest. The robot can change its velocity while in contact by applying appropriate joint torque. However, it has no ability to change its velocity off the contact and moves in a straight line till another contact is made.

Second problem is navigation of a single point mass with one-axis thruster inside the ISS as depicted in Figure 1-5. Here, an astronaut in a spacesuit is modeled as a single point mass with thruster only in one axis. The single point mass cannot alter the bouncing off angle as the reflecting angle must equal incident angle. But while in flight, it can alter velocity in one of the axis by controlling the thruster. Finding the thruster control input $u(t)$ that steers the point mass from x_i to x_g is the problem of interest.

These problems are interesting since both system are inherently not controllable. If both system were equipped with a two-axis thruster, they could have easily navigated around without contacting the obstacles as the controllability matrix is full rank. However, despite not being fully controllable, both system can make use of contact to induce velocity in the direction of controllable null space and act as if it is fully controllable. In essence, through contacting with obstacles, the system is able to reach places that would not have been possible by avoiding obstacles.

1.3.3 Dynamically Climbing a Vertical Surface

Suppose there is one meter wall standing in the path that one needs to go. Let's imagine how one would go on about getting over the wall to continue his way. If the person happens to be an athlete (e.g. basketball player), he would simply leap over in a single jump without ever needing to use hands. This is hard for non-athletes, but a man who is strong enough would put his hand on top of the wall and push off to leap over without needing to anchor his feet on the side of the wall. However, in most cases, a person would need to use all parts of the body (hands, feet, forearm, and so on) and make sequence of contacts both at the side and top surface of the wall to climb over. In few occasions, if the person is very tired, he may not have strength to climb despite making proper contacts and has no choice but to go around the wall.

There are two observation that one can gain from this thought experiment. First observation is that a person's strength(maximum torque one can produce at each joints to be more precise) acts as the limiting factor that determines resulting motion when performing an aggressive maneuvers. For example, an athlete was able to leap over in a single jump which an ordinary man could not have done and this comes

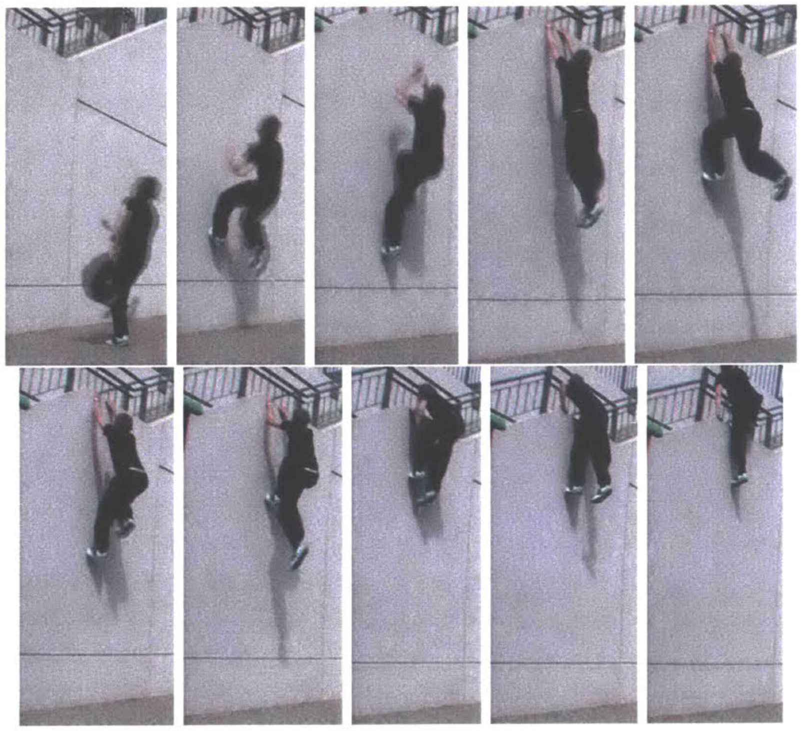


Figure 1-6: Sequence of motions performed by a Parkour practitioner in climbing up a vertical wall of significant height. Vertically jumping over one meter wall is hard. Leaping over a wall higher than one's height is impossible even for strong athletes. However, by running up and making sequence of right contacts, Parkour practitioners are able to climb over a wall that is nearly twice their height. Image retrieved from https://youtu.be/08x3P2E122U.

from the difference in the maximum joint torque between the two.

Second observation, which is of greater interest, is that making contacts with the obstacles(wall in this case) helps one to reach places that would otherwise have not been possible. For example, an ordinary man jumps about 0.5 meter on average. However, he is able to go over the one meter wall easily by making contacts. Similarly, jumping more than one's height is near impossible even for an athlete. But by making sequence of right contacts as shown in Figure 1-6, a Parkour practitioner is able to go over a wall of nearly twice his height.

This motivates following problems as illustrated in Figure 1-7. A general multi-linkage robot(e.g. humanoid) is given as the system S. The vertical wall and ground acts as the obstacles set X_{obs} and starting from x_i on the ground, the robot needs

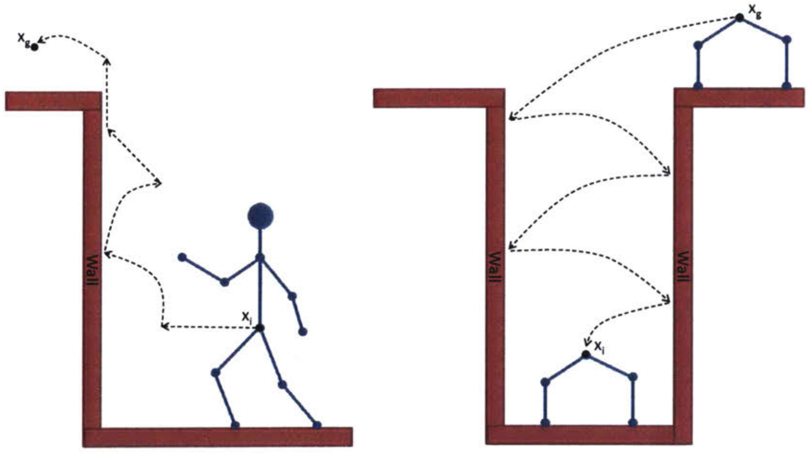


Figure 1-7: Dynamic Climbing of a Vertical Surface Problem. A general multi-linkage robot (e.g. humanoid or quadruple runner) must climb over a wall that is higher than what it can leap over in a single jump. Making appropriate sequence of contacts is crucial and resulting motion will depend heavily on the available torque limits.

reach x_g located over the vertical wall. In this problem, control input $u(t)$ is the input torque applied at each of the joints and each joint has a limit on the max torque that can be applied such that $u(t) \leq u_{max}$ is satisfied for all time. Here, the height of a vertical wall is set higher than what the robot can leap over in a single jump. The problem of interest is automatically finding $u(t)$ that would steer the robot from x_i to x_g.

This problem of dynamically climbing over a vertical surface is interesting for two main reasons. First, the resulting motion has to be aggressive and dynamic in nature. Because the gravity is consistently acting to pull down the robot to the ground, the motion must be performed quickly while making use of the proper contacts with the obstacle. Second, the resulting motion will differ greatly depending on the limits on the max joint torques. Since the motion needs to be as quick as possible, the robot will generate motion that operates near or at the torque limits. Notice this is far different from current robot locomotions which are very conservative and operates well within its torque limits.

Following are real life motions that are well documented in the online video resources which capture the essence of what is described in this problem. A Parkour practitioner often climbs over a high wall that is over the his own height by making sequence of steps and contacts with the wall and its structures. A man climbs out of the deep well by making sequence of contacts along the vertical wall using his arm and legs. A mountain goat safely climbs down between two stiff cliffs by alternatively contacting between two cliffs to reduce its fall down speed.

1.3.4 Quasi-Static Rock Climbing (Bouldering)

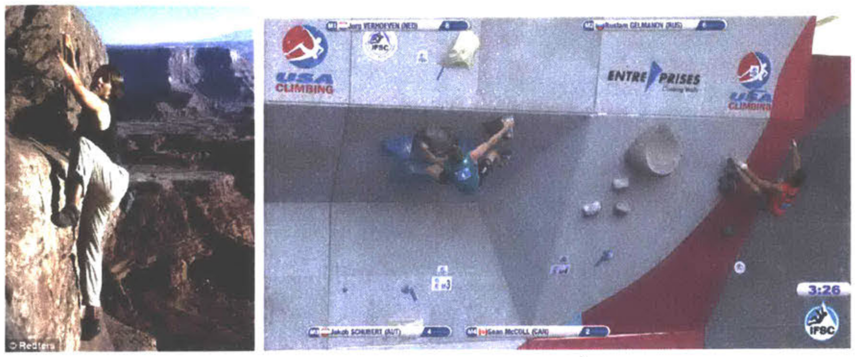


Figure 1-8: Bouldering is a form of rock climbing that does not rely on the ropes. Tom Cruise performed bouldering in his famous opening sequence in the movie Mission Impossible 2 as shown on the left. Professional climbers compete on a very challenging course in the Boulder World Cup as shown on the right. Image retrieved from http://www.ifsc-climbing.org/

In bouldering, one climbs a rock or a mountain without the help of ropes or harnesses. As a consequence, securing a stable foothold and grabbing onto a right place becomes very important. This is well shown in the opening scene of the movie Mission Impossible 2 where Tom Cruise climbs a rocky mountain without relying on a rope as shown in Figure 1-8. If there are only few spots that permit stable foothold, bouldering becomes extremely difficult task. This is demonstrated in the annual World Cup help by International Federation of Sport Climbing, where even the world's best climbers have hard time completing the given course.

This problem differs from the dynamic climbing of a vertical surface discussed previously, as it is quasi-static motion where one transitions from one stable grasp configuration to another. It is interesting on its own as literally every part of the body can and must be used as potential contact points to complete a given task.

1.4 Literature and Background

Figure 1-9 shows where Motion Planning with Contact problem fits in literature with respect to different branches of work. In this section, it is described viewing from motion planning, legged robot locomotion and computer animation perspectives. Discussion is given as to how Motion Planning with Contact problem differs from and extends these fields.

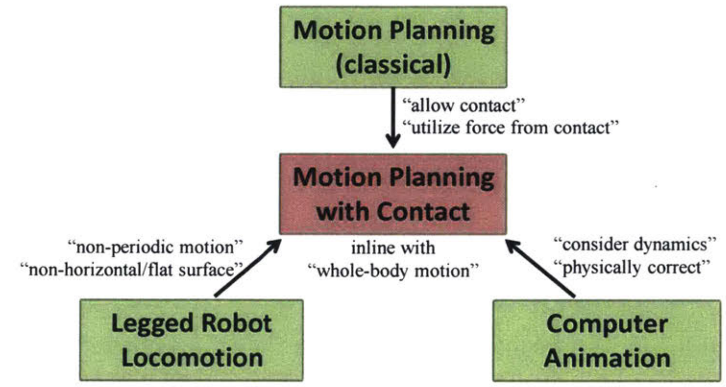
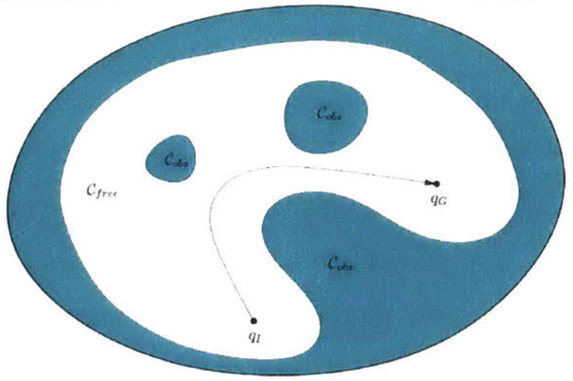


Figure 1-9: The diagram shows where the Motion Planning with Contact Problem fits in the existing literature. From motion planning perspective, this work challenges the basic assumption of avoiding obstacles at all cost. By allowing contacts with the obstacles, the planned motion will utilize forces from the contacts to its advantage. From legged robot locomotion perspective, existing literature focus mostly on periodic motion on a horizontal surface (e.g. walking and running). What is pursued in this work is s motion that is non-periodic and not along a horizontal or flat surfaces. From computer animation perspective, methodologies exists to automatically generate animated motion of a virtual character similar to how human moves. These motions, although looks real, are physically incorrect and this work will generate motions that are dynamically consistent.

1.4.1 From Motion Planning perspective

The basic motion planning problem deals with finding a path or trajectory that starts from a given initial state to a desired goal state while avoiding obstacles. Figure 1-10 illustrates this.

Given robot $\mathcal{A}$ and obstacle $\mathcal{O}$ models, C-space $\mathcal{C}$, and $q_I, q_G \in \mathcal{C}_{free}$.



Automatically compute a path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$ so that $\tau(0) = q_I$ and $\tau(1) = q_G$.

Figure 1-10: Basic motion planning problem is described [44]. Notice how it differs from the motion planning with contact problem as shown in Figure 1-1.

In terms of representation, an important notion in the motion planning literature is the idea of configuration space. Lozano-Perez introduced the idea of configuration space [49] which effectively transforms a motion planning problem into a path planning problem. How to geometrically represent obstacles and compute C_{obs} is an important question and what metric is to be used also play a crucial role.

In terms of computational complexity, motion planning problem even in its basic form is not trivial. Reif proved that the generalized piano mover's problem is PSPACE-hard [75] and it was later shown to be in fact PSPACE-complete by Canny [9]. Hence, as the number of degree of freedom of a robot increases, the motion planning problem inherently becomes challenging.

For low dimensional problems(usually less than $n \leq 3$), combinatorial methods are viable and several algorithms exists that are complete. Trapezoidal decomposition [11], maximum-clearance roadmap [62], and reduced visibility graph [42] are among the popular methods that solves the motion planning problem by building a roadmap through decomposition of the configuration space. Combinatorial methods, however, either do not extend or becomes impractical in the higher dimensions ($n > 3$). Cylindrical algebraic decomposition extends but is shown to be doubly exponential time and space [76]. Canny's roadmap algorithm [10] is singly exponential in time but is too complex to be implemented.

For higher dimensional problems ($n \geq 3$), practical algorithms are mostly sampling based. Two of the most well-known sampling-based algorithms are PRM(Probabilistic Roadmap) introduced by Kavraki and Latombe [35], and RRT(Randomized Rapidly-exploring Tree) by LaValle [43]. These algorithms work well for the motion planning problems in the higher dimensions and many variants exists that improves upon the original algorithm. The most notable among the variants is RRT* algorithm by Karaman and Frazzoli [34], which is the first algorithm to guarantee optimality of the returned solution.

In all of these works, motion planning literature starts from the basic assumption that environment are obstacles to be avoided. The basic motion planning problem and its variants focus explicitly on finding *collision-free* motion from start to goal. This is because the motivational examples from which the motion planning problems came about required that collision be avoided at all cost. Robot manipulator performing assembly should not hit the car body nor other manipulators, a quadrotor navigating through indoor environment should not hit with wall nor people, and an autonomous car should never come into collision with any other object in all times.

This work challenges the basic assumption underlying works in motion planning literature of viewing environment as a must avoid. In many examples, as illustrated in the previous section, contacting the environment is not costly and may in fact be greatly beneficial. This work will extend the motion planning literature by allowing contacts with the environment in the problem formulation such that the resulting motion utilizes the reaction forces from the contacts to its advantage.

1.4.2 From Legged Robot Locomotion perspective

Planning a motion that incorporates contact naturally arises from problems of legged locomotion where the foot makes contact with the ground [87]. Footstep planning is essential for this type of locomotion and number of literature addresses it [12,16,39,71]. More recently, planning for the whole-body motion that incorporates multi-contact interactions has gained a great interest and is actively being pursued.

Approaches based on operational space formulation [36,70,79,80] and centroidal momentum dynamics [26,46,63,64,86] is gaining traction, while many leverage upon the power of optimization techniques [5,14,47,72]. However, these works are focused on generating a quasi-static motion where at least one contact with the surface, usually a horizontal ground, is maintained.

In my PhD thesis, I aim to develop planning methodology that is suitable for motions where flight-phase is dominant without being periodic (cf. a robot running [69] or hopping [74] is dominated by flight-phase but periodic in nature). An example would be an astronaut navigating inside the space station or a parkour practitioner jumping between walls to maneuver around the buildings. These motion are characterized by the fact that the robot has no ability to change its centroidal momentum during flight. Only when in contact, the internal joint torques of the robot can be translated to external forces affecting the motion, which is then propagated through the flight-phase till the next contact is reached. Here, obstacles must be utilized rather than avoided, viewed as an opportunity to change the course of the motion through contact.

1.5 Approaches and Directions

Planning is a search in a state-space. Success of a planning algorithm depends on how efficiently it can perform search. This is especially true for motion planner that deals with contact as pursued in this work. The robot itself has high degree of freedom and each contact introduced acts as a branching factor that can quickly grow exponentially. Hence, dimension of the state-space over which a search has to be performed is substantially high.

The first and extremely important question to answer will be defining over what space the search will be performed. The natural choice from motion planning perspective will be configuration space which enables one to capture the kinematics and reduce the problem to that of a path planning. However, in the presence of dynamics and actuation limits, planning in configuration space often leads to a path that is not dynamically feasible. On the other hand, planning in state space may be an alternative, which can guarantee dynamical feasibility by directly satisfying limits imposed on the joint torques. Nevertheless, dimension of state space is significantly higher and grows with robot's degree of freedom despite many joints being redundant in nature.

In place of the configuration and state space, an approach is taken to perform the search over the work space of contacting points. The dimension of the work space remains constant irrespective of the growth in a robot's degree of the freedom. So the size of the search space is far smaller and does not suffer from the curse of dimensionality.

Nevertheless, searching in work space requires one to perform inverse kinematics at every instance of a sampled point to check for feasibility of the configuration. This is computationally heavy operation and makes the search extremely slow. In view of this, an approach is taken to build up a global pseudo-inverse map that relates a point in the work space to its corresponding points in the configuration space. Building such a map is performed offline where computational time is not restricted. Then the search utilizes the map to quickly generate a plan online without needing to perform inverse kinematics along the way.

The complexity of motion planning with contact problem is PSPACE-hard as the motion planning problem which is a subset has PSACE-complete complexity. Hence, finding an optimal solution is computationally intractable. Instead, an approach is taken to efficiently find approximate solution near optimal using heuristics. Here, singular configurations are chosen as the heuristic based on the physical understanding to guide the search.

1.6 Statement of Contribution

The main contribution of the thesis is in developing a framework for a motion planner that incorporates multiple contact with the environment. The essence of the framework is a fast multi-contact planner that utilizes pre-computed global pseudo-inverse map. Existing multi-contact planner require good initial seeds for it to successfully generate motion. These are hard to find and often manually encoded. In addition, current multi-contact planners are single-query and often slow. Here, we leverage upon a pre-computed global pseudo-inverse map to generate multi-contact motion from current configuration to the goal without need for an initial seed. The pseudo-inverse map is an inverse kinematics map for each contact-state that has a property of global resolution, connected by connectivity functions. Although global pseudo-inverse map is computationally expensive, once computed, it can be used to generate plans fast, possibly online, in a multi-query manner.

As an application of the framework, a recovery planner from fall down is developed that overcomes limitations of existing planner. First, the planner is not specific to a particular robot and can be used to generate motion of an arbitrary robot in general when its kinematics is given. Second, it does not rely on pre-defined motion and can generate feasible recovery motion even when a motor break-down.

In the process, singular configurations are utilized to produce torque efficient motions. Literature focuses on avoiding singularity due to loss of manipulability, but here we leverage singular configurations to plan for torque efficient motions as mechanical advantage is maximized at or near singularity.

1.7 Outline of Thesis

This thesis essentially consists of two parts. In the first part, a general planning framework to generate motion with contacts is developed. In Chapter 2, singularity is presented as a search heuristic with analysis of force and momentum generated through contact in relation to the singular positions. In Chapter 3, a general planning framework that involves building a global pseudo-inverse map is discussed with algorithms for generating nodes and connecting edges of the global map.

In the second part of the thesis, specific motion planners that applies to concrete problems is presented base on the framework developed in the first part. In Chapter 4, a multi-contact motion planner that plans recovery motion from fall down for humanoids is presented. In Chapter 5, a motion planner that solves actuated mathematical billiard problem motivated by an astronaut navigating inside the International Space Station is presented.

The thesis concludes with Chapter 6 where potential future work and applications are discussed in detail.

Chapter 2

Singularity as Search Heuristics

Constraint on the actuation and power resources is often the critical limiting factor for a robot to perform desired tasks. Increasing torque and energy capacity may be a solution, but is seldom viable for robots already built. An attractive alternative is to carefully generate motion trajectories that maximally leverages upon the limited torque and energy resources. In this endeavor, singularity, which is deemed undesirable due to lose of manipulability, could be utilized to an advantage. This chapter presents analysis of force and momentum generated through contact in relation to the singularity. The analysis shows that a motion at or near singularity not only maximally leverages the torque limits to generate forces in quasi-static motions, but is also optimally energy efficient for dynamical motion when it comes to momentum generation. As such, singular configurations can act as a search heuristic for planning contact based motions where torque is often the limiting factor.

Based on a simplified model, we discuss mechanical advantage aspects of a robotic leg and describe range of feasible forces that can be generated together with directions in which singular position becomes minimum torque configuration. Then we define stroke motion and establish upper bounds on the momentum generated through contact. Collinear stroke, where motion is along a straight line, is examined with respect to singularity.

2.1 Introduction

Limited torque is often the major limiting factor for a robot to perform certain motions. For example, humanoids in the Darpa Robotics Challenge rarely recovered from a fall to stand up. On the other hand, little humanoids at Robo-one competition recover graciously to a stand-up position after being knocked down. What prevents the former from performing recovery motion of the latter, lies in the torque. Humanoids at Robo-one are light-weighted and small in size that their joint torques are comparably large enough to perform aggressive and dynamic recovery. However, size and weight of humanoids at the DRC are hefty that a quick recovery maneuver is not possible from given torque limits.

One approach to overcome this issue is to directly increase the maximum torque capacity of the robot actuator. Success of BigDog and MIT Cheetah largely comes from it. BigDog uses hydraulic actuators powered by an engine. High torque available from the hydraulics enables BigDog to place its feet in the desired location quickly in time to prevent falling when disturbed [73]. MIT Cheetah has custom designed and built electrical motor that has significantly higher torque density than commercially available ones [81]. This, coupled with carefully designed leg enable the MIT Cheetah to perform fast dynamical running. However, directly increasing the torque density of the actuation involves re-designing and re-building of a robot hardware which is not often an option.

Limited energy often becomes the critical limiting factor when a robot needs to perform tasks untethered. For example, in a disaster rescue scenario, prolonged operation without tether is a must. Nevertheless, the operation time is severely limited by inefficient energy consumption of a humanoid whose specific cost of transportation is an order of magnitude higher than that of a human [13]. One way to overcome this issue is by equipping a robot with a power source of high energy density and capacity. Atlas in the DRC, for instance, carries onboard a heavy 3.7-kilowatt-hour lithium-ion battery pack. However, even then, the operation time from charge to charge is relatively short.

In the presence of constraints on the torque and energy resources, a more attractive approach is to carefully generate motion trajectories that leverage upon the limited torque to achieve a given task while being energy efficient. Number of works in the literature addresses this through use of optimization [2, 38, 48]. Torque and energy expenditure is encoded as a cost in the objective function and optimization machinery is cranked to produce desired trajectory. Nevertheless, it does not provide insight into the nature of the generated trajectory and little work is done that directly exploits the structure of a robot to an advantage.

The author believes that utilizing singularity is the key in this endeavor. For example, professional weight-lifter goes through sequence of singular positions to lift maximum weight possible from torque resources one has. In standing up from fall down, a person cannot directly push off the ground to a standing up position in one motion, but rather goes through sequence of singular position due to limited joint torque. Passive dynamic walker by McGeer [52, 53] walks down a slope maintaining singular position. It has cost of transportation similar to that of a human, and an order of magnitude better than humanoids that walk with knees bent. As such, singular configurations can act as a search heuristic for planning contact based motions where torque is often the limiting factor.

In this chapter, we present analysis of force and momentum generated through contact in relation to the singular positions. The analysis shows that a motion at or near singularity not only maximally leverages the torque limits to generate forces in quasi-static motions, but is also optimally energy efficient for dynamical motion when it comes to momentum generation. The chapter is organized as follow: Section II refreshes the concept of singularity and Section III shows modeling used throughout the analysis. Section IV presents analysis of force generated through contact for quasi-static motion, with discussion of mechanical advantage of singular position and effect of knee cap in minimizing torque. Section V addresses momentum generated through a stroke motion and how it is energy efficient near singularity.

2.2 Singularity

Singularity refers to configurations where the robot's degrees of freedom changes instantaneously [57]. For open-chain manipulators, it is directly related to drop in the rank of the forward-kinematics Jacobian. But for closed-chain mechanisms, several types of the singularities exists which differ in nature and several papers in the literature deal with its classification. Most notably, Gosselin [21] provides classification based on the Jacobian matrix corresponding to input/output coordinates and Park [67,68] presents a framework that geometrically classifies singularities invariant to the choice of local coordinates describing the kinematics.

This chapter focuses on the singularity that is classified as end-effector singularity, which is also referred as singularity of the first kind. Let θ and x represent input and output coordinates. Assuming that both θ and x have dimension n, equal to the degrees of freedom of the robot, the relationship between the two can be expressed as

$$F(\theta, x) = 0$$

where $F : \Re^n \times \Re^n \to \Re^n$. Differentiating with respect to time gives

$$\frac{\partial F}{\partial x}\dot{x} + \frac{\partial F}{\partial \theta}\dot{\theta} = 0$$

The end-effector singularity occurs when $\det(\frac{\partial F}{\partial \theta}) = 0$ and here, the end-effector loses one or more instantaneous degrees of freedom. This often corresponds to the robot configuration reaching boundary of its workspace.

In the literature, end-effector singularities are to be avoided because it directly leads to lose of manipulability. Inability to move and apply forces in an arbitrary direction is undesirable for manipulation, and large body of work deal with planning a path that avoids singularity [3,15,31,37,66,78]. Singularity also poses problem when performing inverse kinematics as inverse kinematic solutions often run into numerical stability issues at or near singular configurations [59].

Nevertheless, the end-effector singularity could be put to an advantage rather

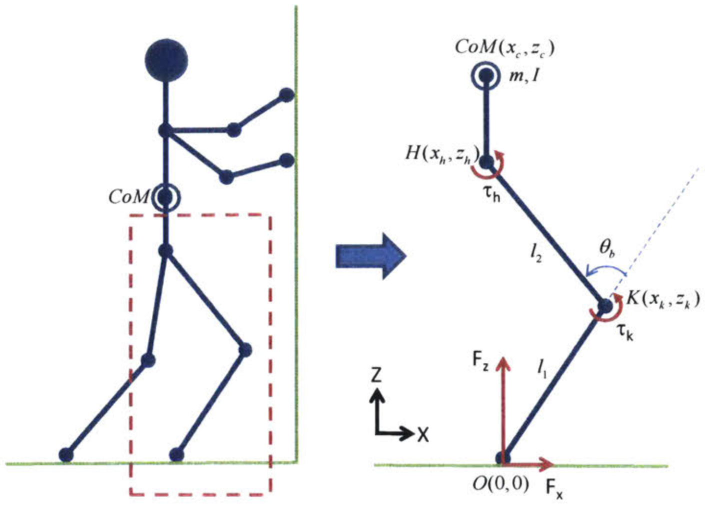


Figure 2-1: The chapter focuses on the anaylsis of how force and momentum are generated through contact in relation to singular positions. For the analysis, leg of a robot is modeled as a double inverted pendulum with massless links. A simple model is chosen over more elaborate models for advantage in gaining intuitive understandings.

than avoided when a motion must be performed within limited torque and energy resources. The end-effector losing degrees of freedom, from series-parallel duality [85], corresponds to existence of family of wrench that the end-effector can resists without having torque applied. Such property can be utilized as external wrench is largely opposed structurally near singularity and the chapter shows this through analysis.

2.3 Modelling

Motion of a robot - generalized velocity of its center of mass - can only be changed through external forces acting it. With the exception of gravity, these external forces come from contacts with the environment. While in contact, torque applied to the robot's joints generates external contact forces through actions of the leg and arm. Hence, understanding how a robot's leg and arm transforms joint torques to reaction forces at contact and generate momentum is essential.

The analysis of chapter focuses on the force and momentum generated by a single robotic leg (or arm) through contact with relation to singular configurations.

Table 2.1: List of Notations

Symbol	**Description**
$\overrightarrow{F^R}$	reaction force vector
F_x	reaction force in x-direction
F_z	reaction force in z-direction
τ_k	torque at knee joint
τ_h	torque at hip joint
$\overrightarrow{K_k}, \overrightarrow{K_h}$	mechanical advantage vectors
l	length of leg
l_1	length of link 1 (feet-knee)
l_2	length of link 2 (knee-hip)
r	link ratio
θ_b	bent angle between link 1 & link 2
$\overrightarrow{r_K}$	knee position vector $\overrightarrow{OK}$
$\overrightarrow{r_H}$	hip position vector $\overrightarrow{OH}$
S_θ	stroke angle
S_L	stroke length
S_T	stroke time

As shown in Fig. 2-1, a single leg (or arm) of a robot is modeled as a planar double inverted pendulum consisting of massless links. Each joints, hip and knee, are independently actuated with limits on the maximum torques that can be applied. Moreover, contact with the environment is treated as a point contact with friction and no slip is assumed.

This simple model is chosen over more elaborate models, as it not only makes the problem analytically tractable, but is also advantageous in gaining intuitive understandings. For the same reason, gravity is neglected in the analysis for the simplicity although its inclusion only adds few terms. Notations used in the chapter is listed in Table 2.1.

2.4 Force Generation

When a robot is stationary or moves very conservatively in a quasi-static manner, the contact wrench sum must counter-balance weight of the robot. Here, how much force each leg and arm in contact can generate from given torque resources becomes a question of interest which this section addresses. First, torque-force relation is established which shows that a given torque input is maximally leveraged to produce force in singular configuration. Then, description of how to graphically construct a region of feasible contact forces from given torque limits is detailed. Based on this, it is shown that, in the presence of knee cap, the singularity is minimal torque configuration for a large range of direction of force to be resisted.

2.4.1 Mechanical Advantage

Relationship between the force and torque can be derived from momentum principles which gives the following:

$$\begin{bmatrix} x_k & -z_k \\ x_h & -z_h \end{bmatrix} \begin{bmatrix} F_z \\ F_x \end{bmatrix} = \begin{bmatrix} \tau_k \\ \tau_h \end{bmatrix}$$

Using geometric relation, this can be rewritten as

$$\begin{bmatrix} F_z \\ F_x \end{bmatrix} = \frac{1}{\sin\theta_b} \begin{bmatrix} z_h & -z_k \\ x_h & -x_k \end{bmatrix} \begin{bmatrix} \tau_k \\ \tau_h \end{bmatrix}$$

Here, we define $\overrightarrow{K_k}$ and $\overrightarrow{K_h}$ which can be viewed as mechanical advantage vectors that maps τ_k and τ_h into $\overrightarrow{F^R}$

$$\overrightarrow{F^R} = \begin{bmatrix} F_z \\ F_x \end{bmatrix} = \overrightarrow{K_k}\tau_k + \overrightarrow{K_h}(-\tau_h)$$

$$\overrightarrow{K_k} = \tfrac{1}{\sin\theta_b}\overrightarrow{r_H} \ , \ \overrightarrow{K_h} = \tfrac{1}{\sin\theta_b}\overrightarrow{r_K}$$

The derived relationship and defined mechanical advantage vectors $\overrightarrow{K_k}$ and $\overrightarrow{K_h}$

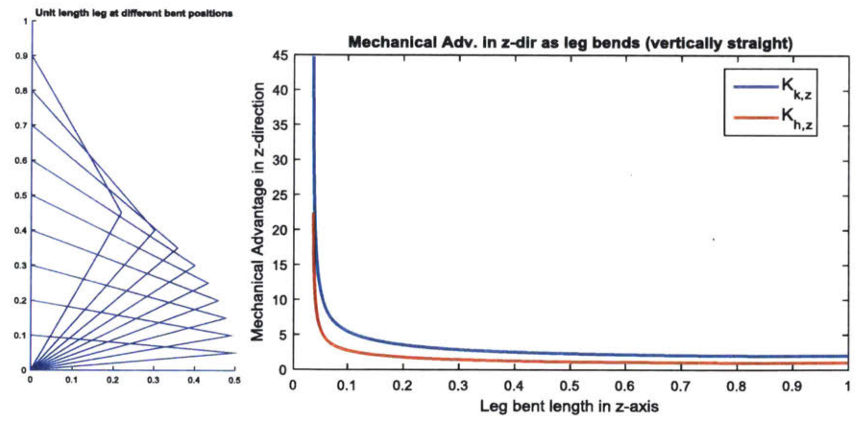


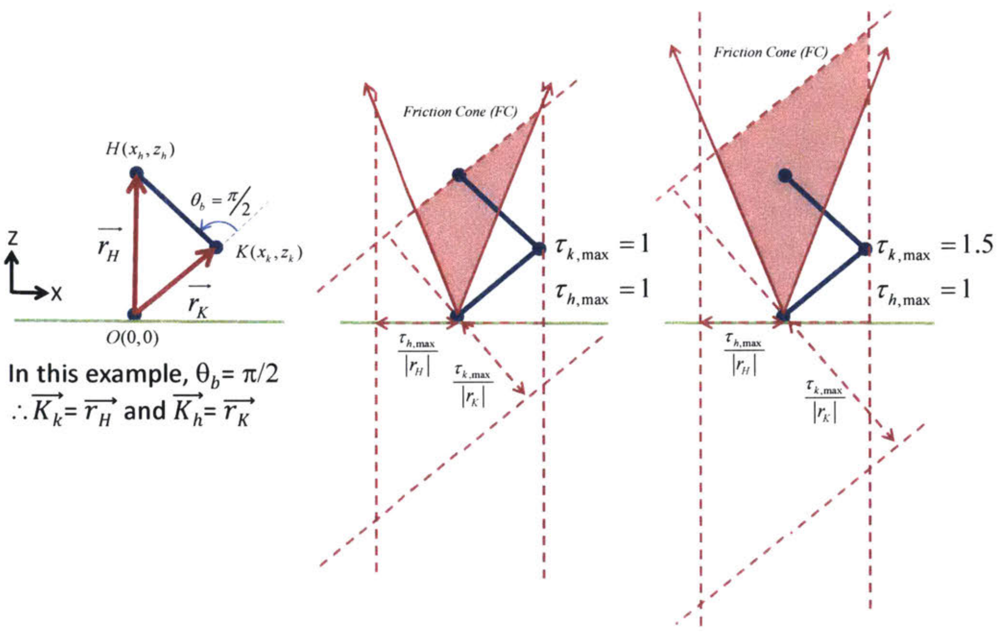
Figure 2-2: Mechanical advantage of a leg is maximized when unbent and quickly decreases as it bends. Operating in near or at the singular position (unbent) is advantageous in terms of generating large reaction force from the given torques.

permit following geometric interpretation. A torque applied to the knee τ_k produces reaction force $\overrightarrow{F^R}$ directed towards the hip ($\overrightarrow{r_H}$), whereas hip torque τ_h produces reaction force $\overrightarrow{F^R}$ directed towards the knee ($\overrightarrow{r_K}$). In addition, how much the leg is bent (θ_b) acts as a common scaling factor given by $\frac{1}{\sin\theta_b}$.

Mechanical advantage of a leg in generating reaction force from applied torques heavily depends on how much the leg is bent. Since $\frac{1}{\sin\theta_b}$ acts as a common scaling factor, mechanical advantage of a leg is maximized at singular position ($\theta_b = 0$) and quickly decreases as it bends. Fig. 2-2 shows a vertically upright leg at different bent positions and how z-component of the mechanical advantage changes respectively. It shows operating at or near a singular position holds significant advantage in generating large reaction force from a given torque.

2.4.2 Range of Feasible Reaction Forces

From the geometric relation, a range of reaction forces possible at a given configuration with torque limits $|\tau_k| \leq \tau_{k,\max}$ and $|\tau_h| \leq \tau_{h,\max}$ can be graphically constructed. As shown in Fig. 2-3, torque limit on knee ($\tau_{k,\max}$) constrains the reaction force to lie between two lines, parallel to $\overrightarrow{r_K}$, separated by a moment arm of distance $\frac{\tau_{k,\max}}{|r_K|}$. The same applies to torque limit on hip, and their intersection together with friction cone



$H(x_h, z_h)$
$\theta_b = \pi/2$
$K(x_k, z_k)$
$\overrightarrow{r_H}$
$\overrightarrow{r_K}$
z
x
$O(0,0)$
In this example, $\theta_b = \pi/2$
$\therefore \overrightarrow{K_k} = \overrightarrow{r_H}$ and $\overrightarrow{K_h} = \overrightarrow{r_K}$
Friction Cone (FC)
$\tau_{k,\max} = 1$
$\tau_{h,\max} = 1$
$\frac{\tau_{h,\max}}{|r_H|}$
$\frac{\tau_{k,\max}}{|r_K|}$
Friction Cone (FC)
$\tau_{k,\max} = 1.5$
$\tau_{h,\max} = 1$
$\frac{\tau_{h,\max}}{|r_H|}$
$\frac{\tau_{k,\max}}{|r_K|}$

Figure 2-3: A range of feasible reaction forces from a contact can be graphically constructed. Here, two cases with different torque limits are shown where shaded regions represent the range of feasible reaction forces that can be generated from the given torque limits.

gives the range of feasible reaction forces.

This can easily extend to find range of feasible reaction force for multiple link leg. As previously explained, each joint torque limit $|\tau_i| \leq \tau_{i,\max}$ introduces two inequality restricting the reaction force. Therefore, it can be easily seen that the range of feasible reaction force of a multiple link leg with given torque limits $|\tau_i| \leq \tau_{i,\max}$ forms a convex polyhedron where each edge corresponds to a limiting torque τ_i. From above argument, the following proposition becomes evident (proof is omitted here).

Proposition 1 (Relation between Range of Feasible $\overrightarrow{F^R}$). *Let L denote a multiple link leg at a particular configuration. Let L' denote a leg equivalent to L in the same configuration, with one or more joints structurally fixed. Then range of feasible reaction force for L' is always greater or equal to that of L.*

Prop.1 implies that having additional links, if it does not contribute to increase in manipulability, may be disadvantageous as it reduces the range of feasible reaction force possible. By the same token, structurally fixing links, if it does not diminish

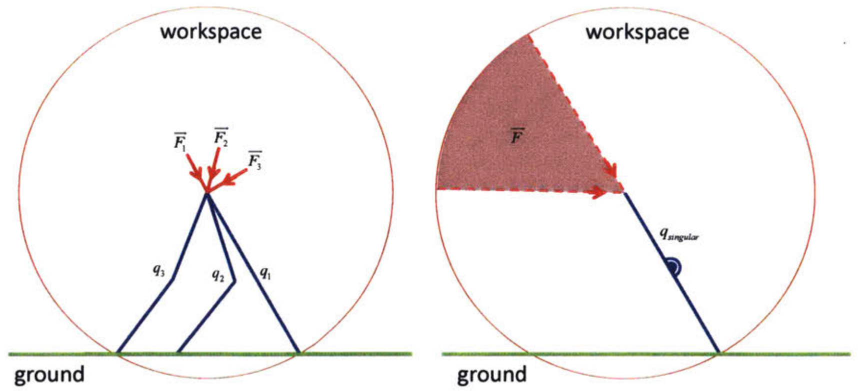


Figure 2-4: Configuration of the leg that minimizes the torque required to resist external force depends on the direction of force to be resisted. Three configuration, q_1, q_2, q_3, shown on the left are positions requiring minimum torque to resist forces directed at F_1, F_2, F_3 respectively. However, in the presence of knee cap, singular position becomes the configuration that minimizes the required torque for a large range of force directions. The range of direction of forces for which the singular position requires minimum torque to resist is shown as shaded region in the right.

manipulability, may be advantageous in terms of force generation. This happens to leg (and arm) by the action of a knee cap - at singular position, knee cap structurally fixes the two link into one without compromising the manipulability, and thereby increases the range of feasible force generated.

2.4.3 Minimum Torque Configuration

A leg in contact with the surface at a singular position can theoretically resist arbitrary force directed towards the leg with no torque applied. Therefore, when a leg makes contact to resist a force, it is always desirable to place it in a singular position in the direction of force to be resisted.

Nevertheless, this is not always possible when the position of robot's body is restricted with respect to the contact surface (e.g. when performing rock-climbing). In Fig. 2-4, for example, hip position of the leg is fixed in relation to the ground and it can be placed at a singular position only at two directions. Here, configuration that minimizes the torque differs depending on the direction of force to be resisted. Fig. 2-4 shows leg configurations that minimizes norm of the torque for different directions of the force. Note that these configuration, in general, are not at singularity.

Singular position, however, becomes a configuration that minimizes required torque for a large range of direction of forces in the presence of a knee cap. A knee cap makes the knee joint to act as structurally fixed at the singular position, and from Prop.1, this increases the range of feasible force. Moreover, it structurally provides torque required to resist the force at the knee joint, and thus helps to minimize the norm of torque input.

Proposition 2 (Singularity as Min. Torque Configuration). *Let $\tau_F = (\tau_{k,F}, \tau_{h,F})$ be torque required by the leg to resist a given force $\overrightarrow{F}$. Then, in the presence of knee cap, singular configuration minimizes $\|\tau_F\|$ when the force $\overrightarrow{F}$ to resist is directed at or away outwards from the singular position.*

Proof. (Sktech) Suppose $\overrightarrow{F}$ is directed at or away outwards from the singular position. Then the moment arm acting on the hip joint by the reaction force $-\overrightarrow{F}$ at the contact is shortest in the singular configuration compared to other feasible configurations and hence $|\tau_{h,F}|$ is minimized. Also, the reaction force $-\overrightarrow{F}$ at contact exerts torque on the knee joint which is structurally provided by the knee cap in the singular position, requiring $\tau_{k,F} = 0$. From the above two statements, it follows that singular configuration minimizes $\|\tau_F\|$, the torque required resist force $\overrightarrow{F}$. □

2.5 Momentum Generation

When a robot is stationary or moves very conservatively, the contact wrench sum only needs to resist weight of the robot and how much force each leg and arm in contact can generate is important. However, when a robot tries to perform a dynamic maneuver, contact wrench sum must equal the desired rate of change of centroidal momentum. Here, the question of how much momentum the leg in contact can generate becomes more relevant. This section presents analysis of momentum generated from a contact in relation to singular configuration. We first define a stroke motion in which a leg pushes off from contact to generate momentum. Then upper bounds on the momentum generated from a stroke is established with respect to the torque limits. Collinear stroke is discussed as a special case of the stroke and it is shown that momentum generated per energy expenditure is maximized approaching singular position.

2.5.1 Stroke Motion

A leg (or arm) in contact needs to push off from the contacting surface in order to generate momentum. Here, we formally define stroke that describes such motion through which a leg generates momentum. Parameters that are relevant are also defined.

Problem Formulation 1 (Stroke). *Let Stroke be a motion of a leg from* $t = t_i$ *to* $t = t_f$ *that satisfies the following:*
1) bent angle of a leg $|\theta_b(t)|$ *is monotonically non-increasing function for* $t \in [t_i, t_f]$
2) leg remains in contact without slipping for $^{\forall}t \in [t_i, t_f]$
3) $\theta_b(t_f) = 0$

Problem Formulation 2 (Stroke Angle, Length, and Time). *Let Stroke Angle* (S_θ) *be defined as* $S_\theta \triangleq |\theta_b(t_f) - \theta_b(t_i)| = |\theta_b(t_i)|$ *and Stroke Length* (S_L) *as* $S_L \triangleq |\overrightarrow{r_H}(t_f)| - |\overrightarrow{r_H}(t_i)|$ *where* $\overrightarrow{r_H}$ *is a position vector of the hip joint. Time duration* $\Delta t = t_f - t_i$ *of a stroke is defined as Stroke Time* (S_T)

An example of a stroke motion is shown in Fig. 2-5. Notice that stroke length S_L and stroke angle S_θ are geometric properties with the following relation

$$S_L = |\overrightarrow{r_H}(t_f)| - |\overrightarrow{r_H}(t_i)| = l_1 + l_2 - \sqrt{l_1^2 + l_2^2 + 2l_1l_2\cos(S_\theta)}$$

Once S_θ (or S_L) is given and the torque inputs are known, stroke time S_T can be calculated by solving through governing equation of motion.

Momentum generated from a stroke can be expressed as

$$m\dot{z}_c(t_f) - m\dot{z}_c(t_i) = \int_{t_i}^{t_f} F_z(t)dt$$

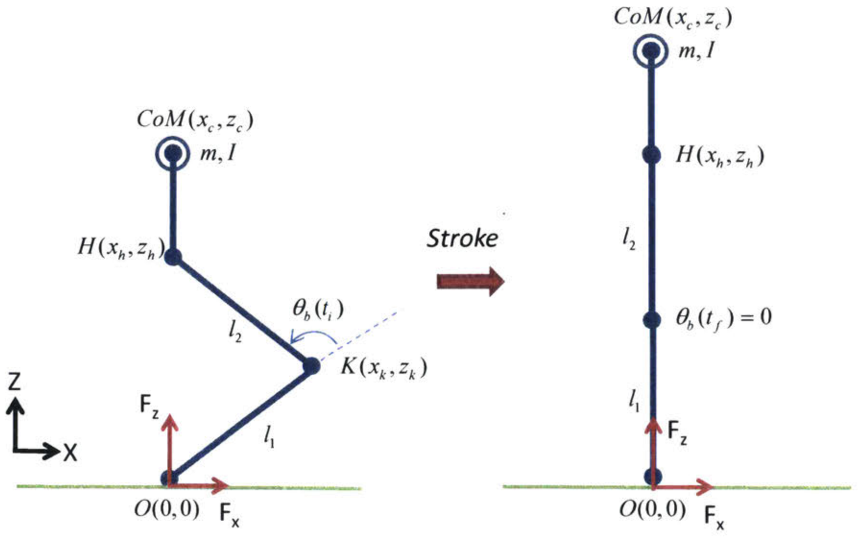


Figure 2-5: A leg (or arm) in contact needs to push off the contacting surface to generate a needed momentum. Here, an instance of a stroke motion is shown as defined, where the leg starts from a bent position and monotonically decreases its bent angle till fully stretched. This is also an example of a collinear stroke as defined, since the contact point, hip joint, and center of mass remain in a single line throughout the stroke.

$$m\dot{x}_c(t_f) - m\dot{x}_c(t_i) = \int_{t_i}^{t_f} F_x(t)dt$$

where x_c, z_c and F_x, F_z are z,x components of CoM position and reaction force at contact as shown in Fig. 2-5. Ideally, to maximize the momentum generated from a stroke, both F_z , F_x and stroke time $S_T = t_f - t_i$ must increase. For the given torque limits, F_z and F_x can be increased by reducing the bent angle θ_b since mechanical advantage in both z and x direction scale commonly with $\frac{1}{\sin\theta_b}$.

However, reducing θ_b to increase F_z and F_x comes at a cost of decreasing the stroke time S_T. It can be easily shown that reducing θ_b decreases both S_θ and S_L which leads to a shorter S_T. In fact, F_z and F_x is maximized when $\theta_b = 0$ (leg unbent) but this makes $S_T = 0$ and no momentum can be generated in singular configuration. Hence F_z , F_x and $S_T = t_f - t_i$ cannot both be increased and the momentum generated from a stroke is bounded above.

2.5.2 Upper Bounds on Momentum Generated

It is useful to establish upper bounds on the momentum generated from a stroke. This can be done by directly calculating the forces and working through the equation of motion. Nevertheless, it is more easily derived by considering the work done by torques at each joints.

From work-energy principle, we have following where last two terms represent the work done by torques at knee and hip respectively

$$T_2 - T_1 = \frac{1}{2}m\left|v_c(t_f)\right|^2 - \frac{1}{2}m\left|v_c(t_i)\right|^2$$

$$= U_{1\to 2} = \int_{\theta_b(t_i)}^{\theta_b(t_f)} \tau_k(\theta)d\theta + \int_{\theta_i(t_i)}^{\theta_i(t_f)} \tau_h(\theta)d\theta$$

Since maximum angle through which the joint torques can do work is π, it follows

$$U_{1\to 2} \le \tau_{k,\max}\int_{\theta_b(t_i)}^{\theta_b(t_f)} d\theta + \tau_{h,\max}\int_{\theta_i(t_i)}^{\theta_i(t_f)} d\theta \le \pi(\tau_{k,\max} + \tau_{h,\max})$$

Assuming the stroke starts from a rest ($v_c(t_i) = 0$), we have

$$\frac{1}{2}m\left|v_c(t_f)\right|^2 - \frac{1}{2}m\left|v_c(t_i)\right|^2 = \frac{1}{2}m\left|v_c(t_f)\right|^2 \le \pi(\tau_{k,\max} + \tau_{h,\max})$$

Hence an upper bound on the momentum generated from a stroke can be established as

$$\left|\Delta m v_c\right| = \left|m v_c(t_f) - m v_c(t_i)\right| \le \sqrt{2m\pi(\tau_{k,\max} + \tau_{h,\max})}$$

This expression gives the upper bound on the momentum generated from a stroke and equality holds when maximum torque is applied at each joint and each joint moves through maximum angle of π through the stroke. It can easily be extended to multiple n-link legs, where the upper bound is given as $\left|\Delta m v_c\right| \le \sqrt{2m\pi\sum_{i=1}^{n}\tau_{i,\max}}$ where $\tau_{i,\max}$ is the torque limit of the i-th joint.

The established upper bound does not contain any information on the directionality of the momentum generated. Depending on the direction of momentum generated, a stroke motion that equals the upper bound may fail to exist. In fact, finding an optimal stroke motion that maximizes the momentum generated in a desired direction is not trivial.

2.5.3 Collinear Stroke

This subsection examines *collinear stroke* where the contact point(O), hip joint(H), and center of mass(CoM) all lie in a single line throughout the stroke motion. In many instances where a robot's leg or arm pushes off the contact surface, the motion is or close to being collinear. Thus, it is worthwhile looking into collinear stroke as a special case of a stroke.

Fig. 2-5 shows an instance of a collinear stroke. In this example, the line through which the contact point, hip joint, and center of mass lie is vertical, but it can also be slanted or even horizontal (e.g. when pushing off a vertical wall). In a collinear stroke, torque applied to hip should be kept zero to ensure that the motion remains collinear. This is because the hip torque τ_h creates a reaction force directed towards the knee as shown in Section 2.4. On the other hand, the knee torque τ_k creates a reaction force inline with the collinear motion. When the collinear stroke is vertical as in Fig. 2-5, vertical force F_z produced by τ_k is given by

$$F_z = K_{k,z}\tau_k = \frac{z_h}{\sin\theta_b}\tau_k = \frac{1}{x_k}\tau_k$$

and the equation of motion for the stroke simply becomes

$$m\ddot{z}_c = F_z = \frac{1}{x_k}\tau_k$$

where x_k can be expressed in terms of stroke angle S_θ or stroke length S_L from the given geometry.

The momentum generated through a collinear stroke can be related to the given parameters, stroke angle S_θ, stroke length S_L and stroke time S_T. Following from the

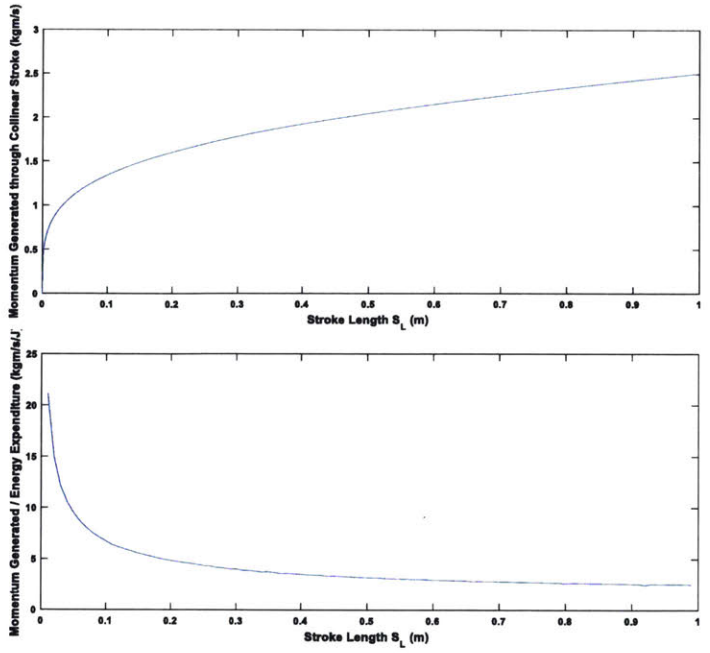


Figure 2-6: Stroke motion near singularity is highly desirable in term of the energy efficiency and momentum generated. Near singularity, even a small stroke length S_l is sufficient to generated momentum comparable to the max momentum generated through full stroke (Above). Moreover, momentum generated per unit energy dramatically increases near singularity, making it energy efficient (Below).

prior analysis, relationship between the momentum generated and S_θ becomes

$$|mv_c(t_f) - mv_c(t_i)| = \sqrt{2m\tau_{k,\max} S_\theta}$$

Assuming length of the links $l_1 = l_2 = 0.5l$ (l = length of the leg), relation between S_θ and S_L can be established through geometry as

$$S_L = l(1 - \sqrt{\frac{1 + \cos(S_\theta)}{2}})$$

Using the relation, momentum generated can then be expressed in terms of S_L

$$|mv_c(t_f) - mv_c(t_i)| = \sqrt{2m\tau_{k,\max} \cos^{-1}(2(1 - \frac{S_L}{l})^2 - 1)}$$

It is interesting to note that the momentum generated is a concave function of

S_θ, and that S_θ is also a concave function of S_L. Because of this, even a small stroke length S_L is sufficient to generate momentum comparable to the max momentum from a full stroke length $\bar{S}_L$. Fig. 2-6 illustrates this where for a leg with equal link lengths, it only needs to bend one-tenth of the leg length to produce more than half of the max momentum generated through bending to the limit of entire leg length.

Finding Stroke Time S_T requires solving through differential equation describing equation of motion and it is closely related to Joule loss in the electric motor. The energy consumed by the motor is proportional to $\int_{t_i}^{t_f} {\tau_k}^T \tau_k dt$. Assuming max torque $\tau_{k,\max}$ is applied throughout the vertical stroke, energy expenditure becomes proportional to S_T

$$\int_{t_i}^{t_f} {\tau_k}^T \tau_k dt = {\tau^2}_{k,\max} \int_{t_i}^{t_f} dt = {\tau^2}_{k,\max} S_T$$

Hence, for equal momentum gained through a stroke motion, having a smaller Stroke Time S_T is more energy-efficient. Since mechanical advantage of a leg is significantly higher near singularity, for equal momentum gained, stroke time S_T is much smaller near singular position (leg unbent). This is shown in the Fig. 2-6 which indicates that momentum generated through stroke per unit energy consumed is maximal near singularity.

These two facts put together makes stroke motion near singularity highly desirable. Stroke motion near singularity operates in a region where mechanical advantage is high (large reaction force possible from given torque limits), momentum generated per energy consumed is maximal (optimally energy efficient), and yet able to generated momentum comparable to the maximum from a small Stroke Length S_L.

2.6 Conclusion

In this chapter, forces and momentum generated through contact has been analyzed with focus on the singular configuration. We discussed mechanical advantage near singularity and described geometrical construction of the range of feasible reactions forces. It is shown that singular position minimizes the torque required to resist forces coming from large range of directions. Moreover, we have established an upper bound on the momentum generated through a stroke motion and showed that operating collinear stroke near singularity is highly desirable in terms of energy efficiency.

The analysis shows that a motion at or near singularity not only maximally leverages the torque limits to generate forces in quasi-static motions, but is also optimally energy efficient for dynamical motion when it comes to momentum generation. These properties make singularity attractive when desired motion must be performed within limited torque and energy resources which floating-based robots bound to have. Therefore, singularity is utilized as a search heuristics in the motion planning framework to follow.

Chapter 3

Global Pseudo-inverse Map

3.1 Motivation

The existing multi-contact planners in the literature share a common shortcoming in that they require good initial seeds to successfully generate a motion. Depending on the task to perform, these initial seeds can be extremely difficult to find and often have to be manually encoded. Moreover, these planners are single-query, slow to converge and vulnerable to local minima.

Hauser [25] presented a multi-contact planner which laid foundation that was followed by subsequent works [4–6, 17, 18]. The planners developed in these work gives a single solution for a task, convergence is slow, and must be recomputed even for similar tasks. Posa [72] developed optimization based trajectory planner to deal with contact but it too relies on having a good initial seed and is single query in nature. Mordatch [55] proposed a multi-contact planner that can perform versatile tasks that closely resemble human-like motion. But these impose soft-constraints that are dynamically feasible. To remedy the slow convergence, Park [48] and Bicchi [1] utilized principle component that can generate motion quickly, but these are suitable for natural movement generation and task specific. Similarly, Kris [24] and Frazzoli [19] presented planners using motion primitives for a fast planning which is close to the approach taken in the thesis but is local in nature.

Here, we propose a multi-contact motion planning framework that utilizes global pseudo-inverse map which are pre-computed offline. The global pseudo-inverse map is a set of inverse kinematic map for each contact-state that has property of global resolution which are then connected by connectivity functions. It is essentially a roadmap that captures the global structure of the problem that can be used to generate multi-contact motion from current configuration to the goal without need for an initial seed. Although global pseudo-inverse map is computationally expensive, once computed, it can be used to generate plans fast online in a multi-query manner.

3.2 Problem Definition

In this framework, several assumptions are to make the problem tractable. First, we assume that the environment, objects, and robot are all rigid-bodies that do not undergo deformation. A contact between two rigid bodies, in general, can be a point contact, a edge contact, or a face contact depending on the situation. Here, we assume all the contacts between the robot and the environment to be point contacts. Both edge contacts and face contacts are thereby reduced to equivalent point contacts. Friction at the point of contact is assumed to be following Coulomb model of friction characterized by a friction cone with static coefficient.

In real scenario, an arbitrary part of the robot can come into contact with the environment. In the framework, arbitrary point of contacts along the robot's body is allowed but must be pre-designated in advance. Once chosen, the planner allows the robot to make contacts only at these pre-designated points along its body. In the following, we provide definition for the key terms used.

3.2.1 Contact State

In the framework, the planner will allow contact to be made only at pre-designated points of the robot. A contact-state c defines which of the robot's pre-designated points are in the state of contact. To be more precise, a contact-state c is a binary vector of size N, where N is cardinality of the pre-designated contact points. Each entry of the vector c corresponds to a pre-designated point and takes on value 1 if the pre-designated point is in contact and 0 otherwise.

To illustrate this, Fig. 3-1 shows a five-linkage robot in two different contact state. In the example, six points in the robot are pre-designated for contact, and these correspond to feet, knee, hip, shoulder, elbow, hand. Hence, contact state c is a binary vector of size six. When the robot is completely lying down on the ground, as shown on the left of the figure, all six points are in contact and robot's contact state is $c = [1, 1, 1, 1, 1, 1]$. On the other hand, when the robot is standing on its feet as on the right, only feet is in contact. In this configuration, the robot's contact state is $c = [1, 0, 0, 0, 0, 0]$.

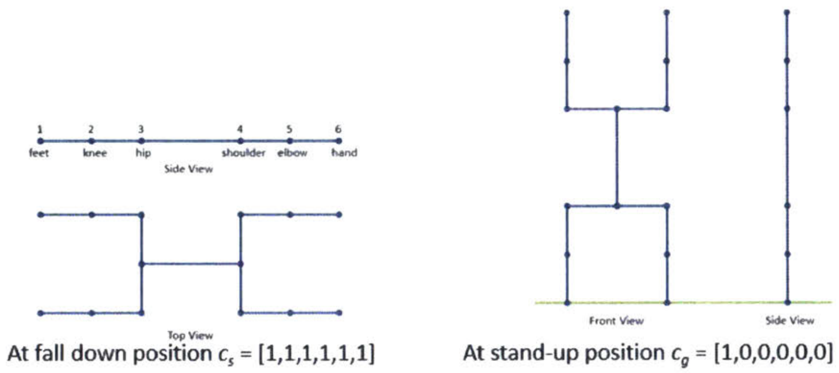

1
2
3
4
5
6
feet
knee
hip
shoulder
elbow
hand
Side View
Top View
At fall down position c_s = [1,1,1,1,1,1]
Front View
Side View
At stand-up position c_g = [1,0,0,0,0,0]

Figure 3-1: Simple skeleton robot is shown in two different contact state.

Two contact states c_i and c_j are are called adjacent contact states if they differ by one bit. That is they share common contact state except for one pre-designated point.

3.2.2 Sample within Contact State

Contact state c only gives information on which of the pre-designated points are in contact. A sample w within contact-state defines coordinates of the points in contact. A sample w is a vector whose dimension equals size of the workspace W multiply by $N(c)$, where $N(c)$ is number of 1 bits in the contact state c.

To illustrate this, Fig. 3-2 shows a five-linkage robot in configurations corresponding to three different samples within a same contact state. Here, knee and hand of the robot is in contact so its contact state is $c = [0, 1, 0, 0, 0, 1]$. Since it is 2D robot with ground being 1D line, the workspace $W = R$ and $N(c) = 2$ as only two contacts are made. Thus, a sample w is real vector of size two giving coordinates of two contact points. From hereinafter, we refer to workspace W as space of the sample, deviating away from the conventional meaning of workspace used for manipulation.

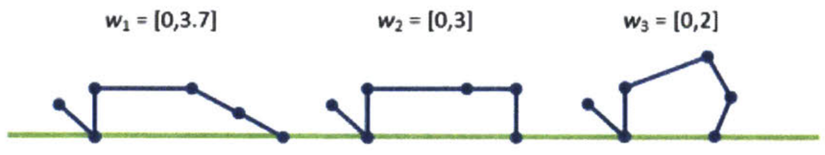


Figure 3-2: Simple skeleton robot with configurations shown for different samples within same contact state $c = [0, 1, 0, 0, 0, 1]$.

A sample within contact state is called feasible if there exists at least one configuration of the robot that satisfies kinematic constraints while making contacts as defined by the sample. If no such configuration exists then the sample within contact state is called infeasible.

Two samples w_i and w_j in two different contact states c_i and c_j are called adjacent samples if following holds: 1) c_i and c_j are adjacent contact states, and 2) w_i and w_j share the common contact coordinates except for one.

3.2.3 Configuration within Sample

When a sample w within contact state c is specified, a robot can be placed in a configuration to satisfy it. We define $q(w)$ as a robot configuration whose contact coordinates equals w.

It is important to note that for a given sample w, there may be multiple configurations $q(w)$ that satisfy w. In fact, due to redundancy of a robot, a sample would rarely fix a robot to one configuration. Given a sample w, configuration $q(w)$ can lie anywhere in the self-manifold of the robot.

Fig. 3-3 is shown to illustrate this point. Here, three different samples, w_1, w_2, w_3, are given within a contact state $c = [0, 1, 0, 0, 0, 1]$. Due to redundancy, placing

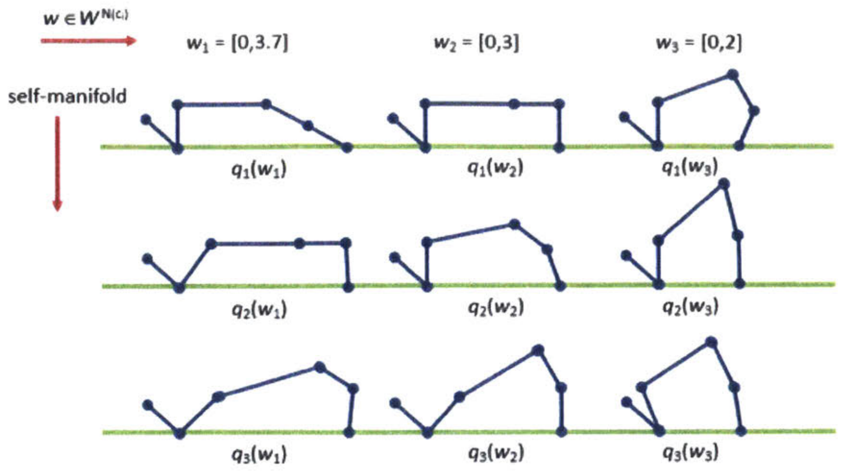


Figure 3-3: Different configurations are shown for given samples that lie within self-manifold.

robot's knee and hand at coordinates specified by w does not fix the robot to one configuration. The figure shows several configurations $q_1(w)$, $q_2(w)$, $q_3(w)$, which lie within the self-manifold and are all feasible.

3.2.4 Redundancy Resolution

For a given sample w in a workspace W, multiple configurations $q(w)$ are possible due to redundancy. Similarly, when a path is given in a workspace, configurations corresponding to samples along the path can be arbitrary and disjointed. However, from motion planning perspective, it is highly desirable to come up with a continuous configuration-space path that correspond to the given workspace path. Once such continuous configuration-space path is established, motion planning along its corresponding workspace path becomes trivial and deemed solved.

The problem of finding or generating a continuous configuration-space path for a given workspace path is called pathwise redundancy resolution. Several works in the robotics literature [51,58,65,77] presents methodology for solving the redundancy resolution.

When the redundancy resolution is extended from a path in a workspace to the entire workspace, it is called global redundancy resolution. The problem of global redundancy resolution is finding continuous pseudo-inverse function that maps samples in the workspace to configurations in the configuration-space. If such function exists and is computed, then motion planning problem along any path in the workspace can easily be found.

Few works in the literature deals with the problem of global resolution [8, 23, 50]. Global redundancy resolution may fail to exist because of joint limits, singularities, and self-collisions. However, computing piecewise-continuous pseudo-inverse function is still highly desirable as it solves the motion planning problems along the paths in the workspace where continuity holds. In the planning framework, we directly utilize global redundancy resolution to compute continuous pseudo-inverse function of contact-states. It is motivated by and closely follows Hauser's recent work [23].

3.3 Overview of Multi-contact Planning Framework

The planning framework consists of two phases. In the first phase, a global pseudo-inverse map is generated that consists of pseudo-inverse map for each of the contact-states which are linked by by connectivity function. In the second phase, a query is made on the global pseudo-inverse map to find motion from given initial configuration to a goal configuration.

In the global pseudo-inverse map construction phase, a set of pre-designated points along the robot's body that are allowed contact are determined. This usually corresponds to joints that can come into contact. A set of contact state C has a total of 2^N contact states where N is number of pre-designated contact points.

For a contact state $c_i \in C$, a pseudo-inverse function $f(c_i)$ is computed that is maximally continuous based on appropriate samples (e.g. grid network or random samples). The pseudo-inverse function $f(c_i)$ maps samples in the workspace defined by the contact state c_i to configurations in the configuration-space while attempting to maintain redundancy resolution. The configurations of samples are added as vertices to a graph $G = (V, E)$ and resolved paths are added as edges to the graph.

This is repeated for each of the contact state $c_i \in C$ and the planner now equipeed with pseudo-inverse functions $f(c_i)$ for all $c_i \in C$.

For every adjacent sample pairs w_i and w_j in adjacent contact state c_i and c_j, connectivity function $k(q(w_i), q(w_j))$ is computed. Connectivity function represents feasibility of motion between two given configurations. If $k(q(w_i), q(w_j)) = 1$, then $(q(w_i), q(w_j))$ is added to the edge of the graph $G = (V, E)$.

In the query phase, a motion is generated using the pre-computed global pseudo-inverse map above. First, for a given initial configuration q_s and a goal configuration q_g, we determine contact states c_s and c_g that these configurations belong to. Then steering is attempted to the neighboring configurations q'_s and q'_g that are vertices of the graph G and within the same contact state c_s and c_g. These steering is done through checking feasibility of sliding between q and q'.

If steering to neighboring configurations within same contact state fails, then configuration q_s and q_g is extended a set of configurations Q_s and Q_g in its adjacent samples where $q''_s \in Q_s$ are feasible configurations that are equivalent to q_s but in its adjacent contact state by breaking one contact. Then steering is attempted to each of q'' to its neighboring configurations in the vertex of the graph within same contact state as before till a successful steering is made.

Once steering to q'_s and q'_g is established, then graph search is performed on the graph $G = (V, E)$ to find a path. Path is not unique and a optimal path with respect to a given objective can be found depending on the information encoded on the edges.

In the following sections, we describe in detail how the pseudo-inverse map is computed for each of the contact state and connectivity is checked for two configurations.

3.4 Pseudo-inverse Function

For a contact state $c_i \in C$, a pseudo-inverse function $f(c_i)$ is computed that is maximally continuous based on appropriate set of samples (e.g. grid network or random samples). The purpose of the pseudo-inverse function $f(c_i)$ is to map samples in the workspace to configurations in the configuration-space while attempting to maintain redundancy resolution. Note that, the dimension of the workspace W will depend on the number of contact points made and hence will be defined by the given contact-state.

To begin with, a set of samples are selected that will sufficiently cover the workspace. Examples of such set could be 1) samples from regular grid, or 2) random samples from uniform distribution, but is left for the user to determine. Once a set of samples are selected, we form a graph $Gw = (V_w, E_w)$ where samples y are vertices V_w and straight line between two adjacent samples (w,w') form edges E_w. For each of the sample vertices, the pseudo-inverse function $f : w \in V_w \rightarrow q(w)$, computes corresponding configuration q in configuration-space through performing inverse kinematics. While doing so, the pseudo-inverse function tries ensure that edges $(w, w') \in E_w$ are path-wise resolvable. The pseudo-inverse function returns a graph $Gc = (V_c, E_c)$, where $q(w)$ are form vertices V_c and edges E_c are established between $q(w)$ and $q(w')$ if (w, w') is path-wise resolvable. The mapping between G_w and G_c is given by $Q[w]$ which connects vertices V_w in workspace to its corresponding vertices V_c in configuration-space.

For each contact-state c_i , pseudo-inverse function $f(c_i)$ is computed via following algorithm. It is adaptation from the Pointwise-Global-Resolution algorithm presented in Hauser's work [23].

Algorithm 1 : $G_c \leftarrow$ Contact-State Pseudo-inverse(G_w), $G_w = (V_w, E_w)$

```
1: Initialize configuration-space graph $G_c = (V_c, E_c)$
2: for each $w \in V_w$ do
3:     Let $Q_n \leftarrow \cup Q[w_n]$, $w \in Neighbor(w)$
4:     for each $q_n \in Q_n$ do
5:         $q \leftarrow$ InverseKinemtics$(w, q_n)$
6:         if Feasible$(q)$ then
7:             Add $q$ to $V_c$ and goto Step 2
8:     $q \leftarrow$ SelectOptimal$(w)$
9:     Add $q$ to $V_c$
10: for all edges $(w, w') \in E_w$ such that $Q[w] \neq \emptyset$ and $Q[w'] \neq \emptyset$ do
11:     if Reachable$(w, w', Q[w], Q[w'])$ then
12:         Add $(q, q')$ to $E_c$
13:     if Reachable$(w', w, Q[w'], Q[w])$ then
14:         Add $(q', q)$ to $E_c$
15: return $G_c$
```

Several subroutines are called for in Alg. 1 and here are details of these subroutines. InverseKinemtics(w, q) is a routine that solves inverse kinematics for a given workspace sample w using q as a initial seed. A common inverse kinematics solver uses Newton-Rhapson method to iteratively converge to a solution from the initial guess. But there are variety of inverse kinematics solvers available and user can choose the solver. In line 6, q found from the InverseKinematics is checked for feasibility. The feasibility test varies from one problem to another and is defined by what is accepted as being feasible. Feasible(q) return true if given configuration q passes feasibility test and false otherwise.

SelectOptimal(w) is a routine that first performs inverse kinematics to form a set of candidate configurations. From the set of configurations, it then select one configuration that is optimal with respect to the cost defined by the problem. The exact detail of the SelectOptimal(w) will differ for different problems but the returned optimal configuration should bring about least discontinuity in the redundancy resolution.

Reachable($w, w', Q[w], Q[w']$) checks whether straight path $\overline{ww'}$ formed by two neighboring workspace samples w and w' is path-wise resolvable. Following Alg. 2 computes Reachable. It is similar to algorithm presented in [83] with inclusion of additional checks for feasibility and slidability.

Algorithm 2 : Reachable(w, w', q, q')

```
1: if $d(q,q') \leq \varepsilon$ then return $true$
2: Let $w_m \leftarrow (w + w')/2$ and $q'' \leftarrow (q + q')/2$
3: Let $q_m \leftarrow$ InverseKinematics($w_m, q''$)
4: if $\neg$Feasible($q_m$) or $\neg$Slidable($q_m, w_m, w'$) then return $false$
5: if $\max(d(q_m, q'), d(q_m, q)) > c \cdot d(q, q')$ then return $false$
6: if Reachable($w, w_m, q, q_m$) and Reachable($w_m, w', q_m, q'$) then return $true$
7: return $false$
```

The subroutine Slidable(q, w, w') checks if a robot in configuration q can slide itself in the direction of $w \dashrightarrow w'$ within the torque limits.

Fig. 3-4 illustrate the computation process of the Pseudo-inverse Function $f(c_i)$. From the given samples w (shown as dots inside the workspace W) that form vertices, corresponding configuration $q(w)$ in the configuration-space C (shown as dots inside the configuration-space C) is computed. The selection of $q(w)$ is made using neighboring configurations as initial seed if they exit, or according to defined cost otherwise, to ensure maximal continuity in the global resolution. Once this process is complete, the algorithm tries to solve for path-wise resolution of the edges in the workspace. When an edge is path-wise resolvable, connections are made and the samples in the workspace can be grouped into connected components (shown as sample with arrows connections in the workspace W). The samples in a connected component can be transversed from one to another through sliding.

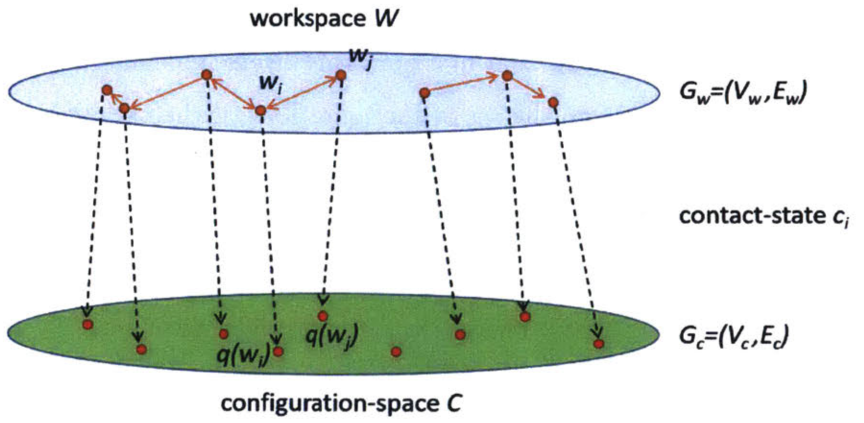


Figure 3-4: From the given samples w (shown as dots inside the workspace W) that form vertices, corresponding configuration $q(w)$ in the configuration-space C is computed. The selection of $q(w)$ is made using neighboring configurations as initial seed if they exit, or according to defined cost otherwise, to ensure maximal continuity in the global resolution. Then each of the edges in the workspace is checked for path-wise resolution.

3.5 Connectivity Function

Pseudo-inverse function maps out feasible motions within samples inside a same contact-state. Once pseudo-inverse function is computed for each of the possible contact-states, then connectivity function generates cross connections between the samples in the different but adjacent contact states.

For every adjacent sample pairs w_i and w_j in adjacent contact state c_i and c_j, connectivity function $k(q(w_i), q(w_j))$ is called for. Connectivity function represents

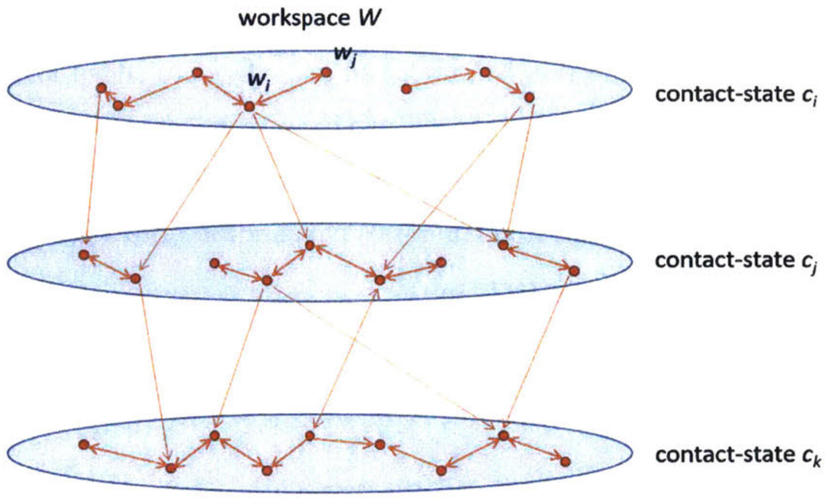


Figure 3-5: Once pseudo-inverse function is computed for each of the contact-state c_i, connectivity between adjacent contact-states are established through connectivity function. Each of the adjacent sample pairs w_i and w_j in adjacent contact state c_i and c_j are checked edges are established for feasible transition between the two (shown as arrow connecting two sample in different contact-state).

feasibility of motion between two given configurations. If $k(q(w_i), q(w_j)) = 1$, then $(q(w_i), q(w_j))$ is added to the edge of the graph $G = (V, E)$. As defined previously, two samples w_i and w_j that are adjacent, share the same contact point locations except for one. Here, without loss of generality, we can assume w_j to have an addition contact over w_i.

Transitioning from w_i to w_j is often feasible as it is making an additional contact which provides extra support. The harder transition is from w_j to w_i where breaking of an existing contact has to occur. If the transition motion is confined to being quasi-static, checking for the center-of-mass being inside the supporting polygon can be used as an effective tool to determine the feasibility of the motion. However, this will restrict the quantity of cross-connections that are made between adjacent samples in different contact-state. Far more cross-connections can be established if dynamic motions are allowed. This, however, greatly increases the computation cost as finding dynamic motion connecting two samples is non-trivial.

For computing connectivity function to establish cross-connections, user can choose a method that is most suitable for the problem domain. For example, it can be a sampling-based planner to find motion from sample w_i to w_j, or an optimization-based planner. Once connection is established and edges are added for the feasible connections, then we now have a graph $G = (V, E)$ that approximately maps out a global roadmap of the problem. Multi-queries can be made on this global map to find multi-contact motion solutions from an inital configuration q_s to a goal configuration q_g which belong to different contact-states.

3.6 Conclusion

In this chapter, we presented a framework for the multi-contact motion planning which utilizes global pseudo-inverse map for a fast planning. The global pseudo-inverse map is a set of inverse kinematic map for each contact-state that which are connected by connectivity functions. It is essentially a roadmap that captures the global structure of the problem that can be used to generate multi-contact motion from current configuration to the goal without need for an initial seed. The construction of the global map is done such that the redundancy resolution is maximized between the paths and the algorithms for construction is detailed.

Although global pseudo-inverse map is computationally expensive, once computed, it can be used to generate plans fast online in a multi-query manner.

Chapter 4

Recovery Planner from Fall Down

4.1 Motivation

The recent Darpa Robotics Challenge aimed at demonstrating capabilities of a robot to operate in a disaster scenarios. The competition presented several tasks that were challenging for a humanoid robot to perform. But perhaps the biggest challenge came not from the tasks itself, but from being able to balance itself as to prevent fall down. As shown in Fig. 4-1, Darpa Robotics Challenge has been marked with compilation of humanoid robots falling down. But of all the fall down events, only one robot was able to recover to a standing up position. All the other robots required having manual intervention to have itself up whenever there was a fall down.

The implication of this is severe and defeats the mission objective of the Darpa Robotics Challenge to a great extent. A disaster scene will be clustered with objects and obstacles that are unknown priori and losing balance will inevitably occur to a robot operating in such environment. Here, the inability to self-recover from a fall down would mean end of the rescue mission for what is an extremely expensive machinery.

The challenge lies in the fact that human-sized humanoid robots have limited torque. For example, small humanoids at Robo-one competitions battle each other and points are awarded to the one that knocks down the opponent. All the humanoids must be able recover from fall down and stand up to fight another round. In fact,

Figure 4-1: Darpa Robotics Challenge has been marked with compilation of humanoid robots falling down. But of all the fall down events, only one robot managed to recover to a standing up position. Image retrieved from https://spectrum.ieee.org/automaton/robotics/humanoids.

majority of the humanoids recover from the fall down in simple maneuvers. However, these simple maneuvers are not possible by human-sized humanoids because its torque-to-weight ratio significantly smaller compared to small humanoids (torque increases by the square of the dimension but weight is increased by the cube). Due to the limitation in the torque, the ensuing recovery motion has to be multi-contact in nature - the robot will have to make contact on several different part of the body (e.g. hand, knee, elbow) to support its own weight during the recovery action. Nevertheless, planning a multi-contact motion is non-trivial as each contact made changes the kinematics and dynamics of the robot.

4.2 Related Work

In the robotics literature, there are several bodies of work that tackle the problem of recovering from fall down dating back to 1995. Inaba et al [29] first worked on this problem with a little humanoid. In this work, when the robot falls down, it performs roll motion such that it is in face down position. Then a pre-defined stand up motion is performed for the recovery. In the subsequent work [28], the pre-defined stand up maneuver was expanded to include hand supporting motion on a table.

Kanehiro [33] presented first human sized robot to stand up and lie down. Here, the concept of contact-state graph is developed and used for generating recovery motion. Fig 4-2 shows the contact-state graph where each node represents a contact configuration and each vertex represents control to maneuver from one contact con-

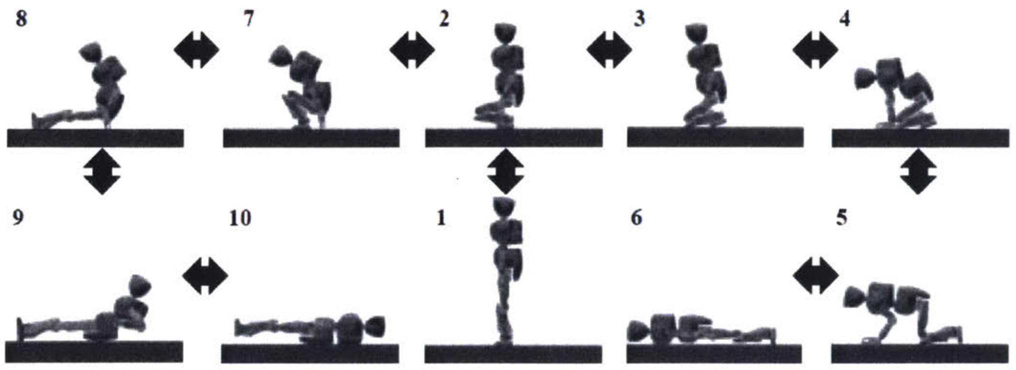


Figure 4-2: Contact-state graph used to generate a recovery motion. A node represents a contact configuration and a vertex represents control to maneuver from one contact configuration to another [33].

figuration to another. The planner is implemented and tested on a HRP robot. The construction of nodes and vertex, however, are done manually.

Stuckler and Behnke [82] presented pre-defined getting-up routine for humanoid robot that is robust to its fall down position. Determining what position the robot is after the fall may not be trivial and in the work, it utilizes stretching out motion to force the robot to be in fully lying down position. Then based on the attitude sensor reading, pre-defined recovery motion is performed depending on whether the robot is in prone or supine position. Similarly, Ishida [30] also worked on planner that works for both prone and supine position relying on predefined motion for standing up.

Terada [84] analyzed 'roll and rise' motion of human sized robot and presented a unique recovery motion based on it. Different from other stand up motions, which are quasi-static, 'roll and rise' motion is dynamic and uses angular momentum gained by swinging leg movement to stand up. Fig. 4-3 shows contact-state graph of the roll and rise motion developed. In the work, angular momentum required to complete stand up is calculated from the anaylsis. Then swinging leg motion is performed to achieve the required angular velocity. Using the motion, the robot was able to roll and standup in simulation and roll and sit-up in the experiment using K1 robot. In the later work [40], the motion was extended to have recovery motion when angular velocity is too high by using hand to push recover.

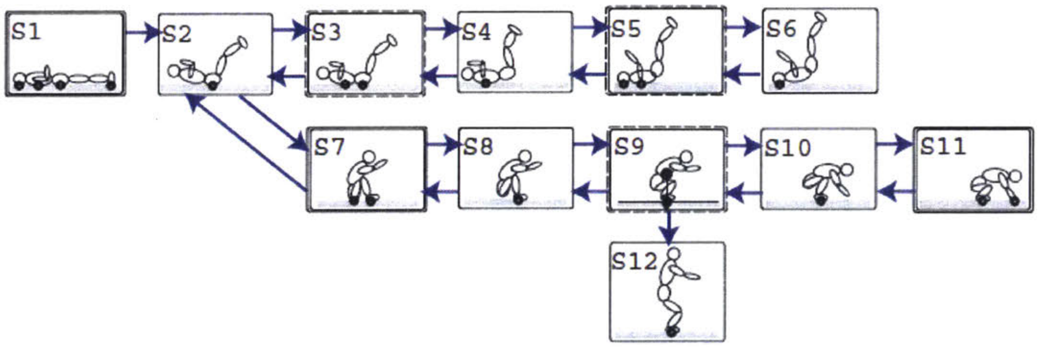


Figure 4-3: Roll and rise motion used to generate a recovery motion. Swinging leg motion is utilized to generated required angular momentum to recover and transitions are defined by a contact-state graph [84].

Several works also exist that take learning based approach. Morimoto [56] proposed a hierarchical reinforcement learning method and applied to 3 link, 2 joint robot for stand up task. Here, upper and lower level is separated to reduce the dimensionality. Upper level learning is implemented with Q-learning while lower level uses a continuous actor-critic method. The robot successfully learned to stand up in 7 out of 10 simulation runs but scalability to higher dof robot is questionable. Mistry [54] presented work where a robot stands up from a chair based on a human demo. Here, the center of mass trajectory of human stand up motion is recorded and robot imitates by following the center of mass trajectory. Similarly, Gonzalez-Fierro [20] has robot stand up motion from sitting on a chair where learning based on human demonstration is used.

As a side note, Kajita [27, 32] developed control motion to reduce landing speed when falling. Safely falling down when the robot loses its balance is important to reduce the potential damage. Several works are also developed in this direction.

The existing recovery planners in the literature share common limitations. First, motions generated from the recovery planner are specific to a particular robot. A recovery motion planner for one robot cannot be directly used to generate recovery motion for another robot with different kinematics and torque limits. Second, the generated recovery motion often consist of sequence of pre-defined motions which are not adaptable. For example, most recovery planners require a robot to be in particular initial configurations from which the recovery action can be taken. If the robot fails to be in one of the these specified configurations, then recovery planner would fail. Similarly, if the torque limits of the robot changes, the planned motion may fail despite kinematics being the same. Last but not least, existing recovery planners are limited to a flat horizontal surfaces. If the surface is slanted or curved, the planner would not work.

4.3 Recovery Motion Planner

The recovery motion planner is developed following the multi-contact motion planning framework using global pseudo-inverse map as described in the Chapter 3. Chapter 3 presented a general multi-contact framework to which several subroutines must be carefully chosen specific to the problem. In the following, we describe the specifics of subroutines used in the recovery planner with justifications for the choices made.

4.3.1 Feasibility Test

The planner requires computation of pseudo-inverse function for each of the contact-states and constantly calls feasibility test in the process. Feasible(q) in Line 6 of Alg.1 is the subroutine used to check feasibility of a given configuration q. In the recovery planner, a given configuration q must satisfy following conditions to be feasible.

$$\begin{aligned}
&G(q) = \tau + \sum_i J_i^T(q) f_i \\
&f_i \in FC_i \\
&|\tau| \le \tau_{\max} \\
&q \in C \backslash C_{obs}^{\circ}
\end{aligned}$$

Here τ is joint torques of the robot, J_i is contact point Jacobian, and f_i is contact forces (reaction force at the contact). The generalized gravity vector is represented by $G(q)$ and FC_i is friction cone at the $i-$th contact point. C is the configuration space with C_obs being obstacles in the configuration space.

The first condition imposes that the configuration to be in static equilibrium where forces and moments are balanced. The second condition check that the reaction forces at the contacts are within the friction cone and the third condition makes sure the joint torques do not violate the torque limitations. Finally, the last condition ensures that the configuration is in the free space of the configuration space and also allows it to be at the surface of obstacles.

Assuming torque limits to be at infinity, the test for static equilibrium above can be simplified into testing whether the center of mass lies within a supporting polygon. The method is presented in [7] describes this procedure and it can be utilized to quickly reject infeasible configurations. Note that for a given configuration, there is multiple pairs of f_i and τ that satisfy the static equilibrium condition. Among the possible pairs, the planner tries to find one requires minimum torque while satisfying other conditions.

4.3.2 Selecting Optimal Configuration

An extremely important step in the computation of pseudo-inverse function is the selection of a configuration q among the possible configurations in the self-manifold. Due to redundancy nature of robots in general, for a given sample w, its inverse kinematics solution $q(w)$ is not unique, but lies within the self-manifold of certain dimension. Which configuration is chosen as the inverse kinematics solution greatly impacts the connectivity of the global pseudo-inverse map and hence the coverage of the planner.

SelectOptimal(w) in Line 8 of Alg. 1 is the routine that selects one configuration among the self-manifold that is optimal with respect to the given problem. In the implementation of recovery planner, the most ideal configuration in the space of self-mainfold is the one that requires minimum torque to maintain its configuration.

The minimum torque configuration is suitable and attractive for the recovery planning problem for several reasons. First, torque is often the limiting factor in performing a recovery motion. Being in the configuration that requires less torque to sustain is favorable as the it decreases the likelihood of subsequent motions from the configuration violating the torque constraints. Second, selecting min torque configuration ensures that the increasing chance of having path-wise resolution along the edges. One of the bottlenecks that prevents edges in the workspace from being path-wise resolvable is the inability to slide due to the friction. For a given sample, if a configuration that minimizes torque is chosen, then the reserved torque (maximum torque minus the torque required to sustain the configuration) is maximized. Higher the reserved torque, higher the friction force that can be overcome and more likely sliding is possible.

Besides the , power consumption also greatly favors selecting minimum torque configuration over others. One of the critical limiting factors in the operation of a legged robot is the power consumption. To contrast, specific cost of transportation ($c_t = (energyused)/(weight * distance)$) for humanoids is at least an order of magnitude higher than that of human [13,41,52,53]. Such high power consumption becomes critical limiting factor, especially when a robot cannot be tethered for its operation. For example, in a disaster rescue scenarios, a prolonged operation without tether is a must which is hampered by inefficient power consumption of the robot. For this, it is highly desirable to generate a trajectory that is power efficient to execute and choosing configurations that minimizes required torque is beneficial.

Computation of Minimum Torque Configuration

For a given sample, computing minimum torque configuration is referred as static multi-contact inverse problem in the robotics literature and is not trivial. Bouyarmane [4] approached static multi-contact inverse problem by formulating it as non-linear optimization problem. Optimization variables are q (configuration) and λ (non-negative coefficient of polyhedral friction cone rays for contact force). With object function chosen to minimize actuator torque, gradient-based optimization is performed but is computationally slow.

Noda [61] introduces planner that minimizes the normalized joint torque and contact forces. In the paper's terminology, normalized joint torque and contact forces is defined as Body Retention Load Vector (BRLV). Important part of the planner is key pose generator that generates static posture with respect to given contacts that minimizes BRLV. It works by iteratively solving through sequence of inverse kinematics that incrementally changes position of the center of mass to minimize BRLV. This approach is quicker than solving non-linear optimization in [4] but is still computationally heavy despite giving suboptimal solution.

Singular Configurations as Approximate Minimum Torque Configuration

The routine SelectOptimal(w) is called frequently throughout the global pseudo-inverse map construction and having an efficient solver is advantageous. However, finding optimal solution requires costly optimization. Instead of running through optimization or series of inverse kinematics as described previously, the planner utilizes singular configurations as an approximate solution to finding minimum torque configuration.

From the analysis performed in Chapter 2, we showed that a motion at or near singularity not only maximally leverages the torque limits to generate forces in quasi-static motions, but is also optimally energy efficient for dynamical motion when it comes to momentum generation. Also, motions exhibited by human or other animals often goes through singular configurations in an effort to minimize effort.

Singular configuration is not always the optimal minimum torque configurations for given sample, but the level of sub-optimality is close. Computation of singular configuration, on the other hand, can be done far more quickly than the latter. In the implementation of the planner, SelectOptimal(w) routine first computes a set of singular configurations and random configurations from the given sample. Then a configuration that minimizes torque is chosen among the configurations in the set. This is found to be efficient yet the resulting configuration sufficiently close to optimal.

4.4 Simulation Result and Discussion

Recovery motion planner is applied to simple 5-link symmetric skeleton robot to validate the working of through simulation.

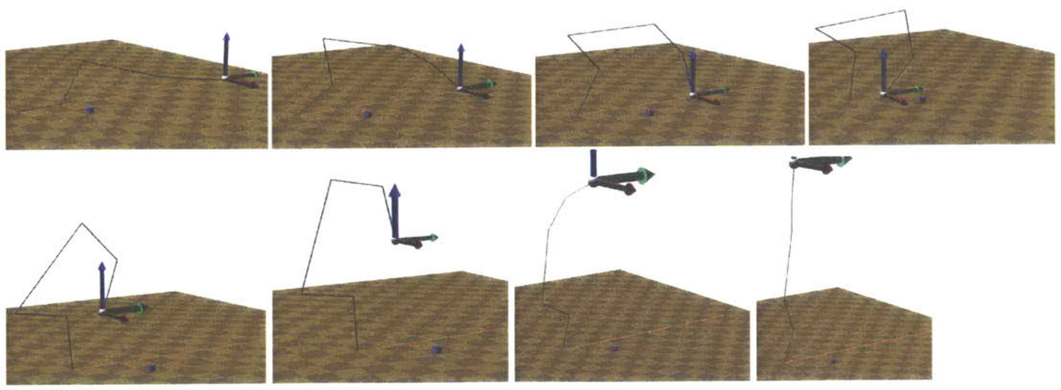


Figure 4-4: The recovery motion planner is applied to 5-link symmetric skeleton robot for validation. Klamp't simualtor is utilized throughout the simulations and one example of recovery motion is shown here.

In the simulation setting, the link ratio of the five links were set to 1:1:2:1:1 and torque limit on each joints were restricted to 100 Nm. For the samples, total of 500 samples were placed at regular interval to cover the workspace. For feasibility test, quasi-static motion was assumed, and checking for the center of mass being inside supporting polygon was used to quickly reject infeasible configurations.

The construction of global pseudo-inverse map required computation of one hour on a personal laptop with single core 2.20GHz processor. The query from the map was completed within seconds and variations in time depending on how far the initial and goal configurations were from nearby configurations in the graph. For the simulator, Klamp't [22] was extensively used for to generate physically accurate simulation involving multiple contacts. Klamp't specializes in providing robust and stable contact forces compare to other robot simulators (e.g., Gazebo) and hence was suitable for the application.

Figures in this section shows several examples of solutions returned from the recovery motion planner. Fig. 4-4 shows one example of a solution where torque limit on each joints were allowed up to 100Nm. In the example shown, the returned solution extensively uses hands to raise itself to the stand up position. The initial torque limit given were sufficient high that such motion is possible.

In the second example, the torque limit on the joints were reduced to one-fifth,

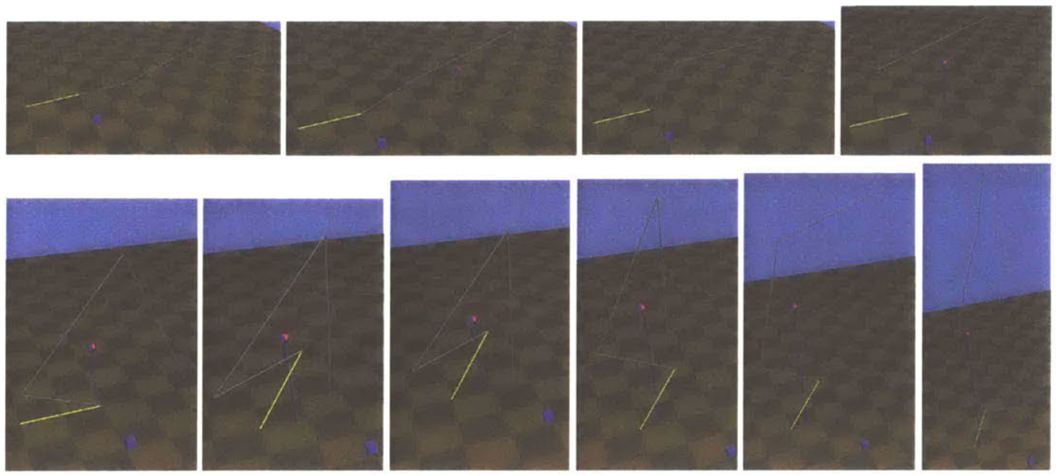


Figure 4-5: An example of solution returned from the recovery motion planner is shown. Here, the torque limits were reduced to one-fifth from the example given in Fig. 4-4 and the resulting motion exhibits robot going through multiple contacts to sustain its own weight.

from 100 Nm to 20 Nm. Fig. 4-5 shows a returned solution based on the changed reduced torque limits. The resulting solution now shows the robot going through several contact states as the torque is limiting. It closely resembles how a person would stand up from fall down position. In the third example, torque limit of the elbow joint was further reduced from 20 Nm to 5 Nm. This was to imitate a situation where robot falls and one or more motors become malfunctioning due to the high impact force from fall down. The returned solution, as demonstrated in Fig. 4-6, shows a motion where action of the elbow is minimized. Note that reduction in torque limits does not require re-computation of the global pseudo-inverse map. From the initially constructed global pseudo-inverse map, a solution can be found for reduced torque limits by pruning the edges in the map that violate the new torque limits prior to the search. As pruning is computationally inexpensive, the motion planner can quickly adapt and generate recovery solutions in the presence motor breakdown due to fall down impacts.

The simulation results show that the recovery motion planner works well in the 2D case. When applying the recovery planner to a high degrees of freedom robot in 3D, several issues need to be addressed for the computation of global pseudo-inverse

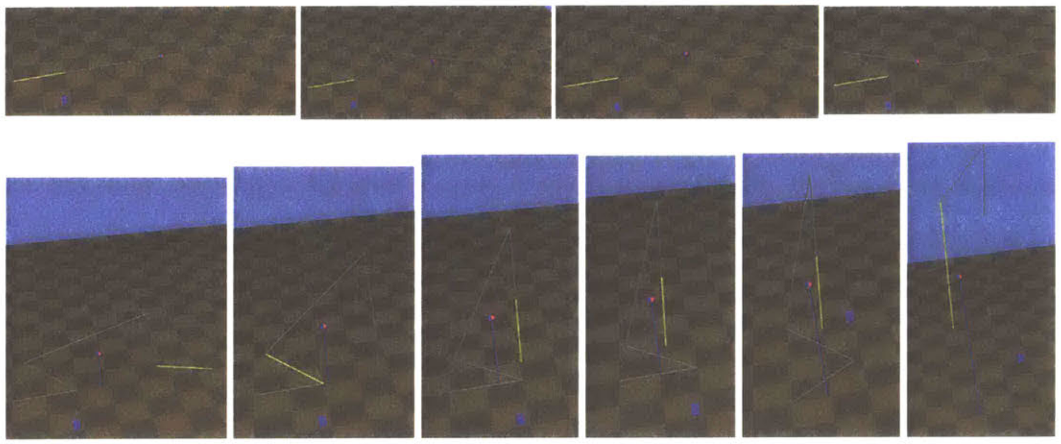


Figure 4-6: A robot can experience breakdown of a motor from fall down due to impacts. The recovery motion planner can quickly adapt to the changes in torque limits caused by a breakdown and generate a plan. Here, a solution is shown where available torque on the elbow joint has been significantly reduced to simulate recovering when a motor at arm fails after the fall down.

map to be tractable.

The first and foremost important issue is the sample selection. Extending from 2D to 3D, samples required to cover the workspace becomes significantly higher. How many sample one has to choose that will sufficiently cover the workspace becomes an non-trivial question. There is a tradeoff between the increasing coverage of the workspace and reducing the computation time required for map construction. Once the number of samples to be placed are chosen, where to place these sample becomes an important consideration. A simple approach is placing the over a regular grid of equal intervals to ensure the even spacing. However, some parts of the workspace region are more interesting than others and placing more sample in those parts are beneficial. An adaptive sampling based on the curvature of the surface could be one of the candidate for distributing the samples.

A robot often exhibit kinematic structure which can be exploited to reduce the computation of the global pseudo-inverse map. First is the symmetry. Often a robot will have several symmetries with respect to its kinematics. One contact state can have identical pseudo-inverse function with another in the presence of symmetry and these redundant computation can be avoided. Also, kinematics of a robot may have several sub-chains which are identical. These common sub-chains can also be used to reduce the computation.

The recovery motion planner in current form is a batch process. A batch of sample are used to construct the map and query is done afterwards which are two separate phases. However, during querying phase, connections made to steer initial and goal configuration can be utilized to update the tree so that it is incremental.

4.5 Conclusion

In this chapter, a recovery motion planner is presented based on the framework developed in the Chapter 3. The recovery planner overcomes several limitations of existing recovery planners. First, the planner is not specific to a particular robot and can be used to generate motion of an arbitrary robot in general when its kinematics is given. Second, it does not rely on pre-defined motion and can generate feasible recovery motion even when a motor break-down. Moreover, the ground surface do not needed to be flat and the planner can generate recovery motions to inclined or even curved surfaces, although global pseudo-inverse map need to be computed for each different surfaces. Simulated results are shown to demonstrate working of the recovery planner and discussion on potential extensions is presented.

Chapter 5

Navigation inside Space Station

Motion planning problem traditionally focuses on generating trajectories that avoid obstacles at all cost. However, obstacles can often be put to an advantage when the external force generated from a contact is utilized. For example, an astronaut inside a space station constantly makes contact with obstacles to navigate around, rather then trying to avoid them. In this chapter, we apply the planning framework discussed in Chapter 3 to solve motion planning problems that is abstracted from astronauts traveling inside a space station. The method works by building a global map which will be refered as *K-step Reachability Map* which consists of a set of trajectories that connect two contact points and k-step number assigned to each trajectory. The set of trajectories is chosen to evenly fill up the state-space through arc-length discretization of the contact surface and k-step number indicates minimum number of contacts require to reach the goal. From the map, multiple queries can be made to generate feasible trajectory from given starting states. It works by connecting the initial state to the trajectories on the map, from which a minimum step trajectory to the goal is easily generated. The method is applied to solve the problem of a robot navigating inside a space station. Discussion of properties of the map, computational complexity and how it can extended be is also presented.

5.1 Introduction

Motion planning has traditionally focused on generating a trajectory that avoids obstacles [45]. This is largely because the motivation for motion planning problems derived from situations where collision with obstacles is highly undesirable. For example, an assembly robot in a car manufacturing factory must plan a motion through a rather complex car body without hitting it. Failure to do so will result in damage to both the car and the manipulator. A quadrotor navigating inside a crowded building must plan a motion that avoids contacting with the environment(e.g. walls, people, furnitures). If a quadrotor hits an object, it can lose stability and in the worst case scenario, fall to the ground and break down. An autonomous car must plan a motion that avoids collision at all cost. If not, accidents can happen which may hurt riders inside.

However, there are situations where contacting with an obstacle or environment is not costly. Moreover, in many cases, contact can actually help a system to maneuver around to reach a goal state which would not have been possible otherwise. For example, an astronaut inside the International Space Station often needs to move from one compartment to another through narrow corridors. Here, planning a motion to avoid any contact whatsoever is not only difficult but unnecessary - in contrary, astronauts actively seek to make contact with and push off the obstacles to navigate through.

Planning a motion that incorporates contact naturally arises from problems of legged locomotion where the foot makes contact with the ground [87]. Footstep planning is essential for this type of locomotion and number of literature addresses it [12,16,39,71]. More recently, planning for the whole-body motion that incorporates multi-contact interactions has gained a great interest and is actively being pursued. Approaches based on operational space formulation [36,70,79,80] and centroidal momentum dynamics [26,46,63,64,86] is gaining traction, while many leverage upon the power of optimization techniques [5,14,47,72]. However, these works are focused on generating a quasi-static motion where at least one contact with the surface, usually

Figure 5-1: Astronauts need to constantly navigate through narrow and crowded modules inside the International Space Station as shown on the left. Rather than seeking to avoid the obstacles, astronauts actively make contact with the obstacles to help them navigate as seen in the right. Image courtesy of NASA retrieved from https://www.nasa.gov/

a horizontal ground, is maintained.

In this work, we apply planning framework in Chap. 3 to present a method that is suitable for motions where flight-phase is dominant without being periodic (cf. a robot running [69] or hopping [74] is dominated by flight-phase but periodic in nature). An example would be an astronaut navigating inside the space station or a parkour practitioner jumping between walls to maneuver around the buildings. These motion are characterized by the fact that the robot has no ability to change its centroidal momentum during flight. Only when in contact, the internal joint torques of the robot can be translated to external forces affecting the motion, which is then propagated through the flight-phase till the next contact is reached. Here, obstacles must be utilized rather than avoided, viewed as an opportunity to change the course of the motion through contact.

The proposed planning method is based on the global map and consists of two phases: map construction and query phase. In the map building phase, a gloabl map (referred as *K-step Reachability Map* from onwards) is created which first chooses a set of trajectories that connect two contact points and evenly fill up the state-space based on the arc-length discretization. Then k-step number is sequentially assigned to each chosen feasible trajectory by propagating backwards from the goal. This k-step number equals the minimum number of contacts required to reach the goal

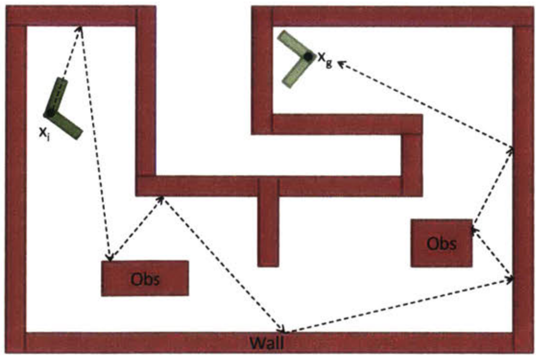


Figure 5-2: Navigation inside the Space Station Problem. A robot has to travel from x_i to x_g inside the space station by making sequence of contacts. While in contact, the robot can change its velocity by applying appropriate joint torques. Nevertheless, it has no ability to change the velocity in mid-flight.

state from the current trajectory. In the query phase, a feasible plan is generated by first connecting a given initial state to the map and then concatenating successive trajectories that each reduces the k-step number by one.

The result is a multi-query planner that, once the map is constructed, very quickly gives a minimum step plan from any initial state. The methodology is motivated from and applied to the problem of an astronaut navigating around the International Space Station, but can be extend to solve more general cases.

5.2 Problem Formulation

This section formulates the problem to be addressed in the chapter. It is motivated from how astronauts move around the clustered environment inside the International Space Station (ISS) by pushing off the obstacles. As shown in Fig. 5-2, the problem is to plan a motion for a robot inside the ISS to navigate from the current state to a desired goal state. Here, planned motion is allowed to make contact with the obstacle. In fact, collision-free motion is not possible as the robot is unable to change its velocity in the mid-flight. Rather, a sequence of contacts with obstacles must be made and utilized to reach the goal state.

To formally state the motion planning problem with contact in general, consider a system with dynamics given by $\dot{x}(t) = f(x(t), u(t))$, where x is the state, u is the control. Let X denote state space, $X_{obs} \subset X$ obstacle set, with given initial state $x(0) = x_{init}$, a goal set $X_{goal} \subset X$, and constraints on control input $u_{\min} \leq u \leq u_{\max}$. The problem is to find, if it exists, a control input $u(t)$ that results in a feasible path $x(t) \in X \backslash X^o_{obs}$ for $\forall t \in [0, t_f]$ such that the system reaches the goal state $x(t_f) \in X_{goal}$ from the initial state $x(0) = x_{init}$ while satisfying the dynamics. Note that the feasible path $x(t)$ is allowed to make contact with the obstacles s.t. $x(t_c) \in \overline{X}_{obs} \backslash X^o_{obs}$ for some $t_c \in [0, t_f]$ and hence not strictly collision-free.

In this chapter, we consider problem described in Fig. 5-2 which will be referred as *Navigation inside the Space Station Problem* henceforth. Here, the kinematics of a robot is not explicitly encoded and only the robot's center of mass motion is considered for simplicity. The workspace is $W \subset \mathbb{R}^2$ and the state $x(t)$ is position and velocity of the robot's center of the mass. The dynamics of the system is given as $\dot{x}(t) = const$ while $x(t) \in X \backslash X_{obs}$ and the state $x(t)$ undergoes impulsive change upon contact with the obstacle, described by the impact map $x(t^+) = \Delta(x(t^-), u)$.

The impact map can be arbitrary, but we specifically look at a perfectly elastic collision that obeys the law of reflection when $u = 0$. It is assumed that the control input u during contact changes the direction of the velocity but not its magnitude such that the kinetic energy before and after the collision is conserved. To what extent direction of the velocity can be altered from that prescribed by the law of reflection depends on the property of the contacting surface, configuration of the robot at contact, and constraints on the control input u. For simplicity, it is assumed that the maximum angle of deviation θ_{max} that can be induced by the control input u, is given for each contacting point along the surface.

5.3 K-step Reachability Map

This section presents map construction phase of the proposed motion planning method. First, discretization of the contact surface is explained from which a set of trajectories that belong to the map is generated. Then a recursive algorithm that assigns k-step number to each trajectories is described. The key properties of the K-step Reachability Map as well as its computational complexity is discussed.

5.3.1 Discretization of the Surface and Trajectory

It is highly desirable for the set of trajectories on the map to evenly fill up the state space. To this end, K-step Reachability Map generates trajectories based on the discretization of the surface in the following manner. First, it will be assumed that the contacting surface is a piecewise simple and closed regular curve $\alpha : I \subset \mathbb{R}^1 \to \mathbb{R}^2$ that is parametrized by arc-length s. Then for each $s \in I$, $\alpha(s)$ uniquely determines a point on the contact surface and vice-versa. The direction of parametrization is chosen such that signed normal vector $n(s)$ always points towards $X_{free} = X \backslash X_{obs}$. This makes the curve negatively oriented for the obstacles.

The contact surface is discretized at a regular arc-length interval ds, which is a parameter that determines the resolution of the planner. Let $s(i) \triangleq i \bullet ds$, then the set of discretized points S along the contact surface becomes

$$S = \{\alpha(s(i)) | i \in [1, N], i \in \mathbb{Z}\}$$

where $N = \lceil l/ds \rceil$ is the total number of the discretized point along the surface with l being the total arc-length of the contact surface. The vector $\alpha(s(i))$ represents coordinate of i-th discretized point along the contact surface when i is an integer taking a value from 1 to N.

From the set S, a set of trajectories T is generated as follow. Let r_{ij} be a trajectory that connects two points on the contact surface, starting at $\alpha(s(i))$ and ending at $\alpha(s(j))$ with $i, j \in \mathbb{R}$. For the navigation inside the spacee station problem, r_{ij} is unique and it is the ray from $\alpha(s(i))$ to $\alpha(s(j))$. Now, T is defined as

$$T = \{r_{ij} | \alpha(s(i)), \alpha(s(j)) \in S, i \neq j, r_{ij} \in X_{free}\}$$

The set T contains all collision-free trajectories that connect two distinct dis-

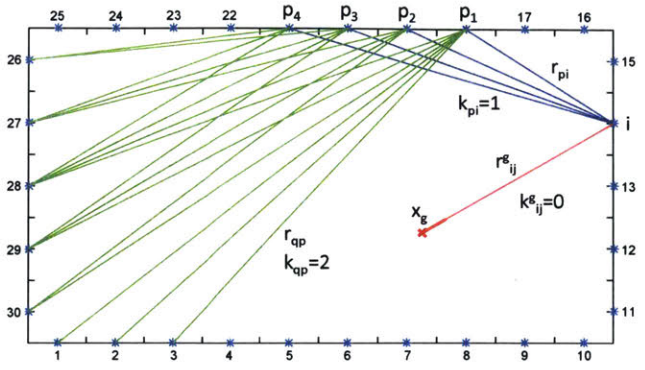


Figure 5-3: K-step Reachability Map assigns a number k_{ij} to each trajectory $r_{ij} \in T$, which equals the minimum number of contacts required to reach the goal from the current trajectory. The assignment is done recursively, starting from the trajectory r^g_{ij} that enters the goal and propagating backwards as described in Alg. 3. The above diagram shows two-step propagation from the goal state. In this example, the contact surface is uniformly discretized and the impact map follows the law of reflection with control input that can deviate it by maximum angle of $\pm 15°$.

cretized points along the contact surface, and forms trajectories of the map. It is a finite subset in the space of trajectories with cardinality in the order $O(N^2)$, $|T| \leq N^2$ to be more precise. Moreover, the trajectories evenly fill up the state-space due to its construction, although more precise quantification is needed.

5.3.2 Assignment of the K-step Number

K-step Reachability Map, $K : r_{ij} \in T \to \mathbb{Z}$, assigns k-step number to each trajectory r_{ij} in the set T. The k-step number, denoted k_{ij}, is defined as the minimum number of contacts required to reach the goal state from the trajectory r_{ij}. The assignment of the k-step number is done recursively propagating from the trajectory r^g_{ij} that directly reaches the goal state (Alg 3).

In the algorithm, Q_k is a queue that contains all the trajectories that is assigned k-step number equal to k. Initially, $k = 0$ with Q_0 having only one trajectory r^g_{ij}, and k is increased sequentially to generate Q_k. It is continued till Q_k becomes an empty set. The final value of $k = k_{max}$ indicates the following: all the trajectories in the set T that can reach the goal through concatenation of trajectories within the set T, requires at most k_{max} number of contacts.

Algorithm 3 : $k_{ij} \leftarrow$ KstepReachabilityMap(T, r^g_{ij})

```
1: $k_{ij} \leftarrow \infty$
2: $Q \leftarrow r^g_{ij}$
3: $k^g_{ij} \leftarrow 0$
4: assigned$[r^g_{ij}] \leftarrow$ *true*
5: **while** $Q$ is not empty **do**
6:     $r_{ij} \leftarrow Q.pop()$
7:     **for all** $r_{pi} \in T$ **do**
8:         **if** $\neg$ assigned$[r_{pi}]$ and $r_{pi} \rightarrow r_{ij}$ reachable **then**
9:             $Q.push(r_{pi})$
10:            $k_{pi} \leftarrow k_{ij} + 1$
11:            assigned$[r_{pi}] \leftarrow$ *true*
12: **return** $k_{ij}$
```

The most computationally expensive part of the algorithm is determining whether the trajectory r_{ij} is reachable from r_{pi} (line 10). To be precise on the terminology used here, the trajectory r_{ij} is reachable from r_{pi} if $\exists u$ satisfying $r_{ij}(t_i) = \Delta(r_{pi}(t_f), u)$ such that the final state of the trajectory $r_{pi}(t_f)$ together with control input u produces the initial state of the trajectory $r_{ij}(t_i)$ through the impact map. Depending on the structure of the impact map, finding the existence of a control input given two trajectories r_{pi} and r_{ij} can become expensive.

For the navigation inside the space station problem, checking reachability of two trajectories r_{pi} and r_{ij} is a simple computation. Since the impact map follows the law of reflection in the absence of a control input, the velocity v^0_{ij} of the trajectory r_{ij} can be expressed in terms of velocity v_{pi} of the trajectory r_{pi} and normal vector $n(i)$ at the contact surface

$$v^0_{ij} = \Delta(v_{pi}, 0) = v_{pi} + 2(v_{pi} \bullet n(i))n(i)$$

Now from the problem formulation, the control input u is assumed to deviate the direction with maximum angle of $\theta_{max,i}$. The reachability then reduces to checking whether the angle between v^0_{ij} and v_{ij} is less than $\theta_{max,i}$.

5.3.3 Properties and Computational Cost

The K-step Reachability Map has following key properties that enables it to act as a roadmap. These properties will be utilized to generate a feasible plan from a given starting state in the query phase.

The first property is about connectivity among the trajectories on the map. For any trajectory $^{\forall}r_{pi} \in T$ with k-step number $k_{pi} = k$, there exist at least one trajectory $r_{ij} \in T$ that is reachable from r_{pi} with k-step number $k_{ij} = k - 1$. This property comes directly from the map construction (Alg.3). If there exists a trajectory $r_{pi} \in T$ that does not have a reachable trajectory $r_{ij} \in T$ with $k_{ij} = k_{pi} - 1$, then it directly contradicts the algorithm (Alg.3, line 6,7,10,11).

This property ensures that every trajectory in the map with k-step number assigned, can be connected to the goal by concatenating number of trajectories in the map equal to its k-step number. Note that there may be trajectories in the map whose k-step number has not been assigned during the map construction (Alg.3, line 8) and remain at infinity. Unassigned k-step number $k_{ij} = \infty$ gives a weak certificate of the reachability of trajectories r_{ij} to the goal. The set $\mathrm{T_{inf}} = \{r_{ij} \in T | \, k_{ij} = \infty\}$ defines a subset of trajectories on the map that cannot reach the goal through trajectories on the map. It is a weak certificate of the reachability in the sense that a trajectory $r_{ij} \in T_{inf}$ can still reach the goal travelling through trajectories outside the map, but cannot do so only with trajectories in the map.

Proposition 3. *For any trajectory $^{\forall}r_{pi} \in T$ with k-step number $k_{pi} = k$, there exist at least one trajectory $r_{ij} \in T$ that is reachable from r_{pi} with k-step number $k_{ij} = k - 1$*

The second property is about assignment of k-step number to trajectories that are outside the map but ends on one of the discretized contact surface points. Let $r_{ij} \notin T$ be a trajectory not on the map but ends on one of the discretized surface point s.t $\alpha(s(j)) \in S$. If its neighboring trajectories $r_{\lceil i \rceil j}, r_{\lfloor i \rfloor j} \in T$, then $\min[k_{\lceil i \rceil j}, k_{\lfloor i \rfloor j}] \leq k_{ij} \leq \max[k_{\lceil i \rceil j}, k_{\lfloor i \rfloor j}]$ and moreover, if $k_{\lceil i \rceil j} = k_{\lfloor i \rfloor j}$ then $k_{ij} = k_{\lceil i \rceil j} = k_{\lfloor i \rfloor j}$

This property is conjectured to be true when the following conditions are met:

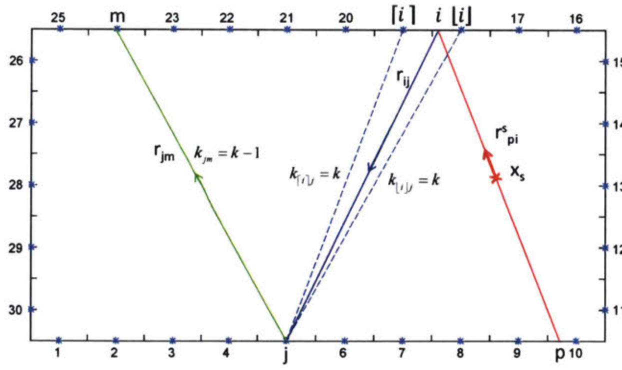


Figure 5-4: To generate a feasible plan from the K-step Reachability Map, a given initial state x_s must first be connected to the map. Connection to the map is executed in two stages. First, the starting trajectory r^s_{pi} which contains the initial state x_s is steered to the trajectory r_{ij} such that $j \in \mathbb{Z}$. This makes the trajectory r_{ij} to land on one of the discretized surface points. The trajectory r_{ij} is then steered to one of the trajectories $r_{jm} \in T$ in the map to complete the connection. Among the all possible pairs of trajectories r_{ij} and r_{jm}, ones that minimize the k-step number is chosen using the properties of the map.

1) curve $\alpha(s(n)), n \in [\lfloor i \rfloor, \lceil i \rceil]$ is simple and regular

2) region enclosed by two trajectory $r_{\lceil i \rceil j}, r_{\lfloor i \rfloor j}$ and $\alpha(s(n)), n \in [\lfloor i \rfloor, \lceil i \rceil]$ contains no obstacle

3) mapping from control input u to space of trajectories is smooth.

These conditions generally hold when the trajectory $r_{ij} \notin T$ is reasonably close to its neighboring trajectories $r_{\lceil i \rceil j}, r_{\lfloor i \rfloor j} \in T$.

Therefore, when the set of trajectories in the map sufficiently fills up the state-space, the above property makes it possible to assign a k-step number to trajectories outside the map. This allows an arbitrary trajectory to be connected to the map through appropriate steering.

The computational complexity of constructing the map is $O(N^3)$ where $N = \lceil l/ds \rceil$ is the total number of the discretized point along the surface as defined previously. This can be shown as follow. The set of trajectories T in the map has cardinality $|T| \leq N^2$. Now, during the construction, each trajectory r_{ij} enters the queue (Alg.3, line 11) at most once. For each trajectory in the queue, reachability check (Alg.3, line 10) is performed at most N times, equalling N when all trajectories r_{pi} are in the map for p from 1 to N. Therefore, the reachability check which dominates the computational cost in the algorithm is executed at most N^3 times.

Proposition 4. *Let $r_{ij} \notin T$ be a trajectory not on the map but ends on one of the discretized surface point. If its neighboring trajectories $r_{\lceil i \rceil j}, r_{\lfloor i \rfloor j} \in T$, then $\min[k_{\lceil i \rceil j}, k_{\lfloor i \rfloor j}] \leq k_{ij} \leq \max[k_{\lceil i \rceil j}, k_{\lfloor i \rfloor j}]$ under the smoothness assumptions. Moreover, if $k_{\lceil i \rceil j} = k_{\lfloor i \rfloor j}$ then $k_{ij} = k_{\lceil i \rceil j} = k_{\lfloor i \rfloor j}$*

5.4 Generating a Plan from the Map

This section describes the query phase of the proposed motion planning method. When a query is made by giving an initial state, a feasible plan to the goal can be quickly generated using the constructed map. First, how the given initial state is connected to the map is explained. Then, generation of a feasible plan once the trajectory enters the map is detailed.

5.4.1 Connection to the Map

To generate a feasible plan, it is necessary to connect the given initial state x_s to the constructed map. This is done by steering the initial trajectory that contains x_s towards a trajectory in the map. Since the control input u only acts to change dynamics when in contact, a trajectory that contains the initial state x_s is unique and we denote it by r^s_{pi} (Alg.4, line 1).

The initial trajectory r^s_{pi} lands on the surface at $\alpha(s(i))$ which is not necessary in the set S of the discretized contact surface points ($i \notin \mathbb{Z}$) as shown in Fig. 5-4. The first step of steering is to apply control input u at the initial contact point $\alpha(s(i))$ such that the ensuing trajectory r_{ij} lands on one of the discretized surface points ($j \in \mathbb{Z}$). Now, there may be more than one value of $j \in \mathbb{Z}$ for which the trajectory r_{ij} is reachable from r^s_{pi} (Alg.4, line 5). Among these trajectories, r^s_{pi} is steered to the one that minimizes the k-step number. This is done by assuming k-step number of the trajectory r_{ij} to be equal to the max k-step number of its two neighboring trajectories

$r_{\lceil i \rceil j}$ and $r_{\lfloor i \rfloor j}$ (Alg.4, line 6) which is a consequence of the second property of the map (Section 5.3.3). Note that in the implementation, Concatenate(P, r_{ij}) function concatenates the trajectory r_{ij} to the tail of plan P.

After the trajectory r^s_{pi} is steered to r_{ij} such that $j \in \mathbb{Z}$, it now remains to steer the trajectory r_{ij} to one of the trajectories $r_{jm} \in T$ in the map. Among the trajectories $r_{jm} \in T$ that can be reached from r_{ij}, one that has minimum k-step number is chosen as the destination. The connection to the map is complete as the trajectories $r^s_{pi} \rightarrow r_{ij} \rightarrow r_{jm}$ steers the given initial state x_s onto the map with $r_{jm} \in T$. This two-step execution is done in a way that minimize the k-step number of the last trajectory r_{jm} that enters the map.

5.4.2 Minimum Step Plan Generation

Once the given initial state x_s is steered to a trajectory in the map, a plan to the goal can be generated very quickly as a consequence of the first property of the map (Section 5.3.3). The connectivity guarantees that a trajectory $r_{pi} \in T$ with k-step

Algorithm 4 : $P \leftarrow$ ConnectToMap(T, k_{ij}, x_s)

```
1: $r^s_{pi} \leftarrow$InitialTrajectory($x_s$)
2: $P \leftarrow$InitializePlan($r^s_{pi}$)
3: $k_{min \leftarrow \infty}$
4: for $j' = 1$ to N do
5:     if $r_{ij'} \in X_{free}$ and $r^s_{pi} \rightarrow r_{ij'}$ reachable then
6:         $k' = \max[k_{\lceil i \rceil j'}, k_{\lfloor i \rfloor j'}]$
7:         if $k' < k_{min}$ then
8:             $k_{min} \leftarrow k'$
9:             $j \leftarrow j'$
10: $P \leftarrow$ Concatenate($P, r_{ij}$)
11: $k_{min \leftarrow \infty}$
12: for $m' = 1$ to N do
13:     if $r_{jm'} \in T$ and $r_{ij} \rightarrow r_{jm'}$ reachable then
14:         if $k_{jm'} < k_{min}$ then
15:             $k_{min} \leftarrow k_{jm'}$
16:             $m \leftarrow m'$
17: $P \leftarrow$ Concatenate($P, r_{jm}$)
18: return $P$
```

Algorithm 5 : $P \leftarrow \text{GeneratePlan}(T, k_{ij}, x_s)$

```
1: $P \leftarrow \text{ConnectToMap}(T, k_{ij}, x_s)$
2: $r_{pi} \leftarrow \text{LastTrajectory}(P)$
3: while $k_{pi} > 0$ do
4:     for all $r_{ij} \in T$ s.t. $k_{ij} = k_{pi} - 1$ do
5:         if $r_{pi} \rightarrow r_{ij}$ reachable then
6:             $P \leftarrow \text{Concatenate}(P, r_{ij})$
7:             Terminate For Loop;
8:     $r_{pi} \leftarrow \text{LastTrajectory}(P)$
9: return $P$
```

number k_{pi} always has at least one trajectory $r_{ij} \in T$ that can be reached with a reduced k-step number $k_{ij} = k_{ij} - 1$.

The planner follows through the trajectories in the map in a greedy manner, each time steering to a trajectory with k-step number one less than the current (Alg.5, line 4-6). The result is a feasible plan that reaches the goal from a given initial state x_s while making minimum number of contacts in the process. In the algorithm, LastTrajectory(P) function returns the tail trajectory that has been concatenated lastly. In the implementation, a trajectory is chosen arbitrarily among the trajectories r_{ij} that have same k-step number. When desirable, a heuristic (e.g. min length) can be introduced to break a tie between trajectories r_{ij} with same k-step number to improve the quality of the returned plan with respect to a designed cost.

The computation time of generating a plan from a given initial state to the goal is $O(N)$. Hence, once the map is constructed (which requires $O(N^3)$ computation), a feasible plan can be generated efficiently and multiple queries can be answered quickly.

5.5 Simulation Result

The proposed motion planning algorithm is applied to the problem of navigating inside the space station as described in Section 5.2. In this example, the space station is a rectangular room with a circular obstacle in the center and a robot has to navigate around by making sequence of contacts. The impact map follows the law of reflection

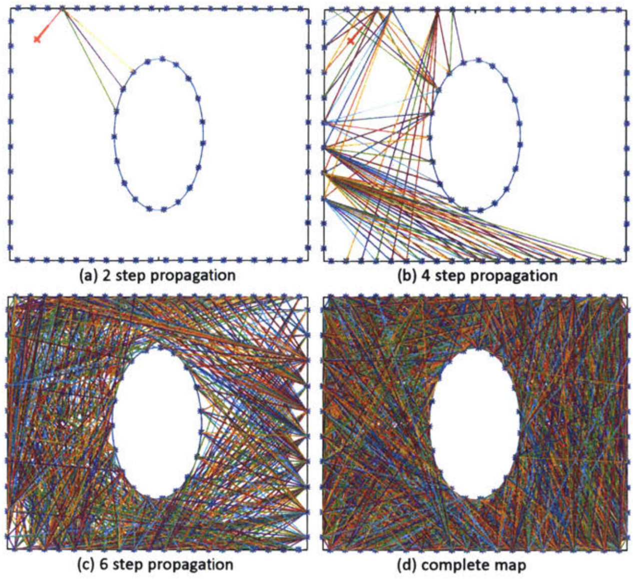


Figure 5-5: The map construction phase of the proposed motion planning algorithm is shown graphically. Starting from the goal state, trajectories are propagated backwards sequentially to build K-step Reachability Map. As seen, large part of the state-space is quickly filled as the number of steps increase. In this example, a robot is inside a rectangular room with a circular obstacle at the center and the maximum angle that can be deviated by the control input u is set to $\pm 15\,\text{deg}$

with the robot having ability to deviate the angle by maximum of $\pm 15\,\text{deg}$ while in contact through controlling its joint torques. The simulation was performed using MATLAB on Core i5-2430 (2.40GHz) workstation.

The planner first builds K-step Reachability Map in the construction phase as shown in Fig. 5-5. The resolution of the constructed map depends on the parameter ds which is the regular arc-length interval that the contact surface is discretized. Averaging over 100 trials, computation times of building the map were 0.22s, 0.84s, 3.89s for resolution level $ds = 1.00$, 0.50, and 0.25 respectively. The Fig. 5-5 shows the case when $ds = 0.5$ and here, the total arc-length of the environment(rectangular room and circular obstacle) is $l = 39.42$ making $N = \lceil l/ds \rceil = 79$.

From the constructed K-step Reachability Map, the planner can answer multiple queries to generate a feasible minimum step trajectory from a given initial state to the goal. Fig. 5-6 shows trajectories generated by the planner for different initial states. The plan generation is done very quickly once the map is built - the computation

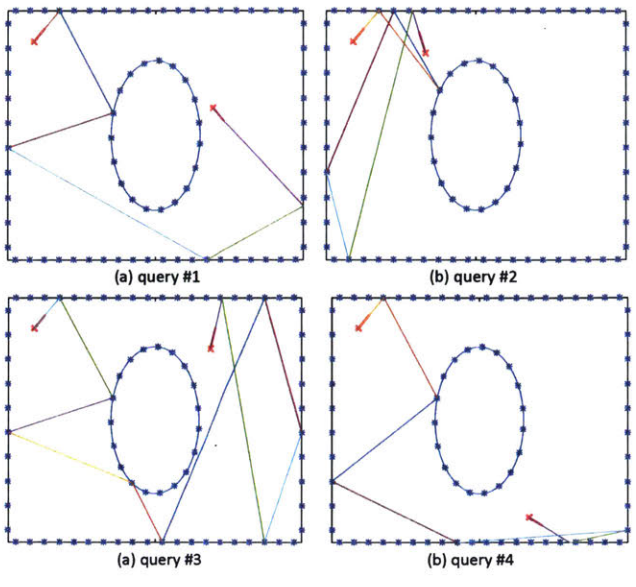


Figure 5-6: Multiple queries can be made on the map to generate a minimum step feasible trajectory from different initial states. Here, four queries were answered by the planner using the map constructed in Fig. 5-5. Based on the key properties of the map, a feasible trajectory that steers a given initial state to the goal in the minimum steps can be very efficiently planned.

time to generate a plan was 0.05s averaged over 100 trials.

5.6 Discussion on Extension

The described motion planning method focuses on generating a feasible plan that makes minimum number of contacts to reach the goal. However, in certain situations, a cost function other than the number of steps may be more desirable. For example, one may be more interested in a plan that minimizes the length or the control input. The planner can be generalized towards this end by making few changes in building the map. Dynamic programming algorithm can be applied in the construction phase to additionally compute cost-to-go value for each trajectory recursively. This will allow the planner to generated optimal plan with respect to the trajectories in the map.

A more important generalization, from the author's perspective, is to make the planner handle changes in the magnitude of the velocity. In the chapter, the problem assumed the magnitude of the velocity to be constant, while allowing only the directional change. This is a reasonable assumption for the cases when there is no gravity acting (e.g. a robot navigating inside the space station), since a velocity profile can be easily imposed onto the planned path. However, in the presence of the gravity, the planner must handle both changes in the magnitude and direction to generate a feasible trajectory.

To handle the change in the magnitude of the velocity, the set of contact surface points S can be expanded to include magnitude of the velocity as an additional dimension. This will introduce significant increase in the computation time of the map construction depending on the level of discretization of the speed. But once the map is constructed, plan generation will remain computationally inexpensive as the properties of the map still hold and can be leveraged.

5.7 Conclusion

In this chapter, a motion planning method based on the framework developed in Chapter 3 has been proposed that is suitable for motions where flight-phase is dominant. The planner explicitly deals with and utilizes contact with the obstacles to generate a feasible plan. The key component of the method is in building the K-step Reachability Map, which recursively assigns k-step number to the selected set of trajectories. The set of trajectories is chosen that evenly fills up the state-space based on the discretization of the contact surface. The k-step number assigned to each trajectory in the map indicates the minimum number of contacts required to reach the goal from the trajectory.

The proposed planning method efficiently generates a feasible minimum step plan from the K-step Reachability Map by leveraging on its properties. Multiple queries can be answered on the map to give trajectories leading to the goal for different initial states. The planner is applied to solve the problem of navigating inside the space station. It can also be extended to solve more general problems.

Chapter 6

Conclusion

This thesis presented a framework for motion planner that utilizes multiple contacts with the environment and its objects. In the robot motion planning problems, environment and its objects are often treated as obstacles to be avoided. However, there are situations where contacting with the environment is not costly. Moreover, in many cases, making contact can actually help a robot to maneuver around to reach a goal state which would not have been possible otherwise. The framework presents a planner that autonomously generate motions, where robot has to make multiple contact with different part of its body in order to achieve a task objective.

The essence of the framework is a fast multi-contact planner that utilizes pre-computed global pseudo-inverse map. Existing multi-contact planner require good initial seeds for it to successfully generate motion. These are hard to find and often manually encoded. In addition, current multi-contact planners are single-query and often slow. Here, we leverage upon a pre-computed global pseudo-inverse map to generate multi-contact motion from current configuration to the goal without need for an initial seed. In Chapter 3, details of how the pseudo-inverse map is constructed is presented. The pseudo-inverse map is an inverse kinematics map for each contact-state that has a property of global resolution, connected by connectivity functions.

To make the construction process more computationally tractable, we leveraged singular configurations to produce torque efficient motions. In Chapter 2, singularity is presented as a search heuristic with analysis of force and momentum generated through contact in relation to the singular positions. Although global pseudo-inverse map is computationally expensive, once computed, it can be used to generate plans fast, possibly online, in a multi-query manner. Its application is presented in Chapter 4 and 5. In Chapter 4, the framework is applied to give a multi-contact motion planner that plans recovery motion from fall down for humanoids. In Chapter 5, a motion planner that solves actuated mathematical billiard problem motivated by an astronaut navigating inside the International Space Station is presented.

6.1 Extension and Improvements

There are several aspects of the presented framework that can be extended as well as improved. The planning framework, in its current form, is batch process in nature. From the samples generated, a global pseudo-inverse map is constructed and then queries are made on this pre-constructed global map. Construction and query phase are separated and do not affect each other. However, when a query is made, computation done to steer a given sample to the map can be utilized to expand the global map. The framework can be extended to be incremental, where map is incrementally built and queries can be part of the building process. This will make the planner asymptotically complete and failure less likely.

The performance of the presented planner largely depends on whether the samples used to construct the global pseudo-inverse map sufficiently covers the landscape of the global structure. Having a large number of sample definitely helps but comes at the cost of computation. Thus, from given computational resources, number of samples will be bounded and this poses questions of how do we choose 'good' samples to have the needed coverage. In the implementation of framework, we have used regular grids to select the samples. This approach is simple and robust, but may unnecessarily waste sample in an uninteresting flat ground. A more adaptive sampling strategy will be needed to improve the performance of the planner in increasing the success rate of the returned plan. An idea toward such strategy will be incorporating curvature information and utilizing homotopy in the selection of the samples.

The framework generates a global pseudo-inverse map based on the torque limits of the robot. However, when those torque limits change, the global map has to be recomputed from scratch even if robot's kinematics is not altered. Being able to adapt to changes in torque limit can be useful when there is possibility of actuators malfunctioning during an operation. For example, when robot hits or falls with high impact, it may break down some of the motors and thereby changing maximum torque that can be produced. The construction of global pseudo-inverse map can be extended to incorporate these torque limit changes. This can be achieved simply by setting the torque limit to infinity during map construction and encoded the maximum torque required for each of the edges and vertices. Then the map can be truncated based on the given torque limit during the query process. Having no torque limit during construction will significantly increase the size of the map as torque feasibility condition accounts for rejection of large portion of edges and vertices. But once computed, the planner will be able to generated motion robust to changes in the actuator torque which can occur through hard contacts.

Bibliography

[1] Antonio Bicchi, Marco Gabiccini, and Marco Santello. Modelling natural and artificial hands with synergies. *Phil. Trans. R. Soc. B*, 366(1581):3153–3161, 2011.

[2] James E Bobrow, B Martin, G Sohl, EC Wang, Frank C Park, and Junggon Kim. Optimal robot motions for physical criteria. *Journal of Robotic systems*, 18(12):785–795, 2001.

[3] Oriol Bohigas, Michael E Henderson, Lluís Ros, Montserrat Manubens, and Josep M Porta. Planning singularity-free paths on closed-chain manipulators. *IEEE Transactions on Robotics*, 29(4):888–898, 2013.

[4] Karim Bouyarmane and Abderrahmane Kheddar. Static multi-contact inverse problem for multiple humanoid robots and manipulated objects. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 8–13. IEEE, 2010.

[5] Karim Bouyarmane and Abderrahmane Kheddar. Multi-contact stances planning for multiple agents. In *IEEE International Conference on Robotics and Automation*, 2011.

[6] Karim Bouyarmane and Abderrahmane Kheddar. Humanoid robot locomotion and manipulation step planning. *Advanced Robotics*, 26(10):1099–1126, 2012.

[7] Tim Bretl, Jean-Claude Latombe, and Stephen Rock. Toward autonomous free-climbing robots. In *Robotics Research. The Eleventh International Symposium*, pages 6–15. Springer, 2005.

[8] Katie Byl, Marten Byl, and Brian Satzinger. Algorithmic optimization of inverse kinematics tables for high degree-of-freedom limbs. In *ASME 2014 Dynamic Systems and Control Conference*, pages V001T04A005–V001T04A005. American Society of Mechanical Engineers, 2014.

[9] John Canny. *The complexity of robot motion planning.* MIT press, 1988.

[10] John Canny. Computing roadmaps of general semi-algebraic sets. *The Computer Journal*, 36(5):504–514, 1993.

[11] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(1):485–524, 1991.

[12] Joel Chestnutt, Manfred Lau, German Cheung, James Kuffner, Jessica Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *IEEE International Conference on Robotics and Automation*, 2005.

[13] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[14] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *IEEE-RAS International Conference on Humanoid Robots*, 2014.

[15] Anjan Kumar Dash, I-Ming Chen, Song Huat Yeo, and Guilin Yang. Singularity-free path planning of parallel manipulators using clustering algorithm and line geometry. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 761–766. IEEE, 2003.

[16] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *IEEE-RAS International Conference on Humanoid Robots*, 2014.

[17] Adrien Escande, Abderrahmane Kheddar, and Sylvain Miossec. Planning support contact-points for humanoid robots and experiments on hrp-2. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2974–2979. IEEE, 2006.

[18] Adrien Escande, Abderrahmane Kheddar, and Sylvain Miossec. Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 61(5):428–442, 2013.

[19] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE transactions on robotics*, 21(6):1077–1091, 2005.

[20] Miguel González-Fierro, Carlos Balaguer, Nicola Swann, and Thrishantha Nanayakkara. A humanoid robot standing up through learning from demonstration using a multimodal reward function. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 74–79. IEEE, 2013.

[21] Clement Gosselin and Jorge Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE transactions on robotics and automation*, 6(3):281–290, 1990.

[22] Kris Hauser. Robust contact generation for robot simulation with unstructured meshes. In *Robotics Research*, pages 357–373. Springer, 2016.

[23] Kris Hauser. Continuous pseudoinversion of a multivariate function: Application to global redundancy resolution. 2017.

[24] Kris Hauser, Timothy Bretl, Kensuke Harada, and Jean-Claude Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Algorithmic foundation of robotics VII*, pages 507–522. Springer, 2008.

[25] Kris K Hauser, Timothy Bretl, and Jean-Claude Latombe. Non-gaited humanoid locomotion planning. In *Humanoids*, pages 7–12, 2005.

[26] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, pages 1–19, 2015.

[27] Hirohisa Hirukawa, Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, and Takakatsu Isozumi. The human-size humanoid robot that can walk, lie down and get up. *The International Journal of Robotics Research*, 24(9):755–769, 2005.

[28] Masayuki Inaba, Takashi Igarashi, Satoshi Kagami, and Hirochika Inoue. A 35 dof humanoid that can coordinate arms and legs in standing up, reaching and grasping an object. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 1, pages 29–36. IEEE, 1996.

[29] Masayuki Inaba, Fumio Kanehiro, Satoshi Kagami, and Hirochika Inoue. Two-armed bipedal robot that can walk, roll over and stand up. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 297–302. IEEE, 1995.

[30] Tatsuzo Ishida, Yoshihiro Kuroki, and Taro Takahashi. Analysis of motions of a small biped entertainment robot. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 142–147. IEEE, 2004.

[31] Tor Arne Johansen, Thor I Fossen, and Stig P Berge. Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. *IEEE Transactions on Control Systems Technology*, 12(1):211–216, 2004.

[32] Kiyoshi FUJIWARA Fumio KANEHIRO Shuuji KAJITA, Kazuhito YOKOI, Hajime SAITO Kensuke HARADA Kenji KANEKO, and Hirohisa HIRUKAWA. The first human-size humanoid that can fall over safely and stand-up again. In *IEEE-RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA*, volume 27, pages 1920–1926, 2003.

[33] Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Shuuji Kajita, Kazuhito Yokoi, Hirohisa Hirukawa, Kazuhiko Akachi, and Takakatsu Isozumi. The first humanoid robot that has the same size as a human and that can lie down and get up. In *Robotics and Automation, 2003. Proceedings.*

ICRA'03. IEEE International Conference on, volume 2, pages 1633–1639. IEEE, 2003.

[34] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[35] Lydia E Kavraki, Jean-Claude Latombe, Rajeev Motwani, and Prabhakar Raghavan. Randomized query processing in robot path planning. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 353–362. ACM, 1995.

[36] Oussama Khatib, Luis Sentis, and Jae-Heung Park. A unified framework for whole-body humanoid robot control with multiple constraints and contacts. In *European Robotics Symposium 2008*, pages 303–312. Springer, 2008.

[37] Jinhyun Kim, Giacomo Marani, Wan Kyun Chung, and Junku Yuh. Task reconstruction method for real-time singularity avoidance for robotic manipulators. *Advanced Robotics*, 20(4):453–481, 2006.

[38] Soonkyum Kim and Frank Chongwoo Park. Fast robot motion generation using principal components: framework and algorithms. *IEEE Transactions on Industrial Electronics*, 55(6):2506–2516, 2008.

[39] James J Kuffner Jr, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Footstep planning among obstacles for biped robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[40] Yasuo Kuniyoshi, Yoshiyuki Ohmura, Koji Terada, and Akihiko Nagakubo. Dynamic roll-and-rise motion by an adult-size humanoid robot. *International Journal of Humanoid Robotics*, 1(03):497–516, 2004.

[41] Arthur D Kuo, J Maxwell Donelan, and Andy Ruina. Energetic consequences of walking like an inverted pendulum: step-to-step transitions. *Exercise and sport sciences reviews*, 33(2):88–97, 2005.

[42] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[43] Steven M LaValle. Rapidly-exploring random trees a new tool for path planning. 1998.

[44] Steven M LaValle. *Planning algorithms.* Cambridge university press, 2006.

[45] Steven M. LaValle. Motion planning. *Robotics & Automation Magazine, IEEE*, 18(1):79–89, 2011.

[46] Sung-Hee Lee and Ambarish Goswami. A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Autonomous Robots*, 33(4):399–414, 2012.

[47] Sêbastien Lengagne, Joris Vaillant, Eiichi Yoshida, and Abderrahmane Kheddar. Generation of whole-body optimal dynamic multi-contact motions. *The International Journal of Robotics Research*, 32(9-10):1104–1119, 2013.

[48] Bokman Lim, Syungkwon Ra, and Frank C Park. Movement primitives, principal component analysis, and the efficient generation of natural motions. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 4630–4635. IEEE, 2005.

[49] Tomas Lozano-Perez. Spatial planning: A configuration space approach. *Computers, IEEE Transactions on*, 100(2):108–120, 1983.

[50] Carlos L Lück. Self-motion representation and global path planning optimization for redundant manipulators through topology-based discretization. *Journal of Intelligent and Robotic Systems*, 19(1):23–38, 1997.

[51] Daniel P Martin, John Baillieul, and John M Hollerbach. Resolution of kinematic redundancy using optimization techniques. *IEEE Transactions on Robotics and Automation*, 5(4):529–533, 1989.

[52] Tad McGeer. Principles of walking and running. *Advances in comparative and environmental physiology*, 11:113–139, 1992.

[53] Tad McGeer. Dynamics and control of bipedal locomotion. *Journal of Theoretical Biology*, 163(3):277–314, 1993.

[54] Michael Mistry, Akihiko Murai, Katsu Yamane, and Jessica Hodgins. Sit-to-stand task on a humanoid robot from human demonstration. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 218–223. IEEE, 2010.

[55] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):43, 2012.

[56] Jun Morimoto and Kenji Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems*, 36(1):37–51, 2001.

[57] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[58] Yoshihiko Nakamura. *Advanced robotics: redundancy and optimization*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[59] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurement, and control*, 108(3):163–171, 1986.

[60] Isaac Newton. *Philosophiae naturalis principia mathematica.* sumptibus Societatis, 1723.

[61] Shintaro Noda, Masaki Murooka, Shunichi Nozawa, Yohoei Kakiuchi, Kei Okada, and Masayuki Inaba. Generating whole-body motion keep away from joint torque, contact force, contact moment limitations enabling steep climbing with a real humanoid robot. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1775–1781. IEEE, 2014.

[62] Petter Ogren. Formations and obstacle avoidance in mobile robot control. 2003.

[63] David E Orin and Ambarish Goswami. Centroidal momentum matrix of a humanoid robot: Structure and properties. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[64] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35(2-3):161–176, 2013.

[65] Giuseppe Oriolo and Christian Mongillo. Motion planning for mobile manipulators along given end-effector paths. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2154–2160. IEEE, 2005.

[66] Georges Pagis, Nicolas Bouton, Sébastien Briot, and Philippe Martinet. Design of a controller for enlarging parallel robots workspace through type 2 singularity crossing. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4249–4255. IEEE, 2014.

[67] FC Park and Jin Wook Kim. Singularity analysis of closed kinematic chains. *Journal of mechanical design*, 121(1):32–38, 1999.

[68] Frank C Park and Jin Wook Kim. Manipulability and singularity analysis of multiple robot systems: A geometric approach. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1032–1037. IEEE, 1998.

[69] Hae-Won Park, Sangin Park, and Sangbae Kim. Variable-speed quadrupedal bounding using impulse planning: Untethered high-speed 3d running of mit cheetah 2. In *IEEE International Conference on Robotics and Automation*, 2015.

[70] Jaeheung Park and Oussama Khatib. Contact consistent control framework for humanoid robots. In *IEEE International Conference on Robotics and Automation*, 2006.

[71] Nicolas Perrin, Olivier Stasse, Léo Baudouin, Florent Lamiraux, and Eiichi Yoshida. Fast humanoid robot collision-free footstep planning using swept volume approximations. *IEEE Transactions on Robotics*, 28(2):427–439, 2012.

[72] Michael Posa and Russ Tedrake. Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic foundations of robotics X*, pages 527–542. Springer, 2013.

[73] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, Rob Playter, et al. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress*, volume 17, pages 10822–10825, 2008.

[74] Marc H Raibert, H Benjamin Brown, and Michael Chepponis. Experiments in balance with a 3d one-legged hopping machine. *The International Journal of Robotics Research*, 3(2):75–92, 1984.

[75] John H Reif. *Complexity of the generalized mover's problem.* Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1985.

[76] Jacob T Schwartz and Micha Sharir. On the "piano movers" problem. ii. general techniques for computing topological properties of real algebraic manifolds. *Advances in applied Mathematics*, 4(3):298–351, 1983.

[77] Sanjeev Seereeram and John T Wen. A global approach to path planning for redundant manipulators. *IEEE Transactions on Robotics and Automation*, 11(1):152–160, 1995.

[78] Shamik Sen, Bhaskar Dasgupta, and Asok Kumar Mallik. Variational approach for singularity-free path-planning of parallel manipulators. *Mechanism and Machine Theory*, 38(11):1165–1183, 2003.

[79] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(4):505–518, 2005.

[80] Luis Sentis, Jaeheung Park, and Oussama Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Transactions on Robotics*, 26(3):483–501, 2010.

[81] Sangok Seok, Albert Wang, David Otten, and Sangbae Kim. Actuator design for high force proprioceptive control in fast legged locomotion. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1970–1975. IEEE, 2012.

[82] Jörg Stückler, Johannes Schwenk, and Sven Behnke. Getting back on two feet: Reliable standing-up routines for a humanoid robot. In *IAS*, pages 676–685, 2006.

[83] Russell H Taylor. Planning and execution of straight line manipulator trajectories. *IBM Journal of Research and Development*, 23(4):424–436, 1979.

[84] Koji Terada, Yoshiyuki Ohmura, and Yasuo Kuniyoshi. Analysis and control of whole body dynamic humanoid motion-towards experiments on a roll-and-rise motion. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1382–1387. IEEE, 2003.

[85] Kenneth J Waldron and Kenneth H Hunt. Series-parallel dualities in actively coordinated mechanisms. *The International Journal of Robotics Research*, 10(5):473–480, 1991.

[86] Patrick M Wensing and David Orin. Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *IEEE International Conference on Robotics and Automation*, 2013.

[87] Eric R Westervelt, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2007.