Sample-Based Motion Planning in High-Dimensional and Differentially-Constrained Systems

by

Alexander C. Shkolnik

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010

Author .....
Department of Electrical Engineering and Computer Science
January 29, 2010

Certified by .....
Russ Tedrake
Associate Professor
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chair, Department Committee on Graduate Students

Sample-Based Motion Planning in High-Dimensional and Differentially-Constrained Systems

by

Alexander C. Shkolnik

Submitted to the Department of Electrical Engineering and Computer Science
on January 29, 2010, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

State of the art sample-based path planning algorithms, such as the Rapidly-exploring Random Tree (RRT), have proven to be effective in path planning for systems subject to complex kinematic and geometric constraints. The performance of these algorithms, however, degrade as the dimension of the system increases. Furthermore, sample-based planners rely on distance metrics which do not work well when the system has differential constraints. Such constraints are particularly challenging in systems with non-holonomic and underactuated dynamics. This thesis develops two intelligent sampling strategies to help guide the search process. To reduce sensitivity to dimension, sampling can be done in a low-dimensional task space rather than in the high-dimensional state space. Altering the sampling strategy in this way creates a Voronoi Bias in task space, which helps to guide the search, while the RRT continues to verify trajectory feasibility in the full state space. Fast path planning is demonstrated using this approach on a 1500-link manipulator. To enable task-space biasing for underactuated systems, a hierarchical task space controller is developed by utilizing partial feedback linearization. Another sampling strategy is also presented, where the local reachability of the tree is approximated, and used to bias the search, for systems subject to differential constraints. Reachability guidance is shown to improve search performance of the RRT by an order of magnitude when planning on a pendulum and non-holonomic car. The ideas of task-space biasing and reachability guidance are then combined for demonstration of a motion planning algorithm implemented on LittleDog, a quadruped robot. The motion planning algorithm successfully planned bounding trajectories over extremely rough terrain.

Thesis Supervisor:

Russ Tedrake, Associate Professor

Thesis Committee:

Tomás Lozano-Pérez, Professor of Computer Science and Engineering, MIT
Nicholas Roy, Associate Professor of Aeronautics and Astronautics, MIT
Steven M. LaValle, Professor of Computer Science, University of Illinois

Acknowledgments

Many people have influenced my thinking over the years, and have directly or otherwise contributed to the work developed in this thesis. First, I would like to express tremendous gratitude to my advisor, Russ Tedrake. Russ's passion for robotics is infectious, and I was fortunate to be his first student. As Russ's lab grew over the years, he made himself available to his students as a priority. Russ provided just the right balance of a hands-on approach, pushing me when needed, but otherwise giving me tremendous freedom.

I would also like to express my appreciation for the other members of my thesis committee, including Nick Roy, who in addition to being on my committee was also a co-PI during the first two phases of the LittleDog project, and Tomás-Lozano Pérez and Steve LaValle, who have been instrumental in the field of motion planning. The interaction with my committee was insightful, and helpful in giving me a comprehensive view of the problems I was trying to address.

The LittleDog project has been an enormous team effort, and it is worth acknowledging the efforts of the team. I worked closely with Michael Levashov in Phase III of the project. Michael developed and identified a physics-based model of LittleDog, which was used in the motion planning algorithms developed in this thesis. As I write this, Michael and Ian Manchester continue to work with the robot, to achieve feedback stabilization to get these plans to execute on the robot. During phase II, Katie Byl, now an assistant professor at UC Santa Barbara, perfected the open-loop half-bound maneuver, enabling the robot to quickly get over gates, steps and gaps with the dog. At the same time, Sam Prentice implemented an A* planner for LittleDog, which planned footholds and body poses for walking over rough terrain. Khashayar Rohanimanesh was a postdoc in the lab, and we worked closely on low level through high level control for LittleDog throughout Phase I of the project. Other team members, including Patrycja Missiuro, Emma Brunskill, and Stephen Proulx also contributed in the first phases. Finally, I should mention the insightful interaction with the other LittleDog team PI's and students, as well as Boston Dynamics (developer of the robot) and DARPA (Who funded the project). Additionally, the work was made much smoother by TIG, the infrastructure group at CSAIL.

In addition to LittleDog, I had the opportunity to work with and mentor several undergraduate student researchers (UROPS), all of whom were insightful and diligent. I worked with Elena Glassman on control for a self-righting maneuver of a falling cat robot, which was eventually built in hardware by Steve Proulx and then by Fumiya Iida, a postdoc in our lab. Elena is continuing her studies in the lab as a PhD student, and is studying dynamic distance metrics for sample-based kinodynamic planning in state space. Michael Price was another talented UROP who did some early coding on LittleDog. Lauren White developed clean code in SD/FAST and C to generate physics based models of multi-link arms. The models were made compatible with Matlab, and were used in the motion planning algorithms in this thesis. Her code became the basis for our planar 5-link LittleDog model which Michael Levashov further developed. Finally, Sara Itani worked on LittleDog in Phase III, and helped tune an openloop bounding gait on the robot on flat terrain.

The resources of MIT are vast, and I particularly enjoyed interacting with the students

and postdocs that study robotics, both in research and in social environments. Tom Kollar and I had semi-regular research meetings, and Tom helped me define the N-Link arm as a task space toy problem. Matt Walter was instrumental in writing the RG-RRT ICRA paper, which is a chapter in this thesis. There were many others, but I especially want to mention some of the folks that I had the most opportunity to interact with, including Rick Cory, John Roberts, Albert Huang, Olivier Koch, Alex Bahr, and the other members of 32-33x and Russ's lab.

The research in this thesis was made possible by funding that came from DARPA (Learning Locomotion Program) and the NSF (through a Graduate Research Fellowship).

I'd like to acknowledge several people who mentored me, and shaped my research career. This includes Shuuji Kajita who I was able to work with through a Summer Research Program funded by the NSF and JSPS; Tommy Poggio and Gabriel Krieman who were my first research advisors at MIT; Steve Potter and Thomas DeMarse, who supervised my masters thesis at the Georgia Tech NeuroEngineering Lab; David Edwards and Pat Marsteller, who introduced my to research at Emory; and Chet Bacon, a (now retired) electronics teacher who sparked my interest in computer science.

Outside of my research at MIT, I've been involved with LiquidPiston, a startup company which my father (Nick) and I founded in 2003 to develop a new type of combustion engine. I was fortunate to be able to work so closely with my father on this project. Of course, I have to thank Russ for allowing me the freedom to pursue this project, as well as all of the people I've had a chance to work with through LiquidPiston.

I'd also like to thank my mother, Valentina, and the rest of my friends and family – I could not have asked for a better support network.

Finally, I am forever indebted to my wife, Lauren, who has stood by me and encouraged me through these years. Lauren has been a beacon of stability and understanding in my life.

Contents

List of Figures

Chapter 1

Introduction

Technology visionaries have eagerly anticipated the time when robots will work side by side with people, taking care of rudimentary chores, assisting the aging population, exploring areas that are too remote or dangerous for humans, preventing casualties in war, etc. Yet robots have so far performed a more perfunctory role, within factories, in surgical applications, and in certain other niche applications. So what has prevented robot proliferation outside of these applications? Factory robots clearly outperform human counterparts in precision and speed of executing certain tasks. The reason for success in the factory may be attributed to four factors. First, such robots typically plan in fewer than $\sim$7 degrees of freedom. Second, the machines are usually bolted to the ground, and have large actuators, so that the robotic arm has complete control over all of its degrees of freedom. Third, there is relatively little uncertainty in a factory setting. Finally, the factory robot typically performs rudimentary repetitive motions.

To break out of the factory, the science of robotics must improve. On one hand, robotics research has focused on improving hardware, including actuators, energy storage mechanisms, and sensors. Additionally, improvements are being made in mechanical design to reduce the burden of the actuators or control systems. Software and algorithmic development has focused on several areas, such as state estimation, including filtering, localization and mapping. High level "intelligence" is also an important component of robotics software, which increasingly deals with human-robot interaction, robot-robot interaction and cooperation, and robot-environment interaction. At a more fundamental level, however, robots must understand how to move around and execute desired behaviors within their environment: this is the problem of motion planning.

Consider that the human body has over 600 muscles, and over 200 bones, more than half of which are found in the hands and feet. Furthermore, more than half of the neurons in the human brain are found in the cerebellum, a region thought to be responsible for fine-tuning motor movement. This demonstrates that humans (and animals) are very high dimensional systems, that devote a significant portion of their brain power to controlling movement. One could make the argument that today, the bottleneck in achieving human or animal-like agility in a robot, is in the lack of a comprehensive motion planning and control strategy

Figure 1-1: Person and dog cross river by hopping stones. Legs are required, and planning steps and motions are essential for such a task. Picture (left) reproduced from http://www.flickr.com/photos/peturgauti; picture (right) by Alison Plant

that is capable of dealing with the dimension of the planning problem, while also taking into account the kinematic and dynamic constraints imposed by the robot and the world it exists in. The dynamic constraints, including the governing equations of motion, become considerably more challenging for motion planning when the system is not "bolted" to the ground, as factory robots are.

In this thesis, I present motion planning strategies for systems having many degrees of freedom, as well as strategies for planning agile maneuvers even when the system is subject to complex dynamic constraints in an obstacle filled world. This type of problem is exemplified by dynamic legged locomotion. In addition to various toy problems, the algorithms developed in this thesis were applied on LittleDog, a small quadruped robot described in section 1.2, to find continuous bounding trajectories over very rough terrain.

1.1 Motivation

The work in this thesis is largely motivated by the problem of legged locomotion, a task that animals and humans are clearly very good at (see for example Figure 1-1). At first glance, the task may seem trivial, but the difficulty is hidden by the fact that most of the brain's processing for locomotion is done at the subconscious level. In fact, the sensorimotor system accounts for a vast majority of the brain. As mentioned above, the human cerebellum, a region thought to be responsible for *fine-tuning* motor behavior, itself makes up more than 50% of the number of neurons in the entire brain. Even more impressive - this region receives 200 million input fibers, compared to 1 million fibers carried by the optic nerve. This processing power is required to enable humans and animals to move and interact gracefully with the environment. State of the art legged robots, on the other hand (see Figure 1-2), either move slowly and deliberately, or quickly and reactively. The latter class is typically blind to the world, and reacts to the environment instead of anticipating it. To become more agile, robots must move quickly *and* deliberately, while taking advantage of knowledge

Figure 1-2: State of the art legged robots: (Left) Honda Asimo "running". Note the flat foot. (Right) Boston Dynamics' BigDog climbing a snowy hill.

about the environment.

Even if one ignores the problem of sensing and uncertainty, e.g. if the state of the system and its environment is fully known, the problem of planning and executing motions for complicated (high dimensional), and particularly for underactuated mechanical systems is still very challenging. A system is underactuated if it can not accelerate all of its Degrees of Freedom (DOF) in arbitrary directions at every instant in time. As mentioned above, part of the reason that factory robots are successful is because they are bolted to the ground, and do not have to worry about falling over. Legged robots do not have this luxury. Underactuated systems are typified by robots that have more DOFs than actuators, such as the Acrobot [Murray and Hauser, 1991] or Pendubot [Spong et al., 1995]. Many important classes of robots are unavoidably underactuated, including most walking, swimming, flying and free-floating (space) machines. Legged robots, for example, are underactuated at the interface between the foot and the ground. The system cannot produce arbitrary ground reaction forces or moments, as it is only possible to push on the ground, but not possible to pull on the ground. This is especially true for point-foot walkers (e.g., [Goswami et al., 1996]), but is also true for flat-foot walkers like Honda's ASIMO's [Honda Motor Co., 2004].

In addition to the complexity introduced by the constraints imposed by underactuated dynamics, real-world robots are subject to additional kinodynamic constraints, for example position, velocity and torque limits, as well as obstacle collisions. It is natural to formulate this type of problem with a motion planning framework, the goal of which is to find feasible trajectories that take a model of the system from an initial pose to a desirable goal pose, while taking into consideration all of the constraints. Sample-based methods, in particular, have been shown to be effective at solving motion planning problems with complex geometric constraints. However, search efficiency for sample-based planning algorithms is hindered by two factors: 1) kinematic path planning problems are known to be computationally

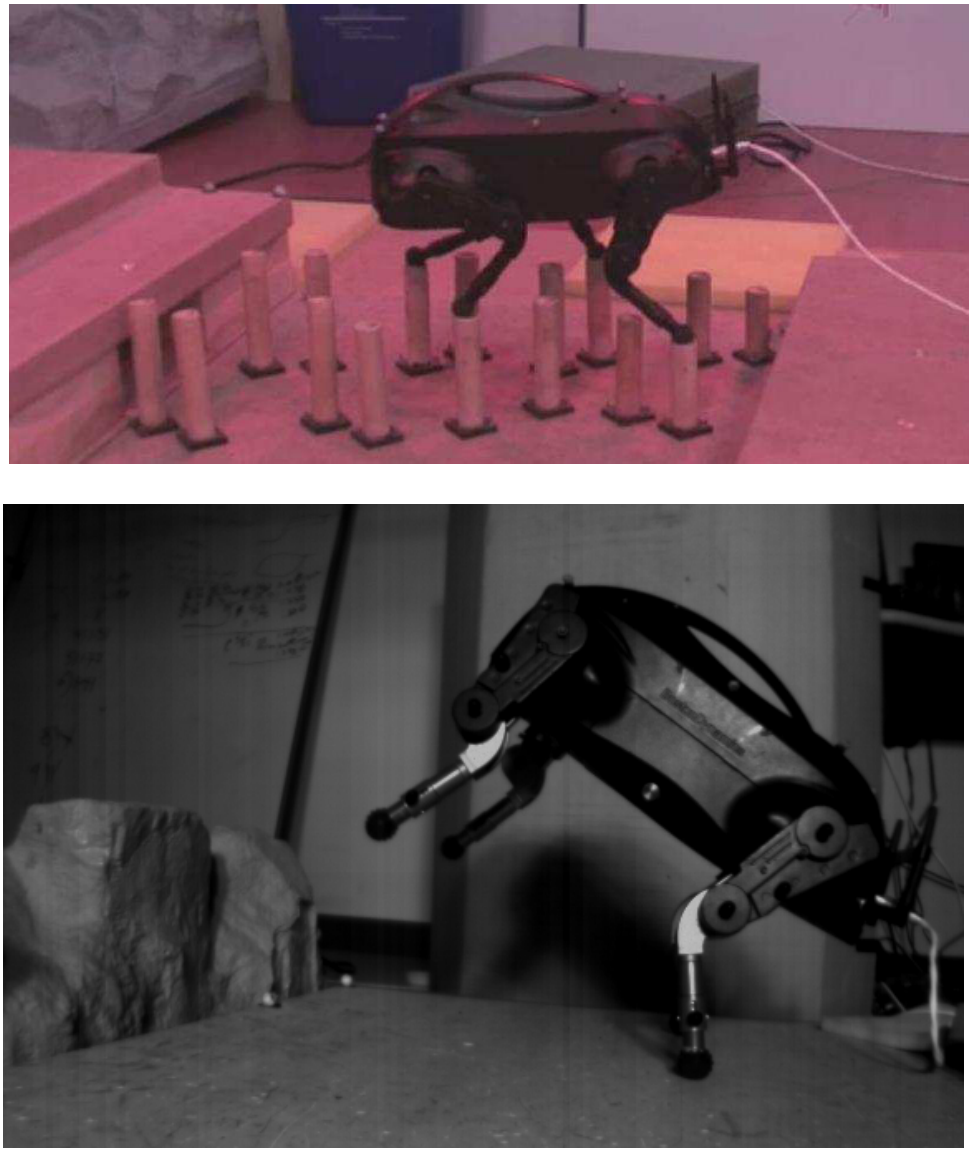


Figure 1-3: TOP: LittleDog robot walking on pegs. BOTTOM: LittleDog on hind legs.

difficult, where search is sensitive to dimensionality of the system, and 2) differential kinodynamic constraints, especially underactuated dynamics, impose additional challenges that significantly reduce search efficiency.

1.2 LittleDog

The algorithms in this thesis were developed with the specific goal of achieving continuous bounding over extremely rough terrain with the LittleDog robot. LittleDog, is a small, 3kg point-footed quadruped robot platform shown in Figure 1-3. This robot was developed by

Boston Dynamics Inc. as part of the DARPA sponsored Learning Locomotion program, in which several universities competed to develop motion planning algorithms to enable the robot to quickly traverse a variety of terrain. The robot state information is provided by motion capture and encoders. Additionally, a laser scanned height map of the terrain is provided to the motion planner before each run. The goal of the program was to push the limits of what robots could do on rough terrain, given reasonable knowledge about the robot state and the world.

Unlike walking gaits, bounding gaits are dynamic gaits which cannot be planned in the configuration space with quasi-steady approximations. LittleDog has 12 actuators (2 in each hip, and one in each knee), and a 40-dimensional state space. The motion planning problem is compounded by differential constraints, including torque and velocity limited motors, and the fact that this robot is very much underactuated during bounding. Attempting to plan an agile bounding gait over rough terrain, even in simulation, requires the development of motion planning tools that are capable of dealing with both, the very high dimensionality of this robot's state space, as well as challenging differential constraints.

1.3 Review of the State of the Art

The thesis combines work from several areas, including:

1. Sample-based motion planning
2. Task space feedback control
3. Underactuated Systems and Control
4. Task Space Control of Underactuated Systems
5. Legged Locomotion

Each of these topics is reviewed in turn. This section provides some background information related to the thesis as a whole, but each chapter will provide additional background information pertinent to that chapter.

1.3.1 Sample-Based Planning

The book by LaValle [LaValle, 2006] provides a broad overview of state of the art planning problems and algorithms. Much of the seminal work in spatial (manipulation) planning was done by Lozano-Pérez and others [Lozano-Perez and Tomas, 1981,Lozano-Perez et al., 1983], particularly in defining configuration space (sometimes referred to as C-space, see definition below). The C-space uniquely determines the position of each DOF in a system, so that an entire configuration of the robot is described by a single point. The C-space can be broken down into free regions which are collision-free, and obstacle regions, corresponding to a configuration in which some part of the robot is in collision. The task of a planning

algorithm can then be simplified to finding a path for a point through free regions of C-space, from a starting configuration point, to a goal point. Most of the work done in the field assumes fully actuated dynamics (e.g. the robot can move in any direction in configuration space), but that work has set the framework for planning with obstacles.

Definition 1. *The **Configuration Space** (C-space) is the set of all possible transformations (configurations) that can be applied to a system. A configuration will completely describe every point on the robot. The configuration space serves as an abstraction layer in motion planning, whereby algorithms that work generally for C-spaces can be applied to more specific systems. The C-space is typically an n-dimensional manifold, where n is the number of degrees of freedom of the robot.*

Definition 2. *The **State Space** (X) is the set of all possible transformations, velocities, and other system variables that represent the entire state of a system at a given instant of time. For rigid body systems with deterministic dynamics, which is representative of the problems of interest in this thesis, the state space consists of the configuration space augmented with the set of all possible velocities for each degree of freedom of the configuration space. Note that C-space and X are used interchangeably in path-planning or motion-planning contexts when it makes sense to do so.*

Definition 3. *Define the subscript $\mathcal{X}_{obs}$ to correspond to the subset that is in collision. On the contrary, the subscript $\mathcal{X}_{free}$ refers to the subset that is not in collision, $\mathcal{X}_{free} = \mathcal{X} \backslash \mathcal{X}_{obs}$*

Path planning for kinematic manipulation tasks, such as the piano-mover problem, has been shown to be PSPACE-Hard [Reif, 1979]. Despite this time complexity, sample-based methods have proven to be effective at solving these types of problems, with fast average-time performance. Algorithms such as Probabilistic RoadMaps (PRMs) [Kavraki et al., 1996], can be used for single- or multi-query planning in C-space. This method involves sampling many points in free configuration space before planning. A fast local planner then decides which proximal points are reachable in order to create a connected graph. After the graph is constructed it is relatively simple to find paths through the nodes from any start to any end node (hence the term multi-query). PRMs allow for fast querying between multiple start / ending conditions, but building the graph initially can be time consuming. The PRM approach may be less suitable for problems with significant constraints in dynamics as it becomes more difficult to find even local connections between samples. The approach is also hindered by dimensionality as the number of samples required in order to maintain a constant sampling density goes up exponentially with the number of DOF.

An alternative sampling-based planner, which is geared toward quickly finding a feasible path from a single initial pose to a single goal pose ("single query"), is the Rapidly-exploring Random Tree (RRT) [LaValle and Kuffner, 2000, LaValle et al., 2001]. In this approach, a tree is created in C-space. The nodes represent collision free configurations that can be reached, and edges represent feasible collision free actions between nodes. The root node of the tree is the initial configuration of the robot, and the tree is grown until a path is found to a goal configuration. The standard RRT algorithm [LaValle et al., 2001] is provided for reference in Algorithm 1 and 2, and is visually illustrated in Figure 1-4.

Algorithm 1 BUILD-RRT (q_0)

T.init(q_0)
for $k = 1$ to K **do**
  if with some probability P **then**
    $q_{rand} \Leftarrow$ RANDOM-SAMPLE() $\in \Re^N$
  else
    $q_{rand} \Leftarrow q_{goal}$
  end if
  EXTEND(T,q_{rand})
end for

Algorithm 2 EXTEND (T, q)

$q_{near} \Leftarrow$ NEAREST-NEIGHBOR(q,T)
$u \Leftarrow$ CONTROL(q_{near}, q_{rand})
$q_{new} \Leftarrow$ NEW-STATE(q_{near}, u)
if COLLISION-FREE (q_{near}, q_{new}) **then**
  T.add-node(q_{near}, q_{new}, u)
end if

In the first step, a C-space sample, $\mathbf{q}_{rand}$, is chosen at random, usually from a uniform distribution over some bounded region of C-space or state space. The closest node on the tree, $\mathbf{q}_{near}$, is selected by the NEAREST-NEIGHBOR() function. This function is typically implemented by choosing the node with the minimum Euclidean distance to the sample, although in state space the cost-to-go from the node in the tree to the sample would be more appropriate [Frazzoli et al., 2002, Glassman and Tedrake, 2010, Tedrake, 2009a], but is difficult to compute.

An action, $\mathbf{u}$ is then determined by the CONTROL() function. Applying this action to the node should make progress towards the random sample, at least with probability > 0. Finally a new state, $\mathbf{q}_{new}$, is computed in the function NEW-STATE(). For dynamic systems, this is done by integrating from $\mathbf{q}_{near}$ while applying control $\mathbf{u}$ for some Δt. During integration, the trajectory is checked for kinodynamic feasibility. If it is collision free, the node and edge are added to the tree.

The expansion process is sometimes biased towards the goal node, which can increase search speed dramatically. The RRT algorithm is efficient because it introduces a Voronoi bias, meaning that it is encouraged to explore open (yet unexplored) regions of the configuration space. Figure 1-4 illustrates a 2D RRT while going through an EXTEND operation. This figure also shows Voronoi diagrams associated with the trees. Each Voronoi region (or cell) contains a single node of the tree. The Voronoi cell delineates the region for which any samples are mapped by the NEAREST-NEIGHBOR function to the corresponding node in the tree. Assuming uniform sampling, the probability that a given node of the tree will be expanded is then directly proportional to the area of its Voronoi cell. Because of the Voronoi bias, the largest regions of unexplored space, which have large Voronoi cells, are

most likely to be chosen for expansion into these regions. Thus, RRT's have a tendency to quickly branch into unexplored regions. When the unexplored regions become smaller and more uniformly distributed, then the search begins to fill in gaps with increasingly uniform coverage, ensuring that the algorithm is probabilistically complete, meaning that it will find

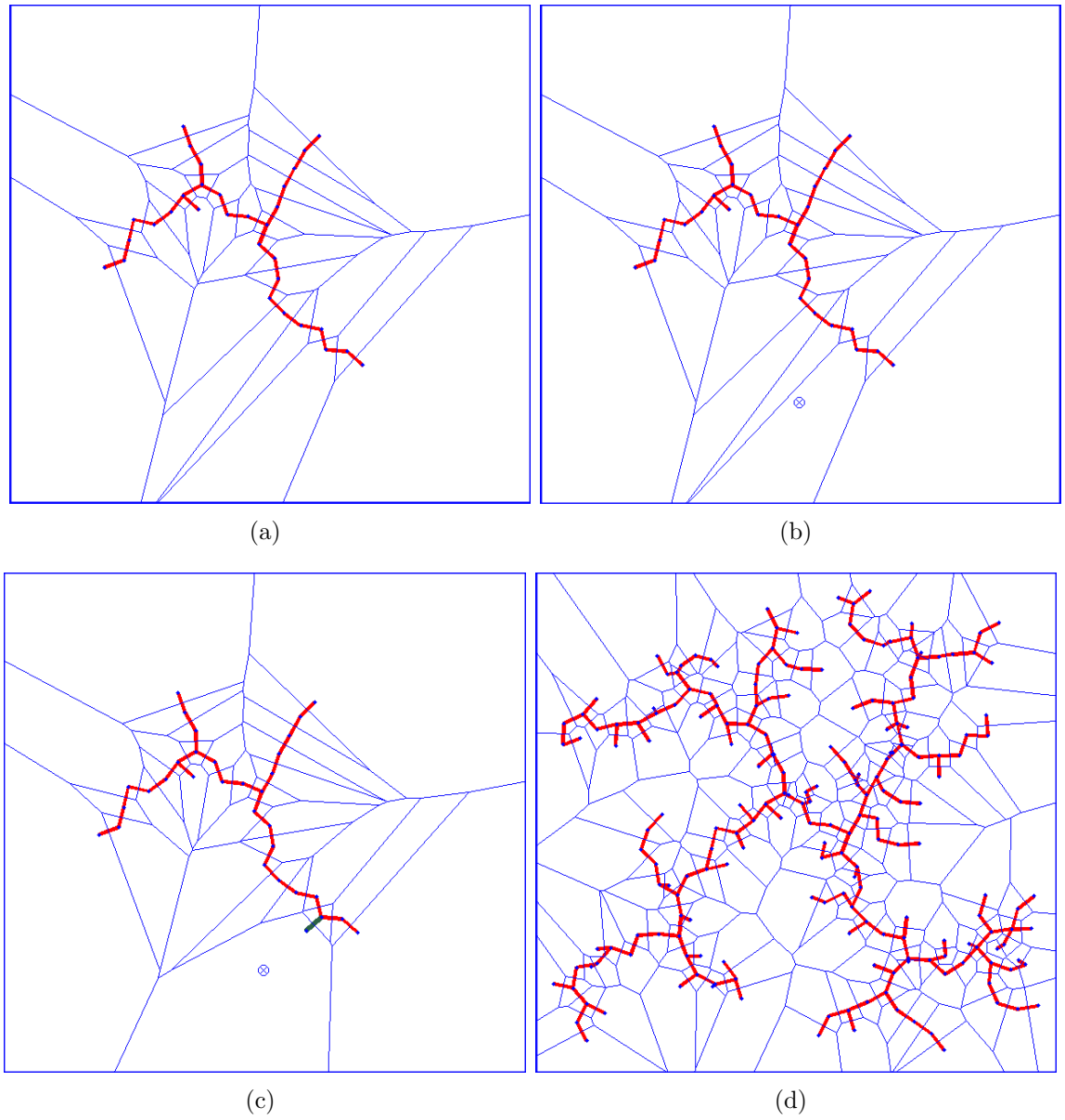


Figure 1-4: Steps of the RRT algorithm are shown, with out any obstacles, starting with 30 nodes. The blue lines delineate the Voronoi regions associated with each node. (a): RRT with 30 nodes. (b): A random sample is drawn (blue circle). (c): The closest node of the tree is selected, a new node is added. The new node is highlighted in green. Note the change in Voronoi regions after adding the new node. (d): RRT after 200 expansions.

a path if one exists as K goes to infinity.

The PRM is also probabilistically complete as the number of samples goes to infinity. Note that we have introduced both, PRMs and RRTs, as using random sampling, but in fact deterministic sampling strategies could be used as well. In that case the algorithms would be resolution complete, rather than probabilistically complete. In the remainder of this thesis we usually refer to random sampling, but most of the concepts will generalize to deterministic sampling as well.

An improvement on the RRT is a bidirectional version called RRT-Connect [Kuffner et al., 2000]. In this algorithm there are two trees - one tree is grown in the forward direction from the start node, while the other is grown backwards from the goal node. Each time a node is added to one tree, using the standard RRT expansion step, the other tree is then iteratively grown towards the new node until a collision occurs, or until the two trees are successfully connected. The RRT-Connect algorithm can be much faster than the single RRT for complicated single query planning problems.

1.3.2 Task Space Control

The planning algorithms discussed thus far have been applied primarily in problems within C-space or state space, where constraints such as joint feasibility and collisions with obstacles are easily checked in the same space. In some cases it is desirable to specify robot behaviors in a different command space. Consider for example a humanoid robot which is trying to pick up a cup. There might be infinite robot configurations which correspond to the humanoid's hand being on the cup. Instead of specifying this behavior in joint angles in configuration space, if we consider the hand position as a task space, this can be more intuitive for the control designer, and can capture more of the intent of an action.

Definition 4. *Suppose $f : \mathcal{C} \rightarrow \mathcal{T}$ is a function mapping from the configuration space, $\mathcal{C}$, to a lower dimensional space, $\mathcal{T}$, where each configuration is mapped to a single point in $\mathcal{T}$. We may choose to call f a **Task Space Mapping**, and $\mathcal{T}$ the **Task Space** (T-space).*

The mapping f can be defined somewhat arbitrarily, however for the purposes of motion planning, it is generally useful to have at least some of the following properties:

1. *Reduced Dimension: If the C-space is an n dimensional manifold, the T-space should be an m dimensional manifold, such that $m << n$*

2. *Topological Applicability: There may be topological considerations for a planning space. For example, it may be beneficial if connected regions of C-space map to connected regions in T-space, and for the opposite to be true as well, where disconnected regions of T-space correspond to disconnected regions in C-space*

3. *Smoothness and differentiability of the Task space Mapping, (or to higher order, being k times differentiable)*

4. *Goal Specificity: It is useful if the motion planning goal set $\mathcal{G}$ can be specified by a point (or region) within $\mathcal{T}$. For this to hold, for all regions of C-space that map to a goal point specified in T-space, $y_g \in \mathcal{T}$, the corresponding C-space region should also be in the goal set*

5. *Extension to state space: for certain problems, the task space mapping may include velocity or other state information.*

6. *Collision Information: Some motion planners can take advantage of having regions of T-space for which all associated regions of C-space are in collision, so that obstacle regions of task space can be defined. In this thesis, we do not require this condition.*

Definition 5. *The term **workspace** is a task space consisting of Cartesian coordinates in 2-D or in 3-D, corresponding to the world where the robot lives and is used to specify World-frame coordinates of points on the robot.*

Choices for task space mappings will be explored later in this thesis, but some examples might include the workspace position of an end effector, or the position of the robot Center of Mass. Because the task-space mapping can be nonlinear and there can be infinite configurations that map to a single task-space coordinate, $\mathbf{y}$ from a point in C-space, $\mathbf{q}$, the problem of inverse kinematics can be difficult or even impossible. Fortunately, the task space mapping can be linearized, presenting an easy to compute relationship between the velocity in joint space, and the velocity in task space:

$$\mathbf{y} = f(\mathbf{q})$$

$$\dot{\mathbf{y}} = \mathbf{J} \cdot \dot{\mathbf{q}},$$

where the Jacobian, $\mathbf{J}$, is defined as:

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}.$$

A trajectory in task space can be mapped to a trajectory in configuration space by using the Moore-Penrose pseudoinverse. This pseudoinverse is defined for a matrix in which the rows are linearly independent as:

$$\mathbf{J}^{+} = \mathbf{J}^{T}(\mathbf{J} \cdot \mathbf{J}^{T})^{-1}.$$

By using the Moore-Penrose pseudoinverse, one can resolve redundancies [Liegeois, 1977] when the dimension of the configuration space exceeds the dimension of the task space, as is often the case, by iteratively applying:

$$\dot{\mathbf{q}} = \mathbf{J}^{+} \cdot \dot{\mathbf{y}} + (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\dot{\mathbf{q}}_{ref}.$$

This equation will produce a joint velocity, $\dot{\mathbf{q}}$, the projection of which exactly (instantaneously) follows a desired task velocity, $\dot{\mathbf{y}}$, if it is possible to do so, and then minimizes the

squared error in the commanded secondary joint velocity command, $\dot{\mathbf{q}} - \dot{\mathbf{q}}_{ref}$. The secondary command offers a redundancy resolution mechanism, which is executed in the null space of the Jacobian (only if it is possible to do so). If the null space term is omitted, the command will be the joint velocities with the smallest norm.

Task space control has been developed in depth for humanoid robot motion planning, where the secondary task typically involves maintaining a desirable base position of the robot whenever possible [Khatib et al., 2004]. This is achieved by defining $\dot{\mathbf{q}}_{ref}$ as a workspace-centering PD type controller, $\dot{\mathbf{q}}_{ref} = Kp \cdot (\mathbf{q}_d - \mathbf{q}) + Kd \cdot (\dot{\mathbf{q}}_d - \dot{\mathbf{q}})$ where the desired position is set to an arbitrary constant pose, and the desired velocity is zero. This type of control was also extended to operation space, where ground reaction forces are commanded [Sentis and Khatib, 2005], and can be used to command velocities or accelerations (see [Nakanishi et al., 2005] for review). For underactuated systems, in Chapter 3, I show that the choice of redundancy resolution mechanism is particularly important for maintaining dynamic stability (preventing the robot from falling over) [Shkolnik and Tedrake, 2007].

1.3.3 Underactuated Systems

A significant body of research is focused on control of underactuated systems (see [Tedrake, 2009b] for a good overview). Multi-link arms with passive joints are a representative subset of general underactuated systems. Mathematically, such models can be used to describe planar walking machines when rolling around the toe joint, or to describe the LittleDog robot standing on its two hind legs. Although underactuated systems are typically not feedback linearizable, it is possible to achieve a partial feedback linearization (PFL) [Spong, 1994a], in which the dynamics of some subset of the degrees of freedom are replaced (through feedback) with the dynamics of a simple linear system. This subset is not restricted to simply the actuated joints, but can also be used to exploit inertial coupling to linearize the dynamics of the unactuated joints. The machinery developed for partial feedback linearization includes a very compact and straightforward mechanism for reasoning about, and controlling, the couplings between actuated and unactuated joints in robotic manipulators, and has been demonstrated in a swing up task on the Acrobot [Spong, 1994b].

The controllability of arms with passive joints, and a PFL-like control strategy was addressed by [Arai and Tachi, 1991], where a point-to-point controller was developed for a 3 link arm which relies on the passive joints having brakes. This early work demonstrated that if there are more actuators than passive joints, then the passive joints can be controlled to position them in a desired configuration. After this occurs, brakes are used to hold the passive joints in place while the actuated joints are then moved to bring the whole system to a desired configuration. Later, the work was extended using a "time-scaling" algorithm to arms without brakes, but under zero gravity assumptions (e.g. in a horizontal plane) [Arai et al., 1998,Lynch et al., 2000]. The idea of treating the underactuated system as a regular dynamic system with a nonholonomic constraint, and conditions for integrability, and stability were addressed in [Oriolo and Nakamura, 1991]. It was shown that without gravity, an arm with passive joints is not controllable, and with gravity, it is. [Reyhanoglu et al., 1999] also addressed controllability and stabilizability using Spong's PFL framework.

Another body of literature addressed stabilization of such systems around an equilibrium point. See [Berkemeier et al., 1999] for one such technique, and for an overview of related. Alternative approaches such as [Stojic et al., 2000] study a 3 DOF biped with a passive toe joint; feedback control of the horizontal COM acceleration is achieved by only looking at linear terms in the dynamics, using LQR to stabilize around a point. Similarly, [Yamakita et al., 2003] studied a 2 DOF biped with passive toe joint (Acrobot) and controlled the angular momentum. The technique was extend to 3 DOF by ignoring non-linear terms, and used to control landing in a gymnastic robot using feedback control. Lastly, there are also energy based approaches, e.g. [Fantoni et al., 2000], though methods such as [Spong, 1994a] that use PFL also use intuition of energy to achieve swing-up.

1.3.4 Task space control of underactuated systems

Some of the prior work in underactuated control has combined task space control with partial feedback linearization to enable feedback control within task space. The work in [Stojic et al., 2000] and [Yamakita et al., 2003] provide approximations of this approach by dropping nonlinear terms. Arai [Arai et al., 1993] developed a control for the Cartesian coordinates of an end effector for an arm with passive joints. That work assumed the task-space dimension exactly equals the number of actuators, and uses the Jacobian inverse. Point-to-point control (from a given position to a desired position, with zero velocities) is achieved with brakes on the passive joints. [Jain et al., 1993] also developed a similar feedback control via generalized Jacobians (invertible) for free floating systems applied to underactuated systems, which resulted in a computationally efficient approach to finding the equations of motion.

Xu and Gu [Gu and Xu, 1994] developed an adaptive control framework to account for dynamic errors due to parameter uncertainty while controlling Cartesian coordinates, for a class of arms where the Jacobian is assumed invertible. The work also discussed stability and internal dynamics of the workspace control approach, and determined that stability analysis remains an open problem. Bergerman [Bergerman, 1996] extended these works to include concepts in motion planning, though the more recent developments of randomized kinodynamic planners were not developed by that point. The work presents a planner for Cartesian space control for an arm with passive joints with brakes. The motion planning consists of iteratively moving a bit, then applying the brake. He also applied sliding control / robust control to account for model error, and analyzed the actuability of the system. The work in [Shin and Lee, 1997] also developed Cartesian control, with simple path planning to avoid dynamic singularities. This was done by looking at constraints in task space for a two DOF arm. Lastly, [Hyon et al., 2004] developed a controller for a planar (gymnastic) Back-Handspring robot, including COM (Cartesian) control with a passive joint. A state-machine (e.g. Raibert) type controller was used to enable a cyclic gait, and all parameters were tuned by hand.

The prior work in task space control with passive degrees of freedom has largely focused on low-dimensional (2 or 3 DOF) systems. The task space is used because it is intuitive, and may offer some advantage over joint control, but none of these prior works have examined redundancy resolution mechanisms. Furthermore, these works have not addressed the

potential advantage of the dimensionality reduction that is possible for higher dimensional systems, and have not approached the problem as a feasible trajectory search problem.

1.3.5 Templates and Anchors

The concept of "embedding" low-dimensional dynamics into a high-dimensional dynamical system has a long history in control, and robotic locomotion research is no exception (e.g., [Westervelt et al., 2002, Poulakakis and Grizzle, 2007]). Full and Koditschek, [Full and Koditschek, 1999] gave this idea some broad appeal by describing "templates and anchors", and suggesting that the embedding of simple models (the templates) can describe neuromechanical invariants in real animals (the anchors). One of the hallmark examples of this work is the spring-loaded inverted pendulum (SLIP) model, which explains center-of-pressure to center-of-mass dynamics in an incredible diversity of creatures, and also has been used to design control systems for dynamic locomotion in robots [Altendorfer et al., 2000]. Most work in templates and anchors (e.g., [Nakanishi et al., 2000, Saranli and Koditschek, 2003]) use an inverse-dynamics mapping to produce the embedding; consequently when the anchor is underactuated, the template must be chosen carefully to guarantee that dynamics remain invertible.

1.3.6 Legged Locomotion

The problem of fast locomotion over rough terrain has long been a research topic in robotics, beginning with the seminal work by Raibert in the 1980's [Raibert, 1986, Raibert et al., 1986]. Many recent approaches have utilized compliant or mechanically clever designs that enable passive stability using open-loop gaits, or otherwise use reflexive control algorithms that tend to react to terrain. Recent applications of these strategies, for example on the Rhex robot [Altendorfer et al., 2001] and BigDog [Raibert et al., 2008], have produced impressive results, but these systems do not take into account knowledge about upcoming terrain. Planning methods such as the RRT are well developed for kinematic path planning in configuration space focusing on maneuvers requiring dexterity, obstacle avoidance, and static stability, but are not suitable for highly dynamic motions, governed largely by underactuated dynamics. In this thesis, I attempt to bridge this gap, by demonstrating a framework for fast global motion planning that respects kinematic and dynamic constraints. This is a necessary step to enable agile locomotion, combining careful foot placement and fast dynamics, on a position controlled robot.

A popular method to achieve legged robot control is to use a dynamic stability criteria for gait generation or feedback stabilization (see [Pratt and Tedrake, 2005] for review of various metrics). One approach is to keep the center of pressure, or the projection of the Zero Moment Point (ZMP), within the support polygon defined by the convex hull of the feet contacts on the ground. While the ZMP is regulated to remain within the support polygon, the robot is guaranteed not to roll over any edge of the support polygon. In this case, the remaining degrees of freedom can be controlled as if the system is fully actuated using standard feedback control techniques applied to fully actuated systems. By using

some simplifying assumptions, such approaches have been successfully demonstrated for gait generation and execution on humanoid platforms such as the Honda Asimo [Sakagami et al., 2002, Hirose and Ogawa, 2007], and the HRP series of walking robots [Kaneko et al., 2004]. These humanoid robots are designed with large feet; while walking, the robot usually has one or both feet on the ground, so a large support polygon exists either in one foot, or in the area between both feet, during double-support. The ZMP constraint defines differential equations which can be simplified by considering lower-dimensional "lumped" models. One approach models the HRP-2 robot as a cart (rolling mass) on top of a table (with a small base that should not roll over), and applies preview control [Kajita et al., 2003b] to generate a center of mass (COM) trajectory while regulating the ZMP position. The ZMP is only defined on flat terrain, but some extensions can be applied to extend the idea to 3D, including using hand contacts for stability, for example by considering the Contact Wrench Sum (CWS) [Sugihara, 2004].

To achieve motion planning for walking, one approach [Kuffner et al., 2003, Kuffner et al., 2002] uses the RRT to find a collision free path in configuration space, while constraining configurations to those that are statically stable. The robot is statically stable when the center-of-mass is directly above the support polygon, therefore guaranteeing that the robot will not roll over as long as the motion is executed slowly enough. After finding a statically feasible trajectory of configurations, the trajectory is smoothed and filtered, and verified to ensure that the ZMP is always in the support polygon. This approach has been extended to account for moving obstacles and demonstrated on the Honda Asimo [Chestnutt et al., 2005]. An alternative approach is to first generate a walking pattern while ignoring obstacles and collisions, and then use random sampling to modify the gait to avoid obstacles while verifying the CWS constraints [Harada et al., 2007]. These planners are impressive in the dimension of the planning problem they are able to handle. However, the resulting plans are relatively slow and cautious.

A quasi-static planner was also developed to achieve climbing behavior and walking on varied terrain with HRP-2 [Hauser et al., 2008, Hauser, 2008]. Contact points and equilibrium (static) stances, acting as waypoints, are first chosen by using a Probabilistic RoadMap (PRM) [Kavraki et al., 1996]. The planner tries to find paths through potentially feasible footholds and stances, while taking into account contact and equilibrium constraints to ensure that the robot maintains a foot or hand hold, and does not slip. Motion primitives are then used to find quasi-static local motion plans between stances that maintain the non-slip constraints.

Although ZMP and other measures of stability have advanced the state of the art in planning gaits over rough terrain, the stability constraints result in overly conservative dynamic trajectories. Because such control systems do not reason about underactuated dynamics, the humanoid robots do not perform well when walking on very rough or unmodeled terrain, cannot move nearly as quickly as humans, and use dramatically more energy (scaled) than a human [Collins et al., 2005]. Animals clearly do not constrain themselves to such a regime of operation. Much more agile behavior takes place precisely when the system is operating in an underactuated regime, for example when the COP is

outside of the support polygon, or when there is no support polygon, and the legged robot is essentially falling and catching itself. Treating the problem this way, e.g., allowing the legged machine to fall in a controlled way with every step and then knowing how to proceed, by taking another step, is in some sense a more satisfying approach than ZMP (or other dynamic stability) based control, which breaks down if the robot begins to fall. Furthermore, underactuated dynamics might be exploited - for example, a humanoid robot can walk faster and with longer strides by rolling the foot over the toe, rather than constraining the foot to be flat on the ground.

Feedback control of underactuated "dynamic walking" bipeds has recently been approached using a variety of control methods, including virtual holonomic constraints [Westervelt et al., 2003, Chevallereau et al., 2003, Westervelt et al., 2007] and time-varying LQR on a transverse linearization [Manchester et al., 2009]. While these methods demonstrate impressive behavior for a stereotyped gait, they need to be combined with motion planning to take into account information about the environment, in order to enable fast locomotion over very rough terrain.

The DARPA Learning Locomotion project, utilizing the LittleDog robot, has pushed the envelope of walking control using careful foot placement. Much of the work developed has combined path planning and motion primitives to enable crawling gaits on rough terrain e.g. [Rebula et al., 2007,Kolter et al., 2008,Pongas et al., 2007,Ratliff et al., 2009]. Our team's approach initially utilized heuristics over the terrain, to assign a cost to potential footholds. A* search was then applied to find feasible stances over the lowest cost footholds, and a ZMP based body motion and swing-foot motion planner was used to generate trajectories to walk over rough terrain [Shkolnik et al., 2007]. In the later phases of the project, we began to integrate dynamic lunging, to move two feet at a time [Byl et al., 2008].

1.4 Contributions

The primary idea in this thesis is to intelligently guide search so that a sample-based motion planning algorithm can more efficiently find plans in high dimensional dynamic systems. The contributions of this thesis are summarized below. Contributions (1,3,5) are more general, and can be applied toward a variety of motion planning problems. The methods developed for these three contributions may be utilized on their own, or in combination with each other.

1. Task-space biasing: To reduce sensitivity to system dimension, a Voronoi bias in a low-dimensional task space can be used to improve search efficiency for some problems. This idea is applicable in general to kinematic path planning problems, or kinodynamic motion planning problems.

2. Task-space control on a four legged quadruped platform is explored, in conditions where the robot is assumed statically stable. The choice of redundancy resolution is shown to have an important effect on walking stability.

3. Task-space control is derived for underactuated systems by using partial feedback linearization. The derivation presented enables a task space Voronoi bias for systems that are underactuated, complementing contribution (1). Furthermore, the derivation includes provisions for optimal hierarchical redundancy resolution, the importance of which was demonstrated in contribution (2).

4. A polar representation of center of mass is suggested as a task space mapping for multi-link arms with passive joints. Task space control is applied within a planning framework on this system to achieve swing-up motions. Mathematically, this type of multi-link model can represent a quadruped standing on two legs, or a biped rolling over its foot.

5. Reachability Guidance: The Reachability-Guided RRT is developed to enable efficient randomized search for problems with differential constraints by changing the sampling distribution to account for local reachability of the tree.

6. A combination of Task-space biasing and reachability guidance is demonstrated to efficiently plan bounding motions over rough terrain for the LittleDog quadruped robot.

1.5 Outline

The thesis is generally divided into two sections. Following this introduction, chapters 2-4 deal with task space control, particularly with respect to integrating task space control in a motion planning framework, where the task space effectively reduces the dimensionality of the search to improve search efficiency. Chapter 5 describes the Reachability-Guided Tree to improve search efficiency in the presence of differential and other kinodynamic constraints. In chapter 6, techniques from task space biasing and reachability guidance are applied toward a motion planner for controlling the LittleDog robot in a dynamic bounding task over rough terrain.

More specifically, the chapters have the following organization:

Chapter 2: In this chapter I develop the Task-space biased RRT algorithm which explicitly alters the RRT sampling strategy to be in a lower dimensional task space, making the planning more efficient. The efficiency is improved by introducing a Voronoi Bias in the task space. The algorithm is demonstrated on an N-link arm in a 2D workspace, with the path-planning task of moving the end of the arm to a goal point in the workspace while avoiding obstacles. The number of nodes explored by the algorithm was empirically shown to remain approximately constant in N, allowing the algorithm to solve this problem with up to 1500 dimensions in less than a minute, compared to the standard RRT implementation which could only solve < 20 dimensions in the same amount of time, implying that the task space bias can significantly improve search efficiency for some problems.

Chapter 3: hierarchical task space control of the Center of Mass and Swing Leg position of a quadruped robot is explored. Because the quadruped has more degrees of freedom than

the task space, there are infinite solutions to this control problem. A whole-body Jacobian is derived for the quadruped, allowing optimization within the null space to pick an appropriate control from the infinite possibilities. The ZMP is shown to more closely follow the COM projection when minimizing body rotations through the null space, allowing the robot to execute COM and swing-leg movements more precisely and more quickly without falling over. The implication is that the null space associated with a task-space Jacobian can be used to improve performance. In this chapter, static stability is assumed, which requires 3 or 4 of the legs to be on the ground at a time. In this way, classical feedback linearization techniques are applied, treating the robot as if it is fully actuated.

Chapter 4: task space control is extended to the case for when a system is underactuated. This can happen, for example in a quadruped when only 2 feet are on the ground, or in a humanoid when the foot is rolling (over the toe). A derivation for a task space controller for underactuated systems is achieved by combining task space control in acceleration space with partial feedback linearization. The resulting derivation can be generally applied to underactuated systems which have more control degrees of freedom than the dimension of the task space. Extending prior work, this derivation also cleanly presents a mechanism for optimal redundancy resolution, while also demonstrating rank conditions for when the controller can be used. When the task space consists of directly controlling the passive Degrees of Freedom of the system, the rank condition is the same as the Strong Inertial Coupling condition for pure partial feedback linearization.

Chapter 5: the Reachability Guided Rapidly exploring Random Tree (RG-RRT) algorithm is described. The standard RRT algorithm typically relies on the Euclidean distance metric to create a Voronoi bias in Euclidean Configuration space, when in fact a dynamic distance indicating the cost to get from one point in state space to another point in state space should be used. However, such a cost function is impossible to compute precisely for most problems, and is expensive even to approximate. The RG-RRT addresses this by using "local reachability" to guide the sampling when building the tree. More precisely, as the Tree is being constructed, the algorithm keeps track of regions which can easily be reached from the current tree. In the algorithm, random samples are drawn from a uniform distribution in state space, but samples that are closer to the tree rather than to the reachable set of the tree are discarded. This effectively dynamically shapes the sampling distribution to produce a significant improvement in RRT motion plan search efficiency under differential constraints, as demonstrated on toy problems including the Pendulum, Acrobot, and Simple Dynamic Car Model.

Chapter 6: task space biasing and reachability guidance are combined to demonstrate motion planning in application to quadruped bounding over rough terrain. This chapter elaborates on motion primitives in their relation to Reachable sets. A simple motion primitive is presented for the LittleDog robot. This motion primitive creates a task space bias, and is easily conformed to the reachability framework. The resulting motion planner is able to plan bounding motions for this small quadruped over rough terrain including steps and logs.

Chapter 7: concludes the thesis, and provides some future directions.

Chapter 2

Path Planning with a Bias in Task Space

The reduction of the kinematics and/or dynamics of a high-dimensional robotic manipulator to a low-dimension "task space" has proven to be an invaluable tool for designing feedback controllers. When obstacles or other kinodynamic constraints complicate the feedback design process, motion planning techniques can often still find feasible paths, but these techniques are typically implemented in the high-dimensional configuration (or state) space. A Voronoi bias in the task space can dramatically improve the performance of randomized motion planners, while still avoiding non-trivial constraints in the configuration (or state) space. This idea is explored in detail in this chapter. The potential of task space search is demonstrated by planning collision-free trajectories for a 1500 link arm through obstacles to reach a desired end-effector position.

Significant portions of this chapter also appear in [Shkolnik and Tedrake, 2009].

2.1 Task Space in the Context of Path Planning

Humans and animals have hundreds of bones and muscles in their bodies. State of the art planning approaches, such as the RRT, can plan for systems with a moderate number of dimensions, but to achieve animal-like agility, it may become necessary to plan in higher dimensional systems. In this Chapter, I demonstrate that fast planning in very high dimensional systems can, at least under certain conditions, be achieved by exploring actions in a lower dimensional task space. This approach is commonly used in feedback control design, and is often called Task Space Control [Liegeois, 1977] (see section 1.3.2). Consider a humanoid robot which needs to find its way from a start pose to cross a room full of obstacles, and pick up an object [Kuffner et al., 2001]. For such problems, it is natural to consider the task space. Rather than specifying a particular set of desired joint angles for the whole robot which accomplishes a grasping task, one may consider only the hand position. There are an infinite number of potential goal configurations which correspond to the desired hand position. Having a large set of goals can sometimes make planning easier, as there will be more trajectories that lead to a goal configuration. Furthermore, the defined task space

can be used to guide the search, even when planning still occurs in the high-dimensional space in order to enforce all of the system constraints.

There are several recent works that utilize task space in the context of motion planning. In [Bertram et al., 2006], for example, the sampling of points is biased by a heuristic function which includes distance to the goal, so that points closer to the goal in task space are more often randomly selected. The selected nodes are then extended with a randomly applied action. In a variation of this work by [Weghe et al., 2007], a standard RRT is implemented which occasionally chooses to grow toward the goal in task space by using the Jacobian Transpose method. The goal of that work was to solve the problem of inverse kinematics at the goal region because infinite configurations may correspond to a single point in task space. Recently, [Diankov et al., 2008] proposed an approach that enables bidirectional RRT search [Kuffner et al., 2000], where a backward tree (grown from the goal) is grown in task space, and a forward tree (from the start pose) is grown in the full configuration space. Occasionally, the forward tree attempts to follow the backward tree using either random sampling of controls or the Jacobian transpose. These works utilize task space as an intuitive way of defining an infinite (continuous) set to describe a goal position, rather

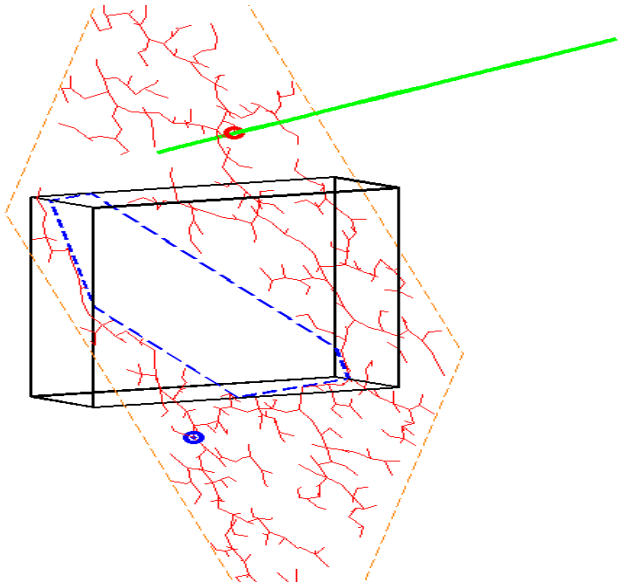


Figure 2-1: Simple example, demonstrating an RRT planning in a 3D configuration space, while constrained to actions in a 2D task space. The task depicted is to find a path from the starting node (blue circle), to the goal in task space (green line), while avoiding an obstacle (Black Box). For simplicity, one may resolve redundancy by restricting the search to lie in a plane (shown by red dash) perpendicular to the goal line, but other suitable choices for redundancy resolution would not be constrained to a manifold. The intersection of the obstacle on this plane is shown in blue dash line. For problems like this, planning in a lower dimensional projection can be much more efficient than planning in the full space. ©2009 IEEE [Shkolnik and Tedrake, 2009].

than using task space as a potential way to constrain the exploration of the full space. Another recent work, [Zucker et al., 2008] uses reinforcement learning to bias search based on features in the workspace. Similar to [Diankov et al., 2008] and [Zucker et al., 2008], the exploring/exploiting tree in [Rickert et al., 2008] also uses workspace obstacle information to guide the planner search. In probabilistic road maps (PRMs), some works demonstrate that workspace information can be taken advantage of by sampling along the medial axis, or the subset of points in the workspace that are furthest from obstacles [Holleman et al., 2000], or by dynamically sampling in regions by using information of workspace geometry while taking into account the past history of sampling [Kurniawati and Hsu, 2006].

Note, that in many of these approaches, the planner takes advantage of collision information in the workspace. This is not always possible however, and the work presented in this chapter does not require this property. The benefits of this will be most evident in later chapters, where we use robot COM as a task space, and there is little collision information that is relevant in that space.

Another body of literature has focused on sample based planning algorithms for systems that are constrained to a manifold defined by a lower-dimensional workspace, for example for a robot that is writing on a wall, it must keep the end effector on the surface of the wall [Tang et al., 2006]. In that work samples are constructed in configuration space that meet the constraints imposed by the lower-dimensional manifold, but no attention is given to how the particular choice of sampling affects planner performance, which is important given that there may be infinite ways to construct such a sampling schema. In the ATACE framework [Yao et al., 2005], configurations are sampled uniformly in C-space, while a workspace constraint is imposed by utilizing a Jacobian pseudoinverse method. A similar C-space sampling based method with task space constraints is provided by [Stilman, 2007]. The method I present is different in that samples are obtained directly within task space. Doing so creates a Voronoi bias in task space, rather than in Configuration Space, which for many systems leads to more direct exploration. Another difference is that the method presented here does not operate on a lower dimensional manifold, but instead considers the projection into task space. Because of this, the method described here could search the full high dimensional C-space, despite the attention to exploring in the lower dimensional task space.

Constraining a search to a lower dimensional space can be a powerful notion. For visualization, consider for example the task of planning in a 3D configuration space. The goal might be to find a path from a starting coordinate to some region of the space, while avoiding obstacles. Suppose the goal region is a line. Then one way to explore the space is to grow in the plane which is perpendicular to the line, and intersects the starting node (as shown in Figure 2-1). If there were no obstacles present, this type of search could produce the most direct (shortest) path in configuration space. Because this plane slices through the 3D configuration space, as long as the plane has a path from start to goal, then it would make sense to actually plan while constrained to this 2D plane, which drastically reduces the set of potential solutions that need to be explored. Now imagine if rather than a 3D configuration space, there is a large DOF system. If a plane can be defined which captures a large portion of the feasible set of trajectories from start to goal, then it would be much

more efficient to plan within this plane rather than the full configuration space. A potential objection would be that if the planning problem can be solved in this low-dimensional space, then it is not a challenging planning problem. However, I would argue that in fact many of the success stories in whole-body motion planning have this property.

This chapter proposes a framework for task space biased exploration for search algorithms. A standard forward RRT [LaValle, 1998] is modified with a task space Voronoi bias. This is accomplished by sampling in the low dimensional task space, and then growing the tree towards the sample in task space. The growth can be done efficiently using the Jacobian Pseudoinverse ($\mathbf{J}^+$) method, commonly used in feedback task space controllers, which eliminates the need for exact inverse kinematics solutions. This task space will have an associated null space, the use of which is left to the freedom of the algorithm implementation. The pseudoinverse method allows for redundancy resolution via a hierarchical control structure which takes advantage of the null space. One can think of the standard RRT as randomly exploring in the null space. However, when actions are constrained (which is usually the case), then as the dimensionality increases, it is increasingly likely that a random vector in configuration space is orthogonal to a direction in task space, producing a trade off between exploring directly in task space, or exploring the configuration space.

If random samples are drawn from a uniform distribution over the task space, and a euclidean distance metric is used for the NearestNeighbor() function, then the proposed algorithm will have the property that the Voronoi bias associated with the RRT will be in task space instead of in configuration space. In fact, these two spaces are quite different, and for certain problems, the Voronoi bias in task space can significantly improve the search performance. The algorithm is demonstrated for path planning a collision free trajectory in an N-link arm, with the task of putting the end-effector at a desired coordinate. The resulting planner is able to efficiently plan paths with up to 1500 degrees of freedom in under a minute.

The proposed method can be thought of as planning in task space, while checking and validating constraints in the full configuration (or state) space. The forward RRT was chosen for its simplicity, but this method can be extended to a variety of planning algorithms, including PRMs [Kavraki et al., 1996], bidirectional RRT's, e.g. [Kuffner et al., 2000] - either by implementing BiSpace [Diankov et al., 2008], or directly if goal(s) are known in configuration space.

2.2 TS-RRT: Planning in Task Space with the RRT

2.2.1 TS-RRT Definitions

In this chapter, a vector in configuration space is denoted with $\mathbf{q} \in \Re^N$, and a vector in task space with $\mathbf{x} \in \Re^M$, with an associated task space mapping $\mathbf{x} = f(\mathbf{q})$, and a Jacobian defined as $\mathbf{J} \in \Re^{M \times N} = \frac{\partial f(\mathbf{q})}{\mathbf{q}}$.

2.2.2 TS-RRT

The proposed planner, given in Algorithm 3, is called Task Space RRT (TS-RRT). It is similar to the standard RRT approach, described in Algorithm 1 in section 1.3.1, except that a random sample, $\mathbf{x}_{rand}$ is generated directly in task space, typically using a uniform distribution over a bounded region. The TS-EXTEND() function, shown in Algorithm 4, is similar to the standard RRT EXTEND() function in Algorithm 2, but works in task space rather than C-space. The TS-NEAREST-NEIGHBOR() function selects the nearest point on the tree in task space. Each node on the tree contains both C-space and task space information, so the computation of TS-NEAREST-NEIGHBOR() would be less than the comparable function in a standard RRT. A full configuration space action is generated in TS-CONTROL(). Ideally, this control attempts to grow the nearest node of the tree towards the random sample in task space. There are many ways of implementing this type of control, and the action can even be random, as done in [Bertram et al., 2006]. A more

Algorithm 3 BUILD-TS-RRT (q_0)

T.init(q_0)
for $k = 1$ to K **do**
  if with some probability P **then**
    $x_{rand} \Leftarrow$ RANDOM-SAMPLE() $\in \Re^M$
  else
    $x_{rand} \Leftarrow x_{goal}$
  end if
  TS-EXTEND(T,x_{rand})
end for

Algorithm 4 TS-EXTEND (T, x)

$[q_{near}, x_{near}] \Leftarrow$ TS-NEAREST-NEIGHBOR(x,T)
$u \Leftarrow$ TS-CONTROL($q_{near}, x_{near}, x_{rand}$)
$q_{new} \Leftarrow$ NEW-STATE(q_{near}, u)
$x_{new} \Leftarrow f(q_{new})$
if COLLISION-FREE (q_{near}, q_{new}) **then**
  T.add-node($q_{near}, q_{new}, x_{near}, u$)
end if

Algorithm 5 Example TS-CONTROL($q_{near}, x_{near}, x_{rand}$)

$\mathbf{u}_{ts} \Leftarrow (\mathbf{x}_{rand} - \mathbf{x}_{near})/\Delta t$
$\mathbf{u}_{ts} \Leftarrow$ CROP-WITHIN-LIMITS($\mathbf{u}_{ts}$)
$\mathbf{J} \Leftarrow$ COMPUTE-JACOBIAN($\mathbf{q}_{near}$)
$\dot{\mathbf{q}}_{ref} =$ SECONDARY-CONTROL($\mathbf{q}_{near}, \mathbf{x}_{rand}$)
$\dot{\mathbf{q}} \Leftarrow J^+ \cdot \mathbf{u}_{ts} + \alpha(\mathbf{I} - \mathbf{J}^+ \cdot \mathbf{J}) \cdot \dot{\mathbf{q}}_{ref}$
return $\Delta\mathbf{q} \Leftarrow \dot{\mathbf{q}} \cdot \Delta t$

efficient approach is to use the Jacobian pseudoinverse in a feedback type controller, as shown in Algorithm 5). A new state in C-space, $\mathbf{q}_{new}$ is integrated using the same NEW-STATE() function as in the RRT. After checking for collisions in the full C-space, $\mathbf{q}_{new}$, its task space mapping $\mathbf{x}_{new}$, and the control action are added to the tree.

Because the sampling occurs in task space, the Voronoi bias when growing the tree is now in this space, rather than in the configuration space. This encourages the planner to quickly explore empty regions within task space, which results in more efficient planning, assuming paths in this space exist and there are no narrow channels, using the selected task space mapping. Approaches for ensuring completeness are discussed in section 2.4.2.

The growth mechanism in Algorithm 5 allows for a secondary command, $\dot{\mathbf{q}}_{ref}$, to be defined in terms of the configuration (state) space. If left as zero, then this controller will simply take the shortest path within the configuration (state) space. However, this function can be used to encourage the system to go toward certain configurations as much as possible, and can be similar in spirit to the workspace centering (e.g. [Sentis and Khatib, 2005]), used in the more typical feedback based task space control.

2.3 Scalability of TS-RRT compared to RRT

2.3.1 Path Planning for the N-Link Arm

In this section, a kinematic search problem is explored, in which an N-link arm needs to find a suitable trajectory from a given start pose, with the task of putting its end effector at a particular coordinate. Figure 2-2 demonstrates a 5-link version of this problem. The task space utilized is the Cartesian coordinate of the end effector. This path planning problem does not have dynamics, and therefore does not have any second-order effects including gravity and torque limits. However, the joints are limited to move less than .05 radians per each Δt, and obstacles are imposed within the Cartesian plane, which are checked for collisions against all links. Furthermore, joint limits of +/- 2.5 radians were imposed on all links. This problem was explored with a constant set of obstacles, shown in red in Figure 2-2. A constant start pose (all angles = 0), and a constant desired goal pose were used. The number of links, N, is variable, with the link length set to 1/N so that the total arm length is always 1.

Two algorithms were explored on this problem. First, a standard forward RRT was implemented. Because there are infinitely many potential goal poses which correspond to the goal task in this problem, the configuration space was randomly sampled until 20 poses were found that were collision free and whose end effector positions were close to the desired position. The RRT Algorithm 1 was then run, with samples drawn uniformly in configuration space ($\in \Re^N$ with bounds of +/- pi). With probability P=.1, one of the 20 goal configurations was chosen at random to be used as the sample to grow the tree toward. A Euclidean metric (all joints weighted equally) was used in NEAREST-NEIGHBOR. Node growth occurred by computing the vector between a sample and the closest node on the tree, and cropping the components of this vector to be within +/- .05 radians.

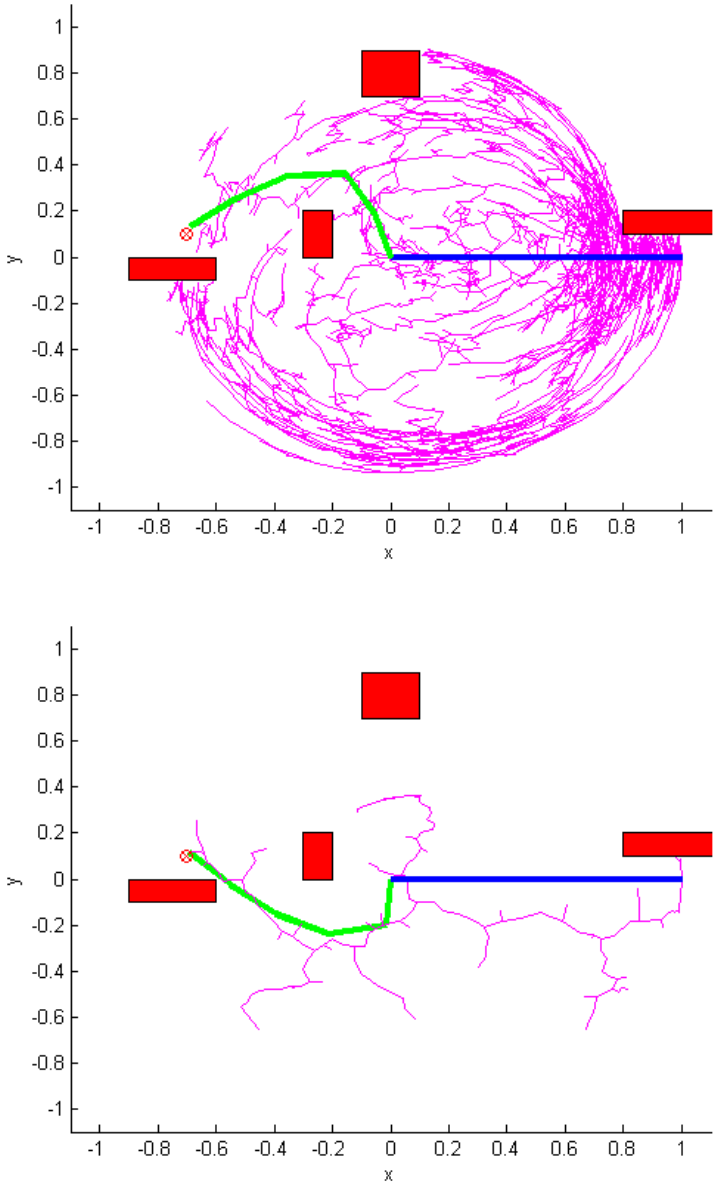


Figure 2-2: Projection of RRT into End Effector Cartesian Coordinates for a 5-link arm (shown in Magenta). The starting configuration ($[0, 0, 0, 0, 0]^T$) is shown in Blue, and the achieved goal configuration is shown in green. **TOP**: Standard RRT search was performed in the 5-D configuration space. This tree has 2000 nodes. Obstacles in the workspace are shown in red, with collision checking performed for 10 evenly spaced points along each link. **BOTTOM**: the resulting RRT using TS-RRT is shown, where exploration is constrained to the 2D task space. This tree has about 150 nodes. ©2009 IEEE [Shkolnik and Tedrake, 2009].

In addition to the standard RRT, a forward TS-RRT was run on the same problem. The Jacobian pseudoinverse was used to calculate a desirable change in q to grow a selected node toward a sample in task space. The null space reference command utilized was set to $\Delta\mathbf{q}_{ref} = -q$, which encourages the arm to straighten when possible. $\Delta\mathbf{q}$ was normalized so that the max norm was .05. Random samples were taken uniformly in task space ($\in \Re^2$ with bounds of +/- 1.1), and a Euclidean distance metric was used in task space to select the closest nodes on the tree. With probability .1 the goal task was chosen as the sample. Most of the code between the two implementations was shared, with an attempt to minimize tuning toward each problem / implementation.

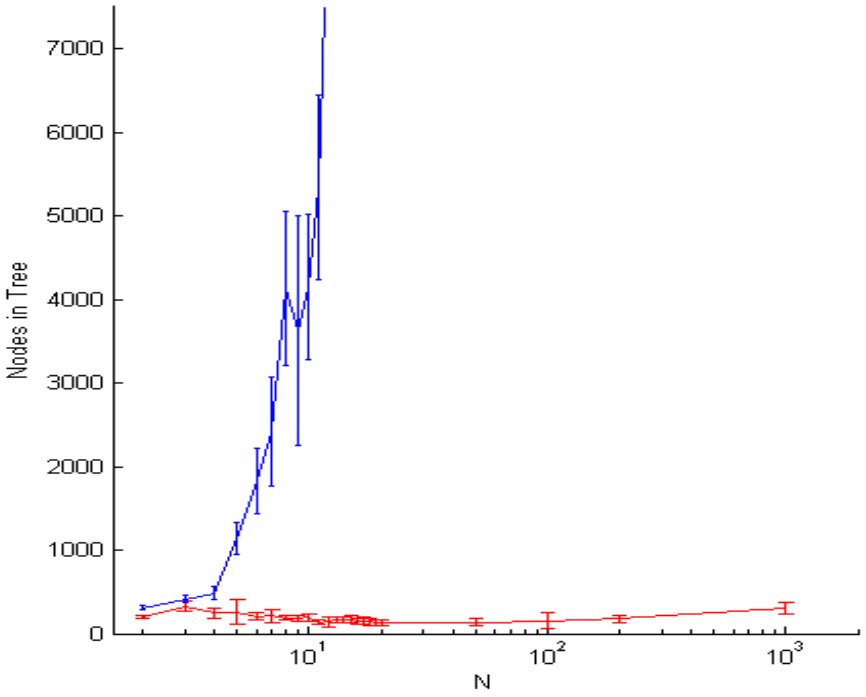


Figure 2-3: The number of nodes that the RRT algorithm searches before finding a feasible trajectory for the problem shown in figure 2-3 is shown as a function of N, the number of dimensions. The obstacles and goal task are held constant, and the link lengths are set to 1/N so that the total arm length remains constant. Note that the **x axis is on a log scale**, starting with N=2, going up to N=1000. **RED**: TS-RRT, which remains approximately constant. **BLUE**: full configuration space RRT, which grows super-linearly. ©2009 IEEE [Shkolnik and Tedrake, 2009].

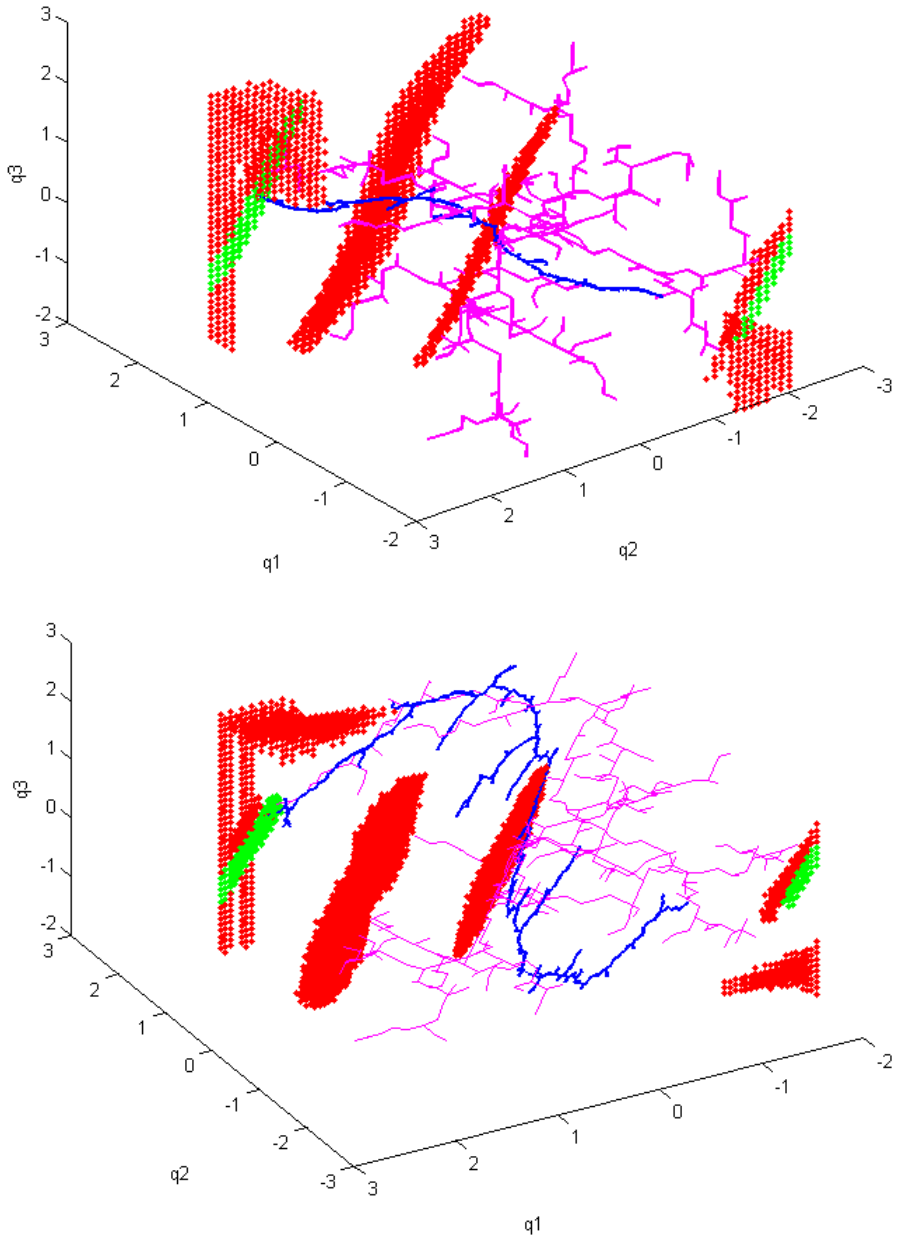


Figure 2-4: **TOP/BOTTOM** are two rotated views of the same 3D plot. This figure shows the configuration space view of an RRT tree (magenta), and a TS-RRT (blue), both for a 3-link arm. Obstacles (same as those shown in Figure 2-2) are shown by red dots, and the goal task point is shown by green dots in the configuration space. The standard RRT has good coverage in the space, as expected. Both trees converge to a collision free trajectory, but it clear that the TS-RRT is much more direct. ©2009 IEEE [Shkolnik and Tedrake, 2009].

2.3.2 Results

Each RRT was run 20 times, while varying the number of links (N) in the arm from 2 to 1000 (TS-RRT), and from 2 to 15 (RRT). The median number of nodes explored by each tree is shown in Figure 2-3, as a function of N. The standard RRT quickly started exploring thousands of nodes as N increased above 4, while the TS-RRT exploration remained approximately constant. Of course, the Jacobian pseudoinverse and null space calculations are expensive to compute, but in my implementation, using standard Matlab functions, they added an overhead of 50 percent to the time needed to grow a node, including collision checking. As an example of the actual time to compute paths, the TS-RRT algorithm took 4-8 seconds with N = 200 (approximately 200 nodes), which was comparable to the standard RRT when using N = 6 (approximately 1500 nodes). Note, the time difference per node is due mainly to increased computation for collision checking in higher dimensions, rather than to the Pseudoinverse computation.

A comparison of an example of trees produced by each algorithm, with N=5, projected into task space, is shown in Figure 2-2. It is clear that the TS-RRT searches more directly and evenly within task space. A comparison of the trees produced, shown in the full configuration space (for N=3), is shown in Figure 2-4. The figure shows two different views of the same two trees, the blue corresponding to TS-RRT, and the magenta being the standard RRT. The rectangular obstacles in task space correspond to large cylindrical (pancake-like) obstacles in configuration space, shown by red dots in the figure. The goal task point is denoted by green dots in the figure. The standard RRT produces a tree which seems to evenly spread out in the 3D configuration space, as expected. The TS-RRT, on the other hand, is more surface-like, but is able to navigate the obstacles.

The standard RRT could only run up to N=15 in a reasonable time (approx. 20 minutes to solve). However, the TS-RRRT is able to find trajectories with N = 1500 in less than a minute. Two example trajectories (and trees) are shown in Figure 2-5. In one trajectory, the arm bends, and then changes the direction of the bend part way through. In the other trajectory (bottom), the arm actually tied a knot through itself in order to reach the goal.

Note that in order to handle singularities, the maximum magnitude of angular displacement, $\Delta\mathbf{q}$, is restricted. One can see that the starting pose is singular, but because the planner is sample based it tends to immediately break out of any singular regions. More acute examples of the influence of singular regions can be seen at the extremes of the workspace in Figure 2-5. Towards the edges of the reachable workspace, the TS-RRT becomes slightly irregular, with some nodes being selected repeatedly and expansion restricted. More clever handling of singularities is possible as is done in standard feedback control, but was not required for the task presented here as the system performance was still quite good.

2.4 Extensions and Possible Modifications

The algorithm is amenable to various modifications, and problem specific implementations.

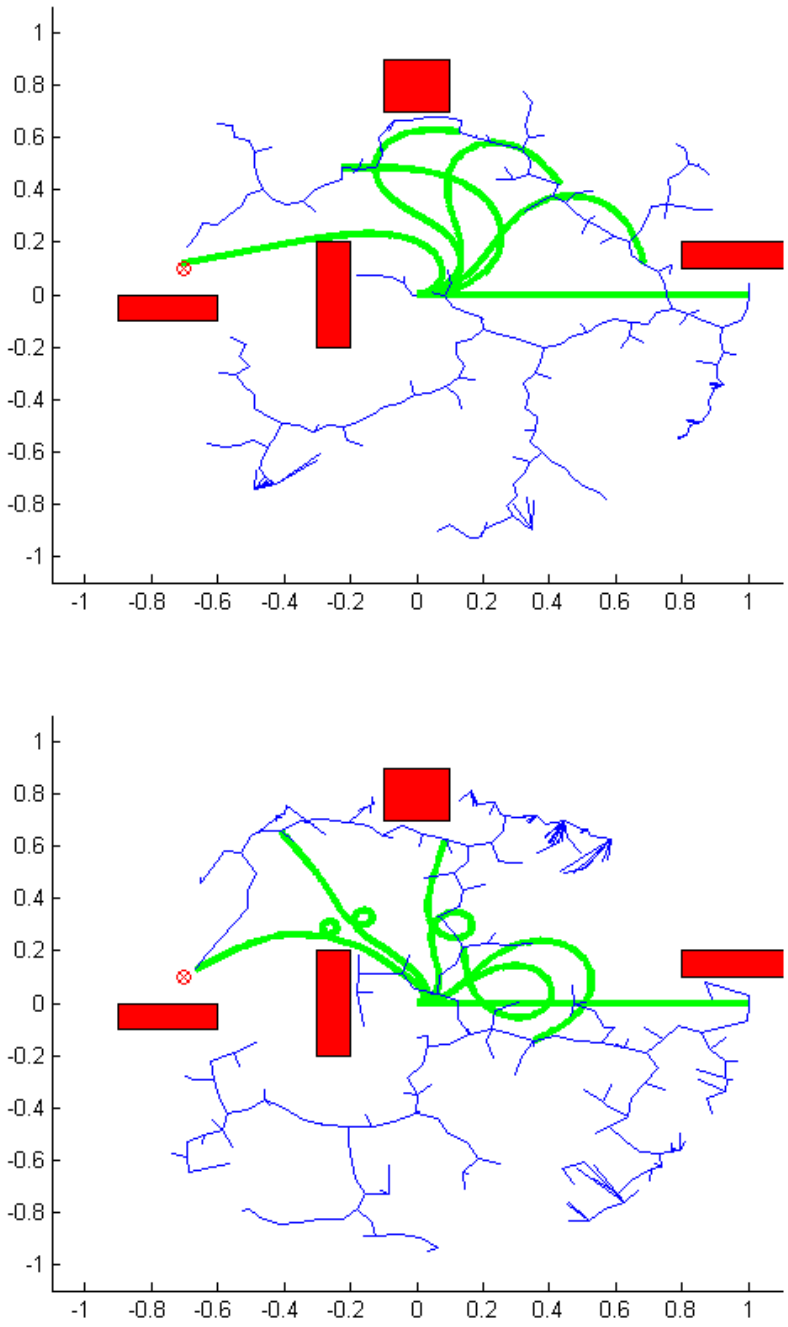


Figure 2-5: Two example trajectories and RRT plans, found with N = 1500. Total computation time was less than one minute. ©2009 IEEE [Shkolnik and Tedrake, 2009].

2.4.1 Dynamic and Underactuated Systems

It is possible to extend the TS-RRT algorithm to dynamic systems where new states are generated by integrating the dynamics forward. The torque (or action) used can be computed once per step, or incrementally, where $\mathbf{u}_{ts}$ is recomputed inside the integration loop as $\mathbf{x}_{near}$ changes. Algorithm 5 utilized a pseudoinverse Jacobian based velocity controller, but any preferred method for computing velocity or acceleration based inverse kinematics ([Nakanishi et al., 2005] for review) can be utilized and integrated forward to generate a new state given a desired point in task space. An example controller is analyzed in the next chapter, where a COM task space controller is derived for LittleDog under the assumption that the robot is quasi-fully actuated.

This method may also be extended to control underactuated systems, if a carefully chosen task space is used. This idea is explored in chapter 4, which demonstrates swing-up control of a 5-link pendulum with up to 3 passive joints, by controlling the center of mass of the system. Potential applications for planning in high dimensional dynamic systems are far reaching, and because the task space control allows for hierarchical control (e.g. the secondary control $\dot{\mathbf{q}}_{ref}$ is only applied in the null space of $\dot{\mathbf{x}}_{ref}$, but $\dot{\mathbf{q}}_{ref}$ may have been computed from another task space with a null space, allowing for a tertiary controller, and so on). An example of this might be planning humanoid grasping movement, by using the Cartesian coordinate of the hand as a primary task, the orientation as a secondary task, and a desirable grip posture (e.g. to leave fingers open) as a a tertiary task. As another example, in the next chapter Center of Mass control is used as a primary task, with swing-foot position as a secondary task to control a quadruped while walking.

2.4.2 On Completeness

The obvious drawback to exploring directly in task space is that the planner may lose completeness, in the sense that it may no longer be possible to guarantee that a path will be found, given that the set of trajectories which can be explored has been severely restricted. Thus the choice of task space is critical, as is the choice of redundancy resolution (e.g. selection of $\dot{\mathbf{q}}_{ref}$). If a task space can encode actions with enough breadth such that feasible solutions exist, it remains unnecessary to explore in the full state space. If a solution exists within the task space, given the redundancy resolution mechanism employed, then this planner will still remain probabilistically complete. Additionally, depending on the redundancy resolution mechanism, taking an action (in task space) in one direction and then taking the reverse action can have the effect that the system returns to the original point in task space, but to a different point in state space. As such, it may be possible that all points in state space are reachable with some action sequence, implying that the proposed algorithm is still probabilistically complete.

The TS-RRT algorithm presented can also be modified so that samples are chosen in configuration space, as in the standard RRT. The standard NEAREST-NEIGHBOR function would be used, so that a configuration space metric is used to evaluate and find the closest node on the tree. The remainder of the algorithm is the same, where a node chosen for

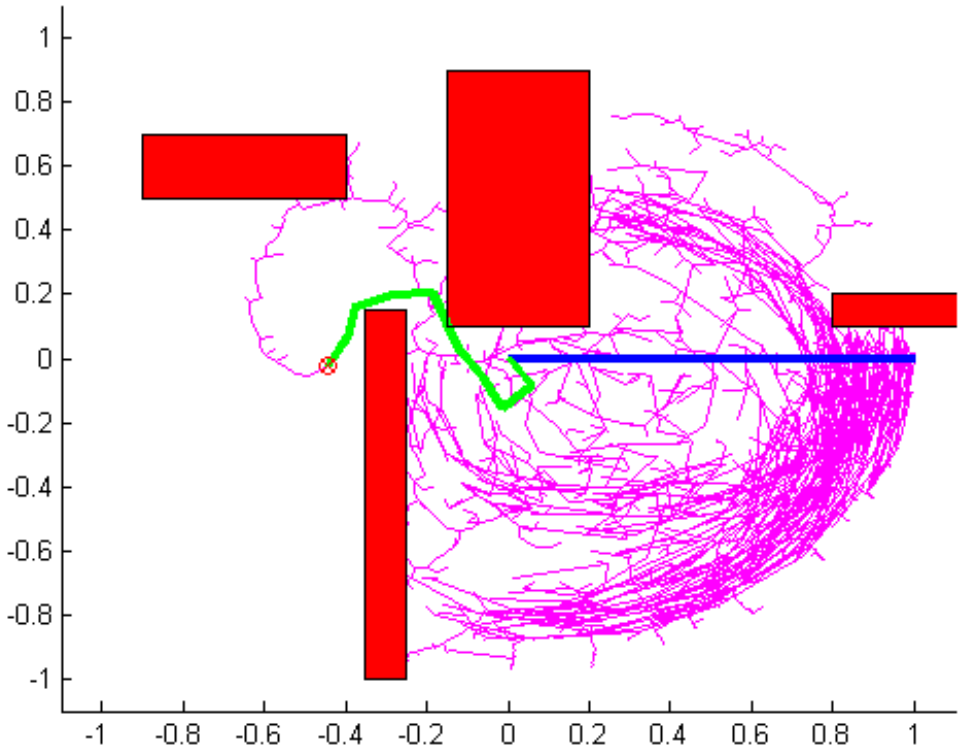


Figure 2-6: With more challenging obstacles, a hybrid approach can be used to search efficiently by exploring directly in task space part of the time, and directly in configuration space the remaining time. ©2009 IEEE [Shkolnik and Tedrake, 2009].

expansion grows toward the sample in task space. Note, it is possible to augment or define the $\dot{\mathbf{q}}_{ref}$ term in Algorithm 5, the secondary task in the control function, with: $(\mathbf{q}_{rand} - \mathbf{q}_{near})/\Delta t$. In such a case the system will try to move toward the sample in configuration space, making the algorithm more similar to the standard RRT. By tuning α, growth can be biased to explore more in task space, or to explore more in configuration space, like the standard RRT. However, it is important to realize that a drawback to such an approach is that the Voronoi bias within task space will be disrupted by the sampling within configuration space, because uniform sampling in one space usually does not map to a uniform sampling in the other space.

2.4.3 Hybrid Approach

Because book-keeping is virtually the same for both the RRT and TS-RRT algorithms, they are quite interchangeable, and in fact, one can combine both approaches by searching in configuration space with some probability, while searching in task space the remaining time.

A variation of may be to search in task space until it is detected that the tree is not expanding, at which point the algorithm can start to sample in configuration space. Any sampling in C-space ensures probabilistic completeness, while sampling in T-Space allows for faster, more direct exploration when possible. Figure 2-6 demonstrates the resulting RRT which used such a hybrid approach. Nodes were expanded using either RRT or with TS-RRT with a probability P = .5, to solve a difficult trajectory problem for a 10 link arm with obstacles that define very narrow passable channels. The solution in this configuration is typically found after exploring fewer than 5000 nodes. Interestingly, I was unable to find a trajectory for this problem setup if using exclusively either the standard-RRT or the non-augmented TS-RRT, even after 200,000 nodes of exploration.

In the worst case, if all extend operations in C-space are inefficient (approximately perpendicular to all solution trajectories), then the maximum slowdown of this approach is a factor of $< 1/P$ compared to searching in T-space alone. However, this may be a small price to pay in order to achieve probabilistic completeness. Generally, the more challenging the problem, defined by the degree of complexity of obstacles (e.g. narrow obstacles), the more it will benefit to explore more often in C-space. A more thorough analysis of such a hybrid approach is left to future work.

2.5 Concluding Remarks

Task Space Control [Liegeois, 1977] is widely used in feedback control, but has been less used in the planning community. For some classes of problems, a carefully chosen task space may improve planning performance because a Voronoi bias encourages exploration of empty regions in task space, rather than in the full configuration space, which allows solutions to be found more directly. It is not unreasonable to assume that problems that can be solved this way in low dimensions can be scaled to higher dimensions in approximately linear time, as the number of nodes needed to find a path stays approximately constant with N, and only the time to check collisions and to compute the Jacobian pseudoinverse increases. Although this chapter was specifically targeted toward modifying the RRT algorithm in particular, the concept of task space exploration may be a powerful tool to enable a variety of planning algorithms to work in higher dimensions. Furthermore, the applications are numerous, especially given that 1) many real world systems are high dimensional and 2) It is often natural to specify low dimensional tasks for many problems.

Chapter 3

Task Space Control for a Quadruped Robot

The previous chapter showed that task space can be useful in the context of efficient motion planning. Task space control has been well studied for fully actuated systems such as robotic arms, however legged robots are underactuated and present additional challenges for implementing task control. This chapter explores task control for Center of Body (COB), Center of Mass (COM), and swing leg foot position for the LittleDog robot, essentially laying a task control framework for generating a walking gait on arbitrary terrain. The method presented can be generalized to any legged robot. I derive whole body Jacobians that can be used to control the robot in these task spaces. The Jacobian based task control presents a natural framework for resolving redundancies, but it is up to the control designer to choose how exactly to do that. A typical approach for redundancy resolution is known as workspace centering, which tries to maintain a desirable posture as much as possible, while executing a task. In this chapter, I explore several redundancy resolution schemes, while executing task space movements. For simplicity, the robot COM is constrained to be over the support polygon, which guarantees static stability, that is when the robot is not moving it will not fall over. However, when executing motions in task space, for example a body movement, the choice of redundancy resolution becomes very important. One choice we consider minimizes body pitching and rolling, which maximizes the utility of the static stability metric, allowing the robot to make faster motions in task space, without falling over.

This chapter derives task space control in some technical detail. The implication for motion planning is that when using Jacobian based methods, which are naturally supportive of hierarchical control, the secondary or tertiary control (which is responsible for redundancy resolution), impacts the ability of the dynamic system to feasibly execute task space motions. Thus, if using task space control in motion planning, care needs to be taken to ensure that redundancy is resolved in the best interest of the planner.

Significant portions of this chapter also appear in [Shkolnik and Tedrake, 2007].

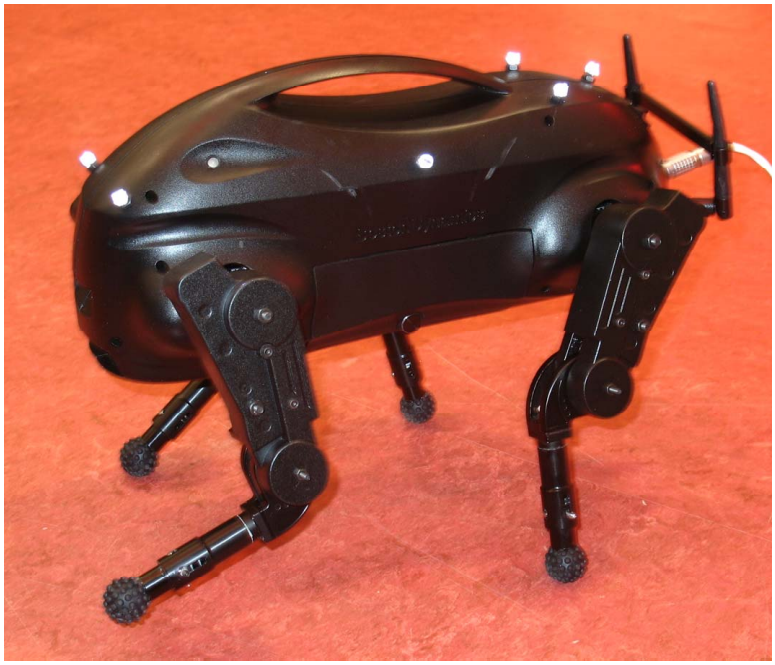


Figure 3-1: **Littledog Robot** A picture of the LittleDog robot, executing the task of keeping its center of mass over the centroid of its support polygon, while moving the front-left foot in a fast Figure-8 trajectory. The robot has a 6 axis IMU and encoders at its 12 actuated joints as well as Vicon motion capture system for body position / orientation. The body weighs 1.8kg, and each leg weighs .25kg; The body is 20cm in length, while the upper arm is 8cm, and the lower arm is 10cm. ©2007 IEEE [Shkolnik and Tedrake, 2007].

3.1 Motivation for COM Control

The notion of true stability for locomotion is difficult to quantify precisely (see [Tedrake et al., 2006, Byl and Tedrake, 2009b]), so heuristics are often utilized for this purpose. These include: maximizing the static stability margin [McGhee and Frank, 1968], maximizing the Zero-Moment Point (ZMP) margin [Hirai et al., 1998], Resolved Momentum Control (RMC) [Kajita et al., 2003a], and Zero Spin Center of Pressure (ZSCP) Control [Popovic et al., 2004], etc (see [Pratt and Tedrake, 2005] for review). Essentially all of these approaches try to control the center of mass of the entire robot, so we pay particular attention to this issue in this work. This is true even for ZMP, where a COM trajectory can be computed to follow a desired ZMP trajectory [Kajita et al., 2003b].

The RMC framework for humanoid robots attempts to control a humanoid robot's whole-body linear and angular momentum (or components of these), using a framework similar to the whole-body Jacobian, while constraining feet movement to specified trajectories [Kajita et al., 2003a]. Linear momentum divided by mass of the system translates to COM velocity, so the high level objective of RMC is similar to this work. However, RMC assumes actuated ankles, and is primarily concerned with how to utilize free DOFs of a humanoid, for example

the arms and torso, to help keep the robot stable. The quadruped robot does not have such flexibility. Further, the work developed here allows for hierarchical control with priority given to the COM velocity rather than to the feet velocities, whereas RMC does not allow for this.

The remainder of this chapter is organized as follows: 1) derivations of the partial inverse kinematics control of the COB; 2) derivations of the whole-body Jacobians associated with the tasks COM control and swing foot control; 3) three redundancy resolution schemes are compared in simulation, and results are presented from experiments on the actual robot.

3.2 Single-Leg Inverse Kinematics Control

A first approach to developing a walking controller for the position controlled walking robot might be to treat all of the legs separately, and control the center of body and the orientation of the body by moving the legs appropriately. When considering each leg separately, one can utilize the relation:

$$\mathbf{P}_G = \mathbf{X}_{B_n} + \mathbf{R}_B \cdot \mathbf{P}_L \tag{3.1}$$

where $\mathbf{P}_G \in \mathbb{R}^{3\times n}$ contains the feet positions in the global frame; $\mathbf{P}_L \in \mathbb{R}^{3\times n}$ contains the feet positions in the robot relative frame; $\mathbf{X}_B \in \mathbb{R}^3$ is the center of body position and $\mathbf{X}_{B_n} \in \mathbb{R}^{3\times n}$ is the $\mathbf{X}_B$ vector repeated n times: $[\mathbf{X}_B \ ... \ \mathbf{X}_B]$; $\mathbf{R}_B \in \mathbb{R}^{3\times 3}$ is the rotation matrix of the body; n is the number of feet being considered (usually 3 or 4). Differentiating and solving for the feet velocities in the relative frame:

$$\dot{\mathbf{P}}_L = \mathbf{R}_B^T \cdot (\dot{\mathbf{P}}_G - \dot{\mathbf{X}}_{B_n} - \dot{\mathbf{R}}_B \cdot \mathbf{P}_L) \tag{3.2}$$

Note that $\dot{\mathbf{P}}_G$ is assumed to be zero for stance feet, and is otherwise the velocity command of the swing foot. $\dot{\mathbf{X}}_B$ and $\dot{\mathbf{R}}_B$ are the commanded COB and orientation velocities.

For each leg, i, one can compute the single-leg Jacobians, $\mathbf{J}_{leg_i} \in \mathbb{R}^{3\times 3}$, of the foot position w.r.t. the robot frame. The joint velocities corresponding to each leg are then:

$$\dot{\mathbf{q}}_{leg_i} = \mathbf{J}_{leg_i}^{-1} \cdot \dot{\mathbf{P}}_{L_i} \tag{3.3}$$

The result is a control for COB, not COM. Further, it may be useful to allow the body rotation to be left unspecified. This increases the workspace of the system by allowing the robot to rotate the body to help reach places that would otherwise be kinematically infeasible. To deal with these issues, whole-body Resolved Motion Rate Control [Whitney and D.E., 1969] is utilized.

3.3 Whole-Body Jacobian Control

Recall that forward kinematics, transforming joint angles $\mathbf{q} \in \mathbb{R}^n$ into some task space $\mathbf{x} \in \mathbb{R}^m$, and the differentiation of this relation is given by:

$$\begin{aligned} \mathbf{x} &= f(\mathbf{q}) && (3.4) \\ \dot{\mathbf{x}} &= \mathbf{J}(\mathbf{q}) \cdot \dot{\mathbf{q}} && (3.5) \end{aligned}$$

where $J(q) \in \mathbb{R}^{m \times n}$ is the whole-body Jacobian associated with the task space of $\mathbf{x}$. The inverse kinematics with nullspace optimization for redundancy resolution can be solved as in [Liegeois, 1977]:

$$\dot{\mathbf{q}} = \mathbf{J}^+ \cdot \dot{\mathbf{x}} + \alpha(I - \mathbf{J}^+ \cdot \mathbf{J}) \cdot \dot{\mathbf{q}}_{ref} \quad (3.6)$$

where $\mathbf{J}^+ = (\mathbf{J}^T\mathbf{J})^{-1} \cdot \mathbf{J}^T$ is the Moore-Penrose pseudoinverse of $\mathbf{J}$, α is a scalar weighting, and $\dot{\mathbf{q}}_{ref} \in \mathbb{R}^n$ is a low priority command in joint space. For the redundant case where $m < n$, the solution, $\dot{\mathbf{q}}$ should achieve the commanded $\dot{\mathbf{x}}$ while also minimizing $\|\dot{\mathbf{q}} - \dot{\mathbf{q}}_{ref}\|$.

One may have multiple tasks, and control them in a hierarchical manner [Khatib, 1987], for example:

$$\begin{aligned} \dot{\mathbf{q}}_1 &= \mathbf{J}^+_{low} \cdot \dot{\mathbf{x}}_{low} + \alpha_1(I - \mathbf{J}^+_{low} \cdot \mathbf{J}_{low}) \cdot \dot{\mathbf{q}}_{ref} \\ \dot{\mathbf{q}} &= \mathbf{J}^+_{high} \cdot \dot{\mathbf{x}}_{high} + \alpha_2(I - \mathbf{J}^+_{high} \cdot \mathbf{J}_{high}) \cdot \dot{\mathbf{q}}_1 && (3.7) \end{aligned}$$

where $\mathbf{x}_{high}$ represents a "high priority task" with associated $\mathbf{J}_{high}$ Jacobian, and conversely $\mathbf{x}_{low}$ is a "lower priority task" with associated $\mathbf{J}_{low}$ Jacobian. $\dot{\mathbf{q}}_1$ is the joint level command that would be assigned by the low priority task, and is passed on as a "suggested" command to the high priority task.

Typically, the lowest priority task in joint space consists of specifying $\dot{\mathbf{q}}_{ref}$ to move down a potential to bring the posture to some standard "favored" position, as in [Khatib et al., 2004]. Note that it is possible to differentiate (3.5), which would allow for control at the acceleration or force level (for review see [Nakanishi et al., 2005]). Because the LittleDog robot is position and velocity controlled, I will limit the discussion here to controllers that resolve kinematic redundancies at the velocity level.

3.3.1 Jacobian of Center of Body

This section presents a derivation of the Jacobian, $\mathbf{J}_{X_B}(q) = \frac{\partial \mathbf{X}_B}{\partial \mathbf{q}_a}$, of the center of body, $\mathbf{X}_B \in \mathbb{R}^3$, where $\mathbf{q} \in \mathbb{R}^{18}$ describes all 18 degrees of freedom, and $\mathbf{q}_a \in \mathbb{R}^{12}$ is the actuated subset of q, not including the 6 unactuated degrees of freedom of the body position and orientation.

The typical approach for computing the Jacobian would be to define forward kinematics and then differentiate w.r.t. actuated joint angles. It is not obvious how to do this for the center of body of the robot, and some assumptions must be made. First, note that given $\mathbf{q}$, one may compute the position of all feet in both relative and absolute coordinate systems.

Given absolute positions of three feet, and three leg lengths, one may compute the position of the center of body by performing trilateration, e.g. by finding the intersection of three spheres, centered at the feet positions with radii corresponding the distance between the foot and COB. However, this derivation is bulky and, is not well defined for cases with four feet.

In order to get around the problem of underactuation, I make an assumption that feet on the ground do not move, that is to say that there is infinite friction, and the ZMP remains inside the support polygon. Consider what happens when changing the leg lengths; first note that the lengths, $L \in \mathbb{R}^4$, are simply the distance from center of body to the feet. This is defined in either the global coordinate frame, or the robot coordinate frame for each foot, i:

$$\begin{aligned} L_i &= \|\mathbf{P}_{L_i}\| \\ &= \|\mathbf{X}_B - \mathbf{P}_{G_i}\| \end{aligned} \tag{3.8}$$

Let us assume that all feet are on the ground (otherwise use the subset of three feet which are on the ground), and the feet are not moving: $\frac{\partial \mathbf{P}_{G_i}}{\partial \mathbf{q}_a} = 0$. One can obtain $\frac{\partial L}{\partial \mathbf{q}_a} \in \mathbb{R}^{4\times 12}$ by differentiating for each leg length individually, corresponding to each row i:

$$\begin{aligned} \frac{\partial L_i}{\partial \mathbf{q}_a} &= \frac{\partial L_i}{\partial \mathbf{P}_{L_i}} \frac{\partial \mathbf{P}_{L_i}}{\partial \mathbf{q}_a} \\ &= \frac{\partial L_i}{\partial \mathbf{X}_B} \frac{\partial \mathbf{X}_B}{\partial \mathbf{q}_a} \end{aligned} \tag{3.9}$$

These derivatives are well defined, and have geometrically understandable meanings. For example, $\frac{\partial L}{\partial \mathbf{X}_B} \in \mathbb{R}^{4\times 3}$ is the change of leg lengths given a movement of the body (imagine a table with springs for legs; the springs change length in a well defined way if the table is moved or rotated). One can now solve for $\frac{\partial \mathbf{X}_B}{\partial \mathbf{q}_a}$ using the Moore-Penrose pseudoinverse:

$$\frac{\partial \mathbf{X}_B}{\partial \mathbf{q}_a} = \left[\frac{\partial L}{\partial \mathbf{X}_B}\right]^+ \frac{\partial L}{\partial \mathbf{P}_L} \frac{\partial \mathbf{P}_L}{\partial \mathbf{q}_a} \tag{3.10}$$

where each row i of $\frac{\partial L}{\partial \mathbf{X}_B} \in \mathbb{R}^{4\times 3}$ is:

$$\frac{\partial L_i}{\partial \mathbf{X}_B} = \frac{\partial ||\mathbf{P}_{G_i} - \mathbf{X}_B||}{\partial \mathbf{X}_B} = \frac{[\mathbf{X}_B - \mathbf{P}_{G_i}]^T}{||\mathbf{X}_B - \mathbf{P}_{G_i}||}$$

and, if one treats $\mathbf{P}_L$ as a vector $\in \mathbb{R}^{12}$, and define $\mathbf{P}_{L_i}$ as a vector $\in \mathbb{R}^{12}$ containing all zeros except the 3 elements corresponding to leg i, then each row i of $\frac{\partial L}{\partial \mathbf{P}_L} \in \mathbb{R}^{4\times 12}$ is:

$$\frac{\partial L_i}{\partial \mathbf{P}_L} = \frac{\partial ||\mathbf{P}_{L_i}||}{\partial \mathbf{P}_L} = \frac{[\mathbf{P}_{L_i}]^T}{||\mathbf{P}_{L_i}||}$$

$$\frac{\partial \mathbf{P}_L}{\partial \mathbf{q}_a} = \begin{bmatrix} \mathbf{J}_{leg_1} & 0 & 0 & 0 \\ 0 & \mathbf{J}_{leg_2} & 0 & 0 \\ 0 & 0 & \mathbf{J}_{leg_3} & 0 \\ 0 & 0 & 0 & \mathbf{J}_{leg_4} \end{bmatrix} \in \mathbb{R}^{12\times 12}$$

3.3.2 Jacobian of Body Rotation

By differentiating (3.1) w.r.t. $\mathbf{q}_{a_j}$, for each joint, j, and assuming that stance feet are not slipping so that $\frac{\partial \mathbf{P}_{G_i}}{\partial \mathbf{q}_a} = 0$ for each stance foot i, one finds that:

$$\frac{\partial \mathbf{R}_B}{\partial \mathbf{q}_{a_j}} \cdot \mathbf{P}_L = -\frac{\partial \mathbf{X}_{B_n}}{\partial \mathbf{q}_{a_j}} - \mathbf{R}_B \cdot \frac{\partial \mathbf{P}_L}{\partial \mathbf{q}_{a_j}} \tag{3.11}$$

The Moore-Penrose pseudoinverse is applied to solve for $\mathbf{J}_{R_{B_j}} = \frac{\partial \mathbf{R}_B}{\partial \mathbf{q}_{a_j}} \in \mathbb{R}^{3\times 3}$ for each joint j:

$$\mathbf{J}_{R_{B_j}} = -\left(\frac{\partial \mathbf{X}_{B_n}}{\partial \mathbf{q}_{a_j}} + \mathbf{R}_B \cdot \frac{\partial \mathbf{P}_L}{\partial \mathbf{q}_{a_j}}\right) \cdot (\mathbf{P}_L)^+ \tag{3.12}$$

3.3.3 Center of Mass Jacobian

The conversion from local frame (L) to global frame (G) for the center of mass, $\mathbf{X}_M \in \mathbb{R}^3$ is specified by:

$$\mathbf{X}_{M_G} = \mathbf{X}_B + \mathbf{R}_B \cdot \mathbf{X}_{M_L} \tag{3.13}$$

Then solve for $\mathbf{J}_{X_{MG}} = \frac{\partial \mathbf{X}_{M_G}}{\partial \mathbf{q}_a}$, where each column j is:

$$\mathbf{J}_{X_{MG_j}} = \frac{\partial \mathbf{X}_B}{\partial \mathbf{q}_{a_j}} + \mathbf{R}_B \cdot \frac{\partial \mathbf{X}_{M_L}}{\partial \mathbf{q}_{a_j}} + \frac{\partial \mathbf{R}_B}{\partial \mathbf{q}_{a_j}} \cdot \mathbf{X}_{M_L} \tag{3.14}$$

3.3.4 Swing Foot Jacobian

The conversion from relative to global coordinates for the position of a specific foot, $\mathbf{X}_{SW} \in \mathbb{R}^3$, is specified by:

$$\mathbf{X}_{SW_G} = \mathbf{X}_B + \mathbf{R}_B \cdot \mathbf{X}_{SW_L} \tag{3.15}$$

Then solve for $\mathbf{J}_{X_{SWG}} = \frac{\partial \mathbf{X}_{SW_G}}{\partial \mathbf{q}_a}$, where each column j is:

$$\mathbf{J}_{X_{SW_{Gj}}} = \frac{\partial \mathbf{X}_B}{\partial \mathbf{q}_{a_j}} + \mathbf{R}_B \cdot \frac{\partial \mathbf{X}_{SW_L}}{\partial \mathbf{q}_{a_j}} + \frac{\partial \mathbf{R}_B}{\partial \mathbf{q}_{a_j}} \cdot \mathbf{X}_{SW_L} \tag{3.16}$$

3.3.5 Hierarchical Controller

As shown in (3.7), the Jacobians and associated null spaces can be "stacked" in a hierarchical manner. Given the goal of stable walking over rough terrain, it is logical to give the control of $\dot{\mathbf{X}}_{M_G}$ the highest priority. Of second priority, then, is $\dot{\mathbf{X}}_{SW_G}$. The final step in developing the full control law is choosing an appropriate $\dot{\mathbf{q}}_{ref}$ to plug into (3.7). Workspace centering is established by choosing $\dot{\mathbf{q}}_{ref}$ along the gradient of the potential: $\|\mathbf{q}_a - \mathbf{q}_0\|$ for some favored position, $\mathbf{q}_0 \in \mathbb{R}^{12}$.

A proposed alternative to workspace centering is to compute the partial IK solution, as in section 3.2; for each leg, i, the corresponding elements of $\dot{\mathbf{q}}_{ref}$ are:

$$\dot{\mathbf{q}}_{ref_i} = \mathbf{J}_{leg_i}^{-1} \dot{\mathbf{P}}_{L_i} \tag{3.17}$$

Using the single-leg inverse kinematics represents a dynamic redundancy resolution method, as opposed to workspace centering which always tries to move the robot back to the static $\mathbf{q}_0$ position. The final control is specified by:

$$\begin{aligned} \dot{\mathbf{q}}_1 &= \mathbf{J}_{SW_G}^{+} \cdot \dot{\mathbf{x}}_{SW_G} + (I - \mathbf{J}_{SW_G}^{+} \cdot \mathbf{J}_{SW_G}) \cdot \dot{\mathbf{q}}_{ref} \\ \dot{\mathbf{q}} &= \mathbf{J}_{M_G}^{+} \cdot \dot{\mathbf{x}}_{M_G} + (I - \mathbf{J}_{M_G}^{+} \cdot \mathbf{J}_{M_G}) \cdot \dot{\mathbf{q}}_1 \end{aligned} \tag{3.18}$$

Kinematic joint limits can also be included as a task. This control is only activated when joints are close to their limits, and is thus not included here for clarity.

3.4 Results

3.4.1 Simulation

The result of the preceding analysis distinguishes three potential controllers: 1) control by single-leg inverse-kinematics, where body orientation must also be commanded (in the case of the simulations below, I use zero pitch and roll, and keep yaw constant); 2) Whole-body Jacobian RMRC based control using static redundancy resolution which chooses trajectories that stay as close as possible to a standing pose (static redundancy resolution); and 3) A hybrid approach, based on whole-body Jacobian control which chooses trajectories that are as close as possible to the single-leg IK solutions (dynamic redundancy resolution).

In this section I explore the performance of these three controllers by looking at a simulation of a quadruped robot with parameters similar to the LittleDog robot. The physics based dynamics simulation was constructed with SD/FAST, and the controller was implemented in Matlab. A spring-damper ground model is utilized with point-feet contacts, with reasonable ground reaction forces (drawn in small blue lines in the figures). The simulation has 4 test-cases to illustrate the performance in sample COM and swing foot trajectory tasks.

- **CASE 1**: Figure 8 trajectory for COM, with 4 feet on ground. In this case, a figure 8 desired trajectory is tracked with the whole-robot center of mass (see Figure 3-4). Four feet remain on the ground.

- **CASE 2**: Figure 8 trajectory for COM, with 3 feet on ground. In this case, a figure 8 desired trajectory is tracked with the whole robot center of mass (see Figure 3-5). Three feet remain on the ground, and one foot is raised in the air.

  Figure 3-2 shows the ZMP and center of mass, comparing the work-centering and hybrid approaches. The whole-body Jacobian with work-centering utilizes more drastic motions, with the involvement of all limbs, which results in the ZMP straying far from the COM projection. This undermines the static stability assumption that was made when selecting the figure 8 COM trajectory to follow, and results in the robot eventually toppling over. On the other hand, the ZMP corresponds fairly well with the COM projection in the hybrid controller, as this controller minimizes drastic movements. Thus, even though the robot is moving fairly quickly (executing the figure 8 motion in 3 seconds), the static stability margin is reasonable to use with the controller using the dynamic redundancy resolution. This illustrates why the hybrid controller is more stable when executing fast movements.

- **CASE 3**: Small, fast, figure 8 trajectory for swing leg. In this case, a small figure 8 is to be tracked by the front right leg, while the COM is to remain over the centroid of the support polygon (see Figure 3-6). In this test, the entire figure 8 trajectory is completed quickly (in 1 sec.).

- **CASE 4**: Very large, slower, figure 8 trajectory for swing leg. In this case, a very large figure 8, requiring the robot to pitch its body up and down to complete it accurately, is to be tracked by the front right leg, while the center of mass is to remain over the centroid of the support polygon (see Figure 3-7). In this test, the figure 8 trajectory is completed fairly slowly, over a period of 15 seconds.

3.4.2 Real Robot

The tests were attempted on the actual robot, with results for the first three cases of the hybrid controller (with feedback) shown in Figure 3-3. A photo of the robot executing the small figure 8 task is in Figure 3-1. The gains on this feedback controller were difficult to tune, with a trade off between oscillations and significant foot slippage vs tracking performance. Performance was deteriorated compared to simulation because feet were slipping, which violated our Jacobian assumptions that grounded feet were not moving. This was enhanced by oscillations due to feedback latencies. Test case 4, large figure-8 with swing foot, could not be achieved at all due to feet slipping.

In another approach, joint trajectories were generated by using the simulator with the hybrid controller for the test cases presented above. These trajectories were recorded, and passed to the actual LittleDog robot as a feed-forward command. The performance was quite

good, and resembled the performance seen in the simulators. If feet did not slip, tracking looked much better than the feedback control shown in Figure 3-3.

3.5 Discussion

In the prior chapter, I illustrated that fast path planning can be achieved by biasing a search in task space. This is achieved by sampling in task space, and is complemented, if possible, by an inner control loop which regulates the system in task space. In this chapter, I explored task space control for a quasi-actuated quadruped robot. The controller developed can be applied within a search based planning algorithm, for example by defining a task space consisting of the horizontal position of the robot COM, the robot heading (yaw), and the position of the swing foot. This leaves a 5 DOF task space, significantly reducing the robot's 18 dimensional configuration space.

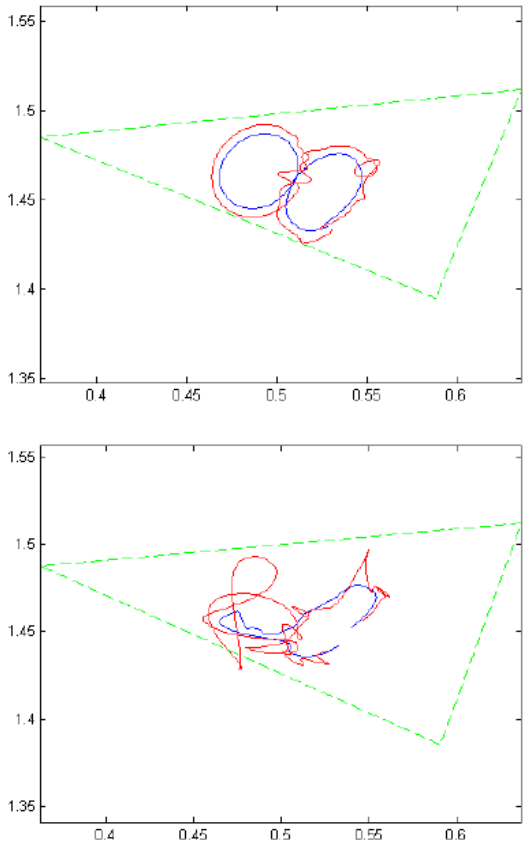


Figure 3-2: ZMP (red) vs COM (blue) for Case 2 with work-centering (bottom) and hybrid control (top). The triangle depicts the support polygon. This figure illustrates that ZMP more closely follows the COM trajectory in the hybrid control, which is the reason why the hybrid control is less likely to fall when executing fast motions under static stability assumptions. ©2007 IEEE [Shkolnik and Tedrake, 2007].

To achieve task control for LittleDog, I derived whole-body Jacobians for center of mass and swing foot trajectory control of a point-foot quadruped robot, despite the fact that the forward kinematics for center of body are ill-defined. Three control methods were explored, including 1) a partial IK solution; and whole-body solutions using either 2) work-space centering and 3) a hybrid controller which utilizes the partial IK solution for dynamic redundancy resolution. The controllers were tested on several test cases in simulation, designed to force the robot to move dynamically or to the edge of kinematic feasibility. The hybrid controller maximized the utility of the static stability criteria, while also following the commanded trajectories with the least error on all four tests. The results demonstrate that task space control is sensitive to how redundancies are resolved. With careful consideration of the problem at hand, it is possible to achieve improved performance over standard approaches such as work space centering, which in a planning framework such as discussed in Chapter 2, would result in improved completeness and faster search times because fewer expansion attempts would result in failure.

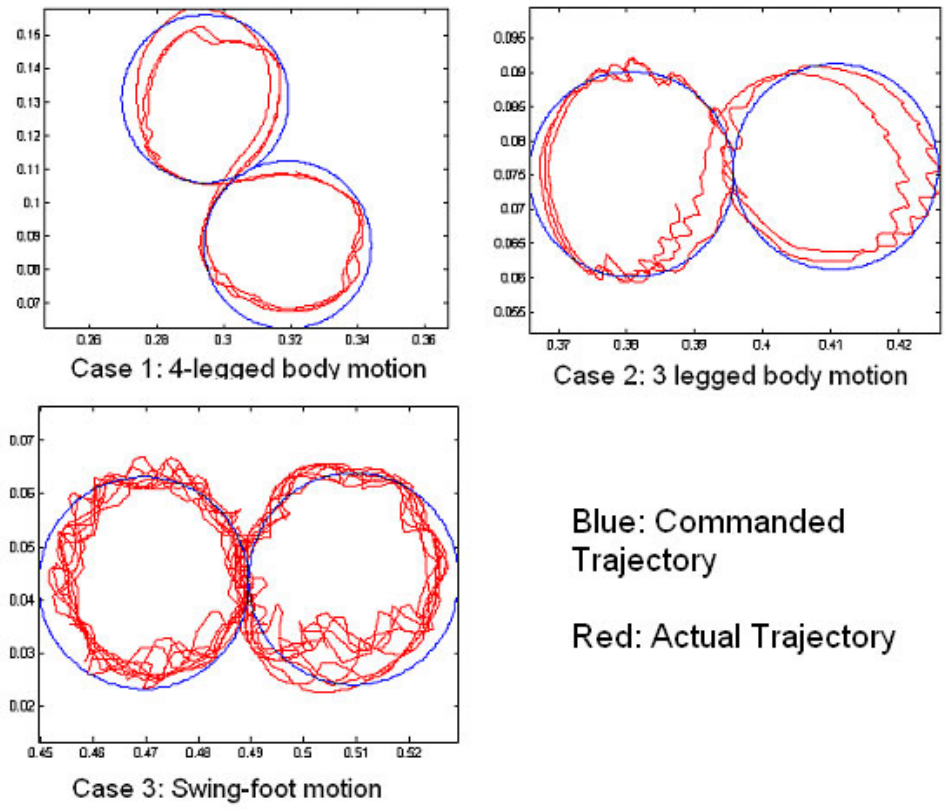


Figure 3-3: **Experiment Results** Cases 1-3 shown (top to bottom). ©2007 IEEE [Shkolnik and Tedrake, 2007].

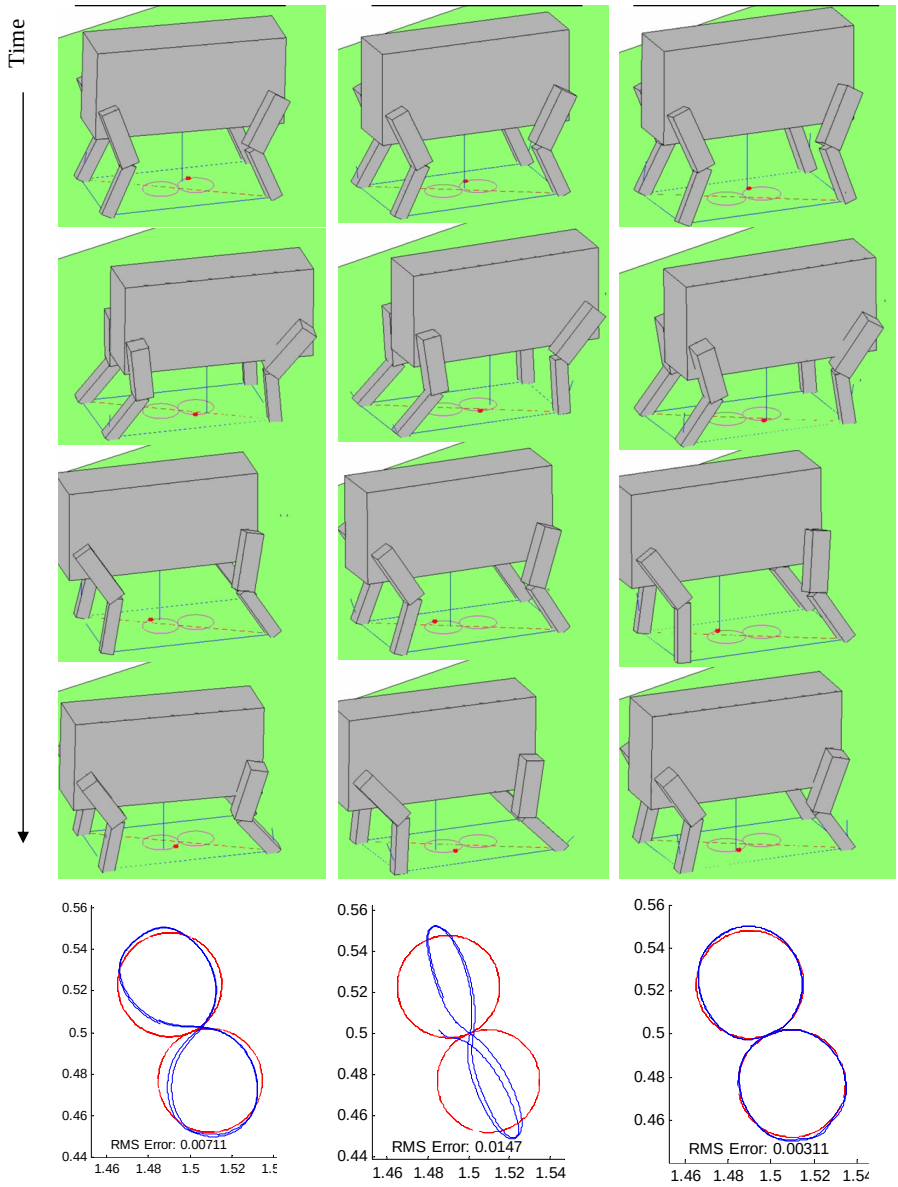


Figure 3-4: **Simulation Results: Case 1.** COM figure-8 Trajectory with four legs. **Left**: Partial IK. Performance is reasonably good. **Center**: Whole-body Jacobian with Work centering. The centering appears to interfere with this task. Note, it is possible to reduce gains on the centering, but this results in worse performance in other tests. **Right**: Hybrid approach. The trajectory following appears very good, and results in the lowest RMS Error. **Bottom**: red line is the commanded trajectory (2 seconds), blue line is the actual trajectory. ©2007 IEEE [Shkolnik and Tedrake, 2007].

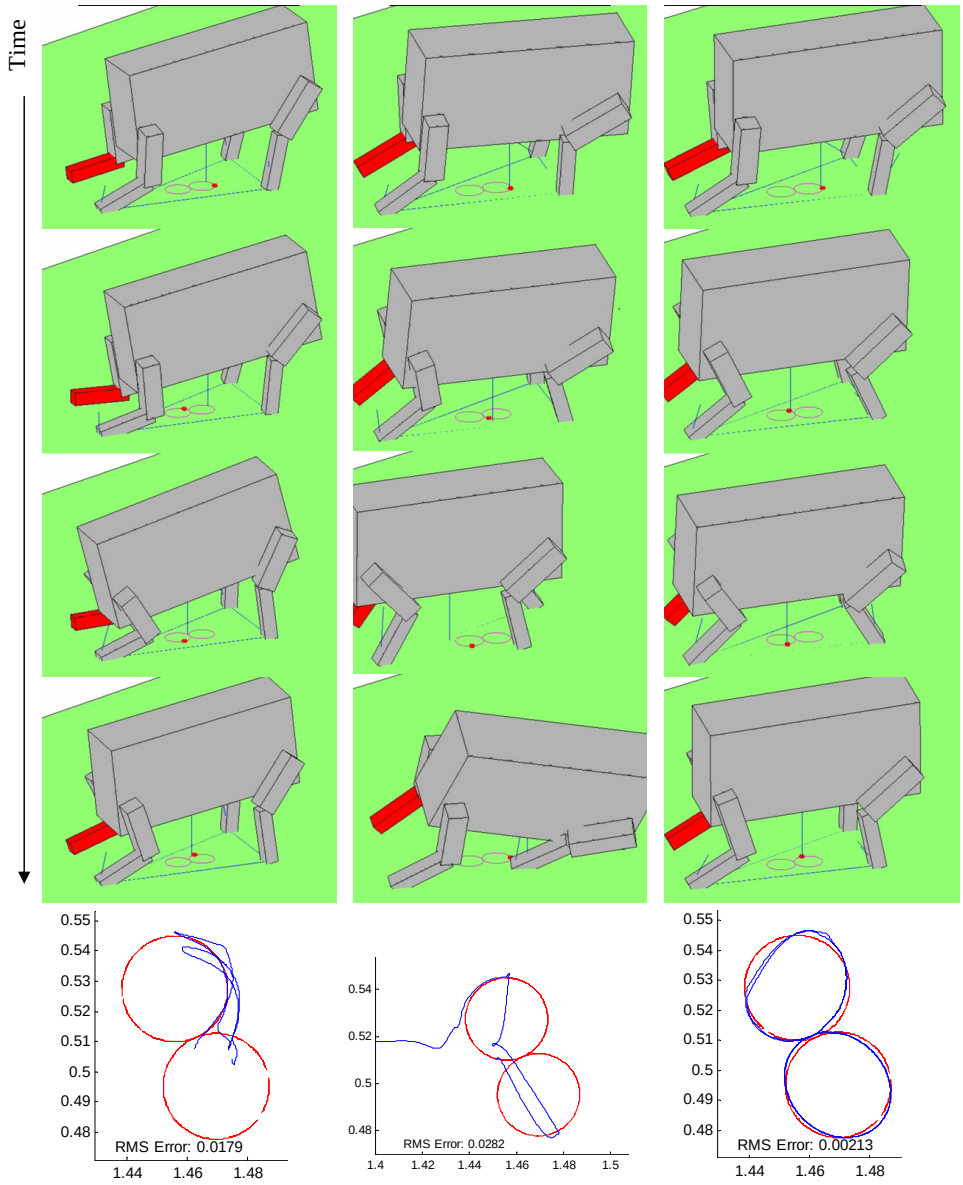


Figure 3-5: **Simulation Results: Case 2.** The red leg designates the swing foot in the air. **Left**: Partial IK. The robot is unable to follow the specified trajectory at the speed given.**Center**: Whole-body Jacobian with Work centering. The robot can not finish the task as he rotates forward and eventually flips over. **Right**: Hybrid approach. This is the only approach that is able to follow the trajectory even somewhat closely. **Bottom**: red line is the commanded trajectory (3 seconds), blue line is the actual trajectory. ©2007 IEEE [Shkolnik and Tedrake, 2007].

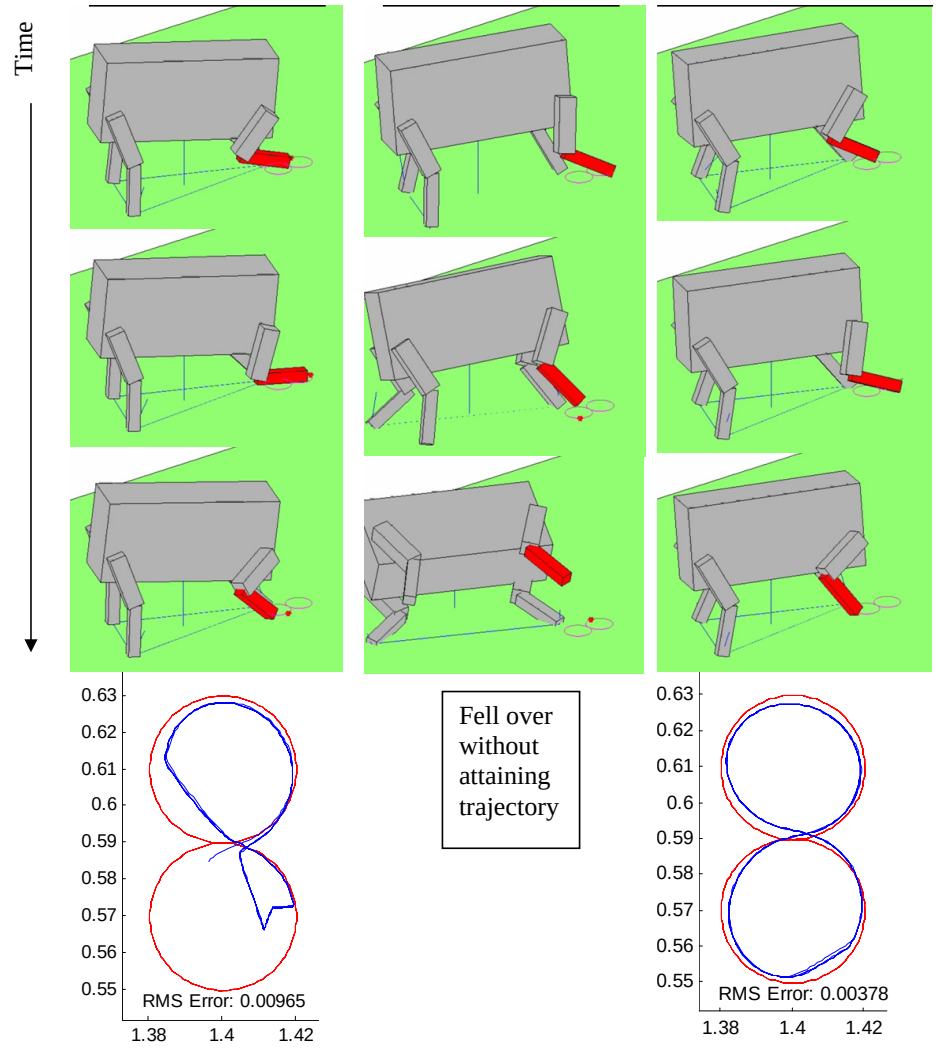


Figure 3-6: **Simulation Results: Case 3.** The red leg designates the swing foot, which is executing a figure-8 trajectory. **Left**: Partial IK. Performance is poor particularly because joint limits are reached. **Center**: Whole-body Jacobian with Work centering. Rapid twisting (jerking) motions cause the robot to loose footing and topple over; the trajectory at this speed can not be achieved with this controller. **Right**: Hybrid approach. The robot seems to minimize twisting and turning motions, and achieves reasonable trajectory following performance. **Bottom**: red line is the commanded trajectory (1 second), blue line is the actual trajectory. ©2007 IEEE [Shkolnik and Tedrake, 2007].

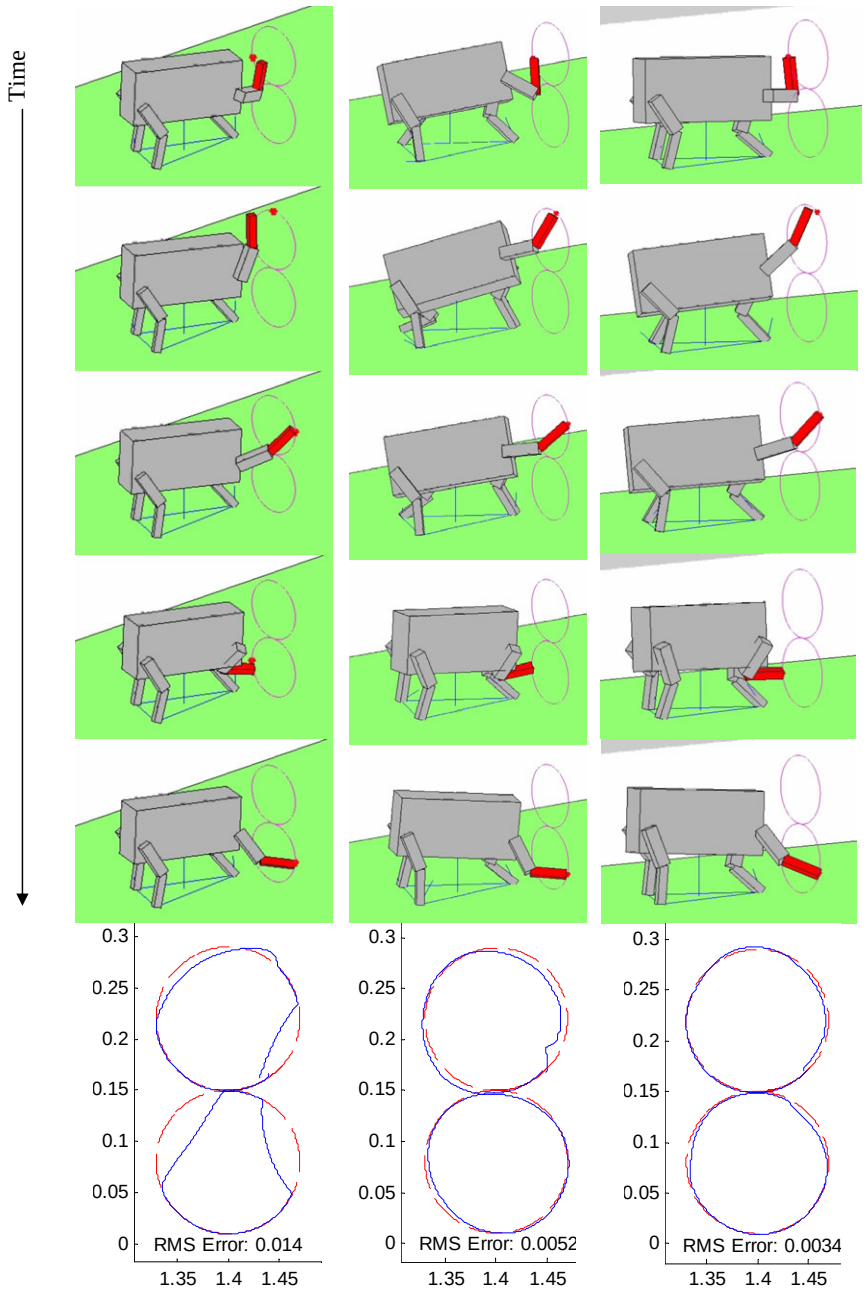


Figure 3-7: **Simulation Results: Case 4.** The red leg designates the swing foot, which is executing a figure-8 trajectory. **Left**: Partial IK. Performance is poor particularly because joint limits are reached. **Center**: Whole-body Jacobian with Work centering. Tracking of the figure 8 is good, but COM tracking is affected here, and the robot goes though twisting motions. **Right**: Hybrid approach. The robot seems to minimize twisting and turning motions, and trajectory following has lowest RMS error. **Bottom**: red line is the commanded trajectory (15 seconds), blue line is the actual trajectory. ©2007 IEEE [Shkolnik and Tedrake, 2007].

Chapter 4

Task Space Control and Planning for Underactuated Systems

The previous chapters discussed task space control in the context of reducing dimensionality within a motion planning framework. As we saw in Chapter 2, task space control is well defined for fully actuated systems. In the last chapter, we saw how a legged robot, which is technically an underactuated system, can be treated as if it is fully actuated - but only when executing conservative motion trajectories which are constrained so that the robot always maintains a large support polygon on the ground. To achieve animal-like agility, however, motion planning should not be restricted by such limiting constraints. Consider, for example, trying to plan a dynamic climbing maneuver with LittleDog, in which two feet are lifted from the ground at the same time (As shown in Figure 4-7). In this case, the robot does not have a support polygon - since the feet form a line contact on the ground. The interface between the robot and the ground is essentially a passive degree of freedom.

In this chapter, I derive task space control for systems with passive degrees of freedom by using partial feedback linearization. This allows one to plan motions for underactuated robots, which may have many degrees of freedom, directly in a low-dimensional task space. The derivation presented contains a pseudo-Jacobian, which preserves the ability to implement hierarchical control so that redundancy can be resolved with lower-priority tasks.

The potential of this approach is demonstrated on the classical problem of swing-up of a multi-link arm with passive joints. By choosing the COM of the multi-link arm as a task space, a simple search based motion planner is able to find feasible swing up trajectories by probing actions in the low-dimensional task space. A general idea of a motion planning framework for high-dimensional underactuated systems is shown in Figure 4-1. It should be evident that the TS-RRT described in Chapter 2 fits in the context of this framework.

Note that significant portions of this chapter also appear in [Shkolnik and Tedrake, 2008].

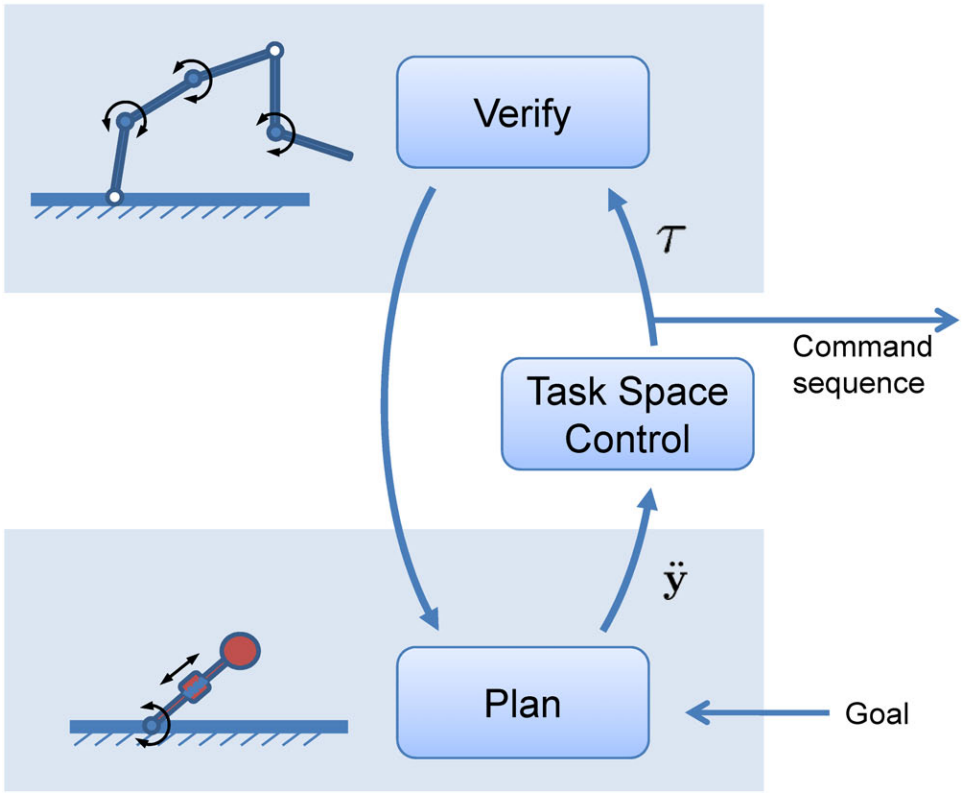


Figure 4-1: Schematic of an underactuated manipulator (top left) and the lower-dimensional fully-actuated model (bottom left) that defines the task space of the problem. Plans generated in task space are mapped back to the active joints of the underactuated system, and are verified to satisfy any constraints in the underactuated system. (White circles represent passive joints.) ©2008 IEEE [Shkolnik and Tedrake, 2008].

4.1 Task Space Control of Underactuated Systems

Without loss of generality, the equations of motion of an underactuated system can be broken down in two components, one corresponding to unactuated joints, $\mathbf{q}_1$, and one corresponding to actuated joints, $\mathbf{q}_2$:

$$\mathbf{M}_{11}\ddot{\mathbf{q}}_1 + \mathbf{M}_{12}\ddot{\mathbf{q}}_2 + \mathbf{h}_1 + \boldsymbol{\phi}_1 = 0 \tag{4.1}$$

$$\mathbf{M}_{21}\ddot{\mathbf{q}}_1 + \mathbf{M}_{22}\ddot{\mathbf{q}}_2 + \mathbf{h}_2 + \boldsymbol{\phi}_2 = \boldsymbol{\tau} \tag{4.2}$$

with $\mathbf{q} \in \Re^n$, $\mathbf{q}_1 \in \Re^m$, $\mathbf{q}_2 \in \Re^l$, $l = n - m$. Consider an output function of the form,

$$\mathbf{y} = \mathbf{f}(\mathbf{q}),$$

with $\mathbf{y} \in \Re^p$, which defines the task space. Define $\mathbf{J}_1 = \frac{\partial \mathbf{f}}{\partial \mathbf{q}_1}$, $\mathbf{J}_2 = \frac{\partial \mathbf{f}}{\partial \mathbf{q}_2}$, $\mathbf{J} = [\mathbf{J}_1, \mathbf{J}_2]$.

Theorem 1 (Task Space PFL)**.** *If the actuated joints are commanded so that*

$$\ddot{\mathbf{q}}_2 = \bar{\mathbf{J}}^+ \left[\mathbf{v} - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}_1 \mathbf{M}_{11}^{-1}(\mathbf{h}_1 + \boldsymbol{\phi}_1) \right], \tag{4.3}$$

where $\bar{\mathbf{J}} = \mathbf{J}_2 - \mathbf{J}_1 \mathbf{M}_{11}^{-1} \mathbf{M}_{12}$*. and* $\bar{\mathbf{J}}^+$ *is the right Moore-Penrose pseudo-inverse,*

$$\bar{\mathbf{J}}^+ = \bar{\mathbf{J}}^T (\bar{\mathbf{J}} \bar{\mathbf{J}}^T)^{-1},$$

then

$$\ddot{\mathbf{y}} = \mathbf{v}. \tag{4.4}$$

subject to

$$rank\left(\bar{\mathbf{J}}\right) = p, \tag{4.5}$$

Proof. Differentiating the output function:

$$\dot{\mathbf{y}} = \mathbf{J}\dot{\mathbf{q}}$$
$$\ddot{\mathbf{y}} = \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}_1 \ddot{\mathbf{q}}_1 + \mathbf{J}_2 \ddot{\mathbf{q}}_2.$$

Solving 4.1 for the dynamics of the unactuated joints:

$$\ddot{\mathbf{q}}_1 = -\mathbf{M}_{11}^{-1}(\mathbf{M}_{12}\ddot{\mathbf{q}}_2 + \mathbf{h}_1 + \boldsymbol{\phi}_1) \tag{4.6}$$

Substituting, we have

$$\begin{aligned} \ddot{\mathbf{y}} =& \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}_1 \mathbf{M}_{11}^{-1}(\mathbf{M}_{12}\ddot{\mathbf{q}}_2 + \mathbf{h}_1 + \boldsymbol{\phi}_1) + \mathbf{J}_2 \ddot{\mathbf{q}}_2 && (4.7)\\ =& \dot{\mathbf{J}}\dot{\mathbf{q}} + \bar{\mathbf{J}}\ddot{\mathbf{q}}_2 - \mathbf{J}_1 \mathbf{M}_{11}^{-1}(\mathbf{h}_1 + \boldsymbol{\phi}_1) && (4.8)\\ =& \mathbf{v} && (4.9) \end{aligned}$$

Note that the last line required the rank condition (4.5) on $\bar{\mathbf{J}}$ to ensure that the rows of $\bar{\mathbf{J}}$ are linearly independent, allowing $\bar{\mathbf{J}}\bar{\mathbf{J}}^+ = \mathbf{I}$. □

In order to execute a task space trajectory one could command

$$\mathbf{v} = \ddot{\mathbf{y}}^d + \mathbf{K}_d(\dot{\mathbf{y}}^d - \dot{\mathbf{y}}) + \mathbf{K}_p(\mathbf{y}^d - \mathbf{y}).$$

Assuming the internal dynamics are stable, this yields converging error dynamics, $(\mathbf{y}_d - \mathbf{y})$, when $\mathbf{K}_p, \mathbf{K}_d > 0$ [Slotine and Li, 1990]. For a position control robot, the acceleration command of (4.3) suffices. Alternatively, a torque command follows by substituting (4.3) and (4.6) into (4.2).

4.1.1 Collocated and Non-Collocated PFL

The task space derivation above provides a convenient generalization of the partial feedback linearization (PFL) [Spong, 1994a], which encompasses both the collocated and non-collocated results. If one chooses $\mathbf{y} = \mathbf{q}_2$ (collocated), then

$$\mathbf{J}_1 = 0, \mathbf{J}_2 = 1, \dot{\mathbf{J}} = 0, \bar{\mathbf{J}} = 1, \bar{\mathbf{J}}^+ = 1.$$

From this, the command in (4.3) reduces to $\ddot{\mathbf{q}}_2 = v$. The torque command is then

$$\tau = -\mathbf{M}_{21}\mathbf{M}_{11}^{-1}(\mathbf{M}_{12}v + \mathbf{h}_1 + \boldsymbol{\phi}_1) + \mathbf{M}_{22}v + \mathbf{h}_2 + \boldsymbol{\phi}_2,$$

and the rank condition (4.5) is always met.

If one chooses $\mathbf{y} = \mathbf{q}_1$ (non-collocated), then

$$\mathbf{J}_1 = 1, \mathbf{J}_2 = 0, \dot{\mathbf{J}} = 0, \bar{\mathbf{J}} = -\mathbf{M}_{11}^{-1}\mathbf{M}_{12}.$$

The rank condition (4.5) requires that $\text{rank}(\mathbf{M}_{12}) = l$, in which case one can write $\bar{\mathbf{J}}^+ = -\mathbf{M}_{12}^+\mathbf{M}_{11}$, reducing the rank condition to precisely the "Strong Inertial Coupling" condition described in [Spong, 1994a]. Now the command in (4.3) reduces to

$$\ddot{\mathbf{q}}_2 = -\mathbf{M}_{12}^+ \left[\mathbf{M}_{11}\mathbf{v} + (\mathbf{h}_1 + \boldsymbol{\phi}_1)\right] \tag{4.10}$$

The torque command is found by substituting $\ddot{\mathbf{q}}_1 = v$ and (4.10) into (4.2), yielding the same results as in [Spong, 1994a].

4.1.2 Redundancy Resolution

In situations where l is greater than p, there are infinite solutions to achieve the desired task-space command. One may use a null space projection to execute a second-priority task in actuator-space:

Theorem 2 (Task Space PFL w/ Redundancy Resolution)**.** *If the actuated joints are commanded so that*

$$\ddot{\mathbf{q}}_2 = \bar{\mathbf{J}}^+ \left[\mathbf{v} - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}_1\mathbf{M}_{11}^{-1}(\mathbf{h}_1 + \boldsymbol{\phi}_1)\right] + \alpha(\mathbf{I} - \bar{\mathbf{J}}^+\bar{\mathbf{J}})\ddot{\mathbf{q}}_2^{ref}, \tag{4.11}$$

and the condition (4.5) is met, then

$$\ddot{\mathbf{y}} = \mathbf{v}.$$

Proof. The proof is identical to Theorem 1, because the null-space terms are canceled by the pre-multiplication of $\bar{\mathbf{J}}$, and therefore do not contribute to $\ddot{\mathbf{y}}$. □

As discussed in the last chapter, redundancy resolution is a very important aspect of task space control, particularly for underactuated systems. The command $\ddot{\mathbf{q}}_2^{ref}$ can be thought of as a lower-priority task, which will be executed in the null-space of the high priority task. If the command is zero, the controller will return the solution with the smallest norm. More clever redundancy resolution tasks can help ensure the underactuated system stays dynamically within a regime where there are more actions that can be taken, a characteristic which can improve search performance.

4.1.3 Example: Feedback Control of the Acrobot

As an initial experiment, the task space controller was implemented on a two link arm with a passive base joint. From herein, N-link arms with mixtures of Passive and Actuated joints will be referred to by shorthand notation with combinations of **P's** and **A's** representing the respective type of joint. This two link system, **PA**, is known as the Acrobot [Spong, 1995]. Direct application of task-space PFL on this system *without planning* clearly demonstrates the power of the feedback linearization as well as the need for motion planning.

Specifically, the Acrobot was controlled to execute behaviors where the task space consists of either 1) the angle from the base to the COM, $\boldsymbol{\theta}_{COM}$ or 2) the angle from base to the end

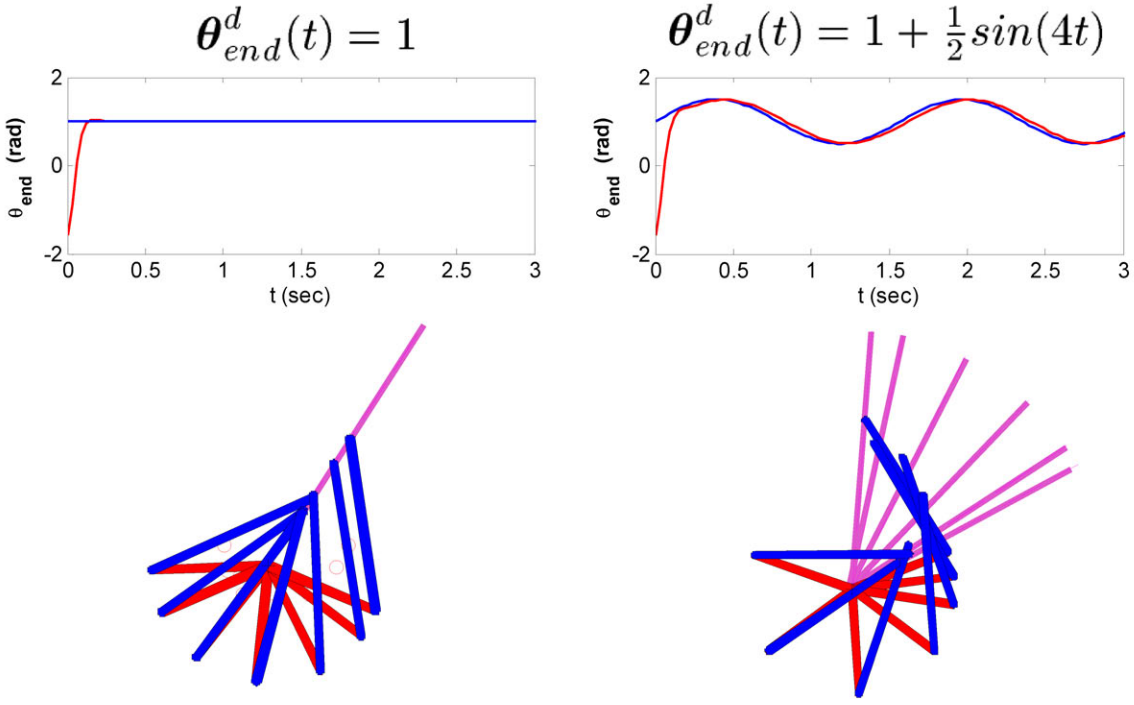


Figure 4-2: Acrobot Control: Top, trajectory of control output function (angle from base to end effector); Blue is desired, red is actual. Bottom, snapshots of Acrobot during execution. Red is first link, Blue is second link, Magenta is commanded angle. ©2008 IEEE [Shkolnik and Tedrake, 2008].

effector, $\boldsymbol{\theta}_{end}$. Two potential output trajectories were considered: 1) maintain a constant angle of 1 radian; $\boldsymbol{\theta}_d(t) = 1$, or 2) follow an arbitrary desired trajectory - in this case defined as $\boldsymbol{\theta}_d(t) = 1 + .5\sin(4t)$. When the task space is $\boldsymbol{\theta}_{COM}$ and the torque is unconstrained both output trajectories are followed well, but the controller commands excessive torques and the actuated joint spins. Similar results were obtained for controlling the end-effector angle, however in the special case if the second link is substantially longer than the first link, then the link does not spin, and instead the system "oscillates" back and forth along the line specified by $\mathbf{y}_{end_d}(t) = 1$, as shown in figure 4-2. This is a demonstration of how the choice of task space can drastically affect the system dynamics. If the goal is to perform swing-up behavior, for the Acrobot it may be better to control the end effector. As the controller can only operate on one degree of freedom with one actuator, treating the system as a pendulum and controlling the center of mass angle ignores the constantly changing length, which makes the pendulum (angle to COM) a poor choice of template. With redundant degrees of freedom for cases where $N > 2$, however, we may regulate both the angle and length of the pendulum, and so this becomes the task space of choice.

Similar results can be attained with higher DOF systems. These simulation experiments reveal the need for motion planning to enforce joint and torque limits. The dynamic constraint imposed by the rank condition in equation 4.5 was never actually violated - degeneracy in the Jacobian always presented itself with excessive torques before the mapping became truly singular.

4.2 Planning

Planning in task space, when done correctly, is a much lower-dimensional planning problem than planning directly in the state space of the robot. In a real sense, the control mapping from task space to (underactuated) joint space is a mechanism for exploiting known properties of the dynamics of the robot in order to dramatically improve the efficiency of search. To be practical, the task-space planner must respect constraints that exist only in the higher-dimensional state space (such as joint and torque limits). The general concept of planning in a low dimensional task space and verifying constraints in joints space is illustrated in Figure 4-1.

As shown in Chapter 2, planning in higher dimensions can be improved by considering actions in lower dimensional task space, ($\ddot{\mathbf{y}}$). Planning can be accomplished by computing a mapping from task accelerations to joint torques, then simulating the full system given those torques, and verifying that no constraints are violated. The intent is that the reduced set of potential actions, combined with the task space control mapping is presented in this chapter, can be powerful enough to enable the planner to find a path from start to goal. However, reachability to the goal will also depend on the choice of redundancy resolution. Depending on the redundancy resolution, the task controller does not necessarily carve out a manifold within state space. The controller constrains the actions, but the entire feasible state space may still be reachable. Thus, completeness remains a function of the higher-level planner, and is not necessarily constrained by the controller.

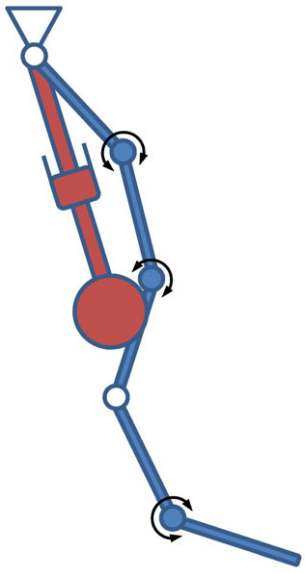


Figure 4-3: A low-dimensional template for the underactuated swing-up task. ©2008 IEEE [Shkolnik and Tedrake, 2008].

If the task space is of low enough dimension, a simple quasi-exhaustive tree search may be performed. More clever algorithms such as RRTs will have better performance, but I begin by exploring a near brute force search approach illustrated in Algorithm 6. Such an approach would become intractable if planning in a high-dimensional joint space, however tractability is maintained when using a reduced dimensional task space controller. Except for line 23, this is essentially a brute force search algorithm. The pruning step of line 23 sets a maximum branching value for each depth, and keeps the tree from expanding exponentially.

4.3 Generalized Swing Up Control for N-Link Pendulum with Passive Joints

4.3.1 Simulations of a 5-link pendulum

Consider a general, and classical, problem in underactuated control - the swing-up task (e.g., [Spong, 1994b]) on a multi-link arm in a gravitational field. The task is to move the center of mass of the pendulum to the vertical configuration. Many of the control solutions investigated in this problem use energy-based methods; for simple pendulum, regulating the total energy of the system is enough to place the system on a homoclinic orbit [Fantoni and Lozano, 2002]. Most work on multi-link pendulum with passive joints have focused on cases with 2 or 3 links [Spong, 1995,Lynch et al., 2000]. With more degrees of freedom, if searching directly for a torque trajectory over the actuators, the problem is crippled by the curse of dimensionality [Bellman, 1957]. In addition to increased system dimension, the problem is

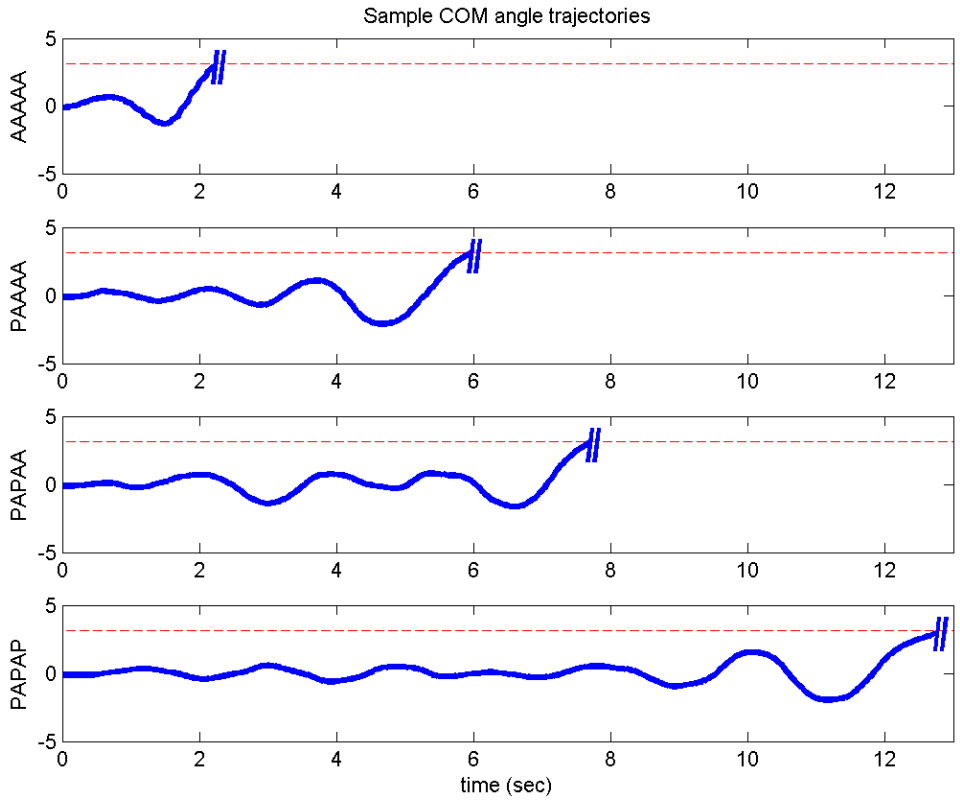


Figure 4-4: COM Trajectories. ©2008 IEEE [Shkolnik and Tedrake, 2008].

further complicated by considering limitations of joint positions, and torques.

For reasons alluded to previously, a two-DOF pendulum, with control over angle and length, was used as a template. This template and an example 5-Link pendulum is illustrated in Figure 4-3. Using this approach, the algorithm was run on the following systems: AAAAA, PAAAA, PAPAA, and PAPAP. In all cases, the motion planner was able to find a trajectory to the goal in reasonable time. Parameters of the system were as follows:

Mass of each link	.2kg
Length	.2m
Moment of Inertia	$.05kg \cdot m^2$
Torque Limitation	2 Nm

In practice, the method in Algorithm 7 was used for pruning, where the metric function, M, was based on the energy difference between a given state and the goal, and penalized by a weighted sum of joint space distance away from the straight pose, where joints closest to the base were more heavily weighted and therefore encouraged to be straighter than joints further from the base. This metric encourages pumping energy into the system, but tries to stay away from poses which wrap around. Nodes are probabilistically pruned, biased by

Algorithm 6 Simple Planning

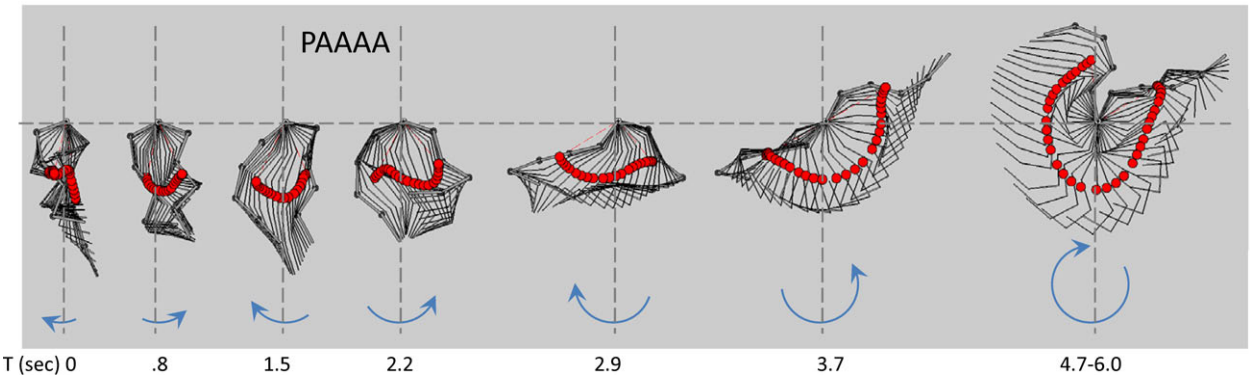
Require: discrete set of actions, $V \in \mathbb{R}^{Nxp}$, Depth limit of D, Depth node limit of B

Add-node (x_0, root)
for depth $d = 1$ to D **do**
  $n \Leftarrow 0$
  for each node, x_n at depth d **do**
    for each $V_i \in V$ **do**
      $tau_H \Leftarrow$ task-control(x_n, V_i)
      if $|tau_H < tau_{LIMIT}|$ **then**
        Simulate: $x_{new} \Leftarrow f(x_n, tau_H, \Delta T)$
        if NO-CONSTRAINTS-VIOLATED(x_{new}) **then**
          Add-node(x_{new}, X_n)
          $n \Leftarrow n + 1$
        end if
      end if
    end for
  end for
  if Goal Reached **then**
    RETURN (GOAL REACHED)
  end if
  if n == 0 **then**
    RETURN (NO PATH FOUND)
  end if
  if $n > B$ **then**
    prune n-B leaves from depth d
  end if
end for

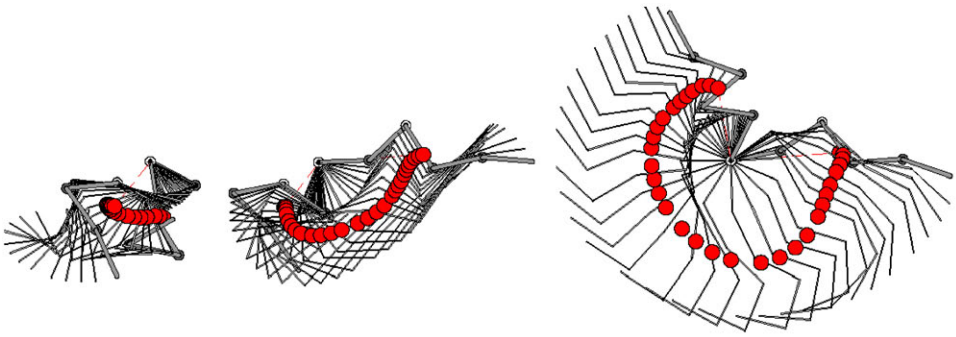
Algorithm 7 Probabilistic Pruning

Require: set of leaves, $\mathbf{X}_{leaves}$, leaf limit B, goal $\mathbf{X}_{goal}$, $0 < \alpha < 1$

Compute a distance metric: $H = M(\mathbf{X}, \mathbf{X}_{goal})$, where H is normalized between 0 (furthest) and 1 (closest). This metric can also account for "'bad poses"' or any other costs on the state.
Sort($(\alpha(H/2) + (1 - \alpha)$)*Random-Number-Generator
Return best N nodes



(a) PAAAA



(b) PAPAA

Figure 4-5: Swingup trajectory of PAAAA and PAPAA. ©2009 IEEE [Shkolnik et al., 2009].

the metric, a step that helps to ensure that some feasible paths remain if most high-scoring nodes are actually headed to a dead-end.

Example center of mass trajectories for all four cases are compared in Figure 4-4. An entire swing-up trajectory is illustrated for the PAAAA case in Figure 4-5(a), and the final three swings for the PAPAA case are shown in Figure 4-5(b).

4.3.2 Computational Efficiency

It is easy to see that Algorithm 6 is

$$O(D \cdot K \cdot N^{(2p)}).$$

This is substantially better than the case without pruning, which would grow exponentially on D.

To illustrate the time savings offered by the dimensionality reduction, consider the PAAAA example of the preceding section, with 1 passive joint, 4 actuators, and a 2-DOF task space consisting of the angle and length to COM (l = 1, m=4, p=2). Let $K = (B/N^p)$

be the average branching factor per node. In the results presented above, I used K=2, D=65 (ΔT=.15), N=11. Then if the search is done in the lower dimensional task space where p=2, the search time is $O(D * K * N^{(2p)}) \propto 2$ million nodes. If instead the search was attempted with the same discretization factor in the full joint space (note the branching factor blows up to $B = K * N^m$), the equivalent time would be $O(D * K * N^{(2p)}) \propto 30$ billion nodes.

4.3.3 Task space guided RRT

The planning algorithm presented above was a simple, quasi brute-force approach, and was meant to demonstrate the power of the dimensionality reduction in the search, as a planning algorithm with similar search density in the full state space would have been computationally intractable. The contribution here is in addressing the underactuated control as a search problem, and using task space to constrain the search. Note, however, that any planner can be used if it satisfies the framework of Figure 4-1.

The Task-Space guided RRT satisfies the approach of Figure 4-1. In fact, the control

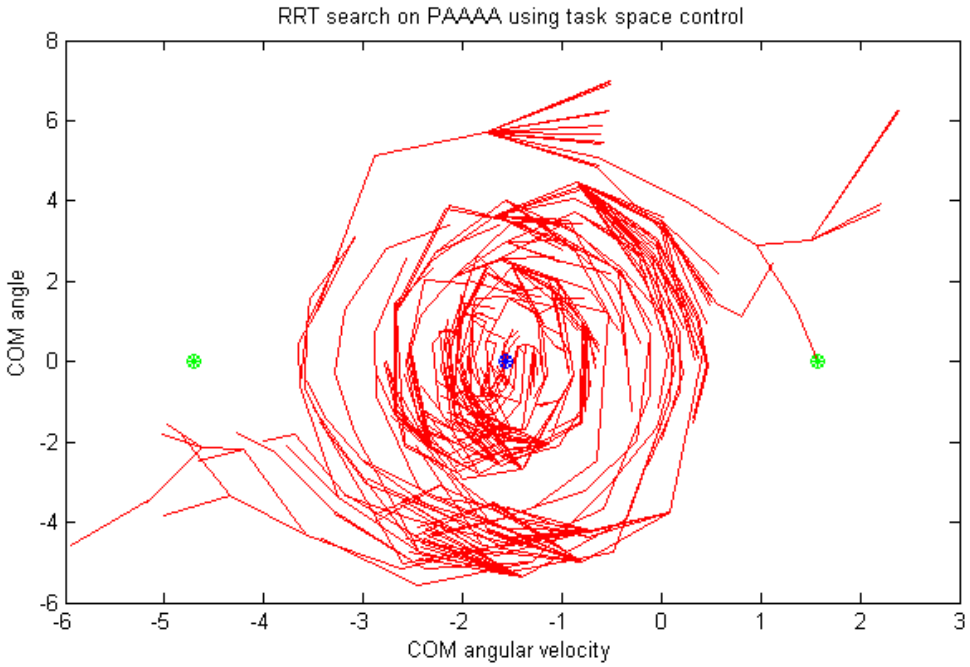


Figure 4-6: This figure shows the projection of an RRT tree built on a PAAAA system, showing the system COM angle and angular velocity. The projection of the root of the tree is shown by the blue circle, and the goal points are shown in green (wrapped around 2*pi). One can see that this projection looks much like an RRT for a pendulum. Using the COM as a task space, the Task Space Guided RRT is able to quickly find plans for swing up on this system.

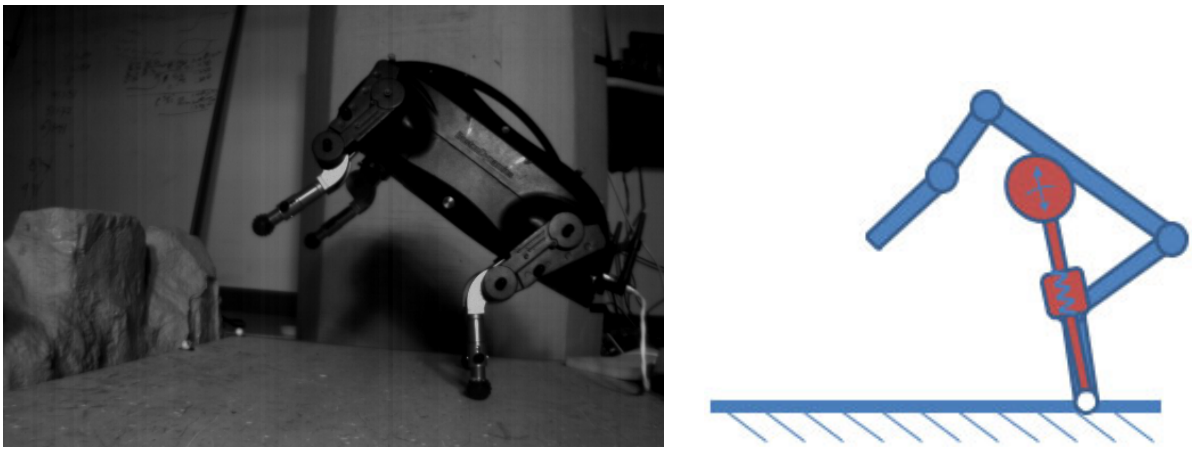


Figure 4-7: LittleDog: Template and Anchor. ©2008 IEEE [Shkolnik and Tedrake, 2008].

specified by equation 4.11 can be directly plugged in to line 5 of the CONTROL() function in the modified RRT algorithm presented in Algorithm 5, in section 2.2.2. This results in a fast RRT based planning algorithm that utilizes the task space feedback controller to constrain explored actions. An example of the result found with this RRT on a PAAAA system, after approximately 20 seconds of search on a standard quad-core Pentium computer, is shown in Figure 4-6. This figure illustrates a projection of the RRT tree from the 10 dimensional state space to the COM angle and COM angular velocity. The COM length is part of the task space, but is neglected here for clarity. It is interesting that despite being a 5 link system, the projection looks very much like an RRT run on a 1-D torque limited pendulum.

4.4 Discussion

4.4.1 Choice of template

A first attempt to generate a swing up controller on the 5 link arm with passive joints, was done by using a 1-DOF pendulum as a template. It seemed logical that length might be stabilized in the Null space, and the planner could focus on increasing energy by controlling only the angle of the COM. In practice, the torques required to keep the 5-link pendulum of constant length were too great, and exhaustive search failed. Allowing the length to vary lets the system act more like a spring. For example, when the COM angular velocity is high, and inertia is greatest, increasing the COM-length helps minimize torques and keeps the system within torque bounds.

4.4.2 Applications to locomotion

Many legged robots can be modeled by rigid body dynamics. Consider for example a sagittal model of the LittleDog quadruped robot. When the robot is standing on its back legs, and

if the robot is moving the two back legs in unison, and moving the two front legs in unison, then the system becomes equivalent to a 5-link PAAAA model. Figure 4-7 shows the robot in this position, as well as a corresponding 5-link model and a 2-DOF template.

The techniques developed in this chapter may be utilized in a bounding gait planner, which can be stabilized in task space using partial feedback linearization. Such stabilization may allow for slight corrections in COM trajectory if possible. Figure 4-8 illustrates an example of this for LittleDog. The top row of this figure illustrates two trajectories. For simplicity, both images in the top row show a trajectory initialized to the same joint pose, and the actuated joints remain in this fixed pose throughout the trajectory. The passive joint at the ground is initialized at 2.8 rad/sec (Left), or 2.0 rad/sec (Right). The figure on the

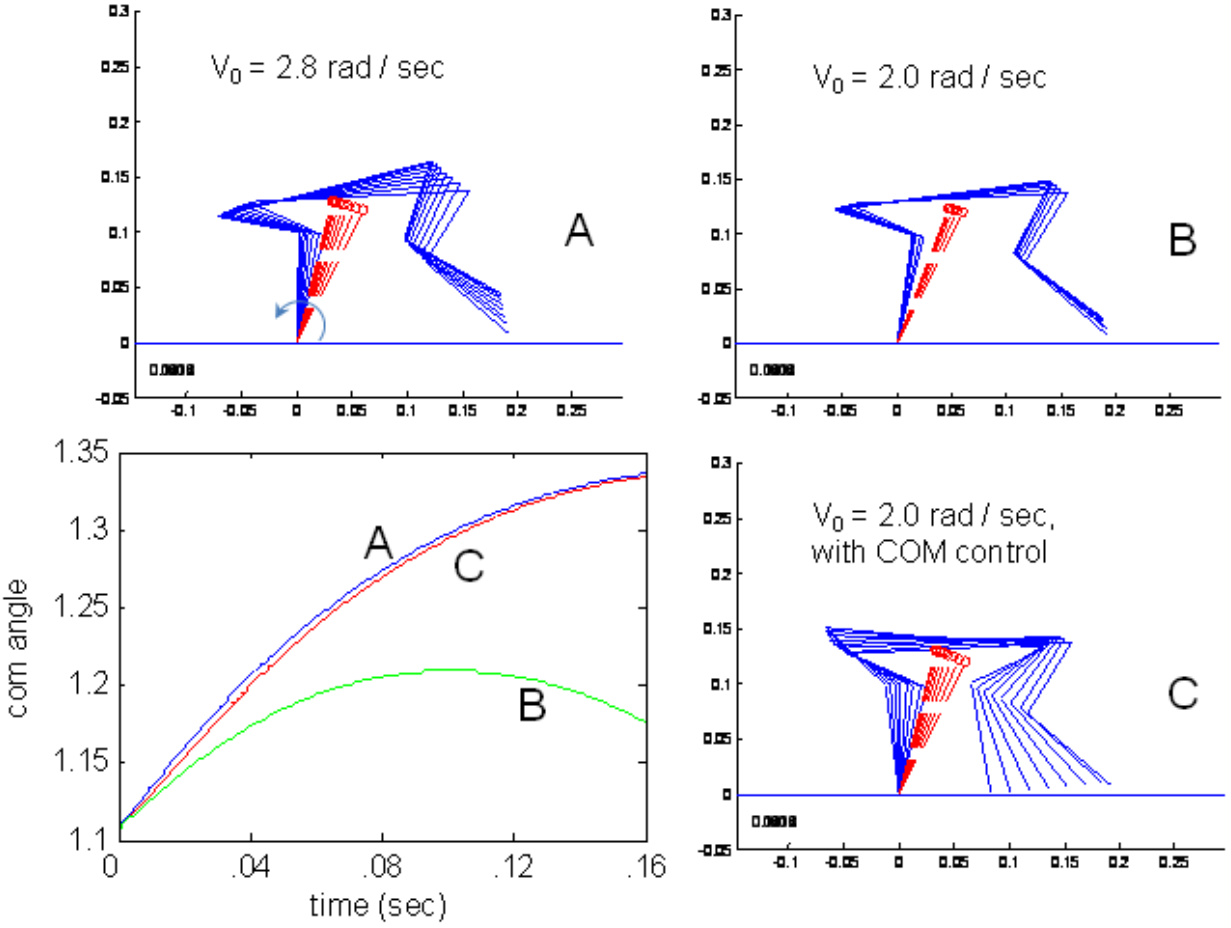


Figure 4-8: **A:** Trajectory with initial COM angular velocity of 2.8 rad/sec; joints held in constant pose. **B:** Trajectory with initial COM angular velocity of 2.0 rad/sec; joints held in constant pose. **C:** Initial COM angular velocity of 2.0 cm/sec; joints are actuated using COM / PFL control, to try to emulate the COM trajectory with initial velocity of 2.8 rad /sec. **Bottom Left:** COM vs time comparing the three trajectories.

bottom left shows the COM angle trajectory over time, illustrating how the two trajectories (A and B) diverge over time. In C, we show that by applying PFL, the robot is able to emulate the COM trajectory in A (2.8 rad/sec), even though it started with a passive joint velocity of 2.0 rad/sec.

An interesting observation is that because the actual robot has 12 actuated DOF, the null space may be utilized for foot placement. However, in practice, planning or PFL stabilization for the LittleDog robot is difficult because the inertial coupling of the limbs is low. Because of this, slight modifications to the COM trajectory require a relatively large joint movement. Furthermore, constraints such as torque and velocity limits break down the capabilities of a pure PFL controller. However, there are related applications, in which inertial coupling may be stronger, for example in humanoid robots. The application of a task space controller in these applications is left to future work.

4.4.3 Concluding Remarks

In this chapter I demonstrated a framework for planning in underactuated systems by utilizing task space control and partial feedback linearization. The power of this approach was demonstrated by application of a feedback controller in the Acrobot (PA), which was able to closely track an arbitrary trajectory of its end-effector. Feedback control can fail due to constraints on the system, including torque limitations, physical obstacles in the environment, joint limits, etc. The task space controller was used in a simple motion planner, as well as in a TS-RRT in order to quickly search for trajectories from start to goal without violating constraints. These planners were able to take advantage of the reduced action space imposed by the task space control. Planning was successful in producing swing-up behavior in 5-link models with anywhere from 0 to 4 passive joints. This toy problem lays the foundation for real-world applications, including planning and stabilizing dynamic gaits of a quadruped robot, and toe-roll in a humanoid.

In order to reduce the dimensionality by the task space mapping, the algorithm requires strong inertial coupling. If there is weak inertial coupling, as in the case for LittleDog, which has very light limbs, the PFL controller is not able to produce arbitrary accelerations in the task space without violating torque and other constraints of the full system. For these cases, alternative planning strategies that take into account the dynamics should be considered. One idea to do this is presented in the next chapter.

Chapter 5

Reachability-Guided Sampling for Planning Under Differential Constraints

Chapters 2 - 4 have focused on the idea of using task space control to reduce the dimensionality when planning on high-dimensional systems. The last chapter in particular considered task space control and motion planning for systems with underactuated dynamics. This chapter addresses planning for more general problems with differential constraints. Obstacle fields with tunnels, or tubes are notoriously difficult for most planning algorithms to handle, including RRTs. Differential constraints are particularly challenging. The essential symptom of this inefficiency is that nodes at the boundaries of these constraints tend to be sampled and expanded upon repeatedly, but little progress is achieved in expanding the tree towards unexplored regions of state space.

In this chapter I describe the Reachability Guided RRT (RG-RRT), which uses a new sampling strategy for the RRT algorithm, based on an approximation of the regions of state space that are local-time reachable from the tree. Taking into account local reachability incorporates information about the dynamics, or any other differential constraints, into the sampling strategy, and provides a dramatic improvement in performance in these severely constrained systems. The reachability-guided sampling method is based on the observation that sampling points at random and checking collisions is relatively cheap, but going through the expansion step of the RRT, which typically requires integrating dynamics, is an expensive step - both in its instantaneous cost and in the added cost of having a larger tree. Therefore, the algorithm attempts to build sparse trees through kinodynamic obstacles, with a simple heuristic that quickly rejects samples if it is likely that the tree can not grow in the direction of this sample. This type of rejection sampling effectively changes the sampling distribution, so that points are selected uniformly from a potentially small portion of the configuration (or state) space in which the tree is capable of growing.

In order to illustrate the mechanisms involved with the new sampling strategy, the focus in this chapter is on a low-dimensional torque-limited pendulum swing-up example, where the Voronoi regions can be illustrated and the entire tree can be visualized effectively. To illustrate the generality of the approach, planning for a nonholonomic vehicle through a tight

corridor is also presented. In both cases, the algorithm makes a substantial improvement in planning speed. Applicability of the algorithm to high dimensional systems is briefly discussed at the end of this chapter, and is addressed in more detail in the next chapter.

Significant portions of this chapter also appear in [Shkolnik et al., 2009].

5.1 RRT Planning Under Differential Constraints

This section describes the performance of the RRT in the presence of kinodynamic constraints, with an emphasis on planning for underactuated systems. in state space.

5.1.1 Basic RRT Operation

Recall the basic RRT algorithm discussed in section 1.3.1 and shown in Algorithm 1. The pattern by which nodes are selected for expansion in an RRT is a function of both the sampling distribution and the nearest-neighbor distance metric. Typically, samples are drawn uniformly over the state space while the most common metric for the nearest-neighbor selection is the Euclidean distance between points. In this case, the expansion pattern of the tree is modeled by the Voronoi diagram over the nodes within the tree. The probability of a node being expanded is directly proportional to the volume of its corresponding Voronoi region. Nodes that have a larger Voronoi region are more likely to be chosen for expansion and are referred to as *major* nodes. In the case of the Euclidean metric, these nodes tend to lie on the outside of the tree during the initial exploration. Conversely, *inner* or *minor* nodes have smaller Voronoi regions and often lie on the inside of the tree. Once the tree has explored the state space, these become major nodes as the algorithm begins to fill in the space. This phenomenon of favoring some nodes over others is referred to as the *Voronoi bias*, and yields an initial preference towards the exploration of the state space. Over time, the Voronoi regions become more uniform in volume and, in the case of planning for holonomic systems, the likelihood of expanding a node tends toward the sampling distribution [Kuffner et al., 2000].

5.1.2 Performance with Kinodynamic Constraints

The efficiency by which the vanilla RRT algorithm is able to grow the tree and explore the space is highly sensitive to the distance metric. In the presence of kinematic constraints (e.g., joint limits, obstacle collisions) and dynamic constraints (including torque limits, and the equations of motion), widely-used metrics like the Euclidean distance are a very poor representation for the true distance between points in the space. For a given state, the equations of motion and actuator limitations prevent the moving in arbitrary directions in state space. As a result, the RRT will repeatedly attempt to extend the same nodes without growing the tree any closer to the sampled region.

Consider the swing-up task for a torque-limited pendulum. The two-dimensional state space consists of the joint angle and the angular velocity, $x = \left[\theta,\ \dot{\theta}\right]^\top$. The task is to bring

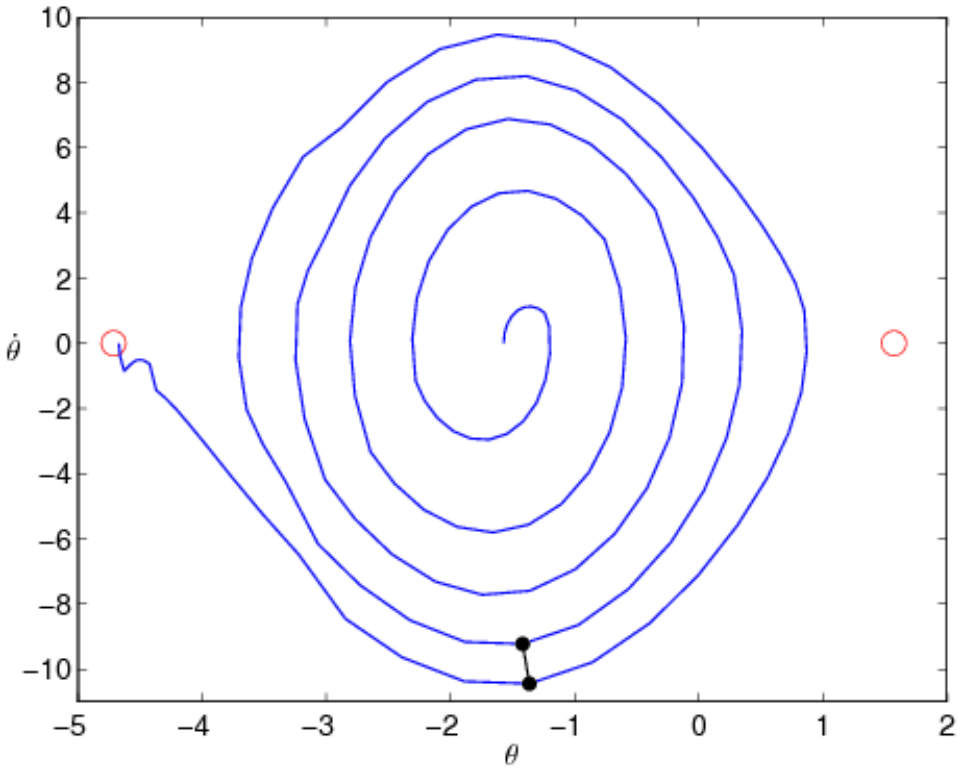


Figure 5-1: Example state space trajectory for the torque-limited, single link pendulum. Due to the torque limits at the shoulder, a swing up phase is necessary to drive the pendulum from the vertical downward position to the stationary upright position. The two black dots connected by a segment shows that the Euclidean metric is not a good measure of cost-to-go. ©2009 IEEE [Shkolnik et al., 2009].

the pendulum from a stationary down position, $x_{\text{init}} = \begin{bmatrix} -\pi/2, & 0 \end{bmatrix}^\top$, to the fixed upright goal pose, $x_{\text{goal}} = \begin{bmatrix} \pi/2, & 0 \end{bmatrix}^\top$. If the torque-limits are severe, then a swing up phase is required in order to drive the system to most states in the state space, including the goal pose. The spiral state trajectories in Figure 5-1 demonstrate this behavior as the pendulum is swung back and forth to reach the goal.

The Euclidean distance metric is ignorant to the fact that the differential constraints on the pendulum dynamics require a swing up phase to reach most of the state space. Consider the pair of points connected with a black line, shown in Figure 5-1. The Euclidean distance metric would suggest that this pair is very close to each other, but the pendulum is unable to transition between these two parts of state space without swing up. While RRT-based planners have successfully solved swing up control for the underactuated pendulum [Branicky and Curtiss, 2002] using the Euclidean metric, it comes at a cost of expanding a large number of fruitless nodes. This might be tolerable for low-dimensional systems like the single-link pendulum, but can become intractable for higher-dimensional systems for which sample-based planners are typically well-suited.

5.1.3 RRT Modifications

The motion planning literature contains a number of attempts to improve the performance in constrained systems. One solution is to use a metric other than the Euclidean distance that is better-suited to the problem. Perhaps an ideal metric would be the optimal cost-to-go in terms of the time or energy required to drive the system between a pair of states [Frazzoli et al., 2002, Glassman and Tedrake, 2010]. Unfortunately, computing the ideal cost is intractable for most higher-order systems since it is equivalent to optimally solving the original motion planning problem [LaValle, 2006]. Instead, RRT-based planners can utilize a sub-optimal heuristic function that has been tuned to the particular problem as the distance measure. The metric is not as accurate as the optimal cost-to-go, but typically performs much better than the Euclidean distance. Unfortunately, these metrics rely upon domain-specific knowledge and are not generalizable across different systems.

Rather than design a system specific metric for planning, another option is to adaptively learn a suitable metric during planning. Such is the approach of Cheng [Cheng, 2005, Ch. 6], who scores nodes according to their consistency with constraints as well as that of their children in the tree. These values are updated while building the tree and used to select nodes during exploration. The RRT extension proposed in [Kim et al., 2005] is similar, in which the algorithm keeps the history of expansion attempts on nodes, considering failed attempts as both collisions, as well as repetition of previous expansions. Furthermore, in place of the Euclidean distance metric, this algorithm calculates the cost-to-go from each node in the tree toward the sample using a linear approximation of the dynamics for each node. The approach alleviates some of the problems with the Euclidean distance metric, but can be computationally expensive to compute, and is sensitive to how well the linearization approximates the actual cost-to-go.

The dynamic-domain RRT [Yershova et al., 2005, Yershova, 2008] combats futile oversampling by altering the size of the Voronoi regions for those nodes that are close to obstacles. By lowering the bias associated with these nodes, the dynamic-domain RRT reduces the likelihood of drawing samples from regions of the state space that are more likely to induce collisions. More specifically, the algorithm identifies boundary nodes as those for which a collision occurred while expanding a sample. The domain of samples that can then be matched with a boundary node is then restricted to a sphere of fixed radius in the state space. One limitation of the dynamic-domain RRT is that it must first encounter collisions before modifying the Voronoi bias. Additionally, it currently supports only spherical domains centered about the boundary nodes, which do not necessarily reflect the best sampling regions.

Other modifications to the RRT assume a discretized set of actions. In the RRT-Blossom algorithm [Kalisiak, 2008], whenever a node is chosen for expansion, all possible (discrete) actions are attempted, while sparse expansion is encouraged by weeding out any expansions of the tree if the expanded node is closer to other nodes in the tree other than the parent. Once a node is visited once, it never needs to be visited again, so redundant exploration is a non-issue. However, in some problems (for example the pendulum) it does not make sense to remove child nodes in this way since child nodes will be near other nodes in the

tree, unless the integration time step Δt is very small, which will increase the size of the resulting tree exponentially. Another variant that assumes discrete actions, but was only demonstrated on path planning problems without differential constraints, is the Rapidly-Exploring Random Leafy Tree (RRLT) [Morgan et al., 2004]. Leaf nodes, corresponding to nodes that are reachable in one time step from the current tree are stored in the tree. The algorithm picks the closest leaf node to a random sample, converts it to a main node in tree, and then generates new reachable leaf nodes from this new node.

5.2 Reachability-Guided RRT

The previous section presented an analysis of the shortcomings of existing RRT-based planners in dealing with systems with differential constraints. In particular, there was an emphasis on the sensitivity of RRTs to the nearest neighbor metric and the implications that this has for building the tree.

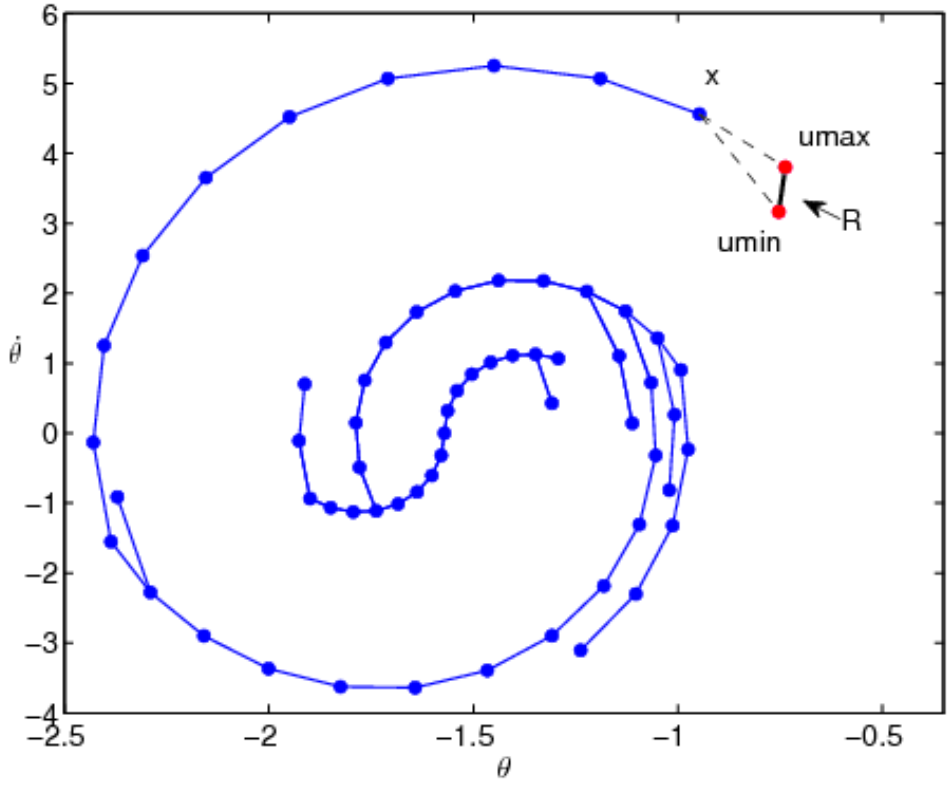


Figure 5-2: The reachable set, $\mathcal{R}(x)$, for the underactuated pendulum consists of an approximately linear segment, the endpoints of which are defined by the points in state space that are achieved by applying the minimum and maximum torques at the shoulder. ©2009 IEEE [Shkolnik et al., 2009].

In this section the Reachability-Guided Rapidly-exploring Random Tree (RG-RRT) is presented, as an alternative planning strategy for general systems that are subject to differential constraints. The RG-RRT takes the form of a modified RRT that explicitly accounts for the limitations of the system dynamics to alter the Voronoi bias so as to emphasize nodes within the tree that exhibit the greatest contribution towards exploring the state space. The RG-RRT alleviates the sensitivity to the distance metric and, in turn, does not require a system-specific metric heuristic. The result is an expansion of the tree that makes efficient use of the system dynamics to more rapidly reach the goal.

5.2.1 Problem Formulation

Consider the problem of motion planning for a system subject to differential constraints. Let $\mathcal{X}$ denote the state space that applies to kinodynamic systems and let $\mathcal{C}$ represent the general configuration space. The objective is to find a time-varying control input, $u(t) \in \mathcal{U}$ for $t \in [0, T]$ that drives the system from some initial state, x_{init}, to an arbitrary goal state, x_{goal}, in finite time, T. The resulting trajectory, $x(t)$, must be free of collisions, i.e. $x(t) \in \mathcal{X}_{\text{free}}$ where $\mathcal{X}_{\text{free}}$ denotes free space, and satisfy the differential constraints according to the state transition function,

$$\dot{x} = f(x, u). \tag{5.1}$$

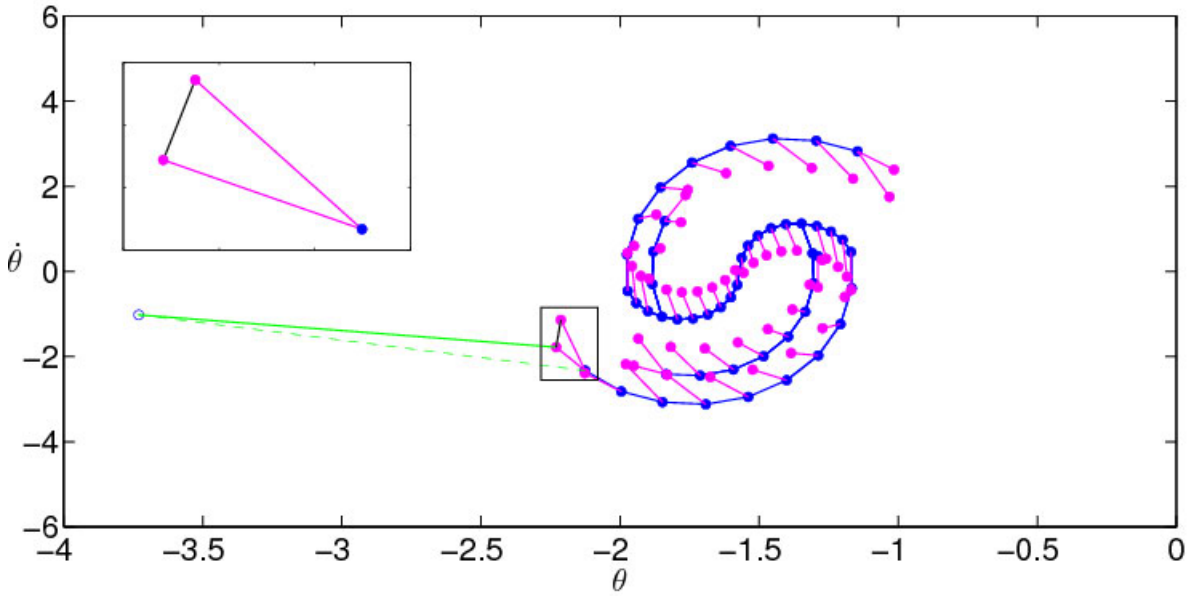
The problem becomes significantly more challenging when considering systems that are additionally constrained to be underactuated or nonholonomic.

5.2.2 The RG-RRT Algorithm

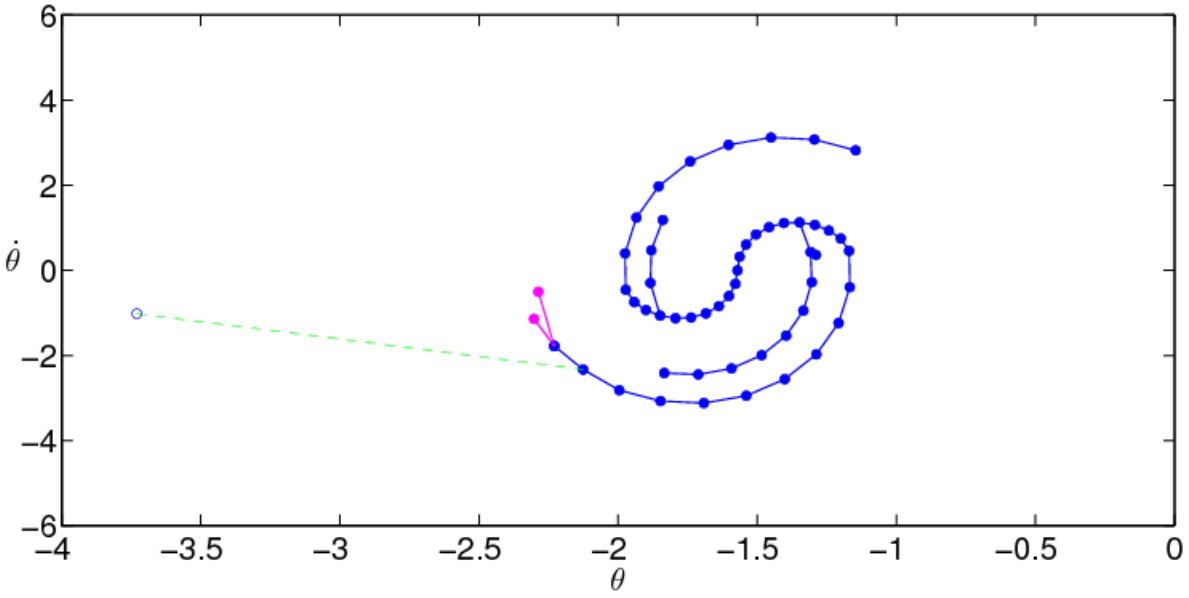
The Reachability-Guided RRT is derived from the premise that sampling is relatively inexpensive compared to the process of searching for collision-free trajectories from a node while satisfying differential constraints. The RG-RRT provides a means for choosing nodes that have a better chance of yielding consistent trajectories without having to redefine the metric function. Instead, the RG-RRT actively constrains the set of nodes under consideration for nearest-neighbor pairing with a sample to those that are actually able to expand towards the given sample. Effectively, it applies the Euclidean metric to nodes for which it is a valid indication of the approximate cost-to-go under the differential constraints. The RG-RRT does so by considering the reachable region of the state space associated with each node.

Definition 6. *For a state $x_0 \in \mathcal{X}$, we define its* reachable set*, $\mathcal{R}_{\Delta t}(x_0)$, to be the set of all points that can be achieved from x_0 in bounded time, $\Delta t \in [\Delta T_{low}, \Delta T_{high}]$, according to the state equations* (5.1) *and the set of available control inputs, $\mathcal{U}$.*

In the case of the torque-limited single-link pendulum, the reachable set for a state, assuming a constant time step, is a curved segment. For simplicity, this segment can be



(a)



(b)

Figure 5-3: Two consecutive steps in a single iteration of the RG-RRT, as demonstrated for the underactuated pendulum. In (a), a random sample is drawn from the state space and paired with the nearest node in the tree and the closest point in its reachable set according to the Euclidean distance metric. Shown in (b), the algorithm then expands the node towards the point in the reachable set and adds that point to the tree. ©2009 IEEE [Shkolnik et al., 2009].

approximated as a straight segment, the bounds of which are found by integrating the dynamics while applying the maximum negative and positive torque, $-u_{\max}$ and $u_{\max}$. Figure 5-2 depicts the reachable set approximation for a node in the tree. Any state along the segment can be (approximately) achieved in time Δt by an allowable control input, $|u| \leq u_{\max}$.

Algorithm 8 $\mathcal{T} \leftarrow$ BUILD-RG-RRT(x_{init})

```
1: T ← InitializeTree(x_init);
2: R ← ApproxR([ ], T, x_init);
3: for k = 1 to K do
4:    RejectSample ← true;
5:    while RejectSample do
6:       x_rand ← RandomState();
7:       ([ ], dist_T) ← NearestState(x_rand, T);
8:       (r_near, dist_R) ← NearestState(x_rand, R);
9:       if dist_R < dist_T then
10:         RejectSample ← false;
11:         x_near ← ParentNode(r_near, R, T);
12:      end if
13:   end while
14:   u ← SolveControl(x_near, x_rand);
15:   [x_new, isFeasible] ← NewState(x_near, u);
16:   if isFeasible then
17:      T ← InsertNode(x_new, T);
18:      R ← ApproxR(R, T, x_new);
19:      if ReachedGoal(x_new) then
20:         return T
21:      end if
22:   end if
23: end for
24: return [ ]
```

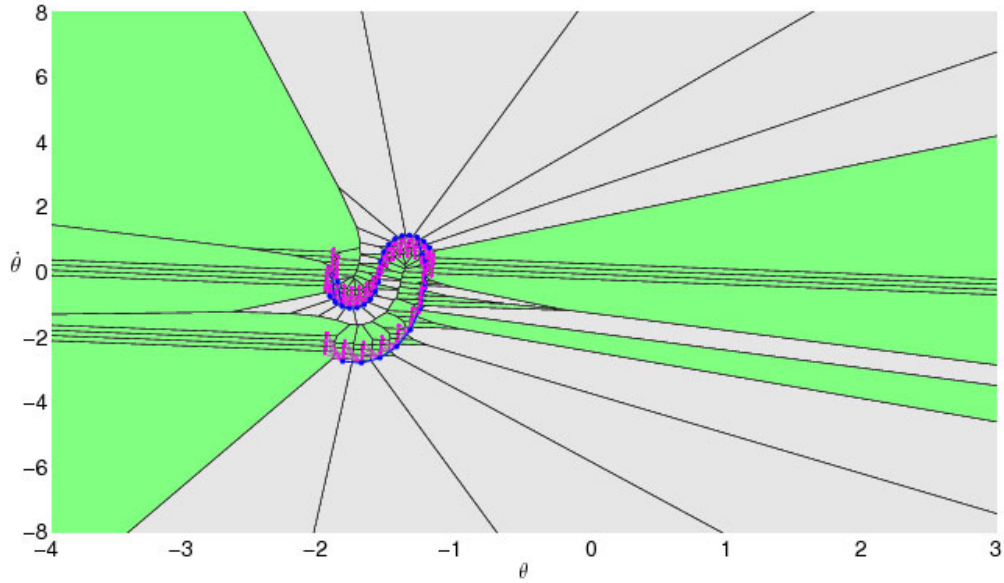
The structure of the RG-RRT algorithm is outlined in Algorithm 8. Given an initial point in state space, x_{init}, the first step is to initialize the tree with this state as the root node. Each time a node is added to the tree, the ApproxR() function solves for $\mathcal{R}_{\Delta t}(x_{new})$, an approximation of the set of reachable points in the state space that are consistent with the differential constraints (5.1). The approximated reachable set for the whole tree is stored in a tree-like structure, $\mathcal{R}_{\Delta t}(\mathcal{T})$, or simply $\mathcal{R}$ for shorthand notation, which contains reachable set information as well as pointers to the parent nodes for each node in the corresponding tree, $\mathcal{T}$. For many systems of interest, the approximate bounds of the reachable set, $\mathcal{R}_{hull}$, can be generated by sampling over the limits in the action space, and then integrating the corresponding dynamics forward. For these systems, assuming a relatively short action time step, Δt, integrating actions between the limits results in states that are within, or

reasonably close to being within, the convex hull of $\mathcal{R}_{hull}$. In such cases it is sufficient to consider only the bounds of the action set to generate the reachability approximation. When the dimension of the action space becomes large, it may become more efficient to approximate the reachable set with a simple geometric function, such as an ellipsoid. We found that the reachable set approximation does not need to be complete, and even a crude approximation of the set can vastly improve planner performance in systems with dynamics. Another benefit of approximating the reachable set by sampling actions is that the resulting points can be efficiently tested for collisions before they are added to the Reachable set approximation. This reduces the likelihood of trajectories leaving free space as part of the exploration phase.

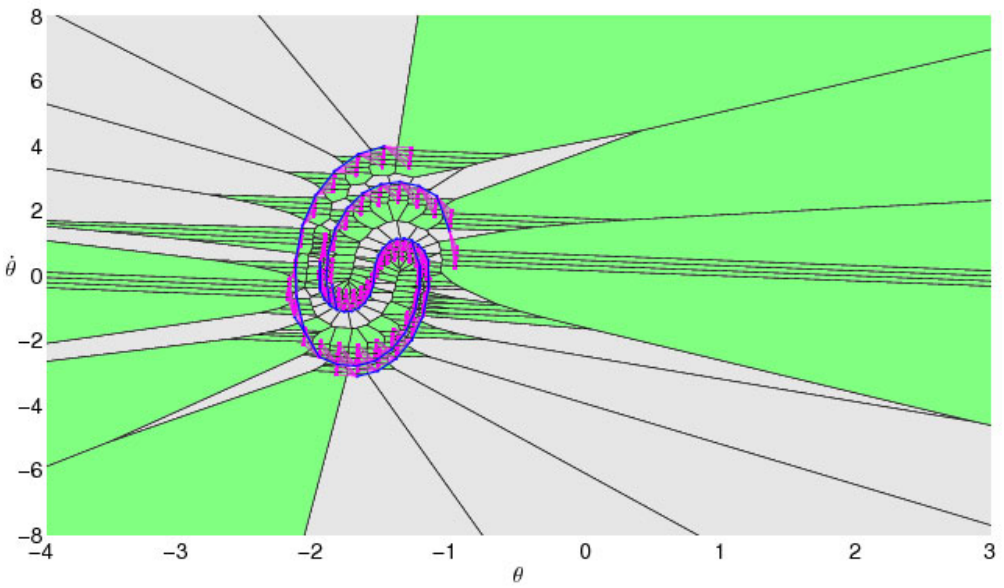
After a node and its corresponding reachable set is added to the tree, a random state-space sample, x_{rand}, is drawn. The NEARESTSTATE$(x_{\text{rand}}, [\mathcal{T}, \mathcal{R}])$ function compares the distance from the random sample to either, the nodes of the tree, $\mathcal{T}$, or to the reachable set of the tree, $\mathcal{R}$. The function returns the closest node, and distance from the node to the sample. Samples that are closer to the tree, $\mathcal{T}$, rather than the reachable set, $\mathcal{R}$, are rejected. Sampling continues until a sample is closer to $\mathcal{R}$, at which point, the $ParentNode$ function returns the node in $\mathcal{T}$, which is the parent of the closest reachable point in $\mathcal{R}$.

Figure 5-3 demonstrates this process for the pendulum example. The figure shows a tree grown from the initial state corresponding to the pendulum pointing down, with zero rotational velocity. The nodes and edges comprising the tree are in blue, while the magenta points that emanate from each node correspond to the approximation of the Tree's reachable set. In Figure 5-3(a), a sample was drawn from the state space. The Euclidean distance from this sample to each node in the Tree, and to each Reachable node was computed. This sample was identified as being closest to the Reachable set. The parent of the closest Reachable node (shown with the dashed green line) is then selected for expansion.

The result of this policy of rejecting samples for which the nearest node in the Tree is closer than its Reachable set is a change in the Voronoi bias. The RG-RRT allows samples only from Voronoi regions for which the differential constraints permit the expansion of the node towards the sample. While samples may be drawn from anywhere within the state space, only a subset of regions are actually used to grow the tree. This has the effect of modifying the Voronoi bias to emphasize nodes that are better suited to explore the state space while growing the tree. Figure 5-4 depicts this phenomenon for the pendulum. The plot presents the Voronoi regions for each of the nodes in $\{\mathcal{R}, \mathcal{T}\}$ and identifies in green the regions for which drawn samples are deemed as valid. Many of the large Voronoi regions, associated with outer nodes of the tree, have appropriately been identified as not suitable for expansion, since it is impossible to grow the tree into these regions in time Δt under the differential constraints. The nodes may still be expanded upon, but towards samples drawn towards the inside of the tree. These nodes, which would otherwise serve as major nodes with standard RRT-based planners using the Euclidean distance, are instead treated as minor nodes.

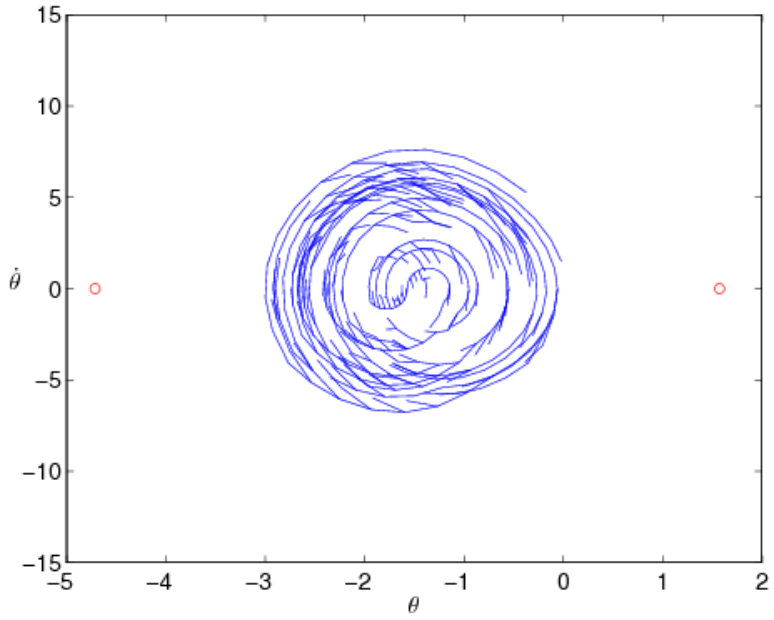


(a) 30 nodes

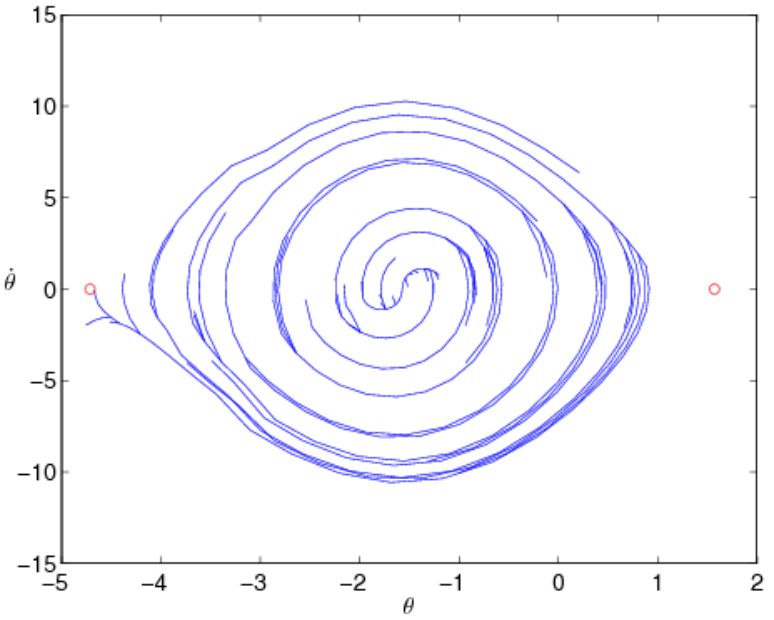


(b) 60 nodes

Figure 5-4: RG-RRT Voronoi diagrams for a pendulum. The blue diagram in (a) corresponds to the tree after 30 nodes have been added while that in (b) corresponds to the tree with 60 nodes. Magenta dots are discretely sampled reachable points affiliated with the tree. Green regions are areas where samples are 'allowed', and correspond to Voronoi areas associated with the reachable set. Samples that fall in any grey areas are discarded. Note that these regions are constantly changing as the tree grows. ©2009 IEEE [Shkolnik et al., 2009].



(a) Standard RRT



(b) RG-RRT

Figure 5-5: The trees for the underactuated pendulum after adding 360 nodes for (a) the standard RRT-based kinodynamic planner and (b) the RG-RRT planner. While the RG-RRT algorithm has reached the goal state, the RRT-based planner has yet to explore much of the state space. The RG-RRT converged in 3 seconds. ©2009 IEEE [Shkolnik et al., 2009].

Upon identifying a suitable node for expansion, the RG-RRT extends the tree from the node. The SolveControl($x_{\text{near}}, x_{\text{rand}}$) function defines an action that (with probability > 0) drives the system from the state towards the sample. Figure 5-3(b) demonstrates this step where the node has been extended. The new point, x_{new}, is added as a node to the tree, and its reachable set, $\mathcal{R}(x_{\text{new}})$, is computed. The RG-RRT then continues with the next iteration of the algorithm.

5.3 Results

This section describes results of implementing the RG-RRT on two different systems in simulation. The first application is to find swing up trajectories for the single-link, torque-limited pendulum. The second application is to solve for collision-free trajectories for a simple nonholonomic car driving amongst obstacles. For comparison, both simulations were also compared against the standard RRT. The two planners used a uniform distribution over the state space to generate samples and the Euclidean metric to identify nodes for expansion.

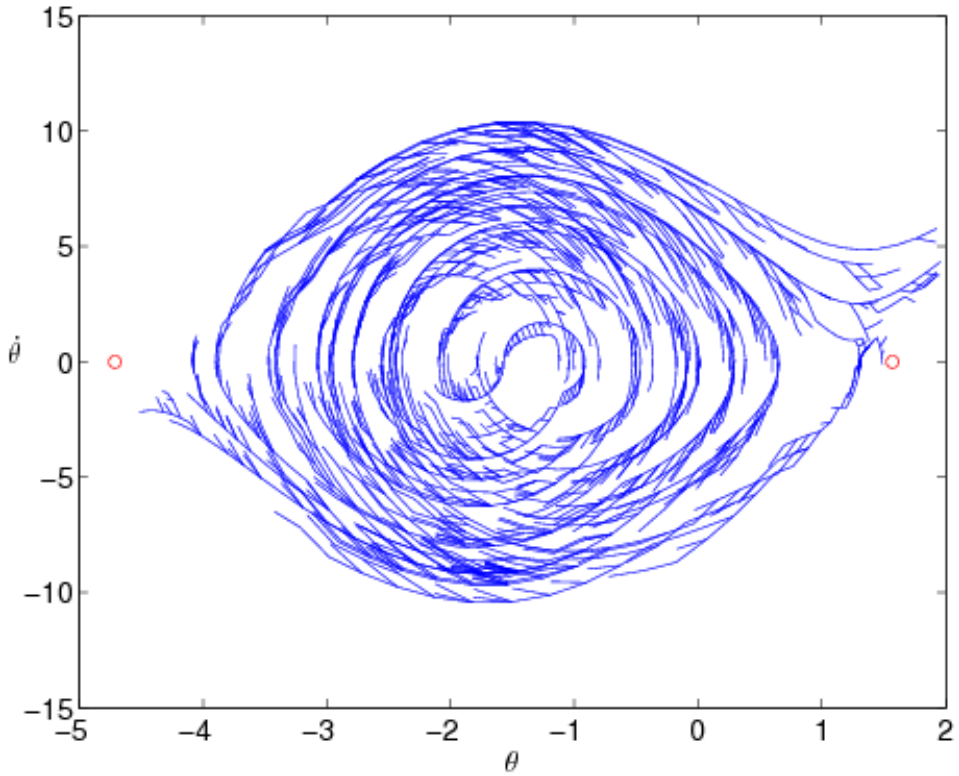


Figure 5-6: The final tree generated with the standard kinodynamic RRT-based planner. The tree consists of over 2300 nodes and required 75 seconds to generate. ©2009 IEEE [Shkolnik et al., 2009].

5.3.1 Swing Up Control for an Underactuated Pendulum

The RG-RRT was utilized to solve for a swing up controller for the underactuated pendulum. The control authority at the shoulder was limited in magnitude. The system parameters were set to a mass of $m = 1$, a length of $l = 0.5$, a damping coefficient of $b = 0.1$, gravitational constant of 9.8, and a maximum torque of $|u| \leq 0.1$.

Figure 5-5 compares the trees for the standard RRT and the RG-RRT after 360 nodes have been expanded. At this point, the RG-RRT algorithm has identified a path to the goal that is consistent with the differential constraints. In contrast, the planner based upon the standard RRT has yet to explore much of the state space. Instead, one can see the aforementioned sensitivity to the Euclidean metric, which has resulted in the expansion of many major (outer) nodes for which the torque limits allow the tree to only grow inwards. This is evident in the large number of overlapping branches that extend inward rather than extending towards the larger Voronoi regions. Referring back to the Voronoi diagrams in Figure 5-4(a), note that the only valid Voronoi regions for these nodes would be on the inside of the tree where sampling is consistent with the ability to grow the tree inwards.

The RG-RRT was able to solve for a swing up controller for the pendulum after expanding the tree 360 times. The overall search took 3 seconds in MATLAB. Meanwhile, the standard RRT-based planner converged to the goal after adding 2300 nodes to the tree, a process that required 75 seconds to complete. All simulations in this work were run on a 3ghz Intel Xeon X5450 CPU. Figure 5-6 presents the final tree generated by the standard RRT.

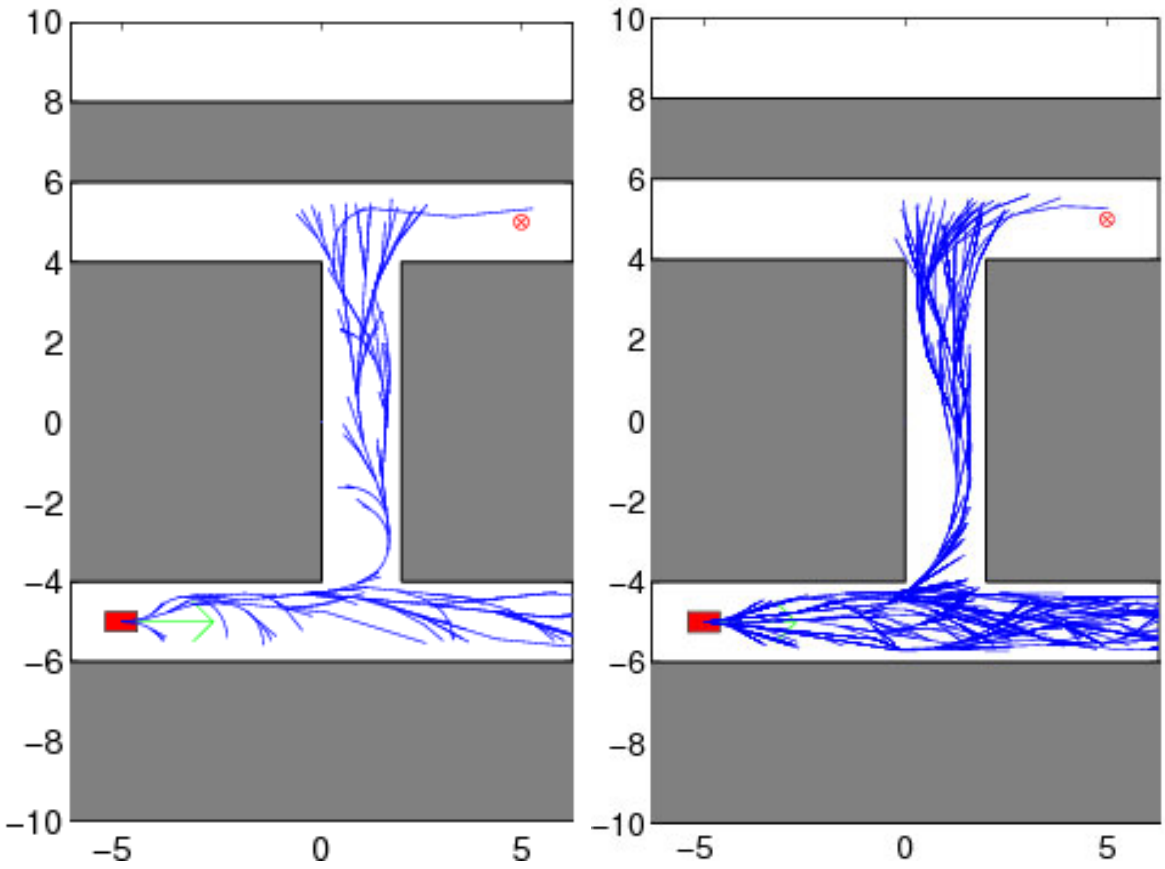
While it's not obvious from the tree, the RG-RRT results in an approximately bang-bang trajectory for swing up control. This behavior is more evident in Figure 5-3(a) where each node corresponds to one of the extreme points, i.e. $|u| = u_{\max}$, that bound the reachable set for its parent node. This control policy agrees with the bang-bang control strategies that have been proposed elsewhere for the inverted pendulum [Furuta et al., 1991].

5.3.2 Motion Planning for the Acrobot

For comparison in a similar problem with a higher dimensional search space, a modified bidirectional version of the RG-RRT algorithm was applied on a simulated torque-limited Acrobot [Spong, 1994b], which is a two link pendulum with an actuator at the elbow, and a passive joint at the base. In the simulation, each link weighed 2kg and was .5m in length with the COM in the center of the link. The reachable set was approximated by discretizing actions into three possible torques values. A standard bidirectional RRT was also run for comparison using the same code base. The results indicate that the standard RRT worked fastest when the control was sampled from a uniform distribution between the torque limits. The bi-directional RG-RRT was run 10 times and took an average of 38 seconds to run, compared to 112 seconds for the standard RRT.

5.3.3 Motion Planning for a Simple Nonholonomic Car

Another class of problem considered is a nonholonomic car navigating through a set of corridors bounded by obstacles as shown in Figure 5-7. The state of the vehicle is represented by its position, heading, and forward velocity, i.e. (x, y, θ, v). The control inputs to the system consist of the angular rate and the forward acceleration, $u = \left[\dot{\theta},\ \dot{v}\right]$, both of which are bounded in magnitude. For our simulation, the vehicle is restricted to forward motion. The task is to find a collision-free trajectory that brings the rectangular-shaped vehicle through the narrow corridors from the lower-left to the upper-right corners of the environment, both with an Eastward heading. The setup of this problem is challenging because the only feasible



(a) RG-RRT (b) Standard RRT

Figure 5-7: A comparison of the final collision-free trajectories that lead the simple car model from its initial position in the lower left-hand corner of the environment to the goal at the upper right. The final tree for (a) the RG-RRT consists of 292 nodes and was generated in under 3 seconds. In contrast, the tree built with the standard RRT-based planner includes 1350 nodes and required 43,000 expansion attempts and 73 seconds to compute. ©2009 IEEE [Shkolnik et al., 2009].

paths require the vehicle to substantially slow down and make a wide turn in order to avoid collision around the tight turns.

One can visualize the state of the system in two dimensions where the location of a point denotes the vehicle's (x, y) position. The direction of a vector extending from the point represents the heading, θ, while the length of the vector denotes the forward velocity, v. In this space, the reachable set, $\mathcal{R}$, is bound by a quadrilateral positioned in front (per the orientation vector) of the point in space where the four corners correspond to the four pairs of saturated control inputs. Any point within this quadrilateral is reachable in time Δt according to the differential constraints.

Both the RG-RRT as well as the standard RRT planner were applied to solve for a sequence of control inputs that drive the vehicle to the goal pose while satisfying the differential constraints and avoiding obstacles. Both algorithms utilized the Euclidean distance metric, and shared much of the same code base. Figure 5-7 compares examples of trees resulting from both algorithms. Each was run a total of 20 times, with the same start and goal poses and obstacle configuration as shown in the figure. On average, the RG-RRT found a path in 2.3 seconds, with a mean of 405 nodes in the tree, and a mean total of 2150 integrations. Five integrations were performed for each expansion, including four required to generate a reachable set for each new node. On the other hand, the standard RRT required 51 seconds on average, with a mean of 1700 nodes in the tree, and 35,000 integrations required. The large number of integrations resulted from a substantial proportion of expansion attempts that failed due to collision.

5.4 Discussion

5.4.1 On Completeness

If the reachable set is exactly known, or if the distance metric exactly measures the distance to the reachable set (perhaps it is not necessary to exactly compute the reachable set itself to do this), then the resulting RG-RRT will be probabilistically complete. Completeness may not hold for some approximations of the reachable set. However, it is trivial to guarantee completeness by occasionally making a standard RRT expansion step by ignoring the reachable set when selecting a node nearest to a sample.

The notion of probabilistic completeness is inherently a weak measure. In practice, it is not possible to run an algorithm for infinite time; to be useful, the RRT must find a plan within a reasonable time frame, whatever that happens to be. Therefore, an algorithm which has no completeness guarantees, but which has empirically shown to perform well (for example by solving a planning problem n times out of n attempts, and did so within a short time frame), is more satisfying than a probabilistically complete algorithm which takes a very long time to converge.

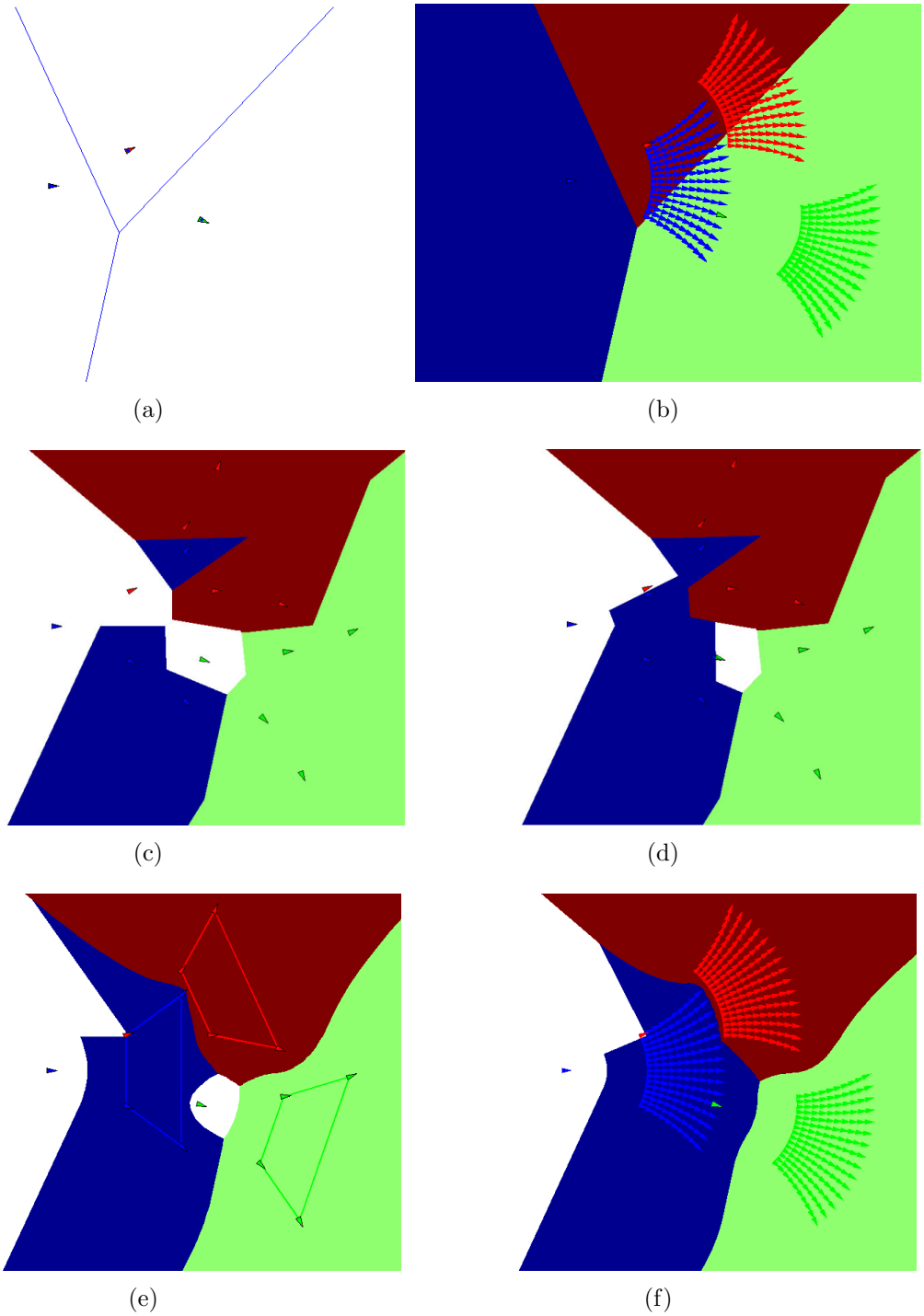


Figure 5-8: Comparison of Voronoi regions when approximating the Reachable Set. (a): 3 nodes of the tree; (b): Reachable set, R, and Voronoi mapping of T; (c) - (e): discrete approximations of R, and corresponding Voronoi mappings; (f): Voronoi mapping of actual R set

5.4.2 Approximating the Reachable Set

A complete description of a continuous space reachable set can be difficult to determine, and the distances between samples and these spaces can be time consuming to compute. In this chapter, I showed that simple discretization of the action space can, at least in some cases, be an effective approach for approximating the reachable set to speed up RRT exploration. To explore the effect of approximating a reachable set, consider for example a kinematic car problem, where each time the car moves, it is limited to a bounded distance, and with some bounded turning radius. An example of a kinematic car tree with three nodes is shown in Figure 5-8(a), together with the Voronoi diagram found using the Euclidean distance metric on the Cartesian coordinate (ignoring rotation of car body). The next image, 5-8(b) shows a finely discretized reachable set for each of the three nodes. This image shows color-coded Voronoi regions, where any sample in those regions are mapped to the corresponding colored node in the tree. It is clear that the Euclidean distance to the three nodes in this tree does not produce a useful Voronoi bias that can be taken advantage of by an RRT planner. Consider that to explore additional actions of the first (blue) node, a sample would have to fall behind (to the left of) the blue node, which does not make much sense for this system.

A Voronoi mapping for the true reachable set of this system is shown in 5-8(f). White regions correspond to regions where samples are discarded. Thus, any samples behind the blue node would be thrown away. However, the blue node could be expanded if a sample lands to the front or right side of this node. The remaining images in this figure demonstrate reasonable approximations of the reachable set. In 5-8(c), the reachable set is approximated using only the four corners of the set. One can see that the resulting Voronoi mapping is fairly similar to the mapping found if using the actual reachable set, with the exception that regions near currently explored nodes (the red and green nodes of the original tree) are white, indicating that samples are discarded. The next image in 5-8(d) is similar, except that when a node is added to the tree, a new node is added to the parent's reachable set approximation, which is moved toward the "inside" of the reachable set from the position of the new node. The result is less white space corresponding to regions where samples are discarded. Lastly, in 5-8(e), the reachable set is approximated by a quadrilateral defined by the four corners of the reachable set. This approximation produces a Voronoi mapping which is very similar to that of the actual reachable set.

As the control dimension increases, it becomes increasingly inefficient to approximate the reachable set using discrete sampling, where the number of samples grows exponentially on dimension. Several recent works, e.g. [Branicky et al., 2008, Green and Kelly, 2007], have examined the problem of reducing the set of discrete trajectories for motion planning by pre-computing control actions that maximize path dispersion and minimize probability of collision, given assumptions or a prior on the probability of encountering obstacles.

In practice, heuristics that approximate the reachable set, even poorly, can perform significantly better than using Euclidean distance to the tree nodes. Some care should be taken, because a poor approximation can lead to parts of state space being classified as regions where samples are inappropriately discarded, which could lock the RRT search. One method for achieving a low dimensional approximation is by using a task space, a

topic which I further investigate in the next chapter. Other approaches would include flow tube approximations, e.g. [Hofmann and Williams, 2006], geometric heuristics, and machine learning approaches, which is left to future work.

5.4.3 Regions of Inevitable Collision

One benefit of generating the reachable set using discretized actions is that points that comprise the reachable set can also be efficiently tested for collisions before they are added to the Reachable set, in order to reduce the likelihood of trajectories leaving free space as part of the exploration phase. This idea has parallels to quickly (but locally) approximating regions of inevitable collision (RIC), while altering the sampling strategy to avoid expanding towards such regions. The RIC is first mentioned by LaValle in [LaValle et al., 2001], and this idea is expanded upon in [Fraichard et al., 2003,Fraichard and T., 2007], and with heuristics in [Chan et al., 2008]. Reachability guidance helps to avoids such collisions by looking one step into the future. The result is that some nodes will have an empty reachable set, which means these nodes will never be expanded upon, but also these nodes in the tree serve to block nearby sampling in state space, because of the dynamic sampling properties of the RG-RRT, which throws away samples that are closer to the tree rather than the reachable set of the tree.

5.4.4 Hybrid Dynamics

Some systems are modeled as having continuous dynamics most of the time, but upon some event (reaching some surface in state space), there is a switching behavior, or discrete impulse in the dynamics, so that the trajectory jumps to a different part of the state space. Mixing continuous dynamics and discrete dynamics is known as hybrid dynamics, and can be challenging for motion planning algorithms such as the RRT to handle. An example of this might include a simulation of legged robot if collisions are modeled as inelastic events that cause an instantaneous change in velocity (and/or change of coordinates) when a foot touches down. The RG-RRT can handle hybrid dynamics by integrating through the impulse when generating the reachable set. If approximating the reachable set by discretizing the action space, some care should be taken, because when going through an impulse, the reachable set may be divided in two, which means any assumptions of a convex reachable set should be applied now to two separate regions instead of one.

5.4.5 Motion Primitives

Motion primitives are sometimes used in planning, in order to reduce the action space. See for example [Frazzoli et al., 2002,Frazzoli et al., 2005], which provides in depth discussion of motion primitives in a planning setting. As long as there is a way to efficiently approximate the result of using a motion primitive for generating a reachable set, the RG-RRT is amenable to this approach. The next chapter will show an example of an application of motion primitives for tackling higher-dimensional motion planning problems.

5.4.6 Concluding Remarks

The RG-RRT algorithm alleviates much of the sensitivity of randomized sampling for systems with differential constraints to the metric that is employed to expand the tree. Essentially, the algorithm rejects samples, and utilizes the metric only for regions of the state space for which the tree can make progress towards the sample. The result is a modified Voronoi bias that emphasizes nodes that are more likely to promote exploration given the constraints on the system dynamics. The RG-RRT takes advantage of the fact that sampling is cheap compared to the process of expanding a node and, therefore, is willing to re-sample until a point is drawn from a region that yields better exploration.

The size distribution for the modified Voronoi regions under the RG-RRT varies as the tree explores the state space. Initially, there is a large variability in the size of the different Voronoi regions but, as the tree expands, the size tends to become more uniform. The lower bound on the size of the regions, typically corresponding to the inside of the tree, is inversely proportional to the resolution of the reachable set. In turn, the sampling of the space will be probabilistically complete so long as the resolution of the reachable set is sufficient relative to the smallest "gap" in the state space.

Overall, the algorithm presented is a simple way of eliminating metric and sampling specificity in the RRT algorithm implementation, which is particularly useful when there are differential constraints, or when reachable sets are otherwise non-spherical because of motion primitives or hybrid dynamics. The algorithm was demonstrated to be effective for planning on systems such as the pendulum, Acrobot, and a simple model of a car.

Chapter 6

Application: Motion Planning to Achieve Bounding Over Rough Terrain with LittleDog

In the previous chapter, reachability guidance was shown to be effective for dynamically biasing sampling to improve RRT search efficiency for dynamic systems. However, discretizing the control space does not scale well with dimension. In this chapter, we build on the sampling ideas developed in this thesis, particularly reachability guidance and task-space biasing. These approaches are combined together with a motion primitive in an RRT based planning framework that quickly finds feasible motion plans for bounding over rough terrain with the LittleDog robot.

Significant portions of this chapter also appear in [Shkolnik et al., 2010].

6.1 Bounding on Rough Terrain

While many successful approaches to dynamic legged locomotion exist, we do not yet have approaches which are general and flexible enough to cope with the incredible variety of terrain traversed by animals. Progress in motion planning algorithms has enabled general and flexible solutions for slowly moving robots, but in order to quickly and efficiently traverse very difficult terrain, extending these algorithms to dynamic gaits is essential.

This chapter describes a motion planning strategy to achieve dynamic locomotion over rough terrain with the LittleDog robot. Recall that LittleDog (Figure 6-1) is a small, 3kg position-controlled quadruped robot with point feet and was developed by Boston Dynamics under the DARPA Learning Locomotion program. The program ran over several phases from 2006 to 2009, and challenged six teams from universities around the United States to compete in developing algorithms that enable LittleDog to navigate known, uneven terrain, as quickly as possible. The program was very successful, with many teams demonstrating robust planning and locomotion over quite challenging terrain (e.g., [Rebula et al., 2007, Kolter et al., 2008, Pongas et al., 2007, Zucker, 2009]), with an emphasis on walking gaits, and some short stereotyped dynamic maneuvers that relied on an intermittent existence of

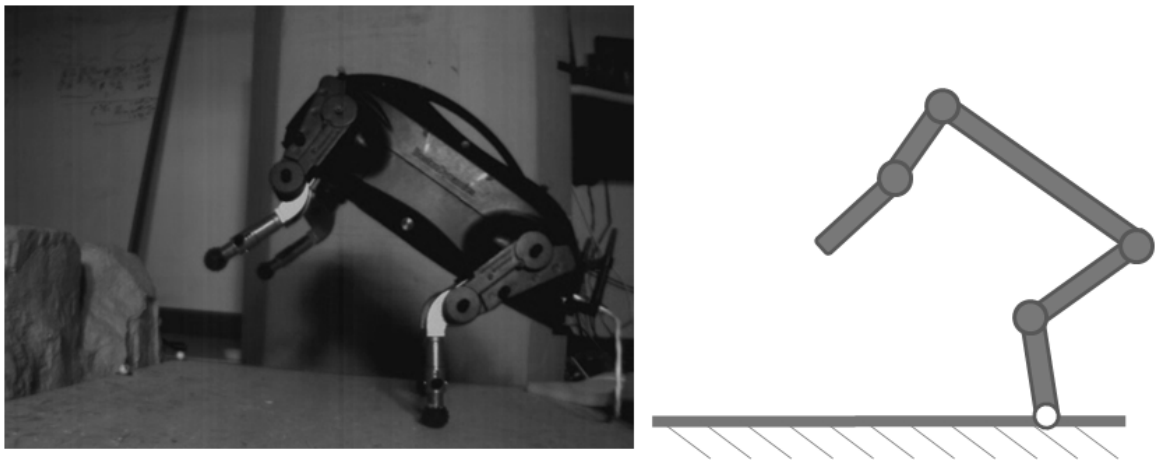


Figure 6-1: LittleDog robot, and a corresponding 5-link planar model (blue circles represent actuated joints, white circles represent passive joints)

a support polygon to regain control and simplify planning [Byl et al., 2008]. In this work, I present a method for generating a continuous dynamic bounding gait over rough terrain.

Achieving bounding on LittleDog is difficult for a number of reasons. First of all, the robot is mechanically limited - high-gear ratio transmissions generally provide sufficient torque but cause severe limits on joint velocities and complicate any attempts at direct torque control. Furthermore, a stiff frame complicates impact modeling and provides essentially no opportunity for energy storage. Finally, the robot is underactuated, with the dynamics of the unactuated joints resulting in a complicated dynamical relationship between the actuated joints and the interactions with the ground. These effects are dominant enough that they must be considered during the planning phase.

6.2 LittleDog Model

An essential component of any model-based planning approach is a sufficiently accurate identification of the system dynamics. Obtaining an accurate dynamic model for LittleDog is challenging due to subtleties in the ground interactions and the dominant effects of motor saturations and transmission dynamics. These effects are more pronounced in bounding gaits than in walking gaits, due to the increased magnitude of ground reaction forces at impact and the perpetual saturations of the motor; as a result, we required a more detailed model. In this section, I briefly describe the model of the LittleDog robot, which was largely developed and identified by Michael Levashov. A more comprehensive description of the model and system identification can be found in [Shkolnik et al., 2010].

The LittleDog robot has 12 actuators (two in each hip, one in each knee) and a total of 22 essential degrees of freedom (six for the body, three rotational joints in each leg, and one prismatic spring in each leg). By assuming that the leg springs are over-damped, yielding first-order dynamics, we arrive at a 40 dimensional state space ($18 \times 2 + 4$). However, to

Figure 6-2: Dog bounding up stairs.
Images from video at http://www.flickr.com/photos/istolethetv/3321159490/

keep the model as simple (low-dimensional) as possible, we approximate the dynamics of the robot using a planar 5-link serial rigid-body chain model, with revolute joints connecting the links, and a free base joint, as shown in Figure 6-3. The planar model assumes that the back legs move together as one and the front legs move together as one (see Figure 6-1). Each leg has a single hip joint, connecting the leg to the main body, and a knee joint. The foot of the real robot is a rubber-coated ball that connects to the shin through a small spring, which is constrained to move along the axis of the shin. The spring is stiff, heavily damped, and has a limited travel range, so it is not considered when computing the kinematics of the robot, but is important for computing the ground forces. In addition, to reduce the state space, only the length of the shin spring is considered.

The model's 7-dimensional configuration space, $\mathcal{C} = \mathbb{R}^2 \times \mathbb{T}^5$, consists of the planar position of the back foot (x, y), the pitch angle ω, and the 4 actuated joint angles $q_1, ..., q_4$. The full state of the robot, $x = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{l}] \in \mathcal{X}$, has 16 dimensions and consists of the robot configuration, the corresponding velocities, and the two prismatic shin-spring lengths, $\mathbf{l} = [l_1, l_2]$, one for each foot. The control command, $\mathbf{u}$, specifies reference angles for the 4 actuated joints. The robot receives joint commands at 100 Hz and then applies an internal

PD controller at 1KHz. For simulation, planning and control purposes, the dynamics are defined as

$$\mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n]), \tag{6.1}$$

where $\mathbf{x}[n+1]$ is the state at $t[n+1]$, $\mathbf{x}[n]$ is the state at $t[n]$, and $\mathbf{u}[n]$ is the actuated joint position command applied during the time interval between $t[n]$ and $t[n+1]$. We will sometimes refer to the control time step, $\Delta T = t[n+1] - t[n] = 0.01$ seconds. A fixed-step 4th order Runge-Kutta integration of the continuous Euler-Lagrange dynamics model is used to compute the state update.

A self-contained motor model is used to describe the movement of the actuated joints. Motions of these joints are prescribed in the 5-link system, so that as the dynamics are integrated forward, joint torques are back-computed, and the joint trajectory specified by the model is exactly followed. This model is also constrained so that actuated joints respect bounds placed on angle limits, actuator velocity limits, and actuator torque limits. In addition, forces computed from a ground contact model are applied to the 5-link chain when the feet are in contact with the ground. The actuated joints are relatively stiff, so the model is most important for predicting the motion of the unactuated degrees of freedom of the system, in particular the pitch angle, as well as the horizontal position of the robot.

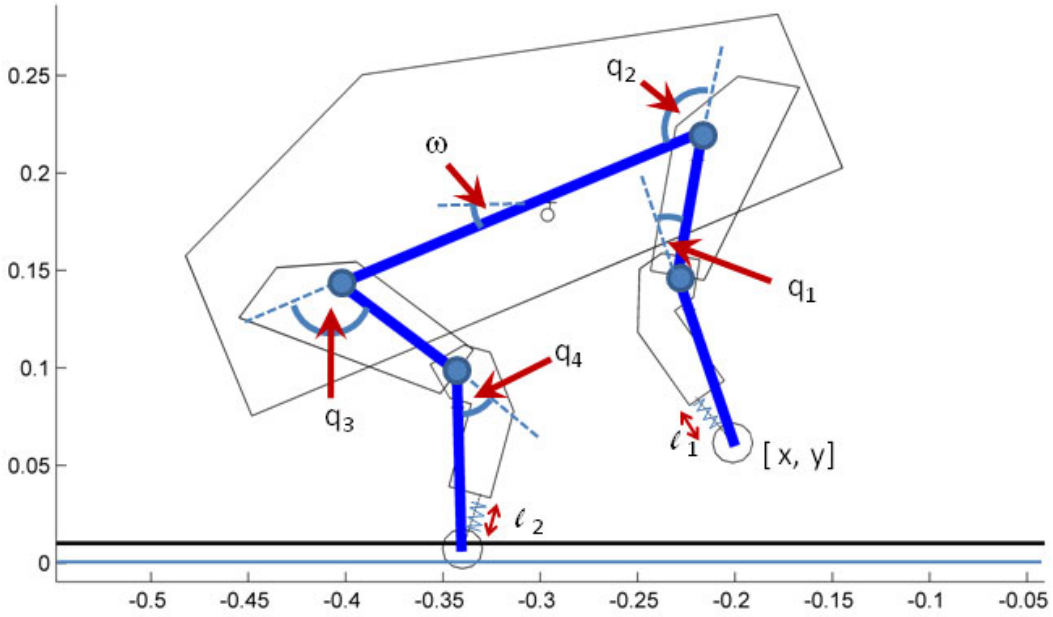


Figure 6-3: LittleDog model. The state space is $\mathcal{X} = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{l}]$, where $q = [x, y, \omega, q_1, q_2, q_3, q_4]$, and $\mathbf{l} = [l_1, l_2]$ are feet spring lengths used in the ground contact model. The diagram also illustrates the geometric shape of the limbs and body, information used for collision detection during planning.

6.3 Motion Planning Algorithm

6.3.1 Problem Formulation

Given the model described in the previous section, we now formulate the problem of finding a feasible trajectory from an initial condition of the robot to a goal region defined by a desired location of the center of mass. We describe the terrain as a simple height function, $z = \gamma(x)$, parameterized by the horizontal position, x. We would like the planned trajectory to avoid disruptive contacts with the rough terrain, however the notion of "collision-free" trajectories must be treated carefully since legged locomotion requires contact with the ground in order to make forward progress.

To address this, we define a virtual obstacle function, $\Gamma(x)$, which is safely below the terrain around candidate foothold regions, and safely above the ground in regions where we do not allow foot contact (cartooned in Figure 6-4). In our previous experience with planning walking gaits [Byl et al., 2008, Byl and Tedrake, 2009a], it was clear that challenging rough terrain could be separated into regions with useful candidate footholds, opposed to regions where footholds that would be more likely to cause a failure. Therefore, we had developed algorithms to pre-process the terrain to identify these candidate foothold regions based on some simple heuristics, and we could potentially use the same algorithms here to construct $\Gamma(x)$. However in the current work, which makes heavy use of the motion primitive described in the following sections, we found it helpful to construct separate virtual obstacles, $\Gamma_m(x)$, parameterized by the motion primitive, m, being performed. Once the virtual obstacles became motion primitive dependent, we had success with simple stereotyped virtual obstacles as illustrated in Figure 6-4. The collision function illustrated is defined relative to the current position of the feet. In the case shown in the figure, the virtual function forces the swing leg to lift up and over the terrain, and ensures that the back foot does not slip, which characterizes a successful portion of a bound. As soon as the front feet touch back down to the ground after completing this part of the bound, a new collision function is defined, which takes into account the new footholds, and forces the back feet to make forward progress in the air.

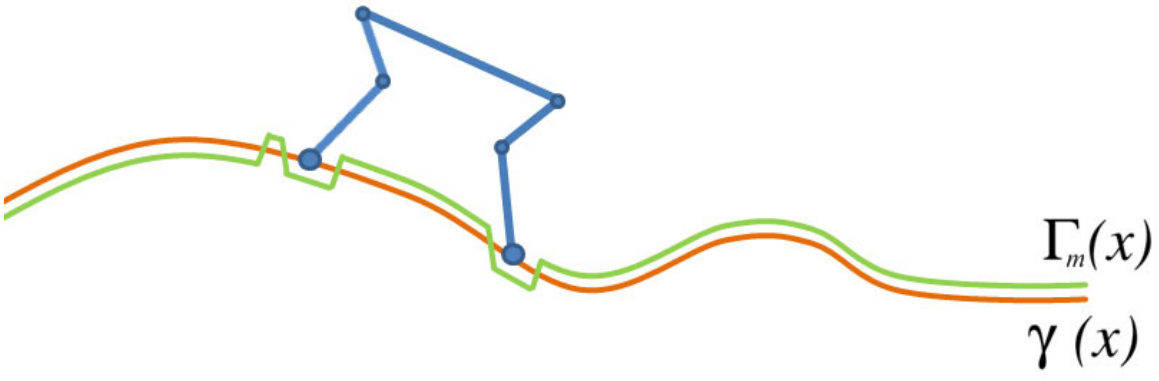


Figure 6-4: Sketch of a virtual obstacle function, $\Gamma_m(x)$ in relation to the ground, $\gamma(x)$.

We are now ready to formulate the motion planning problem for LittleDog bounding: find a feasible solution, $\{\mathbf{x}[0], \mathbf{u}[0], \mathbf{x}[1], \mathbf{u}[1], ..., \mathbf{x}[N]\}$, which starts in the required initial conditions, satisfies the dynamics of the model, $\mathbf{x}[n+1] = \mathbf{f}(\mathbf{x}[n], \mathbf{u}[n])$, avoids collisions with the virtual obstacles, $\Gamma(\mathbf{x})$, and doesn't violate the bounds on joint positions, velocities, accelerations, and torques.

Given this problem formulation, it is natural to consider a sample-based motion planning algorithm such as Rapidly-Exploring Random Trees (RRTs), due to their success in high-dimensional robotic planning problems involving complex geometric constraints [LaValle and Branicky, 2002]. However, as discussed throughout this thesis, these algorithms perform poorly when planning in state space (where the dynamics impose "differential constraints") [LaValle, 2006,Cheng, 2005], especially in high dimensions. When applied directly to building a forward tree for this problem, the standard RRT takes a prohibitive amount of time and fails to make substantial progress towards the goal. The Reachability-Guided RRT, presented in the previous chapter, uses rejection sampling to bias the sampling towards regions in which the tree is able to grow, while taking into account the constraints imposed by the dynamics. This sampling strategy can vastly improve planning efficiency of the RRT for problems with challenging dynamics. However, approximating the reachable set by evenly discretizing the action space on a higher dimensional problem such as LittleDog would be prohibitive. In the following subsections, we describe two additional modifications to the RG-RRT algorithm. First, we describe a parameterized "half-bound" motion primitive which reduces the dimensionality of the problem. Then, we describe a mechanism for sampling with a Voronoi bias in the lower-dimensional task space defined by the motion primitive. Using the RG-RRT algorithm, together with these additional modifications, produces a motion planner that can find bounding trajectories in a reasonable amount of time.

6.3.2 Macro-Actions / Motion-Primitive

The choice of action space, e.g. how an action is defined for the RRT implementation, will affect both the RRT search efficiency, as well as completeness guarantees, and, perhaps as importantly, path quality. In the case of planning motions for a 5-link planar arm with 4 actuators, a typical approach may be to consider applying a constant torque (or some other simple action in joint space) that is applied for a short constant time duration, ΔT. One drawback of this method is that the resulting trajectory found by the RRT is likely be jerky. A smoothing / optimization post-processing step may be performed, but this may require significant processing time, and there is no guarantee that the local minima near the original trajectory is sufficiently smooth. Another drawback of using a constant time step with such an action space is that in order to ensure completeness, ΔT should be relatively small (for LittleDog bounding, .1 seconds seems to be appropriate). Empirically, however, the search time increases approximately as $1/\Delta T$, so this is a painful trade-off.

For a stiff PD-controlled robot, such as LittleDog, it makes sense to have the action space correspond directly to position commands. To do this, we generate a joint trajectory by using a smooth function, $\mathcal{G}$, that is parameterized by the initial joint positions and velocities, $[\mathbf{q}(0), \dot{\mathbf{q}}(0)]$, a time for the motion, ΔT_m, and the desired end joint positions and

velocities, $[\mathbf{q}_d(\Delta T_m), \dot{\mathbf{q}}_d(\Delta T_m)]$. This action set requires specifying two numbers for each actuated DOF: one for the desired end position and one for the desired end velocity. A smooth function generator which obeys the end point constraints, for example a cubic-spline interpolation, produces a trajectory which can be sampled and sent to the PD controller.

If one considers bounding in particular and examines how animals, and even some robots such as the Raibert hoppers, are able to achieve bounding behavior, it becomes apparent that some simplifications can be made to the action space. We define a motion primitive that uses a much longer ΔT and, therefore, a shorter search time, while also producing smooth,

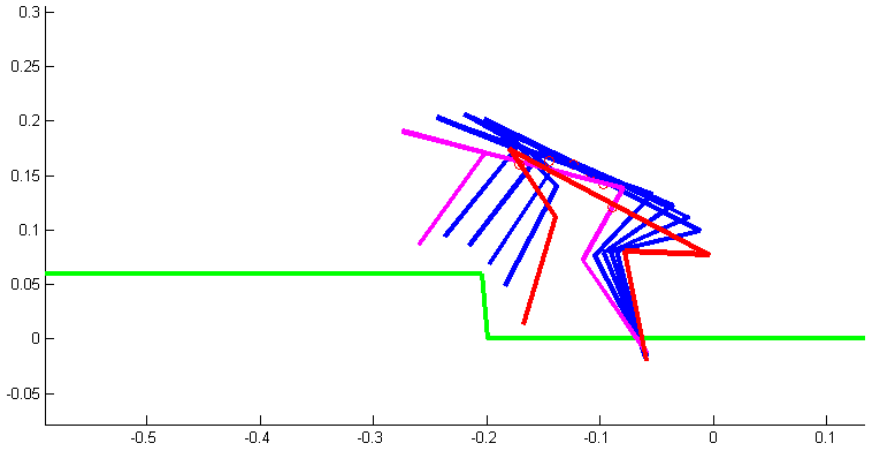


Figure 6-5: First half of a double bound. Initial pose shown in red, final pose shown in magenta. Axes are in meters.

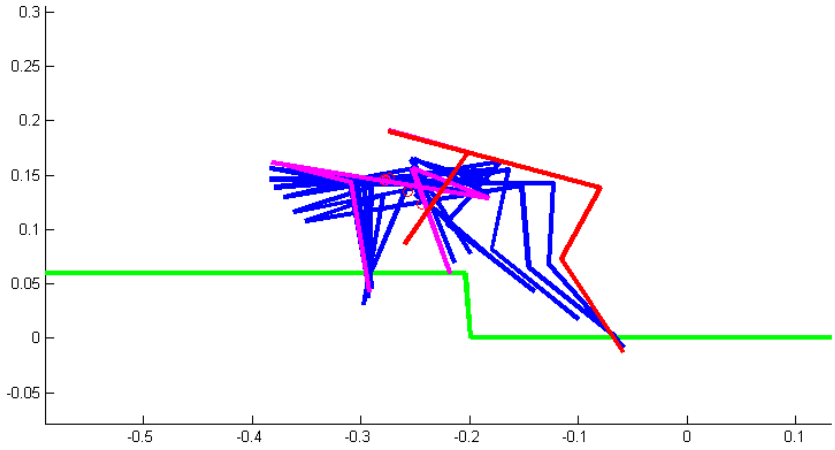


Figure 6-6: Second half of a double bound. Initial pose shown in red, final pose shown in magenta. Notice that in both of these figures, the swing foot "tucked in" in order to clear the step.

jerk-free joint trajectories. The insight is based on the observation that a bound consists of two phases: (1) rocking up on the hind legs while moving the front legs forward and (2) rocking up on the front legs, while the hind legs move forward. In the first motion primitive, which addresses phase-1, the hind legs begin moving first, and the motion is monotonically "opening" the hip angles. This produces a forward acceleration of the COM, which also generates a rotational moment around the pseudo-pin joint at the hind foot-ground interface. In this case, the front legs come off the ground, and they are free to move to a position as desired for landing. In this formulation, the hind legs move directly from the starting pose to the ending pose in a straight line. Because the front feet are not weight bearing, it is useful to "tuck" the feet, while moving them from the start to the end pose, in order to help avoid obstacles. To take advantage of a rotational moment produced by accelerating the stance leg, the back leg begins moving a short time before the front foot leg starts moving. Once both the hind and the front legs have reached their desired landing poses, the remaining trajectory is held constant until the front legs impact the ground. Examples of such a trajectory are shown in Figure 6-5 and the top image in Figure 6-8.

A similar approach is utilized to generate motions for the second phase of bounding, with the difference that the hip angles are "contracting" instead of "opening up". As the front leg becomes the stance leg, it begins moving just before impact with the ground. The back leg movement is delayed for a short period to allow the system to start rocking forward on to the front legs. When both legs reach their final positions, the pose is held until the the back leg touches down. The resulting motions are shown in Figure 6-6 and the second image in Figure 6-8.

Note that the start and end joint velocities in this motion primitive are always zero, a factor which reduces the action space further. Using these motion primitives requires a variable time step, ΔT_{bound}, because this directly influences accelerations and, therefore, moments around the passive joints. However, for each phase, one only needs to specify 4-DOF, corresponding to the pose of the system at the end of the phase.

The motion primitive framework used here is somewhat similar in approach to the work in [Frazzoli et al., 2005]. In addition to formally defining motion primitives in the context of motion planning, that work proposed planning in the framework of a Maneuver Automaton. The automaton's vertices consist of steady-state trajectories, or trim primitives (in the context of helicopter flight), which are time invariant and with a zero-order hold on control inputs. The edges of the automaton are maneuver primitives, which are constrained so that both the start and the end of the maneuver are compatible with the trim primitives. This is illustrated in the context of LittleDog bounding in Figure 6-7. Of course, feet collisions in LittleDog act like impulses or hybrid dynamics, which negate the underlying invariance assumptions of a trim primitive in [Frazzoli et al., 2005] and [Frazzoli et al., 2002], but the idea is still representative.

6.3.3 Approximating the Reachable Set

In the last chapter, I showed that reachability-guided sampling can help deal with issues imposed by dynamic constraints in systems such as the pendulum and Acrobot [Shkolnik

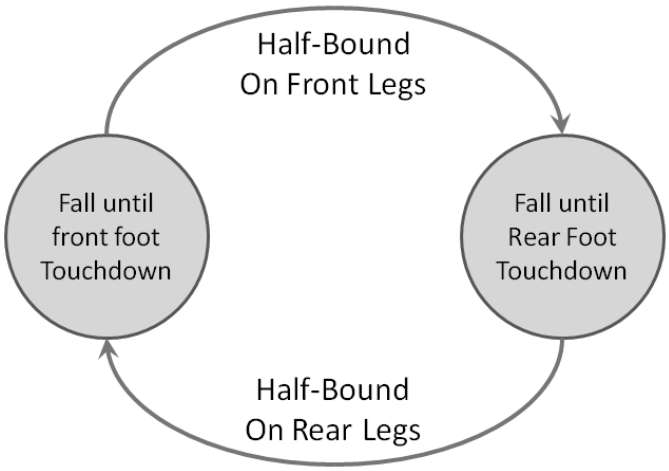


Figure 6-7: Maneuver Automaton for LittleDog Bounding.

et al., 2009]. In those systems, the reachable set was approximated by discretizing the action space. However, for the work described here, even the reduced 4-dimensional action space becomes too large to discretize efficiently. For example, using only 3 actions per dimension (and assuming a constant motion primitive time step), there would be 81 actions to apply and simulate in order to approximate the reachable set for each node.

Instead of discretizing in such a high-dimensional action space, the reachable set can be approximated by understanding the failure modes of bounding. Failures may occur primarily in one of three ways: 1) the robot has too much energy and falls over; 2) the robot has too little energy, so the stance foot never leaves the ground, violating our assumption that one foot always leaves the ground in a bounding gait; or 3) a terrain collision occurs. In the case when the system has a lot of kinetic energy, the best thing to do is to straighten all of the limbs, which raises the center of mass and converts the kinetic energy into potential energy. At the other extreme, if the robot does not have much kinetic energy, an action that lowers the COM while accelerating the limbs inwards tends to produce a rotational moment, if it is possible to do so. Thus, two extreme motions can be generated, for each phase of bounding, which prevent the two common failure modes. The reachable set is then generated by interpolating joint positions between the two extremes. This one-dimensional discretization is usually rich enough to capture the energy related failure modes and generate bounds which strike the right balance to continue bounding further.

As mentioned previously, the reachable set can help the RG-RRT to plan around Regions of Inevitable Collisions (RIC) [LaValle et al., 2001, Fraichard et al., 2003, Fraichard and T., 2007, Chan et al., 2008]. Nodes which have empty reachable sets, which may occur because the node has too much or too little energy, in the case of LittleDog bounding, are in $\mathcal{X}_{RIC}$, even if the node itself is collision free ($\mathcal{X}_{free}$). The RG-RRT takes advantage of this, because when a node has an empty reachable set, the node serves as a "blocking" function for sampling. Because of the rejection sampling of the RG-RRT, such a node can

not be expanded upon, and any samples that map to this node will be discarded, encouraging sampling in other areas that can be expanded.

6.3.4 Sampling in Task Space

In Chapter 2, I showed that the Voronoi Bias can be implemented in a lower dimensional task space [Shkolnik and Tedrake, 2009]. This can be achieved by sampling in task space, and finding the node in the configuration-space tree, the projection of which is closest to the task space sample. Reducing the dimensionality of the search with a task space Voronoi bias can significantly improve search efficiency, and if done carefully, does not impact the completeness of the algorithm.

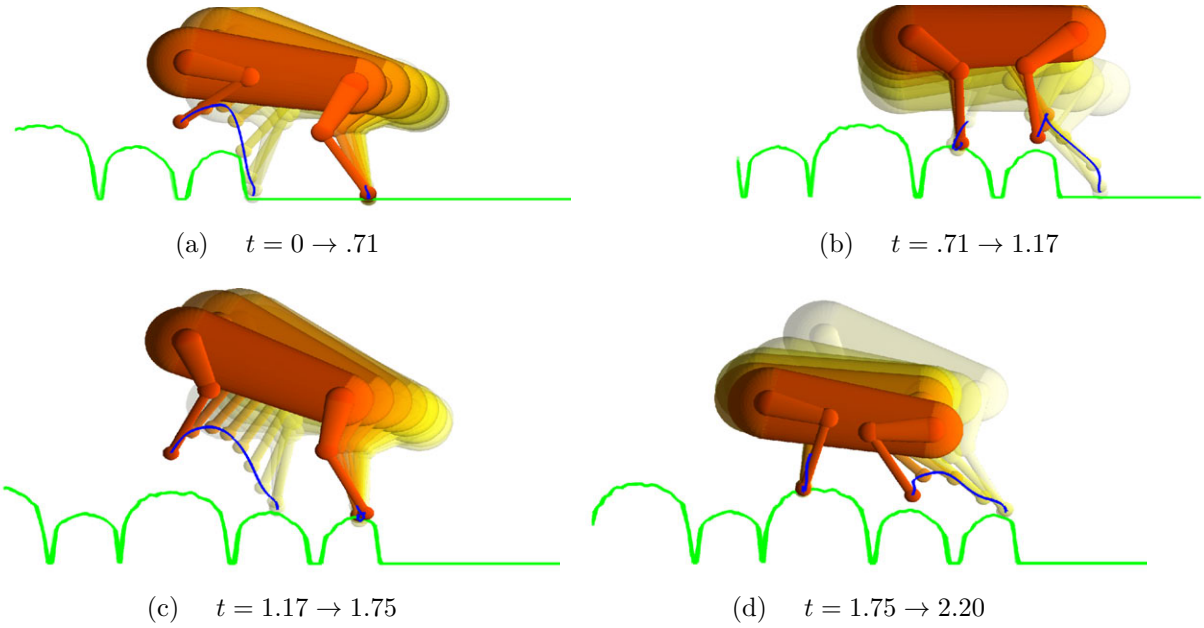
As described in the previous section, our action space involves a half-bound (half a period of a complete bound). At the start and end of an action (e.g. the state at any given node on the tree), the robot is approximately touching the ground with both feet, and joint velocities are approximately zero. Samples are therefore similarly constrained. Additionally, samples are chosen such that they are not in collision, and respect joint bounds, with some minimum and maximum stance width. The region in actuated joint space can be mapped to a region in Cartesian space for the back and front foot, corresponding to a 4-dimensional manifold. A sample is drawn by first choosing a horizontal coordinate, x, of the robot's back foot, and then selecting 4 joint angles from the manifold, while checking to ensure that collision constraints are validated.

Given a sample described above, the y-coordinate of the robot foot is set to the ground position at x, and the passive joint is computed using the constraint that both feet are on the ground. Thus, sampling the 5 dimensional space maps to a point $\mathbf{q}_s$ in the 7 dimensional configuration space of the robot. The 5 dimensional sampling space, which neglects velocities, is significantly smaller than the complete 16 dimensional state space, and produces a task-space Voronoi Bias. When a sample is drawn, the closest node is found by minimizing the Euclidean distance from the sample to the Tree, as well as to the Reachable Region of the Tree. The sampling is repeated until a sample closest to the Reachable Region is found. An action, $\mathbf{u}$, is then created by selecting the 4 actuated joint angles from the sample, $\mathbf{q}_s$. An action time interval ΔT_{bound} is chosen by uniformly sampling from $T \in [.3, .7]$ seconds.

6.3.5 Simulation Results

We have presented three modifications to the standard implementation of the RRT to plan bounding motions with LittleDog: 1) reachability guidance; 2) a simple motion primitive; and 3) task-space biasing. Each of these components could be implemented separately, but they worked particularly well when when combined. To show this, we qualitatively compare results by applying various combinations of these modifications.

First, a standard RRT was implemented, without any modifications, with the task of finding bounding motions on flat terrain. The state-space was sampled uniformly in the regions near states that have already been explored. We experimented with several types of action spaces, including a zero-order hold on accelerations in joint space, or a zero-order



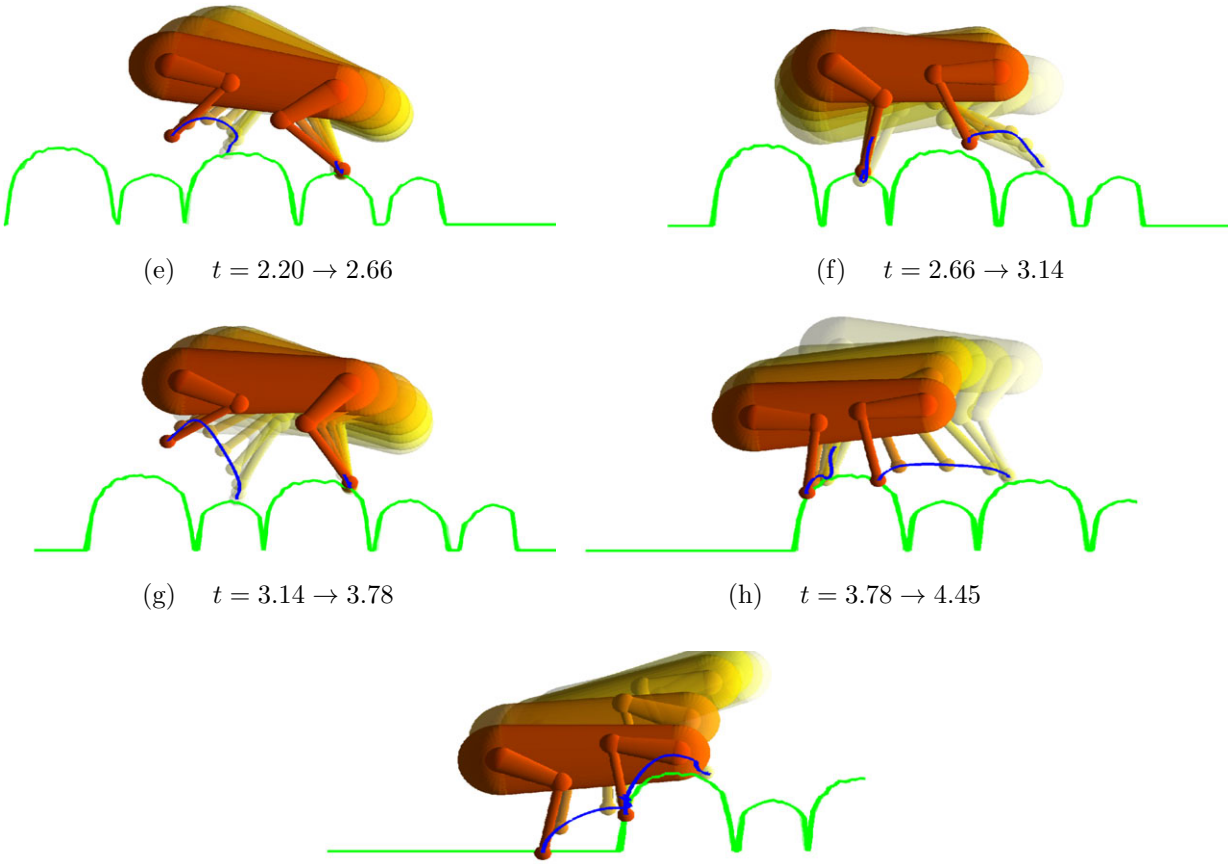
(a) $t = 0 \rightarrow .71$ (b) $t = .71 \rightarrow 1.17$

(c) $t = 1.17 \rightarrow 1.75$ (d) $t = 1.75 \rightarrow 2.20$

Figure 6-8: Bounding trajectory over logs

hold on velocities in joint space. We found the best results by specifying a change in joint positions, (Δq_a), and using a smooth function generator to create position splines with zero start and end velocities. This approach worked best using time intervals of .1 - .15 seconds, but the resulting RRT was not able to find a complete bounding motion plan in a reasonable amount of time (hours). By implementing reachability guidance and task-space sampling, the planner was able to find a double-bound trajectory on flat terrain, after 2-3 minutes of search on average. Some of these plans were successfully executed on flat terrain by the actual robot. In addition to being relatively slow to plan motions, the trajectories returned by the RRT had high-frequency components which were difficult for the robot to execute without saturating motors. The long planning time and sub-optimal gaits were not ideal for this task. Plans might be smoothed and locally optimized, but this is a time-consuming process given the complexity of the dynamics.

When we utilized the half-bound motion primitive, the trajectories produced were smooth by the way the primitive is defined, with one hip movement and one knee movement per each half-bound motion. Furthermore, the half bound has a relatively long duration, typically .5 seconds in length, which shortens search time. The complete planner described in this section is able to plan a double-bound on flat terrain in a few seconds. A continuous bound over two meters is typically found in less than a minute. The complete planner was also able to plan over intermittent terrain, where foot holds were not allowed in given regions. Additionally, the planner was successful in handling a series of 7cm steps, and was also successful in planning bounds to get up on top of a terrain with artificial logs with a maximum height difference of 8cm. The simulated log terrain corresponds to a laser scan of real terrain board used in the Learning Locomotion program to test LittleDog performance. An example of a



(e) $t = 2.20 \rightarrow 2.66$ (f) $t = 2.66 \rightarrow 3.14$

(g) $t = 3.14 \rightarrow 3.78$ (h) $t = 3.78 \rightarrow 4.45$

(i) $t = 4.45 \rightarrow 4.98$

Figure 6-9: Bounding trajectory over logs, continued

bounding trajectory is shown in Figures 6-8 and 6-9. The bottom link in the robot leg is 10cm, and the top link is 7.5cm; given that the bottom of the body is 3cm below the hip, variations of 7cm in terrain height represents approximately 50% of maximum leg travel of the robot.

Planning was also attempted using the motion primitive, but without reachability guidance. To do so, samples were drawn in each iteration of the RRT algorithm, and the nearest neighbor function returned the closest node in the tree. Because the closest node in the tree and the sample often looked similar, it did not make sense to try to expand towards the sample. Instead, once a sample was chosen, a motion primitive action was chosen at random. Using random actions is an accepted approach for RRT's, and the sampling itself produces a Voronoi Bias to encourage the tree to expand into unexplored regions of space, on average. In this case, however, the motion primitive based RRT was never able to find feasible plans over rough terrain without reachability-guidance, even when given over 12 hours to do so. The combination of motion primitives, task-space biasing and reachability-guidance, however, is able to plan trajectories over the log terrain repeatedly, within 10-15

minutes of planning time required on average. Of course, on simpler terrain, the planning time is much faster.

6.3.6 Experimental Results

In experiments with the real robot, open-loop execution of the motion plan found by the RRT quickly diverged with time from the model predictions, even on flat terrain. Trajectories are unstable in the sense that given the same initial conditions on the robot and a tape of position commands to execute, the robot trajectories often diverge, sometimes catastrophically. To demonstrate the feasibility of using the motion primitives described in this paper, these motion primitives were used to achieve a short bound sequence over the log terrain. Unlike in the model, the actual logs are not planar. To address this, tracks were laid on the logs corresponding to the stance width of the robot along the terrain. The planner was constrained to only allow foot holds where the points on adjacent tracks were of the same height. With some tuning, the motion primitives produced bounding over the logs on the real robot, shown in Figure 6-10. This trajectory was successful approximately 20% of the time, even though care was taken to have the same initial position for each trial.

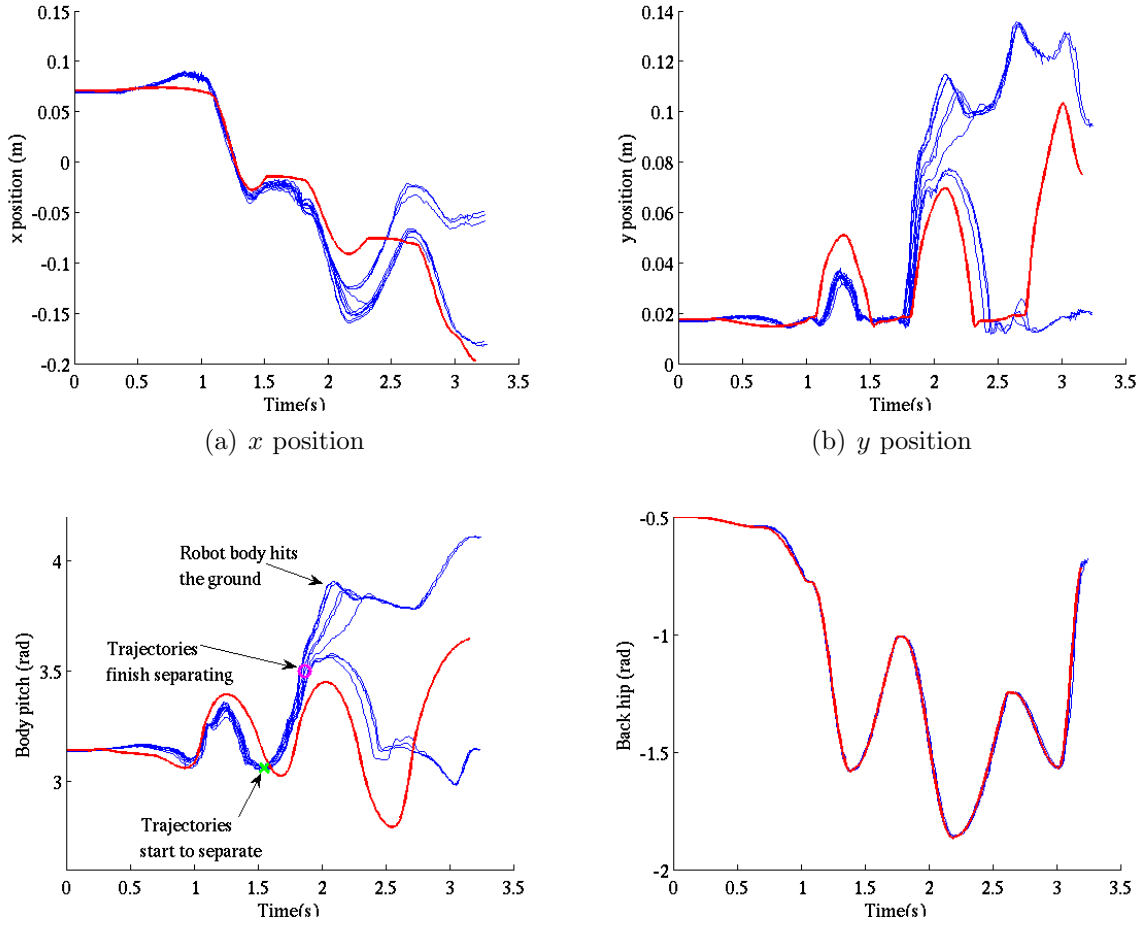
Trajectory optimization and feedback stabilization will be required to enable the execution of the complete trajectory. Feedback will help to compensate for model errors and the inherent instability of the system itself, as demonstrated by our experiments on the actual robot (see Figure 6-11). Care was taken to ensure that the physics model is smooth and continuous, so gradients can be used to help in these tasks. To implement reliable bounding over rough terrain in experiment, sensor-based feedback will be essential. Figure 6-12 shows the sensing and control environment for LittleDog, and also illustrates the additional complexity introduced by various delays and the variety of potentially noisy sensors in the system.

6.4 Concluding Remarks

This chapter described a motion planner for a simulation of the LittleDog quadruped robot to achieve bounding on rough terrain. The robot has a 40 dimensional state space, which was reduced to a 16 dimensional state space in a planar model. A physics based model was developed and identified using data from the real robot. An efficient RRT based planner was implemented. This planner used motion primitives to reduce the size of the action space. Corresponding to the idea of task-space guided RRT search, sampling was done in a 5-dimensional subset of the state space. The reduction in sampling dimension ignores pitch, vertical position, and velocities which significantly improves search efficiency. To handle challenging dynamic constraints associated with bounding, Reachability Guidance was used, so that random samples were chosen such that they were closer to reachable regions of the tree than the tree itself. This ensures that the expansion step of the RRT algorithm can make progress in the direction of the sample, which vastly improves RRT performance. The RRT motion planner was demonstrated to achieve bounding over steps

Figure 6-10: Bounding over logs with LittleDog

and over logs, with terrain height differences corresponding to approximately 50% of the leg length. Without Reachability Guidance, 12 hours was not sufficient to plan in the otherwise identical conditions. Using task-space biasing and Reachability Guidance, the planner returned feasible trajectories within minutes. A primary reason for this is that a standard RRT implementation does not look ahead. Many of the nodes on the outer edges of the tree – the exact nodes which the Voronoi Bias selects most often, because they are



(a) x position

(b) y position

(c) Body pitch

(d) Back hip joint position

Figure 6-11: LittleDog Model compared to Actual Trajectories for a bounding motion on flat terrain: Unactuated coordinates: x (top left), y (top right), and body pitch (bottom left); trajectory of one of the joints (bottom right). The thick red line shows a trajectory generated by planning with RRT's using the simulation model. The blue lines are 10 open-loop runs of the same trajectory on a real LittleDog robot. It is clear that the model captures the actuated dynamics very well. The unactuated dynamics are highly nonlinear, but the model captures much of the system behavior. The response of the robot itself is unstable, even from very similar starting conditions. Figure by Michael Levashov.

near the unexplored regions of state space – correspond to nodes that are either incapable of bounding on the next step because not enough energy is carried over, or have too much energy and no possible action could be applied on the next step to keep the robot from falling over. Expanding on these nodes is futile, but the standard RRT will try to expand them repeatedly causing the algorithm to fail to find a plan.

The motion primitives used for the motion planner were tested on the robot, and shown to be capable of bounding on rough terrain, including the logs terrain board that we were able to plan on. The trajectories implemented on the robot using the motion primitives are smooth, and our model captures the resulting dynamics quite well for short trajectories. However, the forward simulation and the actual behavior of the robot tended to diverge after about 1 second, despite having a relatively elaborate physics model of the robot that was identified on data collected from the robot. This is not entirely surprising, since open-loop bounding trajectories executed on the robot tend to diverge in a similar time frame.

Current work by the MIT LittleDog team is focused on constructing a feedback controller to stabilize the bounding motion plans. As an alternative approach, along the lines of LQR-Trees [Tedrake, 2009a], the motion plan returned may be combined with feedback control, for example based on TV-LQR, which would have some measurable basin of attraction. The plan could then be optimized by maximizing the overall size of the basin of attraction, or by maximizing the narrowest part of the basin, while maintaining the goal position.

We expect this motion planner can be generalized, and potentially applied to a variety of other systems. As future work, we will try similar planning approaches on a dynamic biped walking over rough terrain, and on a forklift operating in a highly constrained environment.

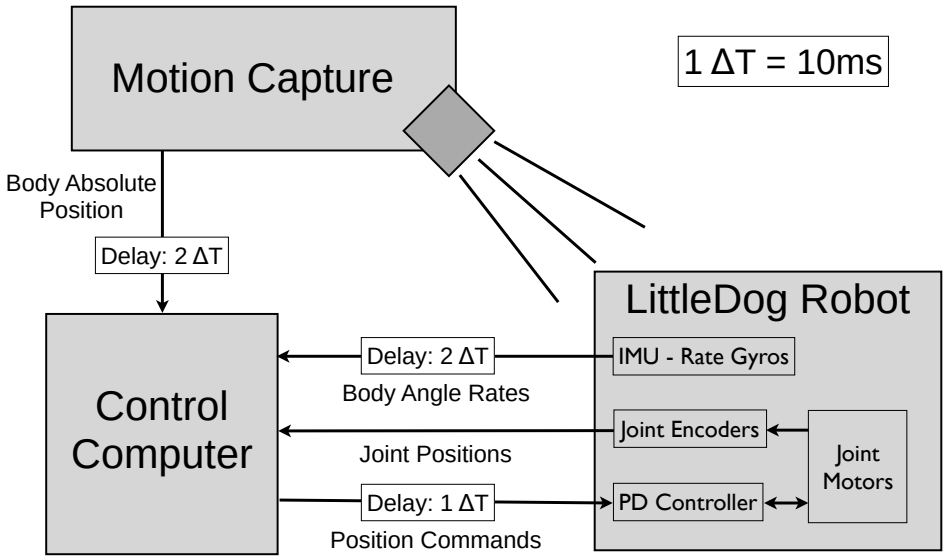


Figure 6-12: The Sensing and Control Environment for LittleDog. Figure by Ian Manchester.

Chapter 7

Conclusions and Future Directions

7.1 Summary

This thesis has drawn on useful ideas from several bodies of research. First, the development of sample-based planning algorithms, such as the RRT, has significantly advanced the state of the art, enabling fast path planning in the presence of challenging geometric constraints in configuration space. Sample-based planning is sensitive to dimension of the action space. However, for many problems of interest, a system's state can be projected into a lower dimensional task space, while preserving the intentions of the planner. This idea has been exploited in task space feedback controllers, which translate commands in task space into commands in the system's action space. I showed that sample-based planning can be made much more efficient, at least for certain types of problems, if a Voronoi bias is implemented in a low dimensional task space rather than in configuration space. For the RRT, this is accomplished by sampling directly within task space, and selecting for expansion the node in the tree, the projection of which is closest to the sample in task space. To maintain completeness while preserving the task space Voronoi bias, the algorithm can occasionally sample in configuration space. The resulting planning algorithm quickly found plans for a 1500-link arm given the task of moving the end-effector from a start position to a goal position while avoiding obstacles collisions. When using task-space guidance on this task, the search efficiency, defined by the number of expansion attempts required by the planner to find a path to the goal, was relatively invariant to the number of links in the arm.

A task space Voronoi bias can be implemented in many ways. The simplest method is to apply a random control action for each expansion step in the RRT, while sampling in task space to choose the node to expand upon. Alternatively, a task space feedback controller may be used to directly guide the expansion of a node towards a sample in task space. The Moore-Penrose pseudoinverse was used for the controller in the 1500-link path planning problem. Despite the dimension, the Jacobian pseudoinverse and matrix multiplications were quick to compute using this approach. Task space control was then explored in application to the LittleDog robot. If at least three feet are in contact with the ground and do not slip or leave the ground, then the robot can move in all directions in task space, as if it is fully actuated.

Derivations for robot center-of-mass and swing-foot task control were presented and used within a hierarchical task controller. The findings showed that redundancy resolution is important when considering the implementation of a task space controller. The choice of redundancy resolution directly affects what task-space actions can be feasibly implemented by the robot. This has direct implications if trying to use task space guidance for motion planning.

Task space control is usually used in fully actuated systems, where a Jacobian can be defined for the task. For underactuated systems, partial feedback linearization can be used to control a subset of the degrees of freedom. By combining these two approaches, a controller was derived that takes advantage of the system' equations of motion to produce a mapping from accelerations in task space, into torques and forces of the controlled degrees of freedom. The derivation allows for hierarchical control including redundancy resolution. The controller was used to demonstrate motion planning to achieve swing-up on a 5-link arm with passive joints. The planner probed actions in task space, which significantly reduced the computational complexity of the search. A pendulum with prism joint turned out to be a good template for this type of problem. This corresponds to controlling the center of mass of the 5-link arm, represented in Polar form. In the standard Template-and-Anchor framework, a template is carefully chosen which encompasses the interesting underactuated dynamics of the system, and the motion planning is done on this simplified system. Instead of this, by using the underactuated task space controller, a template is treated as if it is fully-actuated. Without kinodynamic constraints such as torque limits and collisions, the task space controller would exactly produce the instantaneous commanded motions in task space, assuming that a rank condition is met, which represents strong inertial coupling. When kinematic and additional dynamic constraints of the anchor are incorporated, the fully-actuated template is amenable to search based planning algorithms.

A primary motivation in this thesis was to control the dynamic behavior of the LittleDog robot, to traverse rough terrain with agile, dynamic motions. The robot has very light limbs compared to the body, making the inertial coupling of the system weak. Thus, alternative strategies were considered for motion planning on this system. Reachability guidance was developed as a framework to achieve fast sample-based motion planning given arbitrary differential constraints. The RG-RRT addresses the general problem that the Euclidean distance metric is a very poor approximation of the cost-to-go, which the RRT relies to figure out which nodes in a tree can be most effectively expanded in state space towards a given sample. For an RRT that uses a Euclidean distance metric, the result is that even for low-dimensional problems such as the pendulum, the Voronoi Bias results in repeated expansion attempts of outer nodes in the tree, yet these nodes are often unable to expand outward because of the differential constraints. Reachability-guidance, on the other hand, takes into account regions of state space that are locally (quickly) reachable from the set of already explored nodes of the tree. The Euclidean distance to these reachable regions is then used as a heuristic to determine whether the tree can actually expand towards a given sample. The RG-RRT uses this heuristic while doing rejection sampling in state space. This is accomplished by throwing away samples that are closer to the tree itself than they are

to the reachable regions of the tree. The RG-RRT algorithm greatly accelerates motion planning for dynamic systems, compared to a standard RRT. The algorithm demonstrated an order of magnitude reduction in search speed on the pendulum and nonholonomic car planning problems.

The reachable regions for continuous systems can be difficult to compute. The hull of the reachable set is most important to determine, and this can sometimes be approximated by discretizing the action space. However, doing so is exponential in control dimension, and such an approach would not scale to a system like LittleDog. To address this, reachability guidance can be combined with motion primitives, and task-space biasing. The RG-RRT turns out to work very naturally with motion primitives, because the reachable set can be computed by integrating through the primitive, and Euclidean distance can adequately determine which directions in state space a tree is able grow in. Notably, this holds true even if the motion primitive produces a large jump in state space. A simple motion primitive for LittleDog bounding, corresponding to opening or closing the posture of the robot, was useful for producing smooth bounding behavior. Integrating through this type of motion primitive will either fail because of a collision, or because the robot has too much or too little energy going into the action. Otherwise, if the primitive does not fail, the resulting state will be far away in state space from the corresponding tree node which is being expanded. Because of this, a very coarse discretization of actions in this motion primitive is sufficient to tell the planner what areas of state space, if any, a given tree node is able to grow towards. The resulting motion planner quickly planned feasible bounding trajectories over very rough terrain, despite the 40-dimensional robot state space.

7.2 Future Directions

Several potential future directions are described in the following sections.

7.2.1 Further Analysis of Task Space Mappings

In this thesis I presented several examples of task space mappings. For kinematic path planning problems, the workspace coordinate of a point on the robot is often used as a task space. The end-effector position of an N-link arm is one example of this. The workspace contains information about kinematic obstacles, so a collision in workspace is also a collision in configuration space. This fact can be exploited by planners, and some of the recent related work in path planning has taken advantage of this [Zucker et al., 2008, Diankov et al., 2008]. Work space information is less useful in dynamic applications, for example a task space such as the system center of mass. Many related works in underactuated motion planning including work on Templates and Anchors [Full and Koditschek, 1999], and humanoid robot control (e.g. [Kajita and Tani, 1991]) have used similar task space mappings. These works attempt to characterize the dominating underactuated dynamics of a system by using a low dimensional underactuated template such as a pendulum, or pendulum and spring. The approach in this thesis was different by assuming a fully-actuated task space, and using

efficient search methods to explore feasible commands in this low-dimensional space. A task space that encodes much of the useful behavior in state space is a crucial underlying assumption in this thesis, and this depends on the choice of task space mapping.

To compare mappings, one might characterize the span of feasible actions in state space that can be generated with a given task space. A "better" mapping will be one which has more feasible options that it can present to the planner. Carefully analyzing differences between task spaces, and methods to select an optimal task space is left to future work.

7.2.2 Analysis of Reachability

Reachability guidance offers a conceptually simple method for guiding search on dynamic systems by altering the distance metric to include information about local reachability. How to compute local reachability, and how to compute a distance to this reachable region, can be complicated. In this thesis I focused on approximations of the reachable set. An improved understanding of local reachability might be achieved by analyzing the actual (non-heuristic) reachable sets, for special classes of systems that are amenable to such analysis, or perhaps for very simple dynamic systems, for which the reachable set can be analytically computed. One could then try to characterize the effect of approximating the reachable set, including any resulting bounds on performance of the planner.

Another line of research would be a systematic method for generating the reachable set using approximations of the dynamics which might be easier to work with. The work in [Hofmann and Williams, 2006], is relevant as it abstracts the full dynamics to multiple SISO systems, and solves Linear Programming problems to approximate sets of feasible trajectories, called flow-tubes, which satisfy given constraints imposed by the planner. Other work in progress done by Russ Tedrake, and fellow lab-mates, may also blend well with the RG-RRT ideas. Russ has been developing LQR-Trees [Tedrake, 2009a], which combines RRT type search with LQR feedback control. Elena Glassman has been working on an LQR based heuristic to estimate the cost-to-go as a distance metric [Glassman and Tedrake, 2010]. These approaches use linearized dynamics to approximate dynamic distance between samples and the tree. The linearization is locally valid, which might be exploited when computing local reachability. These types of ideas may be useful as a replacement for the Euclidean distance metric in the RG-RRT, or could be used to approximate the reachable set. This might be done by finding the surface defined by a level set of the LQR-based distance metric. If the cost function is the time-to-go, then the level-set over state space that has a cost equal to the RRT time step, given a fixed start state, would produce an estimate of the reachable set.

7.2.3 RG-RRT for kinematic planning

Another interesting extension of reachability guidance is in application to kinematic path planning (in configuration space, instead of state space). The locally reachable set, without differential constraints, would be a hypersphere centered around the node in the tree. The resulting distance metric would then be Euclidean distance, so the algorithm would behave much like a standard RRT. A sphere can quickly be tested for collisions with

neighboring points. If the sphere is known to be in collision somewhere, the reachable set for the given node can be approximated by discretizing the sphere and checking the resulting set for collisions. The result is similar to the dynamic domain RRT [Yershova et al., 2005, Jaillet et al., 2005], but changes the sampling region more specifically based on geometric information around the tree.

7.2.4 Application and Stabilization

The algorithms described in this thesis were applied in simulation to robot manipulator path planning, and motion planning problems including the underactuated (torque-limited) pendulum, the nonholonomic car, and to a planar model of the LittleDog robot. The algorithms might be applied to a broad range of other systems. For example, humanoid walking using toe-roll is mathematically very similar to the LittleDog robot when it stands on its hind or front legs, and the algorithms in this thesis may be useful for planning agile humanoid walking on rough terrain.

Of course, to execute a plan in hardware, some form of feedback stabilization is typically be required. There are many techniques that can do this, and are beyond the scope of this thesis, but the planning algorithm might need modification to work optimally with a given feedback strategy.

7.2.5 General extensions

The algorithms in this thesis may be generalized to other interesting motion planning problems. This may include, for example:

1. Time-varying problems, involving moving obstacle regions
2. Planning assuming multi-robot coordination
3. Optimal (or quasi-optimal) motion planning that minimizes a cost function through a bias toward low cost plans during exploration
4. Planning for systems with uncertainty. Consider, for example planning in belief space, which can be thought of as a very high dimensional and underactuated problem. Both the TS-RRT and the RG-RRT may be directly applied in this space.

7.2.6 Concluding Remarks

After decades of research, robot technology is beginning its exodus from the factory floor. Production cars are able to park themselves, and some cars brake in response to dangerous situations, or least alert drivers of a potential driving problem. Under the DARPA Learning Locomotion program, several university teams produced excellent results demonstrating walking over rough terrain using the LittleDog robot. Larger robots such as BigDog and a variety of Japanese humanoid robots are consistently improving with time. Motion

planning has been a limiting factor for many applications in robotics. To be effective and computational tractable, a planning algorithm must favor heuristics that quickly solve the planning problem, *most* of the time. Sample-based planners are good at handling geometric complexities. The performance of these planners can be improved by sampling intelligently. By incorporating ideas from task space control, and by using the dynamics of the system to bias the sampling, motion planners are able to solve high-dimensional problems with challenging kinodynamic constraints.

Bibliography

[Altendorfer et al., 2001] Altendorfer, R., Moore, N., Komsuoglu, H., Buehler, M., Jr., H. B., McMordie, D., Saranli, U., Full, R., and Koditschek, D. (2001). RHex: A biologically inspired hexapod runner. *Autonomous Robots*, 11(3):207–213.

[Altendorfer et al., 2000] Altendorfer, R., Saranli, U., Komsoglu, H., Koditschek, D., Brown, H. B., Buehler, M., Moore, N., McMordie, D., and Full, R. (2000). Evidence for spring loaded inverted pendulum running in a hexapod robot. In *Proceedings of the 7th International Symposium on Experimental Robotics (ISER)*.

[Arai et al., 1998] Arai, H., Tanie, K., Shiroma, and N. (1998). Time-scaling control of an underactuated manipulator. *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, 3:2619–2626 vol.3.

[Arai et al., 1993] Arai, H., Tanie, K., Tachi, and S. (1993). Dynamic control of a manipulator with passive joints in operational space. *Robotics and Automation, IEEE Transactions on*, 9(1):85–93.

[Arai and Tachi, 1991] Arai, H. and Tachi, S. (1991). Position control of manipulator with passive joints using dynamic coupling. *IEEE transactions on Robotics and Automation*, 7(4).

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

[Bergerman, 1996] Bergerman, M. (1996). *Dynamics and Control of Underactuated Manipulators*. PhD thesis, Carnegie Mellon University.

[Berkemeier et al., 1999] Berkemeier, M.D., Fearing, and R.S. (1999). Tracking fast inverted trajectories of the underactuated acrobot. *Robotics and Automation, IEEE Transactions on*, 15(4):740–750.

[Bertram et al., 2006] Bertram, D., Kuffner, J., Dillmann, R., Asfour, and T. (2006). An integrated approach to inverse kinematics and path planning for redundant manipulators. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1874–1879.

[Branicky et al., 2008] Branicky, M.S., Knepper, R.A., Kuffner, and J.J. (2008). Path and trajectory diversity: Theory and algorithms. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1359–1364.

[Branicky and Curtiss, 2002] Branicky, M. and Curtiss, M. (2002). Nonlinear and hybrid control via RRTs. *Proc. Intl. Symp. on Mathematical Theory of Networks and Systems*.

[Buchli et al., 2006] Buchli, J., Iida, F., and Ijspeert, A. J. (2006). Finding resonance: Adaptive frequencyoscillators for dynamic legged locomotion. In [Buchli et al., 2006], pages 3903 – 3909.

[Byl et al., 2008] Byl, K., Shkolnik, A., Prentice, S., Roy, N., and Tedrake, R. (2008). Reliable dynamic motions for a stiff quadruped. In *Proceedings of the 11th International Symposium on Experimental Robotics (ISER)*.

[Byl and Tedrake, 2009a] Byl, K. and Tedrake, R. (2009a). Dynamically diverse legged locomotion for rough terrain. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA), video submission*.

[Byl and Tedrake, 2009b] Byl, K. and Tedrake, R. (2009b). Metastable walking machines. *International Journal of Robotics Research*.

[Chan et al., 2008] Chan, N., Kuffner, J., and Zucker, M. (2008). Improved motion planning speed and safety using regions of inevitable collision. In *17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control (RoManSy'08)*.

[Cheng, 2005] Cheng, P. (2005). *Sampling-based motion planning with differential constraints*. PhD thesis. Adviser-Steven M. Lavalle.

[Chestnutt et al., 2005] Chestnutt, J., Lau, M., Cheung, K. M., Kuffner, J., Hodgins, J. K., and Kanade, T. (2005). Footstep planning for the honda asimo humanoid. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[Chevallereau et al., 2003] Chevallereau, C., Abba, G., Aoustin, Y., Plestan, F., Westervelt, E. R., Canudas-De-Wit, C., and Grizzle, J. W. (2003). Rabbit: a testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79.

[Collins et al., 2005] Collins, S. H., Ruina, A., Tedrake, R., and Wisse, M. (2005). Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307:1082–1085.

[Diankov et al., 2008] Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S., and Kuffner, J. (2008). Bispace planning: Concurrent multi-space exploration. In *Proceedings of Robotics: Science and Systems IV*.

[Fantoni et al., 2000] Fantoni, I., Lozano, R., Spong, and M.W. (2000). Energy based control of the pendubot. *Automatic Control, IEEE Transactions on*, 45(4):725–729.

[Fantoni and Lozano, 2002] Fantoni, I. and Lozano, R. (2002). *Non-linear Control for Underactuated Mechanical Systems*. Communications and Control Engineering Series. Springer-Verlag.

[Fraichard and T., 2007] Fraichard and T. (2007). A short paper about motion safety. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1140–1145.

[Fraichard et al., 2003] Fraichard, T., Asama, and H. (2003). Inevitable collision states. a step towards safer robots? In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 388–393 vol.1.

[Frazzoli et al., 2005] Frazzoli, E., Dahleh, M., and Feron, E. (2005). Maneuver-based motion planning fornonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091.

[Frazzoli et al., 2002] Frazzoli, E., Dahleh, M. A., and Feron, E. (2002). Real-Time Motion Planning for Agile Autonomous Vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129.

[Full and Koditschek, 1999] Full, R. and Koditschek, D. (1999). Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332.

[Furuta et al., 1991] Furuta, K., Yamakita, M., and Kobayashi, S. (1991). Swing up control of inverted pendulum. In *Proceedings of the International Conference on Industrial Electronics, Control, and Instrumentation*, volume 3, pages 2193–2198, Kobe, Japan.

[Glassman and Tedrake, 2010] Glassman, E. and Tedrake, R. (2010). Lqr-based heuristics for rapidly exploring state space. *Submitted to IEEE International Conference on Robotics and Automation. ICRA*.

[Goswami et al., 1996] Goswami, A., Espiau, B., and Keramane, A. (1996). Limit cycles and their stability in a passive bipedal gait. pages 246–251. IEEE International Conference on Robotics and Automation (ICRA).

[Green and Kelly, 2007] Green, C. and Kelly, A. (2007). Toward optimal sampling in the space of paths. In *13th International Symposium of Robotics Research*.

[Gu and Xu, 1994] Gu, Y.-L. and Xu, Y. (1994). Under-actuated robot systems: dynamic interaction and adaptive control. *Systems, Man, and Cybernetics, 1994. 'Humans, Information and Technology'., 1994 IEEE International Conference on*, 1:958–963 vol.1.

[Harada et al., 2007] Harada, K., Hattori, S., Hirukawa, H., Morisawa, M., Kajita, S., Yoshida, and E. (2007). Motion planning for walking pattern generation of humanoid. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 4227–4233.

[Hauser et al., 2008] Hauser, Kris, Bretl, Timothy, Latombe, Jean-Claude, Harada, Kensuke, Wilcox, and Brian (2008). Motion Planning for Legged Robots on Varied Terrain. *The International Journal of Robotics Research*, 27(11-12):1325–1349.

[Hauser, 2008] Hauser, K. (2008). *Motion Planning for Legged and Humanoid Robots*. PhD thesis, Stanford.

[Hirai et al., 1998] Hirai, K., Hirose, M., Haikawa, Y., and Takenaka, T. (1998). The development of Honda humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1321–1326.

[Hirose and Ogawa, 2007] Hirose, M. and Ogawa, K. (2007). Honda humanoid robots development. *Philosophical Transactions of the Royal Society*, 365(1850):11–19.

[Hofmann and Williams, 2006] Hofmann, A. G. and Williams, B. C. (2006). Exploiting spatial and temporal flexibility forplan execution of hybrid, under-actuated systems. In *Proceedings of the American Association for Artificial Intelligence (AAAI)*.

[Holleman et al., 2000] Holleman, C., Kavraki, and L.E. (2000). A framework for using the workspace medial axis in prm planners. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on Robotics and Automation*, volume 2, pages 1408–1413.

[Honda Motor Co., 2004] Honda Motor Co. (2004). `http://world.honda.com/ASIMO/`.

[Hyon et al., 2004] Hyon, S.-H., Yokoyama, N., Emura, and T. (2004). Back handspring robot: target dynamics-based control. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 1:248–253 vol.1.

[Jaillet et al., 2005] Jaillet, L., Yershova, A., Valle, L., S.M., Simeon, and T. (2005). Adaptive tuning of the sampling domain for dynamic-domain rrts. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2851–2856.

[Jain et al., 1993] Jain, A., Rodriguez, and G. (1993). An analysis of the kinematics and dynamics of underactuated manipulators. *Robotics and Automation, IEEE Transactions on*, 9(4):411–422.

[Kajita et al., 2003a] Kajita, Kanehiro, S., Kaneko, F., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, K., and H. (2003a). Resolved momentum control: humanoid motion planning based on the linear and angular momentum. *Intelligent Robots and Systems (IROS), Proceedings*.

[Kajita et al., 2003b] Kajita, S., Kanehiro, F., Kaneko, K., Fujiware, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003b). Biped walking pattern generation by using preview control of zero-moment point. In *ICRA IEEE International Conference on Robotics and Automation*, pages 1620–1626. IEEE.

[Kajita and Tani, 1991] Kajita, S. and Tani, K. (1991). Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. volume 2, pages 1405 – 1411. IEEE International Conference on Robotics and Automation (ICRA).

[Kalisiak, 2008] Kalisiak, M. (2008). *Toward More Efficient Motion Panning withDifferential Constraints*. PhD thesis, University of Toronto.

[Kaneko et al., 2004] Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., and Isozumi, T. (2004). Humanoid robot HRP-2. In *ICRA*, pages 1083–1090. IEEE.

[Kavraki et al., 1996] Kavraki, L., Svestka, P., Latombe, J., and Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580.

[Khatib, 1987] Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, 3(1):43–53.

[Khatib et al., 2004] Khatib, O., Sentis, L., Park, J., and Warren, J. (2004). Whole-body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(1):29–43.

[Kim et al., 2005] Kim, J., Esposito, J. M., and Kumar, V. (2005). An rrt-based algorithm for testing and validating multi-robot controllers. In *Robotics: Science and Systems I*. Robotics: Science and Systems.

[Kolter et al., 2008] Kolter, J. Z., Rodgers, M. P., and Ng, A. Y. (2008). A control architecture for quadruped locomotion over rough terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 811–818.

[Kuffner et al., 2001] Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. (2001). Motion planning for humanoid robots under obstacle and dynamic balance constraints. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 1:692–698 vol.1.

[Kuffner et al., 2002] Kuffner, J. J., Kagami, S., Nishiwaki, K., Inaba, M., and Inoue, H. (2002). Dynamically-stable motion planning for humanoid robots. *Auton. Robots*, 12(1):105–118.

[Kuffner et al., 2003] Kuffner, J. J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. (2003). Motion planning for humanoid robots. In Dario, P. and Chatila, R., editors, *ISRR*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 365–374. Springer.

[Kuffner et al., 2000] Kuffner, J. J., Steven, J., and Lavalle, M. (2000). RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001.

[Kurniawati and Hsu, 2006] Kurniawati, H. and Hsu, D. (2006). Workspace-based connectivity oracle: An adaptive sampling strategy for prm planning. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR.* SpringerVerlag.

[LaValle, 1998] LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report 98–11, Iowa State University, Dept. of Computer Science.

[LaValle and Branicky, 2002] LaValle, S. and Branicky, M. (2002). On the relationship between classical grid search and probabilistic roadmaps. In *Proc. Workshop on the Algorithmic Foundations of Robotics*.

[LaValle and Kuffner, 2000] LaValle, S. and Kuffner, J. (2000). Rapidly-exploring random trees: Progress and prospects. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*.

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

[LaValle et al., 2001] LaValle, S. M., Kuffner, J. J., and Jr. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400.

[Liegeois, 1977] Liegeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. Syst. Man Cybern.*, SMC-7(12):868 – 871.

[Lozano-Perez and Tomas, 1981] Lozano-Perez and Tomas (1981). Automatic planning of manipulator transfer movements. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(10):681–698.

[Lozano-Perez et al., 1983] Lozano-Perez, T., Mason, M. T., and Taylor, R. H. (1983). Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):1–34.

[Lynch et al., 2000] Lynch, K. M., Shiroma, N., Arai, H., and Tanie, K. (2000). Collision-free trajectory planning for a 3-dof robot with a passive joint. *International Journal of Robotics Research*, 19(12):1171–1184.

[Manchester et al., 2009] Manchester, I. R., Mettin, U., Iida, F., and Tedrake, R. (2009). Stable dynamic walking over rough terrain:theory and experiment. In *Proceedings of the International Symposium on Robotics Research (ISRR)*.

[McGhee and Frank, 1968] McGhee, R. and Frank, A. (1968). On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3:331–351.

[Morgan et al., 2004] Morgan, S., Branicky, and M.S. (2004). Sampling-based planning for discrete spaces. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2:1938–1945 vol.2.

[Murray and Hauser, 1991] Murray, R. M. and Hauser, J. (1991). A case study in approximate linearization: The acrobot example.

[Nakanishi et al., 2005] Nakanishi, J., Cory, R., Mistry, M., Peters, J., and Schaal, S. (2005). Comparative experimental evaluations of task space control with redundancy resolution. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[Nakanishi et al., 2000] Nakanishi, J., Fukuda, T., and Koditschek, D. (2000). A brachiating robot controller. *Robotics and Automation, IEEE Transactions on*, 16(2):109–123.

[Oriolo and Nakamura, 1991] Oriolo, G. and Nakamura, Y. (1991). Control of mechanical systems with second-order nonholonomicconstraints: underactuated manipulators. In *Proceedings of the IEEE Conference on Decision and Control*, volume 3, pages 2398–2403. IEEE.

[Pongas et al., 2007] Pongas, D., Mistry, M., and Schaal, S. (2007). A robust quadruped walking gait for traversing rough terrain. *ICRA2007*.

[Popovic et al., 2004] Popovic, M. B., Hofmann, A., and Herr, H. (2004). Zero spin angular momentum control: definition and applicability. *Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots*, pages 478–493.

[Poulakakis and Grizzle, 2007] Poulakakis, I. and Grizzle, J. (2007). Monopedal running control: Slip embedding and virtual constraint controllers. *IROS*.

[Pratt and Tedrake, 2005] Pratt, J. E. and Tedrake, R. (2005). Velocity based stability margins for fast bipedal walking. In *Proceedings of the First Ruperto Carola Symposium on Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*, volume 340, pages 299–324.

[Raibert et al., 1986] Raibert, H., M., Chepponis, M., Brown, and B., H. (1986). Running on four legs as though they were one. *IEEE Journal of Robotics and Automation*, 2(2):70–82.

[Raibert et al., 2008] Raibert, M., Blankespoor, K., Nelson, G., Playter, R., and the BigDog Team (2008). Bigdog, the rough-terrain quadruped robot. *Proceedings of the 17th World Congress, The International Federation of Automatic Control*.

[Raibert, 1986] Raibert, M. H. (1986). *Legged Robots That Balance*. The MIT Press.

[Ratliff et al., 2009] Ratliff, N., Zucker, M., Bagnell, J. A. D., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[Rebula et al., 2007] Rebula, J., Neuhaus, P., Bonnlander, B., Johnson, M., and Pratt, J. (2007). A controller for the littledog quadruped walking on rough terrain. *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy*.

[Reif, 1979] Reif, J. H. (1979). Complexity of the movers problem and generalizations. *In Proceedings IEEE Symposium on Foundations of Computer Science*, page 421427.

[Reyhanoglu et al., 1999] Reyhanoglu, M., van der Schaft, A., Mcclamroch, N.H., Kolmanovsky, and I. (1999). Dynamics and control of a class of underactuated mechanical systems. *Automatic Control, IEEE Transactions on*, 44(9):1663–1671.

[Rickert et al., 2008] Rickert, M., Brock, O., Knoll, and A. (2008). Balancing exploration and exploitation in motion planning. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2812–2817.

[Sakagami et al., 2002] Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., and Fujimura, N. H. K. (2002). The intelligent ASIMO: system overview and integration. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 2478 – 2483. IEEE.

[Saranli and Koditschek, 2003] Saranli, U. and Koditschek, D. (2003). Template based control of hexapedal running. In *Proceedings of the IEEE International Conference On Robotics and Automation*, volume 1, pages 1374–1379.

[Sentis and Khatib, 2005] Sentis, L. and Khatib, O. (2005). Control of free-floating humanoid robots through task prioritization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Shin and Lee, 1997] Shin, J.-H. and Lee, J.-J. (1997). Dynamic control of underactuated manipulators with free-swinging passive joints in cartesian space. *ICRA*.

[Shkolnik et al., 2007] Shkolnik, A., Byl, K., Rohanimanesh, K., Roy, N., Tedrake, and R. (2007). Walking on rough terrain with a position controlled quadruped robot. In *International Workshop on Legged Locomotion for Young Researchers*, Cambridge, MA.

[Shkolnik et al., 2010] Shkolnik, A., Levashov, M., Manchester, I. R., and Tedrake, R. (2010). Bounding on rough terrainwith the littledog robot. *Under review*.

[Shkolnik and Tedrake, 2007] Shkolnik, A. and Tedrake, R. (2007). Inverse kinematics for a point-foot quadruped robot with dynamic redundancy resolution. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*.

[Shkolnik and Tedrake, 2008] Shkolnik, A. and Tedrake, R. (2008). High-dimensional underactuated motion planning via task space control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ.

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

[Shkolnik et al., 2009] Shkolnik, A., Walter, M., and Tedrake, R. (2009). Reachability-guided sampling for planning under differential constraints. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA)*. IEEE/RAS.

[Slotine and Li, 1990] Slotine, J.-J. E. and Li, W. (1990). *Applied Nonlinear Control*. Prentice Hall.

[Spong et al., 1995] Spong, M.W., Block, and D.J. (1995). The pendubot: a mechatronic system for control research and education. *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, 1:555–556 vol.1.

[Spong, 1995] Spong, M. (1995). The swingup control problem for the acrobot. *IEEE Control Systems Magazine*, 15(1):49–55.

[Spong, 1994a] Spong, M. W. (1994a). Partial feedback linearization of underactuated mechanical systems. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 1, pages 314–321.

[Spong, 1994b] Spong, M. W. (1994b). Swing up control of the acrobot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2356–2361.

[Stilman, 2007] Stilman, M. (2007). Task constrained motion planning in robot joint space. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3074–3081. IEEE.

[Stojic et al., 2000] Stojic, R., Chevallereau, and C. (2000). On the stability of biped with point foot-ground contact. *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 4:3340–3345 vol.4.

[Sugihara, 2004] Sugihara, T. (2004). *Mobility Enhancement Control of Humanoid Robot based on Reaction Force Manipulation via Whole Body Motion*. PhD thesis, University of Tokyo.

[Tang et al., 2006] Tang, X., Thomas, S., and Amato, N. M. (2006). Efficient planning of spatially constrained robot using reachable distances. Technical Report TR07-001, Texas A&M University.

[Tedrake, 2009a] Tedrake, R. (2009a). LQR-Trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems (RSS)*, page 8.

[Tedrake, 2009b] Tedrake, R. (2009b). *Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines: Course Notes for MIT 6.832*. Working draft edition.

[Tedrake et al., 2006] Tedrake, R., Byl, K., and Pratt, J. E. (2006). Probabilistic stability in legged systems: Metastability and the mean first passage time (FPT) stability margin. *In progress*.

[Weghe et al., 2007] Weghe, M. V., Ferguson, D., and Srinivasa, S. (2007). Randomized path planning for redundant manipulators without inverse kinematics. In *IEEE-RAS International Conference on Humanoid Robots*.

[Westervelt et al., 2007] Westervelt, E. R., Grizzle, J. W., Chevallereau, C., Choi, J. H., and Morris, B. (2007). *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, Boca Raton, FL.

[Westervelt et al., 2002] Westervelt, E. R., Grizzle, J. W., and Koditschek, D. E. (2002). Zero dynamics of underactuated planar biped walkers. *IFAC-2002, Barcelona, Spain*, pages 1–6.

[Westervelt et al., 2003] Westervelt, E. R., Grizzle, J. W., and Koditschek, D. E. (2003). Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56.

[Whitney and D.E., 1969] Whitney and D.E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2):47–53.

[Yamakita et al., 2003] Yamakita, M., Kishikawa, M., Sadahiro, and T. (2003). Motion control for robust landing of acrobat robot (smb). *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 2:1141–1146 vol.2.

[Yao et al., 2005] Yao, Z., Gupta, and K. (2005). Path planning with general end-effector constraints: using task space to guide configuration space search. *International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ*, pages 1875–1880.

[Yershova et al., 2005] Yershova, A., Jaillet, L., Simeon, T., LaValle, and S.M. (2005). Dynamic-domain rrts: Efficient exploration by controlling the sampling domain. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3856–3861.

[Yershova, 2008] Yershova, A. (2008). *Sampling and Searching Methods for Practical Motion Planning Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign.

[Zucker, 2009] Zucker, M. (2009). *PhD Thesis Proposal: A Data-Driven Approach to High Level Planning. Available at http://www.cs.cmu.edu/ mzucker/mz-thesis-proposal.pdf.* PhD thesis, Carnegie Mellon University.

[Zucker et al., 2008] Zucker, M., Kuffner, J., and Bagnell, J. A. D. (2008). Adaptive workspace biasing for sampling based planners. In *Proc. IEEE Int'l Conf. on Robotics and Automation*.